A Theorem Prover for a Diagrammatic Blocks World

Michael Wollowski

Computer Science and Software Engineering Department
Rose-Hulman Institute of Technology
Terre Haute, IN 47803, USA
wollowski@rose-hulman.edu

Abstract

Information can be presented in many ways. In this paper, we are concerned with information that is presented in diagrammatic form, where a diagram is a graphic that has a well-defined syntax. A good example of a diagram is a bar-graph. Bar-graphs have a well-defined structure and if properly labeled and constructed, are a fine form of representing certain kinds of information. We acknowledge that bar-graphs, and most diagrammatic systems are special-purpose systems and as such are limited in their application. However, such limitations never disturbed the human user. Newspapers, PowerPoint presentations, and web-sites are full of diagrams. This poses a challenge for programs that are designed to find or process information that is stored on computers. So far, there are only limited ways to gain access to such information. There are two issues that need to be addressed: the ability to read-off information that is stored in diagrams and the ability to reason with diagrams, so as to infer information that is implicitly contained in the information. In this paper, we summarize the design and implementation of a diagrammatic theorem prover. The domain of the theorem prover is the blocks-world of artificial intelligence. We will show that a well-formed syntax and semantic can be defined for this system. Based on it, we define valid rules of inference with which to give proofs in the system. We have shown this system sound and complete, and implemented it in Scheme.

Introduction

Information comes in many forms. One of them is graphical. While most information can be represented in different forms, some are preferred over others when it comes to certain reasoning tasks. For example, when it comes to planning in the blocks world of artificial intelligence (AI), many people prefer diagrammatic representations over sentential representations. The use of diagrams makes it easier to develop solutions to some of the planning problems in that domain. In addition to serving as a heuristic, diagrams also serve as a medium for developing solutions to problems. It is important to understand that a solution is *not* developed or presented in sentential form. A solution is developed by modifying diagrams and the reasoning is usually valid.

The research described here is embedded in a research project aimed at studying the use of diagrams in proofs and other types of valid reasoning. Not stopping at the heuristic value of diagrams, this project wants to introduce diagrams as valid parts of proofs. That this can be done in Geometry has been shown by [6]. For valid diagrammatic use of Venn diagrams, see [7]. For other diagrammatic systems, see [4,5]. For an in-depth motivation of valid diagrammatic reasoning and for reference to *HyperProof*, an educational software system that is designed to teach valid reasoning using diagrams, see [2]. For cognitive and computational perspectives on diagrammatic reasoning, see [3].

To show that one can develop valid proof systems for the domain of planning in the blocks world of AI, we developed a system in which diagrams carry the bulk of the information about plans. We furthermore implemented the system, thereby showing that the process of giving proofs in our system can be automated. It is our hope that such automated systems lead to more efficient AI systems as well as extend the usefulness of computers when it comes to representing and retrieving graphical information. In this paper, we present our diagrammatic system.

The domain of planning in the blocks world has several advantages when it comes to diagrammatic reasoning. People prefer diagrams when solving problems in this domain. Furthermore, if we look at textbooks and papers in this area, we see that diagrammatic representations are preferred over sentential ones. If diagrammatic representations do have any benefits over sentential representations, this is an area where those benefits should manifest. Equally important, when taking a new look at a field, it is important to start with a simple domain that can be defined easily and studied thoroughly.

A Sample Proof

The system to be presented is about simple block worlds such as found in planning research of Artificial Intelligence. The standard block world consists of a table of infinite length and a finite number of equal-sized blocks. Given a table of infinite size, one can always find a plan in this domain. This can be achieved by first moving all blocks to the table and then assembling the goal state. It should be noted that this procedure does not guarantee that one finds a shortest plan, something of great concern to early planning systems. In the everyday world, tables are of finite size. More interestingly, with a limited table size, there are problems that cannot be solved. For example, suppose that the position of blocks A and B were to be reversed in the right-hand part of the figure 1. Furthermore, suppose that there is only space for two blocks on the table and that one can move only one block at a time. In this case, there is no solution to the planning problem. However, knowing that a problem has no solution can be as important as positive information about a solution.

In order to demonstrate the type of reasoning we want to model, consider the problem given in figure 1. Furthermore, suppose we restrict the plans sought to those in which only the top-most block of a tower can be moved and in which there are not more than two blocks on the table.

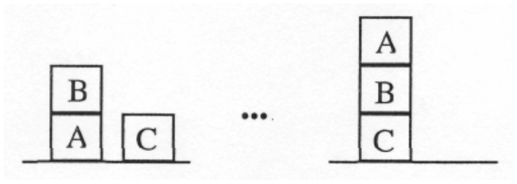

Figure 1: A planning problem.

While the constraints put on the plans sought could be represented in diagrammatic form, this is usually not done. In real-life problem solving situations, they oftentimes are not represented at all; rather, people seem to keep them in mind. In this exposition, we leave them out for the sake of brevity. In the system we developed, they are represented in sentential form.

Suppose that we want to show that every plan as specified in figure 1 includes the situation depicted in figure 2.

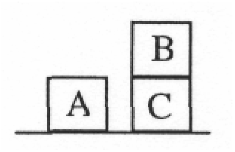

Figure 2: A situation to be shown part of every plan satisfying Figure 1.

The proof to be discussed is displayed in figure 3. Most often, people produce some sort of tree in which frames are nodes and arrows are used to connect parents and children. In this spirit, as the first step of our proof, we write down the initial state at the root of the tree. In order to be able to refer to it in our exposition, we call this frame **F1.**

Given the sentential constraints, there are two valid moves that can be made. Block C can be moved on top of B, resulting in **F2** or B can be moved on top of C, resulting in **F3**. They are written down next to each other in order to indicate that they are siblings and connected by arrows, originating from their parent. From **F2**, there is only one valid move that can be made: to move C back to the table, resulting in **F4**. However, **F4** is identical to **F1**. This means that we produced a cycle. While moving in cycles does not violate any of the present constraints, the prover will realize that one eventually has to break out of the cycle in order to reach the final situation. One could erase **F4** and instead point an arrow back to **F1**. Instead, we will simply put a cross below **F4**, indicating that in terms of what we want to prove, we reached a dead end.

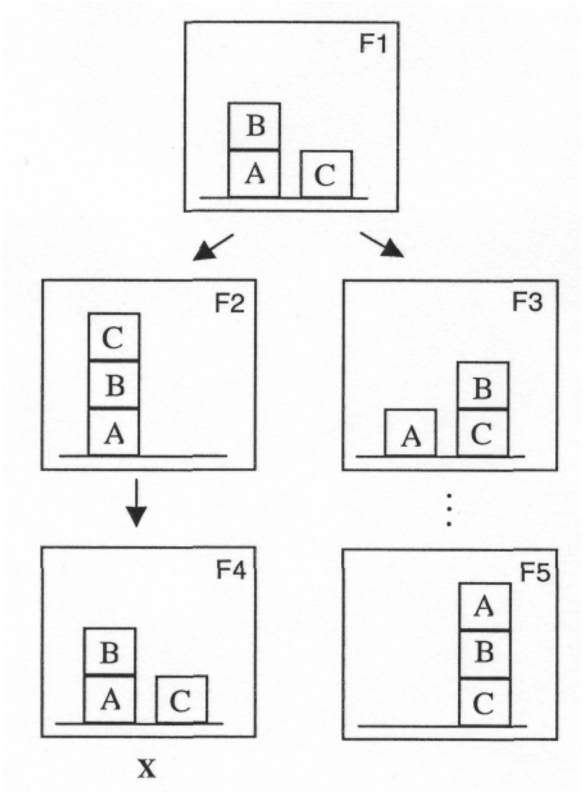

Figure 3: An abbreviated consequence proof.

In order to reach the goal frame, one eventually has to proceed through **F3**. Notice that **F3** is identical to the frame that is to be shown to be part of every plan. With this, we have shown that every plan solving this problem has to include a situation depicted by **F3**. In order to indicate which path leads to the final state, we add **F5**, a frame that represents the final state. Since we do not need to continue the reasoning process, we will merely link-up **F3** and **F5** by triple-dots.

In the above reasoning process, we repeatedly added children that exhausted all legal moves. In this manner, we obtained a tree of frames. We eventually were able to close off some of the branches of the tree, leaving others that contain the desired information. This process of "splitting" a node into children that exhaust their parent is the type of reasoning we modeled.

Syntax of Diagrams

It is important to realize that figure 3 does not represent some *sketch* of a proof, but *is* a valid proof in a properly designed system such as ours. We now outline the design of our diagrammatic proof system.

Just as formal languages are made precise by specifying a syntax and semantics, so can diagrammatic systems be made precise. Consider the diagram in figure 1. The elements of the diagrammatic language for our system are as follows. (i) *Labels,* consisting of the letters of the alphabet, are used to distinguish between blocks. Colors would do too, yet, in a paper, it is so much easier to refer to blocks by letters rather than colors. (ii) *Block squares* are used to represent blocks. (iii) *Horizontal lines* are used to represent tables. (iv) *Boundary rectangles* are used to delineate the diagram. (v) *Connectors:* -> and ..., are used to represent whether a situation is to follow another one immediately or eventually. In addition to diagrams, we have two predicates that can be used to place further constraints on the kinds of planning problems that can be posed in the system. With the predicate *table_size(n),* one can constrain the size of the table. With the predicate *move_size(m),* one can constrain the maximum height of the tower of blocks that can be moved at any single time.

In addition to the elements of the language, we defined rules of well-formedness for diagrammatic objects. An example of a well-formed diagram can be found in Figure 1. A key rule for well-formed diagrams is that *Block squares* have to be drawn on top of other *Block squares* or the *Table line,* in a non-overlapping fashion. In the description of the diagrammatic elements we hinted at the semantics of the diagrammatic objects. In [8], a full specification of the syntax, as well as the semantics in terms of a mathematical model of the domain is given.

In our system, one can give four types of proofs. We will presently discuss a consequence proof. With it, one can show that a certain property holds for all plans that start out in a certain situation and end in some other situation. Additionally, one can give inconsistency, consistency, and non-consequence proofs. With an inconsistency proof, one shows that a planning problem does **not** have a solution. One would use a consistency proof to show that a planning problem has a solution. With a non-consequence proof, one disproves a claim that is made for all plans.

Rules of Inference

Using the proof given in figure 3 we now outline some of the rules of inference of our system. The full proof can be found in [8]. A consequence proof begins by accepting as given information, consisting of the diagram of figure 1, as well as the two sentential constraints: **move-size(l)** and **table-size(2).** Suppose that we want to show the diagram that like the one of figure 1 to a consequence, with an additional frame of figure 2 in the center, such that it is triple-dot connected to both of its neighbors.

After accepting the initial information, we use a step which will become obvious late in the proof. In order to handle cycles effectively, we make them explicit. This is done by the rule **cycle**. Using it, we introduce a copy of the left-most frame and connect it by triple-dots to both neighbors such as shown in figure 4. With this rule, we assume that the introduced frame be the last such frame before the right-most frame. The diagram of figure 4 is added as a child to the "given" information as described in the prior paragraph.

The next two steps of the proof consist of extending the diagram of figure 4 to make explicit the two moves which are possible from the center frame, without violating the sentential constraints. This is done through the rule **move**. We create two new diagrams, each containing a new frame between the center frame and the right-most frame. The frame F2 from figure 3 is connected by an arrow to the center frame in one diagram and the other one contains frame F3 instead. Each of the resulting diagrams are added as children to the diagram described in the prior paragraph. We thereby "split" the diagram of figure 4, into two possible successor diagrams, both of which satisfy the sentential constraints. We now use the rule **cases exhausted** with which we claim that those two children exhaust all possible successor moves that do not violate the sentential constraints.

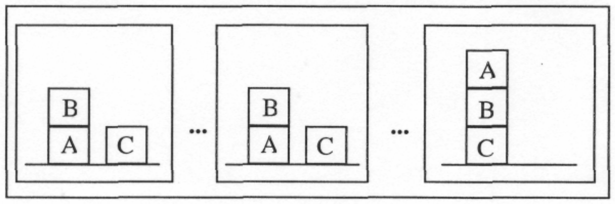

Figure 4: Dealing with cycles.

Continuing with the diagram containing frame F2, there is only one successor move that does not violate the sentential constraints. It is the one in which we move block C back to the table, thereby creating a cycle. This is again accomplished by the rule **move**. It is immediately followed by **cases exhausted** with which we claim that this is the only possible move. We will now use the rule **close** on the resulting diagram, because it contradicts our claim, made in figure 4, that the center frame be the last such frame before the right-most frame. This leaves us with the right-hand path, the one containing frame F3. In it, we find the move of block B on top of C. This frame is identical to the one of figure 2. With this, we have shown that such a move is a consequence of the given information and has to appear in any plan that satisfies Figure 1.

These are the major rules of inference. There are additional rules of inference designed to **generalize** and **specialize** the sentential constraints, to verify that sentential **constraints** are **satisfied** in a diagram and to **merge** diagrams which can also be used to generalize diagrams.

Implementation

Since information is increasingly stored in electronic form and many times made available in electronic form before it is proliferated in other forms, we were interested in developing an automatic theorem prover based on our system. This would show that such information can be searched and more importantly, that diagrammatic information can be manipulated by automatic theorem provers. Developing such as theorem prover was made possible because our proof system is sound and a sub-system of it is complete.

Diagrams are represented through structures. There are three structures: for frames, connectors and what we call an information packets, which contain all information pertaining to a state of the planning problem, e.g. the frames and connectors of figure 1 as well as the sentential constraints. An information packet for figure 1 would be defined as follows. like data structures

```
(define x '((a b) (c)))
(define y '((a b c)))

(define-structure
   (frame number info last next))
(define-structure
   (connector info next))

(define f8 (make-frame nil nil nil nil))
(define c9 (make-connector nil nil))
(define f9 (make-frame nil nil nil nil))

;;; We fill them as follows.

(set-frame-info! f8 x)
(set-frame-number! f8 (get_counter))
(set-frame-next! f8 c9)

(set-connector-info! c9 'dots)
(set-connector-next! c9 f9)

(set-frame-info! f9 y)
(set-frame-number! f9 (get_counter))

; Sentential constraints are represented as
; a dotted pair. The first element is the
; move-size constraint, the second one
; the table-size constraint.

(define s3 (cons 1 2))

;;; Now we create an information packet:

(define-structure (ip diagram sigma))
(define i2 (list (make-ip f8 s3)))
```

The rules of inference were implemented directly from the ones given earlier.

Conclusions

The system presented here serves to demonstrate that one can design and implement systems in which diagrams are not just used to represent information but also to reason validly with them. By proving the system sound and complete, we were able to design and implement a fully automatic special-purpose theorem prover.

A major advantage of the diagrams in our domain seems to be their close resemblance to the situations they represent. The structure of a diagram mirrors certain relationships that hold between objects in the domain. Barwise and Etchemendy [2] define this closeness as a *homomorphic* relationship between the structure of the diagram and the structure of the situation it represents. In our example, the relationship of a square being drawn on top of another square maps into the relationship of a block being physically located on top of another block. This homomorphic relationship seems to enable people to quickly grasp the situation at hand and lead to the development of solutions by mental imagery. This property might be useful in identifying domains that are applicable to diagrammatic reasoning.

One avenue of future work focuses on uniting different kinds of hand-crafted systems. A major concern of this line of research is to develop languages with which to transfer information from one system to the next. Some of this research would be based on heterogeneous rules of inference that are already present in systems such as *Hyperproof* [1]. In Hyperproof, one can inspect a diagram and transfer some of the diagrammatic information into sentential information and vice versa.

A different avenue for further research is to devise a semi-general diagrammatic system. By this we mean a system with a general-purpose syntax but domain-specific semantics, much like an expert system shell. Such a system would contain a basic set of diagrammatic types as well as certain composition schemes. In addition to composition of tokens, the size of a token could carry information, such as is the case in bar-graphs. A general-purpose system, composed of special-purpose systems with similar cores has an interesting benefit. It lends itself to pattern matching across domains, an important feature of common sense reasoning.

References

[1] Barwise, Jon and Etchemendy, John. *Hyperproof.* CSLI Press, Stanford, CA, 1994.

[2] Barwise, Jon and Etchemendy, John. Visual information and valid reasoning. in: *Visualization in Mathematics*, Zimmerman and Cunningham (eds.), Mathematical Association of America, Washington, D.C., 1990.

[3] Glasgow, Janice, Narayanan, Chandrasekaran (eds.). *Diagrammatic Reasoning*, AAAI/MIT Press, Menlo Park, CA, 1995.

[4] Hammer, Eric. *Logic and Visual Information.* CSLI Press, Stanford, CA, 1995.

[5] Harel, David. On visual formalisms. in: *Communications of the ACM*, 31(5), pp 514-530, 1988.

[6] Luengo, Isabel. *A Diagrammatic Subsystem of Hubert's Geometry*. Dissertation, Philosophy epartment, Indiana University, 1995.

[7] Shin, Sun-Joo. *The Logical Status of Diagrams.* Cambridge University Press, Cambridge, UK, 1995.

[8] Wollowski, Michael. *Diaplan: A Decidable Diagrammatic Proof System for Planning in the Blocks World.* Dissertation, Indiana University, 1998.