

Knowledge-Based Approaches for Interactive Student Modelling and Adaptive Web Explanations

Vania DIMITROVA
School of Computing
University of Leeds

E-mail: vania@comp.leeds.ac.uk

Kalina BONTCHEVA
Department of Computer Science
University of Sheffield

E-mail: K.Bontcheva@dcs.sheff.ac.uk

Abstract. The paper presents two knowledge-based approaches to the design of intelligent tutoring systems focusing on diagnosing students and generating adaptive explanations. We discuss the use of a domain ontology encoded with conceptual graphs in two recent works in Artificial Intelligence in Education and Natural Language Processing: STyLE-OLM (an interactive learner modelling system that extracts extended models of the learners' cognition) and HYLITE+ (a natural language generation system that generates adaptive Web pages based on a learner model).

1. Introduction

Domain knowledge plays an essential role in building intelligent tutoring systems since it provides the needed expertise for diagnosing the learner's abilities and for planning adaptive instruction [15]. Therefore, intelligent systems that support terminology understanding have to employ appropriate expert knowledge about domain terms. Terminology is defined as "a structured set of concepts and their designations in a particular subject field" [11], and is directly linked to conceptualisation. Terminological knowledge is represented as *domain ontology* that includes concept type taxonomy and a set of axioms that define concept properties, roles, and relations with other concepts [12].

This paper discusses the use of domain ontologies encoded in Conceptual Graphs (CGs) [14] in two adaptive instruction systems: STyLE-OLM and HYLITE+. The former exploits CGs for planning diagnostic dialogue, reasoning about the student's domain beliefs, and assigning a level of correctness to the student's knowledge. The latter employs CGs for modelling users and defining the content of adaptive web explanations. Both approaches address the understanding of terminology where a learner/user has to familiarise with concepts in a specific domain.

CGs, with their formal structures and operations, appear to be a suitable formalism for constructing domain ontologies. CGs allow concept descriptions using necessary and sufficient conditions as well as

structuring the concepts in a hierarchy. Moreover, CGs provide syntactic clarity together with powerful expressiveness about quantifier processing, which concerns the individual-generic distinction supported by the flexibility of references [3]. The application of inference rules supports extracting knowledge which is not directly encoded in the domain ontology.

Due to their suitability for representing terminology, CGs have been employed in both systems discussed here. In addition, certain characteristics of CGs are essential for interactive open learner modelling and for adaptive web explanations. STyLE-OLM adopted CGs because they are (a) rigorous, permitting reasoning fruitful for interaction planning and student diagnosis; (b) relatively understandable having a clear graphical form useful for learner model (LM) inspection and discussion. HYLITE+ adopted CGs because they support (a) the representation of knowledge at different levels of granularity through use of nested graphs; (b) are easily linked to the lexicon entries.

Next in the paper we briefly outline STyLE-OLM and HYLITE+ (#2). The following two sections present the use of domain ontology for interactive student modelling in STyLE-OLM (#3) and for adaptive web explanations in HYLITE+ (#4). Finally, based on the evaluation of the two systems, we discuss the suitability of CGs in intelligent tutoring systems for terminology learning.

2. Brief description of StyLE-OLM and HYLITE+

STyLE-OLM [9] is an environment for interactive diagnosis where a learner model is constructed with the active participation of the learner being allowed to inspect and discuss the content of the model the computer builds of her. The system is based on a framework for interactive open learner modelling [8] and includes a discourse model that manages the interaction between the computer and the user, a

modal logic mechanism that maintains a jointly constructed user model, and a graphical communication medium that provides both the computer and the learner with a symmetrical power in maintaining the LM. Two instantiations of STyLE-OLM are developed – in Computing and Finance (the last has been integrated in the STyLE¹ terminology learning environment).

The interaction in STyLE-OLM includes two modes: DISCUSS, where the learner discusses aspects of her domain knowledge and influences the content of the LM, and BROWSE, where the learner inspects the current state of her LM. The communication medium employs graphically rendered CGs to present discussed domain propositions (in DISCUSS mode) and beliefs from the learner model (in BROWSE mode). Throughout the discussions, the system makes plausible inferences about what further the learner believes on the basis of what is explicitly asserted, and from this a dialogue strategy is determined. The dialogue is guided by general explanations of misconceptions, for example, that an entity may be believed to be a member of a class if it has some features of that class. The inspection of the LM provides learner control over the diagnostic process and can trigger further discussions. A resultant LM that incorporates the views of both the computer and the learner about the learner's knowledge is extracted.

HYLITE+ [5,6] is a dynamic hypertext system² that generates encyclopaedia-style explanations of terms in two specialised domains: chemistry and computers. The user interacts with the system in a Web browser by specifying a term she wants to look up. The system generates a hypertext explanation of the term; further information can be obtained by following hypertext links or specifying another query.

The system is based on applied Natural Language Generation techniques, a re-usable learner/user modelling component (ViewGen), and a flexible architecture with module feedback. The adaptivity is implemented on the basis of a learner and discourse models which are used to determine, for example, which concepts are unknown, so clarifying information can be included for them. The LM is updated dynamically, based on the user's interaction

¹ STyLE (Scientific Terminology Learning Environment) was built as part of the EU funded Larflast project, see <http://www.larflast.bas.bg/> for more details.

² In dynamic hypertext, page content and links are created on demand and are often adapted to the user and the previous interaction.

with the system. When a user registers with the system for the first time, her model is initialised from a set of stereotypes. The system determines which stereotypes apply on the basis of information provided by the user herself. If no such information is provided, the system assumes a novice user.

3 Interactive Learner Modelling

3.1. Learner Model and Domain Ontology

STyLE-OLM imports a domain ontology built with CGs, which includes a taxonomy of concept types and a set of graphs that represent propositions with relations between concept types and individuals. The LM is yielded by an extended overlay method and incorporates *beliefs*, open for inspection and discussion with the learner, and *misconceptions*, used for dialogue planning. Beliefs are domain propositions represented with CGs, and can be *correct* (student beliefs that are supported by the expert), *erroneous* (student beliefs that are not supported by the expert), and *incomplete* (missing beliefs that the expert believes but the learner does not possess them yet). Misconceptions in STyLE-OLM are explanations for the learner's errors at conceptual level and apply to error patterns, such as misclassification or misattribution. These are defined as bug rules, e.g. a misclassification rule is:

$$\exists A: \{i \in A \ \& \ t \in A\} \Rightarrow isa(i, t)$$

(i.e. an individual i has been wrongly considered as a member of a concept type t because i has features that are part of the definitional features for t , e.g. Visual Basic [i] is an Object-Oriented Language [t] because it contains objects [A]).

Based on the bug rules, CG schemata are defined to determine conditions that STyLE-OLM has to examine in a following dialogue in order to confirm a misconception, e.g. the misclassification error above will trigger the CG schema:

```
Error: isa(i,t). Type: Misclassification_1.
Conditions_to_Check=[]
Gi = find_graph(i)
Gt = definition(t)
A = generalisation_subst(Gi, i, Gt, t)
Ai = restrict(A, θ, i)
At = restrict(A, θ, t)
Conditions_to_Check=add([Ai, At],
                        Conditions_to_Check)
Return: Conditions_to_Check
```

This schema will be applied when the learner makes a wrong statement, e.g.

L. *I think that Visual Basic is an object-oriented language.*³

Following the schema, STyLE-OLM first searches both the CGs from the domain ontology and the CGs that represent the learner's beliefs for a proposition about *t* (object-oriented language). Then, a generalisation is found which is a common feature of *t* and *i* (Visual Basic) - they both contain objects. Restricting the generalisation to *i* and *t*, the schema will return that a possible explanation for the learner's erroneous belief may be that she believes 'Visual Basic contains objects' and 'Object-oriented languages contain objects'. In the following dialogue, STyLE-OLM will check these beliefs by asking appropriate questions, i.e. the next move will be:

STyLE-OLM. *Do you think that Visual Basic contains objects?*

If the learner does not confirm this belief:

L. *I don't think so.*

the system will search for another misclassification rule to explain the erroneous belief *Visual Basic is an object-oriented language*'. Such a rule is:

$$\exists j:\{isa(j,t) \ \& \ \exists A:\{j \in A \ \& \ i \in A\}\} \Rightarrow isa(i,t)$$

(i.e. Visual Basic [*i*] is an object-oriented language [*t*] because both it and Visual C++ [*j*], which is an object-oriented language, allow programming in a visual environment [*A*]). The system will then check the conditions for this misclassification:

STyLE-OLM. *Do you think that Visual C++ is an object-oriented language?*

3.2. Maintaining diagnostic dialogue

The dialogue in STyLE-OLM is organised as a series of dialogue games, which represent dialogue episodes and follow diagnostic tactics. There are three types of game:

Explain learner errors. The system initiates such games to look for reasons that might have caused a learner's erroneous beliefs. CG schemata based on misconception rules are used for planning dialogue content, as shown above.

Explore domain knowledge. These games aim to gather as much as possible information about the learner's domain beliefs. They are initiated either by

the learner when she changes the dialogue focus making a claim or asking a question or by the system when it probes for a learner's beliefs in order to decide how to react to her errors. To plan interaction content, the system extracts from the ontology CGs that contain focus concepts. It generates more CGs by applying specialisation rules (replacing concept types with descendants from the hierarchy).

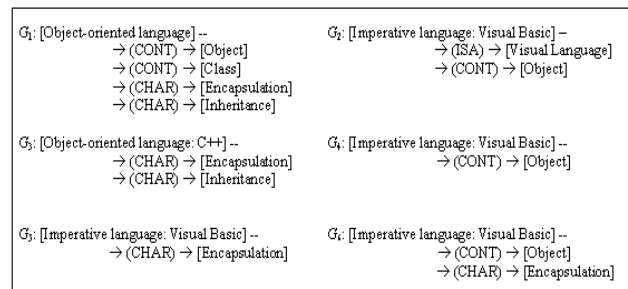
For example, following that *isa(Java, Object-oriented language)*, the system infers that

'Java contains objects and classes and has characteristics inheritance and encapsulation'.

This proposition will be included in a dialogue game that explores the learner's knowledge of object-oriented languages. Note that the system can either ask a question or make a statement with this proposition depending on the current state of the LM.

Negotiate learner model. These games aim at clarifying agents' positions when discrepancies in their views are discovered. Such games may be initiated by both parties when asking the other side for justifications, e.g. the student may challenge the validity of the computer's beliefs about the student's knowledge, while the computer may ask for clarification when a learner's statement is not confirmed by the domain ontology.

A graph *G* constructed by the learner is *confirmed* if *G* is a specialisation of a graph from the domain ontology or if *G* can be projected into a graph from the domain ontology. The example below shows two graphs *G*₃ and *G*₄ that are confirmed from the domain ontology because *G*₃ is a specialisation of *G*₁ and *G*₄ is a projection of *G*₂, where *G*₁ and *G*₂ are from the domain ontology. However, the graphs *G*₅ and *G*₆ are not confirmed by the domain ontology.



Dialogue coherence in STyLE-OLM is maintained through a focus mechanism. Each dialogue game has a *focus space* - list of related concepts that can be discussed in the game to assure the dialogue is coherent. Related concepts are defined as either directly linked in the taxonomy through *isa*

³ Note that the dialogue in STyLE-OLM is conducted in a graphical manner but is presented here in a textual form due to space constraints.

links (e.g. object-oriented language, Java, C++) or belonging to a common CG (e.g. object-oriented language, inheritance, encapsulation).

The relevance of a learner's utterance to the dialogue game focus is verified by comparing the concepts in her proposition with the game focus space. If there is no intersection, STyLE-OLM concludes that the learner has suggested a change in focus and an explanatory dialogue game is initiated (if appropriate) to discuss domain propositions related to the new focus. When the learner stays on focus STyLE-OLM follows the tactic of the current dialogue game but rearranges the propositions to be discussed, so that the ones that are most relevant to the student's move will be discussed first.

3.3. Answering questions

STyLE-OLM enables a learner to ask questions about domain facts. Two question types are catered for: *what-questions* (the user asks for a domain concept with certain properties) and *is-true-questions* (the user checks the validity of a proposition). Giving learners opportunities to clarify aspects of the domain promotes concept learning and meta-cognition. For example, when a learner's erroneous claim:

L. *I think that C is an object-oriented language.*

is challenged by the system:

STyLE-OLM. *Why do you think that C is an object-oriented language?*

the validity of the learner's beliefs is questioned. She searches for possible clarifications, such as the definition of the concept type:

L. *What are the characteristics of an object-oriented language?*

This question is encoded in a CG query form as:

```
(G7) [OBJECT-ORIENTED LANGUAGE] -  
      (CHARACTERISTIC) -> [θ]
```

The answer is a projection of the query upon the graph with the definition of the concept type object-oriented language (graph G_1 above).

STyLE-OLM. *Object-oriented languages have characteristics inheritance and encapsulation?*

Alternatively, the learner may compose a proposition and check its validity, e.g.

L. *Is it true that C has objects?*

To answer this question STyLE-OLM employs a mechanism for confirming a graph, similar to the one used for focus maintenance, as described above.

3.4. Updating the Learner Model

Throughout the dialogue the beliefs of the system and the learner about the learner's knowledge are accumulated in separate belief stores. When the interaction finishes, the belief stores are combined and a resultant learner model is extracted (by using some modal logic plausible reasoning, see [10]). Each belief in the resultant model, represented with a CG, is assigned a degree of correctness: *correct* (if confirmed by the domain ontology), *erroneous* (if not confirmed by the ontology), and *incomplete* (the CG is in the ontology but is not believed by the learner).

An evaluative study of STyLE-OLM with postgraduate students at Leeds University showed the potential of the approach for improving the quality of the learner model and providing the means for reflection [9]. Some participants in the study stressed the need of computer explanations of domain terms generated in a natural language and tailored to the learners' knowledge. This vital feature, which is missing in STyLE-OLM, is illustrated in HYLITE+.

4. Generating adaptive Web explanations

4.1. The Learner Model and the Ontology

The learner model and the ontology are the basis of the generated adaptive hypertext explanations. HYLITE+ uses the ViewGen user/learner modelling framework [2] to access and update the user beliefs as the interaction progresses. The propositions believed by each user are encoded as CGs and the CG reasoning mechanisms are used to detect whether or not the learner already holds a given belief. ViewGen distinguishes the beliefs of each learner by putting them in separate environments, including one for the system itself which contains the domain ontology. The motivation for holding the domain facts as system beliefs, instead of being outside the learner modelling component, is that the system can treat them in the same manner and also deal with incomplete system beliefs.

The separation between system and user belief environments also enables the modelling of discrepancies between propositions believed by the system and those believed by the learner. One example of such discrepancies is taxonomical knowledge, where the system has detailed domain

taxonomy while the learner has only partial and/or incorrect representation. Such discrepancies might come from common misconceptions encoded as stereotypes (e.g., common mistake about photographic emulsion) or be specific to the given learner, in which case they are stored exclusively in her belief space. Misconceptions from stereotypes are ascribed dynamically to all learners who conform to that stereotype.

In this way, differences between the domain ontologies of different agents are easily encoded. ViewGen's operations take such differences into account and can reason correctly about the learner's beliefs when these are affected by different ontologies. For example, when determining the beliefs of a chemistry expert about photo emulsion, ViewGen also considers propositions inherited from gels and sols. However, by default for laymen, ViewGen takes into account the default stereotype and includes beliefs inherited from emulsion instead.

4.2. Generating the explanations

When the user requests information about a topic (e.g., emulsion), HYLITE+ needs to select a set of relevant propositions, from which the generation algorithms produce the hypertext page in the following manner:

- Obtain relevant propositions from the KB given the user's question about a concept.
- Determine which CG propositions are known already by the learner (see below) and mark them as such. In case of contradictory beliefs between system and learner, both system and learner beliefs are added to the propositions to be conveyed, e.g. *bel(user, isa(photo_emulsion, emulsion))* and *bel(system, isa(photo_emulsion, gel))*.
- Inspect each proposition for unknown related concepts and mark them as such.

Unknown concepts in definitions, as well as unknown properties, parts, and super-/sub-concepts, trigger the inclusion of additional information (usually their definition), e.g.:

A personal computer is a type of microcomputer⁴ (a small computer that uses a microprocessor as its central processing unit).

The learner model is also used to detect partially known facts (see below) which are then verbalised by

⁴ The underlined words are hypertext links to a Web page providing their definitions and other information. These pages are generated only if requested.

the generator using contextual phrases such as *besides* and *as well as*. For example, if the learner has already seen an explanation of the concept TAPE DRIVE and follows the link to STORAGE DEVICE, then the description of the types of storage devices is generated using such a phrase:

As well as tape drives, other types of data-storage devices are ...

Incorrect beliefs, detected in the learner model, are explained by providing them in parallel with the correct fact. In the photographic emulsion example shown above, the system generates the following sentence as part of the explanation⁵:

A common error is to believe that photographic emulsion is emulsion, whereas, in fact, it is a gel.

4.3. Consulting the UM during generation

Given the set of relevant propositions selected for generation, the generator first establishes whether any of them are already believed by the learner. ViewGen is queried about each proposition and all already believed CG propositions are annotated as such.

After all believed CG propositions are identified, the system also checks for any partially known propositions. This is necessary because the generator frequently uses CG reasoning operations to present only part of a given conceptual graph (to make the explanation shorter). For example, one of the graphs extracted for an explanation of the DISPERSION concept is⁶:

```
(cg2)
[DISPERSION]<- (CHAR) <- [WASTE_WATER].
:fs(derived_from: cg1)
```

This is in fact a subgraph of the following:

```
(cg1) [WASTE_WATER] -
      (CHAR) -> [CONCENTRATION]
      (CHAR) -> [RESISTANCE]
      (CHAR) -> [DISPERSION].
```

Therefore, if the learner asks later about WASTE_WATER, the generator extracts the whole graph cg1. Naturally ViewGen then returns that this fact is

⁵ The "a common error is..." expression is used to express the uncertainty of this assumption which has been made on the basis of a stereotype. The generator can also deal with incorrect beliefs of individual users.

⁶ fs() is used to encode features which can be associated with concepts, relations and graphs. In this example the feature specifies which graph was used in the derivation of the graph it is associated with.

not already believed by the learner since the learner belief environment only contains graph `cg2`.

Such partially known facts are identified by the generator in the following way:

- Obtain from ViewGen all propositions already believed by the learner.
- Check every obtained graph for a `derived_from` feature. If found, check whether the graph from which it was derived is among the graphs selected for explanation. Then annotate each such parent graph (e.g., `cg1`) as partially known (see example below).
- Use subsumption to determine whether some of the graphs returned as already believed by the learner subsume parts of a graph selected for explanation. If so, annotate the appropriate part as already known and the whole graph as partially known.

In our example, at this step graph `cg2` will be identified as already believed by the learner and also derived from `cg1`. So `cg1` is annotated as partially known and the relevant parts of it are annotated as already believed⁷. The result is:

```
[WASTE_WATER] -  
  (CHAR) -> [CONCENTRATION]  
  (CHAR) -> [RESISTANCE]  
  (CHAR) :fs(um_state: known) -> [DISPERSION].  
  :fs(um_state: partially_known)
```

In other words, a new feature `um_state` is added to the whole graph and also all relations which also occur in the already explained graph. This information is then used by the generator to choose a phrasing which considers that dispersion is previously seen, and uses “*as well as*”, which conveys this fact and makes the generated hypertext more coherent.

Finally, ViewGen is also consulted about concepts that appear in the selected graphs. If no learner beliefs can be obtained about these concepts, then the generator annotates them with a `um_state: unknown` feature, which is used to determine that additional information needs to be provided for this concept, e.g., its definition in parenthesis.

⁷ These added features do not alter the conceptual graph in any way and can be ignored by other applications which want to use them for other purposes. Also the features are removed by the application as soon as the text is generated. In this way the graph can be annotated differently in later explanations if the user beliefs have changed.

5. Discussion

The evaluation⁸ of STyLE-OLM [8] and HYLITE+ [4] confirmed the suitability of conceptual graphs for interactive student modelling and adaptive explanations in terminological domains. The dialogue in STyLE-OLM was handled in a coherent manner, more aspects of the students’ domain knowledge were revealed, some reasons for erroneous beliefs were discovered. Along this line, the users working with HYLITE+ appreciated the relevance of the domain content of the explanations produced, and found the adaptive features adequate and beneficial.

It must be stressed that in both systems CGs were used in combination with another logical framework that handled the maintenance of the learner models and included some commonsense reasoning over the agents’ beliefs. We found that CGs were best suited for representing propositional knowledge and an additional reasoning level was added to reason about agents’ beliefs (which commonly included nested beliefs and non-monotonic reasoners). Although [13] present a user model based on CG actors used in an adaptive system and [1] propose a user modelling mechanism that can build user profiles by observing users’ interaction with a computer system, these projects consider modelling a user’s cognitive state merely on a belief level. The learner models we have described here incorporate learner’s beliefs and misconceptions that provide possible explanations of patterns of learner errors. CGs on their own are not sufficiently flexible to allow such misconceptions to be represented, therefore we use CGs only as means for representing and communicating about domain knowledge and employ a meta-level commonsense reasoning to infer about the users’ beliefs.

Both approaches presented here suppose that domain ontology provides all the necessary factual knowledge. However, this may not always be the case in learning systems. Often, tutoring knowledge is incomplete (lacking a particular piece of information), vague (there are concepts that are difficult to define), ambiguous (it may be hard to distinguish between several possibilities for a learner’s reasoning). Further investigation is needed to address these aspects in tutoring systems, which may consider similar studies in ontology (e.g. [7]).

⁸ The systems have been evaluated independently in small experimental studies with participants from the authors’ organisations.

References

- [1] Baldwin, J. F., Martin, T. P., & Tzanavari, A. User modeling using conceptual graphs for intelligent agents. In B. Ganter & G.W. Mineau (Eds.), *Conceptual structures: logical, linguistic, and computational issues: 8th international conference on conceptual structures*, Springer, 2000.
- [2] Ballim, A., & Wilks, Y. *Artificial Believers*. Lawrence Erlbaum Associates, Hillsdale, NJ, 1991.
- [3] Biebow, B., & Chaty, G. A comparison between conceptual graphs and KL-ONE. In G. W. Mineau, B. Moulin & J. Sowa (Eds.), *Conceptual graphs for knowledge representation: Proceedings of the First International Conference on Conceptual Structures*. Springer-Verlag, 1993, pp. 75-89.
- [4] Bontcheva, K. *Generating Adaptive Hypertext Explanations with a Nested Agent Model*. PhD thesis, University of Sheffield, 2001.
- [5] Bontcheva, K. Tailoring the Content of Dynamically Generated Explanations. M. Bauer, P.J. Gmytrasiewicz, J. Vassileva (eds). *User Modelling 2001: 8th International Conference*, Lecture Notes in Artificial Intelligence 2109, Springer Verlag, 2001.
- [6] Bontcheva, K., & Wilks, Y. Dealing with dependencies between content planning and surface realisation in a pipeline generation architecture. In *Proceedings of the International Joint Conference in Artificial Intelligence (IJCAI'2001)*, Seattle, USA, 2001.
- [7] Da Silva, F. et al. On the insufficiency of ontologies: problems in knowledge sharing and alternative solutions. *Knowledge-Based Systems*. 15, 2002, 147-167.
- [8] Dimitrova, V. *Interactive Open Learner Modelling*, PhD thesis, Computer Based Learning Unit, Leeds University, 2001.
- [9] Dimitrova, V. STyLE-OLM Interactive Open Learner Modelling. *International Journal of Artificial Intelligence in Education*, to appear.
- [10] Dimitrova, V., Self, J.A., & Brna, P. Maintaining a jointly constructed student model. In S.A. Cerri & D. Dochev (Eds.), *Artificial intelligence: Methodology, systems, and applications*. Springer Verlag, 2000, pp. 221-231.
- [11] Galinski, C., & Budin, G. Terminology. In Cole R. A. et al., Eds., *Survey of the State of the Art in Human Language Technology*, <http://www.cse.ogi.edu/CSLU/HLTsurvey/HLTsurvey.html>
- [12] Gruber T. Toward principles for the design of ontologies used for knowledge sharing. *International Journal of Human-Computer Studies*, 43, 1995, pp. 907-928.
- [13] Nenkova, A., & Angelova, G. User modelling as an application of actors. In W. Tepfenhart & W. Cyre (Eds.), *Conceptual structures: Standards and practices: 7th International conference on conceptual structures*. Springer, LNAI 1640, 1999, pp. 83-89.
- [14] Sowa, J. *Conceptual Structures: Information Processing in Mind and Machine*, Addison-Wesley, MA, 1984.
- [15] Wenger, E. *Artificial Intelligence and Tutoring Systems*. Morgan Kaufman, 1987.