Symbolic Expressions within a Spatial Algebra: Unification and Impact upon Spatial Reasoning

Erland Jungert

FOA (Swedish Defence Research Establishment)
Box 1165, S-581 11 Linköping, Sweden

ABSTRACT

Spatial reasoning in symbolic images requires means for identification of spatial relationships. In earlier work an algebra based on symbolic projections for manipulation and transformation of symbolic images has been defined. Although that symbolic algebra was quite powerful for manipulation of symbolic images, it did not include any means for spatial reasoning. With such a goal it was already clear at an early stage that the algebra had to be developed further. This work describes unification of algebraic expressions, which founds a powerful means for spatial reasoning. The paper demonstrates that the algebra constitutes a basis for a tool for spatial reasoning in symbolic images.

1. Introduction

In such systems as, for instance, navigation systems for autonomous or semi-autonomous vehicles, where digitized maps are used there is an obvious need for spatial reasoning. Hence, various means for spatial reasoning have been developed during the last few years and the interest in the area is still growing. Several application areas where techniques for spatial reasoning are necessary exist beside navigation. Among these can path planning based on digitized map information be mentioned. Examples of knowledge based methods for spatial reasoning are e.g. [1], [2] and [3]. Generally, spatial reasoning concerns methods for identification of various types of spatial relationships among different objects in symbolic images, such as, for instance, digitized maps. The algebra, discussed here, is not limited to such kind of images, however, as it can be applied to any kind of images.

The technique that will be described here rests upon a "symbolic algebra" whose basic theories are described by Jungert and Chang [4]. The fundamentals of the theories of that work concern the manipulation and transformation of symbols or objects in the image. The theories described here are a direct continuation of that previous work. As indicated above the main motive for the work has been to develop a means for spatial reasoning. However, the basic ideas of the algebra can be found in [5] and further extensions are described in [8]. So far, no direct work similar to this has been found in the literature. In, e.g. [6], some relationship is found although here most cases are concerned with various types of grammars. Another work that is somewhat related is that by Allen [7]. In Allen's case, however, the work concerns temporal reasoning and the relations which are identified are of interval type and in one dimension.

Nevertheless, there remains a relationship. Finally, [8] discusses problems which are related but only in terms of a structure that is hierarchical. The symbolic algebra, however, is more general since, for instance, quad-tree can be handled as well.

The fundamentals of the algebra are described in Section 2. An important aspect is unification, which is discussed in Section 3. Section 4 and 5 discuss the consequences of overlapping objects and operator precedence. While 6 and 7 concern spatial reasoning and its foundations. Finally in Section 8 there is a discussion about justification local to the so-called "area in focus" for spatial reasoning.

2. Fundamentals of the Algebra

The basic structure in the algebraic expressions are the symbolic projections, originally described by Chang et al. [5]. A symbolic projection is a string of objects, which includes the relative positions of the objects along any of the coordinate axes. In the original version there were just two relational, i.e. "less than" and "equal to". Later it was clear that two operator were not enough and the number was expanded, see e.g. [9]. The number of operators in the symbolic algebra is three and except for the two in the original work only one extra was necessary; the extra one is called "edge to edge with" and is symbolized with I. Hence, the I-operator denotes that two objects are situated edge-to-edge with each other.

Projections of the objects take place along all coordinate axes. Hence, one string for each axis is built up. In the 2-D case the string corresponding to the x-axis is labeled with a U, and the string corresponding to the y-axis is labeled with V. See, e.g. (1) and (2) in the next section.

In [4] it was demonstrated that various classes of objects could be generalized. This was especially true for objects corresponding to the empty space, here called e-objects. A generalized e-object is a subobject of the full empty space, which is the space were arbitrary mobile objects are allowed to move freely. An e-object can have arbitrary shape and size. An important aspect of the objects is also their relative positions. Several laws are identified for the e-objects, including: i) they can be merged whenever they are edge-to-edge with each other or, ii) they can be split, where the split is either parallel to and orthogonal to their projected coordinate axes. e-objects are not the only type of generalized objects. Obstacles, in the case of path planning illustrated later in this paper are another type.

Rules for manipulation of arbitrary object types are also identified in [4]. These rules are called object manipulation rules. In the object manipulation rules objects are split or merged in many ways which are similar to the laws for manipulation of the e-objects. The merge and split operations concern just
subobjects of the same object. Another aspect is found in rules applied to either rows or columns in the images.

An important object manipulation rule that is applied quite often in this work is rule (xiii), which says:

\[(xiii) a_1 a_12 \ldots a_{1n} (a_21 \ldots a_{2n}) (a_31 \ldots a_{3n}) \ldots (a_m 1 \ldots a_{mn}) \leftrightarrow \]

The interpretation of this rule is that a "column view" in the left side of the rule is transformed into a "row view" in the right side of the rule. This is illustrated in expression (1) in next session, which corresponds to a column interpretation of figure 1. Expression (3) corresponds to a row interpretation received from a transformation of (1) by means of rule (xiii).

Other characteristics of the symbolic algebra are that the algebra can be applied not only to hierarchical structures, but to shape analysis as well. Algebraic expressions can also be transformed into graph structures such as in (3).

3. Unification

In [4] it was demonstrated that loss of information in the algebraic expressions can be a serious matter. However, there is one aspect of the algebra that can contribute to the solution of this problem. This aspect is called unification since it contributes to the transformation of all different orthogonal strings into a single string along an arbitrarily chosen coordinate axis. The unified string comprises all available information. Unification is illustrated as follows. Hence, from figure 1 the following algebraic strings is created:

(1) \[U: A a c I B a c I B a d \]

(2) \[V: A a l I C D \]

Figure 1. Visual representation of the expressions in (1) and (2).

If rule (xiii) is applied to the U-string in (1) above, it will be transformed into:

(3) \[U: (A I B I B)(a_1 a_1 a)(C I C I D) \]

This expression is equivalent to:

(4) \[U: (A I B I B)=(a_1 a_1 a)=(C I C I D) \]

As mentioned already in section 2, the interpretation of this transformation is that a column structure, as in (1), is transformed into a row structure (4). Now, since a row in the U-string is equivalent to a column in the V-string then:

(5) \[U: A I B I B \leftrightarrow V: A B \]

Hence, the V-string is actually equal to ABB although including less information compared to the U-string. It is now easy to see that a row in the U-string can be transformed into a column of a V-string by simply interchanging the \(=\)-operators with \(\&\)-operators and vice versa. Now, consider the relationships between the rows in the U-string and the relationships between the columns of a V-string. From this it follows that the \(\&\)-relations of the U-string correspond to the \(\&\)-relations in the V-string. Consequently, a row-oriented U-string can be transformed into a V-string by just substituting the \(\&\)-operators with \(\&\)-operators and the \(\&\)-operators with \(\&\)-operators. (3) then turns into (6).

(6) \[U: A B I A a a a l C C D \]

Observe that the transformed string in (6) actually contains more information than the original V-string (2). Another illustration of this is the following example from [4], where object A has the shape of an L, which cannot easily be identified.

(7) \[U: A B I A =\Rightarrow U: (A I B) A =\Rightarrow U: A A I B A \]

Here there is no need to transform the U-string into a row oriented expression before it is unified. A direct transformation is permitted and here as well the new V-string contains more information than the original. Apparently, there is now enough information available to identify the L shaped object in the image by just looking at the new V-string.

If the above technique is applied to the strings generated from the objects in figure 2, i.e.:

(8) \[U: A A I A \]

(9) \[V: A A I A \]

Then unification of the V-string gives:

(10) \[U': (A')(A')(A') \]

Figure 2. Example of an ambiguous object.

Expression (10) is incorrect, since again the expression does not correspond to the object but rather to three columns which are equal in shape. Therefore, the conclusion is to enclose the object inside a rectangle and work with the following expressions:

(11) \[U: e A I A I A e \]

(12) \[V: e A I A e \]

Now, if another attempt is made to unify towards the V-string, the U-string becomes:

(13) \[U: (e I A) A (A I e) \]

Applying rule (xii) on (13) gives:

(14) \[U: e A A I A A e \]

Obviously, this corresponds better to the object that is described and if (14) is transformed back to the U-string then the result is:

(15) \[U': (e I A A) (A I A I e) =\Rightarrow e A I A A I A e \]
Both (14) and (15) are complete descriptions of the object in question.

Consider again figure 1, and the expressions in (1) and (2):

(16) U: AaC | BaC | BaD
(17) V: AB | a | CD

From (16) the corresponding V-string transformation is:

(18) U: (A | a | C)(B | a | C)(B | a | D) => ABB | a | CCD

and from (17) the transformed V-string becomes:

(19) V: (A | B)a(C | D) =⇒ AaC | BaD

Expression (18) contains more information than (17) while expression (19) contains less than (16). Hence, the question is why is it so? Why does unification of a symbolic expression sometimes give rise to a string with more information and sometimes not? The answer is that the string that contains the most information is the one that contains the largest number of subobjects and that information is kept in the unification step. In the example above, the string containing the most information is the string corresponding to (16)... Consequently, in order to describe an object in terms of the symbolic algebra only one string is needed and that string corresponds to the coordinate axis along which the largest number of subobjects exist.

4. Overlapping objects

In the majority of the examples considered so far only objects shaped as rectangles have been used. Such objects can also be looked upon as arbitrary objects which are enclosed within their least rectangle. However, in many cases such rectangles will overlap, hence there must be a way to handle such rectangular objects. In this section objects with enclosed and overlapping rectangles will be discussed.

Overlapping rectangles or objects are illustrated in figure 3 below.

![Figure 3](image)

Figure 3. An illustration of the case of overlapping rectangles.

The corresponding symbolic algebraic expressions is then:

(1) U: A | B | (B | A) | A | B
(2) V: B | A | (A | B) | B | A

Here, the overlapping portions of the rectangles A and B are denoted with (A B) or (B A). The order of the objects or subobjects that are situated inside the parentheses is irrelevant.

Again stripped strings cannot be used, especially if the expressions should be unified. Hence, the empty space that surrounds the objects must be considered in the expressions. The expressions (1) and (2) become:

(3) U: e | A | B | (B | A) | A | B | e
(4) V: eB | A | (A | B) | B | e

Unification of both (3) and (4) now works and, specifically for (3), the result becomes:

(5) U: (e | A)(B | (B | A) | A)(B | B | e)

Which then can be transformed into:

(6) U': (e | A)(B | (B | A) | A)(B | B | e) =>

As can be seen (B | A) is handled as a single subobject and, as a whole, the strings behave in the same manner as was already pointed out in the section on unification.

5. Operator precedence

In conventional algebra, precedence among operators is specified rigorously. Precedence has to be considered here as well, although the situation in this symbolic algebra is somewhat different than for conventional algebra. The main difference between the symbolic algebra and conventional algebra is that here we are talking about objects and their relations which have spatial extension in at least two dimensions.

In [9] there are discussions about associative and distributive laws which are applicable to the symbolic algebra. For that reason we are not concerned with such operations any further. The only comment is that these laws are implicit to the rules described earlier in the paper.

The discussion is first concerned with expressions of the type given in the section on unification, which is illustrated in expression (4). Precedence is independent of the direction of the actual coordinate axis along which the objects have been projected. Therefore, the following discussion concerns any arbitrary string, i.e. the S-string.

(1) S: (A | B | B) | (a | a | a) = (C | C | D)

In (1) above the subexpressions within parentheses can be considered as units, which correspond to rows parallel to the projection axis. Hence, precedence in (1) depends more on the parentheses and less on the other operators. However, if the expression is unified against some other direction, it becomes:

(2) S': A=B=a=a | a | C=C=D

Here the units are of the type:

(3) X1=X2=...=Xn

In these units the ties between the subobjects are stronger than in units of the type:

(4) Y1 | Y2 | ... | Yn

This is true except when parentheses are present as in (1). As reflected in (3) and (4), the = operator has higher precedence.
than the $l$-operator. Observe that the parentheses in (1) cannot be taken away, because this would change the semantic meaning of the expression.

The units which are discussed above are either columns or rows with respect to the direction of the projection. Hence, they are not objects in the ordinary sense; they are rather a collection of subobjects of various types. A consequence of this discussion is that in (1) rows are units within parentheses, built up by subobjects separated by $l$-operators. Columns are units built up by subobjects separated by $=$-operators.

6. Foundations for spatial reasoning

One of the main reasons for development of the symbolic image algebra has been to find a means for spatial reasoning, i.e. to use algebraic expressions as a knowledge structure for reasoning about spatial aspects. Examples include reasoning about maps and applications on path planning and landmark navigation. Of special interest is the use of unified expressions since then strings corresponding to projections in just one direction will suffice. This is illustrated in the map in figure 4 from which the following expression is created.

(1) $U: (e^{-})(eUll)(#lel#le)(#lel#lebe)(#lel#lel#lel#lel) e) (leAl#lel#lel#lel)(#le~l#le)(#le~le)(~el#le)(eUll#le)$

Figure 4. A simple map with obstacles and free space.

The symbol 'W' corresponds to objects which obstacles. These are generalized objects, just like the empty space objects e. Hence, the map is made up by two types of generalized objects: The empty space, where a vehicle is permitted to travel, and obstructed space. This type of map is discussed further in [3].

The map in figure 4 contains just horizontal cutting lines which are here numbered. At each point where a cutting line touches an obstacle there is a so-called split point. By definition a split point is situated either in the row below the corresponding obstacle (upper split point) or in the row above (lower split point). Consequently, the upper split point is upper with respect to the tile to which it belongs and not to the obstacle touched. The situation for the lower split point is similar. A tile is defined as the area between two cutting lines and between obstacles or edges of the images. The topic of split point is also discussed in [3]. In (1) the split points are indicated with $e$ followed by a back slash and the type of split point. The table below shows the various types that exist.

<table>
<thead>
<tr>
<th>Split Point Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>v^-</td>
<td>upper split point</td>
</tr>
<tr>
<td>v^A</td>
<td>lower split point</td>
</tr>
<tr>
<td>v^ll</td>
<td>lower split point at left edge</td>
</tr>
<tr>
<td>v^al</td>
<td>upper split point at left edge</td>
</tr>
<tr>
<td>v^ar</td>
<td>upper split point at right edge</td>
</tr>
<tr>
<td>v^lr</td>
<td>lower split point at right edge</td>
</tr>
</tbody>
</table>

Examples of split points of type v^al and v^ll, which are situated at the edge of the image, are found in figure 4 along line 2 and 9, respectively.

When more than one split point of the same type is present along a line then this is indicated with a constant between the backslash and the character that indicates the type of split point, this is illustrated in (2) below:

(2) $eW^O$

If more than one split point is present along the same line and if they are of different types then this is indicated as in the following expression:

(3) $eW^O2^O$

Both (2) and (3) can be cut so as to contain one e-object for each split point. Consequently, the expressions will change into:

(4) $eV^W^e$

and

(5) $eUle\le\le\le$

Expression (1) can be unified, because of the presence of the split points in the string. Normally a unified expression contains a lot of redundant information and for that reason it is better to handle the expression in (1). However, quite often substrings are handled and in those cases it is more convenient to handle unified substrings. Some of the strings are fairly long and have a pattern that is repeated frequently. For that reason, it is necessary to rewrite the strings in a more compact way, e.g.:

(6) $(\#lel#lel#lel#lel) => (4[\#le])$

(7) $(\#lel#lel#lel#lel) => (3[\#le]#) or (\#3[\#le])$

or more generally:

(8) $(\#le.../[\#le]) => (n[\#le])$

This compaction works for patterns of the same type, but not for strings where, for instance, split points are included, e.g.:

(9) $(eW^OeW^OeW^OeW^O) => (4[eW^O])$

The subexpressions in (9) can be compacted even further as in (10) and (11) below:

(10) $4[eW^O] => 4#W$ or $4\#E$
The interpretation of this is that any of the characters, # or e, followed by \( \neq \) are in even positions; in odd positions, when followed by \( \neq \). Hence, (7) becomes:

(12) \( (4e\neq)(2e\neq) \)

Finally, (1) becomes:

(13) U: \( (e\neq)(e\neq)(2e\neq)(4e\neq)(2e\neq) \)

It is possible to compact the strings even more, which is evident from the expressions below:

(14) \( (#l(n\#l)) \Rightarrow (#l(n\#l)) \Rightarrow (n+1)\neq \) or \( (n+1)\neq \)

(15) \( (eln\#le)(el\neq) \Rightarrow (n+1)\neq \) or \( (n+1)\neq \)

In this step the odd symbols are included. (14) shows a substring that starts and ends with a \#, that is a string that has an odd number of number-signs and an even number of empty objects and the number of \#s is greater than the number of es. The situation in (15) is similar to (12) although here the string has an odd number of es and even number of \#s.

If applied to the string in (15) the final compacted string becomes:

(16) U: \( (e\neq)(e\neq)(2e\neq)(2e\neq) \)

So far, we have only considered strings where the cutting lines are situated outside the obstacles. This is obviously not always the case. Situations can occur where there are cutting lines that correspond to split points inside the obstacles. An example of this is found in figure 5. The corresponding compacted algebraic expressions is:

(17) U: \( (e\neq)(1e\neq) \)

This can be illustrated with the following example:

(2) U: \( (e\neq)(e\neq)(e\neq)(e\neq) \)

where the area in focus is:

(3) \( <el#le> \Rightarrow <3e> \)

Areas in focus can be matched with expressions of the following type:

\( <el..> \) matches any area in focus that starts with \( e \),
\( <..le> \) matches any area in focus with a \# somewhere in the middle,
\( <..le> \) matches any area in focus that ends with \( e \),
\( <le> \) matches a string with a lower split point.

Obviously, the area in focus does not have to cover parts of just one substring, that is, just a portion of one row in the symbolic image. Therefore, there must be a way of describing an area in focus that stretches over several rows. Furthermore, it is also important to identify the order of the rows, i.e. whether we are moving downwards or upwards from one row to the next. When moving from one row to the next it is of particular importance that the areas before and after really correspond to each other. For that reason it must be possible to indicate that the initial and final parts in both rows coincide. This is indicated with \&', if in the beginning of the row, and \&' for the end of the row. (4) below illustrates this:

(4) \( (&<e..>..)(<e..>..)(&<e..>..) \)

The area in focus in the pattern in (4) corresponds, when matched with a symbolic image, to either a U-shaped object that surrounds the \#-object or, if in a dynamic situation, to a left turn around the same \#-object. (4) is interpreted as follows: The first substring or row indicates that initially there is a specific pattern followed by an empty area which is a part of the area in focus. The second substring, which is below the first one and with an arbitrary initial part, an empty area including an upper split point, and an arbitrary ending part. The third string is found above the second and it is therefore the same as the first one. Here the initial part is the same as in the first substring, which again is indicated by \&'. Following the initial part, there is an empty area, which clearly is identical to the empty area in the area in focus in substring one. After the e there is an obstacle followed by an empty area that also is the area in focus of the substring.

With this technique used as a knowledge structure it is possible to perform spatial reasoning simply by using the descriptions in (4) in inference rules. This is illustrated in (5) below.

(5) \( (&<e..>..)(<e..>..)(&<e..>..) \) then (left-turn)

In the example above the area in focus was matched against a rule from an expected knowledge-base. However, in the opposite situation, i.e. matching the entire knowledge-base against some part of an image, then the area in focus is not required to identify certain relations among present sub-objects in the image. This can be illustrated with the following example that corresponds to figure 6:

(6) \( (e\neq)(e\neq)(e\neq)(e\neq) \)

For e-objects that include split points (i.e. \( e\neq \)), those e-objects change into compound sub-objects of type (el\#le) when the

---

**7. Spatial reasoning**

In a string that describes a symbolic image, as in the examples which have been demonstrated above, there must be a way of identifying certain subareas for reasoning in the image. This is because it should be possible to identify various types of relations in the symbolic images. Consequently, there is a need for matching substrings corresponding to different relationships that might occur within the strings of the symbolic image. For that reason the "area in focus" for reasoning has to be considered. The area in focus is denoted by:

(1) \(...<area-in-focus>...)\
Another observation is that if there are two or more neighbouring split points then the interjacent e-object has to be cut. This is illustrated in (8) below:

\[
(8) \text{el\#le} \Rightarrow \text{el\#lelel\#le}
\]

The new cutting line is just local to the area that is being handled. A similar operation has to be applied when the two neighbouring split points are of different type, i.e. opposite to each other.

If the above observations are applied to the expression in (6) then:

\[
(9) (\text{el\#le}\text{elel\#lelel\#lele})(\text{el\#le})(\text{el\#le}) \\
\Rightarrow ((\text{el\#le})(\text{el\#le})(\text{el\#le}))(\text{elelel\#lelele} \\text{elelel\#lelele})
\]

When applying rule (xiii) to (9) the expression becomes:

\[
(10) (\text{el\#le})\text{e\#e} \\text{el\#le} \\text{e\#e} \\text{el\#le} \\text{e\#e} \\text{el\#le} \\text{e\#e} \\text{el\#le} \\text{e\#e} \\text{el\#le} \\text{e\#e} \\text{el\#le} \\text{e\#e} \\text{el\#le} \\text{e\#e}
\]

The reason for the modification of the rule in (5) is due to the fact that (11) does not involve an area in focus.

8. Local Unification

Local unification means unification applied to the area in focus. An illustration of this can be seen in (1) below:

\[
(1) U: \text{el}\#\text{le} \Rightarrow \text{el}\#\text{lele}\text{el}\#\text{le}, \ldots
\]

The area in (1) contains an area in focus that stretches over three consecutive rows. This example is from figure 4 and the area in focus stretches between line 2 through 5. The reason for the creation of a unified local area is to get a description of the area in focus that contains all available information, i.e. to unify the area in focus. To get the unified area in focus the general rules for unification are applied, hence (1) is transformed into:

\[
(2) U: \text{el}\#\text{le} \Rightarrow \text{el}\#\text{lele}\text{el}\#\text{lecl}\#\text{le}, \ldots
\]

In (2), the first part of the area in focus is expanded according to the following rule:

\[
(3) \text{el\#le} \Rightarrow \text{el\#le}\text{el}\#\text{le}
\]

An alternative to the unification of (1) into (2) is to apply the rule in (3) and rule (xiii) to (1) which gives the following description of the area in focus:

\[
(4) U: \text{el}\#\text{le} \Rightarrow \text{el}\#\text{le}\text{el}\#\text{le}
\]

The expression in (4) corresponds to a projection that goes along the x-axis organized as columns. (4) is, of course, equivalent to (2).

Local unification works for multiple split points as well, which can be seen in the example in expression (5).

\[
(5) U: \text{el}\#\text{le} \Rightarrow \text{el}\#\text{le}\text{el}\#\text{le}, \ldots
\]

which gives

\[
(6) U: \text{el}\#\text{le} \Rightarrow \text{el}\#\text{le}\text{el}\#\text{le}, \ldots
\]

9. Conclusions

The work presented in this paper has emphasized aspects of the spatial algebra, i.e. unification and spatial reasoning. The purpose of unification is to create a single string that contains all available information from the symbolic image. This powerful aspect of the algebra will also influence even other aspects, such as spatial reasoning.

In the future efforts will be concentrated upon further developing the technique for spatial reasoning on selected applications.

REFERENCES