Formalization of the Qualitative Spatial Reasoning of Superposition of Rectangles in Proof Assistant

Fadoua Ghourabi and Kazuko Takahashi

In qualitative spatial reasoning, objects are treated qualitatively, i.e. without performing numerical computation. In this research, we qualitatively reason about superposition of rectangles with visibility requirements. A method is proposed to automatically superpose rectangles while respecting user's specification of visibility criteria. We use proof assistant Isabelle/HOL to strengthen the proposed method. A rectangle is a resizable object that is divided into visible regions and non-visible regions. The superposition of two rectangles is computed from the superposition of their regions. Rectangles are categorized into groups that are formalized as equivalence classes. Using Isabelle/HOL, we prove properties of effectiveness and validity about superposition on the equivalence classes.

1 Introduction

In the 20th century, questions of mathematical rigour were throughly examined by logicians. In particular, the foundation of mathematical proof has been re-examined. Pen-and-paper proofs are often concise and intuitive but prone to present logical flaws, skip important steps or omit important cases. We, therefore, became discontented to work with a mere definition of proof. Proof assistants are increasingly used to provide mathematical proofs where each step is transparent and, thus, the involved logical reasoning can be checked.

The purpose of this research is to extend the use of proof assistants to provide formal proofs in qualitative spatial reasoning (QSR). In QSR, objects are treated qualitatively, i.e. without performing numerical computation. QSR has a wide range of applications in geographical information systems, spatial databases, etc. However, QSR methods rely on pen-and-paper proofs. As far as we know, there is no formal presentation of QSR methods using proof assistants. Rigorous formalization and formal proofs are required when we tackle some of QSR problems, e.g. correctness of QSR programs.

This research is an ongoing work based on the QSR method presented in [1]. The method deals with the problem of superposing rectangles, called *units*, with requirements of visibility. In other words, rectangles are spatial objects that include parts that should be visible. The superposition should keep those parts visible. Possible application of this QSR theoretical study is the superposition of windows of computer software with better end-user experience of visibility.

In this research, we aim at improving the theory of superposition of rectangles by proposing rigorous formalization using proof assistant. It aims at checking the results which were established based on pen-and-paper proofs and revising them when necessary. We do not create new QSR method but rather we strengthen the formalization of an exist-

Formalization of the Qualitative Spatial Reasoning of Superposition of Rectangles in Proof Assistant

Fadoua Ghourabi and Kazuko Takahashi, 関西学院大学理工学部, School of Science and Technology, Kwansei Gakuin University.

ing one using proof assistant. We choose to work with Isabelle/HOL proof assistant [2] to formalize the superposition of rectangles. In the rest of the paper, we use notations and symbols as defined by the syntax of Isabelle/HOL. We also use Isar [3] framework in Isabelle/HOL. Isar provides relatively structured proofs where it is possible to follow the reasoning involved in the proof. It is possible to alternate Isar proof structure with proofs adopted in classical Isabelle/HOL.

The rest of this paper is organized as follows. In Sect. 2, we present a formalization of unit. In Sect. 3, we define valid unit. In Sect. 4, we explain the superposition of rectangles. We prove properties about the results of superposition in Sect. 5 and we discuss the revisions that we undertook in Sect. 6. Finally, in Sect. 7, we summarize and point out directions of further research.

2 Representation of Unit in Isabelle/HOL

The spatial object that we investigate is a rectangle called unit. Some parts of a unit are required to be visible. We assume that the visible parts are rectangles and model them as white rectangular plates. The parts that can be hidden are regarded as black rectangular plates. We also assume that the size of a unit can change in a similar fashion to the way software windows are shrunk or expanded.

2.1 Black Plates

In this paper, we restrict ourselves with units with at most two black plates. Let U be a rectangular unit of size $l \times h$.

- If U has one black plate p, then the size of p is either $l \times v$ or $u \times h$, where $0 \leq u \leq l$ and $0 \leq v \leq h$. Each of the units in Fig. 3 has one black plate that is spread along the length/height of the unit.
- If U has two black plates p_1 and p_2, then necessary at least one of them is of size $u \times h$ or $l \times v$, where $0 \leq u \leq l$ and $0 \leq v \leq h$. Furthermore, p_1 and p_2 must be overlapping. The overlapping part forms a rectangular shared area denoted by $p_1 \sqcap p_2$ (see the units in Fig. 2(c)).

From the white and black plates, we compute regions.

2.2 Regions of Unit

A region can be black, white, grey, or undefined. We denote by b, w, g and $None$ the black, white, grey and undefined regions, respectively. We work with units that are located on the 2D plane. We call the 2D plane *the grey region*. b and g regions can be hidden while w regions should be visible. In Isabelle/HOL, we define the datatype region as follows.

```
datatype region = None | b | w | g
```

For a unit U of size $l \times h$, the regions of U are determined as follows.

(a) Any region of U (black or white) is of size $u \times v$, where $0 \leq u \leq l$ and $0 \leq v \leq h$.

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

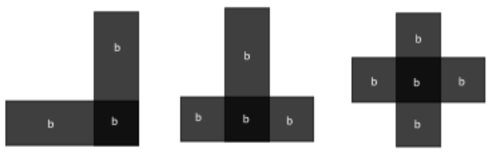
(c) If p is the only black plate of U, then p is also the only b region of U.

(d) 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 (see Fig. 1). Note that the area $p_1 \sqcap p_2$ is one of the b regions generated by p_1 and p_2.

We establish a relation of adjacency on the regions of U.

Definition Two regions r_i and r_j of U are adjacent, denoted by $r_i \sim r_j$, if they share an edge.

The values of l and h are variable. As a consequence, the sizes of the regions of U change while preserving their positions in U as well as the relation $\sim$ and operation $\sqcap$.



(a) (b) (c)

図 1 3, 4 and 5 *b* regions generated by two black plates

Next, we define the core of a unit.

Definition The core region of unit U, denoted by C_U, is a b region defined as follows:

- If U has only one b region, then $C_U = p$.
- If U has two distinct black plates p_1 and p_2, then the core of U is the shared area between p_1 and p_2, i.e. $C_U = p_1 \sqcap p_2$.
- U has no b region, then C_U is regarded as a tiny b region.

Since a unit has at most two black plates as established in the assumptions, then a unit has one and only one core. The unit of Fig. 2(c) has two black plates that share a rectangular area highlighted in darker black. The shared area represents the core.

We represent a unit U as sequence of 4 regions that are adjacent to C_U. We write $U = \langle r_1, r_2, r_3, r_4 \rangle$, where $C_U \sim r_i$, $1 \leq i \leq 4$. Regions r_1, r_2, r_3, and r_4 are adjacent to C_U in a clockwise order starting from the upper adjacent region. For instance, the representations of the units in Figs. 2(b), 3(a) and 6(a) are $\langle w, w, w, w \rangle$, $\langle w, g, g, g \rangle$ and $\langle b, g, w, b \rangle$, respectively.

```
datatype
representation = Rep region region region region
```

We define functions `up`, `rt`, `dn` and `lt` that return the upper, right, below and left adjacent regions, respectively. We show the definition of `up` in Isabelle/HOL. The rest are defined similarly.

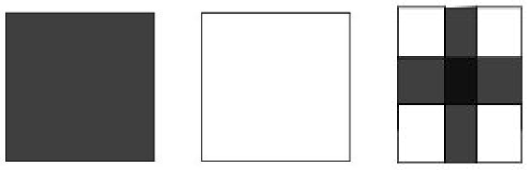
```
fun up ::" representation ⇒ region"
where "up (Rep r1 r2 r3 r4) = r1"
```

Function regions returns the set of all the 4 regions of a representation of a unit.

```
fun regions::"representation ⇒region set"
where "regions U ≡ {up U, rt U, dn U, lt U}"
```

3 Valid Unit

When working with computer, we often rearrange software windows by dragging, resizing, superposing, etc. until obtaining better visibility. A superfluous information in a window, e.g. advertisement column in a webpage, can be hidden and superposed by an important information in another window. The problem of arranging software windows is regarded as the problem of superposing rectangles while keeping all important information visible. We use unit to model a software window and regions to specify which window parts are important, and thus, should be visible. We examine how applications divide the windows into regions and, analogously, we divide units into regions. We restrict ourselves to unit structures that are depicted in Fig. 2 ∼ 9.



(a) B unit (b) W unit (c) PLUS unit

図 2 B, W and PLUS units

We have 3 cases of fitting black plates into a unit while respecting our assumptions in Sect. 2.

- No black plate (W type unit in Fig. 2(b))
- One black plate (B, I1 and I2 type units in

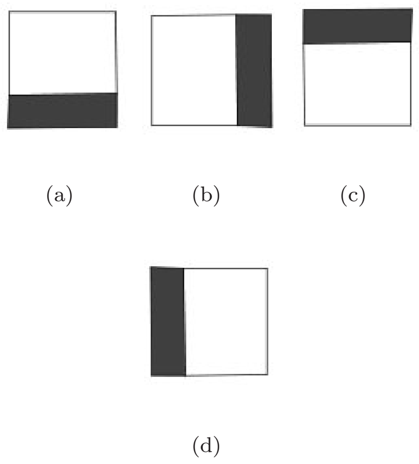


図 3 **I1 type units**

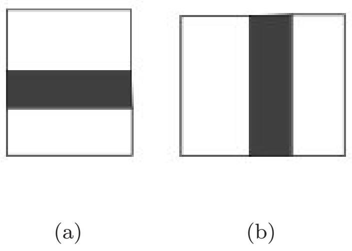


図 4 **I2 type units**

Figs. 2(a), 3 and 4, respectively)

- Two black plates (L1, L2, L3, T1, T2, PLUS type units in Figs. 5, 6, 7, 8, 9 and 2(c), respectively)

For each unit type in Fig. 2 ∼ 9, we choose a representative unit as follows.

```
abbreviation "B ≡ Rep g g g g"
abbreviation "W ≡ Rep w w w w"
abbreviation "I1 ≡ Rep w g g g"
abbreviation "I2 ≡ Rep w g w g"
abbreviation "L1 ≡ Rep b g g b"
abbreviation "L2 ≡ Rep b g w b"
abbreviation "L3 ≡ Rep b w g b"
abbreviation "T1 ≡ Rep b b g b"
abbreviation "T2 ≡ Rep b b w b"
abbreviation "PLUS ≡ Rep b b b b"
```

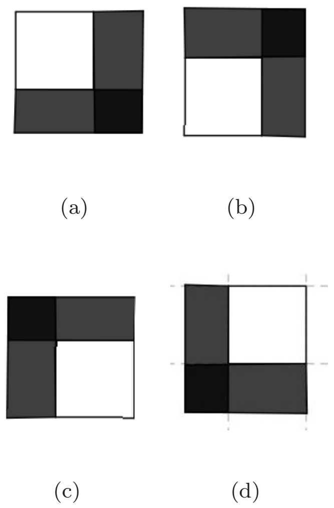


図 5 **L1 type units**

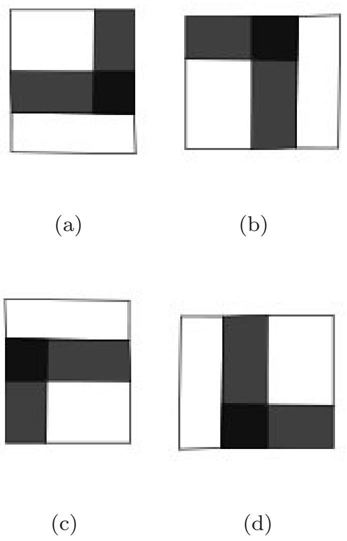


図 6 **L2 type units**

For instance, representative unit of I1 type is chosen to be `Rep w g g g` which correspond to the unit in Fig. 3(a). Furthermore, we introduce the set of representative units $T0 = \{B, W, I1, I2, L1, L2, L3, T1, T2, PLUS\}$.

The units depicted in Fig. 2 ∼ 9 are called valid units that we will define more accurately in the fol-

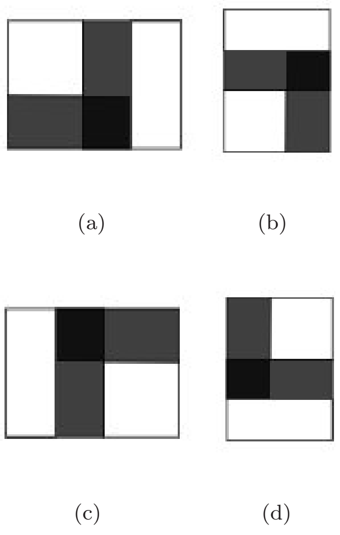


図 7 L3 type units

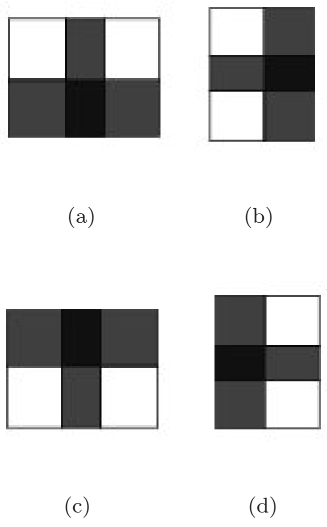


図 8 T1 type units

lowing sections.

3.1 Rotation

A unit rotation is a $\frac{\pi}{2}$ counter-clockwise rotation. A rotation of a unit $U = \langle r_1, r_2, r_3, r_4 \rangle$ is a unit $U' = \langle r_2, r_3, r_4, r_1 \rangle$. For instance, $U_1 = \langle b, g, g, b \rangle$

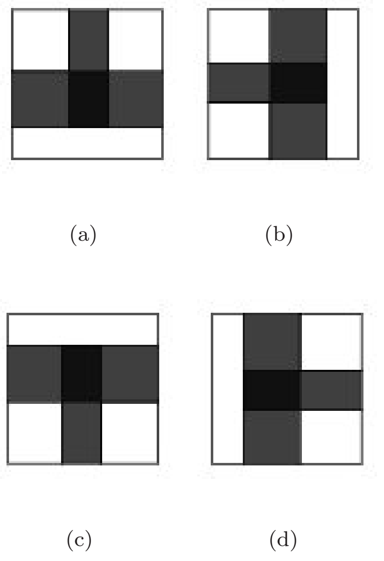


図 9 T2 type units

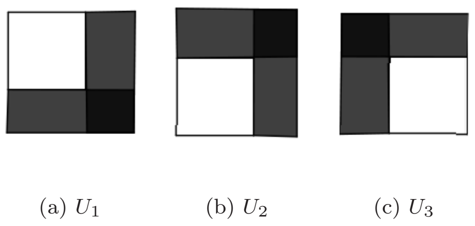


図 10 Rotations of unit $U_1 = \langle b, g, g, b \rangle$

in Fig. 10(a) is rotated to obtain $U_2 = \langle g, g, b, b \rangle$ in Fig. 10(b). We introduce function `rotate` to compute the rotation of a representation.

```
definition rotate::
"representation ⇒ representation"
where
"rotate U = Rep (rt U) (dn U) (lt U) (up U)"
```

We can carry out counter-clockwise rotation, successively. We define the recursive function `rotate_succ` that takes natural number n and return a function of type `representation ⇒ representation`. The call `rotate_succ` n gives rise the identity function if $n = 0$, otherwise n compositions of function `rotate`.

```
primrec rotate_succ::
"nat ⇒  (representation ⇒  representation)"
where
"rotate_succ 0 = id"|
"rotate_succ (Suc n) = rotate o rotate_succ n"
```

In Fig. 10, we can easily check that U_2 = `rotate_succ` 1 U_1 and U_3 = `rotate_succ` 2 U_1 = `rotate` ∘ `rotate` U_1.

We prove few lemmas about successive rotation. These are intermediate lemmas that are used as elementary steps in more elaborate proofs. We briefly explain some of these lemmas.

If we rotate a unit U successively 4 times then we obtain U. We prove the following lemma to handle the cases where we apply $n \geq 4$ successive rotations.

```
lemma rotate_succ_n:
"⋀U::representation. ⋀n::nat.
rotate_succ n U = rotate_succ (n mod 4) U"
```

We also prove that function `rotate` is bijective and invertible. The inverse function $\texttt{rotate}^{-1}$ = `rotate_succ` 3. Based on this result, we prove the following lemma about successive rotation.

```
lemma rotate_succ_inverse:
"⋀U1 U2:: representation.⋀n::nat.
(rotate_succ n U1 = U2) ⟹
             (U1= rotate_succ (3*n) U2)"
```

3.2 Equivalence Classes

To compute all the units in Fig. 2 ∼ 9, we use the formalization of equivalence classes in Isabelle/HOL [4]. We begin by defining a relation `unitrel`. Let U_1 and U_2 be two units. We have $(U_1, U_2) \in$ `unitrel` if U_2 is obtained by successively rotating U_1.

```
definition unitrel::
"(representation ×  representation) set"
where
"unitrel ≡ {(U1, U2)| U1 U2. same_type U1 U2}"
```

Given two units U_1 and U_2, the predicate `same_type` returns `True` if U_1 and U_2 are of the same type. In other words, it checks whether U_2 is obtained by successive rotation of U_1. We give the definition of `same_type` in the following.

```
definition same_type::
"representation ⇒  representation ⇒ bool"
where
"same_type U1 U2 ≡
    ∃n::nat. U2 = rotate_succ n U1"
```

We prove that `unitrel` is reflexive, transitive and symmetric, and hence we deduce that `unitrel` is an equivalence relation.

```
lemma "equiv UNIV unitrel"
```

Predicate `equiv` verifies whether `unitrel` is an equivalence relation on the set `UNIV`, which is Isabelle/HOL polymorphic constant that denotes the universal set. In our case the elements of `UNIV` are of type `representation` × `representation`.

The Isabelle/HOL expression `unitrel``{`$U$`}` denotes the equivalence class generated by unit $U$, i.e. `unitrel``{`$U$`}={`$R$ | ($U$, $R$) ∈ `unitrel}`. A unit U is valid if and only if U is an element of the union of the equivalence classes generated by the representative units in T0. We have the following definition of valid units.

```
definition unit :: "representation set"
where
"unit ≡  (⋃y∈T0. unitrel``{y})"
```

The advantage of formalizing valid units using Isabelle/HOL definition of equivalence classes appears when we carry out proofs of properties about superposition of units, which will be explained further in Sect. 5.

4 Superposition of Units

We call *puton* the operation that performs superposition of units.

4.1 Puton Operation

First, we define a partial function `on` that computes superposition of regions.

```
fun on:: " region ⇒  region ⇒  region"
where
"on b b = b" |
"on b w = w" |
"on w b = None" |
"on w w = None" |
"on g g = g" |
"on g x = x" |
"on x g = x" |
"on None _ = None"|
"on _ None = None"
```

Now, superposing two units U_1 and U_2 by putting U_2 on U_1 means that we put C_{U_2} on C_{U_1}. The core of the resultant unit U_3 is $C_{U_3} = C_{U_2} = C_{U_1}$. Units U_1 and U_2 can be resized if necessary so that the cores have the same size. As explained in Sect. 2, the representation of U_3 is a sequence of the regions that are adjacent to C_{U_3}. Function `on_rep` computes the resultant unit by applying function `on` on the regions of U_1 and U_2.

```
definition on_rep::"representation ⇒ representation
⇒ representation"
where
"on_rep U1 U2 ≡
(if U1=W then Undefined else
Rep (on (up U1) (up U2)) (on (rt U1) (rt U2)) (on
(dn U1) (dn U2)) (on (lt U1) (lt U2)))"
```

Note that it is impossible to put any unit on W type unit. Therefore, in that case, `on_rep` gives rise to the `Undefined` unit $\langle None, None, None, None\rangle$. We say that `on_rep` fails when the computed unit has at least one $None$ region. We define predicate `on_rep_fail` to check wether `on` call in `on_rep` returns $None$.

A function `merge` was introduced to obtain better result of superposition [1]. An example of utilization of `merge` is depicted in Fig. 11. When we put U_2 of type I1 on U_1 of type L3, part of the non-core b region of U_1 is merged with C_{U_2} to form one core region. We obtain a unit of type I2. Function `merge` is defined in Isabelle/HOL according to its description in [1].

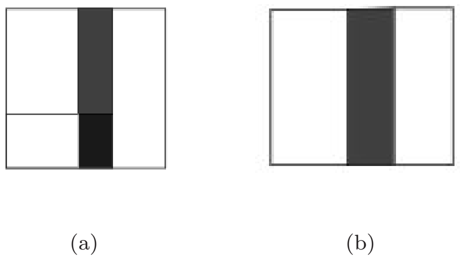


図 11 Superposition and merge of I1 type unit and L3 type unit

Finally, we define the function `puton` that calls `merge` ∘ `on_rep`.

```
definition puton::"representation ⇒ representation
⇒ representation"
where
"puton R1 R2 ≡ (if ¬(on_rep_fail R1 R2) then (merge
(on_rep R1 R2)) else Undefined)"
```

4.2 Properties

When superposing two units, we check properties of effectiveness and validity about the result of `puton`. Function `puton` gives rise to an effective unit if it has at most one b region. The result of `puton` is valid if the computed unit is valid.

```
definition effective::"representation ⇒ bool"
where
"effective R ≡ ¬ (b ∈ (regions R ))"

definition valid::"representation  ⇒ bool"
where "valid R ≡ R ∈ unit"
```

The effectiveness and validity of the result of operation `puton` are discussed in [1]. The paper presents several general statements on these properties. Our objective is to check the correctness of these statements by formally proving/disproving them using Isabelle/HOL.

5 Theorems about Effectiveness and Validity

In Sect. 3, we explained that the units of interest are grouped into equivalence classes. Isabelle/HOL provides a useful predicate `respect`. Let U be a unit and f be a function. f `respect` `unitrel`‘‘$\{U\}$ is true if $f(U_i) = f(U_j)$ for any $\{U_i, U_j\} \subseteq$ `unitrel`‘‘U. Therefore, from a property that respects the equivalence classes, we can build a proof that check properties for a representative of the equivalence class. This simplifies the proofs as we don't have to enumerate all the combinations and check the properties for all the cases, i.e. all possible superposition of units.

We list the theorems that we proved. The detailed proofs are available online [5].

5. 1 Superposition on B and W Type

For clarity, we introduce predicate `puton_sucess` that simply gives the negation of `on_rep_fail`.

Theorem 5.1. *It is impossible to place any unit on the W type unit in Fig. 2(b).*

$$\forall R : representation. \quad R \in unit \Rightarrow \neg\ puton\_sucess\ W\ R \tag{1}$$

Theorem 5.2. *Superposition of any unit on B type (see Fig. 2(a)) is always successful.*

$$\forall R : representation. \quad R \in unit \Rightarrow puton\_success\ B\ R \tag{2}$$

Theorem 5.3. *Superposition of any unit on B type unit is always valid.*

$$\forall R : representation.\ R \in unit \Rightarrow valid\ B\ R \tag{3}$$

We call *straight-plate-unit* a unit of I1 or I2 type depicted in Figs. 3 and 4. *Cross-plate-unit* means units of types L1, L2, L3, T1, T2 and PLUS in Figs. 5, 6, 7, 8, 9 and 2(c). To simplify our formulas in the following theorems, we define the sets $SP = unitrel``\{I1\} \cup unitrel``\{I2\}$ and $CP = unitrel``\{L1\} \cup unitrel``\{L2\} \cup unitrel``\{L3\} \cup unitrel``\{T1\} \cup unitrel``\{T2\} \cup unitrel``\{PLUS\}$. We prove the following.

Theorem 5.4. *Superposition of a unit R on B type unit is effective if and only if R is either straight-plate-unit or B type unit or W type unit.*

$$\forall R : representation. \quad R \in SP \cup unitrel``B \cup unitrel``W \equiv effective\ (puton\ B\ R) \tag{4}$$

5. 2 Superposition on Straight-plate-units

Theorem 5.5. *When successful, the superposition of straight-plate-unit on a straight-plate-unit is always effective.*

$$\forall R1\ R2 : representation. \quad \{R1, R2\} \subseteq SP \wedge puton\_success\ R1\ R2 \Rightarrow effective\ (puton\ R1\ R2) \tag{5}$$

Theorem 5.6. *When successful, the superposition of straight-plate-unit on straight-plate-unit is not always valid.*

$$\exists R1\ R2 : representation. \quad \{R1, R2\} \subseteq SP \wedge puton\_success\ R1\ R2 \wedge \neg\ valid\ (puton\ R1\ R2) \tag{6}$$

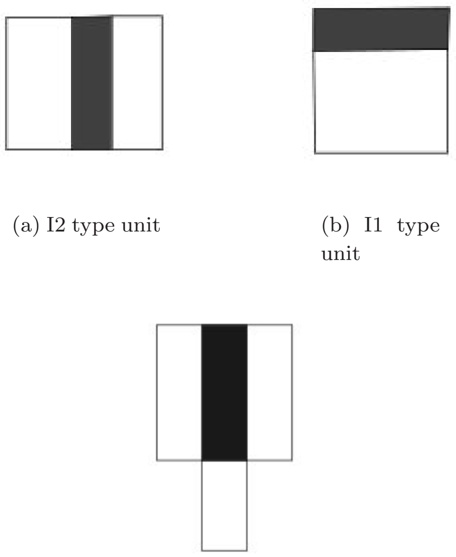
Example Using command `nitpick` of Isabelle/HOL, we generate a counter example for the formula $\forall R1\ R2 : representation.\ \{R1, R2\} \subseteq SP \wedge puton\_success\ R1\ R2 \Rightarrow valid\ (puton\ R1\ R2)$, which is a solution for formula (6). Suppose that $R1 = \langle g, w, g, w\rangle$ of type I2 (depicted in Fig. 12(a)) and $R2 = \langle g, g, w, g\rangle$ of type I1 (depicted in Fig. 12(b)). The unit in Fig. 12(b) is resized so that its core has the same size as the core in Fig. 12(a). Fig. 12(c) shows the result of putting $R2$ on $R1$.

5. 3 Superposition on Cross-plate-unit

Theorem 5.7. *When successful, the superposition of straight-plate-unit on a cross-plate-unit is not always effective.*

$$\exists R1\ R2 : representation. \quad R1 \in CP\ \wedge\ R2 \in SP \wedge puton\_success\ R1\ R2 \wedge \neg\ effective\ (puton\ R1\ R2) \tag{7}$$

Theorem 5.8. *When successful, the superposition*



(a) I2 type unit (b) I1 type unit (c) Invalid unit

図 12 Invalid unit obtained by superposing two straight-plate-units

of straight-plate-unit on a cross-plate-unit is not always valid.

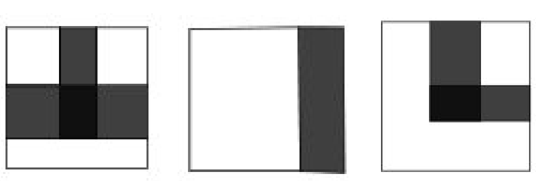
$$\begin{aligned}&\exists R1\ R2 : representation.\\ &R1 \in CP \ \wedge\ R2 \in SP \wedge puton\_success\ R1\ R2 \ \wedge\\ &\qquad \neg\ valid\ (puton\ R1\ R2)\end{aligned} \tag{8}$$

Example Figure 13 shows the result of putting $R2 = \langle g, g, g, w\rangle$ of type I1 on $R1 = \langle b, b, w, b\rangle$ of type T2. The puton operation is successful but neither effective nor valid.

6 Revisions

Due to the usage of proof assistant, we revised the original QSR method presented in [1]. The revisions that we undertook are summarized in the following.

[Reformulation] Only rotation is used to compute the valid unit. Reflection was proposed in



(a) T2 type unit (b) I1 type unit (c) Result of superposition

図 13 Ineffectiveness and invalidity of $R2 = \langle g, g, g, w\rangle$ on $R1 = \langle b, b, w, b\rangle$

the original paper, but we choose to replace it by successive rotations. This modification allows simpler definition based on equivalence classes. Furthermore, in the original paper, validity property is judged by checking the values of the regions of before and after applying puton operation. The idea of validity property is to obtain a valid unit by applying puton operation. We therefore simply check that the result of puton is in the union of equivalence classes.

[Precision] The Theorems presented in Sect. 5 are formalization of statements in the original paper. We provide more precise declaration of the properties to be proved. For instance, the original paper doesn't explicitly specify that properties of effectiveness and validity should be checked on successful puton operations. Unlike the prose, the formalization in Isabelle/HOL must state this condition.

[Correction] Property of validity as defined in [1] is wrong for some cases that were overlooked. The new definition of validity tackles these cases correctly. Furthermore, the statement of Theorem 5.4 omits B and W type units. These units should be included otherwise the theorem can not be proved.

7 Conclusion

In this paper, we presented the highlight of formalization of superposition of rectangles in Isabelle/HOL. We used the formalization to prove properties of effectiveness and validity about the result of puton operation. The formalization in proof assistant did add rigour by revealing omissions or imprecise statements in some of the definitions and proofs in the original work. We believe that a proposed qualitative spatial reasoning method must be accompanied by formalization in proof assistant to underline its accuracy and correctness.

As this is an ongoing work, we plan to extend the formalization and add operation of embedding, which is another definition of superposition of unit. We plan to compare the results of puton and embedding operations. We also plan to refine the theorems in Sect. 5 and add general conditions to specify the cases when puton is effective or valid.

8 Acknowledgments

This work is supported by JSPS KAKENHI Grant No. 25330274. The first author is supported by a postdoctoral fellowship at Kwansei Gakuin University.

参 考 文 献

[1] T. Konishi and K. Takahashi. Superposition of Rectangles with Visibility Requirement: A Qualitative Approach. *International Journal on Advances in Software*, 4(3 & 4):422–433, 2011.

[2] T. Nipkow, L. C. Paulson, and M. Wenzel. *Isabelle/HOL — A Proof Assistant for Higher-Order Logic*, volume 2283 of *LNCS*. Springer, 2002.

[3] M. Wenzel. The Isabelle/Isar Reference Manual, 2002. http://www.cl.cam.ac.uk/research/hvg/Isabelle/dist/Isabelle2013/doc/isar-ref.pdf.

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

[5] F. Ghourabi. Qualitative Spatial Reasoning of Superposition of Rectangles - Isabelle Proofs, 2013. http://www.i-eos.org/Members/fadoua-ghourabi.