Formalizing the Qualitative Superposition of Rectangles in Proof Assistant Isabelle/HOL

Fadoua Ghourabi and Kazuko Takahashi
Graduate School of Science and Technology, Kwansei Gakuin University, Nishinomiya, Japan
{ghourabi, ktaka}@kwansei.ac.jp

Keywords: Knowledge Representation and Reasoning, Qualitative Spatial Reasoning, Theorem Proving, Isabelle/HOL, Rectangle Superposition.

Abstract: We formalize and verify the superposition of rectangles in Isabelle/HOL. The superposition is associated with the arrangement of rectangular software windows while keeping some regions visible and other hidden. We adopt a qualitative spatial reasoning approach to represent these rectangles and the relations between their regions. The properties of the model are formally proved and show some characteristics of superposition operation. Although, this work is limited to 29 structures of rectangles, the superpositions produce hundreds of cases that are tedious to tackle in Isabelle/HOL. We also explain our strategy to optimize the proofs.

1 INTRODUCTION

Qualitative spatial reasoning (QSR) methods describe the objects of the space qualitatively. Such approach provides a low demand in numerical computation and, furthermore, contains enough expressiveness to teach a program to reason about the spatial objects. QSR methods target various applications, such as geographical information systems and robotics (Liu and Daneshmend, 2004; Bennett, 1996). The importance of QSR methods leads us to address their reliability and correctness. While various research works focus on developing QSR methods, little attention is given to their formal verification.

On the other hand, proof assistants are increasingly used to formalize mathematical models (Avigad and Harrison, 2014). With the help of proof assistants, mathematical proofs are formally verified where each step is transparent and, thus, the involved logical reasoning can be checked. In the case of the field of QSR, there are two major approaches: the topological knowledge about objects (e.g. RCC calculus (Randell et al., 1992b)), and the positional knowledge about objects (e.g. direction relations (Frank, 1991; Clementini et al., 1997)). Both of RCC and direction relations are formalized using first-order logic and relation algebra, which motivate us to use them in automated reasoning by a proof assistant such as Isabelle/HOL (Nipkow et al., 2002). The only work that tackle the formal verification in QSR of which we are aware dates back to early 90s. The OTTER proof system for first-order logic has been used to assist pen-and-paper proofs of theorems in RCC (Randell et al., 1992a). As far as we know, there is no (full) formal presentation of QSR methods using proof assistant.

The purpose of this research is to extend the use of proof assistants to the field of QSR. We focus on window allocation problem. When using software, we often rearrange windows by dragging, resizing, superposing, etc. until obtaining better visibility. A superfluous information in a window can be hidden and superposed by a relevant information in another window. A systematic method of superposing windows is proposed in (Konishi and Takahashi, 2012). The window parts that should be visible are pre-specified by the user. The window and its parts form a rectangular structure. The problem of arranging software windows is regarded as the problem of superposing rectangles while keeping some pre-specified parts visible. The rectangles are regarded as spatial objects, and the superposition is examined from a QSR point of view.

This paper is based on the method presented in (Konishi and Takahashi, 2012), and the contribution that we seek is twofold.

1. The original qualitative representation of rectangles is simple but not expressive enough. When checking properties about superposition, extra conditions are defined to detect degenerate situations. Due to the numerous cases of superposition, the question is whether all the degenerate situations are covered. We therefore propose a more expressive representation based on matrices

of direction relations. The matrix representation is more natural as the correspondence between the parts of a rectangular structure and the elements of its matrix representation is straightforward. Furthermore, the extra conditions are not required and a single set membership test is sufficient.

2. We formalize our revised method of superposition in Isabelle/HOL. Proof assistants are interactive systems. The formal proofs are, consequently, tedious to do if the model present hundreds of cases. We therefore take into consideration the practicality of proving in Isabelle/HOL despite the numerous cases of superposition. Our proof strategy relies on establishing equivalence relations between the qualitative representations of rectangles and grouping them into classes.

The structure of the rest of the paper is as follows. In Section 2, we summarize the notations that we use. In Section 3, we define matrices in Isabelle/HOL. In Section 4, we explain the operations over matrices. In Section 5, we give matrix representation to rectangular units. In Section 6, we apply matrix superposition to compute the superposition of units. In Section 7, we introduce properties about superposition of units and explain how we proceed in order to prove them. In Section 8, we conclude with remarks on future directions of research.

2 NOTATION

A matrix m of dimension $p\times q$ has p rows and q columns, where p, $q \in \mathbb{N}$. We write for short "$p\times q$ matrix m". The expression $m(i, j)$ denotes the element of m at the i-th column and the j-th row, where $0 \le i < q$ and $0 \le j < p$. Let $\mathcal{S}$ be the set of matrices, m be a matrix in $\mathcal{S}$ and $f{:}\mathcal{S} \to \mathcal{S}$ be an unary function, we use m^f to mean the term $f(m)$ and f^n to denote the $n(\in \mathbb{N})$ compositions of f. We recurrently use the notation m^{f^n} for $f^n(m)$.

The formalization presented in this paper is done in the proof assistant Isabelle/HOL (Nipkow et al., 2002). The choice of the tool is made principally due to the availability of powerful libraries for reasoning with equivalence classes (Paulson, 2006) and mathematical operations on matrices (Sternagel and Thiemann, 2010) on which the proposed qualitative representation depends. In the following, we present the elements of syntax used in this paper. Isabelle/HOL provides a rich collection of formalized theories, useful proof tactics (e.g. natural and structural induction, and case splitting), elaborate techniques for pattern matching and term rewriting, etc. Furthermore, Isabelle/HOL is a strongly typed system. The expression "m::$'$a *list*" is a type constraint over a variable m. The variable m is of type list whose elements are of variable type $'a$. The type of a function f is written "$f{::}\tau_1 \Rightarrow \ldots \Rightarrow \tau_n \Rightarrow \tau_{n+1}$". In this paper, we use Isabelle/HOL expression "$f\ x_1\ \ldots\ x_n$" to denote the term $f(x_1, \ldots, x_n)$. A list is represented by a sequence of elements between square brackets, i.e. $[e_0, e_1, \ldots, e_n]$. The constructor "#" adds an element to a list, i.e. $v\#[e_0, e_1, \ldots, e_n] = [v, e_0, e_1, \ldots, e_n]$. The i-th element of a list m is given by "m ! i". For an equivalence relation r, the expression $r``\{x\}$ in Isabelle/HOL denotes the equivalence class $[x]_r$.

The formulas that we prove are written using common mathematical symbols. We explain some of the proof in Isabelle/HOL using natural language. We use Isar, which is an extension of Isabelle/HOL, to write structured and human-readable proofs (Wenzel, 1999). The proofs in the classical Isabelle/HOL proof style intertwines with Isar proofs.[1]

3 FORMALIZATION OF MATRICES

In Isabelle/HOL, no type is defined for matrix. We use the types "*'a vec*" and "*'a mat*" provided by the Matrix_Arith theory (Sternagel and Thiemann, 2010).

```
type_synonym 'a vec = 'a list
type_synonym 'a mat = 'a vec list
```

A matrix is implemented as a list of lists, i.e. a list of columns. For example, list $[[a, b],[c, d],[e, f]]$ is of type "*'a mat*", and represents the 2×3 matrix $\begin{pmatrix} a & c & e \\ b & d & f \end{pmatrix}$.

3.1 Dimension

The following issues arise when working with matrices as list of lists. First, a matrix of type "'a mat" has a dimension $p\times q$, i.e. the length of the whole list is q, and each column list has p elements. To make sure that we work with mathematically valid matrices, we use predicate *mat::nat⇒nat⇒'a mat⇒bool* (Sternagel and Thiemann, 2010).

$$\mathit{mat}\ p\ q\ m \triangleq (\mathit{length}\ m = q) \wedge (\forall\, x \in \mathit{set}\ m.\ \mathit{length}\ x = p), \tag{1}$$

[1]For a closer look at the proofs, our Isabelle/HOL theory files are available at http://ist.ksc.kwansei.ac.jp/~ktaka/SuperpositionTheory/.

where *length* and *set* are Isabelle/HOL functions that compute the length of a list and the set of elements of a list, respectively.

Second, empty matrix (i.e. [ ]) and matrices with empty columns (i.e. [[ ]], [[ ],[ ]], etc.) are of type “*′a mat*”. We want to exclude these matrices for the following reasons. In the case of empty matrix, the second part of the conjunction in definition (1) causes vacuous truth that we wish to avoid (i.e. “*mat p* 0 [ ]” holds for any p). Moreover, the transpose of a matrix with empty columns gives rise to the empty matrix, hence we wish to avoid these cases too. We therefore restrict ourselves to a set M of type “*′a mat set*” whose elements satisfy the following condition.

$$m \in M \triangleq \exists p\ q{::}nat.\ mat\ (Suc\ p)\ (Suc\ q)\ m$$

The theorems that we prove in this work are defined for matrices $m \in M$. In the premises, we impose the condition $m \in M$ from which we can deduce the dimension $p \times q$ by adding the following Isar line.

```
from ‘m ∈ M‘ obtain p q
where mat p q m and 0 < p and 0 < q
by (rule M.cases, simp)
```

The command “**by** (rule M.cases, simp)” proves that the dimension of m is $p \times q$ and that p and q are strictly positive.

3.2 Matrix Equality

The equality is established between two matrices of the same dimension. In order to assert that two matrices are equal, we either check that all their respective elements are equal or that their respective column lists are equal, depending on the proof strategy. Lemmas about the equality of matrices are provided by the Matrix_Arith theory.

4 OPERATIONS ON MATRICES

We explain two operations on matrices, namely rotation[2] and superposition.

4.1 Rotation

Before giving a formal definition of rotation, we first examine what does rotation of a matrix intuitively mean. An operation of rotation has a spatial connotation. It involves a circular movement of objects defined by parameters such as the center of rotation, angle of rotation, etc. If we picture a matrix as an object (not only as data-structure), we can perform a $\frac{\pi}{2}$ counter-clockwise rotation. For instance, if we rotate the matrix $\begin{pmatrix} a & c & e \\ b & d & f \end{pmatrix}$ by $\frac{\pi}{2}$ in a counter-clockwise direction, we obtain $\begin{pmatrix} e & f \\ c & d \\ a & b \end{pmatrix}$.

From a computational point of view, function $\rho = R \circ T$ represents the rotation operation over matrices. Functions T and R denote the transpose and the reverse. For a matrix $m \in M$, $m^{\rho} = m^{R \circ T}$ is obtained by reversing the columns of m^T. In the previous example, the transpose is $\begin{pmatrix} a & b \\ c & d \\ e & f \end{pmatrix}$. Then, the result of reversing the columns of the transpose is $\begin{pmatrix} e & f \\ c & d \\ a & b \end{pmatrix}$. Functions *transpose*, *reverse* and ρ implement T, R and ρ in Isabelle/HOL. We use function *transpose* :: *nat* ⇒ *′a mat* ⇒ *′a mat* given in the Matrix_Arith theory, and we define functions *reverse* and ρ (both of type *′a mat* ⇒ *′a mat*) as follows.

$$reverse\ m \triangleq map\ rev\ m$$

$$\rho\ m \triangleq reverse\ (transpose\ (nr\_mat\ m)\ m)$$

Function *rev* reverses a list, and function *reverse* applies *rev* to all the column lists. Note that function *transpose* has two arguments. The first argument is a natural number that corresponds to the number of rows, which is computed by the function call “*nr_mat m*”.

Furthermore, we can perform n successive rotations of matrix m by computing m^{ρ^n}, where $n \in \mathbb{N}$. In Isabelle/HOL, we use the notation for function power, i.e. “ρ ^^*n*”, to implement successive rotations. We prove a collection of useful properties about rotation operation that are stated as inference rules in $(2) - (7)$ depicted in Fig. 1.

The variables in the rules $(2) - (7)$ are universally quantified, and their types are understood from the context. Rule (5) expresses the transitivity of ρ^n. A matrix is invariant by a number of rotations multiple of 4 (c.f. rules (4) and (6)). If matrix m_2 is obtained by n rotations of matrix m_1, then m_1 can be obtained back by $3 \times n$ rotations of m_2 (c.f. rule (7)).

Based on rotation, we establish a relation *rot_rel*. In other words, if a matrix m_2 is obtained by n rotations of matrix m_1, then we write $(m_1, m_2) \in rot\_rel$.

$$\begin{aligned} rot\_rel \triangleq \{(m_1, m_2) | \forall m_1\ m_2. \exists n{::}nat. \\ m_1 \in M \land m_2 \in M \land m_2 = m_1^{\rho^n}\} \end{aligned} \tag{8}$$

We furthermore prove that the relation *rot_rel* is an equivalence relation over the set *M*.

[2]Here, operation of rotation over matrices is not to be confused with “rotation matrix” used to compute the new coordinates of a geometrical object that undergoes rotation in Euclidean space.

$$\frac{m \in M \quad mat\ p\ q\ m \quad even\ n}{mat\ p\ q\ m^{\rho^n}} \quad (2) \qquad \frac{m \in M \quad mat\ p\ q\ m \quad \neg even\ n}{mat\ q\ p\ m^{\rho^n}} \quad (3)$$

$$\frac{m \in M}{m^{\rho^n} = m^{\rho^{n\ mod\ 4}}} \quad (4) \qquad \frac{m \in M}{m^{\rho^{n_1+n_2}} = m^{\rho^{n_1} \circ \rho^{n_2}}} \quad (5)$$

$$\frac{m \in M}{m^{\rho^{4\times n}} = m} \quad (6) \qquad \frac{m_1 \in M \quad m_2 \in M \quad m_2 = {m_1}^{\rho^n}}{m_1 = {m_2}^{\rho^{3\times n}}} \quad (7)$$

Figure 1: Rules on rotations of matrices.

lemma `equiv M rot_rel`

Relation *rot_rel* is obviously reflexive over M since $m = m^{\rho^{4\times n}}$. From property (5), we deduce that *rot_rel* is transitive, and from property (7) that *rot_rel* is symmetric.

4.2 Superposition

The other operation that we consider is superposition of matrices. Intuitively, we superpose two matrices m_1 and m_2 by "putting" m_2 onto m_1. The superposition requires that both of m_1 and m_2 are of the same dimension $p \times q$. The superposition amounts to "putting" elements of m_2 onto elements of m_1.

Let f and s be two functions of type $'a\ mat \Rightarrow 'a\ mat \Rightarrow 'a\ mat$ and $'a \Rightarrow 'a \Rightarrow 'a$, respectively. We define predicate *is_superposition* as follows.

$same\_dimension\ p\ q\ m_1\ m_2 \triangleq mat\ p\ q\ m_1 \wedge mat\ p\ q\ m_2$

$is\_superposition\ f\ s \triangleq$
$\forall m_1\ m_2 \in M. \forall p\ q\ i\ j{::}nat.$
$(same\_dimension\ p\ q\ m_1\ m_2) \rightarrow i < q \rightarrow j < p \rightarrow$
$(f\ m_1\ m_2)(i,\ j) = s\ m_1(i,\ j)\ \ m_2(i,\ j)$

If "*is_superposition f s*" holds then we write that f is a superposition with respect to s. For instance, addition over matrices is a superposition with respect to addition operation.

lemma `is_superposition mat_plus plus`

The implementation of f that we use in this paper is the function *map_thread* defined as follows.

```
fun map_thread::('a ⇒'a ⇒'a)⇒'a mat⇒
'a mat⇒ 'a mat
where map_thread s m1 m2 =
map (λ(v1,v2). map (λ(a1,a2). s a1 a2)
(zip v1 v2)) (zip m1 m2)
```

Since matrices are implemented as lists, the superposition "*map_thread*" applies function s to pair-wise combinations of elements of m_1 and m_2. Function s is applied to elements of m_1 and m_2 that have the same position (i.e. same column and row numbers). For instance, calling "(*map_thread s*) [[a_1, a_2, a_3], ...] [[b_1, b_2, b_3], ...]" gives rise to the matrix "[[s a_1 b_1, s a_2 b_2, s a_3 b_3], ...]". We show that "*map_thread s*" is a superposition with respect to *s*.[3]

lemma `is_superposition (map_thread s) s`

Furthermore, we prove properties about superposition of rotation of matrices shown in rules (9) – (11). In the left-side of the equation in (9), we compute the superposition of matrices m_1 and m_2 rotated by the same number n. We can take ρ^n outside as shown in the right-side of the equation in (9), i.e. we first superpose m_1 and m_2 then rotate the result by n.

More relevant to our formalization is showing how superposition behaves when we rotate m_1 and m_2 by distinct numbers n_1 and n_2, respectively. Depending on whether n_1 is less than n_2, we distinguish the two cases in (10) and (11). In case $n_1 \leq n_2$, we can take ρ^{n_1} outside as shown in the right-side of (10). Therefore, matrix m_2 is rotated by $n_2 - n_1$. Rules (10) and (11) are used as substitution rules to replace the occurrence of left-side of the equality by the right-side of the equality. The two rules are important to optimize the proofs which we will explain later in Sect. 7.2.

Recall that from properties (2) and (3), the dimension of matrix ${m_1}^{\rho^{n_1}}$ depends on whether n_1 is even. Hence, ${m_1}^{\rho^{n_1}}$ and ${m_2}^{\rho^{n_2}}$ are not necessary of the same dimension, and therefore we cannot always perform superposition. A sufficient condition, but not necessary, is to consider only $p \times q$ square matrices, where $p = q$. So far, all the lemmas that we have proved are for arbitrary $p \times q$ matrices, but, as we see in next sections, only square matrices are used to formalize the superposition of rectangles.

The qualitative spatial reasoning approach to the problem of superposition of software windows is based on operations of rotation and superposition of matrices. First, a window is given a qualitative representation using matrices, which is the subject of the next section.

[3]The type of function s is not specified in the statement of the lemma since it is inferred by Isabelle/HOL.

$$\frac{m_1 \in M \quad m_2 \in M}{(map\_thread\ s)\ {m_1}^{\rho^n}\ {m_2}^{\rho^n} = ((map\_thread\ s)\ m_1\ m_2)^{\rho^n}} \tag{9}$$

$$\frac{m_1 \in M \quad m_2 \in M \quad n_1 \leq n_2 \quad mat\ p\ p\ m_1 \quad mat\ p\ p\ m_2}{(map\_thread\ s)\ {m_1}^{\rho^{n_1}}\ {m_2}^{\rho^{n_2}} = ((map\_thread\ s)\ m_1\ {m_2}^{\rho^{n_2-n_1}})^{\rho^{n_1}}} \tag{10}$$

$$\frac{m_1 \in M \quad m_2 \in M \quad n_2 < n_1 \quad mat\ p\ p\ m_1 \quad mat\ p\ p\ m_2}{(map\_thread\ s)\ {m_1}^{\rho^{n_1}}\ {m_2}^{\rho^{n_2}} = ((map\_thread\ s)\ m_1\ {m_2}^{\rho^{3\times(n_1-n_2)}})^{\rho^{n_1}}} \tag{11}$$

Figure 2: Inference rules on matrix superposition.

5 QUALITATIVE REPRESENTATION OF UNIT

The spatial object that we investigate is the software window modelled as a rectangle, called *unit*. Some parts of a unit are required to be visible, which are modelled as white rectangular plates. The parts that can be hidden are black rectangular plates. The size of a unit is unfixed, and changes in a fashion similar to the way software windows are shrunk or expanded. We explain the qualitative representation of a unit that reflects its structure, i.e. locations of its black and white plates.

5.1 Black Plates

Let U be a rectangular unit of unfixed length l and unfixed height h, i.e. of size $l \times h$. In this paper, we use units with at most two black plates, and we consider the following two assumptions about the black plates of a unit U.

1. If U has one black plate p, then the size of p is either $l \times v$ or $u \times h$, where $u \leq l$ and $v \leq h$. Each of the units in Fig. 3(a) and 3(b) has one black plate that stretches along the length/height of the unit.
2. If U has two black plates p_1 and p_2, then they are perpendicular, and at least one of them is of size $u \times h$ or $l \times v$, where $u \leq l$ and $v \leq h$. Furthermore, p_1 and p_2 must be overlapping. The overlapping part forms a rectangular shared area denoted by $p_1 \sqcap p_2$ (e.g. the units in Fig. 3(c) and 3(d)).

5.2 Regions of Unit

We work with units that are located on the 2D plane. The 2D plane is the grey region, simply denoted by g. From the white and black plates, we compute regions

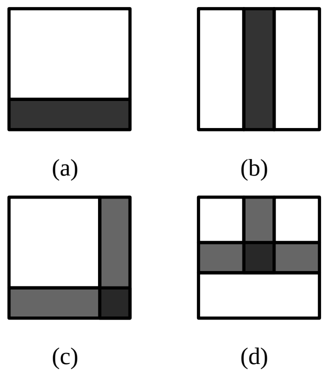


Figure 3: Examples of units with one black plate ((a) and (b)) and with two perpendicular and overlapping black plates ((c) and (d)).

of unit, namely white and black regions that we denote w and b, respectively. We define the datatype *rg* as follows.

datatype `rg = w | b | g | N`

The region N is called undefined region that is relevant for checking the success of superposition which we explain in Sect. 6. The regions w and b of a unit U are determined as follows.

(a) A white plate of U is a w region.

(b) If p is the only black plate of U, then p is also the only b region of U.

(c) If p_1 and p_2 are two distinct black plates of U, then together they generate 3, 4 or 5 b regions depending on their placements (cf. Fig. 4). Note that the area $p_1 \sqcap p_2$ is one of the b regions generated by p_1 and p_2.

Next, we define the core of a unit. The core region of unit U, denoted by C_U, is a b region. If U has only one black plate p, then p is the core C_U. Otherwise, let p_1 and p_2 be the two black plates of U, then the core of U is the shared area between p_1 and p_2, i.e. $C_U = p_1 \sqcap p_2$. Since a unit has at most two black plates as established in the assumptions in the previous section,

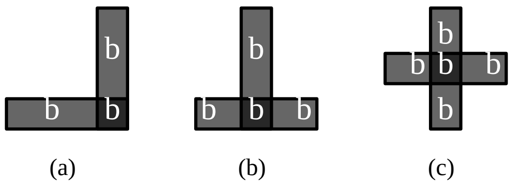


Figure 4: Black plates, b regions and the core regions.

then a unit has one and only one core. In our figures, the cores are highlighted in darker black (cf. Figs. 3 and 6).

5.3 Representation

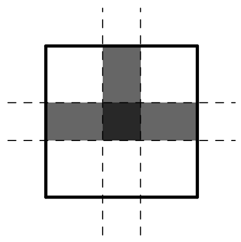
One common approach to represent positional knoweledge is the use of matrix representation. In particular, the *object interaction matrix* (OIM) (Chen et al., 2010) encodes the direction relations between spatial objects. In this paper, we use OIM matrix to represent the positions of the unit regions w.r.t. the core region. Recall that the core C_U is a rectangular black area, thus it has 4 edges and 4 vertices. Extending the edges of C_U divides the plane into 9 tiles where the central tile is bounded and 8 are unbounded (e.g. the tiling in Fig. 5). The tiles decompose the plane into the 9 regions of direction relations, i.e. *up_left, left, bottom_left, up, center, bottom, up_right, right, bottom_right.* The core region occupies the central bounded tile. The intersection of the tiles and the rectangular unit determines the direction relations between the core C_U and the 8 regions of the unit that are connected to the core.[4]

The reference of direction system, being the core, is the center of the matrix representation. The remaining elements of the matrix are the values of the intersections of the unit and the 8 unbound tiles. We use the following 3×3 matrix to represent the positions of the 8 regions that surround the core.

$$\begin{pmatrix} up\_left & up & up\_right \\ left & core & right \\ bottom\_left & bottom & bottom\_right \end{pmatrix}$$

We place a unit in the 2D plane and we substitute the values g, b, w for the elements of the above matrix. The obtained matrix is the qualitative representation of the unit. For instance, the qualitative representations of the units in Fig. 3 are $\begin{pmatrix} g & w & g \\ g & b & g \\ g & g & g \end{pmatrix}$, $\begin{pmatrix} g & g & g \\ w & b & w \\ g & g & g \end{pmatrix}$, $\begin{pmatrix} w & b & g \\ b & b & g \\ g & g & g \end{pmatrix}$ and $\begin{pmatrix} w & b & w \\ b & b & b \\ w & w & w \end{pmatrix}$, respectively.

[4]Regions that share a point are considered connected.



$$\begin{pmatrix} w & b & w \\ b & b & b \\ w & w & w \end{pmatrix}$$

Figure 5: Tiling of the plane and the OIM matrix of a unit.

In Isabelle/HOL, we provide a set *rgM* of all the 3×3 matrices with b core region, and prove that *rgM* is a subset of M defined in Sect. 3.1.

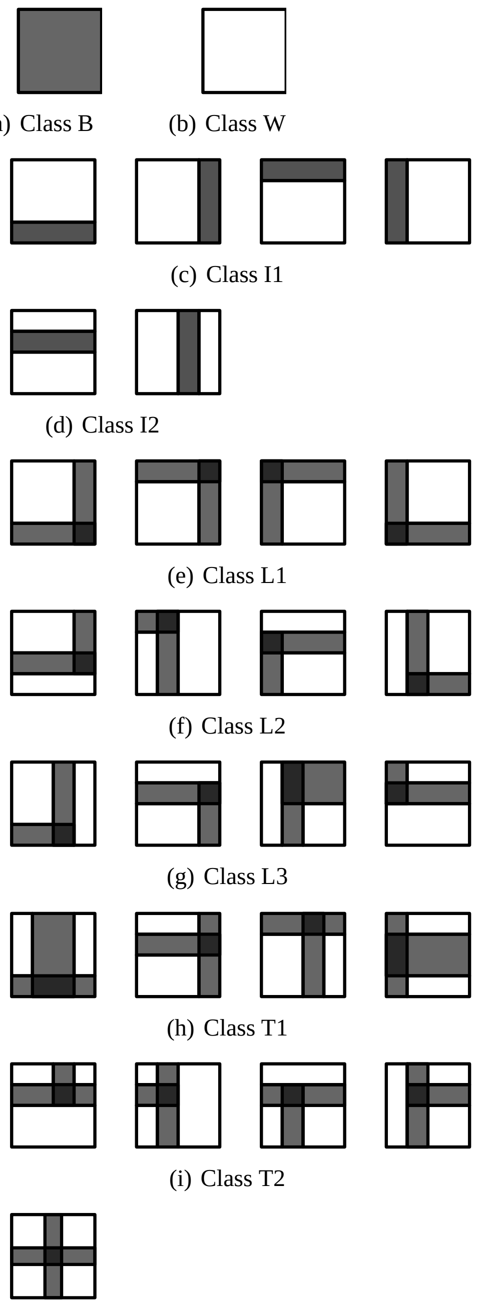
```
definition core::rg mat ⇒ rg
where core m = m!1!1

definition rgM ::rg mat set where
rgM = {m.  (mat 3 3 m)∧(core m = b)}

lemma rgM ⊂ M
```

5.4 Valid Units

There are 29 possible cases of fitting one or two black plates in a rectangular unit. They are depicted in Fig. 6. We call them *valid units*. Since we deal with qualitative representation of units where size does not matter, we observe that some units are obtained by rotating others.

6 SUPERPOSITION OF UNITS

Superposition of units is a non-symmetric binary operation. Superposing two units U_1 and U_2 means that we put U_2 onto U_1 while keeping the w regions of U_1 visible. The superposition, that we define in this paper, operates by putting the core C_{U_2} onto the core C_{U_1}. The cores C_{U_1} and C_{U_2} are not necessarily of the same size. Operations of shrinking, expanding, lengthening are performed on U_1 and U_2 so that their cores fit. Consequently, the sizes of the units and their regions change. Such operations may also affect the amount of visible information. We assume that the operating system or the software application automatically generates scrollbars for the viewing. For the simplicity of our examples, we only modify the size of U_2 and its core. The changes in the size do not affect the matrix representation since it is a qualitative representation.


(a) Class B (b) Class W

(c) Class I1

(d) Class I2

(e) Class L1

(f) Class L2

(g) Class L3

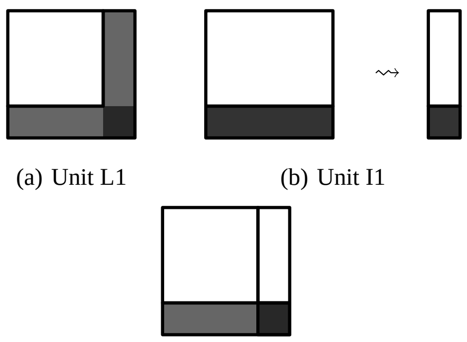
(h) Class T1

(i) Class T2

(j) Class PLUS

Figure 6: Valid units grouped into 10 equivalence classes.

The result of superposition is a unit U_3 whose core is $C_{U_3} = C_{U_2} = C_{U_1}$. The superposition of U_1 and U_2 is computed from the superposition of their respective matrix representations. In Sect. 4.2, we introduced superposition of matrices where function *s* has not yet been defined. In the following section we discuss an implementation of *s*.



(a) Unit L1 (b) Unit I1

(c) Result of I1 on L1

Figure 7: Superposition of unit I1 onto unit L1.

6.1 Function *on*

We define a function *on* that computes superposition of regions.

```
fun on::rg ⇒ rg ⇒ rg
where
on g u = u  | on u g = u |
on N _ = N | on _ N = N |
on w _ = N | on _ w = w | on _ b = b
```

The equation "*on x y = z*" means that region *y* is to be put on region *x*, and the result is region *z*. Function *on* is defined using Isabelle/HOL function definition **fun**. The order of appearance of the equations "*on x y = z*" matters. The pattern matching of "*on w b*" succeeds first with the 5th equation and thus gives rise to *N*, i.e. the superposition should not be allowed since a *w* region is hidden.

The superposition of two matrices m_1 and m_2 is achieved by calling "*(map_thread on)* m_1 m_2".

Example. We take the two units of type L1 and I1 in Fig. 7(a) and 7(b) with matrix representations $m_1 = \begin{pmatrix} w & b & g \\ b & b & g \\ g & g & g \end{pmatrix}$ and $m_2 = \begin{pmatrix} g & w & g \\ g & b & g \\ g & g & g \end{pmatrix}$, respectively. We resize I1, then we put C_{I1} onto C_{L1}. The computation of "(*map_thread on*) m_1 m_2" gives rise to $\begin{pmatrix} w & w & g \\ b & b & g \\ g & g & g \end{pmatrix}$ which corresponds to the unit in Fig. 7(c).

The unit in Fig. 7(c) has one more *b* region besides the core. The *b* region is adjacent to the core to the left. In other words, it shares an edge with the core. We merge it with the core to form one *b* region. The conditions for the merge as well as its impact on the matrix representation of a unit are discussed in the next section.

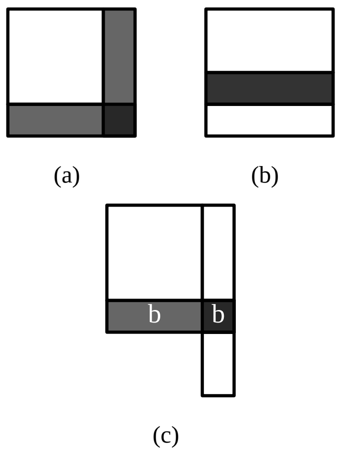


Figure 8: The result of putting I2 type unit on L1 type unit.

6.2 Merging Black Regions

The merge is necessary only if the following two conditions apply.

1. There is only one b region that is connected to the core by an edge. In other words, there is a unique b region located either up or left or bottom or right to the core. Fig. 7(c) illustrates such situation. Besides the core, only left region is black.
2. If the above condition holds, then we check whether the b region and the core are connected to the same regions. Referring to Fig. 7(c) with matrix representation $\begin{pmatrix} w & w & g \\ b & b & g \\ g & g & g \end{pmatrix}$, both b region and core are connected to the same upper w region, and to the same bottom g region. In that case we can merge the left, up left and bottom left regions into one g region. Hence, the matrix representation becomes $\begin{pmatrix} g & w & g \\ g & b & g \\ g & g & g \end{pmatrix}$. In the example illustrated by Fig. 8, we put I2 unit on L1 unit. The result of superposition (cf. Fig. 8(c)) has the matrix representation $\begin{pmatrix} g & w & w \\ g & b & b \\ g & w & g \end{pmatrix}$. The right region is the only b region besides the core. The core is connected to a w bottom region while the right region is connected to a g bottom region. We, therefore, cannot merge the right, up right and bottom right regions into one g region.

The final result is merging the superposition (if applicable) which is given by function *puton* = *merge* ∘ (*map_thread on*).

7 ON PROVING PROPERTIES ABOUT *puton*

7.1 The Properties

We define properties to answer questions about the final result of superposing two units. Let m_1 and m_2 be matrices in *rgM*, the properties of applying "*puton* m_1 m_2" that we want to check are the following.

Success. Are all the w regions visible?

Effectiveness. Does the final result of superposition has only one b region?

Validity. Is the final result of the superposition a valid unit?

The superposition proceeds by putting the core of a unit onto the core of another one. This does not guarantee that all the w regions are visible. In order to judge whether a superposition is successful, we compute the occurrence of N in the matrix representation obtained by *puton*.

```
definition successful::rg mat ⇒ bool where
successful m ≡ (occurs N (concat m))=0
```

Naturally, effectiveness and validity are checked for successful *puton*. "*puton* m_1 m_2" is said effective if its result has only one b region.

```
definition effective::rg mat ⇒ bool where
effective m ≡ (occurs b (concat m)) = 1
```

"*puton* m_1 m_2" is valid if its result is one of the 29 units in Fig. 6.

```
definition valid::rg mat ⇒ bool where
valid m ≡ ∃V::rg mat. V ∈ U0 ∧
                  (m ∈ rot_rel''{V})
```

7.2 Proof Strategy

Given the units in Fig. 6, the number of all the pairwise combinations is $29 \times (29-1) + 29 = 841$. Checking properties of success, effectiveness and validity for all possible superpositions involves numerous computations of *puton*. It is crucial to design an efficient proof strategy to prove a large number of properties.

Earlier in Sect. 4.2, we explained the results (10) and (11) about superposition of rotations of matrices. Namely, if m_1 and m_2 are two square matrices of the same dimension, and given that "*map_thread on*" is a superposition with respect to function *on*, then we

Table 1: Computation of "*map_thread on*" for elements of the classes $\mathcal{C}_1$ and $\mathcal{C}_2$.

$\mathcal{C}_2$ $\mathcal{C}_1$	m_2	$m_2{}^{\rho}$	$m_2{}^{\rho^2}$	$m_2{}^{\rho^3}$
m_1	$f\ m_1\ m_2$	$f\ m_1\ m_2{}^{\rho}$	$f\ m_1\ m_2{}^{\rho^2}$	$f\ m_1\ m_2{}^{\rho^3}$
$m_1{}^{\rho}$	$f\ m_1{}^{\rho}\ m_2$	$f\ m_1{}^{\rho}\ m_2{}^{\rho}$	$f\ m_1{}^{\rho}\ m_2{}^{\rho^2}$	$f\ m_1{}^{\rho}\ m_2{}^{\rho^3}$
$m_1{}^{\rho^2}$	$f\ m_1{}^{\rho^2}\ m_2$	$f\ m_1{}^{\rho^2}\ m_2{}^{\rho}$	$f\ m_1{}^{\rho^2}\ m_2{}^{\rho^2}$	$f\ m_1{}^{\rho^2}\ m_2{}^{\rho^3}$
$m_1{}^{\rho^3}$	$f\ m_1{}^{\rho^3}\ m_2$	$f\ m_1{}^{\rho^3}\ m_2{}^{\rho}$	$f\ m_1{}^{\rho^3}\ m_2{}^{\rho^2}$	$f\ m_1{}^{\rho^3}\ m_2{}^{\rho^3}$

Note: Function f stands for "*map_thread on*".

Table 2: Results of applying (12) on the entries of Table 1.

$\mathcal{C}_2$ $\mathcal{C}_1$	m_2	$m_2{}^{\rho}$	$m_2{}^{\rho^2}$	$m_2{}^{\rho^3}$
m_1	$f\ m_1\ m_2$	$f\ m_1\ m_2{}^{\rho}$	$f\ m_1\ m_2{}^{\rho^2}$	$f\ m_1\ m_2{}^{\rho^3}$
$m_1{}^{\rho}$	$(f\ m_1\ m_2{}^{\rho^3})^{\rho}$	$(f\ m_1\ m_2)^{\rho}$	$(f\ m_1\ m_2{}^{\rho})^{\rho}$	$(f\ m_1\ m_2{}^{\rho^2})^{\rho}$
$m_1{}^{\rho^2}$	$(f\ m_1\ m_2{}^{\rho^6})^{\rho^2}$	$(f\ m_1\ m_2{}^{\rho^3})^{\rho^2}$	$(f\ m_1\ m_2)^{\rho^2}$	$(f\ m_1\ m_2{}^{\rho})^{\rho^2}$
$m_1{}^{\rho^3}$	$(f\ m_1\ m_2{}^{\rho^9})^{\rho^3}$	$(f\ m_1\ m_2{}^{\rho^6})^{\rho^3}$	$(f\ m_1\ m_2{}^{\rho^3})^{\rho^3}$	$(f\ m_1\ m_2)^{\rho^3}$

Note: Function f stands for "*map_thread on*".

have the following.

$$(\mathit{map\_thread\ on})\ m_1{}^{\rho^{n_1}}\ m_2{}^{\rho^{n_2}} =$$
$$\begin{cases} ((\mathit{map\_thread\ on})\ m_1\ m_2{}^{\rho^{n_2-n_1}})^{\rho^{n_1}}, & \text{if } n_1 \leq n_2 \\ ((\mathit{map\_thread\ on})\ m_1\ m_2{}^{\rho^{3\times(n_1-n_2)}})^{\rho^{n_1}}, & \text{otherwise} \end{cases} \tag{12}$$

Suppose that we want to check properties of all the possible applications of *puton* on elements of two equivalence classes $\mathcal{C}_1$ and $\mathcal{C}_2$. We need to compute "(*map_thread on*) m_1 m_2" for all $m_1 \in \mathcal{C}_1$ and $m_2 \in \mathcal{C}_2$ as shown in Table 1. By applying the result in (12), the entries of Table 1 are equivalent to those depicted in Table 2.

The 16 entries in Table 2 can be grouped into equivalence classes using relation *rot_rel*. Namely, we have the following 4 equivalence classes $[f\ m_1\ m_2]$, $[f\ m_1\ m_2{}^{\rho}]$, $[f\ m_1\ m_2{}^{\rho^2}]$ and $[f\ m_1\ m_2{}^{\rho^3}]$. Note that due to rule (4), the entries $(f\ m_1\ m_2{}^{\rho^6})^{\rho^2}$ and $(f\ m_1\ m_2{}^{\rho^9})^{\rho^3}$ in Table 2 are equal to $(f\ m_1\ m_2{}^{\rho^2})^{\rho^2}$ and $(f\ m_1\ m_2{}^{\rho})^{\rho^3}$, respectively, and therefore elements of the classes $[f\ m_1\ m_2{}^{\rho^2}]$ and $[f\ m_1\ m_2{}^{\rho}]$, respectively.

Now to finish computing *puton*, we need to apply function *merge*. To that end, we introduce (infix) predicate "*preserves*". A function g preserves an equivalence relation r if the image of all the elements in $[x]_r$ by g are in $[g\ x]_r$.

$$g\ \mathit{preserves}\ r \triangleq \forall x\, y.\ (x,y) \in r \rightarrow (g\,x, g\,y) \in r$$

We prove that *merge* preserves relation *rot_rel*.

lemma `merge preserves rot_rel`

Applying *merge* gives rise to the 4 equivalence classes $[\mathit{merge}\ (f\ m_1\ m_2)]$, $[\mathit{merge}\ (f\ m_1\ m_2{}^{\rho})]$, $[\mathit{merge}\ (f\ m_1\ m_2{}^{\rho^2})]$ and $[\mathit{merge}\ (f\ m_1\ m_2{}^{\rho^3})]$.

Next, we use (infix) predicate "*respects*" defined in the Equiv_Relations theory (Paulson, 2006). A function g respects an equivalence relation r if g returns the same value for all the elements of an equivalence class generated by r.

$$g\ \mathit{respects}\ r \triangleq \forall x\, y.\ (x,y) \in r \rightarrow g\,x = g\,y$$

The properties that we want to check, namely *successful*, *effective* and *valid* can be regarded as boolean functions. We prove that they respect *rot_rel*. If a property P($\in\{$*successful, effective, valid*$\}$) holds for one element of a class then it holds for all the remaining elements, otherwise it gives False for all the remaining elements.

lemma `successful respects rot_rel`
lemma `effective respects rot_rel`
lemma `valid respects rot_rel`

In order to check properties of *puton* applied to all the 16 pairwise combinations of two equivalence classes $\mathcal{C}_1$ and $\mathcal{C}_2$, it is sufficient to check them for 4 *puton* computations that are the *putons* applied to a representative of $\mathcal{C}_1$ and all the elements of $\mathcal{C}_2$.

8 CONCLUSION

We covered the formalization of superposition of rectangular units in Isabelle/HOL. A unit is given a qualitative matrix representation. The superposition of two units is regarded as superposition function applied to their respective matrix representations. Function *puton* is defined to refine the result of superposition by merging black regions. We defined properties of success, effectiveness and validity, and explained our proof strategy to tackle the numerous cases of superposition. We can think of two directions for future work of this research.

First, we plan to relax all the assumptions on the unit structure that are enumerated in Sect. 5.1. We would like to include more complex unit structures such as the ones depicted in Fig. 9. To that end, we need to extend the set of direction relations to express the locations of regions that are not directly connected. The refinement to higher granularity level allows representing any unit structure. We also plan to formalize the generalization in Isabelle/HOL.

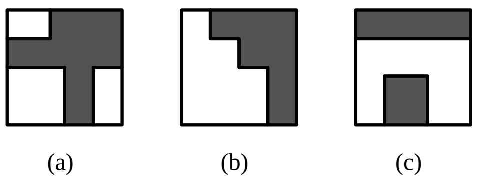


Figure 9: Unit structures whose qualitative representations require extending direction relations.

Second, QSR theories, such as direction relations, rely on composition tables that are relevant for deciding whether QSR constraints are consistent (Renz, 2002; Frank, 1991). In order to construct the composition tables, the work presented in (Randell et al., 1992a) uses OTTER proof system for first-order logic to assist pen-and-paper proofs. The authors have enumerated the challenge of such proofs because of the number of the entries of a composition table. We plan to investigate the proof of the correctness of the composition tables using proof assistant.

ACKNOWLEDGEMENTS

This work is supported by JSPS KAKENHI Grant No. 25330274.

REFERENCES

Avigad, J. and Harrison, J. (2014). Formally Verified Mathematics. *Communications of the ACM*, 57(4):66–75.

Bennett, B. (1996). The Application of Qualitative Spatial Reasoning to GIS. In *Proceedings of The 1st International Conference on GeoComputation*, volume I, pages 44–47.

Chen, T., Schneider, M., Viswanathan, G., and Yuan, W. (2010). The Objects Interaction Matrix for Modeling Cardinal Directions in Spatial Databases. In *Database Systems for Advanced Applications*, volume 5981 of *LNCS*, pages 218–232. Springer Berlin Heidelberg.

Clementini, E., Felice, P. D., and Hernándes, D. (1997). Qualitative Representation of Positional Information. *Artificial Intelligence*, 95(2):317 – 356.

Frank, A. U. (1991). Qualitative Spatial Reasoning about Cardinal Directions. In *Proceedings of the International Symposium on Computer-Assisted Cartography*, pages 148–167. ACSM-ASPRS.

Konishi, T. and Takahashi, K. (2012). Superposition of Rectangles with Visibility Requirement: A Qualitative Approach. *International Journal On Advances in Software*, 4(4):422–433.

Liu, J. and Daneshmend, L. (2004). *Spatial Reasoning and Planning: Geometry, Mechanisms, and Motion*. Advanced Information Processing. Springer.

Nipkow, T., Paulson, L. C., and Wenzel, M. (2002). *Isabelle/HOL: A Proof Assistant for Higher-Order Logic*, volume 2283 of *LNCS Tutorial*. Springer.

Paulson, L. C. (2006). Defining Functions on Equivalence Classes. *ACM Transactions on Computational Logic*, 7(4):658–675.

Randell, D. A., Cohn, A. G., and Cui, Z. (1992a). Computing Transitivity Tables: A Challenge for Automated Theorem Provers. In *Proceedings of Automated Deduction (CADE-11)*, volume 607 of *LNCS*, pages 786–790. Springer.

Randell, D. A., Cui, Z., and Cohn, A. G. (1992b). A Spatial Logic based on Regions and Connection. In *Proceedings of the 3rd International Conference on Knowledge Representation and Reasoning*, pages 165–176.

Renz, J. (2002). *Qualitative Spatial Reasoning with Topological Information*, volume 2293 of *LNCS*. Springer.

Sternagel, C. and Thiemann, R. (2010). Executable Matrix Operations on Matrices of Arbitrary Dimensions. In *The Archive of Formal Proofs*. http://afp.sf.net/entries/Matrix.shtml.

Wenzel, M. (1999). Isar - A Generic Interpretative Approach to Readable Formal Proof Documents. In *Theorem Proving in Higher Order Logics*, volume 1690 of *LNCS*, pages 167–183. Springer.