Technical Report No: 2004/01

Grading Diagrams Automatically

Pete Thomas

30th January 2004

Department of Computing
Faculty of Mathematics and Computing
The Open University
Walton Hall,
Milton Keynes
MK7 6AA
United Kingdom

http://computing.open.ac.uk

Grading Diagrams Automatically

Pete Thomas
Computing Department
Open University
Milton Keynes
MK7 6AA

Abstract

This paper describes a feasibility study into the marking (grading) of diagrams automatically. The diagrams were produced by students during an on-line examination using a simple drawing tool. The students' examination answers, which included a diagram, were submitted over the Internet to an automatic marking tool for grading. This paper describes the methodology adopted by the diagram marking tool and comments on the results achieved. Whilst only a small number of diagrams were available for marking, the grades awarded by the automatic marker were sufficiently close to the marks awarded by four independent human markers to suggest that marking diagrams automatically, for small diagrams at least, is feasible and should be pursued further.

Introduction

For several years we have been investigating the automatic grading of examination papers in which student answers are all free-form text. That is, the questions require answers that are akin to essays and are not multiple choice questions [Thomas, 2001]. The students have typed their answers into a web form using a PC at home and submitted their answers via the Internet to a server for grading. This electronic examination system gives immediate feedback by providing a grade and textual feedback for each answer. The results have been encouraging [Thomas, 2002].

However, until recently, the system would only deal with textual answers; there was no provision for the creation or marking of diagrams. We felt that this was a limitation of the tool. In addition, the introduction of a diagrammatic marking component raises a number of interesting research questions about the correspondences between diagrammatic and sentential forms.

Both grammatical and statistical techniques have been applied successfully to understanding and manipulating sentential language [Gazdar & Mellish, 1989; Manning & Schütze, 1999]. Many automatic marking systems for text generally take a statistical approach: our own work looks for appropriate keywords in answers [Thomas, 2001], and the commercial Intelligent Essay Assessor system [Foltz *et al.*, 2000] uses latent semantic analysis for marking. In contrast, the existing literature on reasoning with diagrams [Mariott *et al.*, 1998; Blackwell & Engelhardt, 2002] concentrates on rule-based and grammatical representations of diagrams and we have not seen this work applied to the grading process.

In this work, we take some first steps towards applying statistical techniques to diagram understanding. Therefore, this work represents an early step in the automatic marking of diagrams, and seems to be the first application of statistical techniques to diagram understanding.

The remainder of this paper describes how our initial diagrammatic marking tool was developed, and some early results of its use. In the next section we briefly describe the specialised drawing tool that was developed for this experiment. The remainder of the paper takes a more detailed look at how the diagram is represented, how a marking scheme is represented, and how the diagram is combined with the marking scheme to generate a mark for the student.

The drawing tool

Given that the student volunteers did not require the use of a drawing tool of any description for their course, we felt that a simple tool that would require minimal familiarisation would be appropriate. The tool should also facilitate the drawing of diagrams appropriate to the examination – in order to minimise the time of use during the examination. In addition, we are primarily interested in diagrams that commonly occur in the teaching of computing (such as E-R and UML diagrams).

Therefore, we adopted a very simple approach in which the drawing tool supported the creation of diagrams that are limited to two simple geometric objects that we have named boxes and links. A box has a rectangular shape and can contain text. A link is a directed line (arrow) and can also be associated with some text. The text associated with each object can be used for labelling and explanation purposes. Furthermore, the role of a link is limited to joining two boxes and can be used to indicate a relationship between them. For example, in a UML class diagram a box labelled with a name would represent a class, and a link, also labelled, would represent an inheritance relationship between two classes as illustrated in Figure 1.

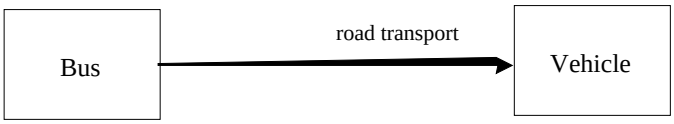

Figure 1 A simple inheritance relationship.

Objects are placed on the drawing canvas using a drag-and-drop technique. Boxes can be placed anywhere on the canvas, but links must join two boxes. A link is constructed with the mouse by left-clicking inside one box, dragging the cursor to another box and releasing mouse button.

A range of editing and file handling facilities are also provided. However, if the user wishes to delete a box from the diagram, any associated links will also be removed (but not before the user has been prompted with an appropriate warning). A link can be removed at any time.

Thus, a link is associated with two boxes and has a direction and text. A box has text and can be associated with zero, one or more other boxes (but not itself, in the current implementation) through a set of links. The location and size of each box and each link on the drawing canvas is also recorded.

Hence, a box has the following attributes:

- text, which may be empty,
- a location on the drawing canvas, consisting of the (x,y) co-ordinates of its top-left-hand corner,
- a size, consisting of its width and height,
- a set, possibly empty, of references to its associated links.

A link has the following attributes:

- text, which may be empty,
- a location on the drawing canvas, consisting of the (x,y) co-ordinates of the top-left-hand corner of its bounding rectangle,
- a size, consisting of the width and height of its bounding rectangle,
- references to its two associated boxes, with the start box reference given first.

Example

Figure 2 shows an example of a drawing constructed with the drawing tool that will be used throughout the remainder of the paper. The drawing represents a solution to the examination question given in Figure 3. The diagram illustrates a computer processor architecture known as a 4-stage pipeline. The boxes represent the four stages of execution of a machine code instruction (fetch, decode, execute and write) and the links shows the order in which the stages occur. In this example, there are two pipelines and an association labelled “forward” between the two execution stages.

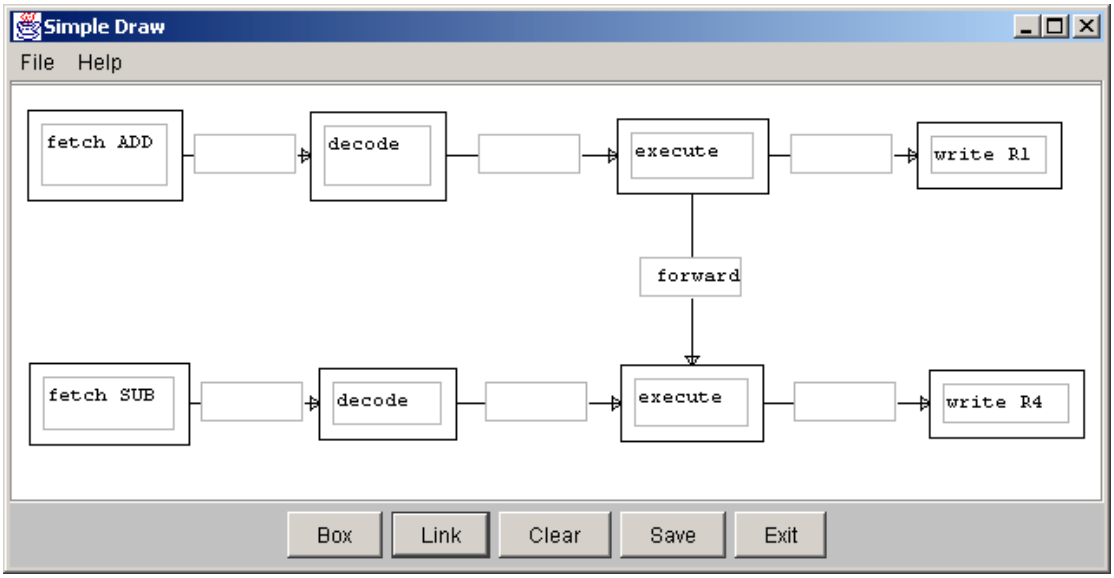

Figure 2 A drawing of two pipelines with a link between them.

Question

Use the drawing tool to draw a diagram that illustrates how the data hazard inherent in the execution of the pair of instructions

ADD R2, R3, R1

SUB R1, R5, R4

by a 4-stage pipeline, can be overcome.

Figure 3 An example examination question

Inherent in the solution is the meaning of the instructions given in the question. That is, both instructions identify three registers and specify that a binary operation (ADD or SUB) is to be performed on the contents of the first two registers with the result being stored in the third register (reading from left to right). The data hazard is the fact that the execution stage of the second instruction requires the value of the result of the first instruction (to be found in register R1) before the completion of the write to R1 stage of the first pipeline.

In the examination, 4 marks were available for a student answer. The human markers were given the marking instructions shown in Figure 4.

Mark scheme

Award marks as follows:

1 for each pipeline,

1 for the link between execute ADD and execute SUB, and

1 for the notion of forwarding the contents of the ALU to replace the contents of R1 (look for the word forward in this link).

Figure 4 The marking scheme for answers to the question in Figure 3.

The goal of the automatic marker

In the above example, the goal of the automatic marker was to recognise:

1.a 4-stage pipeline for the ADD instruction

2.a 4-stage pipeline for the SUB instruction

3.the forwarding of the result of the execute stage of the ADD instruction pipeline to the execution stage of the SUB instruction pipeline represented by a link.

and award marks according to the given marking scheme.

Representing a diagram

One of the fundamental questions facing us in this work is what is the diagrammatic equivalent of a "word". We take the view that, in a sentential representation, a word is the smallest meaningful unit of discourse. In the diagrammatic domain, we are also interested in the smallest meaningful unit. In this work, we take this unit to be an *association*, which is a pair of boxes connected by an arrow (figure 5). Given this definition, we treat a diagram as a (possibly overlapping) set of associations.

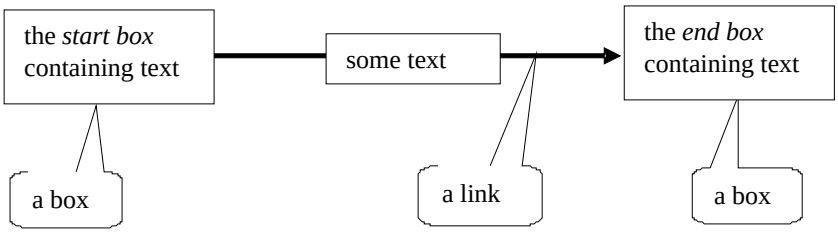

Figure 5 An *association*.

An association can be conveniently described by a constraint multiset grammar (CMG). A CMG consists of a number of productions of the form:

$P \rightarrow P_1 \ldots, P_n$ **where** (
Constraints
) {
Attributes assignment
}

which states that token P can be rewritten by tokens $P_1, \ldots, P_n$ provided that $P_1, \ldots, P_n$ satisfy all *Constraints*. Each token P has a number of *Attributes*, values that are assigned whenever a production is applied, as specified in *Attributes assignment*. Hence, an association can be described by:

Association → Link, Weight **where** (
exists Box_1 Box_2
attached (Link.start, Box_1.area) &
attached (Link.end, Box_2.area)
) {
Association.from = Box_1 &
Association.to = Box_2 &
Association.text = Link.text &
Association.weight = Weight
}

where,

Link → Arrow, Text **where** ( ) {
Link.start = Arrow.start &
Link.end = Arrow.end &
Link.text = Text
}

Box → Text, Area, Weight **where** ( ) {
Box.text = Text.string &
Box.area = Area &
Box.weight = Weight
}

Arrow → $Location_1$, $Location_2$ **where** ( ) {
Arrow.start = $Location_1$ &
Arrow.end = $Location_2$
}

Weight → double **where** ( ) {
Weight.value = double
}

Area → Location, Width, Height **where** ( ) {
Area.x = Location.x &
Area.y = Location.y &
Area.width = Width.value &
Area.height = Height.value
}

Location → int_1 int_2 **where** ( ) {
Location.x = int_1 &
Location.y = int_2
}

The function *attached* (Location, Area) returns *true* if the Location is within the Area. That is,

Area.x <= Location.x <= (Area.x + Area.width) &
Area.y <= Location.y <= (Area.y + Area.height)

However, the drawing tool enforces these constraints by ensuring that a link cannot exist without there being two boxes for which *attached* is *true*.

Hence, the upper pipeline in Figure 2 can be described by the following production.

Pipeline → $Association_1$, $Association_2$, $Association_3$ **where** (
$Association_1$.to = $Association_2$.from &
$Association_2$.to = $Association_3$.from &
$Association_1$.from.text.string = “fetch ADD” &
$Association_2$.from.text.string = “decode” &
$Association_3$.from.text.string = “execute” &
$Association_3$.to.text.string = “write R1” &
) {
Pipeline.assoc1 = $Association_1$ &
Pipeline.assoc2 = $Association_2$ &
Pipeline.assoc3 = $Association_3$ &
}

Marking an answer

In general terms, the act of marking a student’s attempt at answering a question, which we shall henceforth refer to as the *Answer*, is one of comparing the Answer with the Solution and assigning a mark to the Answer. The mark will be a proportion of the overall mark allocated to the Question and represents the level of similarity between the Answer and the Solution. In our experiments with the automatic marking of textual answers, we have found that there can be several acceptable (“correct”) solutions to a question and an Answer must be compared with each one before awarding a mark. In the following discussion, we shall ignore this minor complication and discuss the situation of comparing an Answer with a single Solution.

For simplicity in the feasibility experiment, all diagrams were treated as a set of distinct, unrelated associations rather than representations of a higher level structure consisting of two pipelines related by a single association. The comparison was made on the basis of matching each association in the Answer with each association in the Solution and deriving a similarity measure (a value in the range [0, 1]) for each pair. Each value was multiplied by the weight of the Solution association, and a matrix of resulting values was created. Then a search was made for those entries in the matrix which maximised the total mark. This maximum mark was rounded up to the nearest half mark and the result taken to be the score for the Answer.

The measure of similarity between two associations was derived by comparing the three components of each association – the text of the association and the texts of the two associated boxes. As indicated above, without specific marks being given to each box and link in the Solution, marks would be automatically apportioned equally between components. Hence, the similarity of associations reduces to the similarity of up to three pairs of text.

Given that the textual content of diagrams tends to be quite small, a simple approach to deriving the similarity of a pair of texts was adopted. The similarity of a pair of texts, a value in the range [0..1],

was obtained by calculating the number of words in common in the two texts having removed stop words and punctuation, and making provision for synonyms.

It had been assumed that students would represent the ordering of the stages in a pipeline with links. However, they preferred not to use links but to show this ordering by the relative positioning of boxes on the canvas as shown in Figure 6.

Figure 6 Order indicated by relative position

Therefore a mechanism was required to infer the 'missing' structure. To solve this problem, it was assumed that two adjacent boxes are associated. That is, there are two associations (both without text) between two boxes, one in either direction. In Figure 7, the vertices "fetch" and "decode" are adjacent whereas vertices "fetch" and "execute" are not adjacent. The criterion for the adjacency of two vertices was whether or not there is a third vertex that lies between them. To formalise this notion we introduce the idea of a *region of influence* of a vertex illustrated in Figure 7.

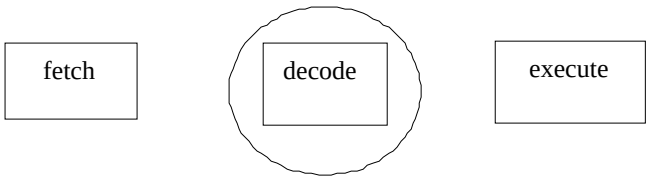

Figure 7 A region of influence.

That is, there are one or more regions of a diagram around a box through which links between other boxes should not travel if they are to be considered adjacent. For example, Figure 8 shows a circle of influence around the box labelled "decode" for which it is not possible to draw a straight line between the boxes labelled "fetch" and "execute" without intersecting the circle. With this criterion, the boxes "fetch" and "execute" would not be considered adjacent.

For simplicity, the notion of a region of influence was implemented as illustrated in Figure 8.

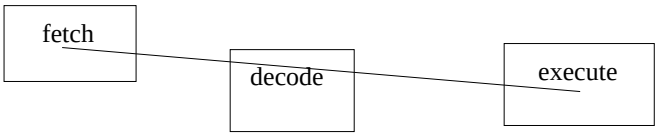

Figure 8 The implementation of a region of influence.

That is, two boxes were considered adjacent if a line drawn between their centres does not intersect the area occupied by a third box. Thus, in Figure 8, the boxes labelled "fetch" and "execute" are not adjacent.

This notion of a region of influence can introduce a large number of inferred links, some of which are reasonable, but others are not. For example, if the diagram in Figure 2 were to be analysed by this mechanism, a number of invalid links would be inferred including one between the boxes labelled "fetch ADD" and "fetch SUB" and one between "fetch ADD" and "write R4". Whilst the former seems a reasonable deduction, the latter does not, and it would seem sensible, therefore, to introduce a limit on the relative distance between adjacent boxes. This was not done in the prototype because the marking algorithm searched all links (actual and inferred) in an Answer looking only for those which best matched the links in the Solution. All those links in an Answer that were not selected were ignored.

Testing the automatic marker

An attempt at marking real student diagrams was made in the context of an on-line mock exam on a postgraduate computing course (April 2003). The students used the simple drawing tool to create their answers to the question posed above. Four independent human markers also used the drawing tool to

view the student diagrams and assign marks. The results of this trial are given in Table 1 (only 5 students attempted the question). The maximum mark for the question was 4.0.

Student	Human Markers (average)	Diagram Marking Tool
1	2.0	2.0
2	3.0	4.0
3	4.0	3.0
4	2.5	4.0
5	2.0	2.5
Mean	**2.7**	**3.1**
St. Dev	**0.837**	**0.894**

Table 1 Human v automatic marking

The main characteristic of the results is that there was no major difference between the human and automatic marks for any student. In addition, given the optimistic approach to inferring associations in the automatic marker, it is not surprising that the automatic marker awards slightly higher marks on average. For example, the drawing produced by student 4 is shown in Figure 9.

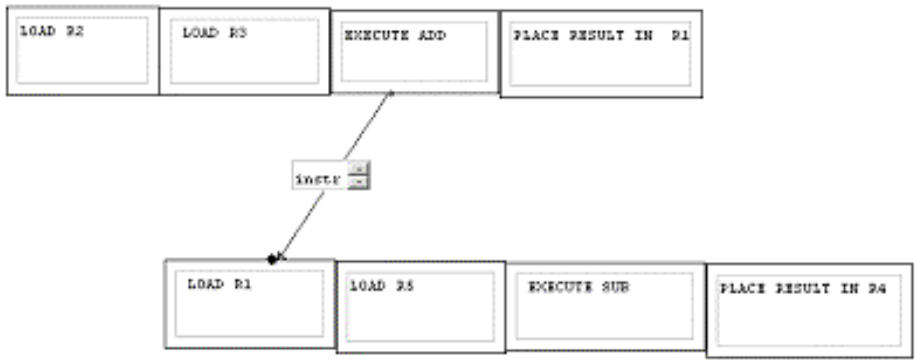

Figure 9 A student drawing

This drawing clearly shows two pipelines (but the first stages are incorrectly labelled) and the process of forwarding the result of the execute stage of the ADD instruction. However, the forwarding is not to the correct stage of the SUB instruction. One could be critical of the human markers by saying that the answer is worth more than 2.5 (out of 4), but it is certainly not completely correct and should not receive full marks as rated by the automatic marker. In this example, the inferred associations between the first two boxes in each pipeline would be given more credit by the automatic marker than the human markers. The final automatically generated mark also benefited from the rounding up process.

Despite the small sample, the results are sufficiently encouraging to suggest that the method of diagram marking presented is a feasible approach to the problem.

Related work

The automatic marking of answers in a textual form has received much attention over the years. ... Our approach is similar in that we currently do not attempt to address any higher-order semantic structures above the “association”; this is similar to looking for key words or phrases in a sentential answer.

In contrast to textual answers, there has been very little work on the automatic marking of diagrammatic answers. The activity in diagrammatic reasoning has instead been directed towards other areas. These include the use of diagrams in formal mathematical proofs [Jamnik, 1998; Sawamura & Kiyozuka, 2000], visual query interfaces to GISs [Anderson & McCartney, 2003; Donlon & Forbus, 1999; Haarslev *et al.*, 2002], and the use of diagrams in supporting human-computer interaction [Meyer *et al.*, 2002]. However, some of the techniques developed in this work is of use in this context. In particular, there has been a significant amount of work on visual grammars for describing visual languages, of which the CMG used here is an example.

There is one diagrammatic marking system of which we are aware, called DATsys [Tsintsifas, 2002;Higgins et al. 2002a]. This system is markedly different from ours, but complimentary. DATsys

is part of the CourseMaster system [Higgins et al. 2002] which was originally intended for assessing programming. CourseMaster is organized around a set of marking modules that take a student's answer and a model answer and generate a mark and appropriate feedback. DATsys is essentially a front end to this marking system that handles diagrammatic input. Using DATsys, a question setter produces two domain-specific diagramming tools. One is used to define the question, the model answer, and a marking scheme. The other is used by the students to produce a diagrammatic answer to the question. The outputs of these two domain-specific tools form the inputs of the CourseMaster marking module. As is obvious from this description, DATsys and our system are intened to address different problems in diagrammatic marking. DATsys provides a method for creating bespoke diagram editors, but does little to address how those diagrams are marked. In contrast, our work concentrates on marking (perhaps ill-formed or inaccurate) diagrams, but does not emphasize how the diagrams are created in the first place.

Our concept of the process of diagram marking borrows heavily from foundational work on machine vision. The overall process of identifying primitive segments in a diagram which are then combined into larger, "semantic" components, is very similar to how many machine vision systems work.

Future work

The results of this initial trial suggest that the approach taken is worthy of further investigation. Therefore, we intend to perform more experiments with larger numbers of students, a variety of questions and an improved implementation of the marking software. In addition, we shall investigate the effects of different criteria for comparing associations.

A longer term aim is to expand the number of diagram objects that the system can deal with in order to investigate the marking of more complex diagrams such as E-R diagrams and certain UML diagrams. We hope that these investigations will provide us with greater insight into machine understanding of diagrams.

Constraint multiset grammars have been used to implement diagram editors such as the drawing tool described here [Iizuka, 2001]. This suggests that it may be worthwhile to describe the marking tool using a CMG and to implement it as a CMG parser.

References

Anderson, M., McCartney, R. (2003) "Diagram processing: Computing with diagrams", *Artificial Intelligence* **145** (1-2): 181-226.

Blackwell, A., Engelhardt, Y., (2002) "A meta-taxonomy for diagram research", Chaper 3 in *Diagrammatic Representation and Reasoning*, Anderson, M., Meyer, B., Olivier, P. (eds), pp. 47-64, London: Springer.

Chok, S.S. and Marriott, K. (1995) Parsing visual languages. In *Proceedings of the Eighteenth Australian Computer Science Conference*, Australian Computer Science Communications, **17**, 90-98.

Donlon, J.J., Forbus, K.D. (1999). "Using a geographic information system for qualitative spatial reasoning about trafficability", *Proceedings of the Qualitative Reasoning Workshop*. Loch Awe, Scotland.

Foltz, P. W., Gilliam, S., Kendall, S. (2000). "Supporting Content-based Feedback in Online Writing Evaluation with LSA", *Interactive Learning Environments*, **8**(2): 111-129.

Gazdar, G., Mellish, C. (1989) *Natural language processing in Prolog*, Wokingham, Addison-Wesley.

Haarslev, V., Möller, R., Wessel, M. (2002), "Visual spatial query languages: a semantics using desscription logic" Chapter 22 in *Diagrammatic Representation and Reasoning*, Anderson, M., Meyer, B., Olivier, P. (eds), pp. 387-410, London: Springer.

Higgins C., Hegazy T., Symeonidis P., Tsintsifas A., (2002) The CourseMaster CBA System, accepted in the Journal of Education and Information Technologies (official Journal of the IFIP Technical Committee on Education), published by Chapman and Hall, ISBN 1360-2357

Higgins C., Symeonidis P., Tsintsifas A., (2002a) Diagram-Based CBA using DATsys and CourseMaster, International Conference on Computers in Education (ICCE?02), Auckland, New Zealand, December 3-6, 2002

Iizuka, K., Tanaka, J. and Shizuki, B. (2001) Describing a Drawing editor by using constraint multiset grammars. *Proceedings of the Sixth International Symposium on the Future of Software Technology* (ISFST 2001), Zhengzhou, China, November, 2001.
www.iplab.is.tsukuba.ac.jp/paper/international/iizuka

Jamnik, M (1998) *Automating Diagrammatic Proofs of Arithmetic Arguments*, PhD thesis, University of Edinburgh.

Manning, C. D., Schütze, H. (1999), *Foundations of statistical natural language processing*, Cambridge, USA: MIT Press.

Marriott, K., Meyer, B., and Wittenburg, K.B. (1998) A Survey of Visual Language Specification and Recognition. In *Visual Language Theory*, eds: Marriott, K. and Meyer, B, Springer-Verlag, New York, 5-85, ISBN 0-378-98367-8.

Meyer, B. (2002) "Diagrammatic evaluation of visual mathematical notations", Chapter 15 in *Diagrammatic Representation and Reasoning*, Anderson, M., Meyer, B., Olivier, P. (eds), pp. 261-277, London: Springer.

Meyer, B., Marriott, K., Allwein, G. (2002), "Intelligent diagrammatic interfaces: state of the art" Chapter 23 in *Diagrammatic Representation and Reasoning*, Anderson, M., Meyer, B., Olivier, P. (eds), pp. 411-429, London: Springer.

Sawamura, H., Kiyozuka, K. (2000) "JVenn: a visual reasoning system with diagrams and sentences", *Proceedings, Diagrams 2000 (LNAI 1889)*, pp. 271-285.

Thomas, P. G., Price, B., Paine, C. and Richards, M. (2001). Experiments with Electronic Examinations over the Internet. *Fifth International Computer Assisted Assessment Conference*, Loughborough University, Loughborough, UK, Loughborough University, 487-502.

Thomas, P.G., Price, B., Paine, C. and Richards, M. (2002) *Remote Electronic Examinations: an architecture for their production, presentation and grading*. British Journal of Educational Technology, **33** (5) 539-552.

Tsintsifas A., (2002), "A Framework for the Computer Based Assessment of Diagram-Based Coursework", Ph.D. Thesis, Computer Science Department, University of Nottingham, UK:

Appendix

In a CMG, a constraint, denoted by the binary operator =, is *true* if the values of its two arguments are equal. If one or other of the operands is not assigned a value, that is, it is *uninstantiated*, that operand is instantiated with the value of the other operand and the constraint is *true*. If both operands are uninstantiated, they remain uninstantiated but the constraint is *true*. The semantics of the function *sum* are as follows. If the arguments of *sum* are all instantiated – to numeric values – the function returns the sum of the numeric values. If one or more of the arguments of *sum* are uninstantiated and the function occurs in a context where its value is set (instantiated), that value is distributed among the arguments in such a way that the instantiated arguments retain their original value and the uninstantiated arguments are instantiated with equal values. That is, if there are *n* arguments, of which *m* are instantiated, $0<=m<=n$, and sum.value is the instantiated value of the function,

$$\text{For all } j = m+1..\, n,\ arg_j = (\text{sum.value} - \sum (i=1..m,\ arg_i))/(n - m)$$

This scheme enables a mark to be set for the complete production and for some or all of its constituent parts and thereby provide differential marking. For example, if the Pipeline has a weight (mark) of 4, say, but the three associations do not have weights assigned, the 4 marks will be divided equally between the three tokens. If, however, a differential mark scheme is required, say with each the first association being worth 2 marks and the remaining two associations being worth 1 each, the constraint semantics ensure that the total mark for the Pipeline adds up to 4.