Rulebase Checking Using a Spatial Representation

Frans Coenen

Department of Computer Science, The University of Liverpool,
Chadwick Building, P.O. Box 147, Liverpool L69 3BX, England.
frans@csc.liv.ac.uk,
WWW home page: http://www.csc.liv.ac.uk/~frans/

Abstract. In this paper we describe an approach to rule base V&V founded on a N-dimensional tesseral representation of space. The advantages offered by the representation are that it is simple, concise and lends itself to computationally effective manipulation of objects (defined in the representation) using standard integer addition and subtraction. The encapsulation of a rulebase in this tesseral representation - as a multi-dimensional "arena" - then provides an appropriate mechanism for the application of spatial algorithms to address rulebase verification. The algorithms make full use of the advantages associated with the representation, and consequently offer a significant reduction in time complexity over more traditional approaches to rulebase verification. In particular this paper demonstrates the potential of tesserally encoded rulebases the full impact of which, in the author's opinion, has yet to be explored.

1 Introduction

In this paper an approach to rulebase verification and validation (V&V) using a multi-dimensional representation is described. The representation is founded on a tesseral approach ([1, 5, 6]) sometimes encountered in spatial reasoning systems ([3]). The advantages offered by this representation are that: (1) it is compact while maintaining expressiveness, (2) it is conceptually simple, and (3) it has a computationally effective integer arithmetic superimposed over it to allow objects captured in this representation to be manipulated.

The approach described also has similarities with earlier work on incidence matrices ([4]) and other similar tabular techniques. A detailed account of the representation and its usage to encapsulate the nature of rulebases is given in section 2. The discussion is illustrated using a rulebase encoded using the Horn clause sub-set of predicate calculus, however, the representation can equally well be applied to any other clausal form. A simpler approach may be used with respect to propositional rulebases. In section 3 the usage of the representation, in the context of rulebase V&V, is illustrated through that application of structural verification algorithm. This algorithm is evaluated in section 4. Some conclusions are presented in section 5.

2 Spatial Representation of Rule Bases

A Horn clause rulebase consists of a set of clauses. Each clause comprising an antecedent and a consequent such that the antecedent is represented by zero, one or more predicates and the consequent a single predicate. Predicates in turn are made up of a predicate name, and zero one or mare arguments which can be constants or variables. In Prolog terms we can describe this in terms of a set of Horn clauses. An alternative representation for such a rulebase, although not as immediately intuitive or supportive of computation, is to consider the rulebase as comprising a number of sets of numeric identifiers:

$$C = \{c \mid 0 \leq c \leq the\ maximum\ number\ of\ unique\ clauses\ in\ the\ rulebase\}$$
$$N = \{n \mid 0 \leq n \leq the\ maximum\ number\ of\ predicates\ that\ can\ be\ contained\ in\ any\ clause\ in\ the\ rulebase\}$$
$$P = \{p \mid 0 \leq p \leq the\ maimum\ number\ of\ unique\ predicates\ in\ the\ rulebase\}$$
$$A = \{a \mid 0 \leq a \leq the\ maximum\ number\ of\ arguments\ that\ can\ be\ contained\ in\ any\ clause\ in\ the\ rulebase\}$$
$$V = \{v \mid 0 \leq v \leq the\ maximum\ number\ of\ unique\ constants\ in\ the\ rulebase\}$$

```
parent(ann,jane).
parent(jane,tom).
parent(ann,susan).

female(ann).
female(susan).
male(tom).

grandParent(X,Y):- parent(X,Z), parent(Z,Y).

grandMother(X,Y):- grandParent(X,Y), female(X).
```

Table 1. Prolog example rulebase

Let us consider the familiar (family) example rulebase given in Table 1. Here we have 8 clauses which we number from 0 to 7, the maximum number of predicates contained in a clause is 3, there are five predicate names (`parent`, `female`, `male`, `grandParent` and `grandMother`) which we number from 0 to 4, the maximum number of arguments is 2, and there are four distinct constants (`ann`, `jane`, `tom` and `susan`) which we number from 0 to 3. Thus:

$$
\begin{aligned}
C &= \{c \mid 0 \leq c \leq 7\} \\
N &= \{n \mid 0 \leq n \leq 2\} \\
P &= \{p \mid 0 \leq p \leq 4\} \\
A &= \{a \mid 0 \leq a \leq 1\} \\
V &= \{v \mid 0 \leq v \leq 3\}
\end{aligned}
$$

We now define each clause as a set whose elements are defined by 5-tuples made up of elements of C, N, P, A and V respectively. Each element of this set then describes an argument within the clause. Thus in the case of the example rulebase:

$$
\begin{aligned}
Clause_0 = \{ &< 0,0,0,0,0 >, < 0,0,0,1,1 > \} \\
Clause_1 = \{ &< 1,0,0,0,1 >, < 1,0,0,1,2 > \} \\
Clause_2 = \{ &< 2,0,0,0,0 >, < 2,0,0,1,3 > \} \\
Clause_3 = \{ &< 3,0,1,0,0 > \} \\
Clause_4 = \{ &< 4,0,1,0,3 > \} \\
Clause_5 = \{ &< 5,0,2,0,2 > \} \\
Clause_6 = \{ &< 6,0,3,0,0 >, < 6,0,3,0,1 >, < 6,0,3,0,2 >, < 6,0,3,0,3 >, \\
&< 6,0,3,0,4 >, < 6,0,3,1,0 >, < 6,0,3,1,1 >, < 6,0,3,1,2 >, \\
&< 6,0,3,1,3 >, < 6,0,3,1,4 >, < 6,1,0,0,0 >, < 6,1,0,0,1 >, \\
&< 6,1,0,0,2 >, < 6,1,0,0,3 >, < 6,1,0,0,4 >, < 6,1,0,1,0 >, \\
&< 6,1,0,1,1 >, < 6,1,0,1,2 >, < 6,1,0,1,3 >, < 6,1,0,1,4 >, \\
&< 6,2,0,0,0 >, < 6,2,0,0,1 >, < 6,2,0,0,2 >, < 6,2,0,0,3 >, \\
&< 6,2,0,0,4 >, < 6,2,0,1,0 >, < 6,2,0,1,1 >, < 6,2,0,1,2 >, \\
&< 6,2,0,1,3 >, < 6,2,0,1,4 > \} \\
Clause_7 = \{ &< 7,0,4,0,0 >, < 7,0,4,0,1 >, < 7,0,4,0,2 >, < 7,0,4,0,3 >, \\
&< 7,0,4,0,4 >, < 7,0,4,1,0 >, < 7,0,4,1,1 >, < 7,0,4,1,2 >, \\
&< 7,0,4,1,3 >, < 7,0,4,1,4 >, < 7,1,3,0,0 >, < 7,1,3,0,1 >, \\
&< 7,1,3,0,2 >, < 7,1,3,0,3 >, < 7,1,3,0,4 >, < 7,1,3,1,0 >, \\
&< 7,1,3,1,1 >, < 7,1,3,1,2 >, < 7,1,3,1,3 >, < 7,1,3,1,4 >, \\
&< 7,2,1,0,0 >, < 7,2,1,0,1 >, < 7,2,1,0,2 >, < 7,2,1,0,3 >, \\
&< 7,2,1,0,4 > \}
\end{aligned}
$$

Note that variables can take any value contained in the set V.

The rulebase can thus be described as a set of sets where each element represents a clause:

$$Rulebase = \{Clauses_1, Clauses_2,\ ...\ Clauses_n\}$$

If we consider the sets C, N, P, A and V to each represent a dimension we can define a 5-D space described as a set, which (for reasons that will become apparent later) we will call `Positive`, that comprises a set of points each defined by a 5-tuple. Thus the set *positive* can be defined as follows:

$$Positive = \{p \mid p : \{C \times N \times P \times A \times V\}\}$$

Interestingly we can extend this concept and also identify a set $Negative$ comprising all 5-tuples where one or more of the coordinates has a negative label. Thus the set $Positive$ could be more accurately described as a subset of a larger set $Universe$ (the universe of discourse) which includes the set $Negative$, i.e.

$$Universe = Positive \amalg Negative$$

(the *disjoint union* of the $Positive$ and $Negative$ sets is equivalent to the set $Universe$).

From a computationally perspective points referenced by a five coordinate system are awkward and cumbersome to handle. It is therefore convenient to compress the 5-tuples into single integers referred to as addresses. We do this using a function $f_{coords2address}$ such that:

$$\begin{aligned}
Dom(f_{coords2address}) &= \{< c, n, p, a, v >\mid c : C, n : N, \\
&\quad\ p : P, a : A, v : V\} \\
Cod(f_{coords2address}) &= \{t \mid t : Integer\} \\
Gr(f_{coords2address}) &= \{<< c, n, p, a, v >, t >\mid t = \\
&\quad\ c + 2^6 n + 2^{12} p + 2^{18} a + 2^{24} v\}
\end{aligned}$$

where Dom indicates the domain of the function, Cod the codomain, and Gr the graph. The effect of the function is to compress the 5-tuple references into a single address with a bit pattern of the form shown in Figure 1. We also define a similar function to “translate” a given address back to a 5-tuple:

$$f_{address2coords}(t) \rightarrow < c, n, p, a, v >$$

Note that:

1. It is not necessary to associate any particular “dimension” with any particular bit grouping.
2. Any other distribution of bits could equally well have been allocated provided that each dimension of interest comprises a continuous sequence of bits.

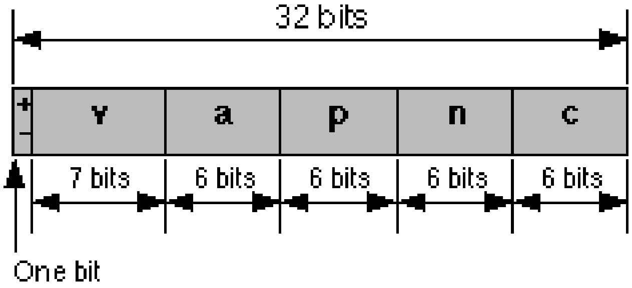

Fig. 1. Address bit pattern using a 32 bit integer

3. To accommodate a larger rulebase a 64 bit integer might be used.
4. The sign bit is included to provide for negative addressing.

In effect we have *tesselated* ($[1, 5, 6]$) the 5-D "argument" space. The advantages of this are as follows:

1. All references are unique and conceptually simple to generate.
2. The representation has the effect of linearising N-dimensional space which in turn has significant benefits with respect to (a) data storage, (b) comparison of sets of addresses and (c) translation through the space.

Thus, assuming the bit pattern given in Figure 1, we can now redefine the set $Universe$ as follows:

$$\begin{aligned} Universe = \{u \mid\ & f_{coords2address}(< c, n, p, a, v >) \rightarrow u, \\ & < c, n, p, a, v > \in \{C \times N \times P \times A \times V\}\} \end{aligned}$$

Returning to our example rulebase we can now express each clause in the rule base in terms of a set of addresses:

$$\begin{aligned} Clause_0 &= \{0, 17039360\} \\ Clause_1 &= \{16777217, 33816577\} \\ Clause_2 &= \{2, 50593794\} \\ Clause_3 &= \{4099\} \\ Clause_4 &= \{50335748\} \\ Clause_5 &= \{33562629\} \\ Clause_6 &= \{12294, 16789510, 33566726, 50343942, 67121158, 274438, 17051654, \\ & \quad 33828870, 50606086, 67383302, 70, 16777286, 33554502, 50331718, \\ & \quad 67108934, 262214, 17039430, 33816646, 50593862, 67371078, 134, \\ & \quad 16777350, 33554566, 50331782, 67108998, 262278, 17039494, 33816710, \\ & \quad 50593926, 67371142\} \end{aligned}$$

$$
\begin{aligned}
Clause_7 = \{&16391, 16793607, 33570823, 50348039, 67125255, 278535, 17055751,\\
&33832967, 50610183, 67387399, 12359, 16789575, 33566791, 50344007,\\
&67121223, 274503, 17051719, 33828935, 50606151, 67383367, 4231,\\
&16781447, 33558663, 50335879, 67113095\}
\end{aligned}
$$

Further compression is achieved by observing that some groups of addresses collectively describe a 5-D (hyper) cube which can be defined using an appropriate infix operator indicating that the prefix operand is the corner address nearest to the origin of the positive space (the address 0), and the postfix operand is the address furthest from the origin. To this end we will use the operator '..' to indicate such a cluster. The example clauses can now be fully described as shown in Table 2. Note also that we have ordered the sets (on the prefix operand in the case of groups of addresses) according to the linear sequence which is a feature of the representation.

$Clause_0 = \{0,\ 17039360\}$
$Clause_1 = \{16777217,\ 33816577\}$
$Clause_2 = \{2,\ 50593794\}$
$Clause_3 = \{4099\}$
$Clause_4 = \{50335748\}$
$Clause_5 = \{33562629\}$
$Clause_6 = \{12294..67383302,\ 70..67371078,\ 134..67371142\}$
$Clause_7 = \{4231..67113095,\ 12359..67383367,\ 16391..67387399\}$

Table 2. Tessellated encoding of example rulebase

2.1 Note on tesseral addition and subtraction

Given a set of tesseral addresses of the form described above we can translate this set through the universe of discourse using simple integer addition and subtraction. To this end we define two functions, $f_{addition}$ and $f_{subtraction}$, as follows:

$$
\begin{aligned}
Dom(f_{addition}) &= \{< X, n >\mid X \subseteq Universe,\\
&\quad n \in Universe\}\\
Cod(f_{addition}) &= \{Y \mid Y, \subseteq Universe\}\\
Gr(f_{addition}) &= \{<< X, n >, Y >\mid \forall x : X, \exists y : Y,\\
&\quad y = x + n\}
\end{aligned}
$$

We define the subtraction function in a similar manner:

$$Gr(f_{subtraction}) = \{<< X, n >, Y >| \forall x : X, \exists y : Y,$$
$$y = x - n\}$$

The computational efficiency with which objects (encoded in the tesseral representation) can be translated through the domain of discourse, using the above $f_{addition}$ and $f_{subtraction}$ functions, is another illustration of the power of the approach.

3 V&V Algorithms

Given the spatial representation described in the forgoing section we use spatial reasoning techniques, such as those described in [3] to look for anomalies in the arguments space. We will illustrate this with an algorithm to address structural verification. Structural verification comprises the identification of anomalies such as subsumption and duplication. We define subsumption as the situation where one rule is a more specialised case of another. Duplication, where two rules are identical, is then a specialised form of subsumption.

The spatial structural verification algorithm described here is founded on ideas concerned with shape fitting, as encountered in tesseral spatial reasoning, where a "shape" described in terms of a set (*Shape*) of tesseral addresses is to be "fitted" into a "location space" also described in terms of a set (*LocationSpace*) of addresses. However, in this case it is not permissible to rotate the shape. The sets *Shape* and *LocationSpace* are defined as follows:

$$Shape = \{s \mid s : X, X \subseteq Univers \wedge 0 \in X\}$$
$$LocationSpace = \{l \mid l : Y, Y \subseteq Positive\}$$

Thus the set *Shape* comprises a set of elements belonging to the set *Univers* with one of the elements equal to the address 0. The latter acts as the reference address for the shape with respect to which it can be manipulated. The set *LocationSpace* is then some subset of the set *Positive*. With respect to structural verification the shape in question describes a clause in the rulebase and the location space the rulebase with the clause in question removed. If the shape can be fitted into the rulebase subsumption exists otherwise it does not.

The derivation of the shape is a two stage process. Given a clause we first remove references to the clause number to arrive at a set *SubShape* using a function $f_{subshape}$:

$$Dom(f_{subshape}) = \{Clause \mid Clause : Rulebase\}$$
$$Cod(f_{subshape}) = \{t \mid t : Positive\}$$
$$Gr(f_{subshape}) = \{< Clause, SubShape >| \forall t : Clause,$$

$$\exists s : SubShape, f_{coords2address}(< 0, n, p, a, v >) \rightarrow s,$$
$$f_{address2coords}(t) \rightarrow < c, n, p, a, v >\}$$

We then need to translate this set of addresses so that the zero address is included. We do this by subtracting one element of the set $SubShape$ (for reasons of convenience we will use the first element encountered) from every member of the set and encapsulate this in a function f_{shape}.

$$Dom(f_{shape}) = SubShape$$
$$Cod(f_{shape}) = Shape$$
$$Gr(f_{shape}) = \{< SubShape, Shape >| \forall s : Shape,$$
$$\exists t : SubShape, s = t - n\}$$

Let us illustrate this using $clause_7$ from Table 2:

$$Clause_7 = \{4231..673752399, 6391..67387399,$$
$$12359..67383367\}$$

then:

$$SubShape = f_{subShape}(Clause_7) \rightarrow \{4224..266368,$$
$$12352..67383360, 16384..67387392\}$$

Note the numeric reordering of the result in line with the linear sequence that is a feature of the tesseral representation and the root of many of its principal advantages. If we now subtract the address 4224 (the first address encountered in this case) from every element we get a shape of the form:

$$Shape = f_{shape}(< SubShape, 4224 >) \rightarrow \{0..262144,$$
$$8128..6734112, 12160..67383168\}$$

The location space set is obtained, more simply, using a function $f_{locationSpace}$:

$$Dom(f_{locationSpace}) = \{Clause \mid Clause : Rulebase\}$$
$$Cod(f_{locationSpace}) = \{s \mid s : Clause, Clause \subseteq Rulebase\}$$
$$Gr(f_{locationSpace}) = \{< Clause, LocationSpace >|$$
$$LocationSpace = Clause \setminus Rulebase\}$$

(The location space is equivalent to the complement of the set $Clause$ in the set $Rulebase$). Thus in the case of $Clause_7$ in our example rulebase (Table 2):

$$\begin{aligned}
&LocationSpace = f_{locationSpace}(Clause_7) \rightarrow \{0, 2, 70..67371078, 134..67371142,\\
&\quad 4099, 12294..67383302, 16777217, 17039360, 33816577, 33562629, 50593794,\\
&\quad 50335748, 33562629\} = Clause_7 \setminus Rulebase = \{4231..673752399,\\
&\quad 6391..67387399, 12359..67383367\} \setminus \{0, 2, 70..67371078, 134..67371142,\\
&\quad 4099, 4225..67113095, 12294..67383302, 12359..67383367, 16391..67387399,\\
&\quad 16777217, 17039360, 33816577, 33562629, 50593794, 50335748, 33562629\}
\end{aligned}$$

We then use a shape fitting algorithm such as that described in [3] to determine if the shape can be "fitted" into the location space. If so subsumption exists. We can summarise this shape fitting algorithm as follows using a function $f_{subsumption}$:

$$\begin{aligned}
&Dom(f_{subsumption}) = < Shape, LocationSpace >\\
&Cod(f_{subsumption}) = Boolean = \{true, false\}\\
&f_{subsumption}(< Shape, LocationSpace >) =\\
&\quad true\ if\ \exists n : LocationSpace\\
&\qquad f_{addition}(< Shape, n >) \subseteq LocationSpace\\
&\quad false\ otherwise
\end{aligned}$$

In the example the shape cannot be fitted into the location space therefore we conclude that no duplication/subsumption exists.

4 Evaluation

Intuitively, to implement structural verification, every clause in the rulebase must be compared with every other clause in the rulebase. This comparison also requires that given a pair of clauses every predicate (and every argument in every predicate) in one must be compared with every predicate (and every argument in every predicate) in the other. Given even a simple rulebase the time complexity required by such an algorithm will be significant. A crude measure of this complexity is given by:

$$\frac{NumberOfClauses^2 - NumberOfClauses}{2}$$

Using the algorithm defined in section 3 the time complexity is significantly reduced, and can be quantified as the number of clauses in the rulebase. There is of course an overhead associated with the implementation of the shape fitting mechanism but this may considered to be roughly equivalent to the overheads associated with the detailed comparison of predicates. It is therefore considered that a significant saving in time complexity is achieved when the proposed spatial algorithm is used to check for structural anomalies.

5 Conclusion

In this paper we have shown how a rulebase can be encoded using a 5-D tesseral representation, and have illustrated the advantages offered by this encoding with two "spatial" algorithms - the first to address structural verification and the second directed at inference verification. The principal advantage of the encoding is the saving in time complexity that are gained when algorithms founded on the representation are applied to address rulebase V&V. Work is currently centred on techniques to identify contradictions and inconsistencies in the rulebase by ascertaining the possible values for variables and matching this with the known set of possible values derived from a domain expert. Work to date indicates that the value passing mechanism adopted in Prolog style rulebases necessitates that an additional sixth dimension be included which encapsulates variable names. Work is also underway considering the advantages offered by the tesseral approach as a model of computation in its own right. It is also worth observing that the tesseral representation can be used to encode any tabular approach to rulebase V&V, and not just incidence matrix style approaches. Perhaps the most significant point that this paper establishes is that there my be great potential in tesserally encoded rulebases the fill impact of which has yet to be explored.

References

1. S.B.M. Bell, B.M. Diaz, F.C. Holroyd and M.J.J. Jackson (1983). Spatially Referenced Methods of Processing Raster and Vector Data. Image and Vision Computing, 1(4), pp211-220.
2. Coenen, F.P. (1995). An Advanced Binary Encoded Matrix Representation for Rulebase Verification. Journal of Knowledge-Based Systems, Vol 8, No 4, pp201-210.
3. Coenen, F.P., Beattie, B., Bench-Capon, T.J.M, Diaz, B.M. and Shave. M.J.R. (1997). A tesseral Approach to N-Dimensional Spatial Reasoning. In Hameurlain, A. and Tjoa, A.M. (Eds), Database and Expert Systems Applications, (Proceedings DEXA'97), Lecture Notes in Computer Science 1308, Springer Verlag, pp633-642.
4. Coenen, F.P. and Dunne, P.E. (1997). Verification and Validation of Rulebases using a Binary Encoded Incidence Matrix Technique. Proceedings EUROVAV'97, Katholieke Universiteit Leuven, Belgium, pp43-53.
5. B.M. Diaz and S.B.M. Bell (1986). Spatial Data Processing Using Tesseral Methods. Natural Environment Research Council publication, Swindon, England.
6. I. Gargantini (1982). Linear Octrees for Fast Processing of Three Dimensional Objects. Computer Graphics and Image Processing, 20, pp365-374.