Towards competent information acquisition interactions between an expert system and its user

E T Keravnou, J Washbrook* and F Dams*

Expert systems are consultative, highly interactive systems, and hence the quality of interaction between system and user is important for the acceptability of the system. Acquisition of data about the problem in hand is a central feature of the interaction between system and user, and it makes a major contribution to the user's perception of the system. It is therefore crucial that the acquisition of this data (both as individual data items and as a sequence of data inputs) is perceived by the user as competent.

This paper identifies central, domain-independent, design goals for the information-acquisition interactions between an expert system and its user, including mixed-initiative interaction, flexibility in user input, and system competence in querying the user. The paper discusses the realisation of these goals in a diagnostic expert system, Skeletal Dysplasias Diagnostician, through an explicit data model which allows for the representation of data with temporal and spatial (locality) aspects and the decide-status function which operates on the data model.

Keywords: information-acquisition interactions, intelligent data handling, medical diagnostic expert systems, competent expert systems, skeletal dysplasias, background knowledge

Expert systems, especially those that dispense advice in critical domains such as medicine, engineering and finance, must be perceived by their users as competent if they are to be accepted. Recommending solutions for real problems which are subsequently proven in real life to be correct is of course indisputable evidence of a system's competence. However, for a system to progress from the research laboratory into a real-life environment, it must not only be able to solve test problems correctly but also to interact intelligently with the user.

It is through the dialogue with the system that the user forms important first impressions. Thus the interaction between the system and its user is of prime importance if the system is to inspire confidence in potential users whilst still in the laboratory, and to retain their confidence once it has progressed to a real-life setting. The principal spheres of interaction are information acquisition and explanation, both of which must be carried out in a way which is acceptable to, and tailored to the needs of, the particular user[1]. Acquisition of information about the problem in hand may be through the system asking questions, or the user volunteering information. At the beginning of a consultation, initial information is acquired through a predetermined sequence of general questions asked by the system and items of data volunteered by the user. The initial information acquisition process is thus mixed-initiative, in that both the system and the user may initiate a particular interaction.

It is possible for an expert system to require that all problem data be entered at the beginning of the consultation, without the necessity (or possibility) of subsequent information acquisition. This, however, is based on a simplified and restrictive model of the consultation process. A more sophisticated model would allow for system-initiated information acquisition at subsequent stages in the consultation. However it is arguable (see below) that information acquisition should continue throughout the consultation, and should be potentially mixed-initiative (i.e. both the user and the system should be able to initiate information input) at all stages. In addition, to be perceived as competent, the system should take into account all relevant information previously acquired before asking questions. This may require the making of substantial chains of inferences.

An expert system which accurately models the reason-

Department of Computer Science, University of Cyprus, Kallipoleos 75, PO Box 537, Nicosia, Cyprus
*Department of Computer Science, University College London, Gower Street, London WC1E 6BT, UK
Paper received 7 January 1992. Revised paper received 7 September 1992. Accepted 8 October 1992

0950-7051/93/030141-16

ing and knowledge of domain experts will also provide a model of their information-acquisition strategies. This can be used as the basis for system information-acquisition interactions. Such interactions are dynamically constructed, being driven by the derivation of the solution[2]. They can be analysed at two levels. At a high level the sequence is structured in terms of the hierarchy of foreground problem-solving strategies that have been instantiated. At a low level each subgroup of interactions corresponding to a terminal problem-solving strategy can be analysed in terms of background auxiliary strategies. The foreground strategies represent domain-expert problem-solving processes, whilst the background strategies provide indirect but indispensable support to these processes by organising the problem data and subsequent system-initiated information acquisition[3]. Foreground problem-solving strategies may be domain-specific, while background data-handling strategies, which often represent common-sense knowledge and reasoning, are usually domain-independent.

Other aspects of an information-acquisition interaction model, such as those related to the specifics of conducting a particular task, or technical aspects related to the design of user interfaces, are outside the scope of this paper.

Although the work presented here is in the context of a diagnostic expert system, competence in the information-acquisition interactions is relevant to every consultative expert system, diagnostic or other. This paper discusses information-acquisition interactions in the context of the diagnostic system SDD (Skeletal Dysplasias Diagnostician)[4]. The design goals identified constitute essential aspects of the overall information-acquisition interaction model of this system[5].

The first section identifies central, domain-independent, design goals for the information-acquisition interactions between an expert system and its user. These design goals have been realised in the medical diagnostic system, Skeletal Dysplasias Diagnostician[4], through background strategies based on an explicit data model and the decide-status function which operates on the data model.

The second, third and fourth sections give a detailed description of the decide-status operation; some of this material has been adapted from Reference 6.

The fifth section discusses how the data model and decide-status function are used to achieve the required design goals for the information-acquisition interactions between an expert system and its user.

INFORMATION-ACQUISITION INTERACTIONS: DESIGN GOALS

Important design goals for the information-acquisition interactions between an expert system and its user are listed below. Achieving these goals is necessary if the system is to be deemed competent.

- *Mixed-initiative interaction*
- *Flexibility in user-volunteered information:*
  - The user must be allowed to use the entire domain vocabulary.
  - The user must be able to revoke information.
  - The system must detect a conflict in the user-volunteered information as soon as it occurs and resolve it with the user.
- *System competence in querying the user:*
  - It must be evident to the user that the system has taken notice of the information volunteered by the user.
  - The system questions must be focused and methodical.
  - There should be non-redundancy in system questions.
  - The system must detect and eliminate any redundancy in the user information.

Mixed-initiative interaction

The information-acquisition interaction between the system and the user must be allowed to be mixed-initiative where the user wishes it to be so. The user must be able to volunteer information initially and be given the opportunity to volunteer further information at subsequent stages. Similarly, the system must be able to request further information by querying the user if the information available at any stage is not sufficient for the system to proceed to the next stage, or to make firm recommendations. Where necessary, appropriate guidance should be given to the user both when volunteering information and responding to a system question[7].

The system should of course be able to perform a consultation even if the user does not wish to volunteer information. This can be initiated through a predetermined sequence of questions.

The explanation model of an expert system is separate from the information-acquisition interaction model[8]. As mentioned above, the sequence of questions raised by the system is intrinsically related to the system's reasoning[2]. Traditionally, the central role of the explanation model is to reveal this reasoning[9]. However, the explanation model has a subsidiary role in relation to information-acquisition interactions. This role concerns individual items of information rather than the system reasoning processes. The user needs to be able to ask, not only why the system is asking a particular question (i.e. how does it relate to the reasoning process), but also what the particular question means. Also, the system needs to be able to recognise when information input by the user is either unclear or meaningless in the context of the consultation, and must have the ability to query the user, asking for clarification.

Flexibility in user-volunteered information

For the user to take full advantage of the facility to volunteer information, the system must allow the user flexibility in the means by which information is entered.

Flexible vocabulary

A menu-driven interface facilitates the entry of user information and ensures that every piece of information is meaningful to the system. However the user may also wish to volunteer information 'freely', without having to go through a menu, especially if the user has domain expertise and does not need guidance in deciding what information to enter and how to word it. This facility is also useful in cases where the domain vocabulary requires a complex network of menus for covering all the possible expressions.

The flexibility to enter information directly does not necessarily imply a completely unconstrained natural language interface (which is anyway beyond the capabilities of the current technology), but rather the use of the entire domain vocabulary (which will undoubtedly include redundant terminology) albeit within specific syntactical constraints which should be simple and convenient.

The system should be capable of dealing with redundant terminology by correlating different expressions which have the same meaning. Internally the system will probably use a non-redundant set of terms, into which it will translate user terms. It is important, though, that when the system interacts with the user it either uses the user's terms, or makes it clear that it has translated the user's terms.

Conflict detection and resolution and retraction

Another aspect of flexibility in user interaction is for the system to be able to detect and resolve conflicting information input, and for the user to be able to retract previously volunteered information.

Conflicting information input may occur if the user information summarises lower-level data (e.g. sensor readings, images, direct observations etc.). The inherent uncertainty of this lower-level data can lead to misinterpretations.

The user may wish to enter possibly conflicting data, provided that a facility to retract previously volunteered information is available. The system should be able to detect and reveal an inconsistency as soon as it occurs and should attempt to resolve it by consulting the user. This should give the user more confidence in the system; the user who is aware of the inconsistency would be surprised if the system did not detect it, whilst the unaware user would be favourably impressed. In both cases it would be evident to the user that the system is taking notice and 'making sense' of the information entered.

The user should therefore be able to volunteer information either directly, in the form which he or she is used to in reporting findings, or through menus, depending on inclination and convenience, and should also be given the option to retract information subsequently.

System competence in querying user

Being unable to use the entire domain vocabulary or revoke information could reduce the usability of the system without necessarily affecting the confidence of the user in the system's judgement. However, that confidence will be seriously undermined if the system's questioning is perceived as incompetent or unintelligent. For a system to be deemed as competent in its questioning, the following requirements must be satisfied.

Use of previously given information

The system must be able to show the user that information entered has been accepted and processed by the system, and is being used both in the diagnostic process and as a basis for any subsequent information acquisition.

For this to happen, the generation of (partial) solutions (in a diagnostic domain this would be the generation of hypotheses) must be driven by input information; thus subsequent questioning based on the currently entertained (partial) solutions would relate back to what the system had already been told, making the questioning focused and methodical. (The details of hypothesis generation and exploration refer to the foreground problem-solving strategies rather than the background data-handling strategies and hence are outside the scope of this paper.)

One significant way of showing the user that the system is utilising input information is for the system to ask questions which naturally follow on from the data volunteered, or to reveal inconsistencies in the user-volunteered information. Furthermore, if some information volunteered by the user needs clarification or triggers other questions, these clarifying or prompting questions should usually be raised by the system immediately after receiving this information; this constitutes intelligent behaviour.

Non-redundant questions

It is important that questions raised by the system should be *non-redundant*, in that their answers should not be deducible from information already entered. In response to a redundant question the user is justified in saying 'but I told you so already', or 'I've told you I don't know this', or 'I couldn't possibly know this at the present point in time'. If many redundant questions are asked, the user will get justifiably irritated. To ensure non-redundancy, the system should be able to make intelligent inferences on the given information; these should include the correlation of statements with equivalent meanings.

Another aspect of intelligent inferencing on input information is the ability to recognise whether a particular item of information would be unknown or not, based on information that the user has already specified as unknown, and, in some domains, based on the temporal context defined by the particular problem. This is particularly important in most diagnostic domains. In the domain of skeletal dysplasias diagnosis, for instance, a feature may be unknown because the relevant radiographs are unavailable, or because the patient is too young for certain features to be observable.

The system should also show that it understands dependencies between items of input information by removing redundant information in the light of new input. When new information volunteered by the user subsumes previously given information, possibly by being more specific, the system should make it evident to

the user that it is aware of the redundancy and has eliminated it; this should inspire confidence and respect. Since the information volunteered by the user is likely to be communicated back to the user by the system in different contexts (for example in run-time explanations, or in summing up final recommendations), there should be ample scope for the system to demonstrate to the user that it has this capability.

Summary

The main requirements for competent information acquisition by an expert system are mixed-initiative interactions, flexibility in the entry of user-volunteered information, and competence on the part of the system in querying the user. These requirements can be achieved if the system has an explicit and flexible data model which is used by an intelligent reasoner to draw inferences from an actual set of data. We have implemented such a data model and a reasoner, the decide-status function.

DECIDE-STATUS FUNCTION: AN INTELLIGENT DATA REASONER

The *decide-status* function is an intelligent reasoner about problem-specific data or *findings*. It provides the platform for achieving the seemingly diverse goals discussed above. Earlier work on an atemporal decide-status function has been reported in Reference 10.

Decide-status operation

The decide-status operation is first illustrated through an example from the domain of skeletal dysplasias. Consider the following problem definition.

Domain-Model:

synonyms((metaphyses flared) (long-bones dumbbell-shaped))
(carpal-centres poor-ossification) ⇒ (carpal-centres small)
(carpal-centres small) ⇒ (epiphyses small)

Problem findings:

(long-bones dumbbell-shaped from-birth)
(carpal-centres poor-ossification from-birth)
(metaphyses irregular from-birth)

Query finding:

(and (metaphyses flared irregular [t_a t_b])
(epiphyses small [t_a t_b]))

The domain model consists of two synonymous findings and two implications. Each of the problem findings has the qualitative-temporal aspect 'from-birth' which translates to the time interval [0 now]. The *query finding* is a compound finding consisting of the two simple findings

- metaphyses being both flared and irregular during the time-interval [t_a t_b],
- epiphyses being small during the same time interval.

Metaphyses and epiphyses constitute the subjects of the respective findings (finding subjects).

The decide-status function reasons backwards from the query finding. A goal tree, having the query finding at its root is being (implicitly) constructed (see *Figure 1*). The nodes of the goal-tree name findings and the branches strategy applications. The leaf node of a 'successful' (i.e. not pruned) branch names a *problem finding*.

The decide-status operation is simply described as deciding the truth status of a finding as *true, false,* or *unknown*, given a group of findings which collectively hold in some context. A firm answer (true or false) returned by the decide-status function for a finding queried against the user observations can be directly corroborated, e.g. through observation. The decide-status function operates on an explicit *data model.*

Formalisation of data model

The data model has a *representational aspect*, i.e. it provides a formal language for representing findings (data), and an *inferential aspect*, i.e. it provides a formal set of relations between findings which form the channels through which intelligent inferences about an actual set of findings can be drawn. The representational and inferential aspects of the decide-status data model are as follows.

Representational aspect: grammar for expressing findings:

⟨finding⟩::=⟨compound-finding⟩|⟨simple-finding⟩|⟨atomic-finding⟩
⟨compound-finding⟩::=(and ⟨findings⟩)|(or ⟨findings⟩)
⟨findings⟩::= <finding⟩⟨findings⟩|⟨finding⟩
⟨simple-finding⟩::=(⟨finding-subject⟩⟨attribute-values⟩⟨time-interval⟩)
⟨attribute-values⟩::=nil|⟨attribute-value⟩⟨attribute-values⟩
⟨finding-subject⟩::=**symbol**
⟨attribute-value⟩::=⟨locality-value⟩|⟨non-locality-value⟩
⟨locality-value⟩::=**symbol**
⟨non-locality-value⟩::=**symbol**
⟨time-interval⟩::=nil|⟨closed-interval⟩|⟨open-interval⟩|⟨open-from-left⟩|⟨open-from-right⟩
⟨closed-interval⟩::=[⟨base⟩⟨limit⟩]
⟨open-interval⟩::=(⟨base⟩⟨limit⟩)
⟨open-from-left⟩::=(⟨base⟩⟨limit⟩)
⟨open-from-right⟩::=[⟨base⟩⟨limit⟩]
⟨base⟩::=**integer**
⟨limit⟩::=**integer**
⟨atomic-finding⟩::=(⟨finding-subject⟩⟨simp-att-vals⟩⟨time-interval⟩)
⟨simp-att-vals⟩::=nil|⟨locality-value⟩⟨non-locality-value⟩|⟨locality-value⟩|⟨non-locality-value⟩
⟨atemporal-finding⟩ is a ⟨finding⟩ without temporal-aspect(s)

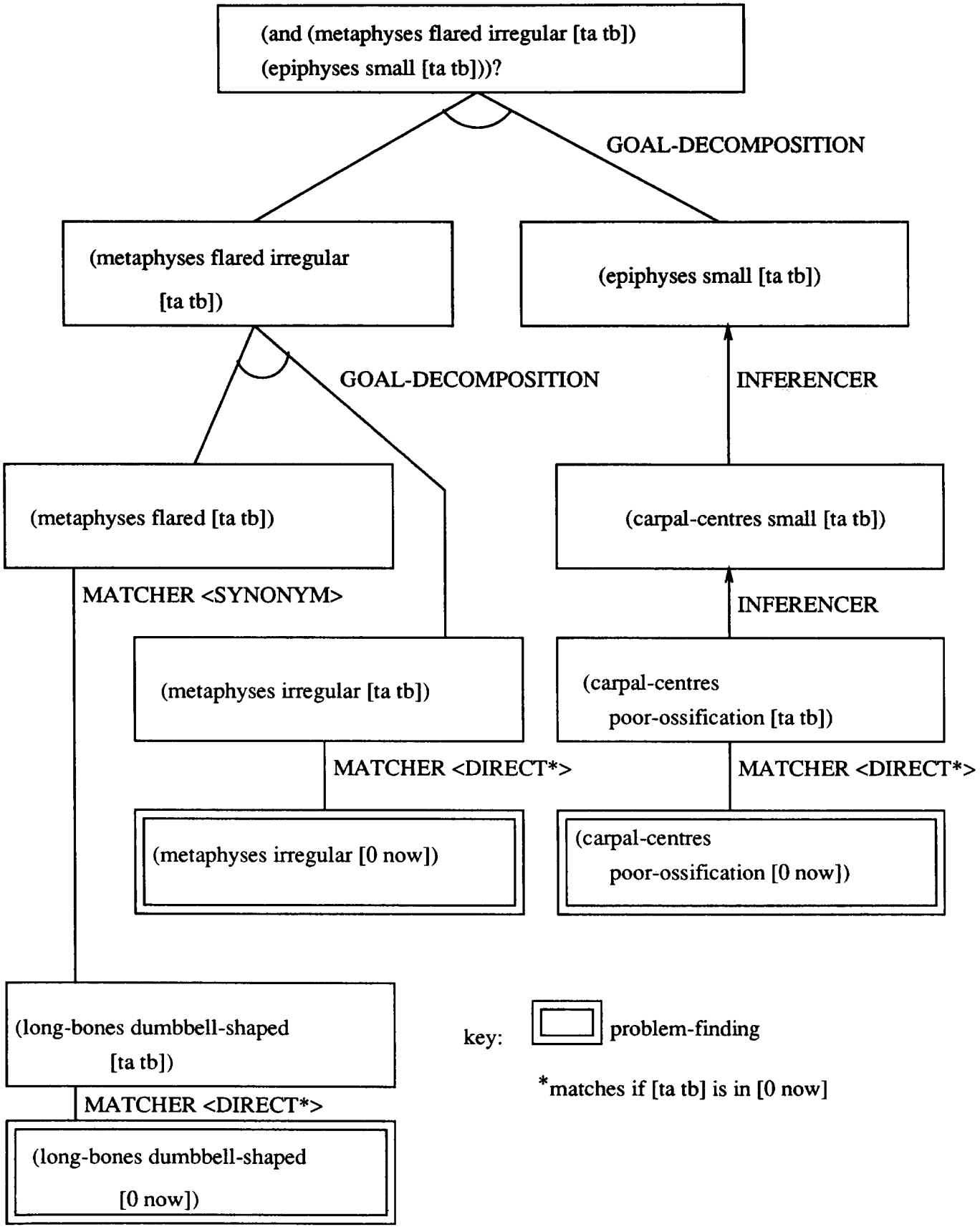

Figure 1 Decide-Status

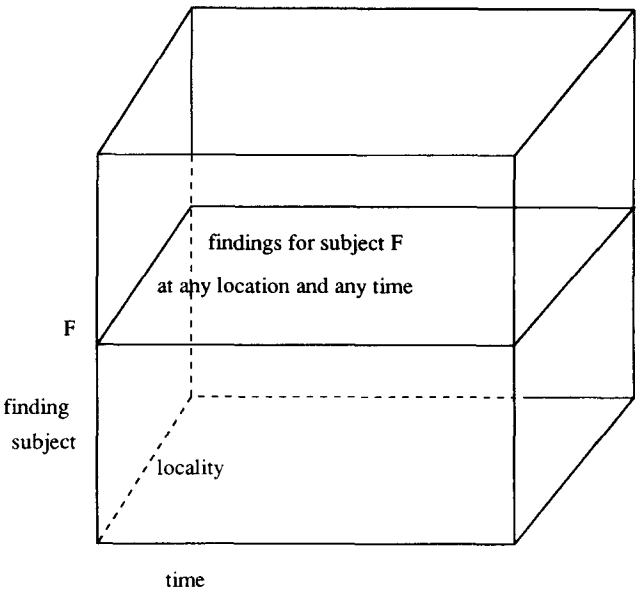

Figure 2 Space of findings is three-dimensional space of finding subject, time and locality

Inferential aspect: model entity relationships:

is-a(⟨finding-subject⟩⟨finding-subject⟩)
part-of(⟨finding-subject⟩⟨finding-subject⟩)
syn-subjects(⟨finding-subject⟩⟨finding-subject⟩); symmetric relation
negative-value(⟨finding-subject⟩⟨attribute-value⟩)
status-value(⟨finding-subject⟩⟨attribute-value⟩)
normal-subject(⟨finding-subject⟩)
abnormal-subject(⟨finding-subject⟩)
negative-finding(⟨finding⟩)
positive-finding(⟨finding⟩)
most-general-positive-finding(⟨finding⟩)
status-finding(⟨finding⟩)
prompting-query(⟨atemporal-finding⟩⟨atemporal-finding⟩)
clarifying-query(⟨atemporal-finding⟩⟨atemporal-finding⟩)
synonyms-value(⟨attribute-value⟩⟨attribute-value⟩); symmetric relation
opposites-value(⟨attribute-value⟩⟨attribute-values⟩); symmetric relation
synonyms(⟨atemporal-finding⟩⟨atemporal-finding⟩); symmetric relation
implies(⟨atemporal-finding⟩⟨atemporal-finding⟩) alternatively expressed as: ⟨atemporal-finding⟩ ⇒ ⟨atemporal-finding⟩
when-to-ask(⟨atemporal-finding⟩⟨earliest-time⟩)
primary-trigger(⟨finding⟩⟨hypothesis⟩)
secondary-trigger(⟨finding⟩⟨context-hypothesis⟩ ⟨alternative-hypothesis⟩)

Decide-status operation

The entire space of findings is partitioned along three dimensions, finding subject, time and locality, as shown in *Figure 2*. The decide-status operation is formally described as follows.

Let Q be a query for some problem, expressed as holds(F)?, where F is a finding. The decide-status function Decide-Status determines whether Q is decidable from the theory P constituting the problem, i.e. whether

$$P \vdash Q$$

or

$$P \vdash \sim Q$$

If Q is undecidable the reply *unknown* is returned. The problem theory P = *domain-model* ∪ *problem-findings*, where the domain model is the instantiation of the inferential aspect of the data model for the particular domain, and the problem findings are the instantiations of the representational aspect of the data model, i.e. assertions about which findings hold for the specific problem. The domain model therefore defines the finding subjects, attribute values and temporal aspects in the domain, as well as the relationships between these entities (e.g. status and negative values for finding subjects, synonymous finding subjects, synonymous findings, opposite attribute values). The meaning of these relationships (given above) will be clarified when the decide-status operation is detailed.

Decide-Status determines whether Q is true, false or unknown from the set of problem findings given the domain model, by attempting to construct a proof for either Q or $\sim Q$. If this is not possible, Q is unknown. The proof is constructed in a 'backwards-reasoning' or 'goal-driven' fashion where Q forms the goal, yielding a focused search strategy. Q ($\sim Q$) follows from a set of problem findings if it is directly subsumed or implied by a subset of the problem findings. The domain model is used to explicate relationships between the query finding F and the problem findings. Starting with the query Q at the root of the proof tree, the goal-decomposition strategy is applied until the goal is decomposed into simple subgoals, corresponding to simple queries (i.e. F is decomposed to simple findings). The domain model is used to generate subgoals from a given goal when that goal cannot be further decomposed. Subgoals generated from the domain model may be compound (e.g. the antecedents of implications may be compound findings) which then need to be decomposed.

Subgoals are themselves query findings. A tree branch terminates successfully if the query finding constituting its leaf node is directly determined from the problem findings, i.e. it directly matches, or conflicts, with problem findings. On the other hand, a tree branch is pruned if the simple, undetermined query finding corresponding to its leaf node cannot be mapped further through the domain model. A simple goal is determined if either itself or at least one of the subgoals (which the goal can be mapped onto using the domain model) is directly determined from the problem findings, i.e. the subgoal leads to a successful tree branch. For a conjunctive compound finding to be determined as true all its component findings must be determined as true. For it to be determined as unknown at least one of its component findings must be undetermined whilst the remaining component findings are determined as true. Lastly, determining one of the component findings as false suffices to

determine the compound finding as false. A disjunctive compound finding is determined as true, false or unknown if at least one of its component findings is determined as true, all of its component findings are false, or at least one of its component findings is unknown whilst the remaining are false, respectively.

Thus if the current subgoal of some simple goal is undetermined, the domain model is used to generate an alternative subgoal for that goal. At the top level there are two generic strategies corresponding to two alternative derivation methods: a *matcher* and an *inferencer*. The matcher and inferencer strategies are described in detail in the third and fourth sections, respectively.

MATCHER

The matcher strategy is invoked for a particular finding F queried relative to a group of findings G, if F has relevant findings in G or F has relevant synonyms in the domain model.

Relevant findings

The *relevant findings* of a finding F from a group G of findings is the subset of G defined as follows:

relevant-findings(F,G) =
$\{P \in G \mid$ *subject*(P) = *subject*(F) and
overlapping(*time-interval*(P), *time-interval*(F)) and
overlapping(*locality*(P), *locality*(F))$\}$

i.e. those elements of G which have the same subject as F and whose *time-interval* and *locality* attributes overlap with the corresponding *time-interval* and *locality* attribute of F (if the time or locality are not given explicitly, the defaults 'now' and 'some-relevant-locality' are assumed, respectively). Finding subjects are discrete, time intervals specify a continuous period of time, and locality attributes can specify a collection of discrete, (physically) disjoint localities, or a continuous space of adjoining localities. A finding, therefore, defines a subspace on the entire space of findings given in *Figure 2*. The subspace could be a single point if the finding's time interval refers to a time point and its locality to a single discrete locality. The relevant findings of a finding, from a group G of findings, are those findings whose subspaces overlap with (e.g. are contained in) the subspace of the given finding (see *Figure 3*).

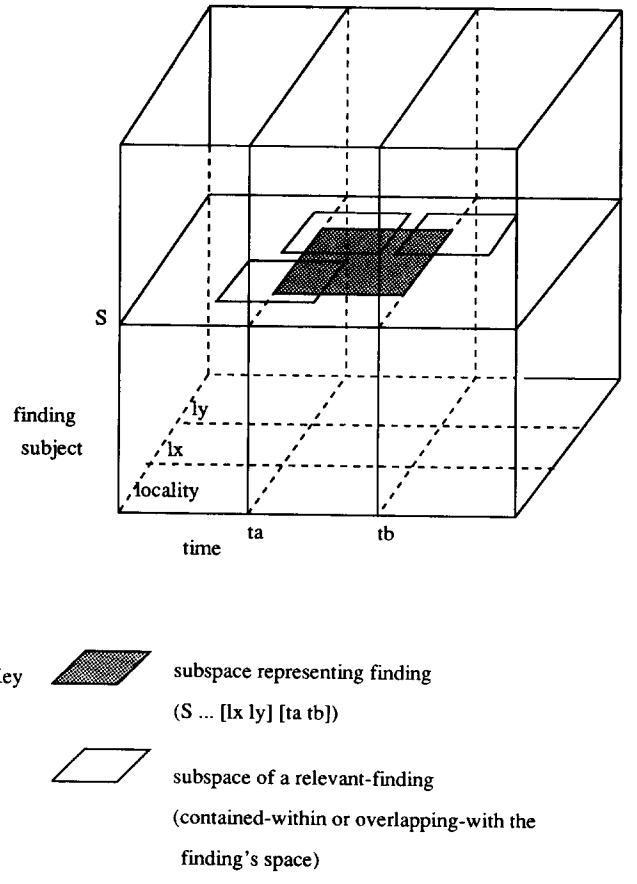

Figure 3 Finding and its relevant findings as subspaces on findings space

Locality attribute

Consider the following findings from the domain of skeletal dysplasias, expressed in natural language: 'all the vertebrae [of the spine] are irregular'*, 'some of the vertebrae are flat, i.e. they have the condition platyspondyly', 'the femoral capital epiphyses are small', and 'the middle of the face is flat'. In each of these findings an abnormality is described which occurs in a specific region (locality). The locality specifies a generic region, or a component, or a subtype, of the subject. The above findings can be expressed in the format ⟨*subject*⟩ ⟨*locality*⟩ ⟨*attribute-values*⟩ as follows: 'spine all-vertebrae irregular', 'spine some-vertebrae flat', 'epiphyses femoral-capital small', and 'face middle flat'.

In addition, generic terms can describe localities relevant to many subjects. The above examples use 'all-vertebrae' and 'some-vertebrae'. It is also possible to talk about 'all-long-bones', 'all-epiphyses', 'all-digits' etc., and similarly about 'some-long-bones', 'some-epiphyses' and 'some-digits'. All these can be expressed in terms of the two generic locality descriptions: *localised* and *generalised*. These two localities are opposing (conflicting) localities. Other generic locality descriptions could be 'upper', 'middle' and 'lower', 'left' and 'right', 'laterally' and 'medially', 'anteriorly' and 'posteriorly' etc. These are not opposing localities since they specify disjoint regions.

Since the data model allows for the explicit representation of taxonomic relationships between finding subjects, the above findings can therefore be rewritten as 'vertebrae irregular generalised', 'vertebrae flat localised' (or 'platyspondyly localised'), 'femoral-capital-epiphyses small' and 'face middle flat'. Locality attributes are special attributes as illustrated by the following example. The finding 'vertebrae flat irregular localised from-birth' can be converted into the two atomic findings 'vertebrae

*A finding will be denoted in quotes, e.g. 'platyspondyly at birth', in natural language or when referring to the user interface, and as a parenthesised expression, e.g. (platyspondyly (atmonths (0 0))) when referring to the internal representation.

flat localised from-birth' and 'vertebrae irregular localised from-birth', which say that some vertebrae are irregular from birth and some are flat from birth*. The scope of a locality attribute is a finding subject if the finding subject describes an abnormality (e.g. 'platyspondyly'); otherwise, in the case of 'normal' (i.e. anatomical component) subjects like vertebrae, its scope is a ⟨*finding-subject attribute-value*⟩ pair, where the attribute value is a non-locality value. This is because it is meaningful to say 'platyspondyly localised' but it is not meaningful to say 'vertebrae localised'.

Suppose that the problem findings include 'vertebrae irregular localised' and that the query finding is 'vertebrae irregular generalised'. The query finding is established as false because the localities *generalised* and *localised* are opposing; in this respect, these two generic localities are treated as any other attribute value from the perspective of the matcher. Suppose, further, that the problem findings include 'long-bones short' and that the query finding is 'ulnae short' (ulnae being two of the long bones). The query finding is established to be true as follows:

'long-bones short' ≡ 'long-bones all-long-bones short'
'ulnae short' ≡ 'long-bones ulnae short'
{ulna} ⊂ all-long-bones

If the query finding were 'long-bones short' and the problem finding were 'ulnae short' the query finding would be unknown.

The relevant findings of a finding could therefore include findings whose subjects are taxonomically related to the subject of the finding. Taxonomic relationships between finding subjects, however, are better manipulated by the inferencer (see the fourth section).

Time intervals

Findings have temporal aspects which default to 'now'. The temporal reasoner which supports the decide-status function is discussed in Reference 11. The temporal aspects of findings are expressed qualitatively, e.g. 'at-birth', 'from-birth', 'under-yrs', '=+yrs', 'infancy', 'mid-childhood' etc. in the SDD system. These qualitative expressions are translated internally into time intervals. Because of the uncertainty inherent in some of the qualitative temporal aspects, e.g. 'from about a certain time' or 'up to about a certain time', the time intervals can be 'open'. Since an open time interval essentially delineates a smaller closed interval, for the purposes of this paper it is assumed that there is no time uncertainty and hence time intervals are closed intervals. Recently, Console *et al.*[12] have reported on a causal temporal framework allowing for 'variable' time intervals. In general much attention has been given to temporal reasoning by the AI community in the recent years, both at the theoretical and applicative levels (e.g. References 13–20).

*But not necessarily the same vertebrae which is a limitation of the representation if this is what is required to be expressed.

Relevant synonyms

The concept of *relevant synonyms* is illustrated through an example from the domain of skeletal dysplasias. The SDD domain model includes the following relations:

syn-subjects (vertebrae vertebral-bodies)
synonyms ((platyspondyly) (vertebral-bodies flat))

Let the query finding F be (vertebral-bodies flat). The relevant synonyms of F are (platyspondyly) and (vertebrae flat). The first synonym is a *direct synonym* obtained from the *synonyms* relation in the domain model. The second synonym is a *direct subject synonym* obtained by directly substituting a synonymous subject for the query-finding subject. Again the synonymous subjects are directly given in the domain model through the *syn-subjects* relation. This example illustrates the two direct routes through which relevant synonyms may be derived. In addition there are two indirect routes. If the subject platyspondyly had any synonymous subjects, or if finding (vertebrae flat) had any direct synonyms, then more relevant synonyms would have been obtained. Thus the first indirect route is to substitute synonymous subjects for the direct synonyms of F and the second indirect route is to determine direct synonyms for the direct subject synonyms of F.

The relevant synonyms of a query finding F are therefore defined as follows:

$$\begin{aligned} \textit{relevant-synonyms}(F) = {} & \textit{direct-syn}(F) \cup \\ & \textit{direct-subj-syn}(F) \cup \\ & \textit{indirect-subj-syn}(F) \cup \\ & \textit{indirect-syn}(F) \end{aligned}$$

$$\begin{aligned} \textit{direct-syn}(F) = {} & \{\Phi \mid \textit{synonyms}(\Phi_a\ \psi_a) \text{ and} \\ & \textit{directly-matches}(F_a\ \psi_a) \text{ and} \\ & \textit{time-interval}(\Phi) \\ & = \textit{time-interval}(F)\} \end{aligned}$$

where X_a is the atemporal finding of finding X.

$$\begin{aligned} \textit{direct-subj-syn}(F) = {} & \{\Phi \mid \textit{syn-subjects}(\textit{subject}(\Phi)\ \textit{subject} \\ & (F)) \text{ and } \textit{attribute-values}(\Phi) = \\ & \textit{attribute-values}(F) \text{ and} \\ & \textit{time-interval}(\Phi) = \\ & \textit{time-interval}(F)\} \end{aligned}$$

$$\textit{indirect-subj-syn}(F) = \bigcup_i \textit{direct-syn}(\Phi_i)$$

where

$$\Phi_i \in \textit{direct-subj-syn}(F)$$

$$\textit{indirect-syn}(F) = \bigcup_i \textit{direct-subj-syn}(\Phi_i)$$

where

$$\Phi_i \in \textit{direct-syn}(F) \cup \textit{indirect-subj-syn}(F)$$

The direct synonyms of a finding are obtained from the *synonyms* relation in the domain model. The findings in the instances of this relation are atemporal and hence the

time interval of the query finding is removed. The resulting atemporal finding is matched against the arguments of the instances of the *synonyms* relation. If a direct match occurs with one of the two argument findings, the other finding augmented with the relevant time interval is a direct synonym of the query finding. An atemporal finding directly matches another atemporal finding if they have identical subjects and the attribute values of the former are subsumed by the attribute values of the latter. For example (metaphyses flared) directly matches against (metaphyses wide irregular), 'flared' and 'wide' being synonymous values (in the context of the subject 'metaphyses'). For efficiency reasons the *synonyms* relation is implemented as tuples (equivalence classes) of synonymous findings, indexed by the relevant subjects. Similarly, the *syn-subjects* relation is implemented as tuples of synonymous subjects.

Completeness of relevant-synonyms definition

The completeness of the *relevant-synonyms* definition is proven below, starting with the following assumptions:

- *Canonical subject assumption:* One subject from each equivalence class of synonymous subjects is designated as the canonical subject.
- *Minimality assumption:* The equivalence classes of synonymous findings are expressed in a minimal way, i.e. only the canonical subjects are used.
- *Completeness assumption:* The knowledge base is complete with respect to all synonym relations, i.e. the equivalence classes of synonymous subjects, the minimal equivalence classes of synonymous findings, and the equivalence classes of synonymous attribute values are complete.

Suppose that the *relevant-synonyms* definition is incomplete, meaning that one of the relevant synonyms of some finding, which is derivable from the knowledge base, is not covered by the definition.

The above procedural* definition of relevant synonyms (which facilitates implementation) is 'equivalent'† to the following declarative definition:

- P is a *relevant synonym* of F if

  syn-subjects(*subject*(P), *subject*(F)) and
  attribute-values(P) = *attribute-values*(F) and
  time-interval(P) = *time-interval*(F)

- P is a *relevant synonym* of F if

  synonyms(X, Y) and
  imatches(F_a, X) and
  imatches(P_a, Y) and
  time-interval(P) = *time-interval*(F)

  where X and Y are atemporal findings; $F_a(P_a)$ is the atemporal finding of $F(P)$, and *imatches*(U_a, V_a) if

  {*subject*(U_a) = *subject*(V_a) or
  syn-subjects(*subject*(U_a), *subject*(V_a))} and
  (*attribute-values*(V_a) ⇒ *attribute-values*(U_a))

  i.e. *imatches*(U_a, V_a) if $V_a \Rightarrow U_a$ (V_a subsumes U_a where subsumption includes identity).

A relevant synonym is therefore not covered if

- the *syn-subjects* relation in the first clause is not satisfied,
- either the *synonyms* or one of the *imatches* relations in the second clause is not satisfied; if an *imatches* relation is not satisfied this means that either the *syn-subjects* relation is not satisfied or the ⇒ relation is not satisfied; the ⇒ relation is not satisfied if synonym-value relations are not satisfied.

In each of the above cases there is a violation of the completeness assumption (the synonymous subjects are incomplete, or the minimal synonymous findings are incomplete or the synonymous attribute values are incomplete). Hence the above definition of relevant synonyms is complete.

Using non-redundant internal representation

The ability of a system to understand synonymous expressions (through synonymous findings, synonymous finding subjects and synonymous attribute values) is considered by the domain experts of the SDD system as an important requirement because it gives much flexibility to the users of the system in entering information directly. However the question that justifiably arises is whether this requirement should be catered for by the user interface or whether the system should support an internal redundant representation, requiring the dynamic correlation of synonymous expressions. If the first option is taken then a front-end processor would translate every user finding to its internal canonical (standard) representation; on output, findings would be translated back to the expressions used by the user. The advantages of this option are that it reduces the run-time processing and, to a large extent, communicates with users in their own terms (with the added possibility of expressing their findings in the system's terms).

The benefit of the second approach is that the information is communicated back to the user as entered without incurring any overheads, whilst with a standard internal representation it may be difficult to translate some item of information back to its original form. The canonical form for some finding, chosen on the basis of uniformity and other criteria which aim to eliminate

*Procedural because it imposes a specific sequence in the operations involved.

†'Equivalent' because all findings returned by the procedural definition are also returned by the declarative definition and any finding returned by the declarative definition which is not explicitly returned by the procedural definition is directly related to one returned by the procedural definition through their attribute values. These findings do not need to be returned explicitly since the matcher can dynamically establish whether such a link exists between two findings.

ambiguity, may not in fact be the preferred form from the user's point of view.

Synonymous attribute values

Attribute values may be synonymous in the context of specific subjects. For example (metaphyses flared) may be taken as synonymous with (metaphyses wide), and (thorax broad) is synonymous with (thorax wide). However metaphyses are not described as broad nor is thorax described as flared. Nonetheless the three terms {broad, wide, flared} are considered synonymous in the SDD system. The tradeoff here is between the simplicity of this representation, which may result in some unnecessary processing, and a more complex representation structure which makes explicit the context under which attribute values are synonymous.

Findings which are synonymous through their attribute values are dynamically determined by the matcher.

Negative, positive and status findings

The *negative finding* for a subject at a given time and locality is an atomic finding whose attribute value is the negative value for the subject (see the decide-status data model above). A negative finding expresses a normal situation and thus excludes the relevant positive findings. Examples of negative findings are (platyspondyly absent) and (hands normal). A *positive finding* therefore expresses an abnormality. For each subject there is a single negative finding for a given time and locality, a *most general positive finding* and a number of more specific positive findings. The most general positive finding for an abnormal subject simply expresses the presence of the given subject, and for a normal subject it expresses the fact that the given subject is abnormal, e.g. (platyspondyly) and (hands abnormal).

Some anatomical-component subjects have a *status finding* which expresses the absence of the given finding subject (at a particular time and locality), e.g. (humeri absent) or (skull-vault absent-ossification). The status finding is an atomic finding whose attribute value is the status value for the subject. A status finding is a positive finding since it expresses an abnormality. However it is a special positive finding because it excludes all other findings for that subject (for the given time and locality).

The matcher operation is as follows

Let Φ be

$$\bigcup_i \{holds(F_i)\}$$

where $F_i \in$ *relevant-findings* (Q, Group) and Ψ be *holds*-(Q).

- *Step 1:* If $\Phi \vdash \Psi$ then return true. If $\Phi \vdash \sim\Psi$ then return false.
- *Step 2:* Let $S \in$ *relevant-synonyms* (Q, Group), $\underline{\Theta}$ be

$$\bigcup_i \{holds\ (F_i)\}$$

where $F_i \in$ *relevant-findings* (S, Group), and Σ be *holds*(S). If $\Theta \vdash \Sigma$ then return true. If $\Theta \vdash \{\Sigma$ and $\Sigma \leftrightarrow \Psi$ then return false.
- *Step 3:* Ψ is unknown.

If Step 1 fails, Step 2 is tried, and, if that fails as well, the query finding Q is determined as unknown by the matcher. In order to decide whether a finding matches, or is refuted, against a group of findings, the matcher uses the following axioms.

- *Axiom 1: negative-finding* (N) and *positive-finding* (P) and *subject* (N) = *subject*(P) and *overlapping*(*locality*(N), *locality*(P)) and *overlapping* (*time-interval*(N), *time-interval*(P)) and *holds*(P) → ∼*holds*(N).
- *Axiom 2: negative-finding*(N) and *positive-finding*(P) and *subject*(N)=*subject*(P) and *overlapping*(*locality*(N), *locality*(P)) and *overlapping*(*time-interval*(N), *time-interval*(P)) and *holds*(N)→ ∼*holds*(P).
- *Axiom 3: negative-finding*(N_1) and *negative-finding*(N_2) and *subject*(N_1)=*subject*(N_2) and *includes*(*locality*(N_1), *locality*(N_2)) and *includes*(*time-interval*(N_1), *time-interval*(N_2)) and *holds*(N_1)→*holds*(N_2).
- *Axiom 4: positive-finding*(P) and *most-general-positive-finding*(M) and *subject*(P)=*subject*(M) and *includes*(*locality*(P), *locality*(M)) and *includes*(*time-interval*(P), *time-interval*(M)) and *holds*(P)→*holds*(M).
- *Axiom 5: non-status-finding*(N) and *status-finding*(S) and *subject*(N)= *subject*(S) and *overlapping*(*locality*(N), *locality*(S)) and *overlapping*(*time-interval*(N), *time-interval*(S)) and *holds*(N)→ ∼*holds*(S).
- *Axiom 6: non-status-finding*(N) and *status-finding*(S) and *subject*(N)=*subject*(S) and *overlapping*(*locality*(N), *locality*(S)) and *overlapping*(*time-interval*(N), *time-interval*(S)) and *holds*(S)→ ∼*holds*(N).
- *Axiom 7: status-finding*(S_1) and *status-finding*(S_2) and *subject*(S_1)=*subject*(S_2) and *includes*(*locality*(S_1), *locality*(S_2)) and *includes*(*time-interval*(S_1), *time-interval*(S_2)) and *holds*(S_1)→*holds*(S_2).
- *Axiom 8: positive-finding*(P) and $\forall$ $V \in$ *values* (P) $\exists$ F such that *positive-finding*(F) and *match-value*(P, F, V) and *holds*(F)→*holds*(P).
- *Axiom 9: positive-finding*(P) and *value* (P, V) and *positive-finding*(F) and *refute-value*(P, F, V) and *holds*(F)→ ∼*holds*(P).
- *Axiom 10: positive-finding*(P_1) and *positive-finding*(P_2) and *subject*(P_1)=*subject*(P_2) and *includes*(*locality*(P_2), *locality*(P_1)) and *includes*(*time-interval*(P_2), *time-interval*(P_1)) and *value*(P_1, V_1) and *value*(P_2, V_2) and *identical-or-synonymous*(V_1, V_2)→*match-value*(P_1, P_2, V_1).
- *Axiom 11: positive-finding*(P_1) and *positive-finding*(P_2) and *subject*(P_1)=*subject*(P_2) and *overlapping*(*locality*(P_2), *locality*(P_1)) and *overlapping*(*time-interval*(P_2), *time-interval*(P_1)) and *value*(P_1, V_1) and *value*(P_2, V_2) and *opposite-values*(V_1, V_2) →*refute-value*(P_1, P_2, V_1).

(includes (X, Y) means that X includes Y. *subject*, *locality*, *time-interval* and *values* are selector functions which apply to findings.)

INFERENCER

The inferencer is invoked for some finding Q queried relative to a group of findings if Q has relevant implicants in the domain model.

Relevant implicants

The *relevant implicants* of a finding are obtained from the *implies* relation in the domain model:

$$implies(F_1, F_2)$$

i.e. $F_1 \Rightarrow F_2$, where F_1 and F_2 are atemporal findings,

$$relevant\text{-}implicants(F) = positive\text{-}implicants(F) \cup negative\text{-}implicants(F)$$

where

$$positive\text{-}implicants(F) = \{P_t \mid implies(P, C) \text{ and } matches(F_a, C)\}$$
$$negative\text{-}implicants(F) = \{P_t \mid implies(P, C) \text{ and } matches(\sim F_a, C)\}$$

P_t is the temporal finding obtained from P and the time interval of F, F_a is the atemporal finding of F and *matches*(X, Y) is true if the matcher returns 'true' when X is queried against the singleton group of findings consisting of Y.

The *implies* relation collectively covers dependency, definitional, or causality relationships between findings.

Generalisations and restrictions

The *is-a* and *part-of* domain relations are also used to obtain more relevant implicants according to the context-free axioms given below.

- *Axiom 1:* *sholds*(P, S) and *is-a*$(s, S) \rightarrow$*sholds*(P, s). *sholds*(P, S) means that proposition (i.e. finding) P holds for subject S, e.g. (long-bones short)$\rightarrow$(ulnae short).
- *Axiom 2:* $\forall$ s such that *is-a*(s, S) *sholds*$(P, s) \rightarrow$ *sholds*(P, S), e.g. {(ulnae short) and (radii short)} $\rightarrow$long-bones short).
- *Axiom 3:* $\forall$ s such that *part-of*(s, S) *sholds*$(P, s) \rightarrow$ *sholds*(P, S), e.g. {(cervical-spine normal) and (thoraco-lumbar-spine normal)}$\rightarrow$ (spine normal).
- *Axiom 4:* $\exists$ s such that *part-of*(s, S) *sholds*(*abnormal*(s), $s) \rightarrow$ *sholds*(*abnormal*(S), $S)$, e.g. (mid-face abnormal)$\rightarrow$ (face abnormal).
- *? Axiom 5:* *sholds*(P, S) and *part-of*$(s, S) \rightarrow$ *sholds* (P, s).

The fourth axiom deals with the most general positive findings. This is because if a part of some subject is abnormal in some specific way, it does not necessarily mean that the whole subject is similarly abnormal, e.g. the trunk and the limbs may be loosely described as parts of one's stature; if the trunk is short it is not necessarily the case that the stature is also short (although the opposite would be unusual). Similarly, Axiom 5 is not really used because it is not truly context-free (except when the property is normality). Axiom 5 may apply under some interpretations only. For example if the spine is abnormal it does not mean that all the vertebrae on each of the regions of the spine (cervical-spine, lumbar-spine, thoracic-spine) are abnormal. On the other hand, if the spine is irregularly ossified it can be inferred that all the vertebrae are irregularly ossified. In other words if we are dealing with a general abnormality (e.g. 'spine abnormal') it is not possible to infer that all the subcomponents are abnormal. With some specific abnormalities, however (e.g. 'spine irregularly-ossified'), it can be inferred that the given abnormality applies to all the subcomponents. The scope of this axiom therefore depends on domain specific knowledge.

Suppose that the query finding is (ulnae severely-short) and that the given finding is (short-limbed-dwarfism). The latter finding is synonymous with (long-bones severely-short) which is a generalisation of the query finding. Hence the inferencer, by dynamically generating implications based on taxonomic relations (*is-a* and *part-of*), enables 'linking' between the query finding and a given finding which is synonymous with generalisations/restrictions of the query finding. This example illustrates the co-operation between the matcher and inferencer.

Inferencer operation

The inferencer operation is simple to express since the inferencer calls Decide-Status recursively to decide the truth status of the antecedents of the relevant implicants of the query finding. If a positive implicant is entailed to hold then the query finding holds (the inferencer returns true) and if a negative implicant is entailed to hold the query finding does not hold (the inferencer returns false). Otherwise the answer to the query is decided to be unknown by the inferencer. The inferencer axioms are given below.

Let Γ be the assertions about which findings hold for the given problem

- *Axiom 1: Let* $P \in$ *positive-implicants*(F). If $\Gamma \vdash$ *holds*(P) then $\Gamma \vdash$ *holds*(F).
- *Axiom 2:* Let $N \in$ *negative-implicants*(F). If $\Gamma \vdash$ *holds*(N) then $\Gamma \vdash \sim$*holds*(F).

If the problem theory is consistent, i.e. both the domain model and problem findings are consistent, then the two axioms are never simultaneously applicable for the same query finding. The chain of implications which is dynamically constructed through the recursive calls between Decide-Status and the inferencer is recorded so that a loop can be immediately detected and the relevant inference chain terminated.

The matcher and inferencer strategies both compete and co-operate. If the matcher cannot return a firm answer, the inferencer is invoked; however it is the matcher that terminates a successful inference chain generated by the inferencer by determining the relevant terminal antecedent finding as true.

Decide-Status is implemented in Franz LISP on a Sun

3/60 running Unix (declarative definitions for the matcher, inferencer and Decide-Status functions are given in Reference 6). The algorithms for the Decide-Status and the inferencer, illustrating the chain recursion between them are as follows.

The definition below assumes that the argument Finding is a simple finding; the generalisation to include compound findings is given in Reference 6. Finding is queried against a group of findings, Group, which are assumed to hold. Impl-Chain is a chain of implications, initially empty, which is constructed dynamically through the recursive calls between Decide-Status and the inferencer.

```
Decide-Status (Finding, Group, Impl-Chain):
  let Truth-Value = Matcher(Finding, Group)
  if firm(Truth-Value) then return(Truth-Value)
  else let Rel-Impl = relevant-implicants(Finding)
       New-Chain = cons(Finding, Impl-Chain)
    return(Inferencer(Rel-Impl, Group, New-Chain))
```

A firm truth value means 'true' or 'false', but not 'unknown'. The relevant implicants of Finding are those findings which could be used to infer Finding.

```
Inferencer (Rel-Impl, Group, Impl-Chain):
  if null(Rel-Impl) return ('unknown')
  else if member(first(Rel-Impl), Impl-Chain)
    then return (Inferencer(rest(Rel-Impl), Group,
    Impl-Chain))
    else if Decide-Status(first(Rel-Impl), Group, Impl-
    Chain) = 'true'
      then if positive (first (Rel-Impl))
           then return ('true')
           else return ('false')
      else return(Inferencer(rest(Rel-Impl), Group,
      Impl-Chain))
```

The inferencer determines whether any of the relevant implicants can be determined from the group of findings.

No formal analysis of the complexity of the decide-status function has been done. However it is a computationally intensive operation; much of this processing is attributed to the inferencer operation since the inferencer may instigate a number of alternative inference chains which may all prove to be unfruitful. The matcher is always tried before the inferencer. As Decide-Status is invoked in many different reasoning contexts in the expert system (see below), its optimisation would be beneficial. Finally Decide-Status is a modular piece of software which facilitates extensions. New strategies or axioms can be easily added. Such extensions may be accompanied by extensions to the data model.

REALISING INFORMATION-ACQUISITION GOALS THROUGH DECIDE-STATUS

Decide-Status operates on an explicit data model. Its operation is conceptually simple ('given a domain model and a set of findings which are assumed to hold, does a particular finding follow or not?'), but powerful enough to either directly implement or support the implementation of the information-acquisition interaction objectives discussed in the first section.

The data model makes explicit the structure of the domain of findings. Findings are not indivisible, atomic entities, i.e. strings of characters; they have an internal meaningful structure, enabling the decomposition of individual findings and the correlation between findings*.

Mixed-initiative interaction

The dependencies between findings (synonyms, generalisations, restrictions, implications) are explicit; Decide-Status, through these dependencies, can correlate expressions which mean literally or essentially the same thing, and can detect redundancy and inconsistency in the user-volunteered information. As a result of this, the user has the option to volunteer information directly rather than through a sequence of menus. It is still possible that some information volunteered by the user will not be understood by the system in which case the system should inform the user.

As explained in the first section, the overall sequence of questions raised by the system can be viewed at two levels of abstraction. The top level gives the sequence of strategy applications generated by the workings of the problem solver and the bottom level gives the question subsequences corresponding to the different strategy applications[2]. Thus some strategy application determines that a given set of information items should be elicited from the user. However it is the data model and the associated relevant reasoning that organises the required information into an intelligible, coherent sequence of questions. Questions for the same subject are grouped together and preceded by the relevant general questions (most general positive finding and, if applicable, status finding). For example if the system wants to ask whether the skull vault is small it will first try to establish whether the skull vault is abnormal and whether it is ossified (i.e. present) in this order. The context for raising specific questions in a meaningful way is established by asking general questions first. If this were not so then an answer could be ambiguous; for example if it is not known whether the patient exhibits platyspondyly and the system raises the question 'platyspondyly severe?' a negative answer would either mean that 'there is no platyspondyly' or that 'there is platyspondyly but it is mild'. In addition to general questions, there may be other contextual questions, either associated with the subject or specific findings of the subject, which need to be raised prior to asking some particular question. At present the data model does not include such contextual associations, but their inclusion is straightforward.

The ordering of the groups of questions for different subjects is not necessarily arbitrary. Again contextual associations between different subjects, if any, can indi-

*In many systems findings have no structure and internally they can be represented by anonymous codes, which of course speeds up the processing; this is often the case with menu-driven interfaces where strings are displayed on the screen but user replies are translated into internal, non-decomposable codes.

cate a particular sequence. In the SDD system the spatial proximity of anatomical components is used to order questions for adjoining anatomical components consecutively, thus saving the user from having to keep swapping between different X-ray films; hence questions on the ribs and the thoracic spine will be grouped together.

A user reply to a system question is immediately taken into consideration, which is reflected in subsequent questions being screened out. The decide-status function is used prior to asking every question to see whether the answer to the question can already be determined. In some expert systems the questions raised by the system are more general than the currently pursued goal requires, where the generality is not for establishing a meaningful context for raising specific questions but for acquiring all relevant information simultaneously (e.g. instead of asking 'does parameter x of object y have value z?' it will ask 'what is the value of parameter x for object y?'). This is absolutely necessary if the system does not allow the user to volunteer information. In the proposed framework this is not so. Immediately after a questioning round by the system, the user is given the opportunity to volunteer information.

The main objective of expert systems technology is to disseminate (and possibly augment) human expertise[21]. Thus, the majority of users of an expert system, although knowledgeable in the given domain, will not be experts themselves. In this respect, it is possible that information volunteered by the user will be erroneous and that the user will not understand a question raised by the system. Requiring the system to trap erroneous user input and to give appropriate guidance to the user when responding to system questions[7,22,23] puts another level of complexity on the system information-acquisition initiatives. Erroneous input which conflicts with other information can be trapped. Clarifying questions (see below) do not aim to trap erroneous input as such, but rather to trap common inaccuracies in the interpretations of actual situations. Serious inaccuracies result in erroneous input. In the SDD system an on-line database of X-ray images will be used in the validation of user input and in providing guidance to the user in answering a system question. System questions may be illustrated by displaying a relevant image and, similarly, user input may be validated by displaying images which illustrate what the user has said. If the displayed image does not match the actual case image then it is likely that the user has misinterpreted the particular situation*.

Flexibility in user-volunteered information

The instantiation of the data model for a specific domain aims to capture the entire vocabulary regarding the expression of findings, in order to give the user the required flexibility. In addition the user may wish to revoke information.

Solutions to truth maintenance are not cheap since assumptions and inference dependencies must be explicitly represented. On the basis of what information is assumed to hold at some stage in the consultation, partial solutions are generated and pursued. These alternative solutions are evaluated against each other. If some information is revoked then its revocation will favour the solutions conflicting with this information and will reduce the promise of those solutions supported by the revoked information. If the revoked information is critical for a particular solution then that solution may be excluded. Decide-Status does not directly enable information revocation but it supports it to some extent. In a diagnostic context the promises of the various competing hypotheses are dynamically recomputed during every diagnostic cycle, always using the information which is believed to hold at that stage (Decide-Status is called to determine whether hypotheses' expectations are satisfied or refuted and whether case findings are accounted by, or are in conflict with, hypotheses). Hence revocations will be reflected in the new 'promise' for an active hypothesis. In practice, however, because Decide-Status is a computationally expensive process, some of its derivations are stored for future use. If a revocation results in the refutation of a previous derivation then inconsistencies will occur. A number of domain-independent truth-maintenance frameworks have been proposed in the literature[26–28] which present theoretically interesting approaches but which may fail in practice because of the associated computational or other overheads. In practice, for reasons of viability, a truth-maintenance framework will be tailored to the specifics of a particular problem solver, or may even be 'hard-wired' in the workings of the problem solver. In the SDD system the case findings are divided into 'hard' findings and 'soft' findings. The distinction between hard and soft findings is made in many medical expert systems, often to distinguish between laboratory findings and co-incidental (circumstantial) findings. SDD's meaning and usage of these terms may be different from other systems, although the concept is essentially the same: some findings are more important than others in a diagnostic context†. Hard findings correspond to diagnostically significant observations of abnormalites. Soft findings describe abnormalities which may be attributed to natural causes. To be acceptable, any hypothesis must account for a reasonable proportion of hard findings, while soft findings may conceivably be ignored. Obviously, revoking a hard finding will affect the solution space but the revocation of a soft finding will not necessarily affect the solution space. Similarly, some findings are rather critical for certain hypotheses, i.e. the presence of some finding results in concluding the hypothesis, or the absence of some finding results in refuting the hypothesis. Again any revocations directly or indirectly resulting in refuting or establishing the presence of such findings must be taken into consideration when some information is revoked. Decide-Status can be used to determine the truth status of such critical findings, which have resulted in considerably enhancing or reducing the promise of potential alternative solutions; the reversal of potentially irreversible decisions on the part of the problem solver may thus be possible.

*The use of images in computer-aided diagnosis in radiology is discussed in References 24 and 25.

†Again Decide-Status is used to decide whether a case finding is soft or hard; this reasoning context is outside the scope of the paper.

Similarly, conflicts in the user-volunteered information can be detected by Decide-Status. If a new observation is determined as false by the previous observations the proof tree constructed by Decide-Status can also be used to explain the conflict (see Reference 10). Decide-Status can be used to determine dependencies between the items of information volunteered by the user. If some observation is deducible from other observations then it is redundant. For example if the user says that the stature is abnormal and subsequently says that the stature is short, the first observation is eliminated. In a diagnostic context pieces of evidence should be independent. Equally, hypotheses' expectations must be non-redundant; again Decide-Status is used to eliminate possible redundancies. In addition the data model (but not Decide-Status *per se*) is used to temporally screen a hypothesis profile, thus eliminating expectations whose (relative) temporal aspects refer to the future with respect to the patient (see Reference 11).

System competence in querying user

The system maintains two lists of findings: Observations and Unknown Findings. The former includes the findings which are currently assumed to hold; some of these findings are directly volunteered by the user whilst others are elicited from the user through questions. Revoking a user observation results in deleting the particular observation from Observations. The Unknown Findings list includes all findings which have been specified by the user, in response to a system question, as unknown. The previously unknown answer to a question may subsequently become known to the user, who will have the opportunity to volunteer this information. When new information is volunteered by the user, the Unknown Findings must be revised, since the new information may result in determining a firm truth status for a previously unknown finding. Decide-Status is thus invoked to determine the truth status of Unknown Findings against Observations.

Prior to asking a question, the system first checks whether the truth status of the query finding can be determined from the Observations list. If *Decide-Status* (query-finding, Observations, nil) returns 'unknown', the system checks whether the user has already said that the given item of information is unknown, i.e. *matcher* (query-finding, Unknown-Findings) returns a firm answer (true or false) or if the query finding is more specific than one of the findings in the Unknown Findings list; specificity here is determined using *is-a* and *part-of* relations. For example suppose that the user was asked the question 'skull abnormal?' to which the reply 'unknown' was given. (Unknown replies are fairly common in the domain of SDD since often the X-ray images for the patient do not give a complete skeletal survey and few clinical findings are usually available). If subsequently the system wants to find whether the skull is small it will not ask the question because it knows that the status of the skull is unknown. Similarly, if the user does not know if the spine is abnormal, the system should not ask whether the cervical spine is abnormal. However if the user does not know about the status of the cervical spine it is still intelligible on the part of the system to ask about the status of the spine.

Since specific questions are always preceded by the relevant general questions (see above), if the Unknown Findings list includes 'platyspondyly severe' then it can be inferred that the presence of platyspondyly has already been established; otherwise if platyspondyly were absent, 'platyspondyly severe' would be false, not unknown.

Finally the data model associates prompting and clarifying questions, and temporal information (*when-to-ask* relation) with specific findings. These associations contribute much to the naturalness of the information-acquisition interaction. Prompting and clarifying questions are raised in response to information volunteered by the user whilst temporal information is used to stop the system from asking unintelligent questions such as 'is the gait of your 2 month old baby waddling?'. *When-to-ask* relations are used in the context of temporal screening. The three relations are as follows:

- *prompting-query*($X_a Y_a$) e.g. *prompting-query* ((platyspondyly) (platyspondyly throughout)),
- *clarifying-query*($X_a Y_a$) e.g. *clarifying-query* ((face flat) (mid-face hypoplastic)),
- *when-to-ask*(W_a T) e.g. *when-to-ask* ((kyphoscoliosis severe) (= + yrs 1)).

where X_a, Y_a and W_a are atemporal findings and T is a relative time point.

Suppose that the user volunteers finding F. If *matcher*(X_a, [F_a]) returns true, i.e. if X_a holds given F_a, then the temporal version of the corresponding atemporal prompting query will be generated. Thus if the user volunteers 'platyspondyly' then the question 'platyspondyly throughout?' is prompted. If the user volunteers 'platyspondyly mild' again the same question will be prompted. However if the user volunteers 'platyspondyly throughout' no prompting will be given; the prompting query is generated but it is immediately quashed because Decide-Status indicates that it is a redundant question. Clarifying queries are dealt with in the same way. Thus if the user volunteers 'face flat', the clarifying question 'do you mean mid-face hypoplastic?' will be raised. A future SDD extension is for the system to infer some prompting and clarifying questions.

Regarding *when-to-ask* associations, suppose that F_a is an atemporal feature of some hypothesis. If *matcher* (F_a, [W_a]) returns true, then, if T refers to a future point in time relative to the patient, the particular expectation of the hypothesis is screened out. Thus if the patient is neonatal then the expectation of 'severe kyphoscoliosis' for any hypothesis will be screened out. However the more general expectation of 'kyphoscoliosis' will not be screened out, since mild forms of this abnormality are observable from birth.

The data model and the decide-status reasoner can therefore directly support the realisation of most of the information-acquisition interactions objectives mentioned in the first section.

CONCLUSIONS

The need for competent conversational structures between expert systems and their users was identified early in the life of expert systems technology; over the years, as the role of expert systems as intelligent advisers has acquired more prominence, so has the need for adequate interaction with users. The conversational structure encompasses the entire interaction between the system and the user. This includes the questions raised by the system, both individually as well as an ordered sequence, the instructions and guidance given to the user by the system for performing actions and answering questions, the system's explanations regarding its reasoning and/or suggestions, and the kinds of initiatives and choices which the user is allowed to take, e.g. volunteer data, volunteer suggestions, and retract data or suggestions. The issue of adequate explanations has attracted more attention than the other aspects of a conversational structure. These other aspects, the information-acquisition ones, are equally important. The focus regarding information-acquisition interaction aspects is on intelligent front ends for making a piece of software or database more usable rather than in the context of expert consultant systems *per se*[7]. However attention is shifting in this direction[29].

To be able to conduct an intelligent information-acquisition interaction, both in the sense of asking intelligent, coherent questions, and in the sense of understanding the user's information volunteering, an expert system must have common sense in a limited way specific to the needs of the particular application. This objective is much easier to achieve than the considerably more ambitious objective of capturing the entire body of world knowledge, as in the CYC project[30].

The argument presented in this paper is that an expert system can communicate more intelligently if it has a deeper understanding of the data for some problem case, i.e. if it is capable of handling these data intelligently; this is the limit of the required common sense. The paper is not about intelligent databases or intelligent front ends. The data model is not a database schema, but rather a schema for capturing the knowledge about the data in the domain. The relations included in the data model are largely common-sense relations such as *part-of*, *is-a*, triggers, and implications, as well as prompts, clarifying questions and *when-to-ask*. The axioms used by the inferencer and the matcher are common-sense axioms, which in fact enhances the generality of the data model.

The paper does not provide a complete generic solution to the problem of intelligent information-acquisition interactions for expert systems, but describes a kernel which points towards a complete solution. The starting point is to make explicit the knowledge about the domain data through a data model; the data model is at the knowledge metalevel, the instantiation of it is at the knowledge object level and the problem data are at the factual level.

ACKNOWLEDGEMENTS

We are grateful to the Leverhulme Trust for their support of the SDD project, to the referees of the paper for their useful comments on an earlier draft of the paper, and to Sophia Prevezanou for her help in clarifying many points.

REFERENCES

1 Gilbert, N 'Explanation and Dialogue' *Knowledge Engineering Review* Vol 4 No 3 (1989) pp 235–247
2 Keravnou, E T and Johnson, L *Competent Expert Systems: A Case Study in Fault Diagnosis* Chapman and Hall (1986)
3 Keravnou, E T, Dams, F, Washbrook, J, Dawood, R, Hall, C and Shaw, D 'Background Knowledge in Diagnosis' *Artificial Intelligence in Medicine* Vol 4 No 4 (1992) pp 1–17
4 Keravnou, E T, Washbrook, J, Dawood, R M, Hall, C M and Shaw, D 'A Model-Based Diagnostic Expert System for Skeletal Dysplasias' *Proc. AIME '89* Springer-Verlag, Germany (1989) pp 47–56
5 Stenton, S P 'Dialogue Management for Co-operative Knowledge-Based Systems' *Knowledge Engineering Review* Vol 2 No 2 (1987) pp 99–122
6 Keravnou, E T, Washbrook, J and Dams, F 'Explicit Data-Modelling in Second-Generation Diagnostic Expert Systems' *Proc. 11th International Conference on Expert Systems and their Applications – Vol 2* Avignon, France (1991) pp 185–198
7 Kok, A J 'A Review and Synthesis of User Modelling in Intelligent Systems' *Knowledge Engineering Review* Vol 6 No 1 (1991) pp 21–47
8 Keravnou, E T and Washbrook, J 'Deep and Shallow Models in Medical Expert Systems' *Artificial Intelligence in Medicine* Vol 1 No 1 (1989) pp 1–28
9 Southwick, R W 'Explaining Reasoning: An Overview of Explanation in Knowledge-Based Systems' *Knowledge Engineering Review* Vol 6 No 1 (1991) pp 1–19
10 Keravnou, E T and Johnson, L 'Intelligent Handling of Data by Integration of Commonsense Reasoning' *Knowledge-Based Systems* Vol 1 No 1 (1987) pp 32–42
11 Keravnou, E T and Washbrook, J 'A Temporal Reasoning Framework used in the Diagnosis of Skeletal Dysplasias' *Artificial Intelligence in Medicine* Vol 2 No 5 (1990) pp 239–265
12 Console, L, Janin Rivolin, A and Torasso, P 'Fuzzy Temporal Reasoning on Causal Models' *Int. J. Intelligent Systems* Vol 6 No 2 (1991)
13 Allen, J F 'Towards a General Theory of Action and Time' *Artificial Intelligence* Vol 23 (1984) pp 123–154
14 Dean, T L and McDermott, D V 'Temporal Data Base Management' *Artificial Intelligence* Vol 32 (1987) pp 1–55
15 Kahn, M G 'Model-Based Interpretation of Time-Ordered Medical Data' *PhD Dissertation* Medical Information Sciences, University of California, USA (1988)
16 Keravnou, E T (Ed.) 'Medical Temporal Reasoning' *Artificial Intelligence in Medicine* Vol 3 No 6 (1991) (special issue)
17 Kowalski, R and Sergot, M 'A Logic-Based Calculus of Events' *New Generation Computing* Vol 4 (1983) pp 67–95
18 Long, D 'A Review of Temporal Logics' *Knowledge Engineering Review* Vol 4 No 2 (1989) pp 141–162
19 Rosser, B, Washbrook, J, Campbell, J, Keravnou, E T and Long, D 'A Framework for Time Dependent Reasoning Systems' *Product P111-1* Esprit Project P2409 Equator (1989)
20 Shoham, Y 'Temporal Logics in AI: Semantical and Ontological Considerations' *Artificial Intelligence* Vol 33 (1987) pp 89–104
21 Slatter, P E 'Cognitive Emulation in Expert System Design' *Knowledge Engineering Review* Vol 2 No 1 (1987) pp 27–41
22 Wolstenholme, D 'Saying "I don't know" and Conditional Answers' *in* Moralee, D S (Ed.) *Research and Development in Expert Systems IV* Cambridge University Press (1987) pp 115–125
23 Wolstenholme, D 'External Data in Logic-Based Advice Systems' *PhD Thesis* Dep. Computing, Imperial College London, UK (1990)
24 Mutalik, P G, Fisher, P R, Weltin, G and Swett, H A 'Expert System Advice as a By-product of Image Acquisition and Reporting: Obstacles to Overcome' *Proc. CAR '91* Springer-Verlag (1991) pp 315–320
25 Swett, H A 'Computer-Aided Diagnosis in Radiology' *Proc CAR' 91* Springer-Verlag (1991) pp 738–743

26 de Kleer, J 'An Assumption-Based TMS' *Artificial Intelligence* Vol 28 (1986) pp 127–224
27 Doyle, J A 'A Truth-Maintenance System' *Artificial Intelligence* Vol 12 (1979) pp 231–272
28 Inoue, K 'Pruning Search Trees in Assumption-Based Reasoning' *Proc. 8th International Workshop on Expert Systems and their Applications* (1988) pp 133–151
29 Keravnou, E T and Washbrook, J 'What is a Deep Expert System? An Analysis of the Architectural Requirements of Second-Generation Expert Systems' *Knowledge Engineering Review* Vol 4 No 3 (1989) pp 205–233
30 Lenat, D B, Ramanthan, V G, Pittman, K, Pratt, D and Shepherd, M 'CYC: Towards Programs with Common Sense' *CACM* Vol 33 No 8 (1990) pp 30–49

BIBLIOGRAPHY

Neal, I M 'First Generation Expert Systems: A Review of Knowledge Acquisition Methodologies' *Knowledge Engineering Review* Vol 3 No 2 (1988) pp 105–145
Wilson, M, Duce, D and Simpson, D 'Life Cycles in Software and Knowledge Engineering: A Comparative Review' *Knowledge Engineering Review* Vol 4 No 3 (1989) pp 189–204