Diagram interpretation and e-learning systems

Conference or Workshop Item

Diagram Interpretation and e-Learning Systems

Neil Smith, Pete Thomas, and Kevin Waugh

Department of Computing
The Open University
Milton Keynes, MK7 6AA, UK
{n.smith, p.g.thomas, k.g.waugh}@open.ac.uk

Abstract The automatic grading of free-form answers is gaining importance. This paper presents the only system we know of capable of grading free-form diagrammatic answers. Our approach identifies the parts of a diagram that carry specific meaning in that domain, termed minimal meaningful units (MMUs), and matches them with equivalent parts in a model solution. The result of the matching is used to calculate the grade and to generate appropriate feedback. The matching is complicated by errors, omissions, and superfluous items in the student answer. Matching and grading are controlled by parameters which can be adjusted for pedagogic reasons. Our approach has been successfully applied to a variety of diagram types and grades diagrams at least as well as a human marker. We describe a series of tools that allow the easy creation of questions, marking schemes, and diagram editors suitable for online assessment and embedding in a VLE quiz engine.

Keywords: automatic grading, imprecise diagrams, e-learning, e-assessment, interpretation.

1 Introduction

As the popularity of e-learning has grown, so has the use of online quizzes for e-assessment. A major shortcoming in online quizzes is the lack of support for freely-created diagram-based responses, analogous to free-text written answers. In systems with diagrammatic responses, marking (grading) will likely be by a human expert. In addition, students want the ability to draw diagrams in online assessment to support their answers even when diagrams were not requested. Therefore, we set out to produce a system that would automatically grade student produced diagrams.

Marking diagrams without human intervention means that the system must deal effectively with answers that are either only partially correct (in some way) or are equivalent but not identical to the model solution. Dealing with this imprecision is the main substance of our research.

Our work contrasts with semi-automatic marking [1, 2] in which the marking system supports the human marking process. These systems present the human marker with an answer that has not been graded. The human grades that answer and the system applies the same grade to equivalent answers. This process is repeated until all answers are marked and can save considerable time.

Another avenue of research into the use of diagrams in computer science concentrates on using diagrams to express reasoning. For instance, Venn/Euler diagrams can be used to express sets and membership, and existential graphs can be used to express formulae in first order predicate calculus [3]. Much work has been done to determine the logical power of such diagrams and to provide correct deduction systems for diagrammatic formulae [4, 5]. However, the focus in that work is on the role of diagrams as well-formed representations of logical statements; imprecisions in the diagrams would eliminate the reasoning power of the diagrams. This contrasts with our work, where the emphasis is on extracting as much information as possible from an imprecise diagram.

2 Motivation

Students regularly produce *imprecise* diagrams: diagrams that do not match the expected diagram either because they do not follow the expected syntax or because the semantics of the student's diagram do not match the semantics of the solution. There are many ways that a diagram can deviate from the expected diagram revealing an equally wide variety of misunderstandings. As well as providing a summative grade for a submission, we endeavour to provide formative feedback to correct these misunderstandings.

We were initially interested in entity-relationship diagrams as they formed part of our computing curriculum. We noticed a set of common errors and wanted a way to allow students to practice their modelling skills. We analysed other diagrams used for modelling in computing and other disciplines and found they have similar properties. We therefore set out to investigate whether we might be able to treat a whole range of diagrams in a similar fashion. Other diagram marking work often concentrates on one kind of diagram, normally entity-relationship diagrams [1].

Our marking approach is to compare the student-drawn diagram with a model answer. This comparison provides the basis for marking and feedback.

We do not require students to use a tool that enforces syntactic correctness on their diagrams. This allows freedom of expression, and many errors, but provides the opportunity to give feedback on a wider range of misunderstandings.

3 Almost Graph-Like Diagrams

Our analysis of diagrams showed that most diagram types with defined syntactic rules, used for communication and modelling, can be considered graphs, with nodes and arcs. Fig. 1 shows a typical entity-relationship diagram (ERD) using a notation that occurs in one of our modules [6]. It consists of a set of entity types representing objects of interest in the model and conventionally denoted by rectangles with rounded corners. Each entity type has a label to identify it. In a correct ERD, entity types should have unique labels but we do not enforce this constraint in the drawing tool. This is an example of relaxing constraints but which makes the process of comparing two diagrams more difficult.

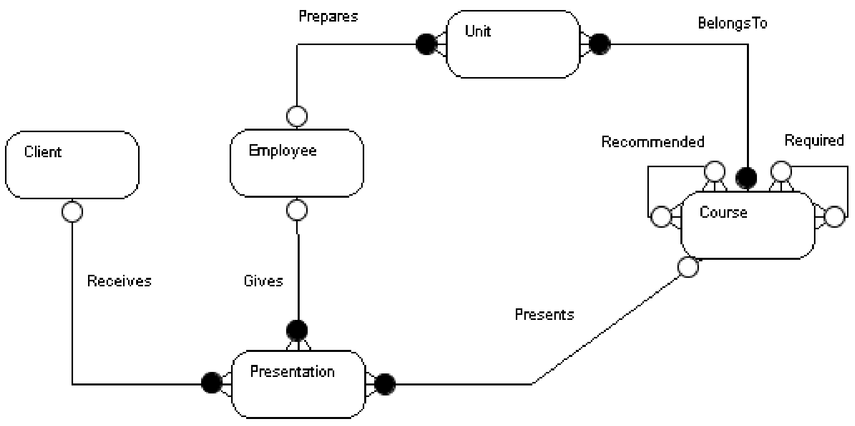


Fig. 1. An entity-relationship diagram.

The entity types (often referred to simply as entities) are connected by piecewise linear lines. The lines represent relationships between entities. A relationship is either binary (between two different entities) or recursive (a relationship between an entity and itself, also known as a self-relationship). A line representing a relationship can have various adornments conveying additional information. In Fig. 1, relationship lines can have ‘crowsfeet’ and/or small circles which convey information about the relationship’s cardinality (how many of one entity type are related to how many of the other entity type) and participation (whether each entity of one type must be related to an entity of the other type). A relationship also has an identifying label. It is clear that this diagram is a graph with adornments.

Fig. 2 shows an ERD with relationships which are represented not by lines but by spatial relationships. Here, rectangles (entity types) occur inside other rectangles, representing subtyping. For example, the *Parent* and *Child* entity types are subtypes of *Person*. These spatial relationships are not labelled. The need to represent these types of relationship means that diagrams are almost graph-like.

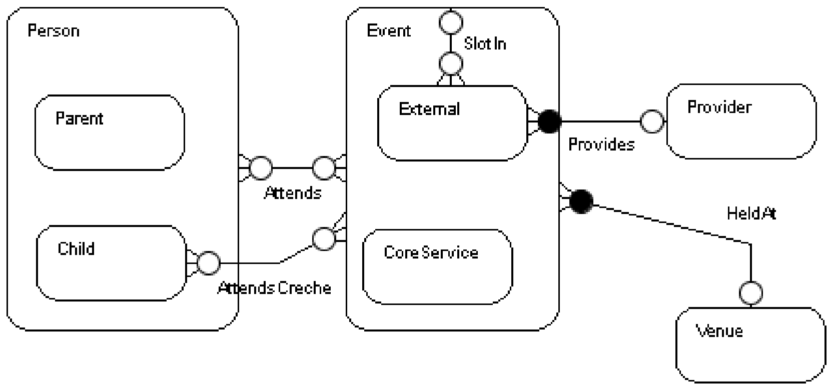


Fig. 2. An ERD with spatial relationships.

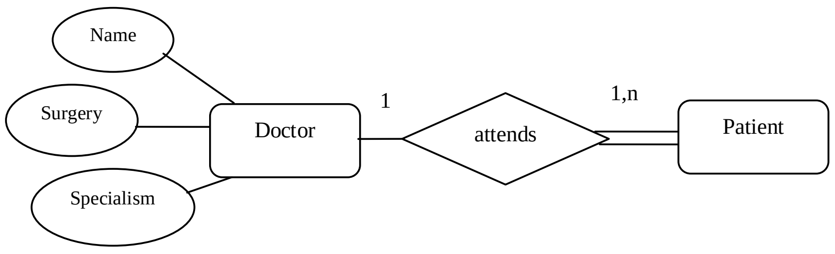


Fig. 3. An ERD in Chen notation.

There are several versions of ERD. For example, Fig. 3 shows an example of the Chen notation [7]. In this notation, the relationship label, *attends*, is placed within a diamond shape and the cardinality and participation attributes are indicated by labels (1 and 1,n) and number of lines respectively. In ERDs, it is also common to represent attributes of entities on a diagram. In Fig. 3, the attributes of a *Doctor* are shown as ovals with labels (*Name*, *Surgery* and *Specialism*) and are associated with the entity type by short, unadorned lines.

Unified Modelling Language (UML) diagrams have less syntactic variation than ERDs. UML class diagrams (Fig. 4) show great similarity with ERDs. Things of interest (classes) are represented by rectangles and relationships (called associations) between them are denoted by lines. The lines have adornments that play similar roles to those in ERDs. UML class diagrams have several types of relationship, all denoted by lines. The relationships are distinguished by different types of line (solid/dashed) and different adornments. Spatial relationships do not occur in class diagrams. It is quite clear that UML class diagrams can also be represented by graphs with the nodes representing classes and edges representing associations.

UML sequence diagrams (Fig. 5) are more complex than either ERDs or UML class diagrams and illustrate a different property. A UML sequence diagram has two kinds of things: *objects* represented by labelled rectangles appearing at the top of the diagram and *activations* represented by long, thin vertical rectangles appearing below the objects. There are sub-activations represented by smaller rectangles attached to the sides of the larger activations. There are four kinds of relationship in this diagram.

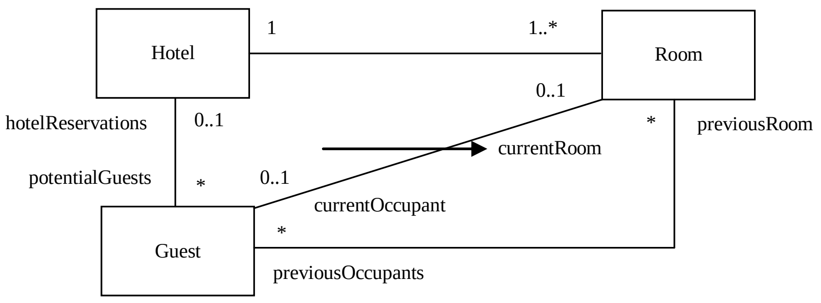


Fig. 4. A UML class diagram showing multiplicities and role names.

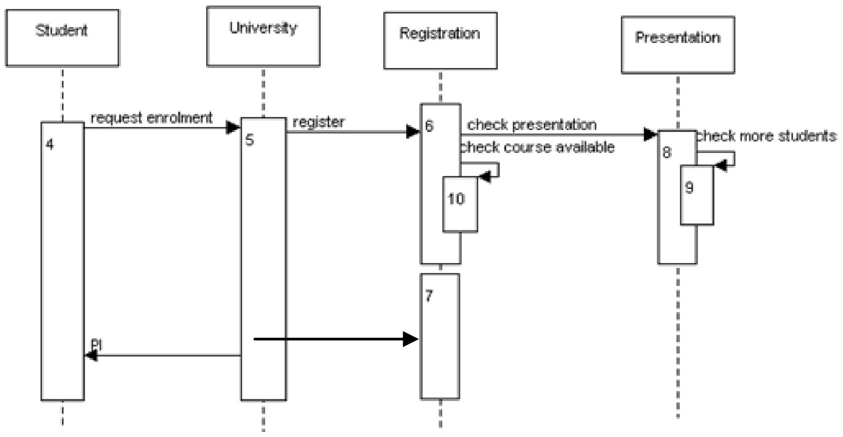


Fig. 5. A UML sequence diagram

- activations are associated with objects, indicated by a vertical, dashed line (known as a timeline),
- activations are associated with one another by arrowed horizontal lines (representing the flow of a message from one object to another),
- activations are associated with sub-activations by an arrowed line,
- activations associated with the same object are in a sequence denoted by their position on the timeline.

It is also possible for there to be associations between an activation belonging to one object and a different object (no example shown on the figure). A relationship between an activation and one of its sub-activations is similar to a recursive relationship in an ERD.

A sequence diagram contains the notion of time which increases downwards. The inclusion of time in sequence diagrams introduces a global spatial relationship which is not present in either ERDs or class diagrams. Again, sequence diagrams are almost graph-like, with the additional spatial representation of time.

Our final example, flow diagrams, comes from biology. An example of a flow diagram is shown in Fig. 6 and depicts the flow of hormones between glands. In this type of diagram there is only one type of object of interest (glands), each represented by a rectangle. The arrowed lines represent the flow of hormones. The significant new feature is the fact that flows can affect flows, indicated by an arrow from a gland (OVARY) to another flow (FSH).

These different diagram types, drawn from different domains, can all be expressed as graph-like structures, with some extensions. This indicates that our approach is widely applicable.

4 Comparing Diagrams

Our approach to automatic marking is based on comparing diagrams: a student diagram is compared with a model solution diagram. This comparison forms the basis

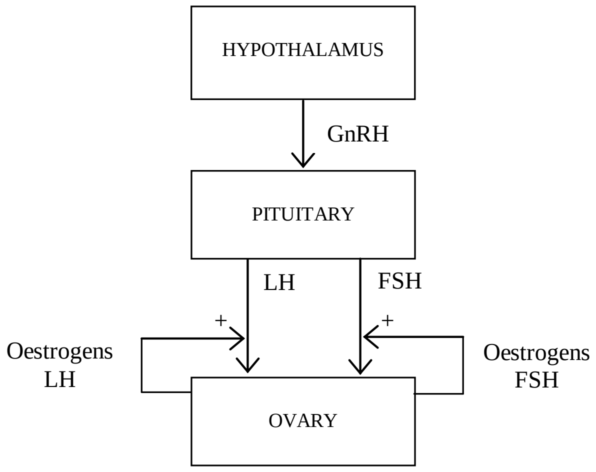


Fig. 6. A Biological flow diagram.

of the marking and feedback generation, as the marker will have identified the parts of the student's diagram that match the model answer and those that do not. In general, there could be more than one model solution; in such cases, the answer is matched with each solution in turn and marking and feedback is based on the best-matching solution. Without loss of generality in what follows we shall assume that there is one model solution.

4.1 Definitions

A concrete diagram, belonging to some *domain* of interest, is constructed from two basic types of *drawn element*: two-dimensional figures which we refer to as *boxes*, and (piecewise linear) lines which we refer to as *links*. Boxes and links can have *attributes* (also known as *adornments* in the case of links).

There are certain combinations of drawn elements that we describe as *minimal meaningful units* (MMUs). An MMU is defined as a partial diagram belonging to the domain of interest which, if any element is deleted, no longer has meaning in that domain. MMUs can be aggregated into *meaningful units* (MUs). For example, a box on its own is an MMU and a link together with the two boxes that it connects is also an MMU.

An *answer* is the diagram produced by a student, which is likely to be imprecise. The *solution* is a model solution diagram. We are interested in comparing an answer with one or more solutions.

4.2 Similarity Measures

Diagrams are compared by identifying and comparing the MMUs in each diagram. The first step is to identify those MMUs in the answer that ‘correspond’ to MMUs in the solution. However, due to imprecision in the student diagram, an MMU in the answer may correspond with an MMU in the solution but not be identical (for example, because the attributes differ in some way). Therefore, we use a measure of similarity between MMUs.

A similarity measure is a value in the range [0..1] in which 0 represents no similarity (meaning that the MMUs have nothing in common) and 1 represents complete similarity (the MMUs are identical). In most diagrams types there are two MMUs: objects, and relationships between objects (including the terminal objects). MMUs may overlap. The similarity between two objects is based on the similarity of their label identifiers and their attributes, and the similarity between two relationships is based on the similarity the objects that they relate and of their adornments. To date, we have assumed that the similarity of objects and relationships of different types is zero. This has proven to be sufficient for our purposes, but is an area for further research.

The similarity of two objects, $sim(o_1, o_2)$, is the weighted sum of the similarities of their identifying labels, $sim(l_1, l_2)$ and their attributes, $sim(a_1, a_2)$:

$$sim(o_1, o_2) = v * sim(l_1, l_2) + (1\text{-}v) * sim(a_1, a_2), \quad 0 \leq v \leq 1$$

Typically, the similarity of the labels is of more importance in determining the similarity of two objects than their attributes ($v > 0.5$).

The computation of the attribute similarity, $sim(a_1, a_2)$, starts by determining which of the attributes of an object in one diagram corresponds to which of the attributes in the other diagram and then finds the average similarity between the matched attributes (we assume that attributes have equal importance). Since the number of attributes of an object in an answer diagram can be different from the number of attributes in the corresponding object in the solution, we choose a correspondence between attributes that maximises the average similarity. If the answer's object has more attributes, the best match of answer to solution attributes is chosen and there is no penalty for extraneous attributes. Hence, if $a_{1,\mathrm{i}}$ i=1..n are the attributes of the specimen object and $a_{2,j}$ j=1..m, are the attributes of the student object,

$$sim(a_1, a_2) = (\textstyle\sum_i sim(a_{1,i}, a_{2,j}))/ n$$

where $a_{2,j}$ is the best match for $a_{1,i}$. In the final match of objects, each object in each diagram is matched with at most one object in the other diagram.

The similarity of two relationships, $sim(r_1, r_2)$, is the weighted sum of the similarities of the objects (or other relationships) which they relate, $sim(o_1, o_2)$, and the similarity of their attributes/adornments in the same relative (corresponding) position, $sim(a_1, a_2)$.

$$sim(r_1, r_2) = w * sim(o_1, o_2) + (1\text{-}w) * sim(a_1, a_2) \qquad 0 \leq w \leq 1$$

If a relationship has an identifying label, it is treated as one of the attributes. Unlike objects, relationship attributes can have different importance when calculating similarities. Determination of weights is discussed later.

$$sim(a_1, a_2) = \textstyle\sum_i w_i * sim(a_{1,i}, a_{2,j}), \text{ where } \textstyle\sum_i w_i = 1$$

When two relationships are compared, the adornments are compared by relative position. For example, the adornments at the ends of both relationships are compared and would have a similarity of 1 only if the adornments were of the same type and had the same value. All positions are compared and the similarity is a weighted sum of the similarities of each position. Spatial relationships are treated similarly.

4.3 The Special Case of Labels

One of the most significant elements on our kinds of diagram is the identifying label. When objects do not have any attributes, the label is the only item of information on which a similarity can be based. In the case of relationships there can be zero or more labels. But comparing labels poses several problems:

1. Synonymous labels can be very different in form but must be viewed as identical.
2. Misspelling, incorrect hyphenation, and non-standard abbreviation can obscure the equivalence between labels in the answer and solution.
3. In several domains, the structure of a label can be a short phrase which requires interpretation. For example, *CoreService* is the name of an entity in Fig. 2 and *check course available* is a label on a message in Fig. 5.

By default, we model object labels as noun phrases and relationship labels as verb phrases. This can be overridden by the question setter.

Subsequent processing of labels identifies their main nouns or verbs which are then compared using edit distance, taking synonyms into account. This process also attempts to deal with punctuation in a phrase (including hyphenation) as well as abbreviation.

The most important issue concerns synonyms. The labels occurring in a model solution are only one manifestation of the labels that can legitimately appear. We include commonly occurring synonyms explicitly in the mark scheme (noting that specialist synonyms can exist in the domain of the question). The multiple synonym problem [8] is ameliorated by stemming (to reduce words to a canonical form) and similarity measures (to estimate the closeness of words). In addition, the scenarios used in our questions are designed to limit the growth in synonyms (see [1] for more in scenario design). However, synonyms remain an issue. One way to deal with them is to examine large corpora of student answers to determine more synonyms [9].

5 The Matching Process

The structural similarity between two diagrams is determined by finding the best overall correspondence of MMUs. The process begins by comparing objects of the same type. For example, in ERDs, there is only one type of object (the entities) so each entity is compared with every other entity. As far as possible, each MMU in the answer is matched with an MMU in the solution to give a *diagram correspondence*. Omissions and errors in the answer will mean that not every MMU in the answer will be matched, and vice versa. The chosen diagram correspondence is the one which maximises the sum of the similarities of matched MMUs. This process is performed separately for each type of MMU.

The matching process is modified in three ways better to deal with imprecision.

1. Having determined the correspondence between objects of a specific type, if there are objects in the answer and the solution which remain unmatched, the contexts of these unmatched objects are compared to see if there is sufficient evidence for further matches. The context of an object is the set of objects to which it is related. This is useful in determining correspondences where labels are synonymous but the system has no *a priori* knowledge of this.
2. An initial correspondence of relationships of a given type is deduced based on the related objects alone (i.e. without considering adornments). This produces a *plausible match* of relationships. This recognises that the most significant determinant of whether two relationships match is the pairs of objects they relate. Adornments are used to update the similarity measure of two relationships once a plausible match has been established.
3. In some domains, there are well-known semantic equivalences between diagrams. For example, in ERDs there are circumstances where a single ‘many-to-many’ relationship can be decomposed into two ‘one-to-many’ relationships without loss of meaning. We call such equivalences *clichés*. Therefore, the system will examine the student answer to see whether it contains clichés which can match with MMUs in the specimen solution.

Once a correspondence between the student answer and model solution has been determined, a given mark scheme, based on MMUs, is applied (a mark scheme is a description of how marks should be allocated to matched MMUs). The correspondence is also the basis upon which feedback is given.

5.1 Weights and Thresholds

A system of thresholds is used to judge whether or not two aspects are sufficiently similar to be considered to match (that is, where there is sufficient evidence to suggest that a student’s relationship should considered an attempt at drawing a relationship found in the expected diagram). For example, if the similarity between two MMUs is above (or below) a threshold, the MMUs will be considered identical (or completely different). Thresholds are also used in the marking algorithm to determine whether or not to award marks for a particular aspect of a student answer.

A weight represents the relative importance of some aspect of a diagram. The importance is normally determined by the *teacher*, the person who wishes to impose a particular pedagogy on the system. For example, if the teacher wishes to emphasise the cardinality of a relationship, the weight for that adornment can be increased. In the absence of such teacher selected weights, the system will assign weights equally to all aspects of a diagram. Weights have a value in the range [0..1].

Weights can also be determined by experiment, for example, to optimise the agreement of the marking system with a sample of human marked diagrams.

6 Supporting Applications

In this work, we envisage diagrams being drawn with the aid of software drawing tools rather than hand-drawn sketches. However, while such tools would produce diagrams conforming to the syntax of a specific domain they would not constrain students to that syntax. They can be specified by the teacher with an appropriate amount of freedom of expression for the student according to the chosen pedagogy.

Each domain will have a drawing tool and a marking tool. The drawing tool is shown in Fig 7 and can run either stand-alone or as an applet. We have an application which enables the teacher to specify the domain by specifying the expected types of MMU, adornments and weights. This information is used to drive a generalised drawing tool and a generalised marking engine. The teacher can create different versions of the drawing/marking applications to suit their pedagogy.

The matching and marking processes depend upon the existence of one or more model solutions and a marking scheme (information on how to allocate marks). An application supports the teacher in developing questions, model solutions, mark schemes, and feedback.

7 Effectiveness

To evaluate the performance of the marking algorithm we compared the performance of the automatic marking tool against the grades awarded by experienced human markers. Our early experiments were with ERDs. Initially we used relatively small sets of data to determine how well our approach was working [10]. These early results gave us confidence that marking tool was working sufficiently well for a more detailed evaluation.

To evaluate the marking tool we constructed a 600 diagram corpus, complete with grades awarded by the human markers, and moderated grades (checked for consistency and accuracy by an independent marker). The diagrams were submitted as student answers in an invigilated examination. Our “gold standard” for marking was the moderated grade awarded to the student answers by the independent human markers. The corpus was split to give a development data set (200 diagrams) and an evaluation data set (394).

An additional corpus, with 169 diagrams produced by students taking a second examination, extended the range of ERD features that the marking algorithm used. This corpus was split to give a development set of 72 diagrams and an evaluation set of 97 diagrams. In both corpora the diagrams were graded from 0-7, with grades rounded to the nearest half mark.

In the first corpus, the automatic marker was within 0.5 marks of the moderated human grade in 91% of cases with only 3 diagrams with the maximum difference of 1.5 marks. Where the automated marker and human marker disagreed there was no evidence of bias in the automated marker’s performance. In the second corpus 80% of grades differed by no more than 0.5 marks, with 2% of diagrams having a difference of 2 or more marks. Again there was no evidence of bias in the automatic marker. In

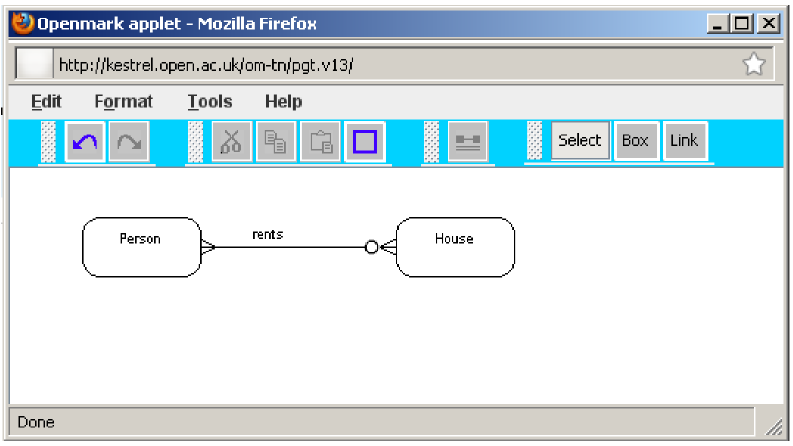

Openmark applet - Mozilla Firefox
http://kestrel.open.ac.uk/om-tn/pgt.v13/
Edit
Format
Tools
Help
Select
Box
Link
Person
rents
House
Done

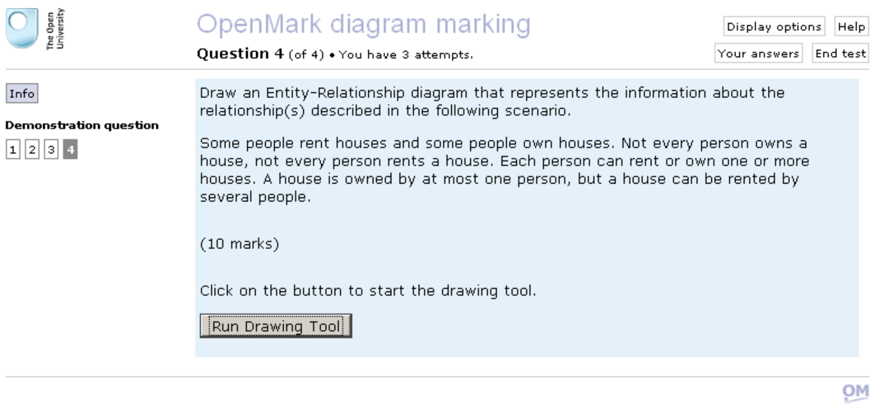
Fig. 7. The drawing tool.

both cases the automatic marker was found to be a better marker (gave grades closer to the moderated, "gold standard", grade) than the original human markers. [11, 12].

8 Deployment

We have built two applications that have been used by students [13]: one deals with entity-relationship diagrams, one with UML sequence diagrams. Both applications have a set of questions of increasing difficulty that a student can work through. The application marks the student's attempts and provides feedback in a graphical manner. An early prototype can be examined by downloading it from our website at http://mct.open.ac.uk/Diagrams. Fig. 7 illustrates the interface to the drawing tool.

More recently, a prototype drawing tool and automatic marking engine have been



Question 4 (of 4) • You have 3 attempts.

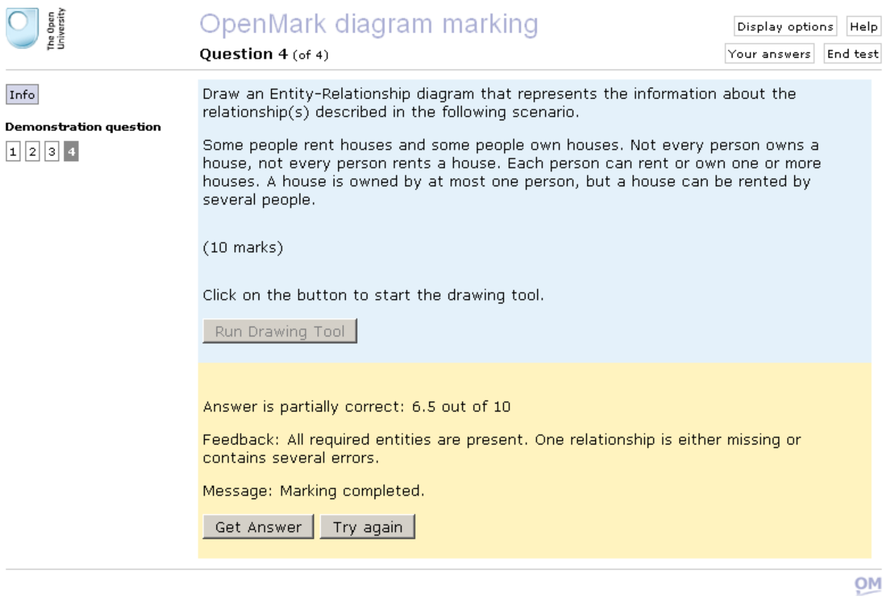
Draw an Entity-Relationship diagram that represents the information about the relationship(s) described in the following scenario.

Some people rent houses and some people own houses. Not every person owns a house, not every person rents a house. Each person can rent or own one or more houses. A house is owned by at most one person, but a house can be rented by several people.

(10 marks)

Click on the button to start the drawing tool.

Fig. 8. A drawing question in Moodle/OpenMark



Question 4 (of 4)

Draw an Entity-Relationship diagram that represents the information about the relationship(s) described in the following scenario.

Some people rent houses and some people own houses. Not every person owns a house, not every person rents a house. Each person can rent or own one or more houses. A house is owned by at most one person, but a house can be rented by several people.

(10 marks)

Click on the button to start the drawing tool.

Answer is partially correct: 6.5 out of 10

Feedback: All required entities are present. One relationship is either missing or contains several errors.

Message: Marking completed.

Fig. 9. Feedback from the marking engine.

incorporated into the Open University's Moodle VLE, via the OpenMark quiz engine. Fig. 8 shows a typical question within an OpenMark quiz. The 'Run Drawing Tool' button launches an applet containing the appropriate drawing tool for the question, illustrated in Fig. 7. Having drawn an answer, the student submits the diagram to the making engine.

In formative mode, the marking engine returns feedback which is displayed beneath the question in the student's browser as illustrated in Fig. 9. The figure also shows that the student can have multiple attempts at the question and, if unsuccessful, view the model solution.

9 Conclusions and Future Work

We have built an automatic marking engine for a class of graph-based diagrams which gives good performance on relatively constrained questions of the type we use in formative assessment. Our experiments with entity-relationship diagrams and UML sequence diagrams have shown that good agreement with human markers is attainable. Nevertheless, we want to perform similar experiments with more diagram types and less constrained questions.

Our work to date has been based on developing bespoke diagram parsers for a small number of domains and subsequently generalizing what we have learned into tools to support a wider range of diagram types. While this is effective, the approach limits the ease with which we can apply our diagram understanding processes to other domains. For example, chemical reaction diagrams contain a significant degree of information encoded in the relative positioning of elements in the diagram. Our work has, however, informed us about future directions and how we might more easily extend it to more domains. Therefore, we have elected to reformulate our approach as a constraint-based parser [14, 15]. To deal with the imprecision inherent in the diagrams we are addressing, we propose to use soft constraints [16] to indicate which features in the diagram can be malformed while still allowing the diagram to be parsed and interpreted.

References

1 Batmaz, F., Hinde, C.J.: A Web-Based Semi-Automatic Assessment Tool for Conceptual Database Diagrams. In Proceedings of the Sixth IASTED International Conference on Web-Based Education, Chamonix, France, pp. 427—432. (2007)

2 Higgins, C.A., Bligh, B.: Formative Computer Based Assessment in Diagram Based Domains. In Proceedings of the 11th Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE), Bologna, Italy, pp. 98—102. (2006)

3 Shin, S.-J.: The Logical Status of Diagrams, Cambridge, UK: Cambridge University Press. (1994)

4 Howse J., Stapleton G.: Visual Mathematics: Diagrammatic Formalization and Proof In Proceedings of International Conference on Mathematical Knowledge Management, (2008)

5 Stapleton, G., Taylor, J., Thompson, S., Howse, J.: The expressiveness of spider diagrams augmented with constants Journal of Visual Languages and Computing 20(1) doi:10.1016/j.jvlc.2008.01.005 (2008)

6 M359, Relational databases: theory and practice, Open University. (2009)

7 Chen. P.: The Entity-Relationship Model - Toward a Unified View of Data. ACM Transactions on Database Systems 1(1): 9-36. doi:10.1145/320434.320440. (1976)

8 Jayal, A. and Shepperd, M.: 2008. The Problem of Labels in e-Assessment of Diagrams, *ACM J. of Educational Resources in Computing*, 8(4), Article 12. (2008)

9 Jordan, S.: Assessment for learning: pushing the boundaries of computer based assessment. Practitioner Research in Higher Education, *3*(1), pp11-19. (2009)

10 Thomas, P.G., Waugh, K, Smith, N.: Experiments in the Automatic Marking of ER-Diagrams. In Proceedings of ITiCSE 05, pp. 158-162. (2005)

11 Thomas, P.G., Smith, N and Waugh, K.: Automatically assessing graph-based diagrams, J. Learning, Media & Technology, 33(3), pp. 249-267. (2008)

12 Thomas, P.G., Smith, N., and Waugh, K: Automatically assessing diagrams. In Proceedings IADIS International Conference e-Learning 2009, (2009)

13 Thomas, P.G., Waugh, K., and Smith, N.: Tools for learning and automatically assessing graph-based diagrams. In Research Proceedings of ALT-C 2007, pp. 61-74. (2007)

14 Henning, C.: CHR grammars, Theory and Practice of Logic Programming 5(4-5): 467-501, DOI:10.1017/S1471068405002395 (2005)

15 Jansen, A., Marriott, K., Meyer, B.: CIDER: A component-based toolkit for creating smart diagram environments. In Proceedings of the 2003 International Conference on Visual Languages and Computing (VLC 2003), pp 353-359. (2003)

16 Bistarelli, S., Frühwirth, T., Marte, M.: Soft constraint propagation and solving in CHRs, In Proceedings of the 2002 ACM symposium on Applied computing, DOI: 10.1145/508791.508793 (2002)