

因果ループ図の形式定義と性質解析

中島 震^{1,a)} Guillermina Cledou^{2,b)}

概要: 因果ループ図はビジネス・ダイナミクスで用いるモデリング記法である。対象の系全体を閉じたループの集まりとして表すことで、複雑に絡み合う系の動的な振舞いを表現する。変化伝播の直感的な解釈にしたがうと、仕様アニメーションあるいはインスペクションによる性質確認を行うことができる。しかし、自動解析が可能ほど厳密に規則が決められているわけではない。本稿は、因果ループ図の操作的な意味に形式定義を与え、振舞い仕様の性質を自動検査する方法を提案する。ネット指向形式体系を採用することで、因果ループ図の特徴である真の並行性を表す。

キーワード: CPS, システム思考, ペトリネット, 可達グラフ, モデル検査

Formal Definition and Analysis of Causal Loop Diagrams

SHIN NAKAJIMA^{1,a)} GUILLERMINA CLEDOU^{2,b)}

Abstract: Causal Loop Diagrams (CLD) are modeling notations employed in Business Dynamics. Such a diagram consists of many tightly coupled loops to represent a system's dynamic behavior. Intuitive operational semantics, describing how changes are propagated along the loops, provide a basis for checking properties by means of specification animation or manual inspection, although they are not rigorous enough. This paper introduces a formal definition of dynamic behavior of CLD to enable automated checking of properties. The definition employs a net-based formal system to represent true concurrency of CLD.

Keywords: CPS, Systems Thinking, Petri Nets, Reachability Graph, Model Checking

1. はじめに

サイバーフィジカルシステム (Cyber-Physical Systems, CPS)[14] は, IoT ビジネス・エコシステム [9] を実現するソフトウェア技術の新しいパラダイムである。ICT からみると, ビジネス・エコシステムは多数の独立なシステムがコネクティビティによって有機的に結びつく巨大な系 System-of-Systems (SoS)[4] とみることができる。

SoS を構成するシステムは直接あるいは間接的につながることで互いに影響し合う。構成システム間の相互作用はループ状になり, 自身の変化が回り巡って自分に影響するフィードバックが生じる。原因と結果をつなぐ因果関係が

明らかでないことも多い。系の何処かで生じた微小な擾乱が伝播することで予想外の変化に至ることから, ソフトウェア工学が対応すべき複雑さ [10] に新たな次元をもたらす。『複雑なシステムは, 隠れた閉じたループの存在によって, 予期しない強い相関が生じ, 系全体に不具合を引き起こす』[8] ことがある。複合的に絡み合う系の隠れた相関を見逃してはならない [7]。

ビジネス・ダイナミクス [12] は, 対象の系全体を閉じたループの集まりとして表現し, 複雑な系の動的な振舞いを描き出すモデリング法である。数学的なモデルとしては非線形ダイナミクスのフィードバック制御と見做せる。一方で, これを厳密に表現する手間は大きく, また, そのような非線形の式は自動的な解析を困難にする。そこで, 系全体の定性的な振舞いを取り扱う因果ループ図 (Causal Loop Diagrams, CLD) の方法が提案された。運輸 [11] をはじめ,

¹ 情報・システム研究機構 国立情報学研究所
ROIS NII, Chiyoda-Ku, Tokyo 101-8430, Japan

² HASLab INESC TEC & University of Minho, Portugal

a) nkjm@nii.ac.jp

b) mgc@inesctec.pt

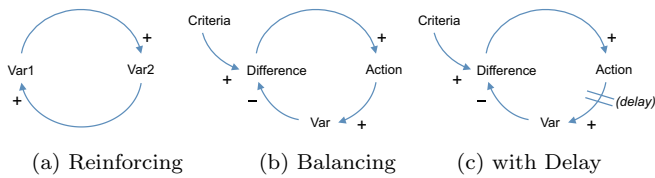


図 1 Basic Components

さまざまなシステムの分析に応用されている。また、宇宙ロケットを対象とした安全性分析の基本的な考え方として採用された [3]。

動的な振舞いが意図通りであるかは、変化伝播の直感的な解釈にしたがって、インスペクションあるいは仕様アニメーションで確認できる。しかし、自動解析が可能なほど厳密に規則が決められているわけではない。

本稿は、CLD の自動検証を目的とした形式モデル Causal Loop Net (CLN) を提案する。CLD の特徴である「真の並行性」を表す目的で導入したネット指向形式体系である。ペトリネット [5][6] と同様に、可達グラフによって対象系全体の動的な振舞いを表現する。提案する CLN では有限グラフになることを保証でき、系が示す興味深い性質を自動解析することが可能になる。

2. 因果ループ図

2.1 ダイアグラム表現

因果ループ図 (Causal Loop Diagrams, CLD) は非線形ダイナミクスのフィードバック制御系を定性的に表現するモデリング法である。変数が互いにおよぼす影響をループ図式で表現する。

CLD は図 1 に示した 3 つの基本コンポーネントを提供する。図 1(a) は変数 Var1 と Var2 とが互いに強め合う (Reinforcing) 関係にあることを示す。Var1 が増加すると Var2 も増加し、また、Var2 の増加は Var1 の増加として現れる。図 1(b) は変数 Difference と Var が均衡する (Balancing) 関係にあることを示す。Difference の増加は Var の増加を導く一方、Var が増加すると Difference を減少させる。つまり、均衡するようにフィードバックが働く。図 1(c) は変化の伝播に遅延 (Delay) がある場合に均衡する関係を示す。本稿では、図 1(b) および (c) の Criteria のように外向き遷移のみが課された変数を端点と呼ぶ。

対象の系を基本コンポーネントの組合せとして表す時、変数共有によって複合的なループを形作る。図 2 は文献 [12] の図 5-21 を再掲したもので、大学の授業で出題される宿題の積み残し (Assignment Backlog) と勉強にかける時間 (Workweek) などの関係を表す。この図を参照すると、系全体としては 2 つのループ構造を持ち、変数 WP (Work Pressure) にフィードバックがかかることが読み取れる。

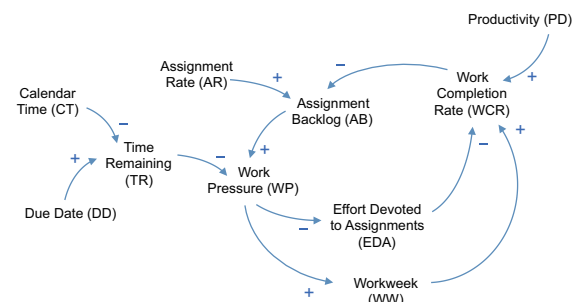


図 2 Example Composed Loops (Figure 5-21 in [12])

2.2 因果ループ図の定義

因果ループ図の構造的な特徴を厳密に表現する。CLD は (V, I, ℓ, τ) からなる 4 つ組で定義できる。ここで、 V は変数名を表す記号の有限集合、 I は変数の初期化関係を表す。 ℓ は 2 変数をつなぐ遷移関係であり、 $\text{Polarity} \stackrel{\text{def}}{=} \{+, -\}$ とする時、 $\ell: V \times \text{Polarity} \times V$ である。また、 $\tau: V \times \text{Polarity} \times V$ は遅延注釈が付記された遷移関係である。ただし、これらに重なりがない ($\ell \cap \tau = \emptyset$)。Polarity の要素は、遷移関係の出発点の変数 v_s 、到着点の変数 v_d とする時、 $\text{sign}(\partial v_d / \partial v_s)$ である。

CLD の動的な振舞いは、遷移関係、変数変化の伝播、の発火系列で決まる。今、大域的な時間軸を決め、ある時点 t で発火可能な遷移関係の集まりを S_t で表す。 S_t から取り出した遷移関係が ℓ の場合は直ちに変化を伝播する。一方、遅延 τ の場合、 ℓ に書き換えた後、ある $d (d > 0)$ を選び S_{t+d} に追加する。つまり、 d 後にあらためて遷移関係を発火させる。

CLD は遷移関係を発火可能とする条件が厳密に決められているわけではない。端点となる変数を初期化し、その変数からの遷移関係を追跡すれば良いだろう。なお、第 4 節で CLD の遷移発火の規則を厳密に定義する。

3. ネット指向モデリング手法

CLD は複数の遷移関係が同時発火でき、真の並行性 (True Concurrency) に基づく。そのような計算モデルとして、ペトリネット [5][6] の考え方を参考にする。

3.1 真の並行性とペトリネット

3.1.1 ペトリネット

ペトリネット*1は、プレースとトランジションという 2 種類のノードを持つ重み付き有向 2 部グラフ (Weighted Directed Bipartite Graph) である。図 3 に簡単な例を示した。

定義：ペトリネット

P : プレース (Places) の有限集合

T : トランジション (Transitions) の有限集合

*1 Place/Transition Nets (PT-nets)

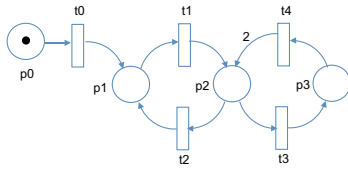


図 3 Petri Net

F : フロー関係 $F \subset (P \times T) \cup (T \times P)$

W : 重み付け関数 $F \rightarrow \mathcal{N}$

M_0 : 初期マーキング $P \rightarrow \mathcal{N}_0$

プレースはトークン (Token) を保持することができる。トークンは互いに区別されない。プレースをトークンの多重集合 (Multiset) とする。プレースが保持するトークンの関係を表すマーキング (Marking) を導入する。初期状態でのマーキングを初期マーキング M_0 と呼ぶ。素朴には、トランジションの入力プレースにトークンが配置されている時、そのトランジションが発火可能になり、遷移発火によって出力プレースにトークンを移送する。

トランジション t に対して、関係するプレースの集合、

入力プレースの集合 $\bullet t = \{ p \mid (p, t) \in F \}$

出力プレースの集合 $t \bullet = \{ p \mid (t, p) \in F \}$

を定義する。

現在のマーキングを M とする時、 $M(p)$ はプレース p のトークン数を表す。マーキング M に対して遷移発火可能なトランジションの集まり $\mathcal{T}(M)$ は次のように表せる。

$$\mathcal{T}(M) = \{ t \in T \mid \bigwedge_{p \in \bullet t} (M(p) \geq W(p, t)) \}$$

同時発火可能なトランジションが複数ある時、競合しない限り、同時に発火する。つまり、真の並行性を示す。

系全体の動的な振舞いをマーキングの変化として表す。 $M \xrightarrow{\mathcal{T}(M)} M'$ は、すべてのプレース ($\forall p \in P$) に対して、次のように定義できる。

$$M'(p) = M(p) - \sum_{t \in \bullet t} W(p, t) + \sum_{t \in t \bullet} W(t, p)$$

この関係にしたがって、初期マーキング M_0 から出発したトランジションの遷移発火によって変化するマーキングの全体は到達グラフ (Reachability Graph) を構成する。到達グラフを調べることで、動的な振舞い性質を知ることができる。

図 3^{*2}を用いて、トランジション発火とマーキングの例を示す。 $M_0 = \{p0 \mapsto \bullet\}$ であり、トランジション $t0$ と $t1$ の発火後、 $M_2 = \{p2 \mapsto \bullet\}$ となる。この時、 $t2$ と $t3$ が競合する。 $t3$ が発火すると $t4$ の後、 $M_4 = \{p2 \mapsto \bullet, \bullet\}$ となるので、 $t2$ と $t3$ が共に発火可能になる。同時発火した結果、 $M_5 = \{p1 \mapsto \bullet, p3 \mapsto \bullet\}$ であり、 $t1$ ならびに $t4$ が同時に発火すると、 $M_7 = \{p2 \mapsto \bullet, \bullet, \bullet\}$ になる。トークンの数が増加していくので有界性を満たさない例である。

^{*2} 無限容量ネット (infinite capacity net)

3.1.2 色付きペトリネット

一般には、トークンを区別して扱いたいことがある。色付きペトリネット (CP-net)[1] は、トークンに色 (Color) を付加することで、トランジション発火等において特定トークンを指定できるようにした^{*3}。直感的には、トークンは PT-nets と同様に遷移発火過程を制御し、トークンの色が値を表すと考えれば良い。つまり、遷移発火と共に値が伝播する。

プレースが保持するトークンを色で分けられた色トークンの多重集合の集まり ($M = \prod_c M^c$) とすると考えやすい。

色付きペトリネットの簡略化した定義を示す。

定義：色付きペトリネット

P : プレース (Places) の有限集合

T : トランジション (Transitions) の有限集合

F : フロー関係 $F \subset (P \times T) \cup (T \times P)$

G : ガード関数 $T \rightarrow \text{Pred}$

E : 式関数 $F \rightarrow \text{Exp}$

M_0 : 初期マーキング $P \rightarrow (\text{Color} \rightarrow \mathcal{N}_0)$

トランジションの発火は、注釈言語で書き表したガード関数 G と式関数 E で規定される。これらの関数は変数を持つことから、変数と色 (値) の対応関係を管理するバインディング b を導入する。 b はトランジションに対して付与されるので、便宜上、 $B: T \rightarrow \text{Binding}$ を導入し、 $b = B(t)$ とする。 b の下での結果を $\llbracket G(t) \rrbracket_b$ ($t \in T$) と $\llbracket E(f) \rrbracket_b$ ($f \in F$) で表す。

色は型の要素であり、変数 x の型情報を $\text{Type}(x)$ とする。バインディング b 下で変数 x の色 ($b(x)$) の型は $\text{Type}(x)$ であり、 $b(x) \in \text{Type}(x)$ と書ける。CP-net は強く型付けされているので、プレース、ガード関数、式関数は、型制約にしたがう。 $G(t)$ は述語であり、 $\llbracket G(t) \rrbracket_b$ が真の時、 t のガード条件が成り立つ。

同時発火可能なトランジション $\mathcal{T}(M)$ は、色 (値) に対して定義される。そこで、便宜上、式関数 $E(f)$ の本体 Exp を色 C のトークン数について定義した式 Exp^c を考えると、 $\llbracket \text{Exp}^c \rrbracket_b \in \mathcal{N}$ である。

色 c を決めた時、マーキング M に対して遷移発火可能なトランジションの集まり $\mathcal{T}^c(M)$ は次のように表すことができる。

$$\mathcal{T}^c(M) = \{ t \in T \mid \bigwedge_{p \in \bullet t} M^c(p) \geq \llbracket E^c(p, t) \rrbracket_{B(t)} \}$$

マーキング M に対して遷移発火可能なトランジションの集まり $\mathcal{T}(M)$ は、 $\mathcal{T}^c(M)$ を色について集めたものと見なせる。

$$\mathcal{T}(M) = \prod_c \mathcal{T}^c(M)$$

PT-nets と同様に、同時発火可能なトランジションは、競

^{*3} CPN Tools の CPN[2] は注釈言語に StandardML を用いて CP-net を具体化したモデリング言語である。

合しない限り、同時に発火する。また、マーキングの変化 $M \xrightarrow{\tau(M)} M'$ は、すべてのプレース ($\forall p \in P$) に対して、次のように定義できる。

$$M'(p) = M(p) - \sum_{t \in \mathcal{T}(M)} \llbracket E(p, t) \rrbracket_{B(t)} \\ + \sum_{t \in \mathcal{T}(M)} \llbracket E(t, p) \rrbracket_{B(t)}$$

マーキングの遷移関係から可達グラフを構成する。

3.2 抽象化の導入

定性的な抽象化について3つの側面から考察する。

3.2.1 定性的な抽象値

第2.2節に述べたように、CLDの変数は定まった値を持つわけではなく、その変化（増加あるいは減少）を伝播する。一方、抽象化した値をとると考えると、その値をCP-netsのカラーに対応させることができるので、CLDの振舞い仕様の形式化が容易になる。

CLDの変数 v_X は、暗黙の変数 X を伴い、その変化分、 δX を表すと考える。また、増加と減少を表す定性的な値 up と $down$ を要素として持つ集合 Q を導入する。 $Q \stackrel{\text{def}}{=} \{up, down\}$ である。さらに、不明あるいは変化なしを表す $none$ を導入して、拡張値の集合 $\hat{Q} \stackrel{\text{def}}{=} Q \cup \{none\}$ を得る。次に、 $\hat{Q}$ 上の関数 $rev: \hat{Q} \rightarrow \hat{Q}$ があり、 $rev(up) = down$, $rev(down) = up$, $rev(none) = none$ と定義する。

Polarity は遷移に伴う変数値の移送関係を規定する。変数値を読み出す関数を $\llbracket _ \rrbracket: V \rightarrow \hat{Q}$ とすると、 $\llbracket v_s \rrbracket \in Q$ に対して、

$$\llbracket v_d \rrbracket = \llbracket v_s \rrbracket \quad \text{if } \ell(v_s, +, v_d), \\ \llbracket v_d \rrbracket = rev(\llbracket v_s \rrbracket) \quad \text{if } \ell(v_s, -, v_d)$$

である。

CLNの値は Q であり、素朴には Q の要素に対応する2つの色を導入すれば良い。式関数 E は、第2.2節に示した変数値の伝播関係に限定する。ガード関数 G も導入した色に合わせて定義できる。

3.2.2 真の並行性と AMAN 戦略

第3.1節で述べたように、PT-nets および CP-nets といったペトリネットは真の並行性を表す。同時に、図3のPT-netsのマーキング遷移シナリオが示すように、無限個のトークンを保持するプレースを持つようなペトリネットも表現できる。この無限性は、当該ペトリネットの構造的な特徴から生じるが、その裏には、トークンの性質がある。また、無限性は、自動解析の観点から好ましくない。

ペトリネットのトークンは、いくつかの役割りを併せ持つ。ひとつは、並行性に関わる役割りで、トークンの動きが実行スレッドに対応する。複数トークンを遷移させることで、マルチスレッド計算を表現する。もうひとつの役割りは、資源を表すことである。トランジション発火によるトークン移動は、移動元で資源が消費され、また、移動先に資源が生成されることに相当する。

CLDは抽象的な表現であり、並行性は本質的であるが、

資源の概念は持たない。そこで、期待される並行度を表現するのに必要な数のトークンを複製すれば良いだろう。本稿では、これを、AMAN戦略 (as many as needed) と呼ぶ。AMAN戦略では、プレースが保持するトークンの個数は特別な意味を持たない。ペトリネットではプレースがトークンの多重集合であるのに対して、AMA戦略のもとではトークンの集合とする。プレース p の出力が n 個のトランジションにつながっている場合、 p がトークンを持つと、 n 個のトークンを複製することで、これら全てのトランジションを発火させる。この方法によると、図3の例のような衝突が生じることはない。

プレースをトークンの集合とすることで、資源としてのトークンというペトリネットの特徴的な性質を捨象したことになる。

3.2.3 非決定的な遅延

一般に、遅延は定量的な性質である (文献 [12] 第11章)。CLDの遅延因果リンクは、定量的なストック・フローモデルのストックを抽象化したと考える。ストックへの書き込みと、ストックからの読み出しという2つのイベントを区別する。ストックへの、これらのアクセスが、同時に生じることはない。そこで、CLDは変数移送の遷移が遅れると解釈している。一方で、CLDは定性的な性質に限定することから、遅延に定量的な値を与えない。

文献 [12] にあるように、遅延によって、可能な遷移系列、つまり、動的な振舞いが大きく影響される。ある因果リンク t_d に遅延注釈が付されている ($t_d \in \tau$) と、真の並行性の観点から発火可能な場合であっても、直ちに遷移発火することはない。

あるトランジション t が遅延注釈を持つ場合 ($t_d \in \tau$) と持たない場合 ($t_s \in \ell$) を比較する。遅延注釈を持つ場合、遷移列中の t_s が現れるであろう位置よりも、後の位置に t_d が現れる。また、その t_d の位置は、ひとつに定まらず、遅延の定量的な値によって変化する。一方、CLDは定量的な値を捨象していることから、 t_d が出現する位置を決めることができない。

遅延の抽象化として考えられることは、どこかに t_d が出現するという点であろう。そこで、ある上限 $d^{upper} \in \mathcal{N}$ を決め、マーキング・グラフを計算する際に、 $d < d^{upper}$ となる d を非決定的に選択する。この方法で得るマーキング・グラフは、 d の値によって異なる形状を示す。すべての可能性を考慮することは困難である。この非決定的な遅延の方法では、対象CLDの可能な全ての動的な振舞いを調べてはいない。つまり、過小近似になる。

4. 因果ループ・ネット

因果ループ・ネット (Causal Loop Nets, CLN) は有向2部グラフによるネット指向モデリング体系である。CLDの変数をCLNのプレースに、CLDの変数移送関係を表す

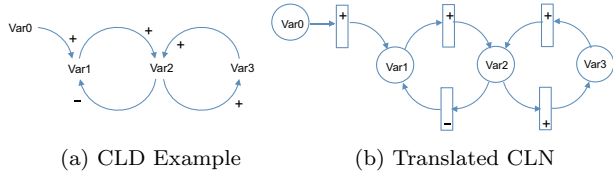


図 4 CLD and CLN

アークを CLN のトランジションに、各々対応させることで、CLD と CLN は構造的に同型になる (図 4 参照)。

4.1 CLN の形式定義

CLN は、トークンの集まりを $Token$ として、有向 2 部グラフ $D = \langle P, T, F, \tau, \nu, M_0 \rangle$ で定義される。

定義：因果ループ・ネット

- P : プレース (Places) の有限集合
- T : トランジション (Transitions) の有限集合
- F : フロー関係 $F \subset (P \times T) \cup (T \times P)$
- τ : 極性注釈 $T \rightarrow Polarity$
- ν : 遅延注釈 $T \rightarrow \mathcal{N}_0$
- M_0 : 初期マーキング $P \rightarrow 2^{Token}$

遅延注釈がない場合、 $\nu(t) = 0$ とする。CLN のトークン (Token) は基本トークンか遅延トークンのいずれかである。

基本トークン BasicToken = $\{\uparrow, \downarrow\}$

遅延トークン DelayToken = $\{\uparrow_d, \downarrow_d\} (d \in \mathcal{N})$

便宜上 $_$ を導入して、

Token = BasicToken $\cup$ DelayToken $\cup \{_$

とする。ここで、 $\uparrow$ は up に、 $\downarrow$ は $down$ に、 $_$ は $none$ に各々対応する。遅延トークンの添字 d は、仮想的な遅延時間を表す。便宜上、 $\uparrow_0 = \uparrow$ と $\downarrow_0 = \downarrow$ と約束する。また、プレースをトークンの集合としたことから、マーキングは、 $M: P \rightarrow 2^{Token}$ である。

マーキング M に対して、遷移発火可能なトランジションの集まりは、発火可能トランジションはトークンの有無を調べれば求めることができる。 $e \in \mathcal{N}_0$ として、

$\mathcal{T}(M) = \{ t \in T \mid \bigwedge_{p \in \bullet t} M(p) \cap \{\uparrow_e, \downarrow_e\} \neq \emptyset \}$

$\mathcal{T}(M)$ は複数のトランジションを含むことがある。複数のトランジションが同時発火することを示し、真の並行性を表す。

4.2 マーキングの変化

CLN の動的な振舞いは、マーキング遷移 ($M \xrightarrow{\mathcal{T}(M)} M'$) から構築されるマーキング・グラフで与えられる。

マーキング M に対して、発火可能なトランジションの集まり $\mathcal{T}(M)$ が定まった時、すべてのプレース p ($p \in P$) について、以下の式で、 $M'(p)$ を表すことができる。ただし、 $\oplus$ は、オーバーライドを表すとした。また、2つの関数 $\bullet \Delta$ と $\Delta \bullet$ は $F \rightarrow 2^{Token}$ である。ここで、 $F \subset (P \times T) \cup (T \times P)$

だったことに注意して欲しい。

$$M'(p) = M(p) \oplus \left(\bigcup_{t \in \mathcal{T}(M)} \bullet \Delta(p, t) \cup \bigcup_{t \in \mathcal{T}(M)} \Delta \bullet(t, p) \right)$$

関数 $\bullet \Delta(p, t)$ は、 $M(p)$ から全てのトークンを削除し、残り時間を 1 だけ減じた遅延トークンを残す。

$$\bullet \Delta(p, t) = \bigcup_{x \in M(p)} Decr(x)$$

補助関数 $Decr(\bullet)$ ($Token \rightarrow 2^{Token}$) は、関係式 $\uparrow = \uparrow_0$ および $\downarrow = \downarrow_0$ を利用した。

$$Decr(\bullet_e) = \begin{cases} \{ \bullet_{e-1} \} & \text{if } e > 0 \\ \emptyset & \text{if } e = 0 \end{cases}$$

関数 $\Delta \bullet(t, p)$ は因果リンク $\tau(t)$ のポラリティにしたがってトークンを追加する。ここで、煩雑さを避けることから、 $\bigcup_{p' \in \bullet t} M(p')$ を、 $M(\bullet t)$ と略記した。

$$\Delta \bullet(t, p) = \bigcup_{x \in M(\bullet t)} Xfer(x)$$

補助関数 $Xfer(\bullet)$ ($Token \rightarrow 2^{Token}$) の定義では、因果リンクが遅延注釈を持たない場合、 $\nu(t) = 0$ とした。

$$Xfer(\bullet) = \begin{cases} \{ \bullet_{\nu(t)} \} & \text{if } \tau(t) = + \\ \{ rev(\bullet_{\nu(t)}) \} & \text{if } \tau(t) = - \end{cases}$$

上記の遷移関係 $M \xrightarrow{\mathcal{T}(M)} M'$ で構成されるマーキング M は、あるプレース p について、 $\{p \mapsto \uparrow, p \mapsto \downarrow\} \in M$ となることがある。つまり、このようなマーキング M はプレース p に対して、2つの値 $\uparrow$ と $\downarrow$ を同時に割り当てることになり、トークン衝突が起こる。このようなトークン衝突を生じるプレースの集まりを P_x ($P_x \subseteq P$) とする。

$$P_x \stackrel{def}{=} \{ p \mid \{\uparrow, \downarrow\} \subseteq M(p) \}$$

トークン衝突は 2つの可能性があることを示す。そこで、いずれかが非決定的に生じると考える。一方だけを含む (複数の) マーキングを生成し、各々を起点とする新たなマーキングの遷移関係を求める。トークン衝突を解消したところで、この非決定分岐を終えれば良い。トークン衝突のプレースを含むマーキング M に対して、非決定分岐を適用して新たなマーキングを求める正規化関数 Nr を次のように定義する。

$$Nr(M) = \begin{cases} Nr(M[p \mapsto D(p, \uparrow)]) & \text{if } p \in P_x \\ \cup Nr(M[p \mapsto D(p, \downarrow)]) \\ M & \text{if } P_x = \emptyset \end{cases}$$

ただし、 $D(p, \bullet) \stackrel{def}{=} (M(p) \cap \{\uparrow_d, \downarrow_d\}) \cup \{\bullet\}$ とした。

マーキング M の遷移先は、トークン衝突の解消を考慮すると、マーキングの集まり $Post(M)$ となる。

$$Post(M) = \{ M' \mid M \xrightarrow{\mathcal{T}(M)} M'' \wedge M' \in Nr(M'') \}$$

初期マーキング M_0 を n_0 として, $M' \in \text{Post}(M)$ について $M \Rightarrow M'$ を求めると状態遷移グラフを得る.

因果ループ図 D に対して, D に対応する CLN から得られる状態遷移グラフを G^D とすると, $G^D \stackrel{\text{def}}{=} \langle \text{Node}, n_0, \text{Edge} \rangle$ と表せる.

5. 検査性質の解析

5.1 検証問題

第4節で述べたように, CLD の変数と CLN のプレースを同一視する. CLN の状態遷移グラフ G^D のノード $n(\in \text{Node})$ はマーキングなので, n における変数 v の値 $\llbracket n.v \rrbracket$ は $M(v)$ であって, $\llbracket n.v \rrbracket \in \hat{Q}$ と考える. ただし, DelayToken は確定した値を持たないことから, $_$ と同一視し none とする.

開始ノード $n_0 (\in \text{Node})$ を端点とするノードの遷移列 σ_i の全体 $\text{Trace}^D \stackrel{\text{def}}{=} \{\sigma_i\}$ を考える. 遷移列 σ_i の r 番目のノードを $\sigma_i(r)$ とする時, $\sigma_i(0) = n_0$ である. また, $\sigma_i(r \dots s)$ は, 遷移列 σ_i の r 番目から s 番目のノードからなる部分遷移列をさす.

検証問題は, Trace^D の要素に, 検査性質 Φ を満たす遷移列が存在するか否かを調べることである.

$$\exists \sigma_i \in \text{Trace}^D : \Phi(\sigma_i) \text{ mod } \sim$$

ここで, 模倣関係 $\sim$ は, 次の $\sim_1$ あるいは $\sim_2$ である. 模倣関係 $\sim_1: \hat{Q}^N \longleftrightarrow Q$ は $q \in Q$ に対して, $q(q|\text{none})^{N-1} \sim_1 q$ とする. 一方, 模倣関係 $\sim_2: \hat{Q}^N \longleftrightarrow \hat{Q}$ は $q \in \hat{Q}$ に対して, $q^N \sim_2 q$ とする.

検証問題の定式化に, 模倣関係を考慮した理由を以下に示す. 検査性質を遷移列の要約情報として表現したい. 遷移列 σ_i は $\hat{Q}$ の列である. $q \in \hat{Q}$ は具体的な値を表さないことから, N 個の連続する q^N を q と同一視しても, 定性的な傾向は変わらない. 模倣関係 $\sim_1$ は, 要約情報として Q の要素を考える場合に相当する. 一方, $\sim_2$ は, $_$ も考慮すべき要素とする場合である.

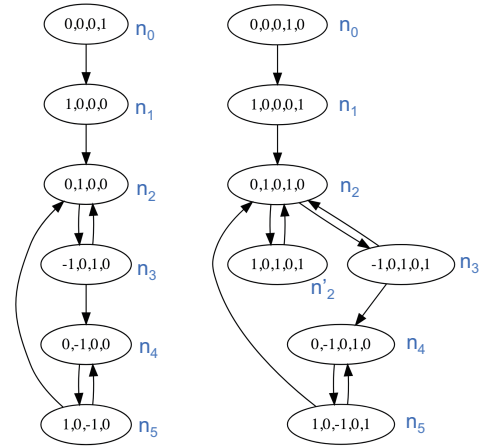
5.2 検査性質

検査性質 P の具体的な形を考える. 自然数 k ($0 < k$) に対して, 有限マップ $f: [1, k] \rightarrow Q$ が CLN 変数に期待する値変化を表す時, f を要約関数と呼ぶ. 要約関数は長さ k の列を表す.

状態遷移列 σ_i に対して, r 番目の状態を起点として着目する変数 v が要約関数 f と等価な値変化を示すことを $\varphi_0(i, r, v, f)$ と表す. k を要約関数の長さ $|f|$ とする.

$$\varphi_0(i, r, v, f) = \forall X \in [1, k]. \exists s. (\sigma_i(r \dots s).v \sim f(X))$$

ある遷移列 σ_i があって, 長さ k の列 $f(X)$ ($\in Q^k$) と模倣関係にある部分遷移列 $\sigma_i(r \dots s)$ が存在する場合に真となる. 典型的な検査性質を φ_0 を用いて表すことができる.



(a) Original CLD (b) Continuous Terminal

図5 Transition Graphs of CLD in Figure 4

(1) 遷移列 σ_i の存在

$\varphi_1(v, f) = \exists i, r : \varphi_0(i, r, v, f)$ は, ある長さの遷移列で, 着目する変数 v が要約関数 f と模倣関係になる状態遷移列 σ_i が存在するか否かを調べる. 存在する時, 真となる.

(2) 目標変数の変化

説明変数 v^m が要約関数 f^m と等価な振舞いを示す時, その遷移列における指定の目標変数 v^ℓ ($\ell \neq m$) の値変化 g^ℓ を求める. $\exists i, r : \varphi_0(i, r, v^m, f^m)$ に対して $g^\ell(X) = \exists s. \sigma_i(r \dots s).v^\ell$

6. 試作ツールによる解析

テキスト表現の CLD を入力し, CLN に変換した後, 状態遷移グラフを生成するデモ・ツールを Scala で作成した. グラフ表現として dot 形式を採用し, Graphviz^{*4} によって描画する.

図5(a)は, 図4に示した CLD の状態遷移グラフ例である. 一方, 図5(b)は, 後に議論するように, 変数 Var0 の値が継続的に, $\uparrow$ となる場合の状態遷移グラフである.

図5(a)は初期マーキングを $M_0 = \{\text{Var0} \mapsto \uparrow\}$ とした場合である. ノードは $\langle \text{Var1}, \text{Var2}, \text{Var3}, \text{Var0} \rangle$ の形式で, 4変数の値を表す. 値1は $\uparrow$, -1は $\downarrow$, 0は $_$ を表す.

入力CLDのテキスト表現は以下の通りである. Var0を4とした他, VarXはXに対応する. `initMark((4, Set(1)))` は, 初期マーキングを示す.

```
var netF4 = newclpn ++ (
  4 ->+ 1,
  1 ->+ 2,
  2 ->- 1,
  2 ->+ 3,
  3 ->+ 2
) initMark((4, Set(1)))
```

次に, 性質解析の例を示す. 模倣関係を $\sim_1$ として,

^{*4} <http://viz-js.com>

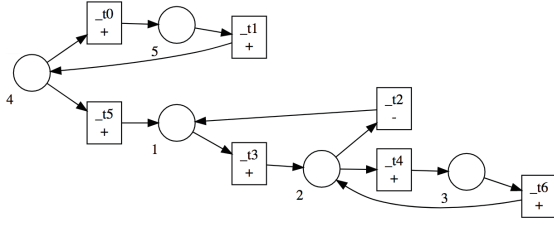


図 6 Continuous Terminal

$\varphi_1(\text{Var}2, \downarrow \uparrow)$ が満たすノードの列 $\sigma_i(5..)$ を長さが短い列から幾つか示す.

$$\begin{aligned}\sigma_1 &= n_4 \ n_5 \ n_2 \ n_3 \\ \sigma_2 &= n_4 \ n_5 \ n_4 \ n_5 \ n_2 \ n_3 \\ \sigma_3 &= n_4 \ n_5 \ n_2 \ n_3 \ n_2 \ n_3 \\ \sigma_4 &= n_4 \ n_5 \ n_4 \ n_5 \ n_2 \ n_3 \ n_2 \ n_3\end{aligned}$$

この時, $\text{Var}3$ は長さ 8 未満として,

$$\begin{aligned}(\sigma_1(5..8)).v^{\text{Var}3} &= - \downarrow - \uparrow \\ (\sigma_2(5..10)).v^{\text{Var}3} &= - \downarrow - \downarrow - \uparrow \\ (\sigma_3(5..10)).v^{\text{Var}3} &= - \downarrow - \uparrow - \uparrow \\ (\sigma_4(5..12)).v^{\text{Var}3} &= - \downarrow - \downarrow - \uparrow - \uparrow\end{aligned}$$

これらと $\sim_1$ 模倣になる要約関数 $g^{\text{Var}3}$ は $\sigma_i(6)$ から始まる遷移列で $\downarrow \uparrow$ である. 一方, $\sim_2$ 模倣の場合は $\sigma_i(5)$ から始まる正規表現 $(- \downarrow)^+ (- \uparrow)^+$ で表せる. 要約関数 $f^{\text{Var}2}$ の指定は $\sim_1$ 模倣が容易である一方, 要約関数 $g^{\text{Var}3}$ の表現としては $\sim_2$ 模倣のほうがわかりやすい.

さて, 第 4.2 節の解釈では, 端点となる変数値は初期マーキングに影響を与えるだけである. 以降は, $-$ となり, 発火に関係しない. 一方, 最初の遷移発火以降も, 継続して同じ初期値の移送を表したいこともある. このように, 端点が持続性を持つ場合は, シェドウとなる変数を導入して小さいループ構造を導入すれば良い.

具体例として, 図 4 の変数 $\text{Var}0$ を考える. 変数 $\text{Var}0$ のシェドウ変数 $\text{Var}0'$ を導入し, 2 つの移送関係,

$$\ell(\text{Var}0, +, \text{Var}0'), \quad \ell(\text{Var}0', +, \text{Var}0)$$

を追加する. この時, CLN のダイアグラム表現は, 図 6 のようになる*5. 図 5(b) は, 図 5(a) と同じ初期マーキング $M_0 = \{\text{Var}0 \mapsto \uparrow\}$ から得られた状態遷移グラフを示す. 次に, 上記の解析例と同様に, $\varphi_1(i, \text{Var}2, \downarrow \uparrow)$ を満たす状態列を調べる.

$$\begin{aligned}\sigma_1 &= n_4 \ n_5 \ n_2 \ n'_2 \\ \sigma_1 &= n_4 \ n_5 \ n_2 \ n'_2 \ n_2 \ n_3 \\ \sigma_2 &= n_4 \ n_5 \ n_4 \ n_5 \ n_2 \ n'_2 \\ \sigma_3 &= n_4 \ n_5 \ n_2 \ n'_2 \ n_2 \ n_3\end{aligned}$$

*5 Graphviz で生成した.

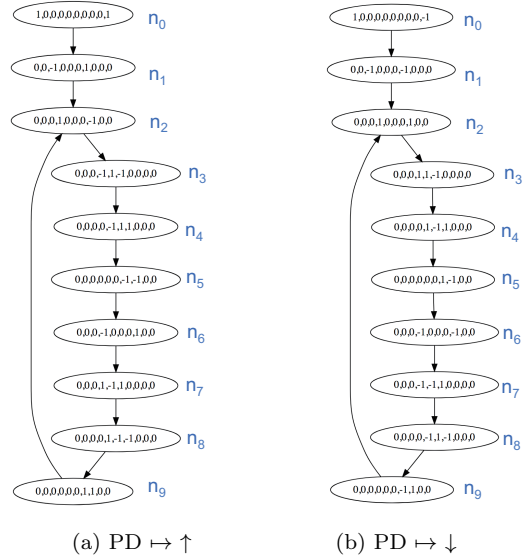


図 7 Two Transition Graphs of CLD in Figure 2

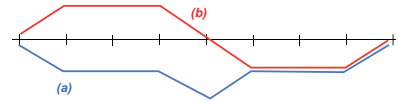


図 8 Qualitative Graphs

などの状態列が存在する.

状態列を正規表現で表すと良いだろう. 先に求めた図 5(a) の場合は $((n_4 n_5)^+ (n_2 n_3)^+)^+$ となる. 図 5(b) の場合は $((n_4 n_5)^+ (n_2 n'_2 \mid n_2 n_3)^* (n_2 n_3)^+)^+$ と表せる.

要約関数 $g^{\text{Var}3}$ の表現を, $\sim_2$ 模倣で調べる. $(- \downarrow)^+ (- \uparrow)^+$ となって, 図 5(a) の場合に一致する. 一方, $g^{\text{Var}1}$ は, 図 5(a) の場合, $(- \uparrow)^+ (- \downarrow)^+$ であるが, 図 5(b) の場合, $(- \uparrow)^+ (- \uparrow \mid - \downarrow)^* (- \downarrow)^+$ となる. この例では, $\text{Var}0$ が持続的か否かで, $\text{Var}1$ は影響を受けるが, $\text{Var}3$ は影響を受けないことがわかる.

図 7 は図 2 の CLD に対する状態遷移グラフである. トークン衝突が起こらないので構造が簡単なことがわかる. 2 つの状態遷移グラフは, 異なる初期マーキングを与えた場合に対応し, 図 7 の初期マーキングは, 各々,

$$\begin{aligned}M_0^{(a)} &= \{\text{CT} \mapsto \uparrow, \text{PD} \mapsto \uparrow\}, \\ M_0^{(b)} &= \{\text{CT} \mapsto \uparrow, \text{PD} \mapsto \downarrow\}\end{aligned}$$

である. $\sigma_i(3)$ から始まる遷移列を比較すると, g^{AB} は次のように表現できる.

$$\begin{aligned}\langle g^{\text{Figure}2}, 3, M_0^{(a)} \rangle &\models \downarrow - \downarrow \uparrow - - \uparrow \\ \langle g^{\text{Figure}2}, 3, M_0^{(b)} \rangle &\models \uparrow - - \downarrow \downarrow - - \uparrow\end{aligned}$$

CLD では, $\uparrow$ および $\downarrow$ を, 各々 δv が > 0 および < 0 とする. 今, 仮に, $-$ の解釈を, $\delta v = 0$ とすると, g^{AB} を定性的なグラフ (図 8) として表すことができる. 初期値の違いによって, 目標変数の変化の仕方が異なることがわかる.

7. 考察

CLNについて、ペトリネットの抽象化という観点から考察する。第3.2節で論じた、真の並行性に関わる抽象化が、CLNの特徴である。プレース内トークンの個数情報を、トークンの有無に抽象化することから、プレースを多重集合ではなく通常の集合とした。また、遷移発火に必要な並行度を得ることから、AMAN戦略でトークンを複製した。一方、集合であっても異なる種類のトークンをプレースに保持できるので、トークン衝突の状況が生じることがある。この衝突は複数の状況が可能であることを示唆すると解釈し、非決定遷移によって状態遷移グラフを構成した。ペトリネットでは、プレースはトークンの多重集合であることから、CLNのようなトークン衝突は生じないが、有界性を満たさないことがある(図3)。

遅延に関しては、CLDは明確な定義を与えていない。遅延の解釈が難しいことは、ダイナクミスを定性的に分析することから生じる。ひとつのCLDが複数の遅延注釈を持つ時、各々の遅延注釈を独立に扱って、各々1単位だけ遷移発火を中断する方法が、実際的かもしれない。

本稿で考察した性質は経路の存在を確認することだった。CLDは定性的な抽象化をしていることから、安全性の検査は、あまり有用な知見を与えない。本稿の要約関数は、模倣関係 $\sim_1$ の下でのパターン問い合わせによる経路抽出である。他の性質の検査も興味深い。たとえば、Maude [13]は真の並行性を持つ形式仕様言語である。CLNをMaudeでエンコードすれば、性質検証の方法として、MaudeのLTLモデル検査等の自動解析法を利用することができるだろう。

8. おわりに

因果ループ図の形式的な意味を与える有向2部グラフによるネット指向形式体系の因果ループ・ネットを導入した。

ペトリネットと同様に真の並行性を表すが、ペトリネットで表現が難しい動的振舞いを表現することができる。一方、遅延注釈の解釈は十分に納得性のある方法になっていない。

謝辞 本研究は第2著者が、2018年1月～4月、NII国際インターンシップ生として実施した。本研究の一部はJSPS科研費JP17H01726の助成を受けた。

参考文献

- [1] K. Jensen : *Coloured Petri Net*, Springer 1996.
- [2] K. Jensen, L.M. Kristensen, and L. Wells : Coloured Petri Nets and CPN Tools for modelling and validation of concurrent systems, *Int. J. Softw. Tools Technical Transfer*, no.9, pp.213-254, 2007.
- [3] N.G. Leveson : *Engineering a Safer World: Systems Thinking Applied to Safety*, The MIT Press 2011.
- [4] M.W. Maier : Architecting Principles for System of Systems, *Systems Engineering*, 1(4), pp.267-284, 1998.
- [5] T. Murata : Petri Nets: Properties, Analysis and Applications, In *Proc. IEEE*, 77(4), pp.541-580, 1989.
- [6] 村田忠夫 : ペトリネットの解析と応用, 近代科学社 1992.
- [7] 中島震 : CPS: そのビジョンとテクノロジー, 研究 技術 計画, 32(3), 研究・イノベーション学会 2017.
- [8] C. Perrow : *Normal Accidents: Living with High-Risk Technologies*, Princeton University Press 1999.
- [9] M.E. Porter and J.E. Heppelmann : How Smart Connected Products are Transforming Competition, *Harvard Business Review*, 92(11), pp.64-88, 2014.
- [10] M. Shaw : Whither Software Engineering Education? An invited talk at IEEE CSEE&T 2011.
- [11] S.P. Shepherd : A Review of System Dynamics Models Applied in Transportation, *Transportmetrica B: Transport Dynamics*, 2(2), pp.83-105, 2014.
- [12] J.D. Sterman : *Business Dynamics: Systems Thinking and Modeling for a Complex World*, Irwin McGraw-Hill 2000.
- [13] M.-O. Stehr, J. Meseguer, and P.C. Olveczky : Rewriting Logic as a Unifying Framework for Petri Nets, *Unifying Petri Nets*, pp.250-303, 2001.
- [14] J.M. Wing : Cyber-Physical Systems, *Computing Research News*, 21(1), p.4, 2009.