2011 年度　修士論文

Simulink モデルにおける
非完全一致のモデルクローン検出に関する研究

2012 年 1 月 31 日(火)提出

指導：鷲崎 弘宜 准教授

早稲田大学大学院 基幹理工学研究科

情報理工学専攻

学籍番号：5110B119-3

村上 真一

目次

第 1 章 はじめに

自動車業界の企業をはじめとして，組み込みソフトウェア開発の現場ではモデルベース開発(Model Based Development:MBD)が広く行われている．MBD とは，従来のソフトウェア開発のようなソースコードによる開発とは異なり，開発の初期段階で実現すべき機能を設計図・モデルで作成し，開発の上流工程から下流工程において，これを検証しながら開発プロセスを進めていく開発手法のことで，開発効率や保守性，ソフトウェア資産の再利用性の向上といった利点がある．

MBD においてモデルを作成，検証するツールとして MATLAB/Simulink[1]が広く使われている．MATLAB/Simulink によって生成された Simulink モデルと呼ばれる Stateflow 図は，MATLAB/Simulink 上で動作のシミュレーションができるブロック線図としてプログラムを表現したモデルであり，その性質は「動く仕様書」とも言われ，高い評価を受けている．

Simulink モデル作成の過程においては，ソースコード作成の過程においてと同様，他の箇所からのコピー&ペーストがしばしば行われる．これによって発生した，ソフトウェア内における同一のブロック群をモデルクローンと呼ぶ．モデルクローンは同一であるにもかかわらず一元管理されておらず，統一して行われるべき修正が困難であるなど，保守性に大きな問題を引き起こしている．

この問題を解決するため，Simulink モデルにおけるモデルクローンを検出する研究が行われており，その成果は ConQAT[2][3]のようなツールの機能として一般に提供されているものもある．しかし，従来の研究では完全一致のモデルクローン検出が主であり，コピー&ペースト後に変更が加えられたために完全一致のモデルクローンではなくなったものの，依然として類似性が高く，統一的に管理されるべき非完全一致のモデルクローンを検出する手法は十分でなかった．

本研究では，従来モデルクローンとして扱われることのなかった非完全一致のモデルクローンに焦点を当て，完全一致のモデルクローンの検出結果をグラフ構造で整理した上で，グラフマイニングのアルゴリズムである AGM アルゴリズム[4]を利用し，Simulink モデルから非完全一致のモデルクローンを検出する手法を提案する．

本論文の構成を次に示す．第 1 章にて本研究の概要を説明し，第 2 章にて Simulink モデルを利用した組込みソフトウェア開発の背景と要素技術について述べる．第 3 章では，Simulink モデルを使用したモデルベース開発において問題となるモデルクローンについて述べ，第 4 章にて，本研究による提案として

非完全一致のモデルクローンを検出する手法を述べる．第 5 章では本提案手法の有効性を確認するためにサンプルモデルを対象に行った適用実験とその結果について述べ，第 6 章では関連研究について述べる．最後に，第 7 章にて本研究のまとめと今後の課題について述べる．

第 2 章 Simulink モデルを利用した組込みソフトウェア開発

2.1. モデルベース開発

モデルベース開発(Model Based Development:MBD)とは，従来のソフトウェア開発のようなソースコードによる開発とは異なり，開発の初期段階で実現すべき機能をモデルで作成し，開発の上流工程から下流工程において，これを検証しながら開発プロセスを進めていく開発手法のことである．

従来のソフトウェア開発では，要件および仕様は仕様書によってまとめられる．仕様書はプログラムからは独立して生成されるものであるため，書き手の意図した情報を読み手に誤解なく伝達することが非常に困難であるという問題がある．また，仕様書の妥当性を確認し，設計をより詳細に詰めていくためにプロトタイプを繰り返し構築していく必要があるが，このプロセスの繰り返しには大きなコストがかかる．特にハードウェア記述言語を用い，仕様書に基づいてそれを解読しながら人手でコーディングを行うには，高いスキルを持った要員を揃えることが必要となる上，コードの品質も要員やスキルに依存してばらつきが出る問題がある．

一方で，MATLAB/Simulink のようなモデルベース開発ツールで作成されたモデルは，ツール上で随時動作のシミュレーションを行える状態で設計を作り込んでいくことが可能なため，「実行可能な仕様書」などと呼ばれ，設計の詳細化や妥当性検証に大きく貢献できる．また，モデルの形で設計の意図を誤解なく共有することや，モデルからの自動コード生成といった特性によって，タイプミスや仕様書に記述された内容がコーディング作業者に正しく伝わらないことが原因で発生するエラーの混入も防ぐことができる．

その他，モデルがコードとは違い，設計をグラフィカルに表示することができることから，システムの内部を機能単位に分析することでコンポーネント間の相互作用を把握すること，過去に開発されたソフトウェア資産を最利用する際に，他人の作成したプログラムコードを参照しようとしたが，解読するのに苦労して結局再利用をあきらめざるを得なかったなどという状況を防ぐことなどができるとされている．

以上のような理由から，組み込みソフトウェアの開発現場では，MATLAB/Simulink と Simulink モデルを代表として，モデルベース開発ツールが広く使われている．

2.2. Simulink モデル

2.2.1. 概要

MATLAB/Simulink[1]上では，数値計算プログラムを機能単位でブロックとしてグラフィカルに表示し，GUI を使ってシミュレーションが実行できるモデルとして組み立てていくことができる．こうして MATLAB/Simulink 上で作られたブロック線図を Simulink モデルと呼ぶ. 処理のプログラミングに必要な代表的な計算式の多くがブロックライブラリとして提供されており，通信，制御，動画像処理，静止画像処理を含め，時間依存システムの設計，シミュレーション，実装，テストなどを可能としている．また，ブロック群は任意の個数で新たな 1 ブロックとしてまとめることができる．これを Subsystem と呼ぶ.

2.2.2. ブロック線図

Simulink モデルのブロック線図の例を図 2.1 に示す．このブロック線図は 1 階の微分方程式を表現したモデルであり，具体的には以下の式のシミュレーションに相当する.

$$\frac{dt}{dx} + 2x = u(t)$$
$$\Leftrightarrow \frac{dt}{dx} = -2x + u(t)$$

図 2.1 のブロック線図では，この式を乗算器（Gain）と積分器（Integrator）を使用して表現している.

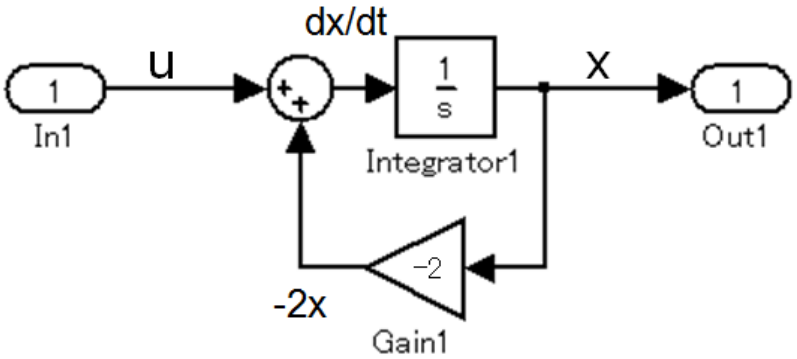

図 2.1 ブロック線図による Simulink モデル

2.2.3. Subsystem

Simulink モデルでは，任意の複数のブロックを Subsystem として 1 つにまとめることができる. Subsystem による表現の例を図 2.2 に示す. この例では，前節で述べた 1 階の微分方程式を実現しているブロック群を，Subsystem として 1 つのブロックにまとめている．このようにブロックを Subsystem 化することで，部品としてブロックの再利用性を高めることができ，開発効率を向上させることができる．実用の Simulink モデルでは，機能ごとにブロックを Subsystem 化することが頻繁に行われている．

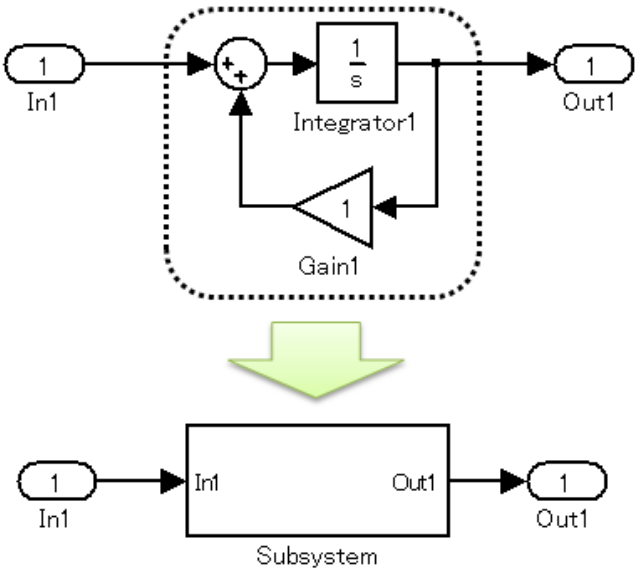

図 2.2 ブロックの Subsystem 化

第 3 章 モデルクローンの問題と既存のモデルクローン検出手法

3.1. モデルクローンの分類

3.1.1. 完全一致のモデルクローン

モデルを作成する過程において，処理の一部として必要となる他箇所と同一の構造を，他箇所からのコピー&ペーストによって作成することがしばしばある．このソフトウェア内における同一の構造を持ったブロック群を，モデルクローンと呼ぶ．以下，次節で説明する非完全一致のモデルクローンと区別するため，一般にモデルクローンと呼ばれているブロック群の構造が完全に一致しているモデルクローンを，完全一致のモデルクローンと呼ぶ．Simulink モデルにおける完全一致のモデルクローンの例を，図 3.1 に示す．

完全一致のモデルクローンは，ブロック群の構造が完全に一致しているにもかかわらず一元管理されていない．その結果として，部品化されていないことによる開発効率の低下や，統一して行われるべき修正が困難なことによる保守性の低下といった問題を引き起こしている．

ソフトウェア開発の効率性と，ソフトウェアの品質向上に貢献できるとして，モデルから完全一致のモデルクローンを検出，評価する研究が複数行われている．

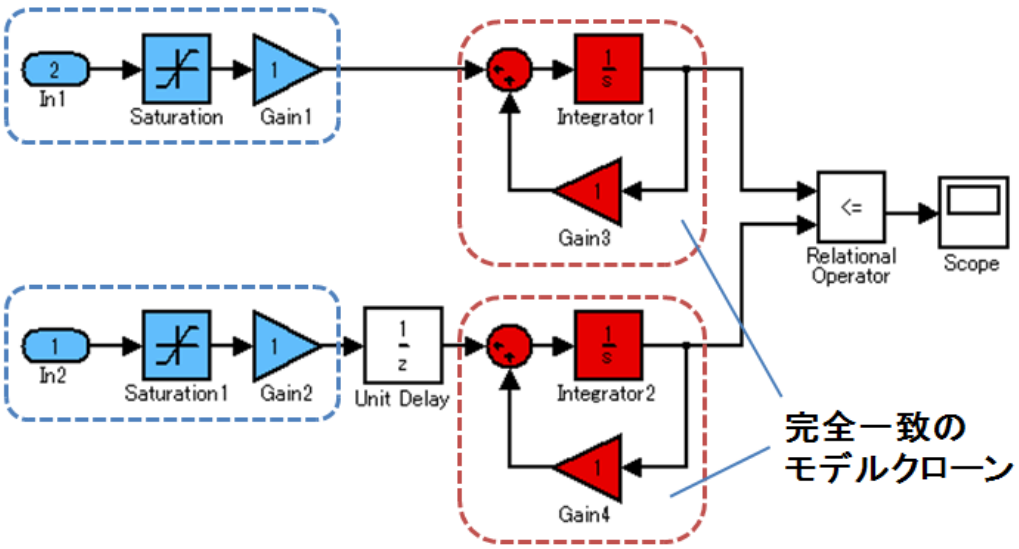

図 3.1 完全一致のモデルクローン

3.1.2. 非完全一致のモデルクローン

モデルクローンの中には，コピー&ペースト後に変更が加えられたために完全一致のモデルクローンではなくなったものの，依然として類似性が高く，統一的に管理されるべきモデルクローンが存在する．これを非完全一致のモデルクローンと呼ぶ．非完全一致のモデルクローンの例を，図 3.2 に示す．

非完全一致のモデルクローンは，その構造が完全一致ではないため，従来の多くの研究によるモデルクローン検出手法では検出することができない．

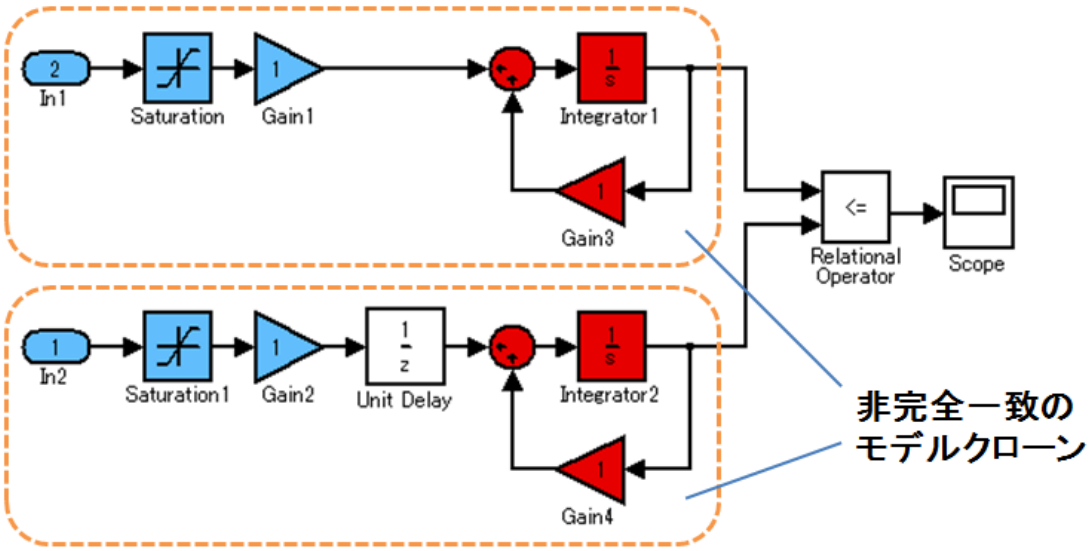

図 3.2 非完全一致のモデルクローン

3.2. 完全一致のモデルクローン検出ツール ConQAT

3.2.1. 概要

Simulink モデルを対象とした完全一致のモデルクローン検出に関する研究の提案手法を，機能の一つとして実装しているツールが複数存在する．ConQAT は，それらの中でも非常に有名かつ多機能なツールである．ConQAT は，Simulink モデルのモデルクローンの検出機能以外にも，Java や C，C#をはじめとした多くの言語を対象に，ソースコードのクローン検出機能や，モデルやソースコードの品質を様々なメトリクスで評価する機能を備えており，ソフトウェアの総合分析ツールとして知られている．また，Simulink も MDL ファイルを解析するライブラリ[5]も公開している．

本研究では，非完全一致のモデルクローンを検出する提案手法の中で，完全

一致のモデルクローンの検出結果を用いるため，その工程で ConQAT を利用する．ConQAT が完全一致のモデルクローンを検出する処理手順を，以下で簡単に説明する．

3.2.2. 完全一致のモデルクローン検出処理手順

ConQAT の，完全一致のモデルクローン検出処理は，以下の 3 ステップによって構成されている．

ステップ 1：構造解析

Simulink モデルのモデルファイル（MDL ファイル）を独自の解析器にかけ，ブロック線図の構造を簡略化したグラフに変換する．その際，あくまで構造の同一性にのみ注目するため，ブロックごとに与えられるパラメータなどの情報は無視される．

ステップ 2：検出処理

ノード数 k=1 から開始し，複数箇所に存在が確認される構造を検出していく．得られた結果を入力とし，k→k+1 として操作を繰り返すことで，次第に大きなノード数のモデルクローンが検出されるようになる．

処理の開始時にユーザが指定した，検出対象とするモデルクローンの最大ノード数の情報に従って処理を終了する．

ステップ 3：出力

モデルクローンとして検出された構造のサイズ，構造，検出箇所数と，検出箇所を Simulink モデルのモデルファイルを着色することで直接示す MATLAB/Simulink のコマンドファイル（M ファイル）を出力する．

3.2.3. 検出例

実際に図 3.3 に示すような Simulink モデルを入力とした時 ConQAT からは，モデルクローンとして検出された構造のサイズ，構造，検出箇所数をまとめて表示するための HTML と，MATLAB/Simulink 上で実行することで対象とした Simulink モデルのモデルファイルにおける完全一致のモデルクローン部分を，直接着色して示すコマンドを集合させたコマンドファイル（M ファイル）の 2 つの出力が得られる．例として図 3.3 に示す Simulink モデルを入力として与えた際に，ConQAT から出力される HTML による結果表示を図 3.4 に，“Clone 0”として検出される“Sum，Integrator，Gain”を対象としたコマンドファイル

をリスト 3.1 に，コマンドファイルを MATLAB/Simulink 上で実行した際の結果を図 3.5 に示す.

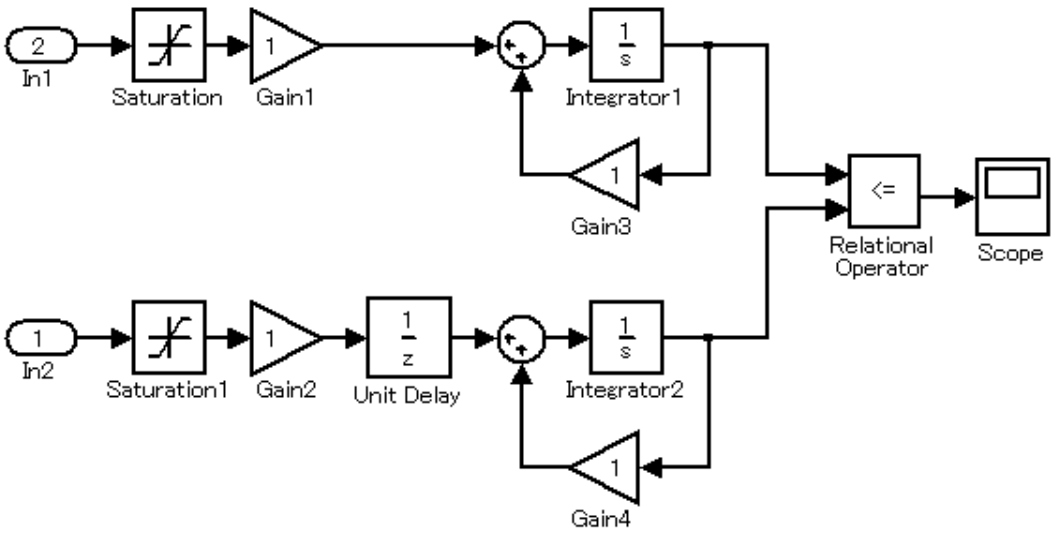

図 3.3 完全一致のモデルクローン検出のための Simulink モデル例

Model Clones (Clones)

Model Clones

Element	volume	weight sum	occurrences	size	weight	models	color-file	Layouted Clone
Clone 1	6	6	2	3	3	clone_sample	m-file	Clone 0 Clone 1
Clone 0	6	4	2	3	2	clone_sample	m-file	Clone 0 Clone 1

図 3.4 HTML による完全一致のモデルクローン検出結果表示

リスト 3.1 完全一致のモデルクローン部分を直接示すためのコマンドファイル

```
set_param(sprintf('clone_sample/Unit Delay'), 'BackgroundColor', 'white');
set_param(sprintf('clone_sample/Saturation'), 'BackgroundColor', 'white');
set_param(sprintf('clone_sample/Saturation1'), 'BackgroundColor', 'white');
set_param(sprintf('clone_sample/Integrator2'), 'BackgroundColor', '[1.000000, 0.000000, 0.000000]');
set_param(sprintf('clone_sample/Gain3'), 'BackgroundColor', '[0.000000, 1.000000, 1.000000]');
set_param(sprintf('clone_sample/Sum2'), 'BackgroundColor', '[0.000000, 1.000000, 1.000000]');
set_param(sprintf('clone_sample/Relational¥nOperator'), 'BackgroundColor', 'white');
set_param(sprintf('clone_sample/In1'), 'BackgroundColor', 'white');
set_param(sprintf('clone_sample/Gain1'), 'BackgroundColor', 'white');
set_param(sprintf('clone_sample/Gain2'), 'BackgroundColor', 'white');
set_param(sprintf('clone_sample/In2'), 'BackgroundColor', 'white');
set_param(sprintf('clone_sample/Sum1'), 'BackgroundColor', '[1.000000, 0.000000, 0.000000]');
set_param(sprintf('clone_sample/Scope'), 'BackgroundColor', 'white');
set_param(sprintf('clone_sample/Integrator1'), 'BackgroundColor', '[0.000000, 1.000000, 1.000000]');
set_param(sprintf('clone_sample/Gain4'), 'BackgroundColor', '[1.000000, 0.000000, 0.000000]');
```

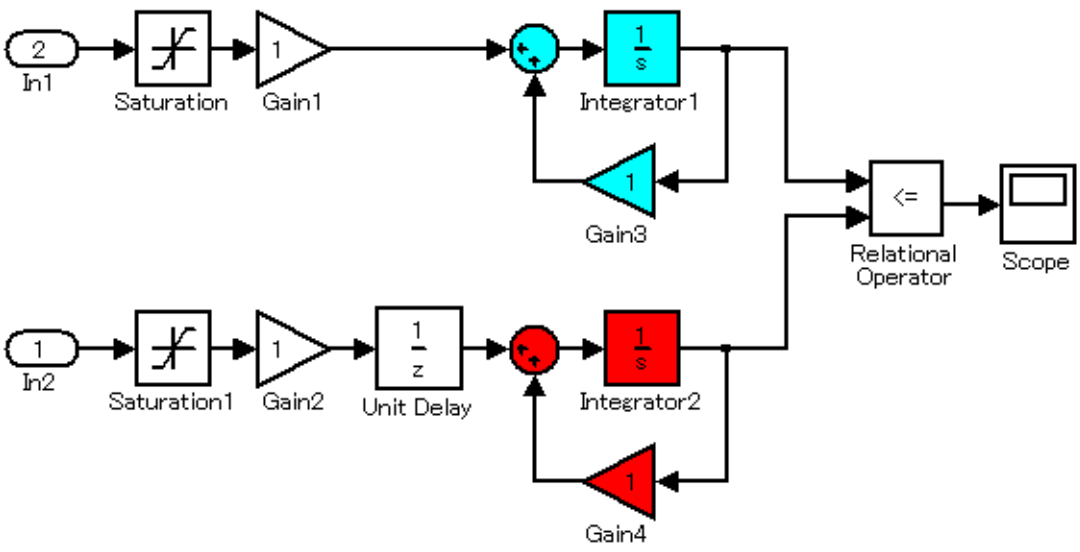

図 3.5 コマンドファイルを MATLAB/Simulink 上で実行した結果

第 4 章 非完全一致のモデルクローン検出手法の提案

4.1. 概要

既存の研究では非完全一致のモデルクローンを検出できない問題を解決するため，我々は完全一致のモデルクローンの検出結果と，グラフマイニングの分野で部分グラフ同型問題を解く代表的なアルゴリズムとして知られる AGM アルゴリズムを用いて，非完全一致のモデルクローンを検出する手法を提案する．

本提案手法は，次の 4 つのステップで構成されている．

ステップ 1：完全一致のモデルクローン検出

対象とする Simulink モデルから完全一致のモデルクローンを検出する．このステップでは既存のツール（ConQAT）を利用する．

ステップ 2：完全一致のモデルクローンの位置関係グラフ化

対象とする Simulink モデルに含まれる Subsystem ごとに，完全一致のモデルクローンの位置関係をグラフ化する．この際，完全一致のモデルクローンとして検出された構造以外のすべてのブロックを無視して考え，完全一致のモデルクローンのみの位置関係をグラフ化する．

ステップ 3：非完全一致のモデルクローンパターン検出

前ステップで Subsystem ごとに作成したグラフの集合を対象に，AGM アルゴリズムを用いて部分グラフ同型判定を行い，グラフの集合から多頻度グラフパターンを検出する．ここで検出された多頻度グラフパターンが，非完全一致のモデルクローンの構造の共通部分を表すパターンとなる．

ステップ 4：非完全一致のモデルクローン出力

検出された多頻度グラフパターンから，実際にその構造が現れる部分を特定し，非完全一致のモデルクローンとなるブロック群を抽出して，その位置情報と構造を出力する．

本研究の提案手法の全体像を図 4.1 に示す．また，図 4.2，図 4.3 に示す Simulink モデルを入力とした場合を例に，各ステップでどのような処理が行われるかを追って，本研究の提案手法の詳細を以降の節にて説明する．

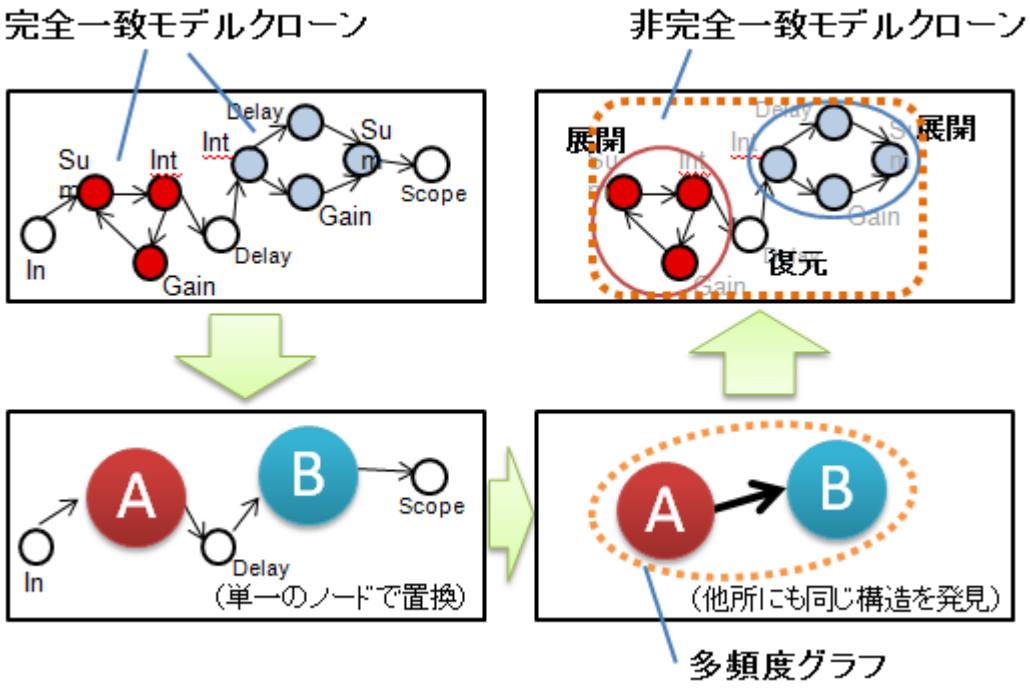

図 4.1 提案手法の全体像

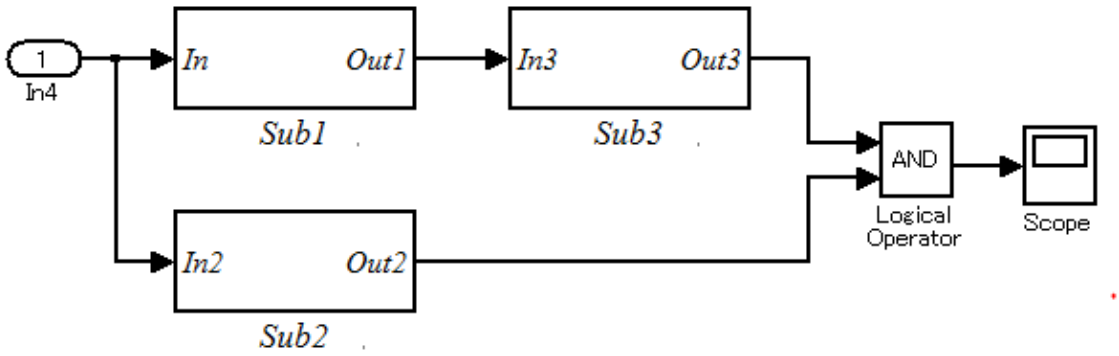

図 4.2 入力 Simulink モデルの全体図

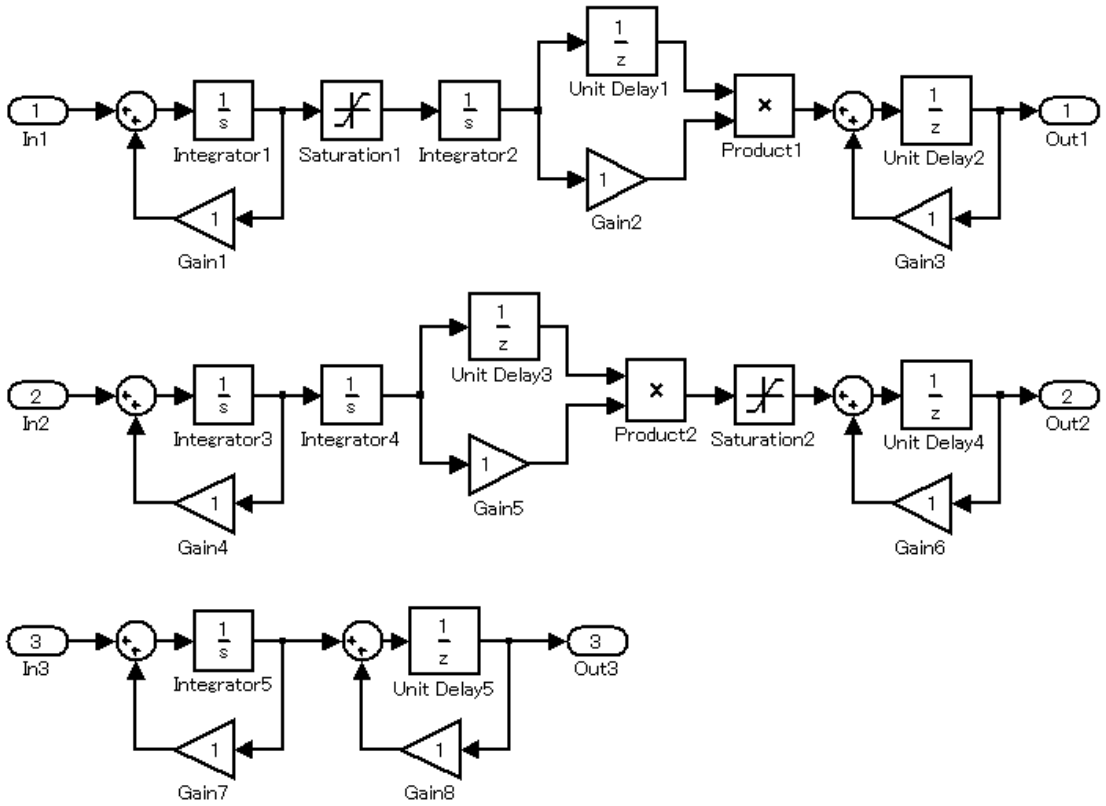

図 4.3 入力 Simulink モデルの詳細図

4.2. 完全一致のモデルクローン検出

単一，もしくは複数の Simulink モデルファイル（MDL ファイル）を入力とし，完全一致のモデルクローンとなっているブロック群の構造を検出する．この工程では，前述の既存ツールである ConQAT を利用する．

実際に図 4.2 で示した Simulink モデルを入力とした時の, ConQAT 上の結果表示を図 4.4 に，同時に出力される MATLAB/Simulink 上で完全一致のモデルクローンとなっている部分の位置を確認するためのコマンドファイル（M ファイル）を利用して，検出された 3 種類 8 箇所の完全一致のモデルクローンの構造と位置を示した結果を図 4.5 に示す．

図 4.4 に示す結果から，4 つのブロックから成り Element ID “Clone 0” とされる完全一致のモデルクローンがモデル内に 2 箇所存在すること, 3 つのブロックから成り Element ID “Clone 1” とされる完全一致のモデルクローンがモデル内に 3 箇所存在すること，3 つのブロックから成り Element ID “Clone 2” とされる完全一致のモデルクローンがモデル内に 3 箇所存在することがわかる．そして同時に出力されるコマンドファイルから，検出された 3 種類の完全一致のモデルクローンが, “Sum, Integrator, Gain” の 3 つのブロックによる構造と，“Integrator, Delay, Gain, Sum” の 4 つのブロック，そして，“Sum, Delay, Gain” の 3 つのブロックによる構造であることがわかる．

Model Clones (Clones)

Model Clones

Element	volume	weight sum	occurrences	size	weight	models	color-file	Layouted Clone
Clone 0	8	8	2	4	4	clone_test	m-file	Clone 0 Clone 1
Clone 1	9	9	3	3	3	clone_test	m-file	Clone 0 Clone 1 Clone 2
Clone 2	9	9	3	3	3	clone_test	m-file	Clone 0 Clone 1 Clone 2

図 4.4 ConQAT による出力

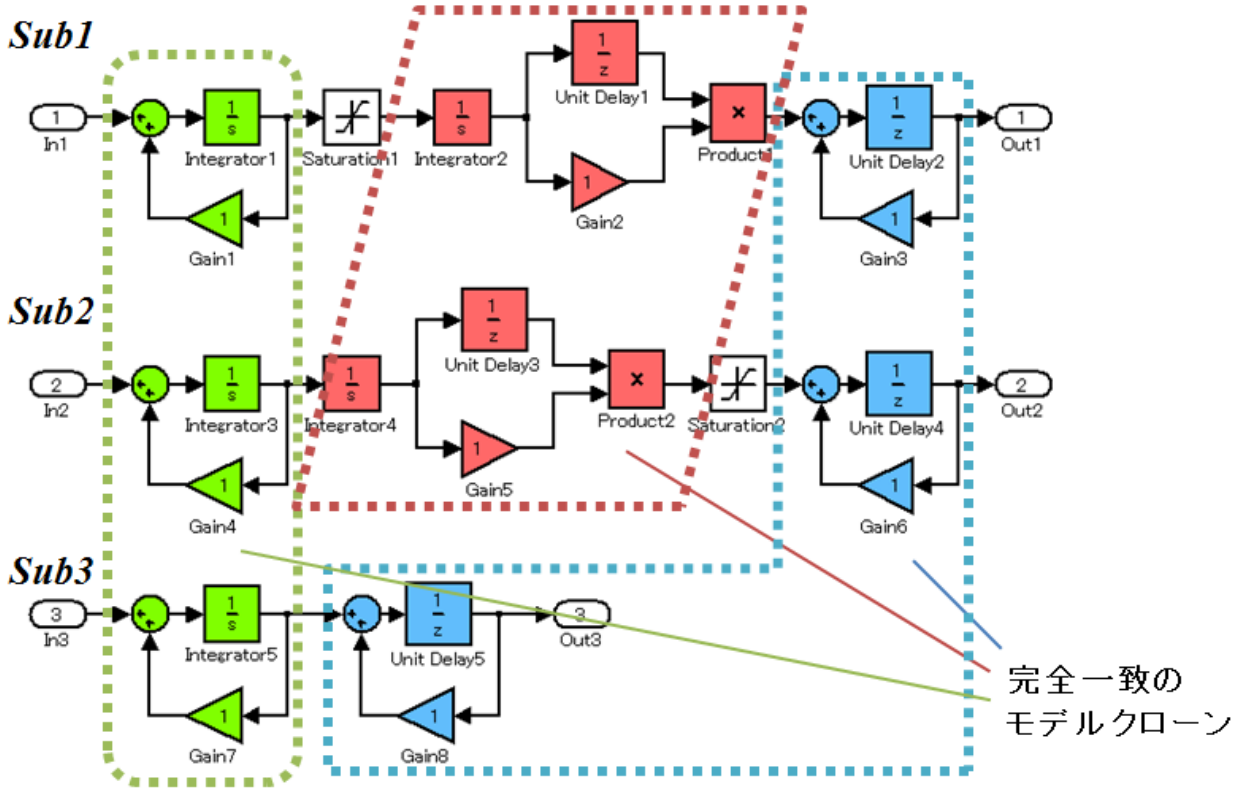

図 4.5 完全一致のモデルクローン

4.3. 完全一致のモデルクローンの位置関係グラフ化

Subsystem ごとに完全一致のモデルクローンの位置関係をグラフ化する．この工程は，次の 3 つのステップで構成される．

ステップ 1：単一ノードによる置き換え

完全一致のモデルクローンを構成しているブロック群を，モデルクローンの種類ごとに単一のノードに置き換える．

ステップ 2：完全一致のモデルクローン以外のブロックの除去

完全一致のモデルクローンのみの位置関係をグラフ化するため，完全位置のモデルクローンを構成しているブロック群以外のブロックを一時的に除去する．

ステップ 3：隣接行列による表現

ステップ 2 によって得られたグラフを，隣接行列によって表現する．

以下，グラフ構造データの表現方法について述べた後，実際に図 4.n の Simulink モデルからどのようなグラフ構造データが得られるかについて，各ステップごとに述べる．

4.3.1. グラフ構造データ

ラベル付きグラフ G は $G=(V, E, L_V, L_E, lb)$ で表される．ここで $V=\{v_1, v_2, \ldots, v_k\}$ は頂点の集合，$E=\{(v_i, v_j) \mid v_i, v_j \in V\}$ は辺の集合，$L_v=\{lb(v_i) \mid \forall v_i \in V\}$ は頂点ラベルの集合，$L_E=\{lb(v_i, v_j) \mid \forall (v_i, v_j) \in E\}$ は辺ラベルの集合，lb はラベル付の関数 $lb:(V \to L_v) \cup (E \to L_E)$ である．グラフ G の頂点，辺，頂点ラベル，辺ラベルの集合をそれぞれ $V(G), E(G), L_v(G), L_E(G)$ と表す．グラフの頂点数をグラフのサイズと呼ぶ．

グラフは隣接行列を用いて表現できる．$num(lb(v_i))$ と $num(lb((v_i, v_j)))$ をそれぞれ頂点ラベルと辺ラベルに割り当てられた正の整数値とすると，サイズが k のグラフ G が与えられたとき，その隣接行列は $k \times k$ の整数行列 X_k であり，その (i, j)-要素 $x_{i,j}$ は，

$$x_{i,j}=\begin{cases} num(lb((v_i,v_j))), & (v_i,v_j)\in E(G) \\ 0, & (v_i,v_j)\notin E(G) \end{cases}$$

となる．ここで，$i,j\in\{1,\ldots,k\}$である．隣接行列の i 行（ i 列）に相当する頂点を第 i 頂点と呼ぶ．隣接行列 X_k のグラフ構造を $G(X_k)$ と表す．同一のグラフに対応する隣接行列は行（列）の割り当て方で複数存在する．そこで，同一のグラフを一意に表現するためにある特定の条件を満たす隣接行列を正準形と呼ぶ．正準形を定義するために，隣接行列のコードを以下のように定義する．

$$code(X_k)=x_{1,2}x_{1,3}x_{2,3}x_{1,4}\cdots x_{k-2,k}x_{k-1,k}$$

関数 $code$ はグラフの構造と辺ラベルからなり，頂点ラベルの情報を含んでいない．そこで頂点ラベルを含めた関数 $CODE$ を，

$$CODE(X_k)=num(X_k)code(X_k)$$

と定義する．ここで関数 $num(X_k)$ は，ラベルに割り当てられた整数値を第 1 頂点から順に並べたもので，

$$num(X_k)=num(lb(v_1))\cdots num(lb(v_k))$$

である．あるグラフと一対一に対応する正準形は，最大の $CODE$ をもつ隣接行列と定義する．同型のグラフを表す隣接行列 X_k と Y_k が与えられたとき，以下のような要素からなる $k\times k$ の変換行列 T_k を用いて相互に変換することができる．

$$t_{ij}=\begin{cases} 1, & G(X_k)\text{の第}i\text{頂点が}G(Y_k)\text{の第}j\text{頂点に相当するとき} \\ 0, & \text{その他のとき} \end{cases}$$

Y_k は $Y_k=T_k^T X_k T_k$ と表される．

グラフ G と G_s が以下の 2 つの条件を満たすような関数 $\phi:V(G_s)\to V(G)$ が存在するとき，G_s を G の部分グラフと呼び，$G_s\subseteq G$ と表す．

(1)　$\forall v\in V(G_s),\ lb(v)=lb(\phi(v))$

(2)　$\forall(v_i,v_i)\in E(G_s),\ lb(v_i,v_j)=lb(\phi(v_i),\phi(v_j))$

さらに以下を満たすとき，G_s を G の誘導部分グラフと呼び，$G_s\subseteq_i G$ と表す．

$$(v_i,v_j)\in E(G_s)\Leftrightarrow(\phi(v_i),\phi(v_j))\in E(G)$$

グラフ上の任意の 2 頂点間にパスが存在するとき，連結グラフと呼ぶ.

グラフデータの集合 GD が与えられたとき，グラフパターン G_s の支持度 $sup(G_s)$ を

$$sup(G_s) = \frac{|\{G \mid G \in GD, G_s \subseteq_i G\}|}{|GD|}$$

と定義する．ユーザが指定した最小支持度以上の支持度を有するグラフパターンを多頻度グラフと呼ぶ.

4.3.2. 単一ノードによる置き換え

前節で述べたグラフ構造データの表現に関する定義を用い，図 4.3 の Simulink モデルを例に完全一致のモデルクローンの位置関係をグラフ化する手順を説明する.

まず，4.2 節で得られた完全一致のモデルクローンを構成しているブロック群を，モデルクローンの種類ごとに単一のノードに置き換える．この際，置き換える単一のノードに，それぞれ固有の ID を付しておく．今回は，各ブロック群を以下のように命名する.

A : “Sum, Integrator, Gain”
B : “Integrator, Delay, Gain, Sum”
C : “Sum, Delay, Gain”

この命名に従い，図 4.3 の Simulink モデルの該当する部分を単一のノードに置き換えたものを，図 4.6 に示す.

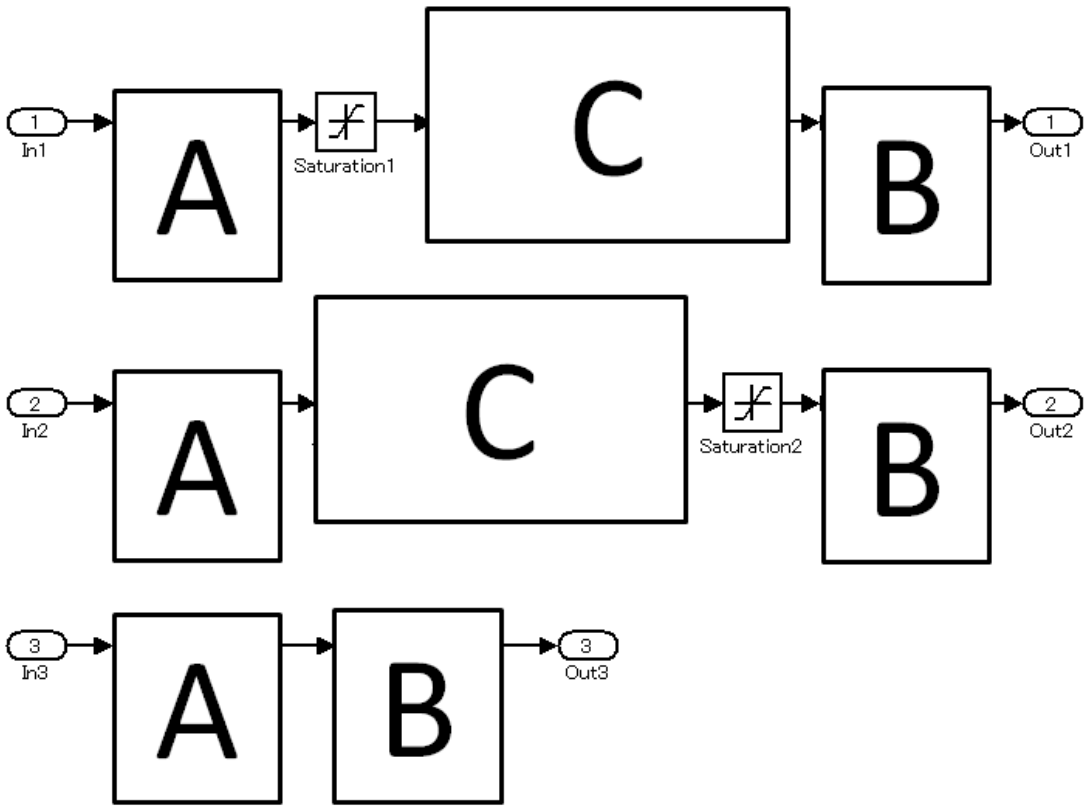

図 4.6 単一のノードによる置き換え

4.3.3. 完全一致のモデルクローン以外のブロックの除去

非完全一致のモデルクローンを検出する準備として，完全一致のモデルクローンのみの位置関係をグラフ化しておく必要がある．そのため，完全位置のモデルクローンを構成しているブロック群以外のブロックを一時的に除去する．

図 4.6 に示した Simulink モデルから，完全一致のモデルクローンを構成しているブロック群以外のブロックを除去した Simulink モデルを，図 4.7 に示す．

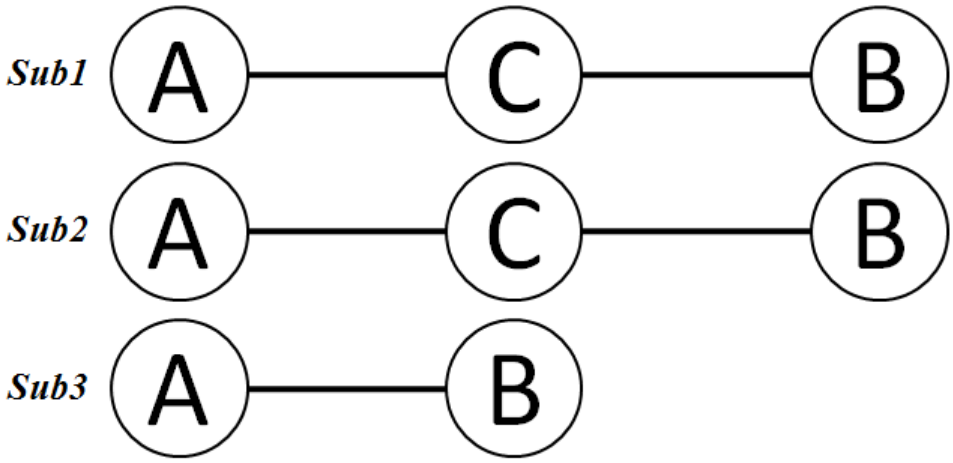

図 4.7 完全一致のモデルクローン以外のブロックの除去

4.3.4. 隣接行列による表現

前ステップで得られたグラフを入力としてグラフマイニングを行うために，図 4.7 の Simulink モデルを，Subsystem ごとに 4.3.1.節で述べた隣接行列による表現に直す.

まず，単一のノードとして置き換えられた頂点ラベル A，B，C に対し，1，2，3 を割り当て，辺ラベルは 1 を割り当てる.

$Sub1, Sub2, Sub3$ を表した隣接行列を以下に示す.

$$Sub1 = \begin{array}{c} \\ C \\ B \\ A \end{array} \begin{array}{c} \begin{array}{ccc} C & B & A \end{array} \\ \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} \end{array} \quad Sub2 = \begin{array}{c} \\ C \\ B \\ A \end{array} \begin{array}{c} \begin{array}{ccc} C & B & A \end{array} \\ \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} \end{array} \quad Sub3 = \begin{array}{c} \\ C \\ B \end{array} \begin{array}{c} \begin{array}{cc} C & B \end{array} \\ \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \end{array}$$

このとき，各 Subsystem の $CODE$ は,

$CODE(Sub1) = 321011$

$CODE(Sub2) = 321011$

$CODE(Sub3) = 321$

となっており，各グラフの持てる最大の値となっているため，これらは正準形である.

4.4. 非完全一致のモデルクローンパターン検出

4.3 節で Subsystem ごとに作成した完全一致のモデルクローンの位置関係を表したグラフの集合を対象に，AGM アルゴリズムを用いて部分グラフ同型判定を行い，グラフの集合から多頻度グラフパターンを検出する．ここで検出された多頻度グラフパターンが，非完全一致のモデルクローンの構造の共通部分を表すパターンとなる．この工程は，次の 5 つのステップで構成される．

ステップ 1：隣接行列結合

サイズkの多頻度グラフパターンを組み合わせて，サイズ$k+1$の多頻度グラフの候補を生成する．結合された隣接行列は，新たに結合されたノード間のリンクの有無によって，複数の正規形のグラフとして生成される．

ステップ 2：部分グラフチェック

結合された隣接行列が多頻度グラフであるためには，そのすべての誘導部分グラフが多頻度グラフでなくてはならない．結合された隣接行列の誘導部分グラフが多頻度グラフであるかどうかをチェックする．

ステップ 3：正準化

結合された隣接行列の正規形を正準化する．

ステップ 4：頻度計算

多頻度グラフパターンの候補の支持度を求める．

ステップ 5：非完全一致のモデルクローン出力

得られた多頻度グラフパターンから，具体的に非完全一致のモデルクローンとなっているブロック群を特定し，出力する．

以下，AGM アルゴリズムの概要について述べた後，実際に 4.3.4.節で得られた隣接行列から，多頻度グラフパターンを検出し，非完全一致のモデルクローンとして出力する手順を説明する．

4.4.1. AGM アルゴリズム

AGM アルゴリズムとは，グラフ構造データベースの中に部分構造として含ま

れる特徴的なパターンを完全探索によって抽出するアルゴリズムであり，POS システムなどのデータベース向けに開発された Apriori アルゴリズム[6]をグラフ構造データに拡張したものである. AGM アルゴリズムの概要を図 4.8 に示す. ここで GD はグラフの集合からなるデータベース，F_k は大きさ k の多頻度グラフの隣接行列の集合，C_k は大きさ k の多頻度グラフの候補の隣接行列の集合，$minsup$ は最小支持度を表している.

AGM アルゴリズムは，サイズが 1 の多頻度グラフパターンから，順にサイズの大きなグラフパターンを抽出していく. はじめにサイズが 1 の 1×1 の隣接行列が頂点ラベルの種類だけ生成され，C_1 に代入される．次に，多頻度グラフパターンの候補の支持度を求め, 多頻度グラフパターンを F_k に代入する. 続いて，既に抽出されている大きさ k の多頻度グラフパターンを組み合わせて，大きさ $k+1$ の多頻度グラフの候補を生成する．この操作は C_k が空集合になるまで続けられ，最後にすべての多頻度グラフパターンが出力される.

図 4.8 の関数 Generate – Candidate は，隣接行列結合と部分グラフチェック，正準化の 3 つの処理からなる.

```
0)  Main(GD, minsup){
1)    C_1 ← {大きさ1の多頻度グラフの候補の隣接行列};
2)    k ← 1
3)    while(C_k ≠ φ){
4)      Count(GD, C_k);
5)      F_k ← {c_k ∈ C_k | sup(G(c_k)) ≥ minsup};
6)      C_{k+1} ← Generate – Candidate(F_k);
7)      k ← k + 1
8)    }
9)  return ∪_k { f_k ∈ F_k | f_k is canonical };
10) }
```

図 4.8　AGM アルゴリズムの概要

4.4.2. 隣接行列結合

より大きなサイズの多頻度グラフパターンを求めるため, サイズ k の多頻度グラフパターンを組み合わせて，サイズ $k+1$ の多頻度グラフの候補を生成する．2 つの隣接行列 X_k と Y_k が与えられ，以下の条件をすべて満たすとき，2 つの隣接

行列を結合し，大きさ$k+1$の隣接行列Z_{k+1}を生成する．

条件 1

$V(G(X_k))=\{x_1,\ldots,x_k\}$，$V(G(Y_k))=\{y_1,\ldots,y_k\}$とすると，$X_k$と$Y_k$が第$k$行，および第$k$列の要素以外の要素が全て等しいとき，すなわち

$$X_k=\begin{pmatrix} X_{k-1} & x_1 \\ x_2^T & 0 \end{pmatrix},\quad Y_k=\begin{pmatrix} X_{k-1} & y_1 \\ y_2^T & 0 \end{pmatrix}$$

であり，$x_i \in V(G(X_k))$，$y_i \in V(G(Y_k))$としたときに，任意の$i=1,\ldots,k-1$に対して，$lb(x_i)=lb(y_i)$ が成り立つとき，x_iとy_iはそれぞれ$G(X_k)$と$G(Y_k)$の第i頂点である．

条件 2

X_kが正準形であるとき．

条件 3

$CODE(X_k)\geq CODE(Y_k)$が満たされるとき．

X_kとY_kが条件 1-3 を満たし結合可能な場合，2 つの隣接行列を結合して，以下のようにZ_{k+1}を生成する．

$$Z_{k+1}=\begin{pmatrix} X_{k-1} & x_1 & y_1 \\ x_2^T & 0 & z_{k,k+1} \\ y_2^T & z_{k+1,k} & 0 \end{pmatrix}$$

このとき，任意の$i=1,\ldots,k-1$に対して，$lb(x_i)=lb(y_i)$ ，および$lb(x_k)=lb(y_k)$ が成り立つ．X_kとY_kは，Z_{k+1}の第 1 生成行列と第 2 生成行列と呼ばれる．

Z_{k+1}の 2 つの要素$z_{k,k+1}$と$z_{k+1,k}$は，X_kとY_kの要素からは決めることができない．それらの可能な値として，$G(Z_{k+1})$の第k頂点，第$k+1$頂点の間に辺がない場合と，あるラベルを持つ辺が存在する場合がある．よってZ_{k+1}は$(|L_E(E)|+1)$個の隣接行列が生成される．以上のようにして作られた隣接行列を正規形と呼ぶ．

例として，実際に 4.3.4.節で得られた隣接行列をもとに，サイズが1×1の隣接行列から，隣接行列が結合される際の流れを説明する．

まず，A，B，C の 3 つの頂点ラベルの種類の数だけ，サイズ1×1の隣接行列

が以下のように生成される．

$$\begin{array}{c} \\ A \end{array}\quad \begin{array}{c} A \\ (0) \end{array} \qquad \begin{array}{c} \\ B \end{array}\quad \begin{array}{c} B \\ (0) \end{array} \qquad \begin{array}{c} \\ C \end{array}\quad \begin{array}{c} C \\ (0) \end{array}$$

サイズ1×1の隣接行列を考える時点では辺は存在しないため，実際には存在しない隣接行列が生成されることはない．この後，頻度計算のみが行われる．

次に，2 つの頂点を持つサイズ2×2の隣接行列が生成される場合を考える．このとき，2 つの頂点と辺の有無の組み合わせとして，${}_3C_2 \times 2 = 6$種類の隣接行列が以下のように生成される．

$$\begin{array}{c} \\ B \\ A \end{array}\quad \begin{array}{c} B \quad A \\ \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \end{array} \qquad \begin{array}{c} \\ B \\ A \end{array}\quad \begin{array}{c} B \quad A \\ \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \end{array} \qquad \begin{array}{c} \\ C \\ A \end{array}\quad \begin{array}{c} C \quad A \\ \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \end{array} \qquad \begin{array}{c} \\ C \\ A \end{array}\quad \begin{array}{c} C \quad A \\ \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \end{array} \qquad \begin{array}{c} \\ C \\ B \end{array}\quad \begin{array}{c} C \quad B \\ \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \end{array} \qquad \begin{array}{c} \\ C \\ B \end{array}\quad \begin{array}{c} C \quad B \\ \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \end{array}$$

3 つの頂点を持つサイズ3×3の隣接行列を生成する工程の前に，図 4.8 で示した AGM アルゴリズムに従って，次節以降で説明する部分グラフチェックと正準化，頻度計算が行われ，隣接行列が絞られる．

3 つの頂点を持つサイズ3×3の隣接行列を生成する際には，以下の 3 つの 2 つの頂点を持つサイズ2×2の隣接行列から合成される．

$$\begin{array}{c} \\ B \\ A \end{array}\quad \begin{array}{c} B \quad A \\ \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \end{array} \qquad \begin{array}{c} \\ C \\ A \end{array}\quad \begin{array}{c} C \quad A \\ \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \end{array} \qquad \begin{array}{c} \\ C \\ B \end{array}\quad \begin{array}{c} C \quad B \\ \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \end{array}$$

順にM_1， M_2， M_3とすると，各隣接行列の$CODE$は

$$CODE(M_1) = 211$$

$$CODE(M_2) = 311$$

$$CODE(M_3) = 321$$

のようになる．ここでM_2とM_3が

1. 第k行，第k列の要素以外が全て等しい
2. X_kが正準形である
3. $CODE(X_k) \geq CODE(Y_k)$が満たされている

ことに注意して，M_2 と M_3 の隣接行列の結合を行うと，以下の 2 種類の 3 つの頂点を持つサイズ3×3の隣接行列が生成される．

$$\begin{array}{c} \\ C \\ B \\ A \end{array}\!\!\begin{array}{c} \begin{array}{ccc} C & B & A \end{array} \\ \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} \end{array} \qquad \begin{array}{c} \\ C \\ B \\ A \end{array}\!\!\begin{array}{c} \begin{array}{ccc} C & B & A \end{array} \\ \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} \end{array}$$

サイズ1×1の隣接行列から，サイズ2×2の隣接行列，サイズ3×3の隣接行列と順に生成されていく多頻度グラフの関係をまとめたものを図 4.9 に示す．

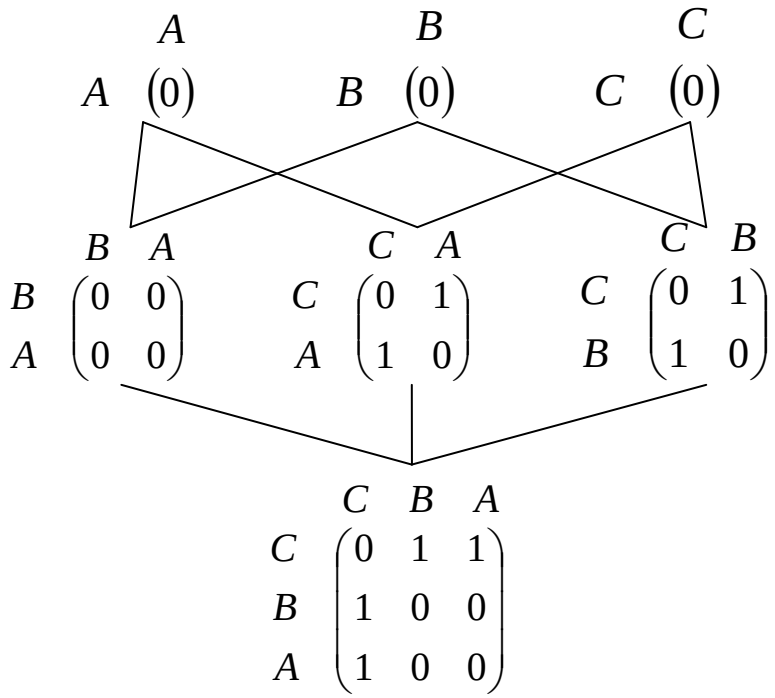

図 4.9 隣接行列の結合

4.4.3. 部分グラフチェック

結合された隣接行列が多頻度グラフであるためには，そのすべての誘導部分グラフが多頻度グラフでなくてはならない．このチェックを部分グラフチェックによって行うことで，多頻度グラフパターンの候補数を減らすことができる．

$G(Z_{k+1})$のサイズkの誘導部分グラフの隣接行列は，i行およびi列のすべての要素を取り除くことで得られるが，それは必ずしも正規形ではない．AGM アルゴリズムは正規形の隣接行列しか探索，生成しないので，その隣接行列に相当するグラフが多頻度グラフであるかチェックするためには，正規形の隣接行列に変換する必要がある．

サイズ$k+1$のグラフの候補を生成するとき，サイズkのグラフの隣接行列は図 4.10 に示すアルゴリズムによって正規形に変換される．隣接行列X_kの左上の$i \times i$の部分行列をX_i，X_iの正準形への変換行列をP_i，$k \times k$の単位行列をI_kとする．図 4.10 の変換行列P'_k，Q_kは以下のように生成される．

$$P'_k = \begin{pmatrix} P_i & 0 \\ 0 & I_{k-i} \end{pmatrix} \qquad Q_k = \begin{pmatrix} I_{i-2} & 0 & 0 \\ 0 & 0 & 1 \\ 0 & I_{k-i+1} & 0 \end{pmatrix}$$

尚，図 4.10 の 4 行目はX_iを正準形に変換する操作，7 行目は第$i-1$頂点を第k頂点に，第j頂点（$i \le j \le k$）を第$j-1$頂点に変更する操作である．

```
0)  Normalize(X_k){
1)    i ← 1;
2)    while(i ≠ k+1){
3)      if (X_i が正規形かつ第1生成行列となり得る){
4)        X_k ← P'_k^T X_k P'_k ;
5)        i ← i+1;
6)      }else{
7)      X_k ← Q_k^T X_k Q_k ;
8)      i ← i+1;
9)      }
10)   }
11)  return X_k ;
12)  }
```

図 4.10　正規化アルゴリズムの概略

4.4.2.節で示した，2 種類の 3 つの頂点を持つサイズ3×3の隣接行列を順にM_4，M_5とし，これらに対して部分グラフチェックを行う．まず，M_4から$1 \le i \le 3$の範囲でi行，i列を除去した隣接行列を生成すると以下の 3 つのようになる．

$$\begin{array}{c} \\ B \\ A \end{array}\begin{array}{c} \begin{array}{cc} B & A \end{array} \\ \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \end{array} \qquad \begin{array}{c} \\ C \\ A \end{array}\begin{array}{c} \begin{array}{cc} C & A \end{array} \\ \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \end{array} \qquad \begin{array}{c} \\ C \\ B \end{array}\begin{array}{c} \begin{array}{cc} C & B \end{array} \\ \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \end{array}$$

これらの誘導部分グラフは，すべて 2 つの頂点を持つサイズ2×2の多頻度グラフである．

同様にM_5から$1 \le i \le 3$の範囲でi行，i列を除去した隣接行列を生成すると以下の 3 つのようになる．

$$\begin{array}{c} \\ B \\ A \end{array}\begin{array}{c} \begin{array}{cc} B & A \end{array} \\ \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \end{array} \qquad \begin{array}{c} \\ C \\ A \end{array}\begin{array}{c} \begin{array}{cc} C & A \end{array} \\ \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \end{array} \qquad \begin{array}{c} \\ C \\ B \end{array}\begin{array}{c} \begin{array}{cc} C & B \end{array} \\ \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \end{array}$$

この場合，1 つ目の誘導部分グラフが多頻度グラフではないため，M_5は多頻度グラフパターンの候補から外れる．

よって，3 つの頂点を持つサイズ3×3の隣接行列としては，M_4のみが多頻度グラフパターンの候補として残る．

4.4.4. 正準化

あるサイズの多頻度グラフパターン候補となる隣接行列をすべて生成した後，グラフデータベースを参照してそれらの支持度を求める．しかし，正規形の中にも同一のグラフを表す隣接行列が複数存在することがあるため，正規形の隣接行列の中でどの行列が正準形であるか求める必要がある．

X_kからi行，i列の要素を覗いた隣接行列をX_{k-1}^iとする．X_{k-1}^iを正規形に変換する行列をT_{k-1}^iとする．正規化された$\left(T_{k-1}^i\right)^T X_k^i T_{k-1}^i$を正準形に変換する行列を$S_{k-1}^i$とする．$X_k$の変換行列$S_k^i$と$T_k^i$は$S_{k-1}^i$と$T_{k-1}^i$から以下のように生成される．

$$S_k^i = \begin{pmatrix} S_{k-1}^i & 0 \\ 0 & 1 \end{pmatrix} \qquad T_k^i = \begin{pmatrix} I_{i-1} & 0 & 0 \\ 0 & 0 & 1 \\ 0 & I_{k-1} & 0 \end{pmatrix} \begin{pmatrix} T_{k-1}^i & 0 \\ 0 & 1 \end{pmatrix}$$

X_kの正準形の候補は以下の式で与えられる．

$$X_{ck} = arg \max_{i=1,\ldots,k} CODE\left(\left(T_k^i S_k^i\right)^T X_k \left(T_k^i S_k^i\right)\right)$$

これは，X_k を正規化した後に $CODE$ を最大化するよう行列の要素を入れ替えたものを意味する．

4.4.3.節で得られた M_4 をこの式に代入すると，
$CODE(M_4)=321110$
となって最大の $CODE$ を持つ以下の正準形が得られる．

$$\begin{array}{c} \\ C \\ B \\ A \end{array}\begin{array}{c} \begin{array}{ccc} C & B & A \end{array} \\ \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} \end{array}$$

4.4.5.頻度計算

すべての正準形が求められたあと，グラフの集合からなるデータベースにアクセスして多頻度グラフパターンの候補の支持度を求める．

例えば，サイズが 3 の多頻度グラフパターンの候補の支持度を計算するとき，サイズが 1 の隣接行列からはじめて，他の隣接行列と組み合わせて結合させることにより，順次サイズの大きな誘導部分グラフを生成していく．このとき，多頻度グラフに含まれない隣接行列は，他のどの隣接行列とも結合を行わない．最後に大きさが 3 の隣接行列が結合によって作られれば，その正準形のカウンタを 1 つ増やす．

図 4.n の 3 つの隣接行列を入力とし，最低支持度を 60%と指定したとき，以下の 6 つの隣接行列が多頻度グラフとして出力される．

$$\begin{array}{c} \\ A \end{array}\begin{array}{c} \begin{array}{c} A \end{array} \\ \begin{pmatrix} 0 \end{pmatrix} \end{array} \qquad \begin{array}{c} \\ B \end{array}\begin{array}{c} \begin{array}{c} B \end{array} \\ \begin{pmatrix} 0 \end{pmatrix} \end{array} \qquad \begin{array}{c} \\ C \end{array}\begin{array}{c} \begin{array}{c} C \end{array} \\ \begin{pmatrix} 0 \end{pmatrix} \end{array}$$

$$\begin{array}{c} \\ C \\ A \end{array}\begin{array}{c} \begin{array}{cc} C & A \end{array} \\ \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \end{array} \qquad \begin{array}{c} \\ C \\ B \end{array}\begin{array}{c} \begin{array}{cc} C & B \end{array} \\ \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \end{array}$$

$$\begin{array}{c} \\ C \\ B \\ A \end{array}\begin{array}{c} \begin{array}{ccc} C & B & A \end{array} \\ \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} \end{array}$$

さらに検出する多頻度グラフの最低サイズを 3 としてフィルタリングすることによって，以下のグラフのみに絞ることができる．これは前述の M_4 である．

$$\begin{array}{c} \\ C \\ B \\ A \end{array} \begin{array}{c} \begin{array}{ccc} C & B & A \end{array} \\ \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} \end{array}$$

4.4.6. 非完全一致のモデルクローン出力

前節で最終的に得られた隣接行列 M_4 を元に，一時的に除去していた完全一致のモデルクローンを構成するブロック群以外のブロックを復元し，非完全一致のモデルクローンとして出力する．

M_4 は，図 4.7 で *Sub*1, *Sub*2 に存在する “A-C-B” のパターンを意味している．そこで，*Sub*1, *Sub*2 内の “A-C-B” のパターン部からブロックを復元し，非完全一致のモデルクローンとして出力する．

非完全一致のモデルクローンとして出力される構造を図 4.11 に示す．

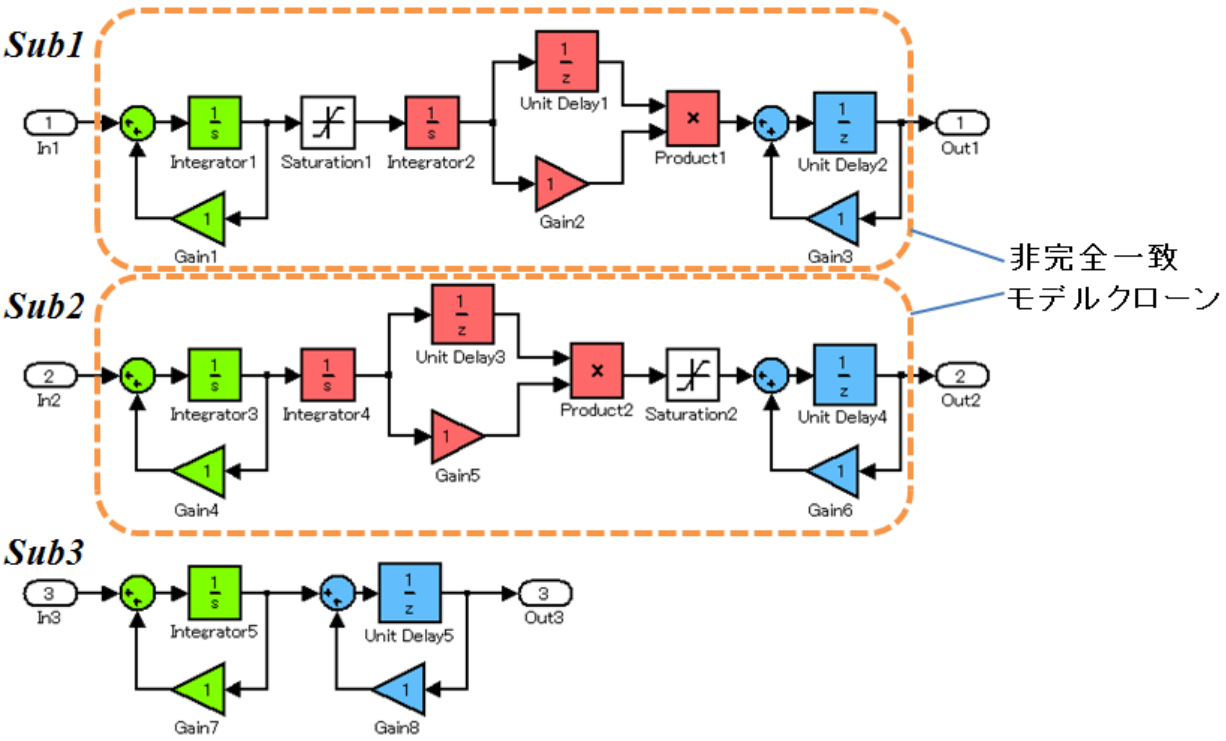

図 4.11 非完全一致のモデルクローン

第 5 章　適用実験

本提案手法の有効性を検証するため，総ブロック数 59，ブロック種 12，Subsystem 数 5 からなるモデルを対象に適用実験を行った．本章ではその適用実験の過程と結果を示す．

入力とした Simulink モデルの全体図を図 5.1 に，各 Subsystem の内容を詳細図として図 5.2 に示す．このモデルを対象に ConQAT を利用して完全一致のモデルクローンを検出したところ，4 種類 15 箇所の完全一致のモデルクローンが検出された．この結果を元に，完全一致のモデルクローンの位置関係を表したグラフを生成し，多頻度グラフを検出して，2 種類 4 箇所の非完全一致のモデルクローンを検出することに成功した．

完全一致のモデルクローンと，非完全一致のモデルクローンの構造と位置を図 5.3 に，完全一致のモデルクローンの位置関係を表したグラフを図 5.4 に，結果をまとめた表を図 5.に示す．

尚，AGM アルゴリズムによる多頻度グラフの検出には，汎用のデータマイニングツールである MUSASHI[7]を用いた．

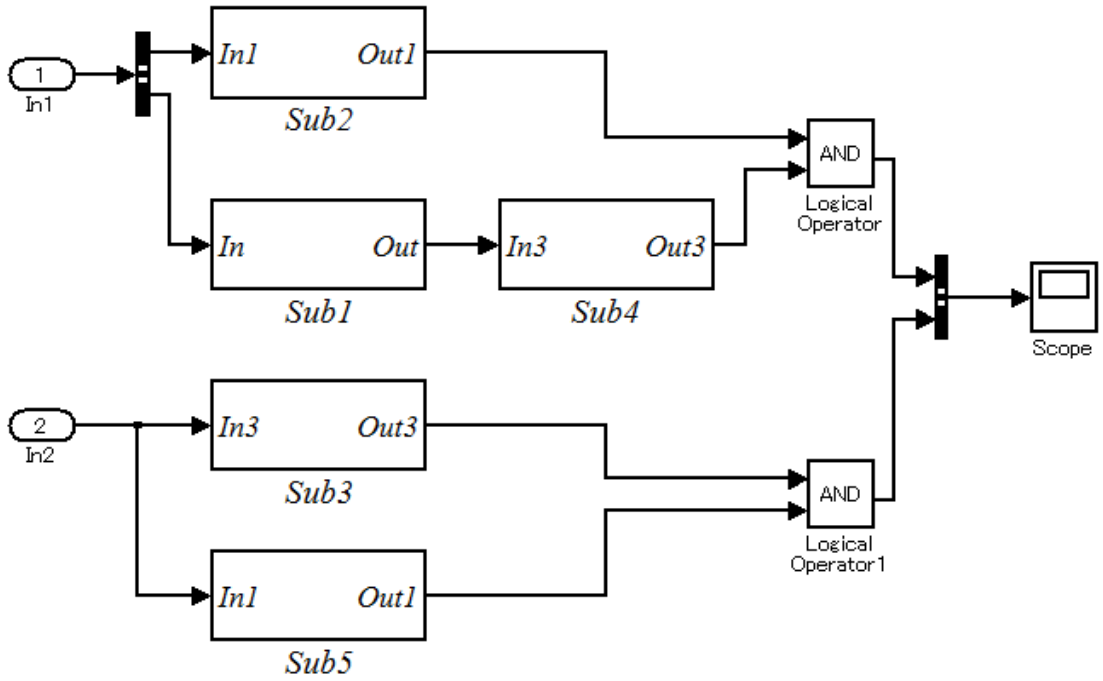

図 5.1 入力 Simulink モデルの全体図

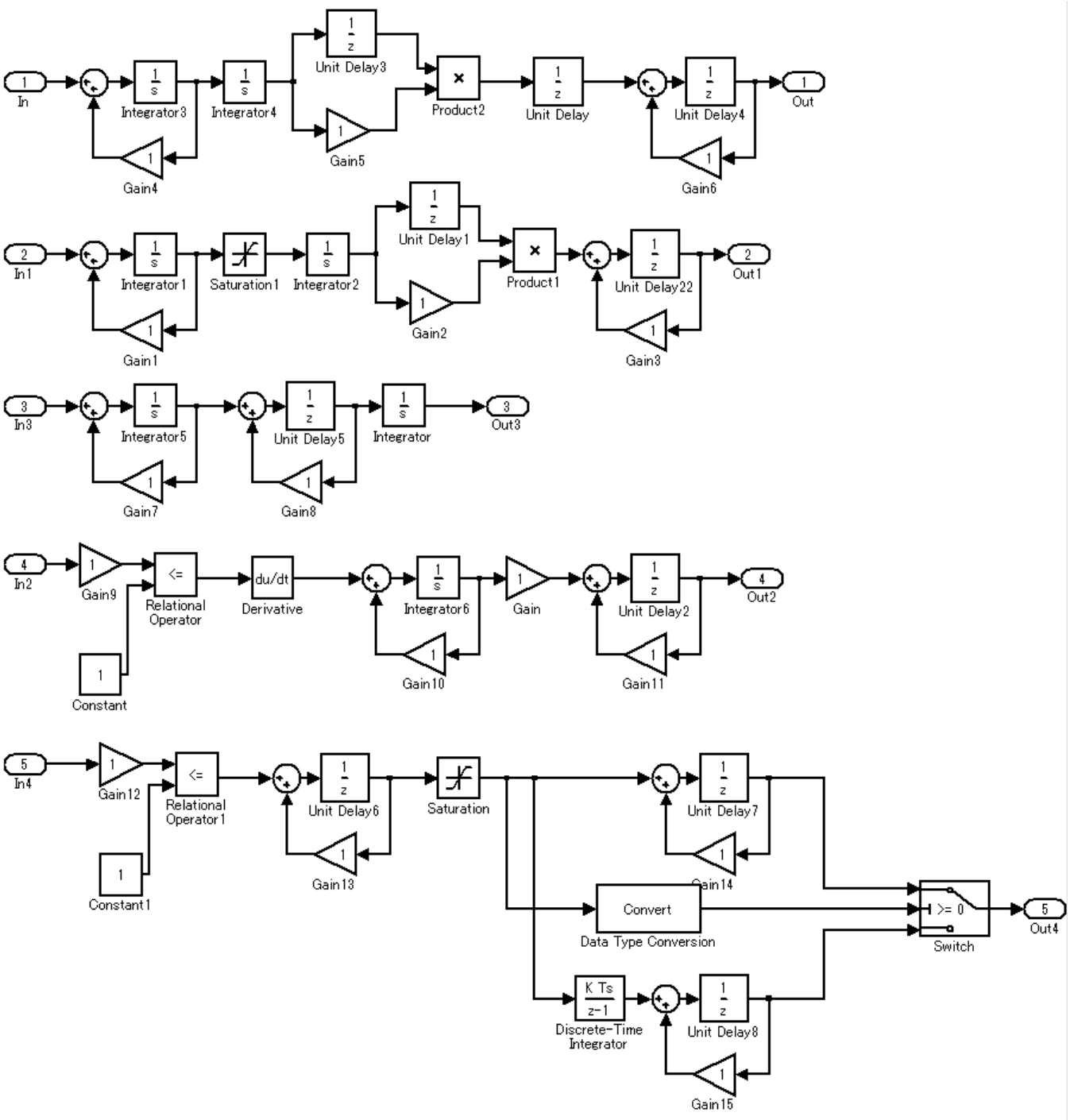

図 5.2 入力 Simulink モデルの詳細図

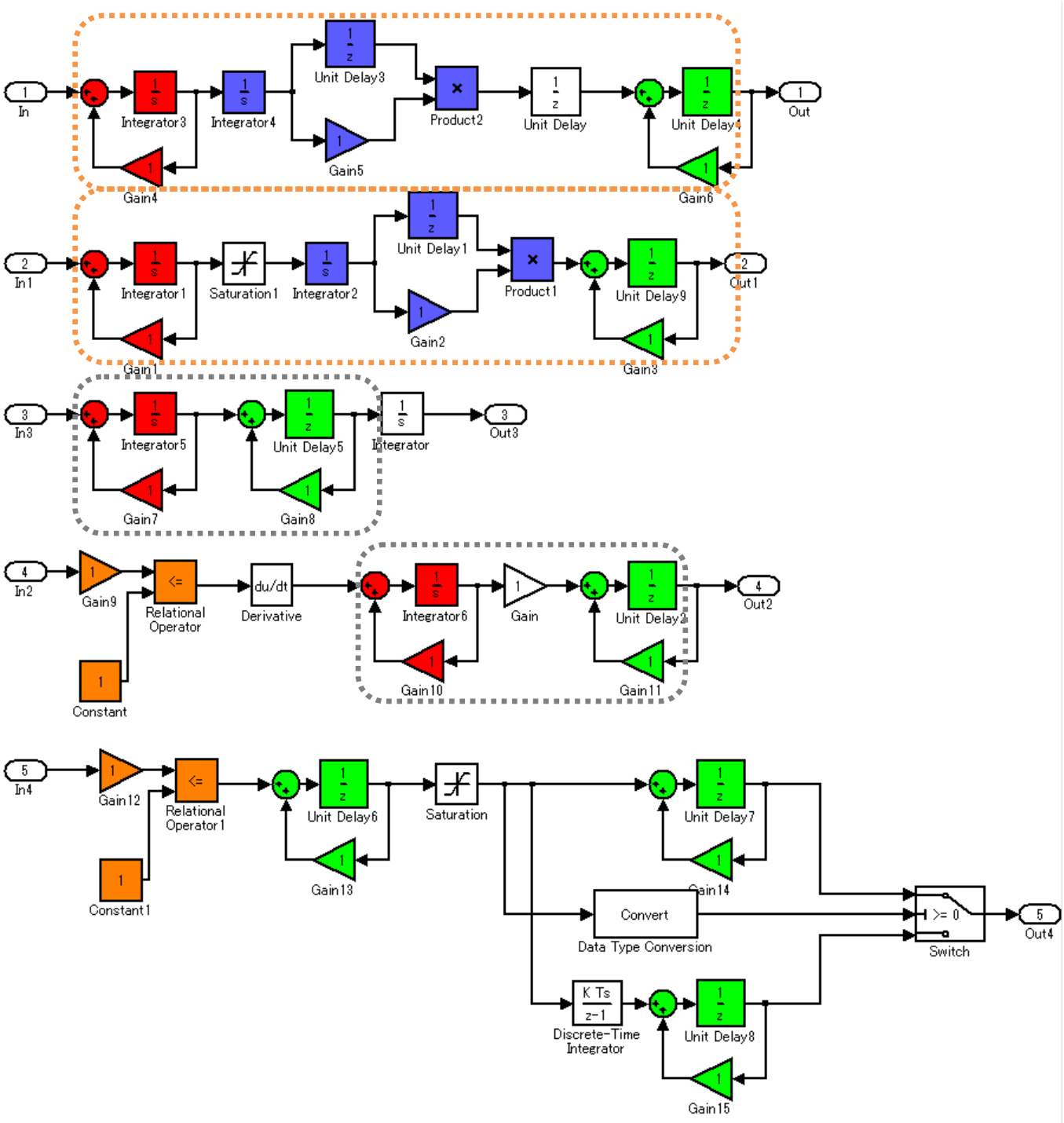

図 5.3 モデルクローンの構造と位置

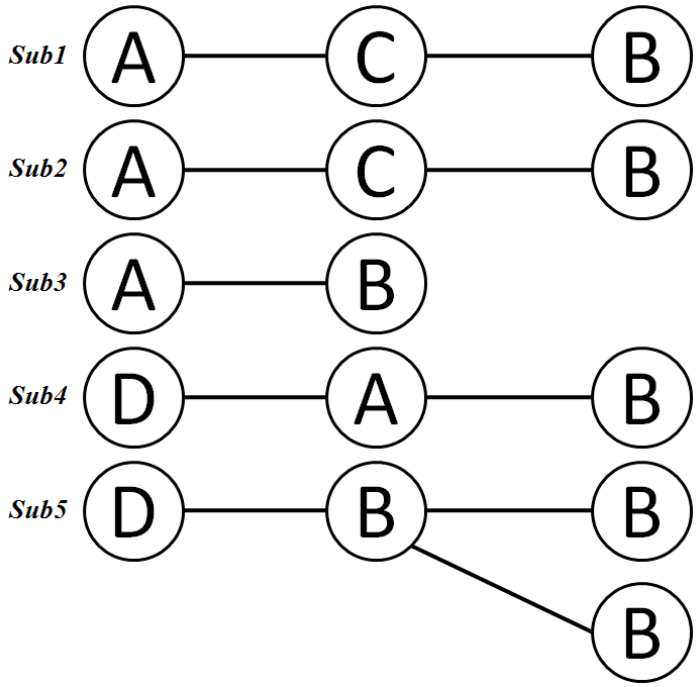

図 5.4 完全一致モデルクローンの位置関係グラフ

Simulinkモデル構成ブロック		完全一致モデルクローン		非完全一致モデルクローン	
種類数	総数	種類数	総数	種類数	総数
12	59	4	15	2	4

図 5.5 モデルクローンの検出結果

第 6 章 関連研究

Bakr Al-Batran らの手法

Bakr Al-Batran らは，既存のモデルクローン検出に検する研究[8][9][10]が，構造の一致のみをクローンとして定義している点を問題と指摘し，ブロックの配列や使用ブロックの種類，個数，結びつきなどといった構造には違いがあるものの，実現している機能，意味という視点では同一のものを Semantic Clone として定義し，これを検出する手法を提案している．[11]

Nam H. Pham らの手法

Nam H.Pham らは，完全一致のモデルクローンと，非完全一致のモデルクローンを検出する独自のグラフマイニングアルゴリズムを提案している．[12]

AGM アルゴリズムが隣接行列のサイズごとにグラフを幅優先で探索するのに対し，Nam H.Pham らの提案する完全一致のモデルクローン検出アルゴリズムではモデル上の特定のノードから開始し，そこから拡大できる構造を他箇所で検出できるかを調べるという深さ優先探索のアプローチをとっている．

Nam H.Pham らの提案する非完全一致のモデルクローン検出アルゴリズムでは，モデルを頂点と辺という要素で捉え，隣接行列の表現を使うというグラフマイニング一般的な手法を用いず，辺をベクトルとして表し，ベクトルの合成で類似度を評価するというアプローチをとっている．しかし，Nam H.Pham らの定義する非完全一致のモデルクローンは，あくまでブロックの配置やパラメータの違いをもって非完全一致としており，本研究で定義した非完全一致のモデルクローンとは異なる．

第 7 章 おわりに

7.1. 総括

本研究では，MATLAB/Simulink の Simulink モデルを対象として，従来モデルクローンとして扱われることのなかった非完全一致のモデルクローンを検出する手法を提案した.

まず完全一致のモデルクローンを検出し，完全一致のモデルクローンを構成するブロック群を単一のノードに置き換えると同時に，その他のブロックを一時的に除去して，完全一致のモデルクローンの位置関係をグラフ構造で整理した．次に，グラフマイニングのアルゴリズムである AGM アルゴリズムを利用して非完全一致のモデルクローンとなる多頻度グラフを検出し，そのグラフを元に非完全一致のモデルクローンとなっているブロック群を検出した.

適用実験として, サンプルとして用意した Simulink モデルに本手法を適用し, 非完全一致のモデルクローンが検出されることを確認した.

7.2. 今後の課題

大規模モデルへの適用実験

モデル分析の研究においては，実用的で大規模な実システムを対象に適用実験を行うことが重要である．しかし，本研究では残念ながらそのようなモデルを入手し，適用実験に使用することが出来なかった.

MATLAB/Simulink の開発元である Mathworks[13]のウェブサイトにて共有されいている複数のモデル[14][15]を対象に本提案手法を適用してみたが，本研究にて定義した非完全一致のモデルクローンを検出することが出来なかった．これは，使用したモデルのサイズが十分大きくなかったことが原因として考えられる他，ソースコードにおける非完全一致のコードクローンがコメントの挿入や引数の追加などによってもたらされるのに対して、Simulink モデルは物理的な計算式をブロックの構造で表すという特性から，想定していたような非完全一致の要素は計算式の意味が破壊されてしまうために存在し得ないという可能性も示唆しており，十分な規模を持った実システムを対象に研究を深めることが必要である.

モデルクローンのメトリクスによる類似度の定量的評価

本研究では，完全一致のモデルクローンを対象とした既存の提案手法では検出できなかった非完全一致のモデルクローンに焦点を当て，これを検出することに成功した．しかし，適用実験で扱ったモデルはあくまで試験用に作成されたモデルであり，恣意性を多分に含んでいるため，現実的なモデルクローンにおいて，どのような非完全一致性（差異）に着目すべきかは十分に考察できていない．

完全一致であるがゆえに類似度が 100%であることが自明な完全一致のモデルクローンとは異なり，非完全一致のモデルクローンは，その類似度を定量的に評価することが重要と考えられる．

モデルクローンを構成しているブロック群の一致度合いや，構成しているブロックの種類などの情報を元に，類似度を定量的に評価するメトリクスを定義することで，検出した非完全一致のモデルクローンをどのように扱うべきか，指標とすることができると考えられる．

ブロックのクラスタリング

本研究では，非完全一致のモデルクローンを検出する範囲として，Simulink モデルの Subsystem を利用した．中規模以上の Simulink モデルにおいては，Subsystem によるモジュール化は頻繁に行われており，機能としてのまとまりが強いブロック群としてみなすことができる．

しかし実際には，機能や，まとまることで 1 つの式による演算を実現しているようなすべてのブロック群が，すべて Subsystem としてまとめられているわけではない．

Simulink モデルのブロックを適切にクラスタリングする技術と組み合わせることで，設計上の意味合いがより強く反映された単位で，非完全一致のモデルクローンを切り出すことができるようになると考えられる．

謝辞

本研究を進めるにあたり，数々のご指導を頂いた早稲田大学大学院の鷲崎弘宜准教授，深澤良彰教授に深く感謝いたします.

また，共に研究に励み，様々な面でご協力いただいた鷲崎研究室，深澤研究室の皆さまに深く感謝いたします.

参考文献

[1] MATLAB/Simulink,
http://www.mathworks.de/products/simulink/

[2] ConQAT,
http://www.conqat.org/

[3] Juergens, E.; Deissenboeck, F.; Hummel, B. ,CloneDetective - A workbench for clone detection research" In 31th International Conference on Software Engineering (ICSE 2009), pp.603-606, 2009.

[4] Inokuchi, A., Washio, T., and Motoda, H, "An Apriori-Based Algorithm for Mining Frequent Substructures from Graph Data", In Proc. of the 4th European Conference on Principles and Practice of Knowledge Discovery in Databases(PKDD 2000), pp.13-23, 2000.

[5] Simulink Library - ConQAT
http://conqat.in.tum.de/index.php/Simulink_Library

[6] Agrawal, R. and Srikant, R. "Fast Algorithms for Mining Association Rules, in Bocca, J. B., Jarke, M., and Zaniolo, C. eds.", In Proc. of the 20th Very Large Data Bases Conference, pp.487-499, 1994.

[7] MUSASHI,
http://musashi.sourceforge.jp/

[8] Al-Batran, B. "Model-Based Clone Detection Using Normal Forms." Master's thesis, Technische Universitat Munchen, 2011

[9] Deissenboeck, F., Hummel, B., Jürgens, E., Pfähler, M., Schätz, B.: "Model clone detection in practice." In: International Workshop on Software Clones. IWSC '10, pp. 57–64, 2010.

[10] Deissenboeck, F., Hummel, B., Jürgens, E., Schätz, B., Wagner, S., Girard, J.F., Teuchert, S.: “Clone Detection in Automotive Model-Based Development.” In: International Conference on Software Engineering , 2008.

[11] Bakr Al-Batran, Bernhard Schätz, Benjamin Hummel, “Semantic clone detection for model-based development of embedded systems,” In Proceedings of the 14th international conference on Model driven engineering languages and systems. (MoDELS 2011), pp.258-272, 2011.

[12] Nam H. Pham, Hoan Anh Nguyen, Tung Thanh Nguyen, Jafar M. Al-Kofahi, Tien N. Nguyen, "Complete and Accurate Clone Detection in Graph-based Models", In 31th International Conference on Software Engineering (ICSE 2009), pp 276-286, 2009

[13] MathWorks,
http://www.mathworks.co.jp/

[14] Embedded Coder Robot NXT Demo,
http://www.mathworks.com/matlabcentral/fileexchange/13399-embedded-coder-robot-nxt-demo

[15] IEEE 802.11a WLAN model,
http://www.mathworks.com/matlabcentral/fileexchange/3540-ieee-802-11a-wlan-model