定性空間推論:PLCA 表現と推論機構

Qualitative spatial reasoning:PLCA expression and reasoning system

住友 孝郎 † 高橋 和子 ††
Takao Sumitomo and Kauzko Takahashi

† 関西学院大学理工学研究科
x Graduate School of Science and Technology, Kwansei Gakuin University
†† 関西学院大学理工学部
School of Science & Technology, Kwansei Gakuin University

定性空間推論の枠組みとなる PLCA 表現とその上での推論システムについて述べる．PLCA 表現では 2 次元平面上の図形に対し, その形に無関係にそれを構成する領域同士の接し方を漏らさず表現することができる．この表現は簡潔なものであり, 既存のデータベースやシステムと容易につなぐことができる．また,PLCA 表現は各領域のもつ性質を付加して拡張することにより, より豊かな表現や推論が可能になる．ここでは拡張例として xPLCA 表現を導入し, 与えられた図形データを xPLCA 表現に変換するシステムを示す．また, このシステムに Prolog を組み込むことで, Prolog の推論機構を利用したさまざまな推論が行えることを示す．

1 はじめに

図形を計算機上で表現するにはテキストデータに比べて多くのメモリが必要であり, 処理にも多くの時間が必要である．このため図形を効率よく扱える仕組みが求められる．一般に図形データはラスタデータ方式またはベクタデータ方式が使用される．これらはいずれも図形や画像データを数値データとして格納し，図形上の操作を数値処理として実現しているためデータ量や計算量が非常に多い．しかし応用分野によっては描かれているオブジェクト同士のつながりやオブジェクト数さえわかれば十分であることが多い．

定性空間推論 [1][3][4][6] [8][13] は，画像や図形などの空間データを定性的にとらえユーザの目的に必要な性質のみを取り出し記号表現によって記述しようとするものである．たとえば,「大阪府と兵庫県は接している」「部屋の中に朝日のあたる場所がある」のような記述ができる．また，細胞 A, B が連続的に変化していく様子を,「A, B が離れている，A, B が接触している，A が B をとりこむ」などのようにオブジェクト同士の位置関係を離散的にとらえる定性的なシミュレーション [5][11][10][2] もターゲットとしている．どのような情報が必要でどのレベルで抽象化を行なうかは目的に依存する．

我々は 2 次元平面における図形を接し方に着目する．しかし図形の形状については考慮しない．形状について考慮しない接し方を取り扱った定性空間推論の手法として RCC(Region Connection Calculus)[12] や 9-intersection model[7] などがある．これらは接し方についての区別の方法を与える．しかし接し方そのものは記述しない．我々は新しい枠組み DLCS 表現 [9][14][15] 改め PLCA 表現を提案した．PLCA 表現は点，線，閉路，範囲という単純なオブジェクトを基本としてこれらを構成要素としたつながり方で図形の接し方そのものを表現する．

PLCA 表現は最低限の枠組みだが, 情報の付加により応用範囲が広がる．本発表では PLCA に情報を付加した xPLCA と Prolog との併用による応用例を示す．

本発表の構成は以下の通りである．
まず，第 2 章で PLCA 表現の枠組を述べる．次に，第 3 章で PLCA 表現の応用例を示す．次に，第 4 章で二重連結辺リストや RCC との関係を調べ PLCA 表現の応用の可能性を考察する．最後に第 5 章で結論を述べる．

2 PLCA 表現

PLCA 表現では，複数の領域がどのように接しているかを対象として取り扱う．この表現は対象となる物を，点 (Point) の集合，それらを結ぶ線 (Line) の集合，それらをつないだ閉路 (Circuit)，それらによって囲まれる範囲 (Area)，の 4 つに分けて 2 次元平面上の図形を表現する．PLCA 表現は図形表現に対して点接・線接という意味で一意的である．

2.1 クラスの定義

Point

Point はプリミティブとして存在するクラスと定義する．直観的にはまさに「点」のことである．

Line

Line は以下の条件を満たすクラスと定義する．直観的には点と点を結んだ「線」のことであり，2 つの線が交差することは許さない．
(条件) 任意のインスタンス l が

$$\begin{aligned} &l.points = [p_1, p_2] \\ &p_1, p_2 \text{ are Point} \end{aligned} \tag{1}$$

のように *Point* の長さ 2 の列である *points* を持つ．*Point* の順番には方向性があり，順方向を上添え字の $+$ で，逆方向を上添え字の $-$ で表す．
たとえば,$l.points = [p_1, p_2]$ ならば

$$\begin{aligned} l^+.points &= [p_1, p_2] \\ l^-.points &= [p_2, p_1] \end{aligned}$$

とする．ただし，l_i^+ と l_i^- のいずれかを指すときは l_i^* と表す．

Line について,

$$\neg\forall l_1, l_2 (l_1.points = l_2.points \Rightarrow l_1 = l_2)$$

であることに注意すること．これは, 両端が一致する線 (曲線) は複数存在するためである．*Line* を集合として取り扱うとこの区別が困難になるため, このように記述する．

Circuit

Circuit は以下の条件を満たすクラスと定義する．直観的には線を結んで環状になった一周できる閉路のことである．
(条件) 任意のインスタンス c が

$$\begin{aligned} &c.lines = [l_0^*, l_1^*, \ldots, l_{n-1}^*] \\ &\quad l_0, l_1, \ldots, l_{n-1} \text{ are Line} \\ &\quad |c.lines| \geq 1 \\ &l_i.points[1] = l_{i+1}.points[0] (0 \leq i < n) \end{aligned} \tag{2}$$

のように *Line* の長さ 1 以上の循環な列である *lines* を持つ．

接し方を表す述語

Circuit のインスタンス c_1, c_2 に対して，ここで以下のような述語 lc(Line Connect),pc(Point Connect) を定義する．直観的には $lc(c_1, c_2)$ は 2 つの閉路の共有線を持つとき，$pc(c_1, c_2)$ は 2 つの閉路の共有点を持つときそれぞれ真になる．

$$lc(c_1, c_2) = \begin{cases} true & if\ \exists l \begin{cases} l \in c_1.lines \\ \wedge\ l \in c_2.lines \end{cases} \\ false & otherwise \end{cases}$$

$$pc(c_1, c_2) = \begin{cases} true & if\ \exists p \begin{cases} p \in l_1.points \\ \wedge\ p \in l_2.points \\ \wedge\ l_1 \in c_1.lines \\ \wedge\ l_2 \in c_2.lines \end{cases} \\ false & otherwise \end{cases}$$

Area

Area [1]は以下の条件を満たすクラスと定義する．直観的には一つながりの範囲のことであり，2 つの範囲が交差することは許さない．
(条件) 任意のインスタンス a が

$$\begin{aligned} &a.circuits = \{c_1, c_2, \ldots, c_n\} \\ &\quad c_1, c_2, \ldots, c_n \text{ are Circuit} \\ &|a.circuits| \geq 1 \\ &\forall i \forall j \{i \neq j \Rightarrow \neg pc(c_i, c_j) \wedge \neg lc(c_i, c_j)\} \end{aligned} \tag{3}$$

のような *Circuit* の集合 *circuits* を持つ．

PLCA 表現 以下の条件を満たすクラスを PLCA 表現のクラスと定義する．
(条件) 任意のインスタンス e が

$e.points = Point$ の集合
$e.lines = Line$ の集合
$e.circuits = Circuit$ の集合
$e.areas = Area$ の集合
$e.outermost = e.circuits$ の要素の中の 1 つ

のような *points*,*lines*,*circuits*,*areas*,*outermost* を持つ．*outermost* は図形の最も外側の枠を表す *Circuit* である．

以下ではある PLCA 表現 e があるとき, 特に指定がなければ $e.points$ を P,$e.lines$ を L,$e.circuits$ を C, $e.areas$ を A と表記する．

[1]領域 (region) と区別するために範囲 (Area) という言葉を使用する．

図 1: 簡単な図

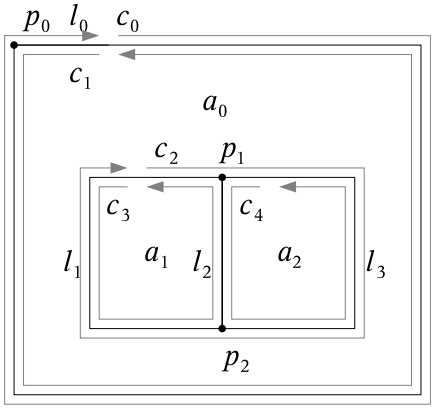

図 2: 各要素の名前

2.2 PLCA 表現の例

例として図 1 の PLCA 表現を考える．各要素の名前を図 2 に示す.

この図には 3 つの $Area$ a_1, a_2, a_3 が存在し，それぞれ c_0 と c_1, c_2, c_3 という $Circuit$ をもつ．$Line, Point$ も同様に決められる．c_{outer} は図形の外郭を表す $outermost$ である．このときにこの図を表す PLCA 表現は次のようになる.

$$
\begin{aligned}
&Points = \{p_0, p_1, p_2\}\\
&Lines = \{l_0, l_1, l_2, l_3\}\\
&Circuits = \{c_0, c_1, c_2, c_3, c_4\}\\
&Areas = \{a_0, a_1, a_2\}\\
&l_0.points = [p_0, p_0]\\
&l_1.points = [p_1, p_2]\\
&l_2.points = [p_1, p_2]\\
&l_3.points = [p_1, p_2]\\
&c_0.lines = [l_0^-]\\
&c_1.lines = [l_0^+]\\
&c_2.lines = [l_1^-, l_3^+]\\
&c_3.lines = [l_1^+, l_2^-]\\
&c_4.lines = [l_2^+, l_3^-]\\
&a_0.circuits = \{c_1, c_2\}\\
&a_1.circuits = \{c_3\}\\
&a_2.circuits = \{c_4\}\\
&outermost = c_0
\end{aligned}
$$

2.3 PLCA 表現の妥当性

2.3.1 妥当な $PLCA$ 表現

PLCA 表現 e が次の 3 つの制約条件を満たすとき e を**妥当な PLCA 表現**とよぶ.

$Point$-$Line$ 間の制約 すべての点は必ずいずれかの線の端である．点だけでの存在は許さない．線の両端の点はすべて $e.points$ に含まれる.

$$
\begin{aligned}
&\forall p \in P, \exists l \in L(p \in l.points)\\
&\forall l \in L(p \in l.points \Rightarrow p \in P)
\end{aligned}
\tag{4}
$$

$Line$-$Circuit$ 間の制約 すべての線は必ずいずれかの閉路に属している．1 つの線はすべての閉路の中で 2 回だけ出現する．閉路を構成する線はすべて $e.lines$ に含まれる.

$$
\forall l \in L \left(\begin{array}{l} |\{c|l^+ \in c.lines \wedge c \in C\}| = 1\\ \wedge |\{c|l^- \in c.lines \wedge c \in C\}| = 1\\ \wedge ((l^+ \in c_1 \wedge l^- \in c_2) \Rightarrow c_1 \neq c_2) \end{array} \right) \tag{5}
$$

Circuit-Area 間の制約 Outermost 以外のすべての閉路は必ずただ 1 つの範囲に含まれる.

$$
\begin{aligned}
&\forall c \in C\\
&\left(\begin{array}{l} c \neq e.outermost\\ \quad \Rightarrow |\{a|c \in a.circuits \wedge a \in A\}| = 1 \end{array} \right)\\
&\forall a \in A(c \in a.circuits \Rightarrow c \in C)
\end{aligned}
\tag{6}
$$

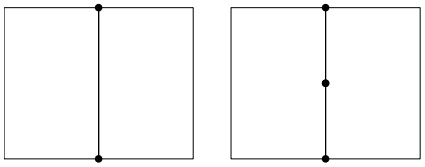

図 3: 冗長性

2.4 最小表現

PLCA 表現には *Line* ついての冗長性がある．これは図 2.4 のように区別する必要のない *Point* があるものである．[2] PLCA 表現 e は次の条件を満たすとき**最小表現**といい, 満たさないものを**冗長表現**という．

$$\neg\exists p \in P(|\{l|p \in l.points \wedge l \in L\}| = 2)$$

この意味を次に説明する．次のような条件を満たす *Point* p_1,*Line* l_1, l_2 が存在する時を考える．

$$\begin{aligned} & L_s = \{l|p_1 \in l.points \wedge l \in L\} \\ & \wedge\ L_s = \{l_1, l_2\} \\ & \wedge\ l_1 \neq l_2 \end{aligned}$$

このとき C_1, C_2, C_3 を次のようにとる．

$$\begin{aligned} C_1 &= \{c|p_1 \in l.points \wedge l \in c.lines \wedge c \in C\} \\ C_2 &= \{c|l_1 \in c.lines \wedge c \in C\} \\ C_3 &= \{c|l_2 \in c.lines \wedge c \in C\} \end{aligned}$$

このとき $C_1 = C_2 = C_3$ となる．これは p_1, l_1, l_2 を 1 つの *Line* に置き換えても接し方を変化させない．

冗長表現から条件を満たす *Point* と 2 つの *Line* を 1 つの *Line* に置き換えることで最小表現に変換できる．

2.5 PLCA 等価

2 つの PLCA 表現 e_1 と e_2 に対し，以下の条件を満たす全単射写像 f が存在すれば e_1 と e_2 が **PLCA 等価**であるという．

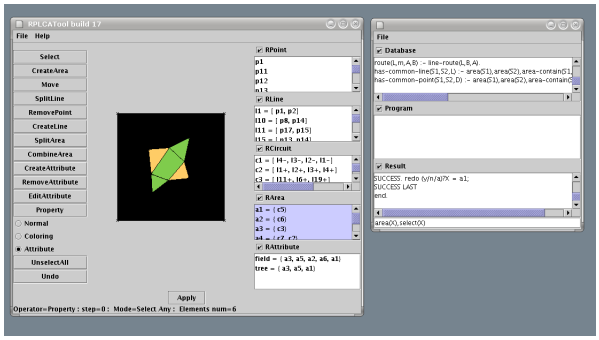

図 4: RxPLCATool

$$\begin{aligned} & \forall p \in e_1.points(f(p) \in e_2.points) \\ & \forall l \in e_1.lines(f(l) \in e_2.lines) \\ & \forall c \in e_1.circuits(f(c) \in e_2.circuits) \\ & \forall a \in e_1.areas(f(c) \in e_2.areas) \\ & \forall l \in e_1.lines(f(l.points) = f(l).points) \\ & \forall c \in e_1.circuits(f(c.lines) = f(c).lines) \\ & \forall a \in e_1.areas(f(a.circuits) \\ & \qquad = f(a).circuits) \end{aligned}$$

3 RxPLCATool

PLCA 表現を入力, 操作するツール RxPLCA を Java を用いて実装した．このツールはユーザが入力した図形データを PLCA の最小表現に変換するものである．さらに, 各 Area の上で成り立つ性質を Attribute という形で付加することによって, 表現と推論に幅を持たせた．また, このツールは Prolog を組み込んでいるため, PLCA 表現に対する質問を Prolog で記述し, Prolog の推論機構を利用して解を得ることができる．

追加の情報として *Attribute* という情報を付加した．これにより表現と推論に幅を持たせた．また, このツールでは Prolog を組み込むことで Prolog の推論機構を使用できる．

3.1 *Attribute* の定義

Attribute は以下の条件を満たすクラスと定義する．直観的にはその *Attribute* の性質を持つすべての *Area* の集合である．

(条件) 任意のインスタンス t が

$$\begin{aligned} & t.areas = \{a_1, a_2, \ldots, a_n\} \\ & a_1, a_2, \ldots, a_n \text{ are Area} \end{aligned} \tag{7}$$

[2] PLCA に情報を付加して区別する必要があるならばこの限りでない

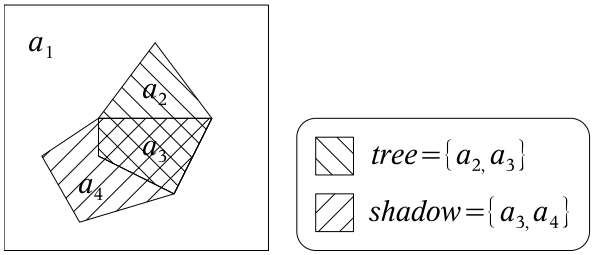

図 5: xPLCA 表現に対応する図

のような *Area* の集合 *areas* を持つ.

以下ではこの *Attribute* を追加した PLCA 表現を xPLCA 表現と呼ぶ.

3.2 RxPLCA と Prolog

xPLCA 表現は *Point*,*Line*,*Circuit*,*Area*,*Attribute* の各要素間の関係によって記述される.

この表現は 1 つのデータベースと考えられ, Prolog を使うと簡単に記述することができ, Prolog の推論機構を用いた推論が行える. RxPLCATool に組み込まれた Prolog には以下のような述語が組み込まれている.

- combinearea(X,Y) : *Area* X,Y が線接しているとき, 2 つの *Area* を結合して真を返す. 線接していなければ偽を返す.
- select(X) : 任意の要素 X を RxPLCATool 上で選択して真を返す. X が存在しなければ偽を返す.

combinearea は xPLCA 表現のデータベースを更新するため, 処理の前後で xPLCA 表現が変化する.

3.2.1 Prolog を用いた例

例として図 5 を表す PLCA 表現について説明する. 図 5 は上から見たとき,$a2, a3$ が木 (tree) であり, $a3, a4$ が影 (shadow) であるような状況を表す. 図 5 の PLCA 表現は次のようになる.

$$
\begin{aligned}
&Points = \{p3, p5, p6, p9\}\\
&Lines = \{l1, l2, l3, l6, l11, l12\}\\
&Circuits = \{c1, c2, c3, c4, c5, c7\}\\
&Areas = \{a1, a2, a3, a4\}\\
&Attributes = \{shadow, tree\}\\
&Outermost = c1\\
&l1.points = [p3, p3]\\
&l2.points = [p9, p5]\\
&l3.points = [p9, p6]\\
&l6.points = [p5, p6]\\
&l11.points = [p9, p5]\\
&l12.points = [p6, p9]\\
&c1.lines = [l1-]\\
&c2.lines = [l1+]\\
&c3.lines = [l6+, l3-, l2+]\\
&c4.lines = [l3+, l12+]\\
&c5.lines = [l2-, l11+]\\
&c7.lines = [l12-, l6-, l11-]\\
&a1.circuits = \{c3, c2\}\\
&a2.circuits = \{c5\}\\
&a3.circuits = \{c7\}\\
&a4.circuits = \{c4\}\\
&shadow.areas = \{a3, a4\}\\
&tree.areas = \{a2, a3\}
\end{aligned}
$$

PLCA 表現から Prolog のデータベースへの変換の方法は付録に示す. この PLCA 表現の Prolog のデータベースでの記述は次のようになる.

```
point(p9). point(p5).
point(p4). point(p6).
line(l4). line(l4,p6). line(l4,p9). line(l4,p6,p9).
line(l11). line(l11,p9). line(l11,p5). line(l11,p9,p5).
line(l2). line(l2,p5). line(l2,p9). line(l2,p5,p9).
line(l6). line(l6,p5). line(l6,p6). line(l6,p5,p6).
line(l12). line(l12,p6). line(l12,p9). line(l12,p6,p9).
line(l3). line(l3,p4). line(l3,p4,p4).
circuit(c7).
circuit(c7,l12,m). circuit(c7,l11,m,l12,m).
circuit(c7,l6,m). circuit(c7,l12,m,l6,m).
circuit(c7,l11,m). circuit(c7,l6,m,l11,m).
circuit(c5). circuit(c5,l2,p). circuit(c5,l11,p,l2,p).
circuit(c5,l11,p). circuit(c5,l2,p,l11,p).
circuit(c2).
circuit(c2,l3,p). circuit(c2,l3,p,l3,p).
circuit(c3).
```

circuit(c3,l6,p). circuit(c3,l2,m,l6,p).
circuit(c3,l4,p). circuit(c3,l6,p,l4,p).
circuit(c3,l2,m). circuit(c3,l4,p,l2,m).
circuit(c1).
circuit(c1,l3,m). circuit(c1,l3,m,l3,m).
circuit(c4).
circuit(c4,l4,m). circuit(c4,l12,p,l4,m).
circuit(c4,l12,p). circuit(c4,l4,m,l12,p).
area(a1). area(a1,c2). area(a1,c3).
area(a3). area(a3,c7).
area(a4). area(a4,c4).
area(a2). area(a2,c5).
attribute(shadow).
attribute(shadow,a3). attribute(shadow,a4).
attribute(tree).
attribute(tree,a3). attribute(tree,a2).

操作の例

ここではそのいくつかの例を示す.

2 つの *Area* S1,S2 が *Line* L を共有するという述語 has-common-line(S1,S2,L) は次のように記述できる.

```
has-common-line(S1,S2,L) :-
  area(S1),area(S2),area(S1,C1),area(S2,C2),
  circuit(C1,L,_),circuit(C2,L,_).
```

2 つの *Area* S1,S2 が *Point* P を共有するという述語 has-common-point(S1,S2,P) は次のように記述できる.

```
has-common-point(S1,S2,P) :-
  area(S1),area(S2),area(S1,C1),area(S2,C2),
  circuit(C1,L1,_),circuit(C2,L2,_),
  line(L1,P),line(L2,P).
```

tree の *Attribute* を持ったすべての *Area* を選択したいとする．このとき次のような節を入力すれば選択できる.

```
attribute(tree,X), select(X).
```

tree の *Attribute* を持ち,*shadow* の *Attribute* を持たないすべての *Area* を選択したいとする．このとき次のような節を入力すれば選択できる[3].

```
attribute(tree,X),
  \+ attribute(shadow,X),
  select(X).
```

[3] \+ は否定を意味する

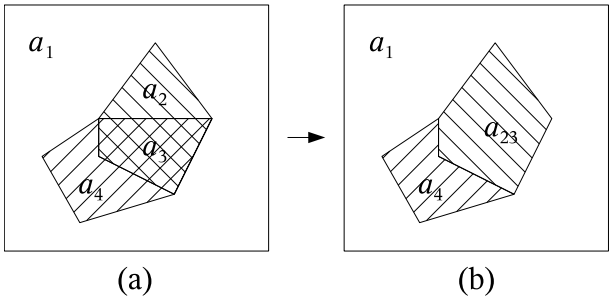

図 6: Combine Area

図 6 のように *tree* の *Attribute* を持った *Area* をすべて結合したいとする．この処理は *tree* に含まれる *Area* について, すべての 2 つずつの組み合わせについて, 2 つの *Area* が線接しているならば結合処理を行えばよい．この処理を RxPLCATool 上の Prolog 上で行うには次のような節になる.

```
attribute(tree,X),
  attribute(tree,Y),
  X \ == Y,
  combinearea(X,Y).
```

以上のように PLCA 表現に Prolog の推論機構を組み込むことで様々なことができるようになる．これらの節には直観に沿ったものがあり, それらを予め作成しておけば直観的な推論の手助けになる.

4 考察

PLCA 表現は形の情報がない．形を考慮しなくて接し方のみを取り扱えばよい問題に対しては PLCA 表現を使えば余計な情報を取り扱わなくてすむ.

PLCA 表現の最小表現は接し方を一意的に表現する．たとえば複数の二重連結辺リストがあるとき, 接し方について等しいものごとに分類する問題を考える．この問題は二重連結辺リストを PLCA 表現の最小表現に変換して, PLCA 表現が等しいものごとに分類すれば解決する.

本発表で使用した PLCA 表現の Prolog のデータベースとしての記述の仕方は 1 度に 1 つの表現しか扱えないが, 記述の方法を変えれば複数の PLCA 表現間での推論, 探索が行えるだろう.

PLCA 表現と二重連結辺リスト　二重連結辺リストは計算幾何学や GIS でよく使われる平面領域分割の代表的な表現方法の 1 つである．PLCA 表現は二重連結辺リストと類似のデータ構造を持つ．二重連結

辺リストは頂点, 辺, 面で表される．各要素の対応は

$PLCA$ 表現		二重連結辺グラフ
$Point$	$\leftrightarrow$	頂点
$Line$	$\leftrightarrow$	辺
$Circuit, Area$	$\leftrightarrow$	面

となる．

二重連結辺リストと PLCA 表現の違いは二重連結辺リストの 1 つのリストは 1 つの図形を表すが, PLCA 表現の 1 つの表現は接し方が同じ複数の図形を表ことである．これは二重連結辺リストは形の情報を持つのに対し, PLCA 表現は形の情報を持たないためである．二重連結辺リストでは頂点は座標を持ち, 辺は両端の頂点を結ぶ直線または特定の形の曲線[4] として扱われる．PLCA 表現では Point は座標を持たず,Line は一本に定まらない[5].

PLCA 表現の Point に座標の情報を付加し, すべての Line を両端の Point の座標を結ぶ特定の曲線としたとき, 二重連結辺リストとなる．

一般に二重連結辺リストの表現から座標データを除いたものは冗長な PLCA 表現とみなすことができる．

4.1 PLCA 表現と他の定性空間推論

代表的な定性空間推論の枠組みに，領域を基本構成要素として領域同士の関係を公理として体系づける RCC に代表される方法がある．RCC は 2 つの領域があるとき, それらの関係を二項関係で与える．それに対し，PLCA 表現 は要素間のつながりで全体を 1 つの表現として表す[6].

RCC には基礎関係を 8 つとする RCC-8 がある．これの基礎関係は図 7 のようになる．それぞれ 1 つめの領域 (r_1) と 2 つめの領域 (r_2) との関係である．DC は 2 つの領域が離れているとき, EC は 2 つの領域が点接または線接しているとき, PO は 2 つの領域が交差しているとき (共有した領域があるとき), EQ は 2 つが等しいとき, TPP は内接するとき, NTPP は内包されるとき, TPPi は外接するとき, NTPPi 内包するときを意味する．RCC-8 では 2 つの領域があるとき, その関係は必ずこの中のどれか 1 つになるものとしている．PLCA 表現の $Area$ は RCC-8 でいう領域とは対応しない．しかし xPLCA 表現上の同一の $Attribute$ を持つ $Area$ の集合を領域として考えることができる．故に xPLCA 表現の $Attribute$ が RCC-8 でいう領域と対応する．2 つの $Attribute$ に対して RCC-8 の関係を定義することが可能になる．

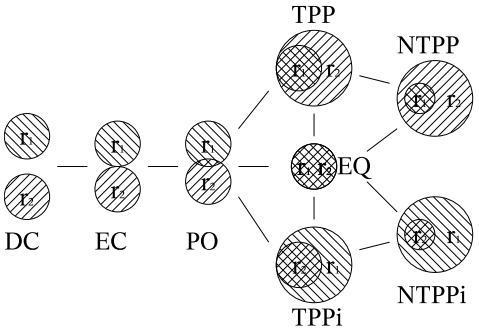

図 7: RCC-8 の関係

5 まとめ

定性空間推論の枠組みとなる PLCA 表現を提案し, その拡張例と応用例を示した．PLCA 表現は, 領域の接し方に着目することで推論力をあげ, 簡単なオブジェクトを基礎とすることで実装や拡張を容易にする．

PLCA 表現の妥当性を判定するアルゴリズムおよび同一性の判定アルゴリズム, さらに本発表では述べていないが, 与えられた 2 次元の図形表現から PLCA 表現を生成するアルゴリズムはすべて実装済である．

PLCA 表現は, 情報の付加によって拡張が考えられる．たとえば, 色や形といった範囲のもつ属性を付加情報として与えることによって応用の可能性が広がる．拡張例, 応用例としてあげた RxPLCATool では PLCA 表現を拡張した xPLCA 表現を導入し, Prolog でシステムを構築することによって直観的な推論が行えた．

また, 範囲の分断などのオペレータを導入することで, 時間的変化に対する扱いも可能になる．今後は,PLCA 表現をベースとしてそこに付加する情報とオペレータを検討し PLCA 表現のさらなる可能性を探りたい．

[4]形の情報がなんらかの方法で保存され決定している．通常は直線.

[5]全体の接し方を崩さなければどんな形でもよい

[6]逆に PLCA 表現は 2 つの要素のみの関係を記述するには向かない.

参考文献

[1] B. Bennett. Modal logics for qualitative. In *Journal of IGPL, Vol.4, No.1*, pages 23–45, 1996.

[2] B. Bennett, Anthony G. Cohn, Paolo Torrini, and Shyamanta M. Hazarika. Describing rigid body

motions in a qualitative theory of spatial regions. In Henry A. Kautz and Bruce Porter, editors, *Proceedings of AAAI-2000*, pages 503–509, 2000.

[3] L. Clarke, B. A calculus of individuals based on 'connection'. In *"Notre Dame Journal of Formal Logic,22(3)"*, 1981.

[4] L. Clarke, B. Individuals and points. In *"Notre Dame Journal of Formal Logic,26(1)"*, 1985.

[5] M. Cristani, A.G. Cohn, and B. Bennett. Spatial locations via morpho-mereology. In *Proceedings of KR'2000*, pages 15–25, 2000.

[6] M. Egenhofer and R. Golledge(ED.). *Spatial and Temporal Reasoning in Geographic Information Systems*. Oxford University Press, 1998.

[7] M. Egenhofer and J. Herring. Categorizing binary topological relations between regions, lines, and points in geographic databases. In *Department of Surveying Engineering*. University of Maine, 1990.

[8] Antony Galton. Towards a qualitative theory of movement. In *Spatial Information Theory: A Theoretical Basis for GIS (COSIT95)*, pages 377–396, 1995.

[9] K.Takahashi and T.Sumitomo. A framework for qualitative spatial reasoning based on connection patterns of regions. In *"IJCAI05 Workshop on Spatial and Temporal Reasoning"*, 2005.

[10] Philippe Muller. A qualitative theory of motion based on spatio-temporal primitives. In Anthony G. Cohn, Lenhart Schubert, and Stuart C. Shapiro, editors, *KR'98: Principles of Knowledge Representation and Reasoning*, pages 131–141. Morgan Kaufmann, San Francisco, California, 1998.

[11] D A Randell, M Witkowski, and M P Shanahan. From images to bodies. In *Proceedings of the Sixth International Conference on Knowleges Representation and Reasoning*, pages 131–141, 2001.

[12] David A. Randell, Zhan Cui, and Anthony Cohn. A spatial logic based on regions and connection. In Bernhard Nebel, Charles Rich, and William Swartout, editors, *KR'92. Principles of Knowledge Representation and Reasoning: Proceedings of the Third International Conference*, pages 165–176. Morgan Kaufmann, San Mateo, California, 1992.

[13] O. Stock, editor. *Spatial and Temporal Reasoning*. Kluwer Academic Press, 1998.

[14] 住友孝郎 and 高橋和子. DLCS: 空間の定性的な表現方法. In *"**日本ソフトウェア科学会第 21 回大会**"*, 2004.

[15] 住友孝郎 and 高橋和子. 定性空間推論の新しい枠組 dlcs とその上での操作. In *"**情報処理学会プログラミング研究会 発表資料**"*, 2005.

A xPLCA 表現の Prolog 上での記述方法

本稿で使用している xPLCA 表現の Prolog 上で表現するための述語を説明する. 以下は

〈述語の名前〉/〈引数の数〉

で表記される.

point/1 第 1 引数で示される名前の *Point* が存在することを意味する.

line/1 第 1 引数で示される名前の *Line* が存在することを意味する.

line/3 第 1 引数は *Line* を, 第 2 引数と第 3 引数は *Point* を指す. 第 1 引数の *Line* は第 2 引数の *Point* から第 3 引数の *Point* への *Line* で在ることを意味する.

circuit/1 第 1 引数で示される名前の *Circuit* が存在することを意味する.

circuit/3 第 1 引数は *Circuit* を, 第 2 引数は *Line* を, 第 3 引数は方向 (順方向なら'p', 逆方向なら'm') を指す. 第 1 引数の *Circuit* に第 2 引数の *Line* が第 3 引数の指す方向で含まれることを意味する.

circuit/5 第 1 引数は *Circuit* を, 第 2 引数は *Line* を, 第 3 引数は方向を, 第 4 引数は *Line* を, 第 5 引数は方向を指す. この述語は第 1 引数の *Circuit* を構成する *lines* の列の一部分を意味する. *lines* の中で第 2 引数の *Line* の次に第 4 引数の *Line* が出現することを意味する. 第 3 引数, 第 5 引数はそれぞれ第 2 引数の *Line*, 第 4 引数の *Line* の方向を意味する.

area/1 第 1 引数で示される名前の *Area* が存在することを意味する.

area/2 第 1 引数は *Area* を, 第 2 引数は *Circuit* を指す. 第 1 引数の *Area* は第 2 引数の *Circuit* を持つことを意味する.

attribute/1 第 1 引数で示される名前の *Attribute* が存在することを意味する.

attribute/2 第 1 引数は *Attribute* を, 第 2 引数は *Area* を指す. 第 1 引数の *Attribute* は第 2 引数の *Area* を持つことを意味する.