Graph Grammars: An ITS Technology for Diagram Representations

Niels Pinkwart[1], Kevin D. Ashley[2], Vincent Aleven[3] and Collin Lynch[4]

[1]Clausthal University of Technology, Department of Informatics, Germany
[2]University of Pittsburgh, LRDC and School of Law, Pittsburgh PA, USA
[3]Carnegie Mellon University, HCI Institute, Pittsburgh PA, USA
[4]University of Pittsburgh, Intelligent Systems Program, Pittsburgh PA, USA
niels.pinkwart@tu-clausthal.de

Abstract

For many educational applications such as learning tools for argumentation, structured diagrams are a suitable form of external representation. However, student-created graphs pose some problems to ITS designers, especially in defined domains. This paper demonstrates a graph-grammar-based approach for ITS construction in domains that benefit from diagram representations. For these, graph grammars offer some helpful affordances. They make it easy to express possible manipulations by which a student might create a diagram, they facilitate the definition of structurally complex and pedagogically interesting constellations of graph elements to which an ITS should respond with feedback messages, and they offer a general parsing mechanism that allows the analysis of student-created diagrams by recognizing these constellations in graphs.

Introduction

Argumentative thinking skills are critical for humans in many aspects of life (Kuhn, 2001). Consequently, teaching argumentative skills is a central goal of education – both on a general level (classroom dialog is a form of group argumentation), and also in specific application areas such as science or law, which define rules about what constitutes an acceptable argument.

Often, argumentation is taught by human teachers through face to face dialog. Intelligent Tutoring Systems for argumentation are still relatively rare. The few ITS systems for argumentation usually allow students to work with explicitly structured argument representations, often, visualized in diagram form. This helps both the system and the learner. From a cognitive perspective, graphical representations can reduce the students' cognitive load and reify important relationships (Larkin and Simon 1987). From a system design point of view, the explicit structure can provide a practical means for ITSs to provide feedback. Examples for systems that follow this approach are the Belvedere system for scientific argumentation (Paolucci, Suthers and Weiner 1996) which compares a student-created argument graph to an ideal solution, the ArguMed system (Verheij 2003) which provides intelligent feedback through an argumentation "assistant" that analyzes structural relations between contributions in diagrams, and the Araucaria system (Reed and Rowe 2004) that allows the specification of argument schemes (premises and feedback messages) instances of which can be detected in student diagrams. Carneades argument diagrams (Gordon 2007) can be used to map legal evidence and allow informing users automatically if a specific claim satisfies a given proof standard (no specific educational usage of Carneades has been reported yet). The ARGUNAUT system (McLaren et al. 2007) analyzes the graph structure and language of arguments using a combination of machine learning and text analysis techniques. This system is not an ITS system in a narrow sense, since analysis results are used to inform the teacher, not to give specific feedback to the student.

These existing tools are either based on relatively straightforward approaches (comparisons between student and ideal graph, search for explicit elements in student graphs), or they are very domain specific. A general and reusable technology for developing Intelligent Tutoring Systems based on graph structured (argument) representations has not emerged yet. Two factors can account for this. Firstly, student-created graphs are *structurally and computationally more complex* than "traditional" user interfaces. This poses problems for some approaches: for instance, propositional logic is not a suitable technology for ITSs that analyze diagrams (since cycles of arbitrary length cannot be detected). Predicate logic is expressive enough, but difficult to apply in practice and not especially suited for graph based representations.

Secondly, *many forms of argumentation are ill-defined*: typically a "correctness" notion for an argument is impossible to define or verify formally because the underlying concepts are open-textured and the quality of an argument may be subject to discussion or even expert disagreement. For these reasons, it is very hard (if not impossible) to develop a detailed and formalized domain model for an ITS. Such a model that describes the process of building "good graphs" would be the prerequisite for applying Cognitive Tutor approaches (Anderson et al. 1995) to argumentation. This second point is more related to the domain of argumentation than to properties of a representation – but in this context it is worth noting that for many educational approaches in ill-defined domains, graph-structured representations are frequently used - see Aleven et al. (2007) for an overview of recent approaches.

In order to appropriately deal with structured diagram representations, an ITS needs components that allow:

a) representing diagram system-internally. This is a prerequisite in order to enable the ITS to do the tasks described in b) and c).
b) handling changes in diagrams. This includes a specification of which diagram changes a user is allowed to do: if it makes sense pedagogically to prevent students from making certain "syntax errors" in diagrams (e.g., by connecting elements that should not be connected, according to the domain model), then the ITS should be able to implement this strategy. This requires a way to express syntax constraints which is expressive enough for graph structures.
c) analyzing diagrams to give feedback. This can be done in various forms – e.g., based on full domain models (if these exist), domain ontologies (with domain concepts and relations), or using semantic constraints.

This paper describes how, for these typical ITS tasks, graph grammar technology can be used as a tool that is specially suitable for extending ITS tutoring to student-authored graphs and especially useful for ill-defined domains. We use the domain of legal argumentation and the LARGO ITS as an illustrating example. While the general system has been published before (Pinkwart et al. 2006), this paper focuses on the underlying technology and its general value for the ITS field.

Example Domain: Legal Argumentation

We present a model and example of legal argument that has been incorporated in an ITS system design based on graph grammars. The model (Ashley 2007) involves tests and hypotheticals as key elements: In US Supreme Court oral arguments, contending attorneys each formulate a hypothesis about how a set of issues in the problem should be decided. They may propose a *test* and identify key points on which the issue should turn, thus relating the *facts* at hand to their hypotheses and choosing how best to describe them. The Justices test those hypotheses by posing *hypothetical scenarios* designed to challenge the hypotheses' consistency with past decisions, and the purposes and principles underlying the legal rules.

The following excerpt from the case of *California v. Carney,* 105 S. Ct. 2066 (1985) involved the legality under the 4th Amendment of a warrantless search of a motor home located in a downtown parking lot. On appeal, the State's attorney, Mr. Hanoian, proposed a "bright line" test of when the "vehicle exception" to the warrant requirement should apply:

> MR. HANOIAN: If the vehicle has wheels on it, I think that that makes it mobile and it would be subject to the exception….If it still has its wheels and it still has its engine, it is capable of movement and it is capable of movement very quickly.
> JUSTICE: Even though the people are living in it as a home and are paying rent for the trailer space, and so forth?
> JUSTICE: Well, there are places where people can plug into water, and electricity, and do. There are many places, for example, in the state I came from where people go and spend the winter in a mobile home. And you think there would be no expectation of privacy in such circumstances?
> MR. HANOIAN: Well, I am not suggesting that there is no expectation of privacy in those circumstances, Your Honor.

In the excerpt, Mr. Hanoian proposes a test, whether the vehicle/home is capable of self-locomotion, and then has to respond to the Justice's challenge hypothetical: a motor home that is hooked up to water and electricity. Mr. Hanoian responds that such a vehicle might still be moved in a hurry, but concedes the owners would have some expectation of privacy.

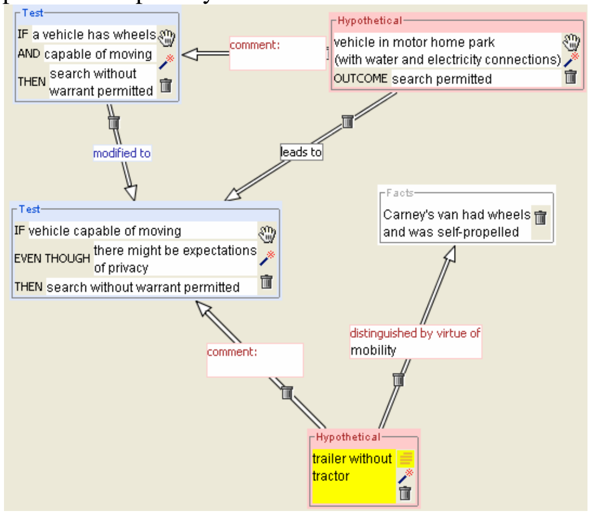

Test
IF a vehicle has wheels
AND capable of moving
THEN search without warrant permitted
comment:
Hypothetical
vehicle in motor home park (with water and electricity connections)
OUTCOME search permitted
modified to
leads to
Test
IF vehicle capable of moving
EVEN THOUGH there might be expectations of privacy
THEN search without warrant permitted
Facts
Carney's van had wheels and was self-propelled
comment:
distinguished by virtue of mobility
Hypothetical
trailer without tractor

Figure 1. An example LARGO Argument Diagram

Examples of oral argument can give beginning law students an overview of a process that they may only dimly perceive, even as they engage in it in a class. The LARGO system allows students to graphically represent the dialectical pattern of hypothetical reasoning described above, and gives feedback on student-created argument diagrams. Figure 1 shows an example graph created in LARGO. It contains the tests and hypotheticals from the above excerpt, an element representing the facts, and five relationships among these. For example, the "trailer without tractor" hypothetical is distinguished from the current fact situation.

Students can also link the single elements in their graph to passages in a transcript of an oral argument (not shown in the Figure), using a text highlighting feature. These "transcript links" enable referencing external sources from graph elements, a common ITS requirement.

Graph Grammars in LARGO

The ITS graph grammar mechanism we propose here, illustrated with examples from the LARGO ITS, is based on the grammar format proposed by Rekers and Schürr (1997). A grammar consists of a set S of symbols (which can have attributes), a start axiom, and a set of production rules. Words in the grammar represent student-created diagrams, while rules of the grammar are used to handle changes in diagrams and to analyze the diagrams in order to give feedback. Details on the implementation of these principles are demonstrated in the next subsections

Diagrams as Grammar Words

The symbols of the grammar contain node and edge types within graphs, which correspond to the domain concepts and relations, as well as some help symbols (in S_c):

$S = S_n \cup S_e \cup S_c$

S_n = {test, hypothetical, fact}

S_e = {distinction, analogy, modification, causality, relatedness}

Technically, a word in the grammar (i.e., a diagram) is a 4-tuple (N, E, M, C). N (nodes) and E (edges) contain the diagram data. M (metadata) consists of summary information such as element counters, while C (characteristics) contains the ITS specific metadata: an element of C is an aspect of a diagram that the ITS can give feedback on.

The start axiom of the grammar in LARGO is:

(∅, ∅, M, C)

Here, N and E are initially empty (i.e., the diagram does not contain elements). M contains a counter for the number of elements (initially 0) and information about irrelevant regions in the argument transcript (that LARGO is supposed to comment on, should these be referenced from a diagram). C contains a list of initial graph characteristics. It may appear strange at first sight that the start axiom can already contain graph characteristics. This makes sense, however, if one wants to express the *absence* of specific, pedagogically relevant, elements in a diagram (which is of course fulfilled for empty graphs). In LARGO, characteristics about *important regions* in the transcript (not referenced from a diagram) are included in C. These lists constitute the only task-specific information in the tool – i.e., using LARGO with another legal case only requires a change of these two lists.

The diagram shown in Figure 1 has the following representation (N, E, M, C), assuming that the node of type "test" with the content "if vehicle has wheels…" is linked to the characters 1050-1150 of the transcript, and the node of type "hypothetical" with text "vehicle in motor home park…" to the characters 1520-1550, respectively. Details for the second test and the other hypothetical are omitted in the example for space reasons.

N = {test_A, test_B, hypothetical_A, hypothetical_B, fact}
E = {modification, relatedness_A, relatedness_B, causality, distinction}
with:
test_A.conditionText = "IF vehicle has wheels AND capable of moving",
test_A.conclusionText = "THEN search without warrant permitted",
test_A.linkStart = 1050,
test_A.linkEnd = 1150,
hypothetical_A.text = "vehicle in motor home park (with water and electricity connections) OUTCOME: search permitted",
hypothetical_A.linkStart = 1520,
hypothetical_A.linkEnd = 1550,
fact.text = "Carney's van had wheels and was self-propelled",
modification.from = test_A,
modification.to = test_B,
relatedness_A.text = nil,
relatedness_A.from = hypothetical_A,
relatedness_A.to = test_A,
relatedness_B.text = nil,
relatedness_B.from = hypothetical_B,
relatedness_B.to = test_B,
causality.from = hypothetical_A,
causality.to = test_B,
distinction.text = mobility,
disctincion.from = hypothetical_B,
distinction.to = fact

In our notation, attributes of a symbol are encoded using an object-oriented notation (symbolname.attributename). Letter indices for symbols are used to distinguish between different occurrences of a symbol. The above example illustrates how each diagram element (node or edge) has an explicit representation in the grammar expression, and how information about these diagram elements such as the contained text or the links to external sources (such as transcript passages) can be encoded as attributes.

As mentioned, the metadata M and the characteristics C partially depend on task-specific information, in the case of LARGO specifically on the transcript locations of important tests and hypotheticals, and of irrelevant passages. For example, if there is one irrelevant passage (characters 100-200 of the transcript), and one important hypothetical (2500-2600), then M and C are as follows:

```
M = {counter, irrelevant}
C = {characteristic}
with:
counter.testCount = 2.
counter.hypoCount = 2,
counter.factCount = 1,
counter.relationCount = 5,
irrelevant.linkStart = 100,
irrelevant.linkEnd = 200,
characteristic.linkStart = 2500,
characteristic.linkEnd = 2600,
characteristic.id = missed_hypo,
characteristic.referenceList = ∅
```

M here contains counters for the total numbers of nodes for each of the types, and for the total number (here: 5) of relations in the diagram. Technically, this is not absolutely needed for the production rules (the numbers could be calculated on the fly through grammar production rules), but their availability speeds up the parsing process. The element "characteristic" means that the diagram does not contain a hypothetical which references a specific part of the transcript (in this case, the characters 2500 to 2600). The accuracy of this information is checked and updated whenever the user asks for advice (cf. next subsections).

Grammar Rules for Editing Diagrams

Production rules of the graph grammar can be used to allow an ITS to process and analyze argument diagrams. The approach discussed here contains production rules of two different types. Rules of the first type (*generation rules*) are designed to enable an easy specification of possible manipulations by which a student might create a diagram (aspect "b" of the list in the introduction) These rules are described in detail in the remainder of this subsection. The second type of rules (*feedback rules*) is described in the following subsection.

Technically, a production rule is represented by a structure

$L = (N_L, E_L, M_L, C_L) \rightarrow (N_R, E_R, M_R, C_R) = R$

of two 4-tuples over S, together with some constraints regarding the attribute values of elements in L and R. A rule L → R can be applied to a graph G iff G contains a subgraph G' which matches L in terms of contained elements and attribute constraints. The result of the rule application is the graph $(G \cup R) \setminus G'$. Thus, a rule application replaces the subgraph that matches the left side of the rule with the graph in the right side. In our notation for a production rule, the L → R expression is specified together with the constraints for rule applicability. Constraints for the left hand side (L) are listed under a section "Condition", and constraints about the right hand side (R) are listed under a section "Result". For variables (i.e., placeholders for a subset of symbols in S), capital letters are used. Number indices for symbols are used to indicate that the same symbol appears on the right and left hand side of a rule, but that its attributes change as a result of the rule application. Letter indices for symbols are used to distinguish between different occurrences of a symbol. I.e., if a rule requires two tests to be contained in a graph, then the notations $test_A$ and $test_B$ are used to distinguish between them.

As mentioned above, the possible user actions in the system are represented as "generation grammar rules". As long as the user edits the diagram, only grammar rules of this type are used. These allow a fine-granular specification of the user options in the system with respect to possible (and impossible) diagram manipulations and thus *define the action space of users in the system by constraining the graphs that students can create* for pedagogical or domain-specific reasons. Edits have to be explicitly allowed in form of production rules, otherwise the user cannot perform them in the system.

Our example system LARGO was designed with the pedagogical premise not to restrict users in their diagram creation. As such, there are only very few syntax restrictions in the LARGO grammar. For example, the following rule specifies that students can make arbitrary text changes within hypothetical and fact nodes:

```
// Rule 10: Change text in hypos and facts is allowed
({N1}, ∅, ∅, ∅) → ({N2}, ∅, ∅, ∅)
with
  N1 ∈ {hypothetical, fact}
Result:
  N2.text ∈ STRING
```

If the user edits the text content of the fact node in our example diagram to "van had wheels", this rule applies. In the application of the rule, N_1 matches the existing fact and N_2 represents the changed fact with the new text.

Another, more complex, example of a generation rule in LARGO allows edges of any type to be added to the graph:

```
// Rule 4: Edges can be added anytime
({N1,M1},∅,{counter1},∅) →
({N2,M2},E,{counter2},()
with:
  N, M ∈ Sn and E ∈ Se
Condition:
  N1 ≠ M1
Result:
  E.from = N2,
  E.to = M2,
  counter2.relationCount =counter1.relationCount+1
```

This example rule expresses that all edge types are allowed between all node types ($N, M \in S_n$ and $E \in S_e$). The only restriction is that loops (edges between a node and itself) are not allowed. In the argumentation model underlying LARGO, it is not reasonable to link up an element to itself. Therefore, the grammar prevents students from doing so. This is an example of a syntax constraint expressed in grammar rule format. If, in our example diagram of Figure 1, a user adds a "leads to" edge (causality relation) from the fact node to the lower test node ($test_B$), this action matches this example rule 4 and is thus allowed. Here,

- N_1 matches the fact node,
- M_1 matches $test_B$,
- E represents the new causality relation (to distinguish it from the existing one, let us name it $causality_B$ here).

As a result of the rule application, the following attribute values are set:

```
causalityB.from=fact, causalityB.to=testB,
counter.relationCount = 6
```

Grammar Rules for Feedback Generation

Production rules of the second type (feedback rules) are used to analyze the graphs and to detect diagram characteristics. They *express the system's domain-specific pedagogical knowledge*, namely, interesting patterns (constellations of graph elements) to which it can respond with feedback messages to the student. Whenever the user asks for advice in the system, the graph grammar engine parses the diagram by applying all possible "feedback rules" repeatedly and thereby generating "characteristics" symbols in the fourth component of the 4-tuple.

In LARGO, diagram characteristics stand for weaknesses in the graph (aspects that might indicate that a student has not understood the argument model) or opportunities for reflection (appropriate diagram parts that would be worth further reflection). The LARGO example graph of Figure 1 has several such characteristics. If none of the hypothetical nodes marks up the characters 2500-2600 of the transcript, then the "missing hypothetical" characteristic from the example in the subsection "diagrams as grammar words" of this paper applies for this diagram – and thus remains in the set C. If some hypothetical in the diagram referenced the transcript characters 2500-2600, the following rule would remove the corresponding characteristic from C.

```
// Rule 29: Important hypothetical referred to
({hypothetical}, ∅, ∅, {characteristic}) →
({hypothetical}, ∅, ∅, {foundlocation})
Condition:
  characteristic.id = missed_hypo,
  [hypothetical.linkStart, hypothetical.linkEnd] ∩
  [characteristic.linkStart, characteristic.linkEnd] ≠ ∅
Result:
  foundlocation.type = hypothetical,
  foundlocation.linkStart = characteristic.linkStart,
  foundlocation.linkEnd = characteristic.linkEnd
```

In LARGO (as in most ITS systems that rely on graph based representations), the primary function of the diagram analysis component is to find specific, pedagogically interesting, patterns in student-created graphs. An example from the legal argumentation domain is a diagram that contains a hypothetical which is both related to the facts and to a test, and at the same time also contains another test. Tests represent proposed decision rules (by attorneys) while hypotheticals are challenges to these tests (by Justices). The described diagram constellation is a pedagogical opportunity to invite the student to reflect upon how the hypothetical scenario relates to the different tests in the light of the facts of the case (e.g., by comparing the decision rules). LARGO can detect the pattern with the following graph grammar rule:

```
// Rule 59: Reflection on hypothetical
({hypothetical,fact,test},{E,F},{counter},∅) →
({hypothetical,fact,test},{E,F},{counter},
 {characteristic})
with
  E, F ∈ S_e
Condition:
  E.from ∈ {hypothetical, fact},
  E.to ∈ {hypothetical, fact},
  F.from ∈ {hypothetical, test},
  F.to ∈ {hypothetical, test},
  counter.testCount > 1
Result:
  characteristic.id = discuss_hypo_mult_tests,
  characteristic.referenceList =
    {hypothetical, fact, test, E, F}
```

The attributes of a characteristic contain a reference list which lists all the diagram elements that a detected instance of the characteristic in the graph is related to. This is used to highlight them in the feedback message and to insert text snippets into the hints (see Figure 2).

In Figure 1, the above example characteristic (rule 59) can be detected. The example rule applies due to the following matches and constraint satisfactions:

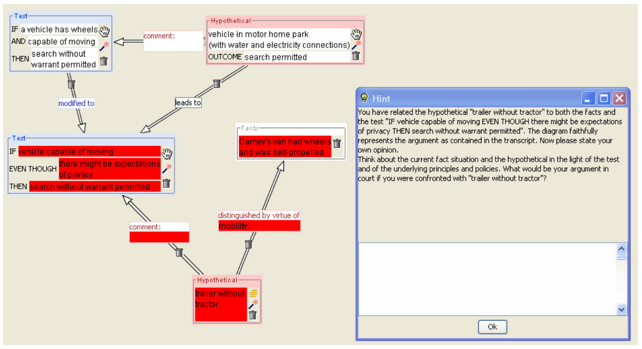
Rule Symbol	**Matching graph element**
hypothetical	hypothetical$_B$
fact	fact
test	test$_B$
E	distinction
F	relatedness$_B$
counter	counter

- distinction ∈ S_e=> E ∈ S_e
- relatedness$_B$ ∈ S_e=> F ∈ S_e
- distinction.from = hypothetical$_B$=> E.from ∈ {hypothetical, fact}
- distinction.to = fact => E.to ∈ {hypothetical, fact}
- relatedness$_B$.from = hypothetical$_B$ => F.from ∈ {hypothetical, test}
- relatedness$_B$.to = test$_B$ =>F.to ∈ {hypothetical, test}
- counter.testCount = 2 => counter.testCount > 1

As a result of applying this rule, a new element "characteristic" is generated in the fourth component of the 4-tuple. Its attribute values are:

```
characteristic.id = discuss_hypo_multiple_tests,
characteristic.referenceList = {hypothetical_B, fact,
test_B, distinction, relatedness_B}
```

By the end of the parsing process, the system has catalogued each characteristic about which it can provide feedback and included them in the set C in the structure. The system then needs to select which advice to provide. Providing a means for choosing which advice to provide at any given time is an important decision because it is very likely that multiple feedback rules match a graph, or even that the same rule matches different parts of a graph. Then, many characteristics can be observed simultaneously. In pilot studies with LARGO, which has 49 feedback rules, sometimes more than 100 characteristics were found in one diagram. Thus, the *selection* problem is of key importance.


Test
IF a vehicle has wheels
AND capable of moving
THEN search without warrant permitted
comment:
Hypothetical
vehicle in motor home park (with water and electricity connections)
OUTCOME search permitted
modified to
leads to
Test
IF vehicle capable of moving
EVEN THOUGH there might be expectations of privacy
THEN search without warrant permitted
Facts
Carney's van had wheels and was self-propelled
comment:
distinguished by virtue of
mobility
Hypothetical
trailer without tractor
Hint
Ok

You have related the hypothetical "trailer without tractor" to both the facts and the test "IF vehicle capable of moving EVEN THOUGH there might be expectations of privacy THEN search without warrant permitted". The diagram faithfully represents the argument as contained in the transcript. Now please state your own opinion.
Think about the current fact situation and the hypothetical in the light of the test and of the underlying principles and policies. What would be your argument in court if you were confronted with "trailer without tractor"?

Figure 2. Graph Grammar Based Feedback in LARGO

The graph grammar can be helpful in this selection process. As described in (Pinkwart et al. 2006), each characteristic (and each feedback rule) can be associated to a typical usage phase – e.g., orientation, transcript markup, diagram creation, analysis, or reflection. The usage phase of a student can be approximated through an analysis of all the feedback rules that match his current graph. This can be based on the heuristic that a large number of detected characteristics of a specific phase suggest that the student is in this phase. Using this information and additional tool interaction data such as visible diagram parts, temporal interaction sequences, or the feedback history, those characteristics that match the current usage context of the student can be prioritized. Figure 2 shows a screenshot of a feedback message in LARGO. Together with the hint, the elements in the diagram that the message relates to are highlighted. These correspond to the referenceList attribute of a characteristic and help the student understand which parts of his diagram a system comment refers to.

Conclusion and Discussion

For many educational applications such as learning tools for argumentation, structured diagrams are a suitable form of external representation. Yet, student-created graphs pose some unique problems for ITS development. This paper describes a graph grammar based approach for the construction of Intelligent Tutoring Systems in domains that benefit from diagram representations, potentially with links to external learning resources. Examples of these domains include database design (Suraweera & Mitrovic 2002), scientific inquiry (Paolocci et al 1996), software design (Baghaei et al 2007) and causal reasoning (Easterday et al 2007). For these application areas, graph grammars are an appropriate technology. They make it easy to represent diagrams system-internally, they facilitate the specification of possible manipulations by which a student might create a diagram, they allow the definition of structurally complex and pedagogically interesting constellations of graph elements to which an ITS should respond with feedback messages to the student, and they offer a general parsing mechanism that can be used to analyze student diagrams for occurrences of these graph constellations in order to give feedback.

The example used in this paper to illustrate the technology comes from the domain of legal argumentation. In this domain, the approach has proven to work in practice both technology-wise and also with respect to learning gains caused by the system (Pinkwart et al. 2006). The graph grammar method is general enough to be applied for the construction of ITSs also for other argumentation domains and, even beyond, in domains that benefit from structured graph representations.

To apply the presented approach in other domains, the concepts of these domains (node and edge types) and the domain-specific pedagogic feedback rules need to be adapted – the core grammar parsing engine is independent of the domain, thus effectively reducing the time and programming efforts needed for ITS development. To modify the student options in the ITS system (e.g., to limit the types of diagrams that can be created in order to guarantee syntax constraints), a change of the generation rules is sufficient – no programming effort is required. If an existing ITS should be used in the same domain but with other tasks (e.g., the same argumentation model, but a different topic to argue about), then almost no changes (except from updating the grammar axiom with links to the new external learning resources) are required in the ITS.

Compared to existing work in the ITS literature on student-created argument diagrams, the graph grammar formalism presented in this paper has some clear advantages. In contrast to special-purpose systems (e.g., Gordon 2007), the design approach is general enough to be applied in other domains. At the same time, graph grammar based ITS systems do still allow for explicitly specifying domain-specific pedagogical rules and feedback messages – in contrast to pure machine learning approaches (e.g., McLaren et al. 2007). Graph grammars can plausibly be used in conjunction with other ITS approaches. Intuitively, they fit with constraint-based modeling (Ohlsson 1994): both approaches do not require expert solutions of problems, as e.g. (Paolucci et al. 1996) does. Instead, they rely on local conditions in student solutions to give feedback. For the specifications of these conditions, graph grammars offer a formalism that is specially suitable for diagrams, and more powerful than tools that rely on simple pattern matching (Reed and Rowe 2004; Verheij 2003). In summary, we believe that graph grammars constitute a quite powerful and promising ITS technology. Through their natural use of graphs as atomic structures, they offer affordances that make it easy to develop ITS systems for diagram representations.

References

Aleven, V., Ashley, K., Lynch, C., & Pinkwart, N. (Eds.) 2007. *Proc. of the Workshop on AIED Applications for Ill-Defined Domains at AIED.* Los Angeles (CA).

Anderson, J., Corbett, A., Koedinger, K., & Pelletier, R. 1995. Cognitive Tutors: Lessons Learned. *The Journal of the Learning Sciences* 4(2): 167-207.

Ashley, K. 2007. Interpretive Reasoning with Hypothetical Cases", *Proc. of FLAIRS,* Key West (FL).

Baghaei, N., Mitrovic, A., & Irwin, W. 2007. Supporting collaborative learning and problem-solving in a constraint-based CSCL environment for UML class diagrams. *International Journal of CSCL* 2(2-3): 159-190

Easterday, M., Aleven, V., & Scheines, R. (2007). 'Tis better to construct than to receive? The effects of diagramming tools on causal reasoning. In *Proc. of AIED*, 93-100. Amsterdam, IOS Press.

Gordon, T. F. 2007. Visualizing Carneades argument graphs. *Law, Probability and Risk* (Advance Access).

Larkin, J., & Simon, H. 1987. Why a diagram is (sometimes) worth ten thousand words. *Cog.Sci* 11:65-99

Kuhn, D. 2001. *The Skills of Argument*. New York (NY), Cambridge University Press.

McLaren, B. M., Scheuer, O., De Laat, M., Hever, R., De Groot, R., & Rose, C. P. 2007. Using Machine Learning Techniques to Analyze and Support Mediation of Student E-Discussions. In *Proc. of AIED*. Amsterdam, IOS Press.

Ohlsson, S. 1994. Constraint-based Student Modelling. *Proceedings of Student Modelling*, 167-189, Berlin.

Paolocci, M., Suthers, D., & Weiner, A. 1996. Automated Advice-Giving Strategies for Scientific Inquiry. In *Proceedings of ITS*, 372-381. Berlin, Springer.

Pinkwart, N., Aleven, V., Ashley, K., & Lynch, C. 2006. Schwachstellenermittlung und Rückmeldungsprinzipen in einem intelligenten Tutorensystem für juristische Argumentation. In *Tagungsband der 4. e-Learning Fachtagung Informatik*, 75-86. Bonn, GI.

Reed, C., and Rowe, G. 2004. Araucaria: Software for Argument Analysis, Diagramming and Representation. *International Journal of AI Tools* 14:961-980.

Rekers, J., & Schürr, A. 1997. Defining and parsing visual languages with layered graph grammars. *Journal of Visual Languages and Computing* 8:27-55.

Suraweera, P., and Mitrovic, A. 2002. KERMIT: A Constraint-Based Tutor for Database Modeling. In *Proceedings of ITS*, 201-216. Berlin, Springer.

Verheij, B. 2003. Artificial argument assistants for defeasible argumentation. *Art. Intelligence* 150:291-324.