FORMULAS FOR THE BLOCKS WORLD

John McCarthy
Computer Science Department
Stanford University
Stanford, CA 94305
jmc@cs.stanford.edu
http://www-formal.stanford.edu/jmc/

1997 Jan 14, 2:37 p.m.

Abstract

1 Introduction

This paper contains formulas for several axiomatizations of a blocks world with moving blocks and painting blocks. Later versions will have more explanations, but I wanted to get the formulas down first.

2 Effect axioms and monotonic frame axioms

We begin by cutting some corners so as to keep the axioms short.

$$clear(x,s) \wedge clear(l,s) \rightarrow at(x,l,move(x,l,s)) \tag{1}$$

Remark: We have cut a corner in (1) by letting a block itself be a place, i.e. the top of the block. To uncut this corner, we can write

$$clear(top(x),s) \wedge clear(l,s) \rightarrow at(x,l,move(x,l,s)). \tag{2}$$

The price is dealing with the extra function *top*.

$$clear(x,s) \equiv (\forall y)(\neg at(y,x,s) \tag{3}$$

$$color(x,c,paint(x,c,s)) \tag{4}$$

2.1 Frame axioms

The following four axioms assert that the colors of objects don't change when objects are moved or when other objects are painted and that objects don't change where they are at when other objects are moved or when any objects are painted.

$$color(x,c,move(y,l,s)) \rightarrow color(x,c,s) \tag{5}$$

$$color(x,c,s) \wedge y \neq x \rightarrow color(x,c,paint(y,c',s)) \tag{6}$$

$$at(x,l,s) \wedge y \neq x \rightarrow at(x,l,move(y,l',s)) \tag{7}$$

$$at(x,l,s) \rightarrow at(x,l,paint(y,c,s)) \tag{8}$$

Writing frame axioms this way means that if there are m properties and n actions (or events) mn frame axioms are required. By introducing *frames* as objects, this can be reduced to m axioms. In a practical system, this would be needed.

2.2 Some reifications

Writing $move(x,l,s)$ for the new situation resulting from the move gives the most compact notation. It is more common to write $result(move(x,l),s)$. Using $move(a,s)$ or more generally $move(e,s)$ for the action or event has the advantage that we can write sentences $(\forall e)(\ldots result(e,s)\ldots)$, i.e. we can quantify over actions or events. (The AI trade mostly talks about actions, but actions are just a kind of event, and much of the theory applies to events in general).

We can also reify the fluent $at(x,l)$ so instead of writing $at(x,l,s)$, we write $holds(at(x,l),s)$. We also write $holds(clear(x),s)$.

Sometimes it is convenient to use functions instead of predicates and write $color(x, s) = white$ instead of $color(x, white, s)$. We might further reify *color* and write $value(color(x), s)$ instead of $color(x, s)$. Likewise we can write $location(x, s)$ and reify that to $value(location(x), s)$.

With these reifications the axioms become

$$\begin{aligned} &holds(clear(x), s) \wedge holds(clear(l), s) \\ &\rightarrow value(location(x), result(move(x, l), s)) = l \end{aligned} \tag{9}$$

$$value(color(x), result(paint(x, c), s)) = c \tag{10}$$

$$holds(clear(l), s) \equiv (\forall y)(location(y, s) \neq l) \tag{11}$$

$$value(color(x), result(move(y, l), s)) = value(color(x), s) \tag{12}$$

$$x \neq y \rightarrow value(location(x), result(move(y, l), s)) = value(location(x), s) \tag{13}$$

$$x \neq y \rightarrow value(color(x), result(paint(y, c), s)) = value(color(x), s) \tag{14}$$

$$value(location(x), result(paint(y, c), s)) = value(location(x), s) \tag{15}$$

2.3 Frame axioms using frames

We begin with a purely mathematical device.

2.3.1 How to assert n objects distinct with $n + 1$ axioms

Suppose we have n objects $x_1, \ldots, x_n$, and we want to assert that they are all distinct. The obvious way to do it is to have $\frac{n(n-1)}{2}$ axioms $x_i \neq x_j$. We can do it with by writing the axioms $x_1 < x_2, x_2 < x_3, \ldots x_{n-1} < x_n$, which comes to $n - 1$ axioms. We then write $x < y \wedge y < z \rightarrow x < z$ and $x < y \rightarrow x \neq y$. This ordering trick permits proving that the x_is are all distinct. The transitivity of $<$ lets us infer the $n(n-1)$ inequalities we want.

Notice that the ordering has no relation to any properties of of the x_is, and any of the $n!$ total orderings of $(x_1, \ldots, x_n)$ will do.

It might be considered more elegant to write functions from the x_is to the integers and use the transitivity, etc. of the integers to infer the inequalities. Thus we would write $index(x_1) = 1, index(x_2) = 2, \ldots index(x_n) = n$, which comes to n equations. The index functions are arbitrary, because all they have to do is assign different indexes to different objects.

Within a computer program, we may give the objects names, e.g. write equations $name(Fido) = "Fido"$, etc. and rely on the computer to assert that the strings "Fido", "Rover", etc. are distinct.

When we have only a few objects, it isn't too much trouble to write the inequalities directly.

2.3.2 Assignment and Contents Functions

These functions are taken from [McC62][1] with slight changes.

We work with a state vector ξ. It will be the situation s in our application, but we use ξ here in order to emphasize the mathematical notion. There are two functions.

$c(var, \xi)$ denotes the contents of the variable var in the state ξ.

$a(var, value, \xi)$ denotes the new state obtained by assigning the value $value$ to the variable var in the state ξ and leaving all the other variables with the values they have in the state ξ.

We have the axioms

$$
\begin{aligned}
c(var, a(var, value, \xi)) &= value && (16)\\
var1 \neq var2 \rightarrow & \\
c(var1, a(var2, value, \xi)) &= c(var1, \xi) && (17)\\
a(var, c(var, \xi), \xi) &= \xi && (18)\\
var1 \neq var2 \rightarrow & \\
a(var1, val1, a(var2, val2, \xi)) &= a(var2, val2, a(var1, val1, \xi) && (19)
\end{aligned}
$$

These axioms capture the notion that an assignment of a value produces a new state in which just that variable has its value changed. These functions are just what we need when we introduce frames.

[1] http://www-formal.stanford.edu/jmc/towards.html

2.3.3 What is a frame?

We can write the axioms for moving and painting as follows:

$$clear(x,s) \wedge clear(l,s) \rightarrow move(x,l,s) = a(location(x),l,s) \tag{20}$$

$$paint(x,c,s) = a(color(x),c,s) \tag{21}$$

Notice that the assignments are made to the terms $location(x)$ and $color(x)$ rather than to simple variables as in (16). In order to avoid contradiction as well as to describe moving and painting, we need axioms that these objects are different. We have axioms

$$\begin{aligned} block_i &\neq block_j \text{for each two blocks} && (22)\\ x1 \neq x2 \rightarrow location(x1) &\neq location(x2) && (23)\\ x1 \neq x2 \rightarrow color(x1) &\neq color(x2) && (24)\\ location(x) &\neq color(y) && (25) \end{aligned}$$

Using situations as frames in this way makes certain commitments that might not be immediately obvious. Consider $distance(x1,x2)$ which is supposed to be some function of $location(x1)$ and $location(x2)$.

We will not want to use $c(distance(x1,x2),s)$, because our axioms would not permit $c(distance(x1,x2),s)$ to change when the locations changed. Instead we use another function $value(term,s)$ and the equations

$$\begin{aligned} value(location(x),s) &= c(location(x),s) && (26)\\ value(color(x),s) &= c(color(x),s) && (27)\\ value(distance(x,y),s) &= \text{distancefun}(value(location(x),s), value(location(y),s)) && (28) \end{aligned}$$

The effect is to take $location(x)$ and $color(x)$ as *primary fluents* and $distance(x,y)$ as a *secondary fluent* whose value is dependent on the values of primary fluents. Deciding what fluents to make primary is an epistemological decision, i.e. it is a feature of the theory being used. $clear(x)$ must also be a secondary fluent.

[2]

[2] I used the word *frame* by analogy with the notion of co-ordinate frame in geometry.

3 Projection in the Blocks World

By projection we mean inferring facts about the situation that results from performing a sequence of actions in an initial situation $S0$. Consider the situation of figure 1.

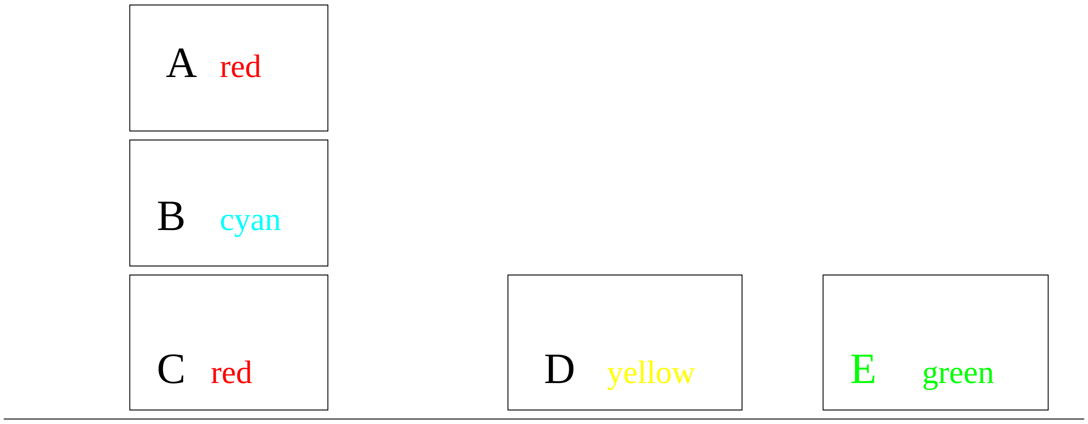

Figure 1

We call it $S0$ and describe it as follows:

The geometers of the early part of this century pounded the table about how frames were arbitrary and as much as possible should be done independently of specific frames. In AI there is some choice in deciding what fluents to make primary, but there isn't as much choice as in geometry. The reason is that geometry admits large groups of transformations among equivalent frames. We don't have any use for transformations interchanging locations and colors, although we might have some use for transforming the co-ordinate system used to give locations. Interchanging colors comes up when we consider search heuristics, because it can reduce the number of cases that have to be examined. That's outside the scope of this article.

$$
\begin{array}{l}
value(location(A), S0) = top(B) \\
value(location(B), S0) = top(C) \\
value(location(C), S0) = Table \\
value(location(D), S0) = Table \\
value(location(E), S0) = Table \\
value(color(A), S0) = Red \\
value(color(B), S0) = Cyan \\
value(color(C), S0) = Red \\
value(color(D), S0) = Yellow \\
value(color(E), S0) = Green
\end{array} \tag{29}
$$

These formulas are somewhat lengthy. Building enough machinery would permit a more compact description, e.g.

$$
config(S0) = \ '((A\ Red\ B)\ (B\ Cyan\ C)\ (C\ Red\ Table)\ (D\ Yellow\ Table)\ (E\ Green\ Table)). \tag{30}
$$

Compactness is not our present goal, so we won't bother.

The above is the positive information about $S0$. We also need negative information such as that these are all the blocks in the scene, that they are distinct and that blocks are distinct from situations and colors—considerably more.

Here are some items:

1. Blocks are distinct from colors and both are distinct from situations. We can write

$$
\begin{array}{l}
is\text{-}block(x) \rightarrow type(x) = 1 \\
is\text{-}color(x) \rightarrow type(x) = 2 \\
is\text{-}situation(x) \rightarrow type(x) = 3.
\end{array} \tag{31}
$$

   We assume that, as in section 2.3.1, we have axioms that make the natural numbers all different.

   We also want axioms

$$
\begin{array}{l}
is\text{-}block(A) \wedge is\text{-}block(B) \wedge \ldots \\
is\text{-}color(Red) \wedge is\text{-}color(Cyan) \wedge \ldots \\
is\text{-}situation(S0).
\end{array} \tag{32}
$$

Another way of making blocks, colors and situations, etc. distinct from each other is to use a multi-sorted logic.[3] A sorted logic could provide different variables for the blocks, colors, etc. and a convention that the variables range over the appropriate sort.

2. The blocks are distinct from each other and so are the colors. Any of the devices of section 2.3.1 will work.

3. We need to say that A, B C, D and E are all the blocks present in the situation $S0$ and its successors. The brute force way of doing this is to write

$$is\text{-}block(x) \rightarrow x = A \vee x = B \vee x = C \vee x = D \vee x = E, \tag{33}$$

or if the variable x is declared to be of sort *block*

$$x = A \vee x = B \vee x = C \vee x = D \vee x = E. \tag{34}$$

[much more to come]

References

[McC62] John McCarthy. Towards a mathematical science of computation. In *Information Processing '62*, pages 21–28. North-Holland, 1962. Proceedings of 1962 IFIP Congress.

/@steam.stanford.edu:/u/jmc/f96/blocks-formulas.tex: begun Fri Dec 13 11:35:08 1996, latexed January 14, 1997 at 2:37 p.m.

[3] "Sorted" is the term used by the logicians. Computer scientists use "type". This is a mistake, because the logicians' notion of type is also useful in computer science.