計測自動制御学会論文集
Vol.57, No.8, 367/377（2021）

# PLCの動作順序を監視するホワイトリストの自動生成手法

藤田 真太郎*・澤田 賢治*
新誠一*・細川 嵩**

# On Auto-generation Method of Whitelist for PLC Operation Sequence

Shintaro Fujita*, Kenji Sawada*,
Seiichi Shin* and Shu Hosokawa**

Since the damage from cyber-attacks increases, there is an urgent need to research and develop security countermeasures for control systems. In the control system, the controller is an important device. This paper then considers a whitelisting system that models the normal operation sequence of a PLC (Programmable Logic Controller) and detects deviations from the model as abnormal. We propose three methods to auto-generate the whitelisting system by LD (Ladder Diagram): Petri net model generation, constraint condition derivation, and LD conversion. The first method generates Petri net models form SFCs (Sequential Function Charts) that are compatible with LDs. The second method derives whitelist conditions from Petri net models to check whether PLC performs the correct operation sequence. The third method implements the whitelist conditions into LD. The auto-generated whitelisting system enables us to monitor the state transitions of the PLC programs. Further, this paper carries out an experimental validation of the methods using a testbed system.

**Key Words:** whitelist, control system security, PLC, petri net

## 1. はじめに

2010年のStuxnet以降，電力システムや工場などの産業用制御システム (ICS) に対するサイバー攻撃やウイルス感染などの脅威の報告数が増加している．2015年のウクライナの電力会社3社に対するサイバー攻撃[1]や2017年にはランサムウェアによる世界的な被害が発生している[2]．2020年4月にはイスラエルの水処理施設の制御器がサイバー攻撃の標的となり被害が発生している[3]．こうしたサイバー攻撃による被害を防ぐために，制御システムの異常検知や異常発生時の被害拡大防止の研究開発が行なわれている．たとえば，制御器のファームウェアにランタイム動作監視機能を追加し制御プログラムの不正変更を監視する手法[4]や制御プログラムの実行時間を監視する異常検知手法[5]が提案されている．

制御システムのセキュリティ対策では，技術開発の信頼性確保のため制御器やフィールド機器を含む実環境を想定したテストベッドによるセキュリティ技術検証が必要となる．著者らは制御器としてPLC (Programmable Logic Controller) を使用した制御システムのテストベッド構築やPLCのためのホワイトリスト式検知技術[6),7)]を開発している．PLCは，JIS B3501:2004で規格化されており，規格ではプログラマブルコントローラと呼称されている．PLCのホワイトリストは，PLCのI/Oの正しい振る舞いをリストとして登録し，リスト外のI/Oの振る舞いを異常と検知する．文献6)の手法では，I/Oの正しい動作順序を離散事象システムとしてペトリネットでモデル化し，モデルをPLCの制御プログラム形式の1つであるラダープログラム（LD）に変換している．LDに変換されたモデルは正しい振る舞いを記録するリスト機能と現在の振る舞いが正常かを判定する照会機能を発揮する．LDを利用できるPLCならば，ファームウェアやハードウェアの変更や改造の必要もなくホワイトリストを実装できる．そのため，ホワイトリストのために機器を増設することなく，フィールド機器を制御しているPLCにホワイトリストを実装することが可能である．

先行研究6)は正しい振る舞いの条件やペトリネットモデルの作成に適用対象の制御システムの仕様書を必要とする．一方で，PLCのプログラムは制御現場で定期的に保守点検を受け，初期仕様から異なっている場合がある．1つの理想は，PLC内に実装されている制御プログラムからホワイトリストを自動生成することである．そこで，本論文ではPLCのホ

* 電気通信大学　調布市調布ヶ丘1–5–1
** 制御システムセキュリティセンター
仙台市青葉区荒巻字青葉6–6
* The University of Electro-Communications, 1–5–1 Chofugaoka, Chofu
** Control System Security Center, 6–6 Aoba, Aramaki, Aoba-ku, Sendai
（Received September 7, 2020）
（Revised March 16, 2021）

 

ワイトリストの自動生成手法を提案することを目指す．自動生成では先行研究 6) の「制御システムの仕様書から I/O の正しい振る舞いにかかわる制約条件を導く」という設計プロセスを採用できない．この課題に取り組むために，本論文ではホワイトリストの設計プロセスを変更し，PLC の制御プログラムからペトリネットモデルを生成し（ペトリネットモデル生成），モデルから I/O の正しい振る舞いに関する制約条件を導出する方法（制約条件の導出）を考える．さらに，導出された制約条件を LD として PLC に実装する方法（制約条件の LD プログラム化）を新たに提案する．

1 つ目のペトリネットモデル生成の実現のために，LD と記述の相互変換が可能な Sequential Function Chart（SFC）を利用し，SFC プログラムからペトリネットモデルを生成することを考える．ホワイトリストは制御プログラム入力により異常が発生するかを判定する．そこで，2 つ目の制約条件の導出では，SFC（もしくは LD）プログラムの入力情報に基づき正常か異常かを判定できる制約条件を入力と状態を有するペトリネットモデルから導出する．3 つ目の制約条件の LD プログラム化では，制約条件を LD で記述可能な論理演算で表わし，リスト機能と照会機能を LD で実現する．ペトリネットモデルの設計プロセスが先行研究 6) と異なっているため，LD プログラム化に関しても新たに提案する．提案手法によって，SFC プログラムとその構造情報から，PLC にホワイトリストを実装するための LD の生成が可能となり，求まるホワイトリストは PLC の I/O の動作順序を監視することが可能となった．また，提案手法を用いて文献 7) のテストベッドのホワイトリストを生成し，実機検証を行なった．

**表記：** $\mathbf{0}_n$ はすべての要素が 0 の $n$ 次元のベクトルである．0-1 変数の集合を $\{0,1\}$ で表わす．$\{0,1\}$ 上の $n \times m$ 行列の集合を $\{0,1\}^{n\times m}$ で表わす．$\boldsymbol{A}^T$ は行列 $\boldsymbol{A}$ の転置行列を表わす．ベクトル $\boldsymbol{x}$ に対して $\|\boldsymbol{x}\|_1$ は 1 ノルムを表わす．集合 $\mathbb{S}$ に対して，$|\mathbb{S}|$ は要素数を表わす．2 つの集合 $\mathbb{S}_1$，$\mathbb{S}_2$ に対して．$\mathbb{S}_1 \times \mathbb{S}_2$ は直積集合を表わす．

## 2. 制御システムとテストベッド構成

電力などのインフラや工場での一般的な制御システムの構成例を **Fig. 1** に示す．

制御システムは，HMI (Human Machine Interface) やエンジニアリング PC などの計算機，ネットワーク機器，PLC や DCS (Distributed Control System) などの制御器，センサやアクチュエータなどのフィールド機器で構成される．また，インターネットに接続された制御システムでは遠隔地からの操作・監視を受けるための端末が存在する場合がある．文献 7) では，HMI，ネットワーク機器，制御器，フィールド機器に注目したテストベッドを構築している．このテストベッドは，ベンダー非依存にするために制御器に OpenPLC [8),9)] と呼ばれるソフトウェア PLC を用いている．

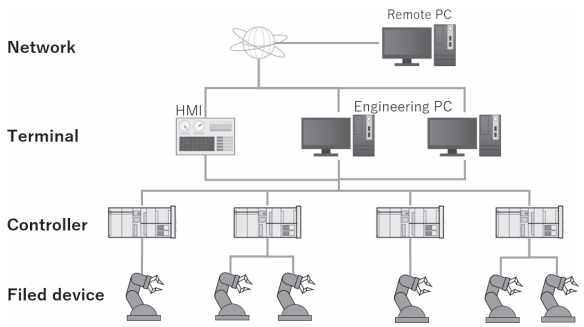


**Fig. 1** General control system configuration

OpenPLC は，Thiago Rodrigues Alves によって開発されているオープンソースソフトウェアである．OpenPLC は，**Fig. 2** に示すように，OpenPLC をインストールした計算機と I/O デバイスを利用することで PLC としての機能を模擬する．

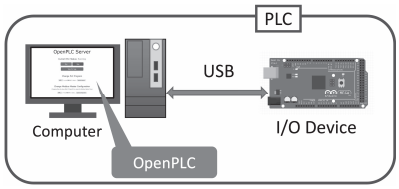


**Fig. 2** PLC simulation with OpenPLC

OpenPLC の推奨開発環境である PLCopen Editor では IEC 61131-3 で規格化された 5 種類のプログラミング言語 (LD 言語，ST 言語，FBD 言語，IL 言語，SFC 言語) で開発することができ，各言語を ST 言語に変換できる．ST 言語で書かれた制御プログラムをブラウザ経由で OpenPLC にアップロードすることで，制御プログラムの書き込みを行なう．I/O デバイスは計算機からの指令に基づいて，フィールド機器を制御する．いくつかのハードウェア [10)~15)] が I/O デバイスとして使用可能である．

テストベッドの構成を **Fig. 3** に示す．OpenPLC と HMI は異なる Windows 10 の計算機上で動作している．2 つの計算機は Modbus/TCP 通信によって接続されている．このテストベッドでは攻撃パターンとして，攻撃者が Modbus/TCP 通信のネットワークに攻撃者端末を物理的に接続したり，HMI をマルウェアに感染させたりする状況を想定している．これにより，攻撃者は通信の傍受，Modbus/TCP 通信のパケット送信，HMI の不正操作が可能となる．そして，ホワイトリストによる異常検知が対象とする攻撃は，PLC の動作順序に影響を与える攻撃である．その攻撃は，リプレイ攻撃や正常コマンドの乗っ取りなどによって，PLC に送信される Modbus/TCP 通信のパケットの順序を乱す攻撃である．リプレイ攻撃は傍受した通信の再送によって，正常コマンドの乗っ取りは HMI の不正操作によって行なうことができる．

また，**Fig. 4** に I/O デバイスとして使用した Arduino MEGA 2560，**Fig. 5** にフィールド機器として使用したロボットアームを示す．

Arduino MEGA 2560 にはロボットアームのほかに，制御

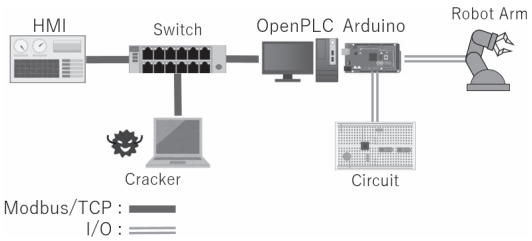


**Fig. 3**　Testbed configuration

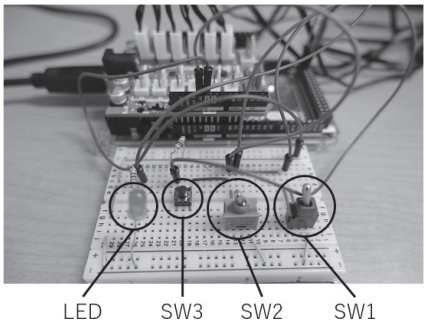


**Fig. 4**　Arduino MEGA 2560 and electronic circuits

**Fig. 5**　Robot arm

システムの操作のためのスイッチが 3 個と LED が 1 個接続されている．スイッチは Fig. 4 の右側から順に，ロボットアームの電源スイッチ (SW1), 動作の開始･停止スイッチ (SW2), 動作のリセットスイッチ (SW3) である．

本システムはファクトリーオートメーションにおける部品の搬送制御システムを模擬したもので，ロボットアームによるブロックの搬送作業を行なう．本システムには，ロボットアームの起動，ブロックの搬送，搬送作業の一時停止，ロボットアームの初期位置復帰の 4 つの機能がある．この 4 つの機能は SW1, SW2, SW3 の操作によって切換えられる．SW1 が ON になったときにロボットアームが起動し，SW2 の ON/OFF によってブロック搬送作業の開始/一時停止を制御する．また，搬送作業が一時停止状態にあるときに SW3 を ON にすることでロボットアームを初期位置に復帰させる．

テストベッド上での先行研究 6) のホワイトリスト式検知技術の検証については文献 7) を参照されたい．

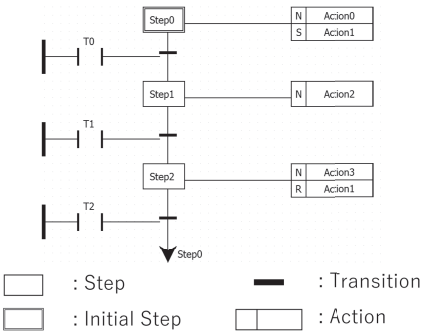


**Fig. 6**　Example of SFC

## 3. SFC

本論文では，SFC を用いて PLC の制御プログラムを作成する．プログラムを状態遷移図の形式で記述する SFC はそれ自身が制御システムの 1 つのモデル化である．**Fig. 6** に SFC の例を示す．SFC は主に，Step, Initial Step, Transition, Action の 4 つの要素によって構成されている．

Step は Action と Transition が接続されており，アクティブと非アクティブの 2 つの状態をもっている．初期状態は非アクティブである．非アクティブの状態では Step に接続された Action は実行されず，Transition の動作によって状態がアクティブに変化したときに Action が実行される．Initial Step は Step と同じ動作をするが，初期状態はアクティブである．また，Initial Step は 1 つの SFC のプログラムに対して必ず 1 つだけ存在する．

Transition の接続元と接続先には Step または Initial Step が接続されている．また Transition には発火条件と呼ばれる条件式が対応付けされており，発火条件が True かつ接続元の Step または Initial Step の状態がアクティブのとき発火と呼ばれる動作を行なう．発火とは，発火した Transistion の接続元の Step または Initial Step の状態を非アクティブにし，接続先の Step または Initial Step の状態をアクティブにする動作である．

Action は実行する動作と修飾子の 2 つの情報をもつ．実行する動作は LD，ST，FBD，IL によって記述された個別プログラムである．修飾子は N，R，S などの記号のことであり，動作の実行タイミングを表わしている．各修飾子の意味は，N は Step がアクティブである間のみ動作を実行，R は Step がアクティブになったとき動作を停止，S は Step がアクティブになったとき動作を開始である．

ロボットアーム制御システムの SFC プログラムを **Fig. 7** に示す．SW1, SW2, SW3 はそれぞれ変数 r_power_switch, r_switch, r_reset_switch に対応付けされており，スイッチが ON のときに各変数の値が True，OFF のときに False になる．また，変数 isTargetAngle はロボットアームが目標位置に到達したときに True となる変数である．さらに，Step0 はテストベッドの初期状態，Step1 はロボットアームの起動

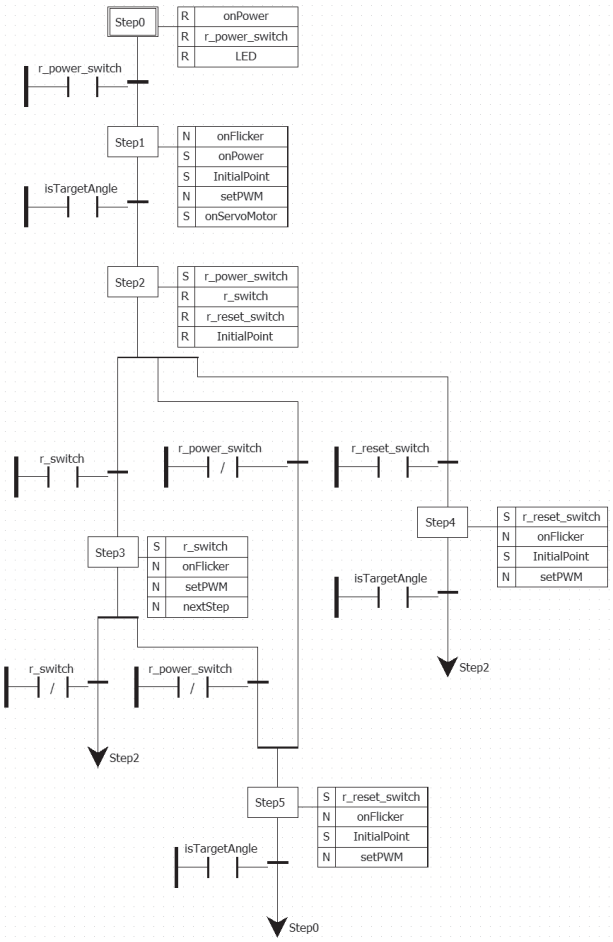


**Fig. 7** SFC of testbed

中，Step2 はロボットアームの起動完了又は搬送の一時停止，Step3 はブロック搬送状態，Step4 と Step5 はロボットアームの初期位置復帰中の状態をそれぞれ表わしている．

## 4. ホワイトリストと自動生成

### 4.1 ホワイトリスト

ホワイトリスト式検知技術は通常動作をあらかじめリストとして登録し，リストに登録されていない動作をすべて異常と判断する．本来のホワイトリストには，動作を登録しておくリスト機能，リストと動作を比較して異常を判断する照合機能，異常動作を防ぐ抑制機能の 3 つの機能がある．ただし，ICS では異常動作の抑制により機器の緊急停止が発生すると，かえってシステムに深刻な被害が発生する場合があり，抑制機能による動作の停止手順はシステムごとに設計する必要がある．したがって，本論文が対象とするホワイトリスト式検知技術の自動生成はリスト機能と照合機能のみである．制御システムの通常動作とは，通信コマンドの内容やセンサ，アクチュエータの動作順序がその設計仕様から逸脱しないものである．本技術の利点は，リスト更新が通常動作の変更タイミングに限定できる，通常動作を阻害するゼロデイ攻撃を検知できる，リスト照合時のシステム負荷が低い，である．

文献 6) はこれを PLC 上で実現している．文献 6) の手法では I/O（センサ，アクチュエータ）の正しい動作順序をリストとして定義する．この動作順序をペトリネット[16]でモデル化し，ペトリネットの遷移と制約条件を PLC の制御プログラムである LD に変換する．この LD は I/O が正しい実行順序より外れたら異常が発生したと判断するため，われわれはラダー式ホワイトリスト，もしくはホワイトシーケンスと呼称している．LD は PLC の代表的な制御プログラムであるため，ハードウエアやファームウエアの改造をすることなく，PLC にホワイトリスト機能を実装できる．

PLC の動作順序を監視するホワイトリストを自動生成するために，リスト機能として LD による正しい動作順序の表現方法と，照合機能として LD によるリスト照合手法を先行研究 6) と同様にペトリネットに基づき提案する．この自動生成を実現するためには，以下に述べる 3 つの課題がある．1 つ目の課題は，PLC に実装されている制御プログラムからの動作順序の取得方法と動作順序のペトリネットモデル変換である．2 つ目の課題は PLC の正常な動作順序にかかわる制約条件をペトリネットモデルから導出することにある．ペトリネットモデルは制御プログラムの入力（LD なら接点，SFC なら Transition）と入力結果（LD ならコイル，SFC なら Step）により正常か異常かを検知できるが，LD による照会機能の実装には LD もしくは SFC の入力情報のみで制約条件を定めなければならない．3 つ目の課題は PLC への機能実装のための制約条件の LD 変換である．制御プログラムに異常入力による異常な状態遷移が発生したとき，LD 変換後の制約条件も異常入力を受け入れ，異常遷移を再現する必要がある．先行研究 6) では制御プログラムの現在の状態が異常かどうかを判定できるが，状態の遷移が異常かどうかは判定できない．

### 4.2 ペトリネットモデルの生成

ペトリネット[16]はプレースとトランジションの 2 種類のノードをもつ 2 部有向グラフであり，$N = (\mathbb{P}, \mathbb{T}, \mathbb{F}, W, m_0)$ の 5 項組で表現される．ここで，$\mathbb{P}$, $\mathbb{T}$ はそれぞれプレースとトランジションの集合，$\mathbb{F}$ はプレースとトランジション間のアーク集合，$W$ はアークの重みを与える関数，$m_0$ は初期マーキングであり，それぞれ

$$\mathbb{P} = \{p_1, p_2, \cdots, p_n\}, \qquad \mathbb{T} = \{t_1, t_2, \cdots, t_m\},$$
$$\mathbb{F} \subseteq (\mathbb{P} \times \mathbb{T}) \cup (\mathbb{T} \times \mathbb{P}),$$
$$W : \mathbb{F} \mapsto \mathbb{N}, \qquad m_0 : \mathbb{P} \mapsto \mathbb{N}$$

と定義される．ここで，$n$, $m$ はそれぞれプレースとトランジションの要素数である．また，マーキングとは各プレースがもつトークンの要素数を表わしている．**Fig. 8** はペトリネットモデルのグラフ表現の例である．

本節では 1 つ目の課題である PLC の制御プログラムから動作順序の情報を取得し，ペトリネットモデルを生成する方法を述べる．具体的には，LD と相互変換可能な SFC を用い，SFC 経由で通常動作をペトリネットモデル化する．SFC は，

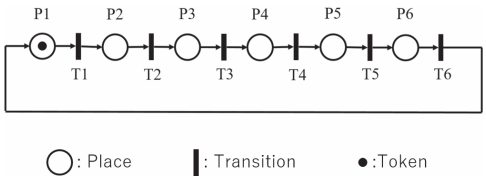


**Fig. 8**　Example of petri net model

ヨーロッパで利用されていた自動化シーケンスの仕様記述法である GRAFCET を元に規格化された PLC のプログラミング言語である [17]．GRAFCET はペトリネットを参考にしており，SFC 自体もペトリネットに類似した性質をもつ．そのため，LD に比べてペトリネットへの変換が容易である．対象とする制御プログラムを SFC に限定することで，1 つ目の課題を解決する．LD と SFC の相互変換に関しては PLC プログラミングの国際規格 IEC 61131-3（JIS B 3503）に従うものとする．

SFC をペトリネットモデルに変換するために，SFC の形式表現として $S = (\mathbb{S}, \mathbb{T}_S, \mathbb{F}_S, \mathbb{A}, \mathbb{C}, G_a, G_c, s_{\text{init}})$ の 8 項組

$$\mathbb{S} = \{s_1, s_2, \cdots, s_i\}, \qquad \mathbb{T}_S = \{t_1, t_2, \cdots, t_j\},$$
$$\mathbb{F}_S \subseteq (\mathbb{S} \times \mathbb{T}) \cup (\mathbb{T} \times \mathbb{S}), \qquad \mathbb{A} = \{a_1, a_2, \cdots, a_k\},$$
$$\mathbb{C} = \{c_1, c_2, \cdots, c_\ell\},$$
$$G_a : \mathbb{S} \mapsto \mathbb{A}, \qquad G_c : \mathbb{T} \mapsto \mathbb{C},$$
$$s_{\text{init}} \in \mathbb{S}$$

を定める．ここで，$\mathbb{S}$, $\mathbb{T}_S$ はそれぞれ Step，Transition の有限集合，$\mathbb{F}_S$ は Step と Transition 間のアーク集合，$\mathbb{A}$ は Action の有限集合，$\mathbb{C}$ は Transition の発火条件の有限集合，$G_a$，$G_c$ はそれぞれ Step と Transition に Action と発火条件を定める関数，$s_{\text{init}}$ は Initial Step である．また，$i$，$j$，$k$，$\ell$ はそれぞれ Step，Transition，Action，そして発火条件の個数である．SFC では複数の Transition に対しても同一の発火条件を設定することが可能である．このとき，同一の発火条件を有する Transition 群はすべて同じものとして区別されるのではなく，Transition に接続された Step との接続関係によって区別される．したがって，$G_c$ は Transition を区別するために定めた関数である．$G_a$ も同一の Action を有する Step でも Transition との接続関係によって区別するために定めた関数である．SFC の形式表現は文献 17) でも紹介されている．該当の方法は，同一の発火条件をもつトランジションを表現するために Step と Transition の 2 つの情報が必要となる．対して，本論文では制御器として OpenPLC を利用しているため，Step 情報を取得することができない．Transition 情報のみで同一の発火条件をもつトランジションを表現するために，SFC の形式表現を 8 項組 $S$ とする．OpenPLC において SFC は xml 形式のデータであるため，xml の構造解析によって $\mathbb{S}$，$\mathbb{T}_S$，$\mathbb{F}_S$，$\mathbb{A}$，$\mathbb{C}$，$G_a$，$G_c$，$s_{\text{init}}$ を取得することが可能である．

SFC をペトリネットに変換するためには，$\mathbb{S}$，$\mathbb{T}_S$，$\mathbb{F}_S$，$s_{\text{init}}$ を用いて $N = (\mathbb{P}, \mathbb{T}, \mathbb{F}, W, m_0)$ を

$$\mathbb{P} = \mathbb{S}, \qquad \mathbb{T} = \mathbb{T}_S, \qquad \mathbb{F} = \mathbb{F}_S,$$
$$W : \mathbb{F} \mapsto 1, \qquad m_0 : m_0(p) = \begin{cases} 1 & \text{if} \quad p = s_{\text{init}} \\ 0 & \text{otherwise} \end{cases}$$

とすればよい．ここで，SFC のアークは重みがないため，ペトリネットにおけるアークの重みはすべて 1 とする．

### 4.3　制約条件の導出

つぎに，2 つ目の課題であるペトリネットモデルに変換された SFC の正常な動作順序にかかわる制約条件を導出する．テストベッドの制御器である OpenPLC は，Step 情報を取得することができない．そこで，本論文では Step 情報が取得できない場合でもホワイトリストを生成するために，Transition 情報のみを用いた制約条件の導出方法を提案する．

プレースを $n$ 個，トランジションを $m$ 個もつペトリネットモデルの $k$ ステップ目の状態遷移は，トークンがどのプレースにあるかを示したマーキング $\boldsymbol{m}[k] = [\, m(p_1[k]), m(p_2[k]), \cdots, m(p_n[k])\, ]^T \in \{0,1\}^n$ と接続行列 $\boldsymbol{A} = \boldsymbol{A}_+ - \boldsymbol{A}_-$ $(\boldsymbol{A}_+, \boldsymbol{A}_- \in \{0,1\}^{n \times m})$ とトランジションの発火による入力 $\boldsymbol{u}[k] = [\, t_1[k], t_2[k], \cdots, t_m[k]\, ]^T \in \{0,1\}^m$ によって

$$\boldsymbol{m}[k+1] = \boldsymbol{m}[k] + \boldsymbol{A}\boldsymbol{u}[k] \tag{1}$$

と表わせる．ここで，$\boldsymbol{A}_+$ はトランジションからプレース，$\boldsymbol{A}_-$ はプレースからトランジションへの接続関係を表わした行列であり，$m(p[k])$ はステップ $k$ においてプレース $p$ がもつトークンの個数を返す関数である．SFC は Transition の発火によるイベント駆動型のプログラムであるため，ペトリネットモデルのステップ $k$ もイベント駆動で動作する．なお，接続行列は PIPE2 [18] などのペトリネットツールによってペトリネットモデルから生成可能である．またペトリネットには入力 $\boldsymbol{u}[k]$ の発火に関する制約条件

$$\boldsymbol{m}[k] - \boldsymbol{A}_-\boldsymbol{u}[k] \geq \boldsymbol{0}_n \tag{2}$$

が必要である．SFC に対するこの条件はある Transition が発火するには接続元の Step の状態がアクティブである必要があることを示す．すなわち，(2) 式が SFC の正常な動作順序にかかわる 1 つ目の制約条件である．

4.2 節で生成されたペトリネットモデルは，プレースとトークンの位置関係によって，SFC のどの Step がアクティブかを表現する．Initial Step は 1 つのみであるため，ペトリネットモデルにおける初期マーキングのトークンの総数は 1 個 $(\|\boldsymbol{m}[0]\|_1 = 1)$ である．また，SFC において同時にアクティブになれる Step は 1 つのみであるため

$$\|\boldsymbol{m}[k]\|_1 = 1 \tag{3}$$

が常に成り立つ．このトークンの制限によりペトリネットモデル内に存在するトークンは増減しないため，1 つのトラン

ジションから複数のプレースへ接続が存在しない．この事実を数式で表現するために，一度接続行列を列ごとに

$$\left.\begin{aligned}\boldsymbol{A}_+ &= [\boldsymbol{a}_+(1), \boldsymbol{a}_+(2), \cdots, \boldsymbol{a}_+(m)] \\ \boldsymbol{A}_- &= [\boldsymbol{a}_-(1), \boldsymbol{a}_-(2), \cdots, \boldsymbol{a}_-(m)]\end{aligned}\right\} \tag{4}$$

とする．するとトークンの制限による接続関係の構造に対する制限は列ごとの 1 ノルムによって

$$\left.\begin{aligned}\|\boldsymbol{a}_+(1)\|_1 = \cdots = \|\boldsymbol{a}_+(m)\|_1 = 1 \\ \|\boldsymbol{a}_-(1)\|_1 = \cdots = \|\boldsymbol{a}_-(m)\|_1 = 1\end{aligned}\right\} \tag{5}$$

と表わせる．この (5) 式は，各トランジションの入出力プレースは 1 つずつしか許されないことを表わしている．すべてのトランジションの入出力プレースが 1 つずつであるペトリネットは，状態機械と呼ばれる．(5) 式による制約が課せられた接続行列 $\boldsymbol{A}$ によって，$G_a$，$G_c$ が決定される．また，トランジションが 1 つ発火するごとにトークンが 1 個移動するため，一度に発火できるトランジションの数は常に 1 つだけである．すなわち，

$$\|\boldsymbol{u}[k]\|_1 = 1 \tag{6}$$

が SFC の正常な動作順序にかかわる 2 つ目の制約条件である．

2 つ目の課題を解決するために，(2) 式と (6) 式をトランジションだけの情報に変換することを考える．(2)，(3)，(5) 式より，現在ステップ $k$ から 1 つ前のステップ $k-1$ において SFC の状態が正常である場合

$$\boldsymbol{m}[k-1] - \boldsymbol{A}_-\boldsymbol{u}[k-1] = \boldsymbol{0}_n \tag{7}$$

が成り立つ．(1) 式と (7) 式を用いて (2) 式を変形すると

$$\begin{aligned}\boldsymbol{m}[k] \ &-\boldsymbol{A}_-\boldsymbol{u}[k] \\ =\ & \boldsymbol{m}[k-1] + \boldsymbol{A}\boldsymbol{u}[k-1] - \boldsymbol{A}_-\boldsymbol{u}_[k] \\ =\ & \boldsymbol{m}[k-1] - \boldsymbol{A}_-\boldsymbol{u}[k-1] + \boldsymbol{A}_+\boldsymbol{u}[k-1] \\ & -\boldsymbol{A}_-\boldsymbol{u}[k] \\ =\ & \boldsymbol{A}_+\boldsymbol{u}[k-1] - \boldsymbol{A}_-\boldsymbol{u}[k]\end{aligned} \tag{8}$$

を得る．すなわち，本論文の SFC では発火の制約条件として

$$\boldsymbol{A}_+\boldsymbol{u}[k-1] - \boldsymbol{A}_-\boldsymbol{u}[k] \geq \boldsymbol{0}_n \tag{9}$$

が成り立つ．(9) 式より入力 $\boldsymbol{u}[k]$ は 1 つ前の入力 $\boldsymbol{u}[k-1]$ によって正常かどうか判定することができる．ここで，(5)，(6) 式より $\|\boldsymbol{A}_+\boldsymbol{u}[k-1]\|_1 = \|\boldsymbol{A}_-\boldsymbol{u}[k]\|_1 = 1$ であるため，(9) 式は必ず等号が成り立つ．以上より，ペトリネットモデルにおいて PLC の正常な動作順序をトランジション情報によって表現することができた．正常な動作順序の制約条件をまとめると

$$\boldsymbol{A}_+\boldsymbol{u}[k-1] - \boldsymbol{A}_-\boldsymbol{u}[k] = \boldsymbol{0}_n \tag{10}$$

$$t_1[k] + t_2[k] + \cdots + t_m[k] = 1 \tag{11}$$

となる．(10) 式はトランジションの発火の順序に関する制約条件であり，(11) 式は一度に発火するトランジションの個数に関する制約条件である．両式を満たす動作をホワイトリストは正常な動作として登録し，それ以外の動作を異常と判定する．

### 4.4 制約条件の LD プログラム化

本節では，3 つ目の課題である PLC へのホワイトリスト機能実装のための制約条件 (10)，(11) の LD 変換について考える．制約条件を含めたペトリネットモデルの LD 変換手法が文献 6), 19) などで提案されている．既存手法の場合，現在の入力が LD の制約条件を満たすかどうかを判定できても，LD に異常な状態遷移を発生させるかどうかの判定ができるとは限らない．本論文では，遷移順序の制約条件である (10) 式が LD の異常遷移を再現できるよう変換する．

(10) 式の LD 変換の前に，まず，(11) 式の四則演算を介した LD 変換方法を提案する．(11) 式は Population count におけるカウント数が 1 個の場合や SOS1 (Special Ordered Set of type 1) [20] に類する．LD は論理演算によるプログラム記述が通常だが，論理演算化が困難な SOS1 の LD 上の記述は四則演算を利用する．LD 上の四則演算には FUN（ファンクション）機能を利用する．FUN の詳細については付録 A を参照されたい．**Fig. 9** の $y$ は異常を示すためのブーリアン型変数である．この LD では，ブーリアン型変数の $t_1$，$t_2$，$\cdots$，$t_m$ を整数型に変換し，それらの総和を加算ブロック (ADD) により求めている．総和が 1 と等しいか比較し，等しくないとき異常と判定し，$y$ が True となる．

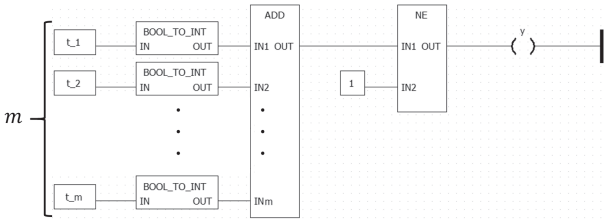


**Fig. 9** LD of Eq. (11)

つぎに，(10) 式の論理演算を介した LD 変換方法を提案する．(10) 式の係数行列やベクトルの各要素はすべて 0-1 変数であり，0-1 線形不等式と論理演算の関係 [21] から論理演算化可能である．$\boldsymbol{A}_+$ と $\boldsymbol{A}_-$ の $i$ 行 $j$ 列目の要素を $(A_+)_{ij}$ と $(A_-)_{ij}$，$\boldsymbol{u}[k]$ の $j$ 番目の要素を $t_j[k]$ とし，(10) 式を $i$ ごと $(1 \leq i \leq n)$ に分割すると

$$\sum_{j=1}^{m}(A_-)_{ij}t_j[k] - \sum_{j=1}^{m}(A_+)_{ij}t_j[k-1] = 0 \tag{12}$$

となる．(12) 式を論理式に変換すると

$$\sum_{j=1}^{m}(A_-)_{ij}t_j[k] = 1 \leftrightarrow \sum_{j=1}^{m}(A_+)_{ij}t_j[k-1] = 1 \tag{13}$$

となる．$\boldsymbol{u}[k]$ と $\boldsymbol{u}[k-1]$ の非ゼロ要素がそれぞれ $j$ 番目と $\ell$ 番目のとき，(13) 式は

$$(t_j[k] = 1) \wedge ((A_-)_{ij} = 1) \leftrightarrow (t_\ell[k-1] = 1) \wedge ((A_+)_{i\ell} = 1) \quad (14)$$

と変形できる．このとき動作順序が正しいとは，$j$ と $\ell$ に対して (14) 式が真であることである．つまり，$j$ によって

$$\mathbb{L}(j) = \{\ell \mid (A_-)_{ij} = 1, (A_+)_{i\ell} = 1, 1 \le \ell \le m\} \quad (15)$$

と定めたとき，$\ell \in \mathbb{L}(j)$ となれば動作順序が正しい．$\ell \in \mathbb{L}(j)$ を満たすような $\ell$ に対して (14) 式は

$$(t_j[k] = 1) \leftrightarrow (t_\ell[k-1] = 1) \quad (16)$$

となる．$\mathbb{L}(j)$ の要素数が複数あるのとき，$\ell_1$，$\ell_2$，$\cdots$，$\ell_{|\mathbb{L}(j)|}$ のうちに (16) 式を満たす要素が 1 つあればよいので，$t_j[k]$ の動作順序の制約条件は (16) 式を改めて

$$(t_j[k] = 1) \leftrightarrow (t_{\ell_1}[k-1] = 1) \vee (t_{\ell_2}[k-1] = 1) \vee \cdots \vee (t_{\ell_{|\mathbb{L}(j)|}}[k-1] = 1) \quad (17)$$

となる．すべてのトランジション $t_j$ $(1 \le j \le m)$ に対して (17) 式を導出することで (10) 式を論理演算で表現することができる．また，以降の表記を簡単にするため論理式において $(t_j[k] = 1)$ を単に $t_j$，そのときの (17) 式の必要条件 $(t_{\ell_1}[k-1] = 1) \vee (t_{\ell_2}[k-1] = 1) \vee \cdots \vee (t_{\ell_{|\mathbb{L}(j)|}}[k-1] = 1)$ を $t_\ell(j)$ とし，制約条件を

$$t_j \leftrightarrow t_\ell(j) \quad (18)$$

と表記する．

(17) 式を LD に変換する．この際，ペトリネットモデルと SFC の動作を対応させるために，(17) 式のトランジション情報 $\mathbb{T}$ を SFC の Transition の発火条件情報 $\mathbb{C}$ に戻す必要がある．これには 4.2 節で定めた関数 $G_c$ を用いる．$G_c$ はトランジション $t$ の発火を SFC の発火条件に対応させる関数である．そのため $G_c$ を，現在満たされた発火条件 $c \in \mathbb{C}$ がどのトランジション $t_j[k]$ に該当するかの判定に利用する．このとき，$G_c$ による $\mathbb{T}$ から $\mathbb{C}$ への写像が単射であるとは限らないため，$c = G_c(t_j[k])$ となる $j$ が複数存在する場合を考える必要がある．$j$ は $c$ を用いて

$$j \in \mathbb{J}(c) = \{j \mid G_c(t_j[k]) = c, 1 \le j \le m\} \quad (19)$$

と求めることができ，それらを $j_1$，$j_2$，$\cdots$，$j_{|\mathbb{J}(c)|}$ とすると制約条件である論理式 (18) は

$$\left.\begin{aligned} t_{j_1} &\leftrightarrow t_\ell(j_1) \\ t_{j_2} &\leftrightarrow t_\ell(j_2) \\ &\vdots \\ t_{j_{|\mathbb{J}(c)|}} &\leftrightarrow t_\ell(j_{|\mathbb{J}(c)|}) \end{aligned}\right\} \quad (20)$$

となる．(20) 式のいずれかを満たしているとき，発火条件 $c$ が正常であると判定できる．また，(11) 式の制約条件より，$j_1$，$j_2$，$\cdots$，$j_{|\mathbb{J}(c)|}$ のうち発火できるものは 1 つのみである．そのため (20) 式の論理和を取ることで発火条件 $c = t_{j_1} \vee t_{j_2} \vee \cdots \vee t_{j_{|\mathbb{J}(c)|}}$ の制約条件は

$$c \leftrightarrow t_\ell(j_1) \vee t_\ell(j_2) \vee \cdots \vee t_\ell(j_{|\mathbb{J}(c)|}) \quad (21)$$

となる．この (21) 式が，SFC の発火条件 $c$ が正しい順序で発火しているための制約条件となる．

論理式の LD への変換方法は文献 17) で紹介されているが，(21) 式を該当手法で LD に変換すると，(21) 式を満たさない $c$ は発火できないためホワイトリスト上で異常遷移が発生しない．すなわち，PLC の制御プログラムの異常遷移をホワイトリスト上で再現できず，異常を検知できない．そこで，遷移が正常か異常かを表わす $y \in \{0, 1\}$ を新たに導入する．$y = 0$ を正常，$y = 1$ を異常とする．(21) 式が満たされているときが正常であるため，その否定として $y$ を

$$\begin{aligned} y &= \overline{P \leftrightarrow Q} \\ &= \overline{(P \to Q) \wedge (Q \to P)} \\ &= \overline{(\overline{P} \vee Q))} \vee \overline{(\overline{Q} \vee P)} \\ &= (P \wedge \overline{Q}) \vee (Q \wedge \overline{P}) \end{aligned} \quad (22)$$

と設定する．ただし，(22) 式において $P = c$，$Q = t_\ell(j_1) \vee t_\ell(j_2) \vee \cdots \vee t_\ell(j_{|\mathbb{J}(c)|})$ である．ここで，(22) 式は，$c = 1$ のときにその発火が正常かどうかを判定する制約条件である．$c = 0$ つまりは $\overline{P}$ のとき，ホワイトリストは制約条件 (22) によって異常判定を実行しない．そのため (22) 式の $(Q \wedge \overline{P})$ は常に偽となる．よって，(22) 式は

$$\begin{aligned} y &= (P \wedge \overline{Q}) \vee (Q \wedge \overline{P}) \\ &= (P \wedge \overline{Q}) \\ &= c \wedge \overline{t_\ell(j_1)} \wedge \overline{t_\ell(j_2)} \wedge \cdots \wedge \overline{t_\ell(j_{|\mathbb{J}(c)|})} \end{aligned} \quad (23)$$

となる．(23) 式は，(21) 式を満たさない $c$ の発火によって $y = 1$ となり異常遷移が検出される．(23) 式を LD に変換したものが，**Fig. 10** である．論理式から LD への変換は付録 A を参照されたい．Fig. 10 の LD のコイル $y$ を監視することで異常を検出できる．

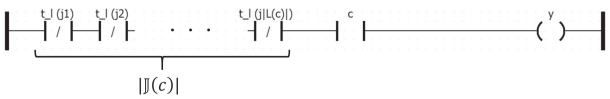


**Fig. 10**　LD of Eq. (23)

(23) 式は現在の発火 $c$ の異常を検知できるが発火したトランジションの特定はできない．(20) 式の $t_\ell(j)$，$t_\ell(j')$ $(j, j' \in \mathbb{J}(c))$ について，$j \neq j'$ のとき (5) 式の制約により $\mathbb{L}(j) \cap \mathbb{L}(j') = \phi$ となり共通要素をもたない．(20) 式において動作順序が正しいとき $t_\ell(j) = 1$ となる $j$ がただ 1 つ存在することになり，直前に発火したトランジション $t_\ell$ が $\ell \in \mathbb{L}(j)$ を満たす $j$ に対して

$$\left.\begin{array}{rcll} t_\ell(j) & \leftrightarrow & \overline{t_\ell(j')} & (j' \neq j) \\ \overline{t_\ell(j')} & \leftrightarrow & \overline{t_{j'}} & \end{array}\right\} \tag{24}$$

が成り立つ．発火条件 $c$ と (24) 式を組合せると

$$\begin{aligned} c \wedge t_\ell(j) &= t_{j_1} \vee t_{j_2} \vee \cdots \vee t_j \vee \cdots \vee t_{j_{|\mathbb{J}(c)|}} \\ &= 0 \vee \cdots \vee t_j \vee \cdots \vee 0 \\ &= t_j \end{aligned} \tag{25}$$

となり，$c$ と $t_\ell(j)$ から $t_j$ を決定することができる．LD に変換するために (25) 式を整理すると

$$t_j = c \wedge t_\ell(j) \qquad (j \in \mathbb{J}(c)|) \tag{26}$$

となる．(26) 式を各 $j$ について LD に変換したものが **Fig. 11** である．Fig. 11 の LD は発火したトランジション情報を保存するために，単なるコイルではなくセットコイルを用いている．

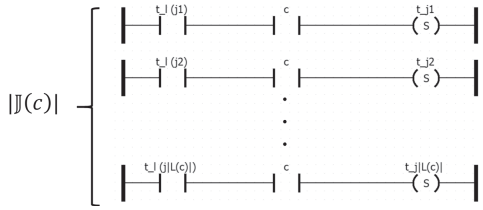


**Fig. 11** LD of Eq. (26)

## 5. 検 証 実 験

検証実験では，2 章で述べたテストベッドの制御器である OpenPLC に対して 4 章のホワイトリストを実装する．そして，正常な動作を行なった場合と模擬サイバー攻撃を行なった場合のホワイトリストの動作を検証する．模擬サイバー攻撃は，HMI が乗っ取られていたことを想定し，HMI から PLC に対して Modbus/TCP 通信パケットを送信することで不正操作を行なう．

提案した変換手法により，Fig. 7 に示したロボットアームの SFC をペトリネットに変換したものを **Fig. 12** に示す．また，各トランジションの発火条件を $G_c$ によって対応させたものが **Table 1** である．

まず，一度に発火できるトランジションの総数の制約条件

**Table 1** Firing rule of transition

| Transition | Firing rule |
|---|---|
| T1 | r_power_switch = True |
| T2 | isTargetAngle = True |
| T3 | r_switch = True |
| T4 | r_power_switch = False |
| T5 | r_reset_switch = True |
| T6 | r_switch = False |
| T7 | r_power_switch = False |
| T8 | isTargetAngle = True |
| T9 | isTargetAngle = True |

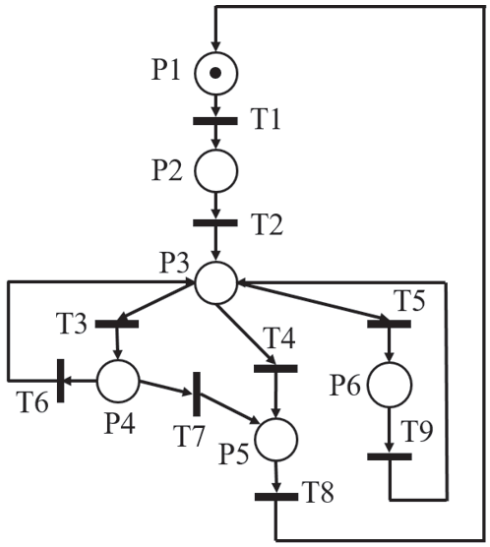


**Fig. 12** Petri net model for testbed

(11) はトランジションの個数が 9 個であるため

$$\mathrm{T1} + \mathrm{T2} + \cdots + \mathrm{T8} + \mathrm{T9} = 1 \tag{27}$$

となる．この (27) 式を LD で表現したものが，**Fig. 13** である．

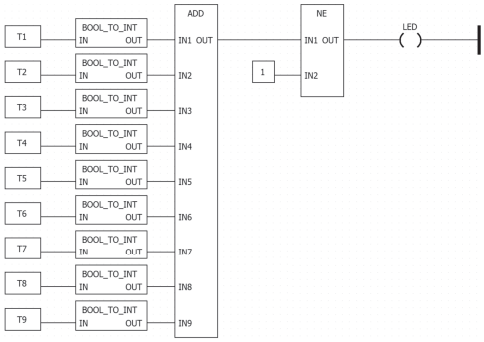


**Fig. 13** Constraints on the total number of transitions that can be fired

つぎに，発火条件の順序制約を LD に変換する．変換の一例として，Table 1 の isTargetAngle の発火によるトランジションの正常動作を LD にする．Table 1 より isTargetAngle は，トランジション T2, T8, T9 の発火条件である．各トランジションに対して (17) 式を導出すると

$$\mathrm{T2} \leftrightarrow \mathrm{T1}, \qquad \mathrm{T8} \leftrightarrow \mathrm{T4} \vee \mathrm{T7}, \qquad \mathrm{T9} \leftrightarrow \mathrm{T5} \tag{28}$$

となる．isTargetAngle の発火を X(= T2 ∨ T8 ∨ T9)，異常検知を Y とすると，(22) 式は

$$Y = \mathrm{X} \wedge (\overline{\mathrm{T1}} \wedge \overline{\mathrm{T4}} \wedge \overline{\mathrm{T5}} \wedge \overline{\mathrm{T7}}) \tag{29}$$

となる．この (29) 式を LD に変換したものが **Fig. 14** である．また，LD プログラム上では，isTargetAngle が X，LED が Y である．

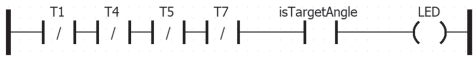


**Fig. 14**　LD expressing te order constraint of isTargetAngle

ホワイトリストの正常な状態遷移動作の検証結果を **Fig. 15** に示す．Fig. 15 の 1 番上のグラフは発火したトランジションの時系列データである．中央のグラフは，トランジション T2, T8, T9 の発火条件である isTargetAngle の時系列データである．1 番下のグラフは異常検知時に点灯する LED の時系列データである．グラフからはトランジションが T1, T2, T3, T6, T5, T9 の順に発火したことがわかる．これは制御システムの状態遷移をモデル化した Fig. 12 において，状態が P1，P2，P3，P4，P3，P6，P3 の順で遷移する発火順序である．isTargetAngle の動作についても 1 度目の発火（18 秒あたり）では直前のトランジションの発火が T1 であるため判定条件より T2 が発火したと正しく判定ができている．2 度目の発火（75 秒あたり）では直前の発火が T5 であることから，発火したのは T9 と正しく判定できている．このことから，ホワイトリストが正しく動作していることが検証できた．

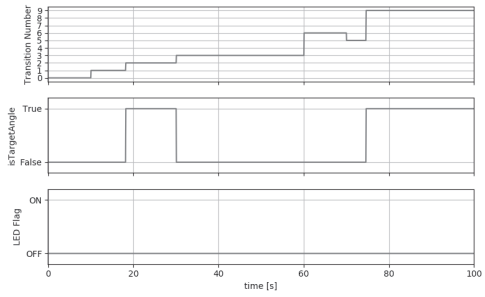


**Fig. 15**　Whitelist of normal operation in testbed

ホワイトリストによる異常検知の動作検証の結果を **Fig. 16** に示す．グラフの表わす情報は正常動作のときと同じで上から順に，トランジションの発火，isTargetAngle の状態，LED の点灯の時系列データである．異常検知の検証では，制御システムに正常動作の検証時と同じ（P1，P2，P3，P4，P3，P6，P3）順序で状態遷移を行なう．制御システムが正常動作を行なっている途中で攻撃者端末から不正なパケットを送信して isTargetAngle を異常変化させる．サイバー攻撃による isTargetAngle の異常変化は 50 秒付近で発生している．isTargetAngle の異常変化が発生したとき，直前に発火したトランジションは T3 である．これはホワイトリストに登録されていない状態遷移の順序であるため，異常を検知し LED が点灯している．このことから，ホワイトリストに登録されていない順序でトランジションが発火したとき異常を検知することが検証できた．

本論文の自動生成手法に基づくホワイトリストは SFC プログラムの動作順序の異常遷移を検知することができた．この検知技術は現実のサイバー攻撃においてネットワークや HMI に侵入した攻撃者などが生産ラインの機器を暴走や破壊するような命令を PLC に送る攻撃の検知に対して利用することができる．

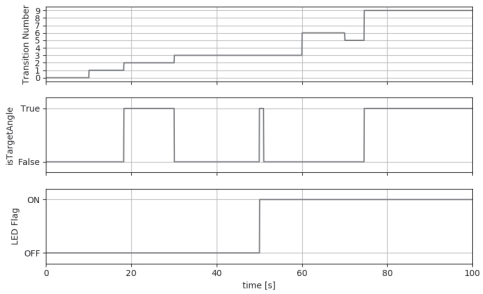


**Fig. 16**　Whitelist of abnormal operation in testbed

## 6. おわりに

本論文では，PLC の動作順序を監視するホワイトリストの自動生成手法を提案した．提案手法は PLC の制御プログラムを SFC に限定することにより，PLC の動作順序をペトリネットモデル化している．そして，ペトリネットモデルの動作順序の制約条件を四則演算と論理演算で表現し，実機に実装するために LD に変換する．本提案手法によって SFC の構造データからホワイトリストを生成できること，そして実機検証によって模擬攻撃を検知できることを示した．

提案手法は，SFC の Action 情報を使用せずに SFC をペトリネット化しているため，PLC の動作順序は監視できるが，動作内容が正しいかは監視しない．今後の課題として，Action 情報をホワイトリストに組込んだ動作内容の監視手法を検討する．また，IEC 61131-3 で規定されている SFC の選択シーケンスと並列シーケンスについても検討の必要がある．選択シーケンスは優先度をもつ条件分岐のことであり，本論文の手法は条件分岐を含んでいても正常に機能する．ただし，コントローラホワイトリストはシーケンスの順序を監視するものであるため，条件分岐の優先度を考慮していない．そのため，より精緻な異常検知のためにはこの優先度を考慮してコントローラホワイトリストを考える必要がある．そして，SFC が複数の Step を並列で処理させる並列シーケンスを含む場合，生成されるペトリネットモデルが状態機械を満たさない．そのため，並列シーケンスを含む SFC のホワイトリスト生成のために，動作順序を考慮したペトリネットモデルの状態機械化の手法を検討する必要がある．最後に，ホワイトリスト生成の元である SFC や生成したホワイトリスト自体が間違っていた場合，ホワイトリストによる異常検知が正常に機能しない．この問題を回避するためのホワイトリストの検証方法は別途報告予定である．

**謝辞**　OpenPLC を用いた実験環境の構築やペトリネットにおける正常状態の表現などさまざまな助力をいただきました電気通信大学卒業生の望月明典様と秦康祐様に感謝の意を表します．本成果は，技術研究組合制御システムセキュリティセンター (CSSC) が受託した，内閣府が進める戦略的イノベー

ション創造プログラム (SIP)「重要インフラ等におけるサイバーセキュリティの確保」（管理法人：NEDO）の委託業務の一部により得られたものです．

## 参 考 文 献


1）G. Liang: The 2015 Ukraine Blackout: Implications for False Data Injection Attacks, IEEE Transactions on Power Systems, **32**-4, 3317/3318 (2016)
2）A. Chuquilla, T. Guarda and G.N. Quiña: Ransomware: WannaCry Security is everyone's, 14th Iberian Conference on Information Systems and Technologies (CISTI), 1/4 (2019)
3）Foreign intelligence officials say attempted cyberattack on Israeli water utilities linked to Iran, The Washington Post, 2020/05/09 (2020)
4）H. Yang, L. Cheng and M.C. Chuah: Detecting Payload Attacks on Programmable Logic Controllers (PLCs), IEEE Conference on Communications and Network Security (CNS), 1/9 (2018)
5）D. Formby and R. Beyah: Temporal Execution Behavior for Host Anomaly Detection in Programmable Logic Controllers, IEEE Transactions on Information Forensics and Security, **15**, 1455/1469 (2019)
6）A. Mochizuki, K. Sawada, S. Shin and S. Hosokawa: On Experimental Verification of Model Based White list for PLC Anomaly Detection, The 11th Asian Control Conference 2017, 1766/1771 (2017)
7）S. Fujita, K. Hata, K. Sawada, S. Shin and S. Hosokawa: OpenPLC based control system testbed for PLC whitelisting system, AROB2018, 2385/2390 (2018)
8）T.R. Alves, M. Buratto, F.M. de Souza and T.V. Rodrigues: OpenPLC: An open source alternative to automation, IEEE Global Humanitarian Technology (GHTC), 585/589 (2014)
9）http://www.openplcproject.com/
10）https://www.raspberrypi.org/
11）http://arduino.org/
12）https://www.unipi.technology/
13）http://www.modbustools.com/modbus_slave.html
14）https://startiot.telenor.com/learning/esp8226-openrtos-and-managed-iot-cloud/
15）http://www.pixtend.de/
16）T. Murata: Petri Nets: Properites, Analysis and Applications, Proceedings of the IEEE, **77**-4, 541/580 (1989)
17）関口，高橋，青木，下川，鷹田：シーケンス制御工学—新しい理論と設計法—，電気学会 (1988)
18）http://pipe2.sourceforge.net/
19）D. Thapa, S. Dangol and G. Wang: Transformation from Petri Net Model to Programmable Logic Controller using One-to-One Mapping Technique, CIMCA-IAWTIC'06, 228/233 (2005)
20）H.P. Williams: *Model Building in Mathematical Programming*, 5th ed., John Wiley & Sons (2013)
21）井村順一：ハイブリッドシステムの制御—II—モデリング，システム/制御/情報，**51**-7, 306/312 (2007)


## 《付　　録》

### A. LD の動作と論理式との対応

PLC のプログラミング言語である LD (Ladder Diagram) について補足説明をする．LD とはリレー回路やシーケンス図をプログラムするための言語である．そのため，LD の主要素はリレー素子の名称に倣い，接点 “-| |-” とコイル “-( )-” と呼ばれる．また，接点には ON のとき通電する a 接点 “-| |-” と OFF のとき通電する b 接点 “-|/|-” がある．**Fig. A. 1** に示した LD を例に，その動作を説明する．両端にある太い縦線は母線である．

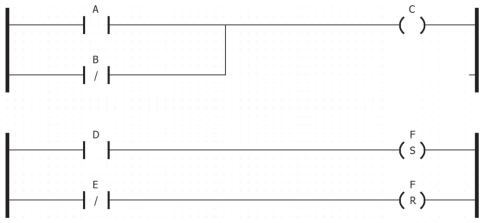


**Fig. A.1** Example of LD

Fig. A. 1 の上部の LD は，a 接点である A と b 接点である B がコイル C に並列接続されている．そのため，A が ON または B が OFF のとき C が動作する．これは，論理式によって

$$C = A \vee \overline{B} \qquad (A.1)$$

と表記することができる．

つぎに，Fig. A. 1 の下部の LD は，a 接点である D がコイル F “-(S)-，セットコイル” に，b 接点である E がコイル F “-(R)-，リセットコイル” にそれぞれ接続されている．セットコイルによって動作したコイルは，リセットコイルによって停止されるまで動作し続ける．つまりこの LD は，接点 D が一度 ON になれば，接点 E が OFF になるまで接点 D の状態に関係なくコイル F が動作を続ける．以上が，LD の基本的な動作である．

最後に，FUN（ファンクション）について説明する．FUN は，接点やコイルの代わりに配置できるブロックであり，各ブロックごとに固有の動作を行なう．FUN の例として，**Fig. A. 2** に加算ブロック (ADD) と比較ブロック (GE) を示す．各 FUN の EN 入力と ENO 出力はオプションであり，利用する場合は，EN 入力が ON のとき FUN を実行し ENO 出力から ON を出力する．利用しない場合は，常時 FUN を実行する．

ADD の動作は，IN1 と IN2 から入力される数値の和を OUT から出力する．GE の動作は，IN1 $\geq$ IN2 のとき，OUT から ON を出力する．FUN には，上記 2 種類以外に，減算や

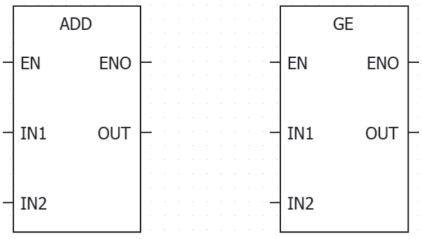


**Fig. A.2** Example of FUN

未満などのその他四則演算，比較演算ブロックやビット演算ブロックなどがある．

［著　者　紹　介］

**藤　田　真太郎**（学生会員）

2020 年電気通信大学情報理工学研究科機械知能システム学専攻博士前期課程修了．同年，同大学同専攻博士後期課程に進学，現在に至る．

**澤　田　賢　治**（正会員）

2009 年大阪大学大学院工学研究科機械工学専攻博士後期課程修了．同年電気通信大学システム工学科助教，2010 年同大学知能機械工学科助教，2015 年同大学 i-パワードエネルギー・システム研究センター准教授となり現在に至る．博士 (工学)．2019 年計測自動制御学会制御部門パイオニア技術賞受賞．2016 年より制御システムセキュリティセンター顧問．ハイブリッドシステムや制御系セキュリティに関する研究に従事．システム制御情報学会，電子情報通信学会，電気学会，IEEE 会員．

**新　　誠　一**（正会員）

1980 年東京大学大学院工学系研究科修士課程修了．同年，同大学工学部計数工学科助手．87 年工学博士（東京大学）．同大学講師を経て，88 年筑波大学電子・情報工学系助教授．92 年東京大学工学部助教授．2001 年，同大学情報理工学系研究科助教授．2006 年電気通信大学教授．2012 年制御システムセキュリティセンター理事長（兼任）．91 年，93 年，98 年計測自動制御学会論文賞，92 年同賞武田賞受賞．2006 年同技術賞受賞．2018 年経済産業大臣賞を受賞．制御理論を中心に広く工学全体に興味をもつ．

**細　川　　嵩**

2013 年電気通信大学電気通信学研究科電工工学専攻博士後期課程単位取得退学．2013 年より技術研究組合制御システムセキュリティセンターの研究員となり現在に至る．2015 年博士（工学）取得．制御システムセキュリティに関する教育と研究に従事．電子情報通信学会，IEEE 会員．2018 年より電気通信大学客員研究員．