オブジェクトペトリネットの構造描画

4W−03

鈴木 圭　三浦 孝夫
法政大学工学部電気電子工学科
塩谷 勇
産能大学経営情報学部

概 要

本稿で定義する有向グラフの一つであるオブジェクトペトリネット（OPN）の構造的配置を損なうことなく美しく描画する方法を提案する。これを可能とするため、OPN をリストで表現し、構造的な配置に必要な情報を損なうことなく入力、描画するアルゴリズムを述べる。

1　前書き

グラフ描画を考える際、美しく描画するということを考えることが多いが、美しく配置されると構造的配置が失われてしまうことがある。構造的配置が失われてしまうと、ユーザにとってそのグラフは見やすいグラフであるが、理解しづらいグラフとなってしまう。グラフを美しく描画する方法として参考文献 [1] では、スプリングモデルや、拡大モデルなどが挙げられているが、本稿ではその中の階層モデルを用いて OPN の構造的配置を損なわないよう美しく描画する。このことによりユーザにとって理解しやすいグラフ描画を実現する。

2　オブジェクトペトリネット

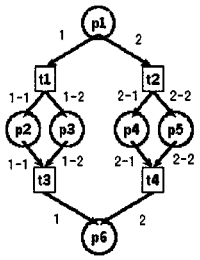

図 1：プライムブロック

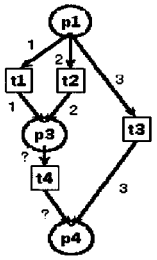

図 2：エンプティノード

本稿ではオブジェクト ID を持つカラーペトリネットを OPN と定義する。OPN は 1 ソース、1 シンク、輪無し、つまりオブジェクト ID の発火条件を満たす形を基本形（プライムブロック）とし、このプライムブロックを用いて OPN を描画する。このことにより OPN を構造的かつ美しく描画することが可能となる。

Graph Drawing of Object Petri Net
Kei Suzuki , Takao Miura
Hosei University, Dept.of Elec. and Elec. Eng.
Kajino-cho 3-7-2, Koganei, Tokyo, JAPAN
Isamu Shioya
Sanno University, Dept.of Management and Information
Kamikasuya 1563, Isehara, Kanagawa, JAPAN

(1) オブジェクト ID

OPN では全てのトークンがオブジェクト ID を持つ。オブジェクト ID は幹と枝を持ち、任意のトランジションで発火、分岐したトークンは分岐前の ID を幹とし、分岐後に加えられる ID を枝とする。図 1 のトランジション t1 で分岐したトークンはそれぞれ幹が 1、枝が 1 と 2 のトークンに分岐している。分岐したトークンがトランジションで発火、合併する際は同じ幹を持つ全てのトークンが揃わなければならない。図 1 ではトランジション t1 で分岐したオブジェクト ID が 1-1、1-2 のトークンが t2 でそろっているので合併することができる。

(2) プライムブロック

オブジェクト ID の発火条件を満たす基本形をプライムブロックと定義する（図 1）。オブジェクト ID の発火条件に適応しない形、すなわちプライムブロックで表現できない形を描画する際には、プライムブロックを変形した形で表す（図 2）。このことでプライムブロックによる構造的配置を保つことが可能となる。

3　リスト表現

OPN のリスト化はユーザーが行うものとする。本稿で提案するアルゴリズムはリスト化されたオブジェクトペトリネットを描画するものである。

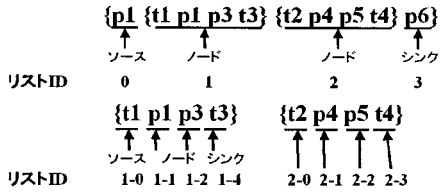

図 3：リスト ID

(1) オブジェクトのリスト表現

リストの要素はソース、シンク、ノードの三種類で表す。リストの要素をノードと言い、その中でも最初のノードをソース、最後のノードをシンクとする。ソースは一番上に、シンクは一番下に、ノードはソースとシンクの間に左から順番に配置される。図 1 を例に挙げると、入力したリストは {p1 {t1 p2 p3 t3} {t2 p4 p5 t4} p6}、ソースは p1、シンクは p3、ノードは {t1 p2 p3 t3}、{t2 p4 p5 t4} となる。さらに {t1 p2 p3 t3} はソース t1、シンク t3、ノード p2、p3 となり、{t2 p4 p5 t4} はソース t2、シンク t4、ノード p4、p5 となる。

(2) 非オブジェクトのリスト表現

非オブジェクトを表現するための特別なリスト要素を三つ挙げる。:(エンプティノード) は、空のソースまたはシンクを与え、実際にはノードとして配置されない。= (共有) は、共有プレースまたは共有トランジションを作り出す。> (ポインタ) はリストによる結線ではなく、ポインタで指定された結線を行う。

(3) リストの入力条件

リスト入力の条件は以下のようになる。入力するリストを、t*と p* (*は任意の数字) と限定する。t*の場合はトランジション、p*の場合はプレースとする。また原則として、トランジション内およびプレース内では同じ数字を使わないこととする。例外として、共有を表す時のみ使用できる。

4 ノードの描画

ノードの描画は、最初にリストにリスト ID を与え、その ID を元にフレームでウィンドウ上に枠組みを作り、その上にノードを配置する。

(1) リスト ID

まず入力したリストのノードにそれぞれ ID をつけていく。この ID をリスト ID と呼ぶ。図 1 のリスト {p1 {t1 p2 p3 t3} {t2 p4 p5 t4} p6} を用いると、p1 : 0, {t1 p2 p3 t3} : 1, {t2 p4 p5 t4} : 2, p6 : 3 と ID をつけていく。この作業は各リストのノードの数が 1 になるまで続けられる。結果各ノードのリスト ID は、t1 : 1-0、t2 : 2-0、t3 : 1-3、t4 : 2-3、p1 : 0、p2 : 1-1、p3 : 1-2、p4 : 2-1、p5 : 2-2、p6 : 3、となる。これよりノードは階層的な ID を持つことが分かる。またリスト ID はオブジェクト ID と同様に幹と枝に分けることができる (図 1)。

(2) フレームの配置

リスト ID を元にフレームでウィンドウに枠組みを作る。リスト ID の長さが 1 のものはルートウィンドウを、リスト ID の長さが 2 以上のものは幹部分のリスト ID を持つフレーム中を分割していく。リスト ID の長さが 1 のものから順番に配置していく。この作業はリストのノードの数が 1 になるまで続けられる。

(3) ノードの配置

次にノードの配置を行う。ノードに与えられたリスト ID と同じ ID を持つフレーム上にノードを配置する。これでノードの配置は終了する。これよりフレームを配置した時点でノードの配置も決まることが分かる。結果を図 3 に示す。

(4) 非オブジェクトの配置

非オブジェクトのフレーム、ノード配置についてエンプティノードを用いて述べる。リスト ID の決定、フレームの配置までは OPN と同じように行う。次にノードを配置する。この時にエンプティノードは配置しない。しかし、ノードの配置はフレームの配置に依存するので、OPN によって行われた構造配置を損なうことなく非オブジェクトの配置をすることが可能となる (図 2)。

(5) トランジションとプレースの判別

ノードにはトランジションとプレースの二種類がある。この判別はリスト入力時の制約で必ず t* もしくは p* で与えられるので、これにより判断することができる。

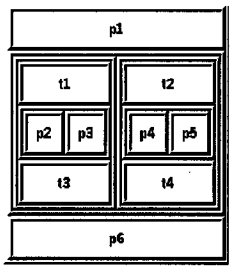

図 4 : フレーム、ノードの配置

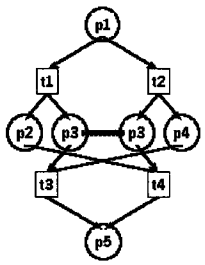

図 5 : ポインタ、共有

5 結線

結線はリスト ID を元に行われる。またソースからノードへ、ノードからシンクへ結線される。図 1 を用いると、ソースは p1、ノードは {t1 p2 p3 t3} と {t2 p4 p5 t4}、シンクは p6 である。

(1) ソースの結線

ソースからノード (p1 から {t1 p2 p3 t3}、{t2 p4 p5 t4}) へ結線する。しかし各ノード {t1 p2 p3 t3}、{t2 p4 p5 t4} はリストのノードの数が 1 ではない。このときは各ノードのソースに、つまり t1 と t2 に結線する。

(2) ノードの結線

次にノードからソース ({t1 p2 p3 t3}、{t2 p4 p5 t4} から p6) へ結線する。しかし、ノード {t1 p2 p3 t3} のリストのノードの数が 1 でないので各ノードのシンクからシンク、つまり t3、t4 から p6 に結線する。{t1 p2 p3 t3}、{t2 p4 p5 t4} 内も同様に行う。

(3) 結線による非オブジェクト

結線による非オブジェクトをポインタと共有を用いて表す (図 4)。図 4 のリストは {p1 {t1 p2>t4 =p3 t3} {t2 =p3 p4>t3 t4} p5} である。共有を表す"=p3"により二つの p3 は青色の線で結ばれている。またポインタ"p2>t4"により p2 からの結線が t3 では無く t4 につながっている。同様に"p4>t3"により p4 からの結線は t3 につながっている。これらにより美的、構造的な描画を損なうことなく非オブジェクトの描画が可能となった。

6 結び

OPN をリスト表現にすることにより、それぞれのノードは階層的な意味を持ち、プライムブロックを用いることにより、構造的な配置を損なうことなく、美的に描画できた。本稿では輪無しに限定して考えた。その結果 OPN に限定された描画のみ可能となった。

参考文献

[1] Di Battista,G.,Eades,P.,Tamassia,R.,Tollis,I.,Graph Drawing:Algorithms for the Visualization of Graphs,Prentice Hall,Upper Saddle River,1999.

[2] 椎塚久雄: 実例ペトリネット, コロナ社 (1992)