Artificial Intelligence in Medicine (2006) **36**, 85—117

ELSEVIER

ARTIFICIAL INTELLIGENCE IN MEDICINE

http://www.intl.elsevierhealth.com/journals/aiim

An intelligent tutoring system for visual classification problem solving

Rebecca S. Crowley [a,b,c,*], **Olga Medvedeva** [a]

[a] *Center for Pathology Informatics, University of Pittsburgh School of Medicine, Pittsburgh, PA, USA*
[b] *Center for Biomedical Informatics, University of Pittsburgh School of Medicine, Pittsburgh, PA, USA*
[c] *Intelligent Systems Program, University of Pittsburgh, Pittsburgh, PA, USA*

Received 24 May 2004; received in revised form 31 December 2004; accepted 10 January 2005

KEYWORDS
Intelligent tutoring systems;
Knowledge-based systems;
Cognitive tutoring systems;
Classification problem solving;
Ontologies

Summary

Objective: This manuscript describes the development of a general intelligent tutoring system for teaching visual classification problem solving.
Materials and methods: The approach is informed by cognitive theory, previous empirical work on expertise in diagnostic problem-solving, and our own prior work describing the development of expertise in pathology. The architecture incorporates aspects of cognitive tutoring system and knowledge-based system design within the framework of the unified problem-solving method description language component model. Based on the domain ontology, domain task ontology and case data, the abstract problem-solving methods of the expert model create a dynamic solution graph. Student interaction with the solution graph is filtered through an instructional layer, which is created by a second set of abstract problem-solving methods and pedagogic ontologies, in response to the current state of the student model.
Results: In this paper, we outline the empirically derived requirements and design principles, describe the knowledge representation and dynamic solution graph, detail the functioning of the instructional layer, and demonstrate two implemented interfaces to the system.
Conclusion: Using the general visual classification tutor, we have created SlideTutor, a tutoring system for microscopic diagnosis of inflammatory diseases of skin.

1. Background

Knowledge-based systems (KBS) have a long history of use in medical decision support [1], but are rarely designed specifically for educating health profes-

* Corresponding author at: Center for Pathology Informatics, University of Pittsburgh Medical Center, Shadyside, Cancer Pavilion, Room 307, 5230 Centre Avenue, Pittsburgh, PA 15232, USA. Tel.: +1 412 623 1752; fax: +1 412 647 5380.
E-mail address: crowleyrs@msx.upmc.edu (R.S. Crowley).

0933-3657/$ — see front matter
doi:10.1016/j.artmed.2005.01.005

sionals [2]. One reason for this may be the absence of general architectures or frameworks for incorporating medical knowledge bases in larger instructional systems. In this paper, we describe our adaptation of the intelligent tutoring system (ITS)—a well-studied and successful paradigm for creating intelligent educational systems. The advantage of the ITS paradigm is that it provides a basic pedagogic approach with proven efficacy in domains outside of medicine [3–6]. The disadvantage of the ITS paradigm is that it was not designed to use large, frequently changing, or existing knowledge bases. In response to this, we have incorporated aspects of both ITS and KBS design to create a general architecture for instruction in visual classification problem solving. This architecture has three important advantages: (1) it modularizes all domain and pedagogic knowledge into ontologies making the system itself domain-neutral and general; (2) it preserves all of the major pedagogic features associated with cognitive tutoring systems—a highly effective subtype of ITS; and (3) it allows for significant flexibility in the pedagogic components—more flexibility than is generally achieved in current ITS.

1.1. Intelligent tutoring systems

Intelligent tutoring systems are adaptive, instructional systems that strive to emulate the well-known benefits of one-on-one tutoring when compared to other instructional methods. Model tracing ITS (MTITS) are a subtype of ITS in which expertise is represented as a set of production rules that guide the student through the problem space, correcting errors in the intermediate steps and offering hints specific to the current problem state [7]. Model tracing ITS are thus well suited to complex, multi-step problems, in which identifiable steps could be described as ''on the solution path'' or ''off the solution path'' at any given time. Cognitive intelligent tutoring systems (CITS) — a subtype of MTITS — incorporate domain-specific production rules that are based on a cognitive theory of skill acquisition, such as ACT-R [8]. Often, the intermediate cognitive steps are first identified using empirical methods such as cognitive task analysis [9]. Most MTITS and CITS have been developed for domains that are highly procedural and do not require substantial declarative knowledge bases. Examples include mathematics and science instruction [4,6], flight simulator training [10], and training in the workplace [5]. These tutors are among the most rigorously evaluated ITS, and have been shown to be highly effective in increasing student performance. The instructional ''gold standard'' is considered to be one-on-one human tutoring, which is associated with a 2-sigma effect over classroom learning[1] [11]. Cognitive Tutors and other MTITS have been shown to bring students 1-sigma or more above standard instructional methods [4–7]. In contrast, meta-analyses of many traditional computer-assisted instruction systems have exhibited only a 0.3–0.5 sigma effect [12,13].

Only a handful of medical ITS, of any type, have been developed [14–19], and very few of these have been evaluated. The GUIDON project [14,20–23] extensively explored the development of KBS for teaching classification problem solving. Knowledge-based tutors were envisioned as a way to provide instruction in domains that required significant declarative knowledge requirements, embedded with procedural skills. Many medical domains and tasks share these characteristics. GUIDON used MYCIN's rule set to teach medical students to reason about causative organisms in infectious meningitis and bacteremia given a patient's history, physical examination, and laboratory results. Before a case was presented, consultation with MYCIN was used to generate an AND/OR tree representing Goals (OR nodes) and Rules (AND nodes). GUIDON then used the AND/OR tree to structure the discussion with the student. GUIDON interacted with the student using a mixed-initiative method of dialogue. Students could assert, specify, and hypothesize and GUIDON would respond. But GUIDON could also redirect student attention to other problem aspects. Unlike MTITS and CITS, GUIDON made no attempt to limit student options in the problem space or guide them towards the most efficient solution.

One of the central advances of GUIDON was the separation of domain knowledge (represented by MYCIN's ~400 rules) from pedagogic knowledge (represented by GUIDON's ~200 tutorial rules). The modularity of the pedagogic system permitted incorporation with any expert system that utilized the EMYCIN architecture [24,25]. The other central advance of GUIDON was the recognition that KBS like MYCIN can simultaneously perform very well at making decisions and very poorly at teaching others to make decisions [23]. MYCIN's rules represented compiled expert knowledge, that students found ''difficult to understand, remember, and incorporate into a problem-solving approach'' [14]. The authors determined that the ordering of premise clauses contained implicit procedural knowledge

[1] The relative effect size of an educational intervention can be described in relationship to the effect of conventional classroom learning. An intervention which raises performance by two standard deviations over classroom learning is described as having a ''2-sigma'' effect size.

about diagnostic hierarchies and strategies that could not be referenced by GUIDON's tutorial system. This led Clancey and Letsinger to reconfigure the MYCIN rule set — creating NEOMYCIN — an expert model utilized in GUIDON 2 [23]. NEOMYCIN was distinguished from MYCIN by its forward-directed use of data, top-down refinement strategy, use of an etiological taxonomy, and incorporation of rules for (1) expressing implicit "world relations" and (2) reasoning about evidence-hypothesis connections. Because NEOMYCIN reasoned more like human experts, GUIDON-2 appeared to provide more understandable explanations and advice to students.

Like GUIDON, current MTITS typically use domain-specific production rules to accomplish the model-tracing. In domains such as algebra, geometry and physics there is no pressing need to create general rules and scalable architectures, because each type of problem solving can only be modeled by a unique set of production rules. For example, a tutoring system that teaches construction of a geometry proof would require a completely different set of productions when compared to a tutoring system that teaches conversion of an algebra word problem to its equivalent equation form. In contrast, visual classification problem-solving tasks share a very similar structure, and can therefore be modeled with a very general set of production rules, instantiated with domain knowledge to create a plethora of domain-specific models. Thus, a tutoring system for dermatopathology and a tutoring system for X-ray classification of fractures could conceivably share a single general framework but utilize different domain and pedagogic content. In domains that include expansive declarative knowledge bases, this more general approach is necessary to construct systems capable of tutoring across large parts of a domain.

1.2. Knowledge-based systems, ontologies and the UPML component model

Research in KBS over the last 20 years has demonstrated the value of combining ontologies and general reasoning methods [26] (often called abstract problem-solving methods or abstract PSMs). Ontologies are formal specifications of concepts and their relationships [27]. The addition of instances to the ontologically defined classes and relationships creates a knowledge base that can be used to abstract declarative knowledge within a domain. Abstract PSMs combined with the domain knowledge declaratively defined in the knowledge base produce domain specific solutions, but may use a very small number of generic rules. As previously described, very few ITS use these more abstract methods for knowledge representation and reasoning, due partly to the specific domains for which these systems have been developed.

Reusable problem-solving methods and ontologies permit scalability, and ease acquisition and maintenance of knowledge [26]. But they also introduce a specific problem when used in ITS. Knowledge-based systems are designed to solve problems, but the purpose of an ITS is to teach humans to solve problems. Therefore, ITS require pedagogic components that provide feedback on intermediate steps in the solution. In most ITS, this is accomplished with additional rule clauses that are intimately associated with the expert model of task performance. Among cognitive tutors, it is unusual to have pedagogic models that are entirely separate from the model-tracing component [28]. Feedback is very specific to the intermediate steps modeled by a set of domain specific production rules. A more generic approach that couples abstract PSMs and ontologies, must also disentangle the pedagogic and expert models. Otherwise pedagogic feedback becomes increasingly limited and inflexible, because (1) the more abstract productions support an even more limited set of feedback types and (2) the system has no way to instruct across sets of general productions, making all feedback specific to the single general rule to which it applies.

Increasingly, methods for developing highly modular and reusable expert systems are resulting from research on the Semantic Web. The unified problem-solving method description language (UPML) [29] provides a specification for creating distributed reasoning systems. The UPML component architecture [30] provides a detailed specification of relationships between ontologies and abstract problem-solving methods that maximizes the modularity of the components. A *task* defines the problem to be solved by the KBS as a set of subgoals. A *problem-solving method* describes a reasoning process of the KBS. A *domain model* describes the declarative knowledge used by the KBS during problem-solving. Task, domain model, and PSM are entirely independent, and are specified using ontologies. Relationships between parts of the model are described using *bridges* and *refiners*. Bridges are used to model relationships between two architectural components (for example task and PSMs), while refiners model the adaptation of an architectural component.

Given the absence of general frameworks for developing ITS in medical domains, we elected to create our own framework—incorporating aspects of both KBS and ITS design. In particular, we use the highly modular UPML component architecture to create two interlacing KBS—one that produces

the expert model as a dynamic and advancing representation of the solution, and one that produces an instructional layer tailored to the specific student. The instructional layer is therefore independent of the expert model, and is able to inspect the student's progress across the entire solution, providing feedback that is specific to intermediate states that are defined by more than just a single rule.

The visual classification tutor (VCT) reproduces the interaction style of the cognitive tutors, and achieves the scalability of modern KBS, but also provides a novel method for separating the pedagogic system so that it can be tailored to the individual student while operating within the confines of a more general framework. In the VCT, abstract PSMs create a dynamic solution graph (DSG) — a representation of the current problem state and valid next steps — that is updated with each student action. The VCT can be used with different interfaces, different domain ontologies and different pedagogic ontologies to create different visual classification tutoring systems. Using the VCT, we have implemented SlideTutor—a web-deployed system for teaching microscopic classification of inflammatory diseases of skin. The purpose of this paper is to describe the general architecture, knowledge representation, and functionality of the VCT, and to detail how it was used to create SlideTutor.

2. The VCT—a domain neutral system for cognitive tutoring of visual classification problem solving

Visual classification problem solving is a common cognitive task in which the problem-solver uses visual cues to deduce the correct class membership for an instance of unknown class. In medicine, visual classification problem solving is an important aspect of expert performance in radiology, hematology and pathology. An ITS for visual classification problem-solving requires a developmental cognitive model of this task—because the system must be able to provide guidance and advice specific to a particular student's needs, but must also adapt as the student gains proficiency.

Our developmental model of visual classification problem solving derives from two sources: (1) previous empirical and theoretical work in radiology [31—33] and also in non-visual domains [34—38] and (2) an empirical cognitive task analysis [39,40] that we performed to explore differences in the visual diagnostic processes of novice, intermediate, and expert pathologists.

The model on which we have based our tutoring system is summarized in Table 1, as differences in five basic areas of classification problem solving—search and detection, feature identification, feature refinement, hypothesis triggering, and hypothesis evaluation. Early in the development of expertise, students lack basic abilities in searching for regions of interest, and limiting their diagnostic search space. They are frequently incorrect when determining the meaning of particular visual evidence, lack the ability to refine evidence, and do not know how to process evidence towards a diagnostic conclusion. Intermediates, on the other hand, rarely exhibit errors related to search and detection of regions of interest. Instead, they have difficulty in correctly attaching symbolic meaning to visual evidence and knowing when they need to refine evidence. They are often uncertain of the meaning of particular visual cues even when they are correct, and consider an overly broad set of hypotheses. Unlike experts, intermediates do not use backwards reasoning selectively on a small hypothesis set.

The implications of this developmental model for the instructional framework of the VCT are shown in the lower pane of Table 1, as the ITS instructional design requirements. A fundamental characteristic of these requirements is that they assume the ability of the system to change instructional approach as the student progresses through a set of developmental stages. Taken together, these implications argue for the following design principles for our system:

1. The system must be able to determine both correct and incorrect student actions, determine the general class of error that has been made, and leave open the specific instructional response to this error—which may change based on the student model for a particular student, the particular classification task, or the interface itself.
2. The system must reason with the student, in order to provide correct feedback as intermediate solutions are developed. For example, the system should accept hypotheses and even diagnoses based on an early, incomplete or incompletely refined set of evidence. When additional evidence is identified, the system should require that students revise their diagnoses or statements of diagnostic support.
3. The system must be able to reason both forwards (evidence to hypothesis) and backwards (hypothesis to evidence) so that it can support both strategies among students. More novice students may need to learn what decisions remain to be made before a given hypothesis can be confirmed, while more expert students may need to be encouraged to look for particular evidence that separates one or more hypotheses. Both

Table 1 Developmental cognitive model of visual diagnostic expertise and resulting ITS instructional design requirements

Developmental model of skill acquisition by skill type				
	Search and detection	Feature identification	Feature refinement	Hypothesis triggering	Hypothesis evaluation
Novice	Poor search skills [39,40] Limited perceptual recognition of regions of interest (ROI) [31,32,39,40]	Very limited ability to recognize and name histopathologic features [39,40]	Features almost never refined [40]	Few hypotheses triggered [39,40] Triggered hypotheses are overly general - lack knowledge of diagnostic space [40]	Very limited ability to connect features to hypotheses [39,40] Lack knowledge base needed for backwards reasoning
Intermediate	Accurate search skills [39,40] Accurate perceptual recognition of areas of interest [32,39,40]	Many features identified, poor filtering [37,39,40] Frequent errors in feature identification [39,40] Express uncertainty even when feature identification is correct [40]	Feature frequently refined - errors in refinement [40]	Many hypotheses triggered, very early in diagnostic process [34,39,40] Hypothesis set overly broad [40] Fail to change hypotheses even when confronted with inconsistent data [34]	Error prone in connecting features to hypotheses [39,40] Backwards reasoning made necessary but also more challenging by broad hypothesis set [37]
Expert	Accurate search skills [39,40] Rapid perceptual recognition of areas of interest [32,39,40]	Few, highly salient features identified [33,40]	Features only refined when necessary [40]	Many hypotheses triggered, but mainly after final diagnosis has been made, to rule out alternatives. [38,39,40]	Accurate in connecting features to hypotheses [39,40] Backwards reasoning to differentiate among more focused set of hypotheses [36,39,40]
ITS instructional design requirements					
Increasing expertise	*EARLY:* (1.1) monitor location on slide and direct more novice students to areas of interest using symbolic and visual cues (1.2) provide explicit strategies to compensate for poor search skills. *LATE:* (1.3) monitor location on slide to ensure that ROI has been found but otherwise limit feedback on search and detection	*EARLY:* (1.4) require that students connect visual features on the slide with feature names and provide feedback; (1.5) encourage complete articulation of features (1.6) allow full exploration of feature space, but provide feedback on relevancy *LATE:* (1.7) Allow students to "jump" to the diagnosis as long as the ROI has been seen; (1.8) encourage students to search for features in order most efficient for narrowing search;	*EARLY:* (1.9) encourage complete feature refinement and provide feedback *LATE:* (1.10) permit students to reason without fully refined features (1.11) permit sequences in which new hypotheses require re-examination of feature refinement	*EARLY*: (1.12) permit any hypothesis that is consistent with at least one piece of evidence; (1.13) encourage refinement and specialization of hypotheses *LATE:* (1.14) encourage hypotheses that are consistent with all of the features; (1.15) help students learn sets of hypotheses that share similar features (1.16) permit sequences in which new features or feature refinements require re-examination of triggered hypotheses	*EARLY:* (1.17) provide feedback on student asserted relationships between features and hypotheses; (1.18) support both forwards and backwards reasoning, but guide students to use backwards reasoning to help build mental model of diagnostic space *LATE:* (1.19) help students learn when hypotheses should be excluded as students further refine features; (1.20) support both forwards and backwards reasoning but guide more advanced students to use backwards reasoning more selectively

scenarios require the ability to reason from hypothesis to evidence.

In the remainder of this manuscript, we describe the design and implementation of a general knowledge-based tutoring system for visual classification that meets these requirements.

3. Composite general architecture of the VCT

The architecture and implementation we describe is novel—to our knowledge no other ITS has taken this approach. Like other ITS [41], the VCT includes an expert model against which student actions are tested and a pedagogic component that responds to incorrect actions and requests for help. But unlike other ITS [4,6] the VCT does not rely on a set of domain specific production rules as the basis for the expert model. In contrast, the VCT consists of set of abstract PSMs that build a DSG from three separate sets of frames that describe the domain knowledge, task sequence, and the case to be classified. Interaction with the DSG is filtered through an instructional layer created by a second set of abstract PSMs from a set of frames that describe the pedagogic knowledge, pedagogic task sequence, and eventually data related to a particular student.

The components of the VCT (Fig. 1) conform to a proposed standard for KBS—the UPML component model [30]. The expert model is composed of domain model, domain task, and abstract PSMs. The domain model represents the domain knowledge used to solve problems. The domain task represents the goals of the problem-solving process. In combination with the case data, the abstract PSMs create the DSG against which student actions are tested. The instructional model is composed of a separate pedagogic model, pedagogic task model, and PSMs. The pedagogic model represents the pedagogic knowledge used to teach problem solving. The pedagogic task represents the goals of the instructional process. In combination with data reflecting the current state of the student model, the instructional model is manifested as a highly flexible and context-specific instructional layer between the student interface and the DSG.

4. Interaction scenario

In order to clarify our methods, we first provide a single interaction scenario that incorporates examples of a wide range of system functionality. The scenario is taken from the domain of dermatopathology and uses SlideTutor, but similar scenarios could be generated for any domain in which visual classification is feature based. In Sections 5—8, we use these examples to detail the conceptual implementation and function of each of the VCT components. Individual actions enumerated in parentheses, match the tabular format of the interaction scenario shown in Table 2. A video clip showing the interaction scenario in the SlideTutor case-focused interface is available at http://slidetutor.upmc.edu/video.html (accessed: 31 December 2004).

A first year pathology resident is using the SlideTutor system to learn microscopic diagnosis of inflammatory skin diseases. A case is selected by

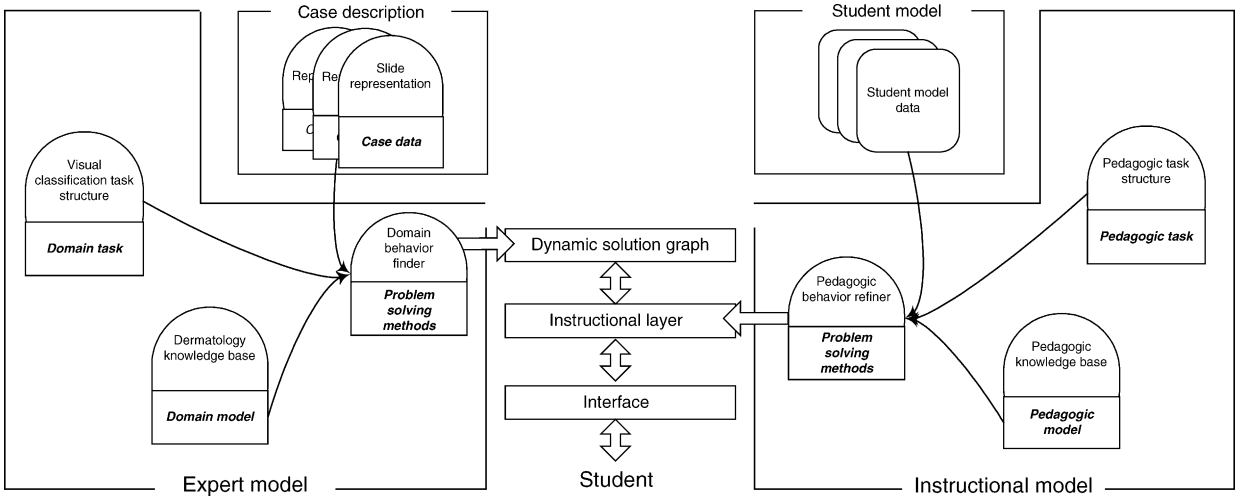

Figure 1 Visual classification tutor component architecture.

Table 2 Student actions and tutor responses corresponding to interaction scenario in text

Action	Student action (SUBTASK type and value)	Tutor response	Description and references
2.1	Identify-Feature: blister	Accept	See corresponding state of DSG in Fig. 5A
2.2	Identify-Attribute (of blister): location = intraepidermal	Failure	Discrete values of attribute modeled (see Section 5). See Table 5, Error S3
2.3	Help Request	Best-next-step	Hint after bug targets previous error (see Section 7)
2.4	Identify-Attribute (of blister): location = subepidermal	Accept	Discrete values of attribute modeled (see Section 5)
2.5	Identify-Feature: mucin	Failure	See example of error delivery in Fig. 7. See Table 5, Error I4
2.6	Identify-Absent-Feature: mucin	Accept	Pertinent negative (see Section 5)
2.7	Assert-Hypothesis: erythema multiforme	Accept	Hypothesis consistent with at least one feature (See Table 7) (Task description)
2.8	Assert-Hypothesis: suction blister	Accept	Task description
2.9	Identify-Feature: inflammatory infiltrate	Accept	
2.10	Assert-Support-Link: inflammatory infiltrate and Erythema Multiforme	Accept	Support-link represented in DSG as present or absent edge between nodes (see Section 6)
2.11	Assert-Support-Link: inflammatory infiltrate and suction blister	Failure	See Table 5, Error E1
2.12	Identify-Feature: neutrophilic inflammatory infiltrate	Failure	Alert after correct action (see Section 7) See Table 5, Error E6
2.13	Identify-Attribute (of neutrophilic inflammatory infiltrate): location = dermis	Accept	
2.14	Identify-Attribute (of neutrophilic inflammatory infiltrate): quantity = minimal	Failure	Continuous values of attribute modeled (see Section 5). See Table 5, Error S2
2.15	Assert-Hypothesis: acquired epidermolysis bullosa	Accept	
2.16	Help Request	Best-next-step	Feature identification precedes hypothesis triggering in current pedagogic task (see Table 6 - State 3). See corresponding state of DSG in Fig. 5B
2.17	Identify-Feature: eosinophilic inflammatory infiltrate	Failure	Ambiguous error states (see Section 7). See Table 5, Errors I2 and I3
2.18	Identify-Attribute (of eosinophilic inflammatory infiltrate): location = dermis	Accept	
2.19	Identify-Attribute (of eosinophilic inflammatory infiltrate): quantity = moderate	Accept	
2.20	Identify-Feature: epithelial necrosis	Failure	See Table 5, Error I1
2.21	Identify-Feature: nuclear dust	Accept	
2.22	Assert-Diagnosis: acquired epidermolysis bullosa	Failure	EC represents integrated evidence-hypothesis relationship (Fig. 6). Student action fails because no edge from EC to asserted diagnosis (see Fig. 5C). See Table 5, Error E10
2.23	Help Request	Best-next-step	Hints are given only for required nodes (see Section 7). See corresponding state of DSG in Fig. 5C
2.24	Assert-Hypothesis: linear IgA dermatosis	Accept	
2.25	Help Request	Best-next-step	
2.26	Assert-Hypothesis: dermatitis herpetiformis	Accept	
2.27	Help Request	Best-next-step	
2.28	Assert-Hypothesis: dermatitis herpetiformis-like drug eruption	Accept	
2.29	Assert-Diagnosis: linear IgA dermatosis	Accept	

Table 2 (*Continued*)

Action	Student action (SUBTASK type and value)	Tutor response	Description and references
2.30	Problem Done	Failure	See Table 5, Error C1
2.31	Help Request	Best-next-step	
2.32	Assert-Diagnosis: dermatitis herpetiformis	Accept	
2.33	Help Request	Best-next-step	
2.34	Assert-Diagnosis: dermatitis herpetiformis-like drug eruption	Accept	
2.35	Problem Done	Accept	

the tutoring system. At the conclusion of the student-system interaction, this case will match to FS-A in Fig. 2, but this is not known by student or system until the problem is solved. The student is presented with a brief clinical history, and a virtual slide that the student uses to pan and zoom—as they would use a microscope. The interface also provides a palette for students to build a diagrammatic representation of their reasoning. When student actions are correct, the diagram is updated to reflect the new state of the argument. When student actions are incorrect, students may modify the diagram to correct errors.

The student begins by scanning through the slide, and finds an abnormal discontinuation of the epidermis. She correctly specifies the location of the abnormality in the image and correctly describes this abnormal feature as ''blister'' (2.1). The system accepts the feature. After looking closely at the blister, the student erroneously concludes that the blister is intra-epidermal (2.2). She is told that although she is wise to evaluate location of the blister, the blister is not intra-epidermal in this case. The diagram reflects this error. The student asks for a hint (2.3), and is told to delete the incorrect location. She properly asserts that the blister is subepidermal (2.4). She suggests dermal mucin in another area (2.5), but is told that dermal mucin is absent in this case, and that absence of mucin is an important negative finding. The student corrects this mistake by asserting absence of dermal mucin (2.6) and notes the location of the absent mucin as within the reticular dermis. The student suggests that this could be erythema multiform (EM) (2.7) or suction blister (2.8). The system adds these hypotheses to the diagram because both are consistent with at least one piece of evidence—the subepidermal blister. In an area that seems hypercellular, she indicates the presence of inflammatory infiltrate (2.9)—a general category with subclasses delineated by cellular type. The system considers this to be correct because both neutrophilic and eosinophilic inflammation (subtypes of inflammatory infiltrate) are present in this slide at the location specified by the student.

The student tries to create explicit support for the diagnosis of EM with inflammatory infiltrate in her diagram (2.10). The system considers this feature supportive and modifies the diagram, because EM

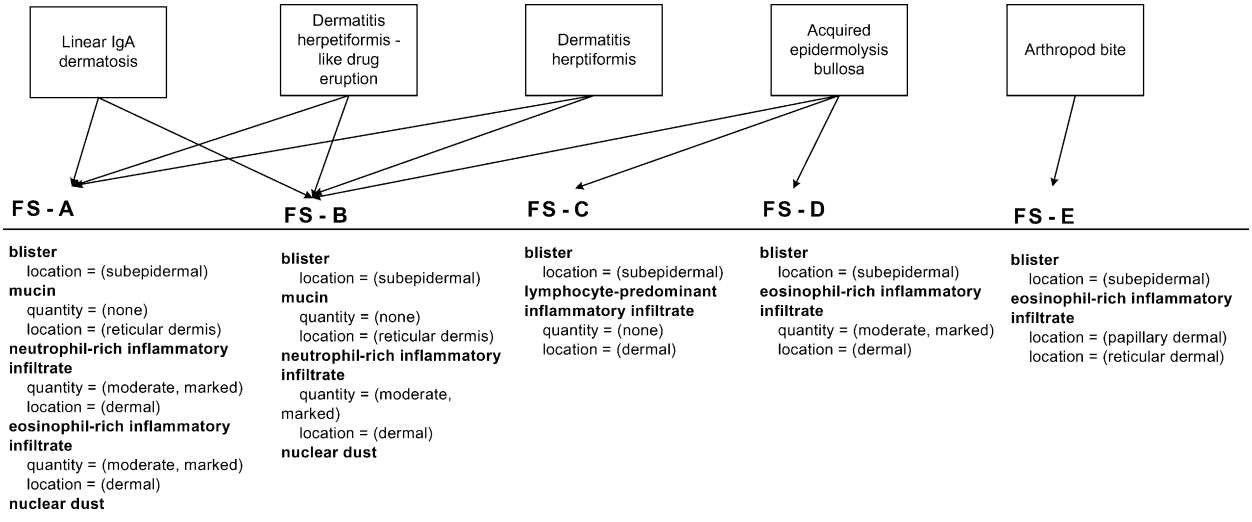

Figure 2 Relationship of FEATURE_SPECIFICATION to DISEASE instances for five feature specifications and five diseases.

is associated with lymphocytic infiltrate (another subtype of inflammatory infiltrate). But when the student tries to support suction blister with inflammatory infiltrate (2.11), the system marks this as incorrect in the diagram and the student removes it. After further consideration, the student modifies the more general concept of inflammatory infiltrate to the more specific concept of neutrophil-rich inflammatory infiltrate (2.12). The system accepts this modification, but also alerts the student that by further specifying the type of infiltrate—the supportive relationship between the inflammatory infiltrate and EM no longer applies. Although lymphocytic infiltrate is seen in EM, neutrophilic infiltrate is not. The student amends the diagram by deleting this explicit relationship. Since she has excluded EM and suction blister she deletes these hypotheses from the palette. She correctly specifies the location of the neutrophilic infiltrate (2.13) and then suggests that the infiltrate is minimal in quantity (2.14). The system responds that although minimal neutrophilic infiltrate can be seen in cases of this entity—the severity is not best described as minimal in this case. The student corrects the error.

The student thinks of acquired epidermolysis bullosa (AEB) (2.15), and adds this to the diagram. But at this point she is stuck. There don't seem to be further features, and these are all the hypotheses she can come up with. She asks for a hint (2.16). The system suggests that there are still further features to find. Uncertain where these features are, she asks for more hints and is taken to the correct field and magnification for the next feature. The student thinks she knows what's here and indicates that there is eosinophil-rich inflammatory infiltrate in a particular area of the image (2.17). The system suggests that the eosinophil-rich inflammatory infiltrate is present—but not in the area that she has indicated. The student asks for additional hints, and the system annotates the correct region. A final hint moves the slide and annotates an area in which the eosinophilia is particularly easy to recognize. The student correctly asserts the location (2.18) and degree (2.19) of the eosinophilic infiltrate. She finds an area near the blister that seems necrotic. She identifies epithelial necrosis (2.20), but is told that this is not a significant feature of this case. The student removes the feature. Thinking that they could be fragments of neutrophils instead, she asserts nuclear dust (2.21), which is accepted into the diagram.

The student feels certain that this is AEB. She tries to make the hypothesis a full-fledged diagnosis (2.22). But the system responds that although blister, neutrophilic dust, and neutrophilia are features of AEB—the eosinophilia is not consistent. The student is unsure what to do next. Are there other important features that she has missed? She asks for another hint (2.23). The system responds that she has found all of the features, but has not asserted all the hypotheses that apply given these features. After additional requests for hints (2.25, 2.27), she is told that all of the features are consistent with linear IgA dermatosis, dermatitis herpetiformis (DH) and dermatitis herpetiformis-like drug eruption (DHLDE). She asserts these hypotheses (2.24, 2.26, 2.28). Which diagnosis is it? She guesses linear IgA dermatosis (2.29) and the system promotes the hypothesis to the status of diagnosis in the diagram, indicating that this is a perfectly acceptable diagnosis. The student tries to conclude the problem (2.30) but is warned that there are more acceptable diagnoses. With additional hints (2.31, 2.33), she is told that DH and DHLDE are also consistent with all of these features. The student adds them to the reasoning palette (2.32, 2.34), and correctly indicates that the problem has been solved (2.35).

5. Expert model—representation of domain knowledge, task and case data

The expert model of the VCT provides the evolving solution that student actions are tested against. Abstract PSMs are applied to (1) instances in the domain knowledge base; (2) instances defining the task sequence and (3) instances derived from case data, to create a DSG. We first describe the three models, and then detail how these frames are used to generate the DSG.

5.1. Domain model

The domain model[2] expresses the relationships between evidence and disease entities. The class concepts and relationships of the domain model are general, and apply widely throughout pathology and other areas of medicine in which classification is feature-based. The class structure of the domain model is depicted in Fig. 3 in relationship to the case model. Our representation slightly extends the

[2] We use the term model to refer to a subset of the complete ontology, specific to a particular purpose. For example, the domain model is the subset of the complete ontology that specifies only the concepts relevant to domain knowledge. In contrast, the task model specifies only the concepts relevant to task knowledge. Separation of domain, task and case concepts are a foundational principle of the UPML component architecture.

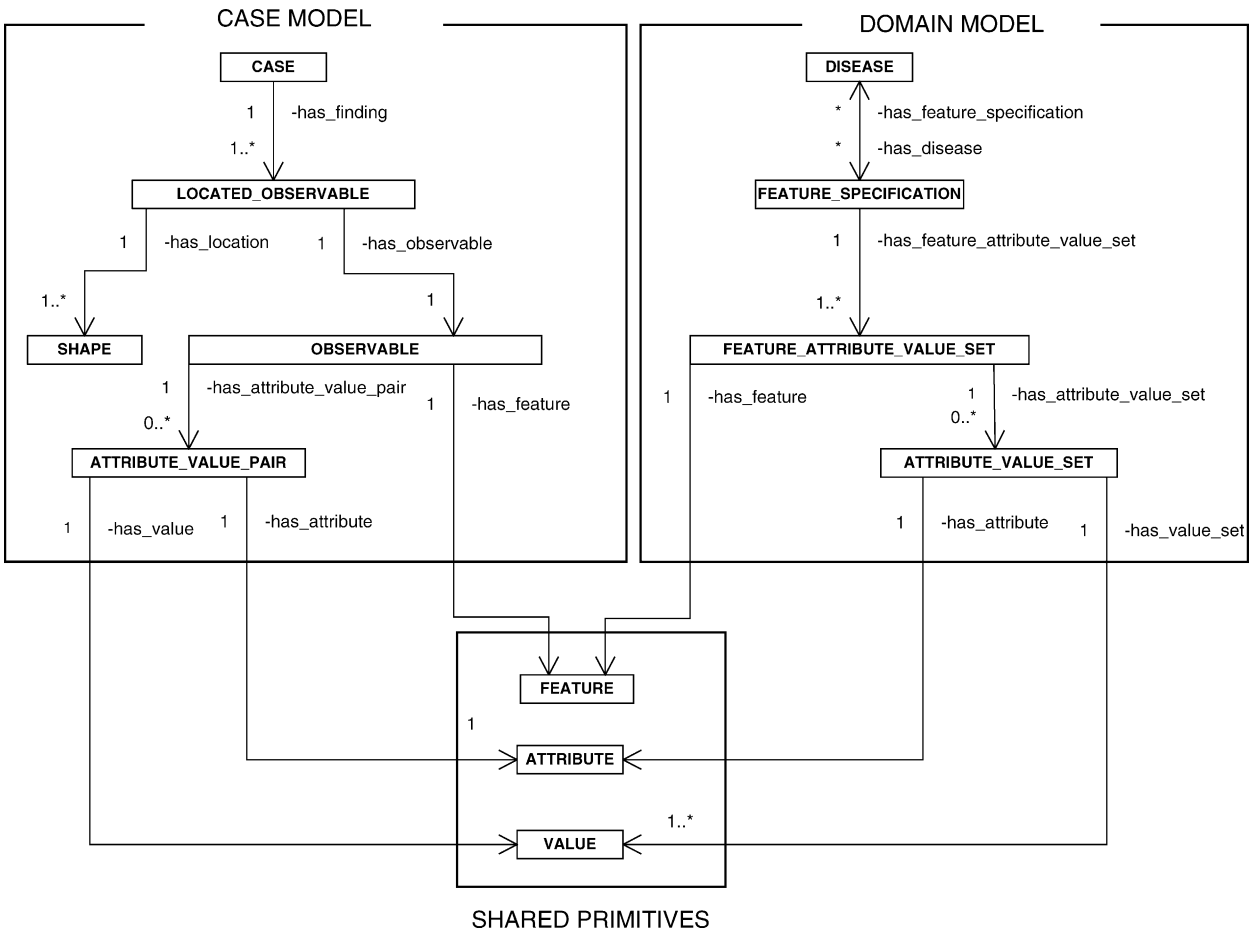

Figure 3 UML class diagram of domain and case showing relationships to shared feature, attribute and value primitives.

ontology for classification problem solving described by Motta and Lu [42], by adding additional attributes to features. A similar representation has also been used to create ontologies for petrographic analysis [43]. Diseases[3] are hierarchically represented. Any disease may have more than one parent (multiple inheritance). Diseases have one or more FEATURE_SPECIFICATION instances—each of which correspond to a combination of evidence. This structure is quite analogous to what Evans and Gadd have termed "facets" in describing clinical reasoning [44]. Each FEATURE_SPECIFICATION consists of one or more FEATURE_ATTRIBUTE_VALUE_SET (FAVS). Each FAVS is in turn composed of two primary components: (1) a single FEATURE and (2) one or more ATTRIBUTE_VALUE_SETS. Instances of FEATURE represent distinct perceptual primitives of visual entities (such as blister) that form the "atoms" of visual feature recognition. Instances of ATTRIBUTE_VALUE_SET represent the additional cognitive steps required for refining these features (such as the distinction of a blister's location relative to the epidermis as subepidermal or intraepidermal). ATTRIBUTE_VALUE_SETS are composed of a single ATTRIBUTE and a set of VALUE.

The domain model therefore represents many-to-many relationships between FEATURE_SPECIFICATION and DISEASE. An example of this relationship is shown in Fig. 2, for five instances of DISEASE under consideration in the interaction scenario, and five of twelve total FEATURE_SPECIFICATIONS associated with these five diseases.

Continuous and discrete values of attributes are modeled differently:

- For continuous values such as the quantity of inflammation, multiple possible values are expressed as a value range of the ATTRIBUTE in a single FEATURE_SPECIFICATION. For example, in Fig. 2, FS-A indicates that the value range of neutrophil-rich inflammatory infiltrate is moderate to marked.

[3] We use the term 'disease' as the class name throughout this manuscript, in order to clarify relationships within a medical context. However, this class is more appropriately named 'visual entity', because the DOMAIN model applies to any visual classification task.

- For discrete values such as the location of inflammation, when an ATTRIBUTE of one FEATURE has multiple values within a single case (logical AND), these are represented as multiple ATTRIBUTE_VALUE_SETS within a single FEATURE_SPECIFICATION. For example, in Fig. 2, FS-E contains two separate ATTRIBUTE_VALUE_SETS for location of the eosinophil-rich inflammatory infiltrate, one for papillary dermis and one for reticular dermis.
- For continuous or discrete values, when an ATTRIBUTE of one FEATURE can have only one of many values (logical OR), these are represented as separate FEATURE_SPECIFICATIONs (not shown in Fig. 2).

Identifying the absence of some features is often a critical aspect of problem-solving. Students must learn when it is important to identify pertinent negatives, because they form branch points in decision making. These 'explicitly absent features' must therefore be represented in the DSG so that the instructional layer can separate these pertinent negatives from the sea of absent features the student might correctly try to note in any individual case. For example, when the student in our interaction scenario says that mucin is present, the system responds that actually it is absent, and that the absence of this feature is salient (Table 2, Action 2.5). Explicitly absent features are represented using the same method as present features, except that the quantity attribute is defined as 'none'. For example, the absence of mucin in the reticular dermis is expressed in FS-A and FS-B (Fig. 2) as a FAVS with feature: mucin, location: reticular dermis, and quantity: none.

5.2. Case

In the VCT, the CASE provides the location and semantic content of each feature encoded in a single tutor case. The class relationships for CASE are shown in Fig. 3. Each CASE consists of a set of LOCATED_OBSERVABLE, each of which combines a single OBSERVABLE and a set of SHAPE that defines the geographic distribution of the OBSERVABLE in the image. OBSERVABLES are composed of one FEATURE and one or more ATTRIBUTE_VALUE_PAIR. ATTRIBUTE_VALUE_PAIR are composed of one ATTRIBUTE and a single VALUE. For continuous values, the single values in ATTRIBUTE_VALUE_PAIR in the case model contrast with the value ranges expressed in ATTRIBUTE_VALUE_SET in the domain model.

Fig. 4 shows the CASE instance used in the interaction scenario. In combination with Fig. 2, it can be surmised that the CASE which is loaded at the start of the interaction scenario will be determined to represent FS-A by both student and system by the conclusion of problem solving. Note that the mod-

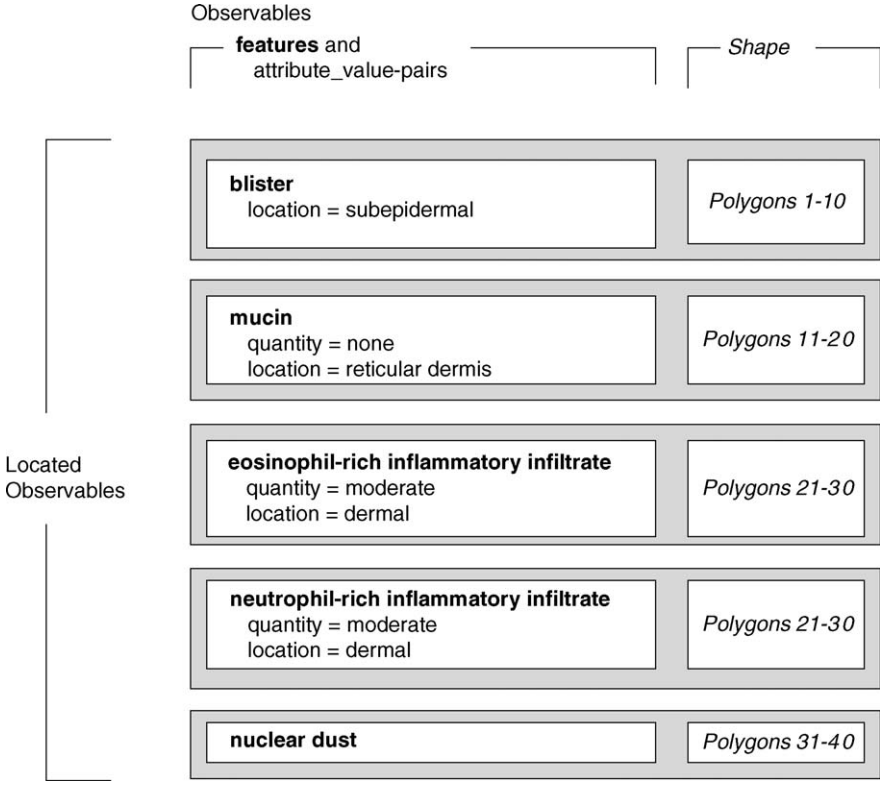

Figure 4 Instance of CASE used in interaction scenario.

erate quantity of inflammatory infiltrate in the ATTRIBUTE_VALUE-PAIR for CASE is within the range of the ATTRIBUTE_VALUE_SET (moderate, marked) for FS-A. One requirement for instances of CASE to match to instances of DISEASE is that all features and attributes match, and that single values within ATTRIBUTE_VALUE_PAIR are within the value ranges of the corresponding ATTRIBUTE_VALUE_SET.

The relationship of the domain model and case model is also shown in Fig. 3. ATTRIBUTE, VALUE and FEATURE are primitives shared by both domain and case models. Instances of CASE are created using a case authoring tool, which constrains authoring to use only the FEATURE_SPECIFICATIONS enumerated in the domain knowledge base. This is more fully described in Section 12.

5.3. Domain task

The task model represents an abstraction of the goal structure for classification problem solving. The task model is composed of a small set of SUBTASK instances, with one SUBTASK for each general cognitive goal in a particular classification problem-solving exercise. The task model used in the interaction scenario (Table 3) contained one SUBTASK for each of the following types of cognitive goals: (1) identify a feature in the image; (2) determine that a critical feature is absent; (3) refine the feature to include attributes and values; (4) assert a hypothesis; (5) accept a hypothesis as a diagnosis; (6) assert a supporting relationship between a feature and a hypothesis; (7) assert a refuting relationship between a feature and a hypothesis and (8) find a feature that distinguishes between one or more competing hypotheses. Each of these generic SUBTASKs may be instantiated to form one or more case-specific goals.

Relationships between instantiated goals are derived from the spatial structure of TASK, and ultimately specify the entire problem space of valid actions for a given case. The Has_Parent slot of SUBTASK is used to create this spatial structure. For example, a SUBTASK of type Assert-Diagnosis contains a Has_Parent slot that may be filled by Assert-Hypothesis or Identify-Feature. When filled by Assert-Hypothesis (as is true in the task shown in Table 3), the diagnosis must be preceded by the formation of a hypothesis. When filled by Identify-Feature, the diagnosis may be asserted directly from the evidence without the need to first trigger and test a hypothesis.

The behavior of each type of goal during problem solving is defined in the Has_Role slot of SUBTASK. This slot specifies whether the goal has an inherent ordering in the subtask (direct role), is an attribute of an already achieved goal (property role), or reflects relationships between the goals (indirect role). Examples of goals with direct role include feature identification and hypothesis triggering because (1) features must be found before they are refined, and (2) hypotheses must be based on features. An example of a goal with a property role is feature refinement, because refinement of a feature must be preceded by identification of the feature. Examples of goals with indirect roles

Table 3 Subtasks types used in TASK model

Has_Type	Has_Parent	Has_Role	Explanation of subtask
Identify-Feature	None	Direct	Present feature in case to be identified and localized
Identify-Absent-Feature	None	Direct	Pertinent negative feature in case to be noted
Identify-Attribute	Identify-Feature	Property	Qualities of feature to be identified only after feature is identified
Assert-Hypothesis	Identify-Feature	Direct	Goals for making hypotheses only instantiated after a feature node already exists which is supportive of this hypothesis
Assert-Diagnosis	Assert-Hypothesis	Direct	Assertion of diagnosis follows assertion of hypothesis
Support-Link	Identify-Feature Assert-hypothesis	Indirect	Reasoning step to indicate that identified feature (constrained or unconstrained by attributes) supportive of particular hypotheses
Refute-Link	Identify-Feature Assert-Hypothesis	Indirect	Reasoning step to indicate that identified feature (constrained or unconstrained by attributes) refutive of particular hypotheses
Identify-Distinguishing-Feature	Assert-Hypothesis Assert-Hypothesis	Indirect	Features to be identified when competing hypotheses exist where features distinguish between competing hypotheses

include (1) supporting and refuting hypotheses, and (2) distinguishing features. Any step, whether direct, indirect or property, may be either required or not required, but it is up to the instructional layer to determine which of these two states is in effect.

By modifying TASK, the goal instances created in the DSG will alter the behavior of the expert model that is used to evaluate student actions. Consequently the same system can be used to model different strategies for solving a single problem, dependent on the state of the student model.

6. Dynamic solution graph

The DSG is a directed acyclic graph that models the current problem state and all valid-next-steps. The DSG generates the initial problem state at the beginning of each problem, using knowledge derived from the domain, task and case models. After each student action, the graph structure is updated by a set of abstract PSMs that may add or delete nodes and arcs, or change the node state. These changes are determined by the behavior encapsulated in the Has_Role and Has_Parent slots of the instantiated goals.

The system does not know the solution of the problem a priori, and only reaches the solution with the student. Although any individual state of the DSG represents only the current problem state and all valid next-steps, sequential DSG states define one path through the problem space—the path taken by the student. The graph representation supports the ability to reason in both directions (design principle 3) and the dynamic nature of the graph enables the system to reason with the student (design principle 2).

Fig. 5A—C depicts the DSG in three different states, corresponding to the states following actions 2.1 (Fig. 5A), 2.15 (Fig. 5B), and 2.17 (Fig. 5C) in the interaction scenario (Table 2). Completed nodes are shown as filled. The current problem state is defined by the set of all completed nodes. All other nodes represent valid-next-steps. Bolded nodes depict calculated best-next-steps.

6.1. Basic structure

The DSG is composed of a set of nodes and a set of arcs. Each node represents one instantiated SUBTASK. Once instantiated, we refer to the nodes as subgoals because they are specific to the context of the problem state. During instantiation, both case and domain knowledge are incorporated. As a result, an individual node in the DSG encapsulates the properties that are essential elements in the achievement of this subgoal (Table 4). For example, in Fig. 5, the node entitled ‘‘nuclear dust’’ is an Identify-Feature node and therefore has properties FeatureName, Area, and Magnification (Table 4). Each property contains a value or set of values. As determined by the instructional layer, a correct action by the student must match all of these node properties. For example, when the student identified nuclear dust in the interaction scenario (Action 2.21 in Table 2), the student’s feature name matched the node FeatureName, the student’s feature location was within the set of regions defined by the Area, and the student’s degree of magnification was greater or equal to the Magnification. Arcs in the DSG represent temporal relationships between the current problem state and valid-next-steps. As the DSG advances, new nodes are added which represent additional valid-next-steps. For example, after the assertion of ‘‘blister’’ the solution graph contains new goals for attributes associated with blister and for hypotheses that are supported by blister (Fig. 5A).

The evidence cluster node is an additional node used to express an integrated relation between features and hypotheses. Cluster nodes are used for disambiguating Feature-Hypothesis relationships as the student traverses the problem space. The method for disambiguation is shown with other graph definitions in Fig. 6. The domain knowledge base contains a one-to-one mapping between FEATURE_SPECIFICATION and DISEASE SET. However, any feature may be present in multiple FEATURE_SPECIFICATION and any DISEASE may be present in multiple DISEASE SETs. Because the DSG reasons only one step ahead of the student, the system must deal with an incomplete set of features in bounding the student’s problem solving. Until all features have been identified, the evidence cluster provides the only way to determine the DISEASE SETs that are valid at any given time. For example, in Fig. 5A—C, as additional features are identified and refined, the evidence cluster points to fewer and fewer hypotheses. At the end of problem-solving only the one-to-one mapping between FEATURE_SPECIFICATION and DISEASE SET will remain. This can be seen in Fig. 5C, where the evidence cluster points to three hypotheses—linear IgA dermatosis, DHLDE Eruption and DH. These three hypotheses share the FS-A feature specification (Fig. 2). In this state, the problem has been solved. Note that in the interaction scenario, the student identifies blister early on, and therefore the set of valid hypotheses is initially quite large. If the student had identified nuclear dust first, the set of valid hypotheses would be small initially, and would grow as additional features were identified.

In addition to bounding the student’s solution, the evidence cluster can be used by the pedagogic

system to permit only hypotheses that are consistent with ALL identified features (hypothesis is a member of the DISEASE_SET in evidence cluster) as opposed to ANY identified features (hypothesis is a child node of Identify-Feature). The current default instructional model implements the latter criterion.

6.2. Creating the graph

To create the initial problem state, information from CASE and TASK are used to instantiate initial DSG nodes. Identify-Feature nodes are instantiated with feature names, areas and magnifica-

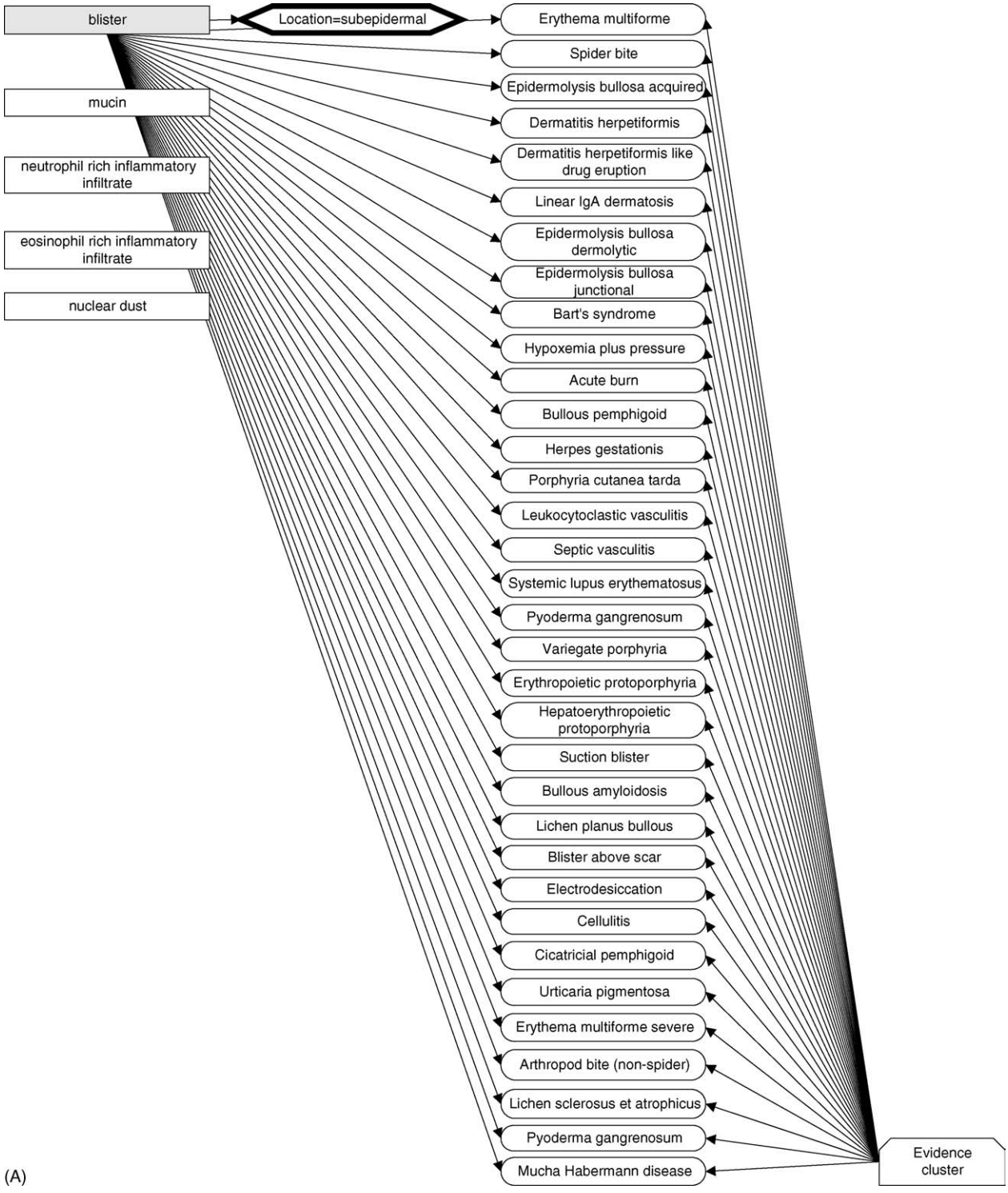

Figure 5 (A) State of DSG following identification of blister (Action 2.1 in Table 2). (B) State of DSG following identification of blister (Action 2.15 in Table 2). (C) State of DSG following identification of blister (Action 2.17 in Table 2).

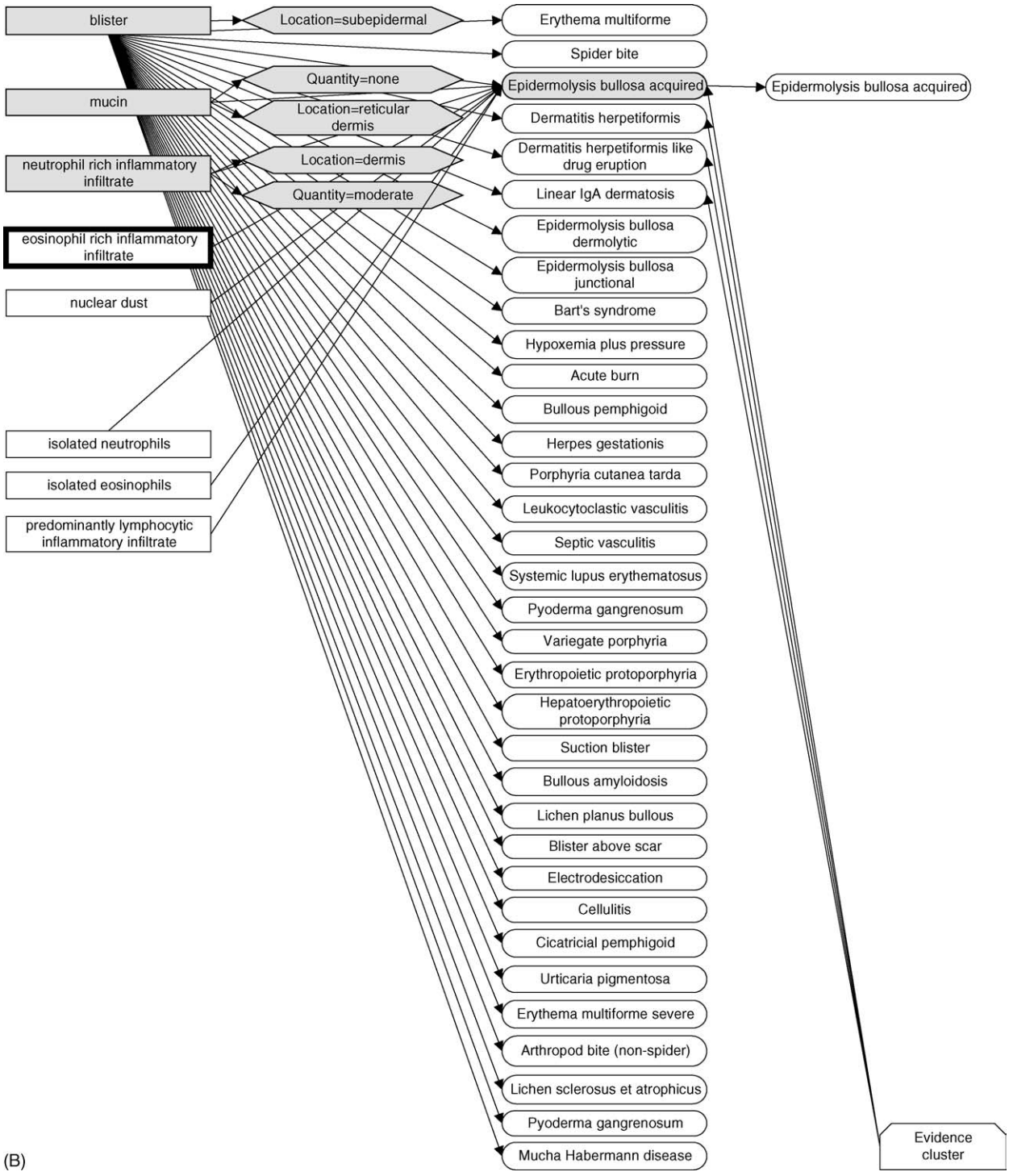

Figure 5. (*Continued*).

tion from CASE. Identify-Attribute nodes are instantiated with attribute name and value. Both nodes contain structural information derived from TASK. At the start of the problem only Identify-Feature and Identify-Attribute nodes are present.

6.3. Determination of best-next-step

As shown in Fig. 5, the DSG includes all valid-next-steps in problem solving (unfilled nodes). In order to help students traverse the problem-space, the DSG must select a single best-next-step for each cycle

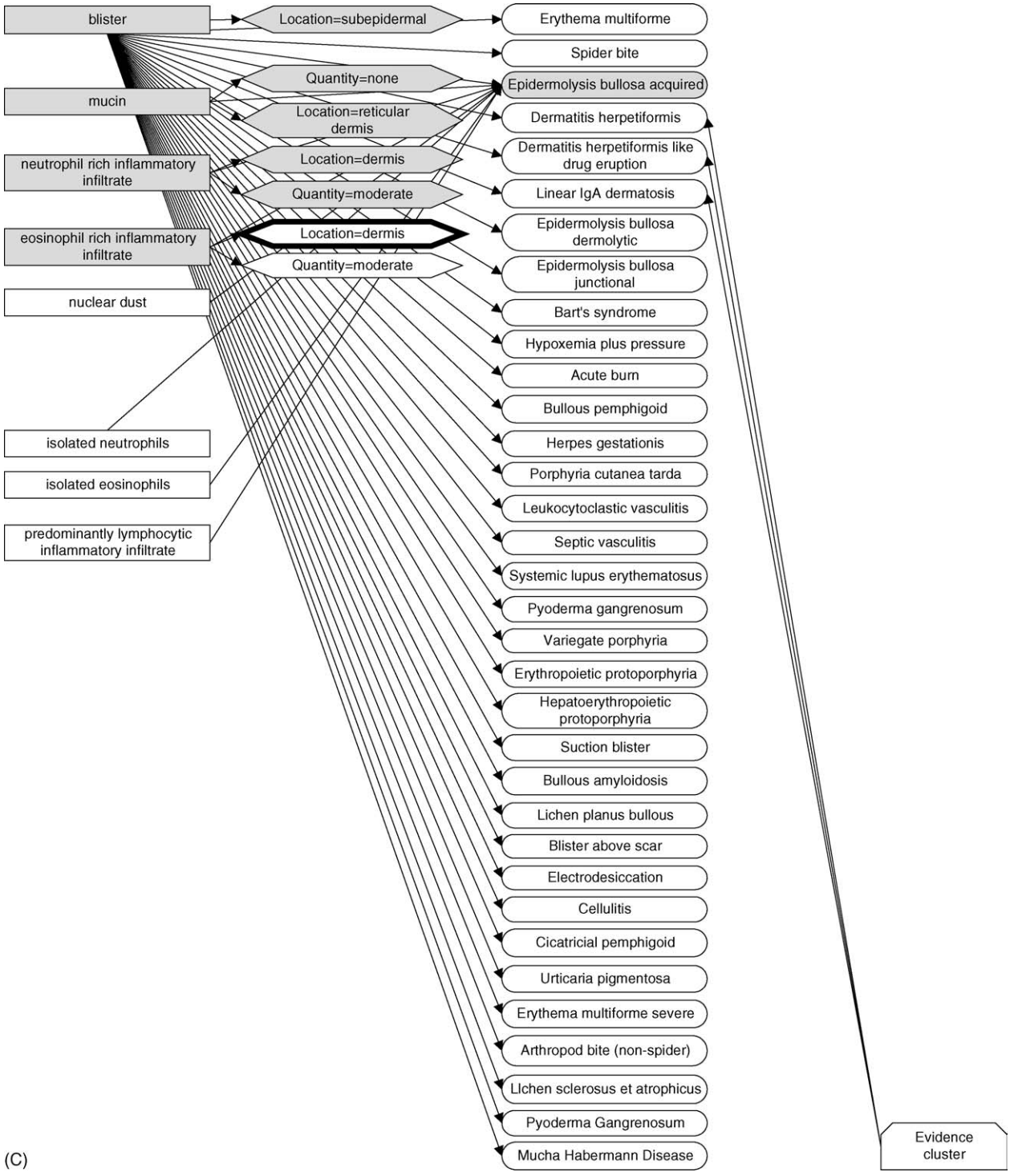

Figure 5. (*Continued*).

(shown as bolded nodes in Fig. 5). These best-next-steps are context-specific to the state of the problem and what is known about the student, and are used by the pedagogic system to deliver context-specific help. For example, in the interaction scenario, the hint given to the student when there is still evidence to identify (response to Action 2.16) is different than the hint given when all evidence has been identified (response to Action 2.23). Identification of best-next-step in the DSG is analogous to conflict-resolution among separate production rules in more traditional ITS architectures.

The best-next-step is determined each cycle based on information derived from the instructional

Table 4 DSG node structure and behavior

DSG node slots	Direct nodes				Indirect nodes		
	Identify-Feature	Identify-Attribute-	Assert-Hypothesis	Assert-Diagnosis	Support-Link	Refute-Link	Identify-Distinguishing-Feature
Is_Goal	Indicates whether the node is a goal node, based on current problem state						
Is_From_Case	Indicates whether the goal is derived from the case						
State	Reflects whether the node has been completed by the student or not						
Properties	Feature name: e.g. 'blister' Area: set of polygons associated with feature Magnification: lowest observable magnification	Attribute name: e.g. 'location' Attribute value: value of attribute derived from case, e.g. 'subepidermal'	Name: e.g. 'linear IgA dermatosis'	Name: e.g. 'linear IgA dermatosis'	Name: e.g. 'blister - linear IgA dermatosis'	Name: e.g. 'nuclear dust-arthropod bite'	Name: e.g. 'nuclear dust'
General behavior of DSG node (applies to all nodes)	Value of state slot changes when completed by student Completed nodes generate all direct parents and children based on contents of has-parent slot of Task model Node cannot be deleted if derived from case Nodes connected to other completed nodes cannot be deleted on update						
Specific behavior of DSG node by subgoal type	Completion of Identify-Feature nodes results in update of evidence cluster based on integration of additional information	AV nodes cannot be completed before parent evidence nodes because Identify-Attribute nodes have property role derived from Task; Completion of Identify-Attribute nodes results in update of evidence cluster based on integration of additional information	On update, diagnosis node of same name as hypothesis node will be created only if hypothesis node Is-goal		All INDIRECT nodes are dynamic and reflect the existing relationship between the DIRECT nodes and can become incorrect when these relationships are no longer valid		

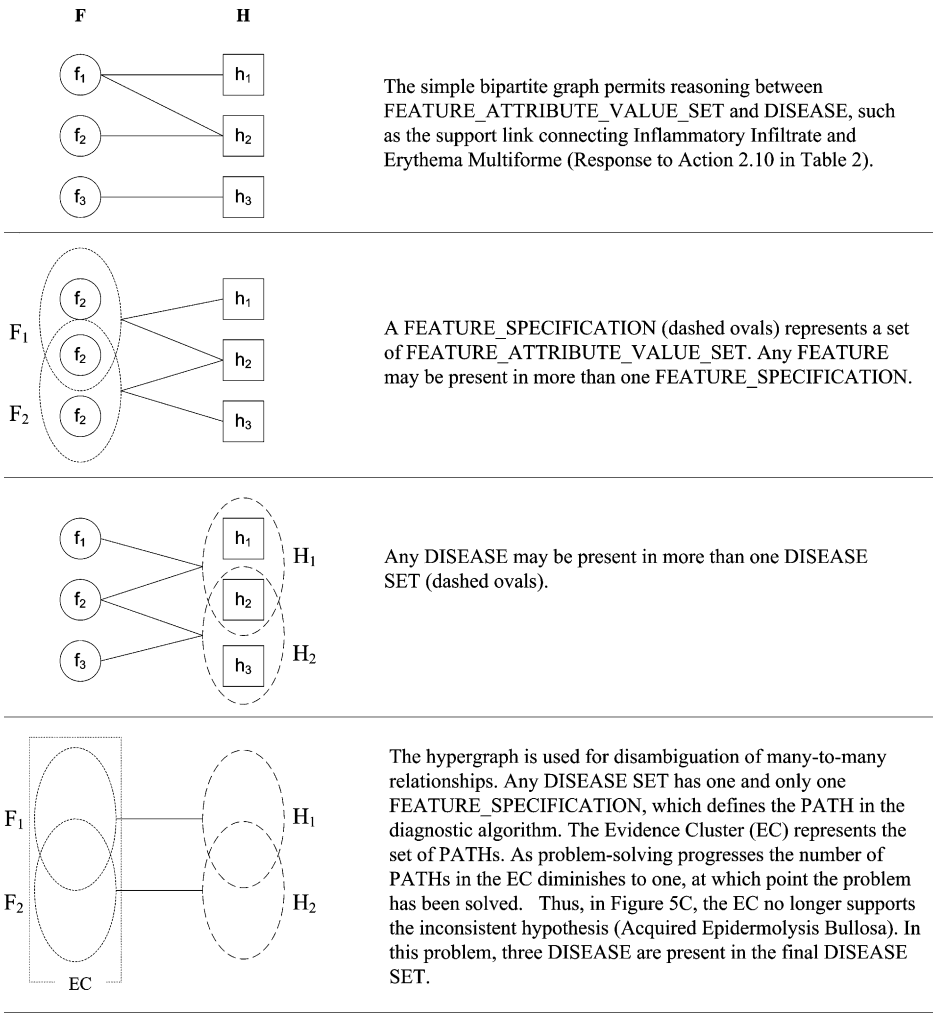

Figure 6 Expert model definitions.

model. Each node contains a value for hint priority and an indicator for whether or not it is a required node. The best-next-step in any state is the required node of highest priority. For example, in Fig. 5A, the best-next-step is to refine the feature blister by adding that it has a subepidermal location, because feature-refinement has a higher priority than feature identification or hypothesis formation for the current pedagogic model. Priorities and requirement indicators may be altered by the pedagogic model, based on the state of the student model.

6.4. Updating the graph

From the initial state, each subsequent student action is translated into an event that propagates through the DSG and alters its structure. When the event propagates, alterations to individual nodes are specific to the type of node (Table 4). For example, after a correct identification of a feature (student action matches a particular Identify-Feature node exactly) the graph updates by (1) changing the state of the Identify-Feature node to identified, (2) adding Assert-Hypothesis nodes which are supported by this feature, (3) creating arcs between the Identify-Feature and each Assert-Hypothesis node supported that feature, (4) updating the evidence cluster, and (5) calculating the new best-next-step.

6.5. Reasoning forward and backward

The DSG models both the forward and backward intermediate reasoning steps that are seen in empirical studies of developing expertise. As demonstrated in the GUIDON project [14,23], forward directed reasoning provides a much more natural method for students to progress through the problem space. However, backward or goal directed reasoning can be a powerful operator and is often used by experts to perform more complex kinds of reasoning. These include checking of

solutions, and distinguishing between alternative hypotheses by seeking evidence that is differentially expressed among the alternatives.

The DSG supports forward directed reasoning, by maintaining the set of all valid-next-steps and calculating the best-next-step with each cycle. Consequently, by iteratively cycling the DSG, it can be used to traverse the entire problem space to arrive at the goal, just as a forward chaining system would be used to arrive at the solution state. Forward-directed reasoning is used by the instructional model to guide the student through step-by-step. In fact, the structure defined by the TASK ensures that as the DSG is successively augmented, the direction of the graph models the forward-directed process from evidence to diagnosis.

If we define the TASK differently, the DSG can be iteratively cycled from a solution state to add goal nodes until an initial problem state is reached. For example, if we define the TASK to reason from diagnosis to evidence, then the graph will recreate the entire problem space from the solution state backward toward all initial states.

However, typically we are only interested in selected aspects of backward reasoning in order to perform particular instructional actions. Therefore we do not recreate the entire backward solution. Instead, the current TASK produces the entire sequence of forward directed reasoning and simultaneously models single backward or goal-directed reasoning steps. In particular, we model backward reasoning from hypothesis to evidence in order to (1) offer more specific kinds of remediation and (2) help students learn how to distinguish among hypotheses.

Remediation can be made more specific by identifying whether a student could be using backward reasoning to set goals for searching for particular features. For example, when a hypothesis is added, goals for finding features and attributes relevant to the added hypothesis appear in the next state of the DSG. Fig. 5B corresponds to the state following Action 2.15 in Table 2, and shows nodes for isolated neutrophils, isolated eosinophils and predominantly lymphocytic inflammatory infiltrate have been added to the DSG, after AEB was asserted. Arcs connect these goals to the AEB hypothesis. In this case, none of these features are actually present in the slide. In other situations, backward reasoning goals may overlap with features present in the case. If in State 5B, the student were to assert the presence of a predominantly lymphocytic inflammatory infiltrate, the system would provide explicit feedback that this feature is not present, but would simultaneously reinforce the student for searching for a feature that should be present, given that EM is under consideration.

Identify-Distinguishing-Feature is an additional subgoal that can be supported by adding nodes for features and attributes, using the backward directed reasoning. Specifically, the structure of the DSG can be used to determine the intersection and complement of features for a given set of hypotheses. For example, linear IgA dermatosis and arthropod bite can be distinguished by the presence of nuclear dust which is present in linear IgA dermatosis but absent in arthropod bite (Fig. 2).

The DSG therefore supports the essential characteristics of a cognitive tutor—it provides the ability to distinguish between correct and incorrect actions, models the set of valid actions for the next step, and selects a single step as the next-best step which can be used by the instructional layer to guide the student through the problem [8]. Furthermore, the DSG supports the foundational principles we established from our developmental model (Table 1): (1) it determines general classes of errors but provides flexibility in tutor response, (2) it reasons with the student supporting intermediate solutions and revision, and (3) it models both forward and backward reasoning. Finally, the DSG enables a more scaleable architecture because all domain and pedagogic knowledge is maintained in separate knowledge bases.

7. Instructional model

Pedagogic and domain knowledge are entirely separate in our system. The DSG is used to determine accepted actions versus failures, and to determine the best-next-step in problem-solving. But all instructional content is provided by a separate system that responds to states in which (1) student actions do not match an existing DSG node or (2) the student requests help.

In symmetry to the expert model, the instructional model is composed of a pedagogic model, pedagogic task and PSMs (Fig. 1). The pedagogic task is rudimentary in the current system consisting only of the goals to deliver hints and alerts, and the priorities attached to these goals. In the default state of the pedagogic task described in this manuscript only a single response (error) or set of responses (hints) is delivered, regardless of the student. In future versions of the system, more complex reasoning will be utilized to produce different hints and alerts based on student model state.

The pedagogic model contains the declarative knowledge required for two types of case-specific interventions: (1) explanations delivered by the

system as ***alerts*** when the student makes a particular kind of error, and (2) explanations delivered by the system as ***hints*** when the student requests help.

7.1. Errors

Errors constitute student actions that generate a 'failure' response from the expert model because they do not fulfill the criteria for a complete match at any node. When this is the case, student actions will match to one or more error states (Table 5). Error states can be categorized as errors of feature identification, feature refinement, hypothesis triggering and hypothesis evaluation, as outlined in our developmental model of expertise (Table 1). Error states for slide search have not yet been implemen-

Table 5 Errors remediated by type

Process	#	Error remediated by tutor
Feature identification	I1	Feature identified by student is not present
	I2	Feature identified by student exists at another location
	I3	Feature identified by student exists elsewhere, but second feature present in this location has been missed
	I4	Feature identified by student is explicitly absent
	I5	Feature identified by student is not present in this case, but can be present for one or more hypotheses under consideration (including correct)
	I6	Feature identified as absent is present in location currently under consideration
	I7	Feature identified as absent is present in another location not currently seen in viewer
	I8	Absent feature identified by student is absent but not important to note
	I9	Magnification used by student is too low to identify absent feature
	I10	Magnification used by student is too low to identify feature
Feature refinement	S1	Feature doesn't require further specification of attributes and values
	S2	Wrong attribute for feature
	S3	Correct attribute for feature, but incorrect value for attribute
	S4	Attribute can have that value for hypotheses under consideration but not in this case (used for backwards reasoning)
	S5	Attribute can have that value for hypotheses not currently under consideration but not in this case (used for forwards reasoning)
Hypothesis triggering	T1	No feature identified to support this hypothesis
Hypothesis evaluation	E1	Feature indicated as supporting hypothesis does not support that hypothesis because feature does not match
	E2	Feature indicated as refuting hypothesis does not support that hypothesis because feature does not match
	E3	Feature indicated as supporting hypothesis does not support that hypothesis because one or more attribute value pairs do not match
	E4	Feature indicated as refuting hypothesis does not support that hypothesis because one or more attribute value pairs do not match
	E5	Feature previously indicative of supporting hypothesis now does not support hypothesis because attribute value pairs have been added
	E6	Feature previously indicative of supporting hypothesis now does not support hypothesis because feature has been further specified within feature hierarchy
	E7	Feature previously indicative of refuting hypothesis now does not refute hypothesis because attribute value pairs have been added
	E8	Hypothesis previously supported by one feature is no longer supported by any feature because attribute value pairs have been added
	E9	Diagnosis does not fit with feature(s) found so far
	E10	Diagnosis fits with some features that have been identified but not other features that have been identified
	E11	Diagnosis now inconsistent with identified feature because new feature added
	E12	Diagnosis now inconsistent with identified feature because new attribute value pairs of feature added
Problem completion	C1	Student indicates problem-done before all required subtasks are completed

ted, primarily because our developmental model suggests that the target users (intermediates) have largely mastered these skills.

Within each category, individual errors describe specific reasons for failure. For example, errors of identification occur when there is no match to an Identify-Feature node in the DSG. Ten specific errors are recognized in this category relating to incorrect present and absent feature assertions, incorrect locations for these assertions, and incorrect viewer magnification when identifying features (Table 5).

When the student action does not match any valid-next-step, a 'failure' response returns from the DSG. A separate set of pedagogic PSMs are used to match the student actions to the error states described in the pedagogic model. For example, in the interaction scenario, when the student asserts mucin (Action 2.5 in Table 2) the DSG returns a 'failure' but the student actions match to Error I4 (Table 5), because an Identify-Feature node exists for absence of mucin, but the student has identified mucin as present. In contrast, when the student identifies epithelial necrosis (Action 2.20 in Table 2), the system matches the student actions to Error I1 (Table 5).

For each error state the pedagogic model maintains a response (alert) that can be delivered to the student. Alerts are composed of context-specific text with accompanying tutor actions. Context-specific text is generated with text templates. A simple markup language is used for insertion of context specific information derived from the current structure of the DSG and/or the incorrect student action. Tutor actions are general kinds of non-text interventions delivered along with the text. They specify the general form of intervention, but do not indicate the specific target object on the interface side. For example, the system may flag an incorrect step in the diagrammatic reasoning interface. Like the text templates, tutor actions are made specific to context and interface object by insertion of values derived from current structure of the DSG. For example, when the student incorrectly identifies mucin as present, the system delivers the text template for Error I4, using the value 'mucin' derived from the DSG node for this feature (Fig. 7). Additionally, the system flags the incorrect node by changing the behavior of the object to blinking, and the color of the object to red.

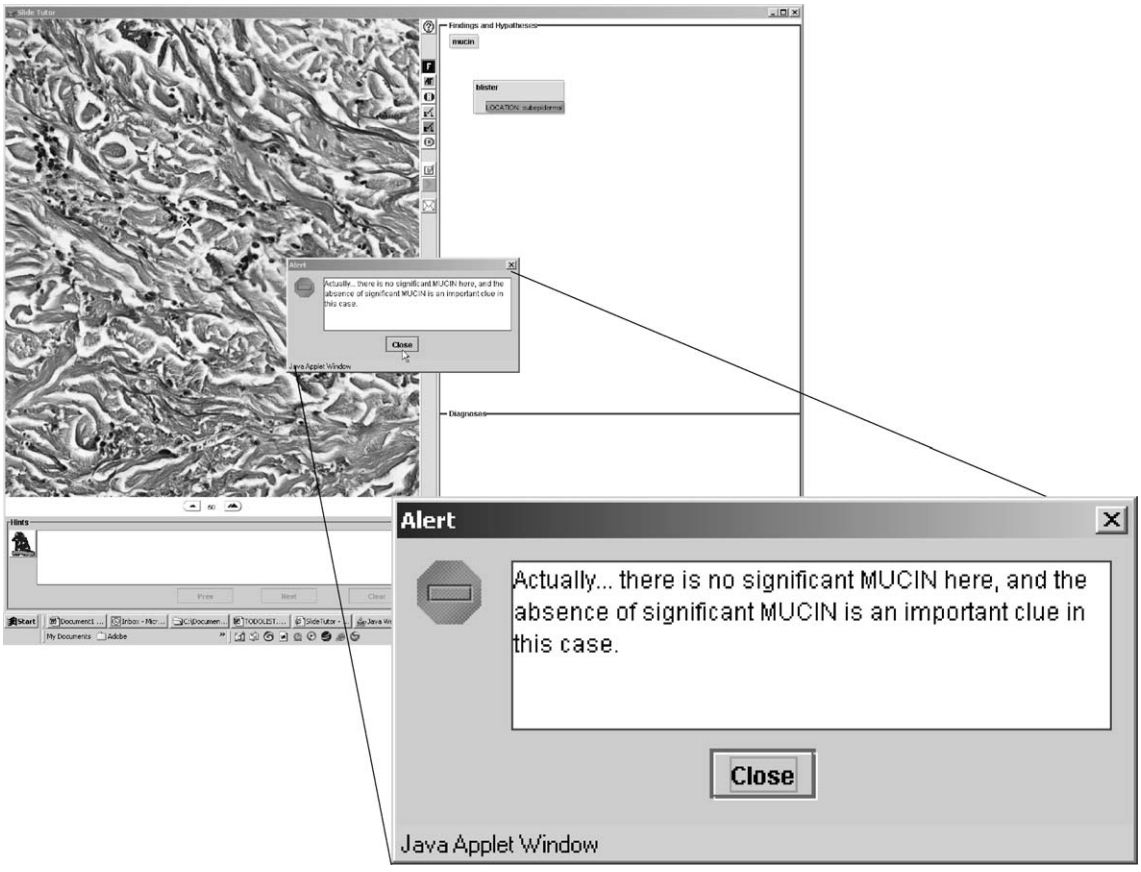

Figure 7 Example alert (response to Action 2.5 in Table 2).

There are three extensions to the basic error logic:

- *Ambiguous error states*. In some cases, student actions match to multiple error states because there is ambiguity about which error the student has made. In this case, the system must determine which alert should be delivered. In the interaction scenario, the student incorrectly identifies eosinophil-rich inflammatory infiltrate (EII) at a particular location (Action 2.17 in Table 2). Eosinophil-rich inflammatory infiltrate is present elsewhere on the slide (Table 5, Error I2). But there is also another feature present in this location (nuclear dust) that has not yet been identified (Table 5, Error I3). Was the student correctly identifying EII as they traversed the slide, but incorrectly bounding the area in which it is seen, or was the student looking specifically at nuclear dust and incorrectly labeling this EII? The system cannot differentiate between these two conditions, but must assign blame to one of them in order to (1) explain the error, and (2) maintain the assessment of student skills needed for the student model. When multiple error states apply, the error state with the highest priority in the pedagogic task is selected by the system. In the case of Action 2.17 (Table 2), Error I2 has higher priority than Error I3, and therefore the I2 alert is delivered. Currently error state priorities are static and reflect the default pedagogic model. In the future, priorities will be dynamic and altered dependent on the state of the student model.
- *Recommendations*. Some kinds of feedback do not fit well into the standard division of feedback as hints and errors. In the interaction scenario, when the student correctly identifies the quantity of EII as moderate, the system accepts the correct response (Action 2.19 in Table 2). Although this is the correct quantity in this case, the student should know that quantity may take a range (including the correct value) and still be consistent with the diagnosis that applies in this case. Additional recommendations are delivered separately to the interface to distinguish them from true errors, but may be concatenated when multiple recommendations apply.
- *Alerts after correct actions*. Because the expert model reasons with the student, some previous assertions may become inconsistent with additional evidence. In particular, this is true of supporting and refuting relationships. In the interaction scenario, when the student further refines neutrophil-rich inflammatory infiltrate, the system accepts the modification (Action 2.12 in Table 2). However, the existing supportive relationship to EM is no longer valid. The instructional layer alerts the student to this new inconsistency. Interactions of this kind are very difficult to construct with traditional cognitive tutor architectures because they typically provide feedback only for a single rule.

7.2. Requests for help

When the student does not know what to do next, they may request help from the tutoring system. Requests for help return a single best-next-step from the expert model. The type of best-next-step returned determines the hint delivered by the instructional layer, and therefore the hint is specific to the current problem state. As described in Section 6, the best-next-step is determined by the expert model. The best-next-step is dependent on (1) the state of the problem, and (2) the pedagogic task.

The state of the problem (Table 6) defines the valid-next-steps from which a best-next-step is selected. The pedagogic task determines whether any valid-next-step is required or not required (see Section 5), and only required valid-next-steps may be returned as best-next-steps. For example, the default state of pedagogic task requires hypotheses. Thus, when all features have been identified and refined (State 5 in Table 6), the system suggests that the student assert hypotheses, because hypothesis is a required node. In the interaction scenario, this interchange is shown in Action 2.23 (Table 2). In future versions of the system, we will use this mechanism to change required nodes as expertise develops, allowing more advanced students to skip steps that were required early on. For example, we can then allow students to ''jump'' to the diagnosis after the region of interest has been seen (see Table 1, Design Requirement 1.7).

For each DSG node type, the pedagogic model maintains a set of responses (hints) that can be delivered to the student. Like alerts, hints are composed of context-specific text and accompanying tutor actions. But unlike alerts, hints are ordered lists, providing increasingly specific advice. Early hints provide general guidance, and later hints are more directive. In the interaction scenario, the student requests help before she has completely identified the evidence (Action 2.16 in Table 2), and progresses through the hints available for this problem state (Fig. 8). At first the system provides only the feature name, shows the student the area of the case in which the feature can be found, and encourages the student to try to find the feature on her own. In subsequent hints, the system draws

Table 6 Problem states and corresponding hint sequences for early intermediate student

State	Problem state description	Hints delivered to early intermediate student
1	Student not in area of interest	Direct student to area of interest
2	Student in area of interest but has not yet started to identify features	Advise student that they have found area of interest and should look for features
3	More features can be identified	Advise student that more features are present Move viewer to area closest to current field that contains feature Draw around area containing feature Provide name of feature Show annotated example of feature (if available) Describe correct interface actions to identify feature
4	Feature has been identified but not fully refined with attribute value pairs	Advise student that feature just identified has important qualities (attribute value pairs) that should be indicated Provide attribute name Provide value name Describe correct interface actions to assert qualities and assist by opening correct menu
5	All features have been identified and fully refined but no hypotheses have been asserted	Advise student that all features have been found and hypotheses should now be considered
6	Hypothesis consistent with all evidence has not been asserted	Provide hypothesis name and supporting evidence Describe correct interface actions to assert hypothesis
7	No remaining hypotheses consistent with all evidence	Advise students that they are ready to make a diagnosis
8	All hypotheses consistent with all evidence have been asserted, but differential diagnosis does not yet contain all members	Provide name of hypothesis to add to differential diagnosis Describe correct interface actions to assert diagnosis
9	All hypotheses consistent with all evidence have been asserted, and differential diagnosis contains all members	Indicate that problem has been solved

around the feature, and then tells the student the exact set of actions to take. These 'hint hierarchies' closely resemble the general structure of hints used in other cognitive tutors [8]. Students can continue to ask for hints until all hints in the hint hierarchy have been displayed.

The pedagogic model defines the hint text and tutor actions that are delivered in specific problem states (Table 6). Hint text is generated with text templates and a simple markup language, for insertion of context specific information, derived from the current structure of the DSG. Tutor actions correspond to general kinds of non-text interventions that are useful in particular problem states. They specify the set of actions to be taken on the interface side. For example, the system may draw around an interface object or open a menu. Tutor actions are made context-specific by insertion of values derived from current structure of the DSG.

There are two extensions to the basic hint logic:

- *Subgoal level and node hints.* The instructional layer distinguishes between states in which the student has or has not previously attempted a particular kind of subgoal. When the student has not previously attempted a particular kind of

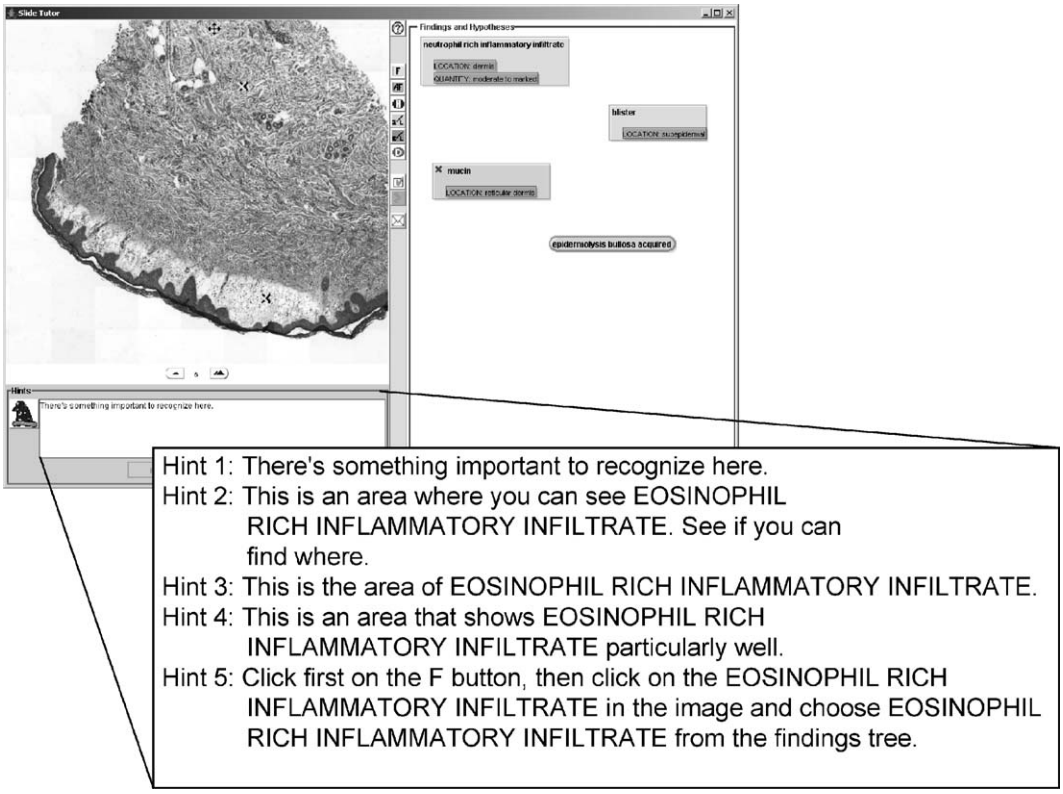

Figure 8 Example hint sequence (response to Action 2.16 in Table 2).

subgoal—there is no existing DSG node of the same node type as the best-next-step. In this condition, the instructional layer delivers a subgoal level hint, aimed at providing guidance about what kind of subgoal to attend to. For example, if the student begins the problem by asking for a hint, the expert model will suggest that the student look for areas that are abnormal, and try to identify features. In contrast, if the problem state contains a completed subgoal of the same node type as the best-next-step, the instructional layer delivers a subgoal node hint, aimed at providing guidance toward completing a given node. The hints shown in Fig. 8 represent a set of subgoal-node hints.

- *Hints after errors*. The pedagogic model maintains a separate set of responses to requests for help that is used when the previous student action generates an error. Action 2.3 in Table 2 shows an example of this type of hint. Unlike other hints, hints after bug contain only a single message and set of tutor actions.

The errors detected and remediated (Table 5), flexibilities permitted (Table 7), and responses to help requests (Table 6), implement the instructional design requirements derived in response to our developmental model of expertise (Table 1).

8. Diagrammatic reasoning palettes for visual classification problem solving

Tutor interfaces combine an image viewer with a VCT diagrammatic reasoning palette used by students to construct a graphical representation of their reasoning. This palette acts as the shared medium between student and system, reifying the process of classification problem solving. Unlike many domains in which ITS have been used [4,6,41], classification problem solving in medicine has no formal notation. Therefore, an important aspect of our work is to create and evaluate possible notations for their effect on learning. One benefit of our architecture is that the separation of the TASK from DOMAIN and CASE, permits significant latitude in the kinds of problem representations that the palettes can support.

For example, the VCT can currently be used with two different palettes to create significantly different interactions with students. The *case-focused palette* presents a local view of the problem, which is fundamentally constructivist. Fig. 9 shows the case-focused palette following Action 2.21 in the interaction scenario. Features and absent features appear as square boxes containing their attribute:-

Table 7 Flexibilities and constraints of ITS—requirements and implementation

ITS flexibility/constraint	Pedagogic design requirement (from Table 1)	Implementation of instructional layer interaction with DSG
Location of features must be indicated	Help students connect visual features to symbolic feature names (1.4)	Student indicated location must match location derived from case and stored in DSG evidence node for feature
Features can be added in any order, but hints provide optimum order specified in diagnostic algorithm	Allows more novice students to fully explore feature space (1.6) but encourages more advanced students to search for features most efficiently (1.8)	Any valid-next-step is allowed but best-next-step follows most efficient sequence
Attributes can be added in any order, but hints suggest refining features immediately after it is asserted	Encourage feature refinement at the time it is asserted to help more novice students learn the salient refinements (1.9)	Any valid-next-step is allowed but best-next-step after feature identification is feature refinement
Hints take students through all features before suggesting hypotheses. But only one supporting feature must be identified before a hypothesis can be asserted and even accepted as diagnosis. If diagnosis is made incorrect by addition of features or refinement of features, then student will later be required to revise	Encourage complete feature articulation among novices (1.5). Permit hypotheses consistent with any one feature (1.12) in order to allow students to explore relationships to other hypotheses (1.17). Allow students to jump to diagnosis as long as region of interest has been seen (1.7). Help more advanced students learn when to exclude hypotheses as features are further refined (1.19)	Best-next-steps complete feature identification before suggesting hypotheses. Hypotheses can only be added to DSG when preceded by one or more supporting features. Evidence cluster must support relationship between all previously asserted features and diagnosis
Features do not need to be refined before hypothesis can be asserted or accepted as the diagnosis	Allow more advanced students to reason without fully refined features (1.10). Permit sequences in which new hypotheses require re-examination of feature refinement (1.11)	Attribute value nodes are not required in default pedagogogic model
All diseases with feature specifications matching the case data must be asserted as hypothesis and diagnoses	Encourage hypotheses that are consistent with all of features (1.14), and help students learn sets of hypotheses that share similar features (1.15)	Problem is not complete until all Diseases matching the FEATURE SPECIFICATION are asserted

value pairs. Hypotheses appear as separate rounded boxes, and may be connected to features using support and refute links. Hypotheses may be moved into the Diagnoses area of the palette when a diagnosis can be made (dependent on the state of the DSG and the student model). Only the features present in the actual case are represented, but any valid hypothesis can be added and tested. At the end of each case, the diagram shows the relationships present in a single case. These diagrams will be different for each case.

In contrast, the *knowledge-focused palette* (Fig. 10) presents a global view of the problem. The interface is algorithmic. Students see the diagnostic tree unfold as they work through the problem. Fig. 10 shows the knowledge-focused palette following Action 2.21 in the interactions scenario (same problem state as Fig. 9). Features and absent features appear as square boxes containing their attribute:value pairs. After attributes and values have been correctly added by the student, they are iconized as diamonds, but may be moused over to display their meaning. As features are added, they are connected to form a path toward the diagnoses. When students complete any level of the algorithm by correctly identifying and refining the feature, the tutor reifies all of the other possible choices at that level. The current path (all identified features) is shown in yellow to differentiate it from other paths. Hypotheses appear as separate

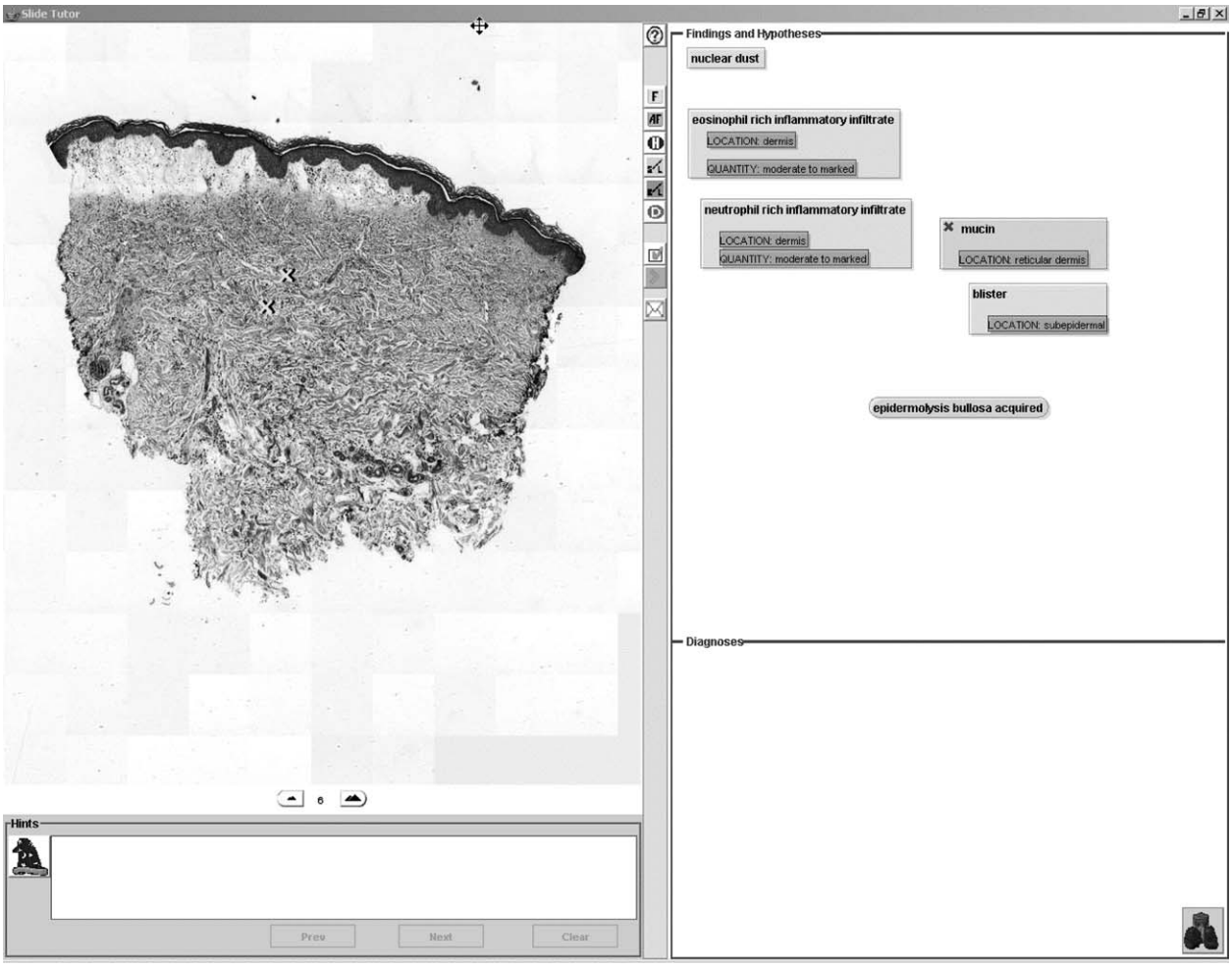

Figure 9 Case-focused diagrammatic reasoning palette (state following Action 2.21 in Table 2).

rounded boxes. When students make a hypothesis, the tutor places the hypothesis in the appropriate position on the diagnostic tree. When the hypothesis fits with the current evidence it is shown connected to the current path. When the hypothesis does not fit with the current evidence, it is shown connected to other paths with the content of the associated features and attributes hidden as boxes containing '?' until students specifically request the identity of the feature or attribute. A pointer is always present to provide a cue to the best-next-step. In the problem state shown in Fig. 10, the pointer on the right-hand side of the diagram shows a rounded box containing '?H', because all of the features have been identified and the best-next-step at this state is to make a hypothesis. The student can see that AEB can have five different FEATURE_SPECIFICATIONS (creating five different paths), including some that share features with the current case (blister, eosinophil-rich inflammatory infiltrate, neutrophil-rich inflammatory infiltrate). The student can also determine from the diagram that Acquired Epidermolysis Bullosa is not the diagnosis in the current case. By the conclusion of problem solving the entire diagnostic tree is available for exploration. The knowledge-focused palette therefore expresses relationships between features and hypotheses both within and across cases. Students can use the tree to compare between cases. At the end of each case, the diagram shows the same algorithm, but highlights the FEATURE_SPECIFICATION of the current case.

9. Advantages of the approach

The approach we describe is novel, and has significant advantages for designing medical tutoring systems. The advantages include:

- *Use of a paradigm with proven effectiveness.* An important aspect of this project is to test the feasibility of an established instructional method

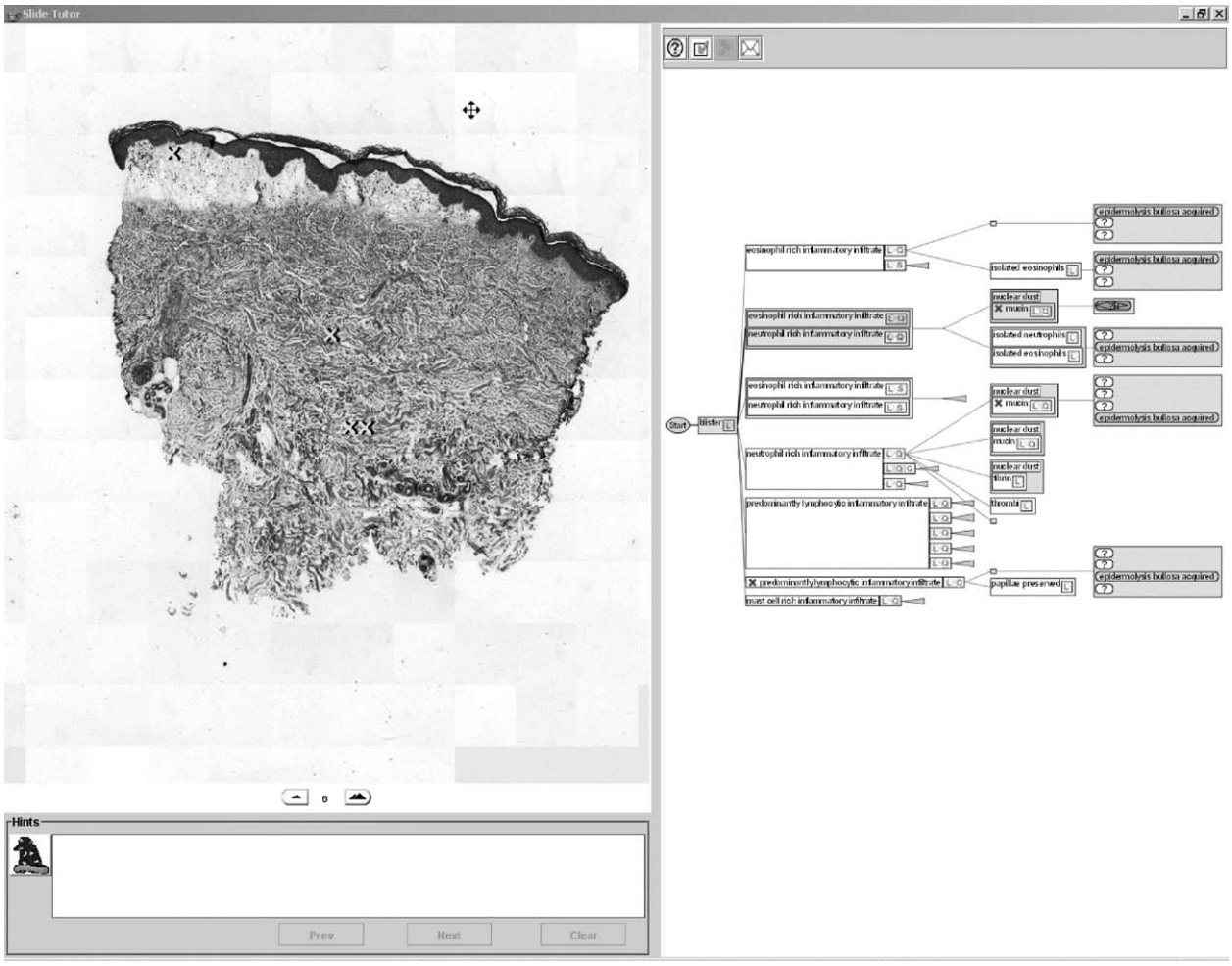

Figure 10 Knowledge-focused diagrammatic reasoning palette (state following Action 2.21 in Table 2).

in a declarative-knowledge rich domain. One advantage of the architecture that we propose is that it closely reproduces the intermediate feedback and goal-structuring of MTITS, a class of systems that have proven efficacy in other domains [3—6].

- *Scalability, ease of maintenance, and reusability.* The use of separate domain model and pedagogic model enhances scalability of the system because new domains or pedagogic content can be added by extending the knowledge bases in an ontology editing environment, without altering the code. The VCT domain model extends an existing ontology for classification problem solving [42], and therefore any other ontology that also uses this representation can be easily incorporated. The separate domain model is also reused to constrain case authoring.
- *Support for forward and backward reasoning*. The separation of domain task from domain model enhances the flexibility of the system to support more than one kind of student reasoning. The DSG can be constructed to use forward reasoning, backward reasoning, or a combination of both by minor alterations to the task model.
- *Flexibility of the instructional layer.* The separation of the instructional system from the expert model enhances instructional flexibility and individualization. Unlike most cognitive tutors, the instructional responses (hints and errors) are not additional clauses to an expert model rule. Rather, they are separate sets of declarative elements that are mapped to general problem states or error types. Instructional responses are easily changed both by the author when the instructional content is created, and by the system in choosing a particular response based on the state of the student model. Because the DSG maintains the problem-state as the solution advances, the instructional layer can use even more complex methods for individualization of instruction. For example, the typical immediate feedback cycle in which each student action is answered by a tutor response, can be changed in

our system. The model could be permitted to advance over many student actions and potentially off the correct solution path for multiple steps. In this case the instructional layer could provide feedback at longer intervals, gradually fading the instructional scaffolding as students gain expertise.

10. Implementation of the VCT

The VCT is implemented as a client—server system, with a multi-agent architecture, written in the Java programming language. Agents communicate using Java Agent Services—a library that supports the Foundation for Intelligent Physical Agents (FIPA) standard [45]. The multi-component nature of the system permits replacement of individual components to create different domain specific tutoring systems.

10.1. Server

The server includes the Tutor Engine, Xippix Image Delivery System [46], and Protocol Collection System for obtaining and storing detailed information regarding user—system interaction.

10.1.1. Tutor engine

All abstract PSMs are implemented in Jess—a Java production rule system [47,48]. Ontology classes for the VCT are created in Protégé-2000 [49,50]. Case information acquired through an authoring system (Section 12) is stored in Protégé. Domain and case Protégé projects share a third project containing the FEATURE, ATTRIBUTE, and VALUE primitives (Fig. 3). All Protégé classes and instances are converted to Jess templates and facts using a modification of JessTab [51]. The DSG exists as a set of Jess facts, which are utilized by the abstract PSMs of the instructional layer. For development purposes, we utilize an extension of JGraph [52] to visualize the DSG in Jess Working Memory. Hint and error message templates are stored in Protégé, and can be re-used or modified for different tutoring systems. Instances are created using a markup language that makes general templates context specific. Instances may include pointers to interface objects, and actions to be performed alone or in combination with text feedback.

10.1.2. Image delivery system

The image delivery system [46] is a commercial application that permits large image files to be delivered in smaller pieces for zooming and panning to a Java image viewer.

10.1.3. Protocol collection system

Inter-agent messages containing time-stamped, low-level interface actions (such as image navigation and button presses), complete student actions (such as feature identification or hypothesis creation), and tutor responses are collected and stored in an Oracle 9i database for further analysis.

10.2. Client

The VCT student client is a Java WebStart [53] application that communicates with the server via HTTP. The student client is implemented as a framework that can support a range of domain specific interface elements. The client includes a Java image viewer that communicates directly with the Xippix server [46] to update the image, based on user requests. The client image viewer acts as a virtual microscope. Diagrammatic reasoning interfaces use SpaceTree [54,55] for tree-based selection (both interfaces) and algorithm representation (knowledge-focused interface).

11. SlideTutor—an instantiation of the VCT

Using the VCT—we have created several tutoring systems that employ different domain knowledge bases and different interfaces. SlideTutor is one tutoring system created for the domain of dermatopathology. The system is our test bed for evaluating the effects of the VCT on learning. SlideTutor is designed for use by pathology and dermatology residents and fellows. More than fifty users have participated in formative evaluations of the system to date. SlideTutor was created by the addition of domain knowledge, and annotated, digitized microscopic slides to the VCT framework. Student interfaces are tutor-specific after addition of these components.

Replacing the domain knowledge base alters the domain of the tutoring system, and creates different diagrams and menus in the resulting palette. DinoTutor is a visual classification tutoring system that uses a knowledge base for visual classification of dinosaurs. It is used for interface training of participants in our formative evaluations. Like SlideTutor it can be used with both case-focused and knowledge-focused palettes. Similarly, replacing the pedagogic knowledge base alters the feedback and action sequences that the tutor provides in response to student errors and requests for help.

It is also certainly possible to create tutoring systems that differ on the basis of pedagogic knowl-

edge bases although we have not yet done this. The modular architecture we have chosen will ease the addition of adaptive changes to the instructional layer in response to the student modeling system, when this component is added.

11.1. Domain knowledge base

The previously described class structure for classification problem solving was instantiated in the domain of inflammatory diseases of skin. The current knowledge base consists of a subclass of lesions called subepidermal vesicular dermatitides, which includes 33 different diseases containing 50 different FEATURE_SPECIFICATIONs, and represents an estimated 10% of the entire set of inflammatory diseases of skin. Complete instantiation of the knowledge base will be the subject of future work. Knowledge acquisition starts with published diagnostic algorithms for classification of inflammatory diseases [56] that are further disambiguated by a knowledge engineer and domain expert in dermatopathology, using an iterative, interview-based method. The information is encoded in Protégé by the knowledge engineer and then validated by the domain expert.

11.2. Cases

Sixty glass slides representing diseases in the knowledge base were obtained from the archives of five academic pathology departments, digitized using the Aperio slide scanner [57], converted to pyramidal tiff format, and annotated using the VCT authoring system for inclusion in SlideTutor.

12. Authoring cases for VCT

Tutor cases are developed using a general authoring system that allows us to create the case as a Protégé project describing features, attributes, and values, as well as the locations in which they are found on the virtual slide. The authoring system is a Protégé plugin that uses the same commercial virtual slide system as the ITS. The classes used are described in Section 5. The authoring system partially constrains choices by authors. To author a newly acquired case, the user starts by indicating the diagnosis from the set of diagnoses currently in the domain knowledge base (Fig. 11A). The system displays all possible FEATURE_SPECIFICATIONs and the author selects the FEATURE_SPECIFICATION from the knowledge base that represents the current case (Fig. 11B). The authoring system then automatically creates instances of ATTRIBUTE_VALUE_PAIR and OBSERVABLE. When authors draw around particular visual features, they choose the correct OBSERVABLE from the list of those associated with the FEATURE_SPECIFICATION (Fig. 11C). The OBSERVABLE is saved in combination with the annotated shape as a LOCATED_OBSERVABLE. Additionally, authors may annotate particularly salient areas demonstrating a feature, which will be used in hints for feature identification (see Fig. 8).

Constraints imposed by the authoring plugin limit errors in authoring that produce unintended behavior in the tutor, but also ensure consistency among the cases. Only cases with FEATURE_SPECIFICATIONs deemed pedagogically relevant — either classic examples, or common exceptions — are easily added. The authoring plugin checks consistency by displaying the implied feature specification, which represents the solution that the student and tutoring system will reach by the conclusion of problem-solving in the ITS. If no FEATURE_SPECIFICATION is indicated, the tutoring system will not be able to classify the case fully. Authors are in fact free to produce cases with sets of LOCATED_OBSERVABLEs that do not match any FEATURE_SPECIFICATION in the knowledge base, but must manually create the needed instances. In these cases, the tutoring system will lead students to a set of diagnoses that fit this evidence. In some cases, this may be a null set.

13. Discussion and conclusions

Intelligent tutoring systems have significant advantages over existing training methods for medical training. They provide a simulated environment, in which students can practice without consequence to real patients. Within this simulated environment, the pedagogic system offers constant feedback and help aimed at efficiently bringing students to mastery. By constantly monitoring and maintaining a representation of how the student is progressing, the system can adapt to provide individualized training. Despite the significant potential for ITS in medical training, very few systems have been developed.

We have advanced a general method for intelligent tutoring of visual classification problem solving, and used this system to create SlideTutor—an ITS for diagnostic classification in dermatopathology. The VCT recreates the essential characteristics of cognitive tutoring systems, fulfills the unique requirements for tutoring of classification problem solving, and utilizes an architecture that scales by

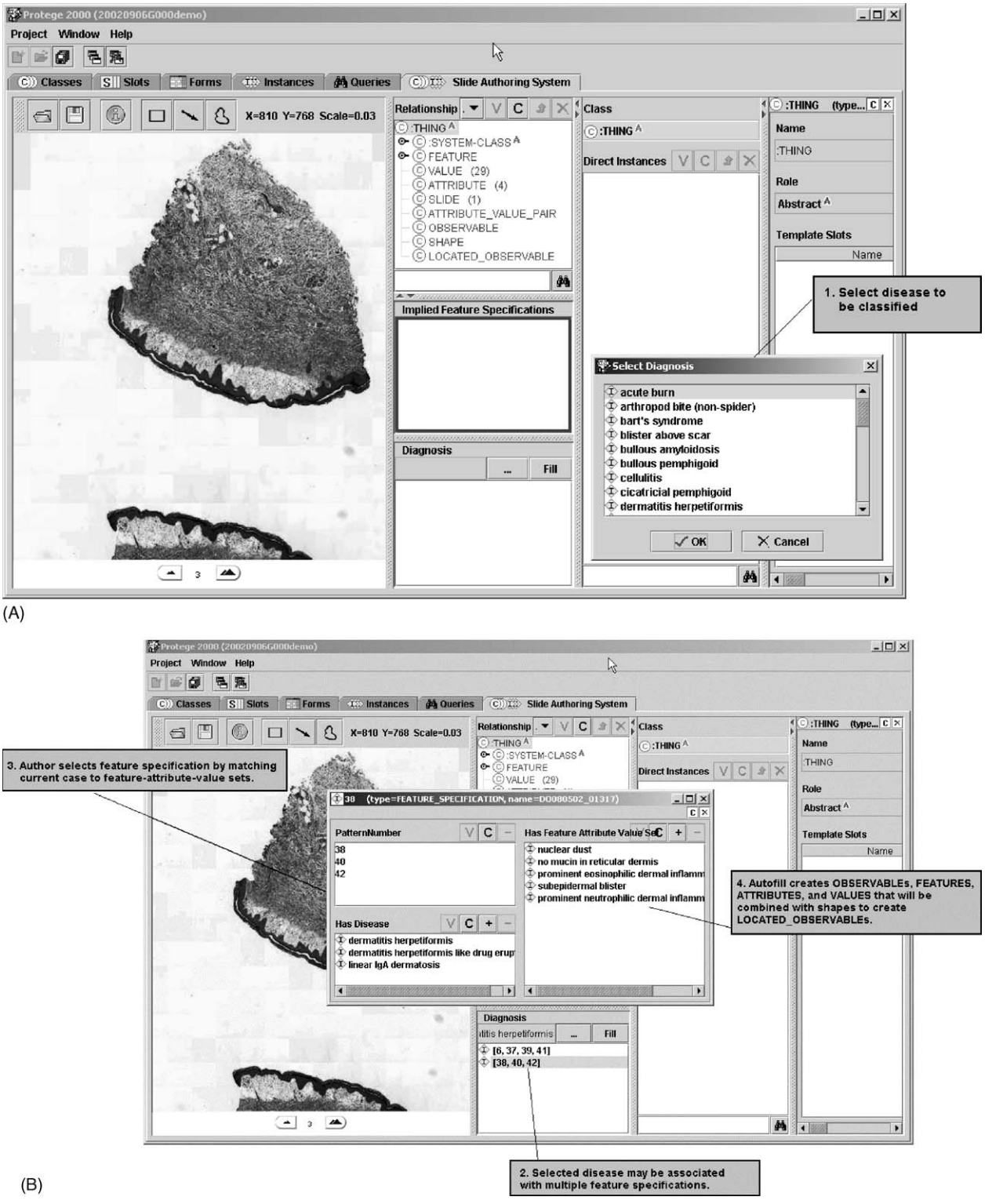

Figure 11 (A)–(C) Authoring system interface.

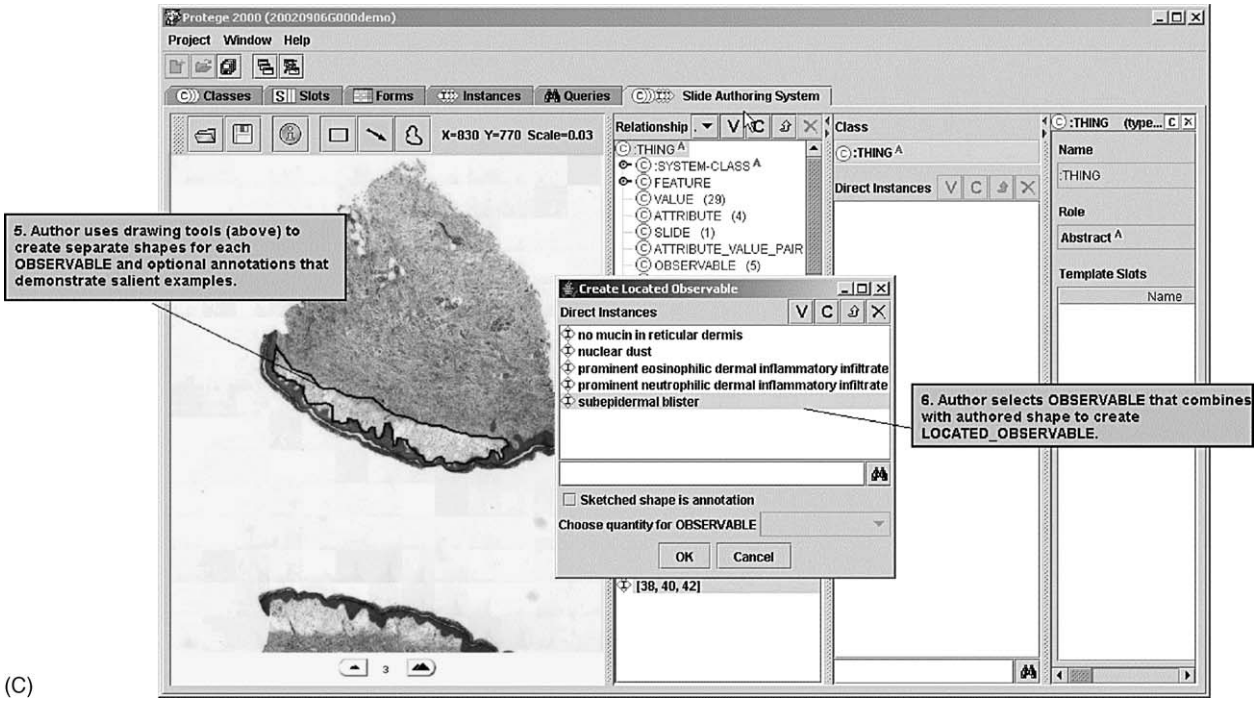

Figure 11. (*Continued*).

expanding the domain knowledge base. The system is built on an underlying developmental cognitive model of expertise, which has guided design choices throughout the project.

Our architecture shares many characteristics of GUIDON, including the complete separation of domain and pedagogic knowledge—although this is accomplished by a different method. GUIDON generated a static AND/OR tree at the beginning of problem solving from the production rules that modeled expertise in a particular domain. In our case, we generate a graph using a set of abstract PSMs where the structure of the graph updates with student actions. Domain knowledge is represented in separate knowledge bases. A second difference from GUIDON is that the DSG can support both forwards and backwards reasoning simultaneously.

The VCT also shares some similarities with existing cognitive tutors. Like these systems, the VCT discriminates correct from incorrect intermediate actions, identifies particular kinds of student errors, and provides hints by stepping forward in the expert model. Unlike most existing cognitive tutors, the pedagogic model is entirely separated from the expert model — enabling us to easily tailor pedagogic interventions (for example by providing different hints to different kinds of students) but also to substitute an entirely different method of interacting with the student — for example an instructional layer that annotates and critiques a complete trace of student actions rather than providing immediate feedback with each action.

The modular architecture we have chosen has significant advantages for our project because it enables parametric evaluation of particular components against possible alternatives. Currently we are performing a study of students randomized to one of the two interface conditions described above, in which we measure performance before and after tutoring. In future work, we will be comparing student modeling systems for their predictive power and evaluating the immediate feedback of model-tracing versus a delayed feedback instructional layer. In each case, we are able to replace one component in the architecture (interface, student modeling system, or instructional layer) without significant alteration to other components. The ability to evaluate our system under a number of different conditions supports iterative improvement, but also enables us to test more general theories about teaching and learning in these systems.

There are, however, important limitations to our approach. The most significant limitation is that our system does not model or provide feedback on probabilistic relationships between evidence and hypotheses. Our approach requires a limited set of combinations of features and attributes (the evidence). This set includes the most frequent combinations, and those that represent common exceptions. There is no attempt to model all combinations

of evidence, or to deal with incomplete evidence. The DSG provides a highly deterministic response, which permits no shades of gray in the pedagogic feedback. This approach is intended to help students quickly build relevant schemas for canonical examples and common exceptions [58]. We hypothesize that this approach will more rapidly move students towards a level of expertise in which likelihood information could be incorporated, as students encounter more complex cases beyond the confines of the tutoring system. But clearly an important aspect of medical reasoning — the ability to reason under uncertainty — is not addressed in our current system.

The VCT is based on an information processing theory of cognition. How far can the information processing approach be taken towards incorporating aspects of reasoning under uncertainty? Normative approaches to the study of decision making under uncertainty have shown that subjective assessments of probabilities, even among experts, often do not conform to expectations based on probabilistic models [59,60]. Availability, representativeness and anchor adjustment are well-known heuristics, which generate systematic errors [61]. Interestingly, some investigators have shown that exposure to case sets in which base rates and conditional probabilities are manifest results in decisions that closely correspond to Bayes' rule [62]. A more formal understanding of how experts manage to capture the probabilistic information inherent in a set of cases, and particularly how these skills develop, would be of significant value in developing training systems. An ITS that incorporates into its developmental cognitive model a more complete understanding of these processes might provide a richer and more effective instructional experience for students.

Acknowledgements

The authors wish to thank Elizabeth Legowski, Eugene Tseytlin, Girish Chavan, Drazen Jukic, Charles Freidman and Maria Bond for their help with this project. The work reported in this manuscript was supported by grants from the National Library of Medicine (LM007891), and the Competitive Medical Research Fund, Office of Research, Health Sciences, at the University of Pittsburgh. This work was conducted using the Protégé resource, which is supported by grant LM007885 from the United States National Library of Medicine. SpaceTree was provided in collaboration with the Human—Computer Interaction Lab (HCIL) at the University of Maryland, College Park. A preliminary description of this architecture was reported in Proceedings of the AMIA Fall Symposium, 2003.

References

[1] Middleton B, Detmer W, Musen M. Diagnostic decision support. In: Osheroff J, editor. Computers in clinical practice: managing patients, information and communication. Philadelphia: American College of Physicians; 1995. p. 59—75.
[2] Lillehaug S, Lajoie S. AI in medical education—another grand challenge for medical informatics. Artif Intell Med 1998;12:197—225.
[3] Koedinger K, Anderson J, Hadley W, Mark M. Intelligent tutoring goes to school in the big city. Int J Artif Intell Educ 1997;8:30—43.
[4] Koedinger K, Anderson J. Effective use of intelligent software in high school math classrooms. In: Brna P, Ohlsson S, Pan H, editors. Proceedings of the world conference on artificial intelligence in education. Charlottesville, VA: AACE; 1993. p. 241—8.
[5] Lesgold A, Lajoie S, Bunzo M, Eggan G. Sherlock: a coached practice environment for an electronics troubleshooting job. In: Computer assisted instruction and intelligent tutoring systems: establishing communication and collaboration. Hillsdale, NJ: Erlbaum; 1993. p. 201—38.
[6] Gertner A, VanLehn K. Andes: a coached problem solving environment for physics. In: Gauthier G, Frasson C, VanLehn K, editors. Proceedings of the fifth international conference on intelligent tutoring systems (ITS-2000). Berlin: Springer; 2000. p. 131—42.
[7] Anderson J, Corbett A, Koedinger K, Pelletier R. Cognitive tutors: lessons learned. J Learn Sci 1995;4(2):167—207.
[8] Anderson J. Rules of the mind. Hillsdale, NJ: Erlbaum; 1993.
[9] Lovett M. Cognitive task analysis in service of ITS design. Intell Tutoring Syst 1998;1452:234—43.
[10] Mitchell C. Horizons in pilot training: desktop tutoring systems. In: Sarter N, Amalbert R, editors. Cognitive engineering in the aviation domain. Mahwah, NJ: Erlbaum; 2000.
[11] Bloom B. The 2 sigma problem: the search for methods of group instruction as effective as one-to-one tutoring. Educ Res 1984;13(3—16).
[12] Kulik J, Kulik C, Cohen P. Effectiveness of computer-based college teaching: a meta-analysis of findings. Rev Educ Res 1980;50:525—44.
[13] Kulik C, Kulik J. Effectiveness of computer-based instruction: an updated analysis. Comput Human Behav 1991;7:75—95.
[14] Clancey W. Guidon. J Computer-based Instruct 1983;10:8—14.
[15] Eliot C, Williams K, Woolf B. An intelligent learning environment for advanced cardiac life support. In: Proceedings of the AMIA annual fall symposium; 1996. p. 7—11.
[16] Evens M, Brandle S, Change R, Freedman R, Glass M, Lee Y, et al. CIRSIM-Tutor: an intelligent tutoring system using natural language dialogue. In: Proceedings of the 12th Midwest AI and cognitive science conference (MAICS); 2001. p. 16—23.
[17] Azevedo R, Lajoie S. The cognitive basis for the design of a mammography interpretation tutor. Int J Artif Intell Educ 1998;9:32—44.
[18] Sharples M, Jeffery N, du Boulay B, Teather B, Teather D, du Boulay G. Structured computer-based training in the interpretation of neuroradiological images. Int J Med Inform 2000;60:263—80.

[19] Smith P, Obradovich J, Heintz P, Guerlain S, Rudmann S, Strohm P, et al. Successful use of an expert system to teach diagnostic reasoning for antibody identification. In: Proceedings of the fourth international conference on intelligent tutoring systems; 1998. p. 354–63.
[20] Clancey W. Knowledge-based tutoring—the GUIDON program. Cambridge, MA: MIT Press; 1987.
[21] Clancey W. Heuristic classification. Artif Intell 1993;27:289–350.
[22] Clancey W. Methodology for building an intelligent tutoring system. In: Kintsch W, Miller H, Poison P, editors. Methods and tactics in cognitive science. Hillsdale, NJ: Erlbaum; 1984.
[23] Clancey W, Letsinger R. NEOMYCIN: reconfiguring a rule-based expert system for application to teaching. In: Proceedings of the seventh international joint conference on AI; 1981. p. 829–35.
[24] Aikins J, Kunz J, Shortliffe E, Fallat R. PUFF: an expert system for interpretation of pulmonary function data. Comput Biomed Res 1983;16:199–208.
[25] Bennett J, Engelmore R. SACON: a knowledge-based consultant for structural analysis. In: Proceedings of the IJCA; 1979. p. 47–9.
[26] Musen M. Modern architectures for intelligent systems: reusable ontologies and problem-solving methods. In: Chute C, editor. Proceedings of the 1998 AMIA annual symposium. 1998. p. 46–52.
[27] Gruber T. Towards principles for the design of ontologies used for knowledge sharing. Stanford University Knowledge Systems Laboratory; 1993. Report nr. KSL93-04.
[28] Heffernan N, Koedinger K. An intelligent tutoring system incorporating a model of an experienced human tutor. In: Proceedings of the international conference on intelligent tutoring system. Springer; 2002.
[29] Fensel D, Motta E, van Harmelen F, Benjamins V, Crubezy M, Decker S, et al. The unified problem-solving method development language. Knowledge Inform Syst 2003;5(1):83–131.
[30] Fensel D, Benjamins V, Decker S, et al. The component model of UPML in a nutshell. In: Proceedings of the first working IFIP conference on software architecture (WICSA1). Kluwer; 1999.
[31] Lesgold A, Rubinson H, Feltovich P, Glaser R, Klopfer D, Wang Y. Expertise in a complex skill: diagnosing X-ray pictures. In: Chi M, Glaser R, Farr M, editors. The nature of expertise. Hillsdale, NJ: Erblaum; 1988. p. 311–42.
[32] Nodine C, Kundel H, Lauver S, et al. Nature of expertise in searching mammograms for breast masses. Acad Radiol 1996;3:1000–6.
[33] Norman G, Brooks L, Allen S, Rosenthal D. Sources of observer variation in dermatologic diagnosis. Acad Med 1990; 65(Suppl):S19–20.
[34] Arocha J, Patel V. Novice diagnostic reasoning in medicine: accounting for evidence. J Learn Sci 1995;4(4):355–84.
[35] Allen V, Arocha J, Patel V. Evaluating evidence against diagnostic hypotheses in clinical decision making by students, residents and physicians. Int J Med Inform 1998;51:91–105.
[36] Kushnirik A, Patel V, Marley A. Small worlds and medical expertise: implications for medical cognition and knowledge engineering. Int J Med Inform 1998;49:255–71.
[37] Joseph G-M, Patel V. Domain knowledge and hypothesis generation in diagnostic reasoning. Med Decis Mak 1990;10:31–46.
[38] Patel V, Groen G. The general and specific nature of medical expertise: a critical look. In: Ericsson K, Smith J, editors. Toward a general theory of expertise: prospects and limits. New York: Cambridge University Press; 1991. p. 93–125.
[39] Crowley R, Naus G, Stewart J, Friedman C. Development of visual diagnostic expertise in pathology—an information processing study. J Am Med Inform Assoc 2003;10(1): 39–51.
[40] Crowley R. Expertise in microscopic diagnosis: an information processing study. Master's Thesis. University of Pittsburgh School of Information Science; 2001.
[41] Wenger E. Artificial intelligence and tutoring systems—computational and cognitive approaches to the communication of knowledge. Los Altos, CA: Morgan Kaufmann Publishers; 1987.
[42] Motta E, Lu W. A library of components for Classification Problem solving. IBROW Project Deliverable D1-IST1999; 1999.
[43] Abel M, Silva L, Mastella L, Campbell J, De Ros L. Visual knowledge modeling and related interpretation problem-solving methods. In: Conferencia Latinoamericana de informatica 2002. Montevideu, Uruguay: CLEI; 2002.
[44] Evans D, Gadd C. Managing coherence and context in medical problem-solving discourse. In: Evans D, Patel V, editors. Cognitive science in medicine: biomedical modeling. Cambridge, MA: MIT Press; 1989. p. 211–55.
[45] http://www.fipa.org/ (accessed: 31 December 2004).
[46] http://www.twocats.com/ (accessed: 31 December 2004).
[47] Friedman-Hill E. JESS, The Java Expert System Shell. SAND 1997; 98-08206 (November).
[48] http://herzberg.ca.sandia.gov/jess/ (accessed: 31 December 2004).
[49] Musen M, Fergerson R, Grosso W, Noy N, Crubezy M, Gennari J. Component-based support for building knowledge-acquisition systems. In: Proceedings of the conference on intelligent information processing (IIP 2000) of the international federation for information processing world computer congress (WCC 2000); 2000.
[50] http://protege.stanford.edu/ (accessed: 31 December 2004).
[51] http://www.ida.liu.se/∼her/JessTab/ (accessed: 31 December 2004).
[52] http://www.jgraph.com/ (accessed: 31 December 2004).
[53] http://java.sun.com/products/javawebstart/ (accessed: 31 December 2004).
[54] Grosjean J, Plaisant C, Bederson B. SpaceTree: Supporting Exploration in Large Node Link Tree, Design Evolution and Empirical Evaluation. HCIL-2002-05, CS-TR-4360, UMIACS-TR-2002-40 Boston, October 2002. A revised version appeared in Proceedings of the IEEE symposium on information visualization, p. 57–64.
[55] http://www.cs.umd.edu/hcil/spacetree/ (accessed: 31 December 2004).
[56] Ackerman A, Chongchitnant N, Sanchez J, et al. Histopathologic diagnosis of inflammatory skin diseases. An algorithmic method based on pattern analysis, 2nd ed., Baltimore, MD: Williams and Wilkins; 1997.
[57] http://www.aperio.com/ (accessed: 31 December 2004).
[58] Schmidt H, Norman G, Boshuizen H. A cognitive perspective on medical expertise: theory and implications. Acad Med 1992;65:611–21.
[59] Kahneman D, Tversky A. On the psychology of prediction. Psychol Rev 1973;80:237–51.
[60] Edwards W. Conservatism in human information processing. In: Kleinmuntz B, editor. Formal representations of human judgment. New York: Wiley; 1968.
[61] Tversky A, Kahneman D. Judgments under uncertainty: heuristics and biases. Science 1974;185:1124–31.
[62] Gluck M, Bower G. From conditioning to category learning: an adaptive network model. J Exp Psychol Gen 1988;8:37–50.