B-006

ロボット制御システム設計の UML アクティビティ図に対するペトリネットによる正当性検証

Verification of System Design for Robot Control in UML/Activity Diagram by using Petri Net

關屋 貴詞 † **和崎 克己** ‡
Takashi Sekiya Katsumi Wasaki

1 はじめに

ロボット制御システム設計などに用いられるモデル駆動開発は，生産性・再利用性の向上といった点で有用である．UML[1] を用いた上位設計は，レビュー工程に適している反面，記述規則に厳密なルールがないこと，記述に自然言語を含むなどの理由から記述に曖昧性を含み，構造・振る舞いの検証系も別途必要であった．一方，ペトリネット [2] に代表される形式記述法は，並列動作を伴う制御システムの検証能力が高いが，抽象度の問題により記述性が低い．

UML 図による設計をペトリネットのサブクラスへ変換することで構造的整合性を検証する手法としては，文献 [3]，[4]，[5] などが既存の研究として挙げられる．しかしながら，複数プロセス間の同期動作に未対応であること，使用可能なノードが限定されていること，システムの振る舞いに対する検証能力が不十分であることなど，課題も残されていた．

本研究では，設計と検証系との中継手段として，UML アクティビティ図の記述規則と変換後のネット構造条件を定義し，あるクラスのペトリネットへの変換と解析を提案する．システム全体のデータフローを図 1 に示す．文献 [3] では未定義である特殊なノードの記述規則を定め，アクティビティ内におけるハイパーリンクや複数プロセス間の同期動作，階層化表現を含む設計への対応の検討も行った．

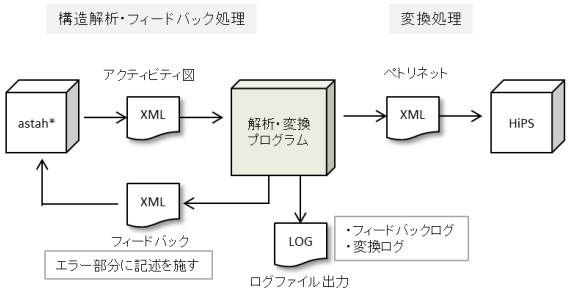


図 1: UML アクティビティ図からペトリネットへの変換データフローの概要

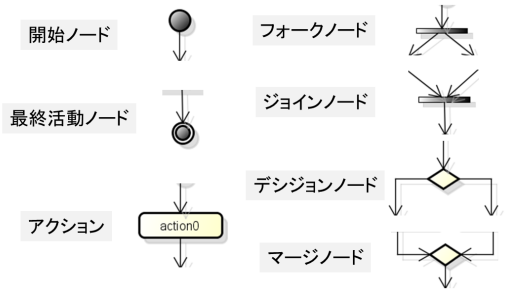


図 2: アクティビティ図の基本ノード

2 UML アクティビティ図と変換規則

2.1 UML アクティビティ図

アクティビティ図は，システムの振る舞いを表現する UML 図であり，開始ノードから最終活動ノードへのトークンの流れを表したものである．フォークや条件分岐を用いることで並列処理の記述や同期関係が表現可能であり，ペトリネットの表現と類似している．

基本的なアクティビティ図は，図 2 に示される 7 種類のノードによって記述される．開始ノードと最終活動ノードはそれぞれアクティビティの開始と終了を意味する．アクションは具体的な処理を表し，フォーク/ジョインノードによって並列・同期動作を表現し，デシジョン/マージノードによって条件分岐と合流を表現する．

具体的なアクティビティ図の例を図 3 に示す．図 3 の例ではアクティビティ開始後，まず action0 が実行され，その後フォークによって並列処理が開始する．左側のプロセスでは条件分岐によって action1 か action2 のどちらか一方のみが実行される．右側のプロセスでは action3 が実行され，ジョインによって左右のプロセスは同期合流し，最終活動ノードに至り終了する．

図 2 で挙げたノードの他にもいくつかのノードが存在する．本研究では，図 4 に示したコネクタ，シグナル送信ノード，シグナル受信ノード，フロー終了ノード，振る舞い呼び出しアクションを特殊ノードとして扱う．これらについては，次節で説明する．

2.2 記述規則

元のアクティビティ図と変換後のペトリネットとの対応関係は一意に保たれることが望ましい．アクティビティ図の各ノードの入出力エッジ本数について規則を設けることで，入出力の数によって元のアクティビティ図におけるノードの種類を区別することとした．基本的な

† 信州大学大学院工学系研究科,
Graduate School of Science and Technology, Shinshu University
‡ 信州大学工学部情報工学科,
Department of Information Engineering, Faculty of Engineering, Shinshu University

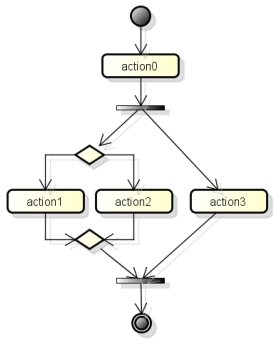


図 3: アクティビティ図の例

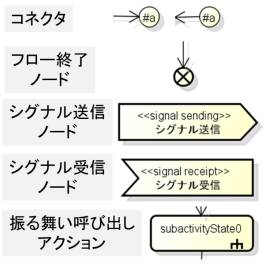


図 4: 特殊ノード

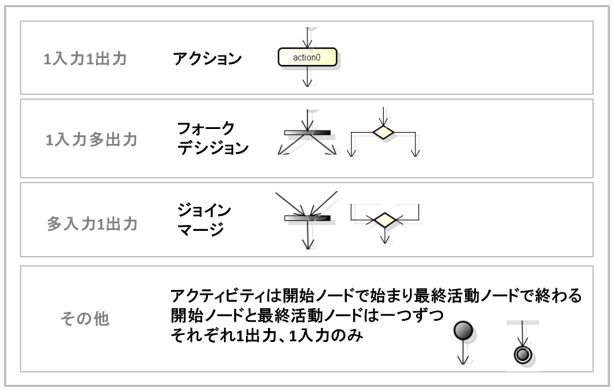


図 5: 基本ノードの記述規則

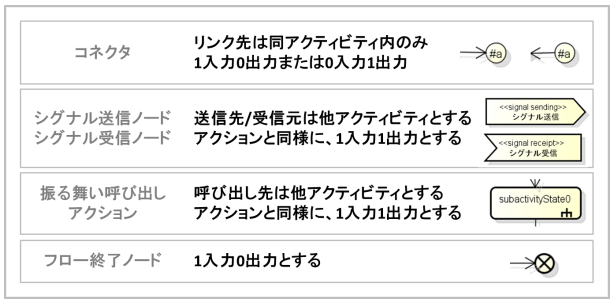
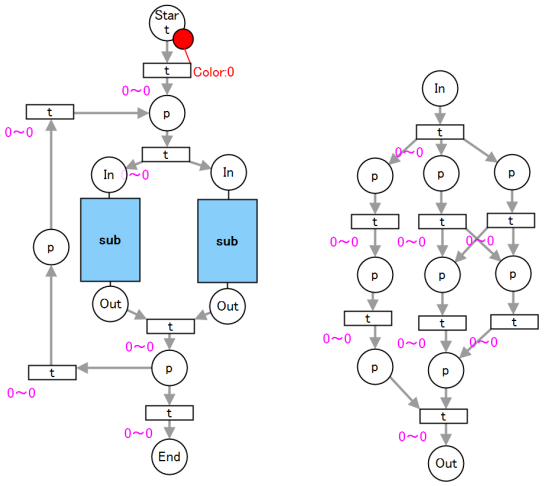


図 6: 特殊ノードの記述規則

ノードに対する記述規則を図 5 に示す．

図 6 は特殊ノードに対する記述規則である．コネクタは，ハイパーリンクによってフローの繋がりを記述するためのノードで，規模が比較的大きなアクティビティなどを視覚的にわかりやすく表現するために役立つ．コネクタは同ページ内のみでハイパーリンクを持つことが出来るとし，1 入力または 1 出力のみとする．シグナル送信/受信ノードは，ハイパーリンクによって他アクティビティとの同期動作を記述するために用いる．送信先/受信元は他のアクティビティであるとし，入出力数はアクションと同様に扱う．振る舞い呼び出しアクションは，サブルーチン的に他のアクティビティを呼び出すことが出来る．呼び出し先が終了次第，引き続き残りの処理が実行される．入出力数はアクションと同様に扱う．フロー終了ノードは，そのフロー自体は終了するが，アクティビティ全体としては終了せず引き続き動作を継続するというノードである．1 入力のみとする．特殊ノードを用いることで，複数プロセスの同期動作や階層化の表現が可能となる．



(a) メインページ (main)　　(b) サブページ (sub)

図 7: ペトリネットツールにおけるネット図の階層化表現

3　アクティビティ図から拡張自由選択ネットへの変換手法

3.1　拡張自由選択ネット

ペトリネットとは，プレースととトランジションの 2 種類のノードを持つ二部有向グラフであり，並列システムを視覚的にモデル化することが可能で，ネットの構造的解析・マーキング状態の挙動的解析の双方の検証能力を有する．ペトリネットはその構造により，自由選択ネット (FC ネット)，拡張自由選択ネット (EFC ネット)，非対称選択ネット (AC ネット) などのサブクラスに分けられ，FC ネットは EFC ネットに包含され，EFC ネットは AC ネットに包含される．EFC ネットは競合構造と同期構造との共存が可能なサブクラスであり，任意のプレースの組が出力トランジションを共有しているならば，これらのプレースの出力先トランジションは全て等しいという特徴を持つ．一般にペトリネットの解析は困難であるが，ネット構造を EFC ネットに制限することで，検証が容易となる．

変換後のネット図の検証には，本学で開発されたペトリネット設計・検証ツール [6] を使用する．図 7 はツール上におけるネット図の階層化表現を表したものであり，(a) がメインページ，(b) がメインページで呼び出されるサブページである．メインページの sub と書かれた青い四角形がサブページを表しており，前後のプレース In とプレース Out がサブページにおけるプレース In，プレース Out にそれぞれ対応している．プレースまたはトランジションを In/Out ポートに指定することで，他のページからサブページとして呼び出すことができる．サブページを使用することで，複雑な設計の階層化や，機能の部品化が可能となる．

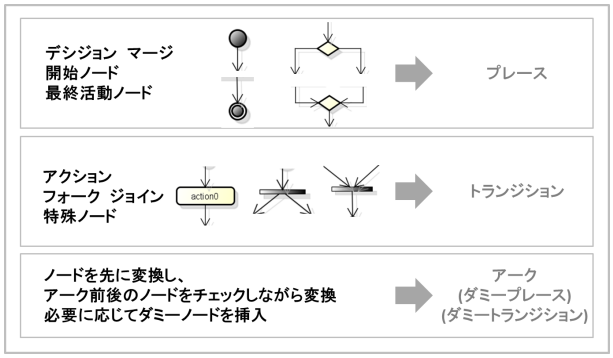


図 8: 基本ノードの変換規則

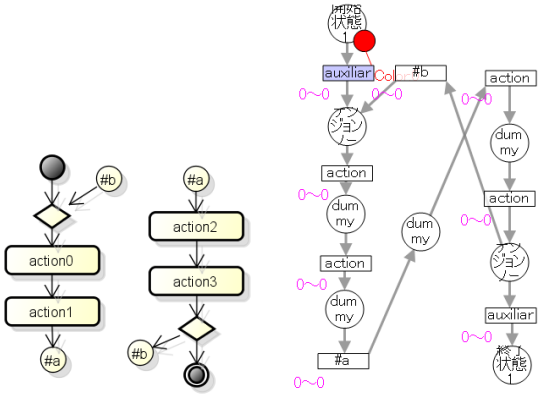


図 9: コネクタの変換例

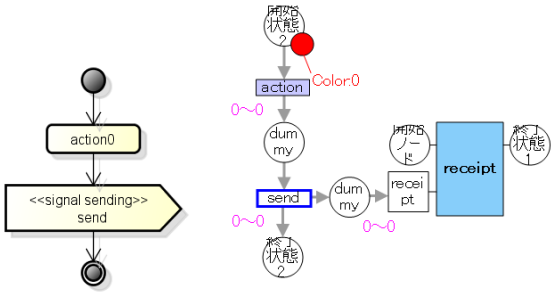


図 10: シグナル送信ノードの変換例

3.2 変換規則

3.2.1 基本ノードの変換

基本的な変換規則を図 8 に示す．開始ノードと最終活動ノードはそれぞれプレースに変換し，開始ノードから変換されたプレースにはトークンを一つ置く．アクションとフォーク/ジョインノードはトランジションに，デシジョン/マージノードはプレースに各々変換する．エッジはアークに変換するが，変換後のネット図においてプレースとトランジションが交互に連なるよう変換する必要があるため，前後のノードのタイプを調べ，必要に応じてダミープレース/ダミートランジションを追加しながら変換を行う．

3.2.2 特殊ノードの変換

特殊なノードの変換規則について図 8 に示す．コネクタは同ページ内でのハイパーリンクであるから，変換後のネット図では一つのトランジションとして扱うことで，リンク部分が実際に繋がったネット図を得ることができる．リンク先のコネクタを探索し，エッジの接続を張り替えた後，トランジションに変換し，他方のコネクタを削除することで実現する（図 9）．振る舞い呼び出しアクションは，呼び出し先の開始ノードを IN ポート，最終活動ノードを OUT ポートとしたサブページとして変換する．シグナル送信ノードは，送信先であるシグナル受信ノードを IN ポートとしたサブページへのソーストランジションとして変換する（図 10）．シグナル受信ノードは，受信元であるシグナル送信ノードを OUT ポートとしたサブページからのターゲットトランジションとして変換する．シグナルの送受信の関係を示しているだけであり，実際に連携して動作するわけではないので，変換時にはサブページ部分にトークンを配置する必要がある．フロー終了ノードはシンクトランジションとして扱う．

4 自動変換の適用例 (ロボット制御)

4.1 対象とするロボット

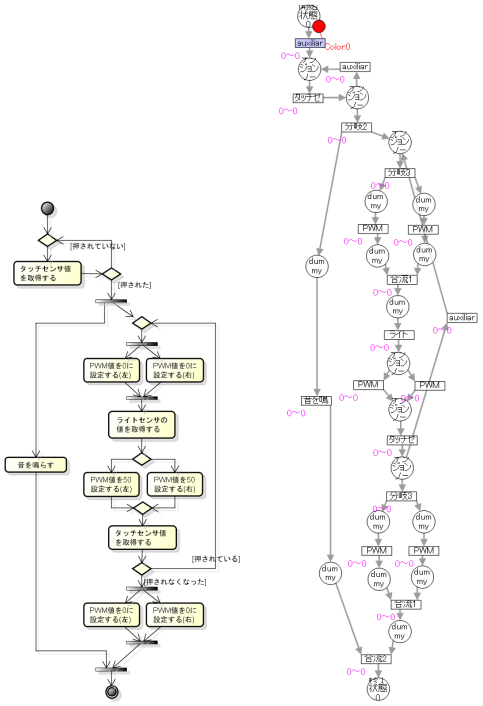
対象とするロボットとして，LEGO 社の Mindstorm NXT を用いた．リアルタイム OS とサーボモータ，複数のセンサ，各種ブロックを組み合わせることでさまざまなロボットを組み立てることが可能で，組込みシステム設計の教育にも利用されている．NXT には独自の API が用意されており，アクティビティ図におけるアクションに各関数を対応させることで,NXT 上で動作させることを想定したシステム設計を行い，構造解析・変換の試行を行った．

4.2 構造解析と変換の手順

アクティビティ図の XML ファイルについて，第 2.2 節で示した記述規則を満たしているか解析を行う．記述規則を満たしていれば，引き続きネット構造条件について解析を行う．EFC ネットであるか否かは，複数の出力をもつプレースと複数入力を持つトランジションとが接続されている場合に，その前後の関係によって判定できる．したがって，変換後のネット構造を調べるには，デシジョンノード→ジョインノードの構造の探し，その前後関係を調べればよい．アクティビティ内にデシジョン→ジョイン構造が含まれなければ，変換後のネット図は FC ネットに包含される．構造を含む場合，ジョインの他方の入力元ノードがデシジョン以外であれば，変換後のネット図は AC ネットに包含される．全ての入力元ノードがデシジョンの場合，デシジョンの出力先ノードが全て一致したならば，変換後のネット図は EFC ネットに包含される．一致しなかった場合，変換後のネット図は AC ネットに含まれない．

4.3 具体的なアクティビティ図からの変換例

図 11 は NXT で作成した 2 モータ，1 ライトセンサをもつライントレーサのアクティビティ図による設計と，変換後のペトリネットである．タッチセンサが押されると開始音を鳴らすと同時にライントレースを開始し，



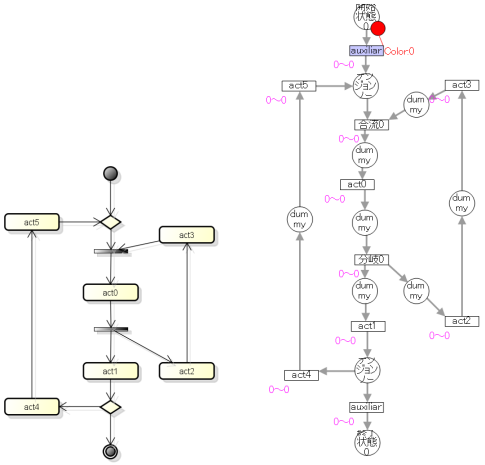
(a) ライントレーサのアクティビティ図　(b) 変換後のペトリネット

図 11: ライントレーサのアクティビティ図による設計と変換後のペトリネット

タッチセンサが再び押されなくなるまでトレースを継続する制御となっている．図 11(a) のアクティビティ図の各ノードは第 2.2 節で挙げた記述規則を満たしているので，ペトリネットへの変換処理が行われる．開始ノードとマージノードが接続されているので，変換後のネット図ではダミートランジション (auxiliary) が挿入されている．また，一つ目のフォークノードの左分岐先はアクションなので，変換後のネット図ではダミープレース (dummy) が挿入されている．このように，第 3.2 節の変換規則に則って変換が行われた．

4.4 シミュレーション結果と初期マーキング配置の問題

複数のアクティビティ図を用いてペトリネットへの変換を試行し，トークンの流れをシミュレーションした結果，アクティビティ図の記述規則や変換後のネット構造条件を満たしているにも関わらず，初期マーキングの問題により正常に動作しない例が存在した．図 12(a) の act0，act2，act3 を含むループのように，フォークノードで開始しジョインノードで合流する同期ループ構造は，変換後のネット図 12(b) でも同様に同期ループ構造となる．このループ部分のみを考えると，トランジションはすべて 1 入力 1 出力であるためトークンは増減せず，構造上は安全なため許可されるべきであるが，現在初期マーキングは開始ノードから変換されたプレースに 1 つ配置するのみとしているため，トランジション合流 0 が発火することができず，すぐに停止してしまう．これを回避するためには，ループ構造を解析し，必要に応じてループ上のプレースにトークンを配置する必要がある．



(a)　(b)

図 12: 初期マーキング検討の必要性

5 おわりに

アクティビティ図によるシステム設計の構造を解析することで，記述規則と変換後のネット構造条件の定義を満たしているかを検証し，エラー時の設計者へのフィードバックとエラーが存在しない場合のペトリネットへの変換，フィードバック時と変換時のログファイル出力を実行できることを確認した．エラー時には該当部分のノード/エッジに色を付けたアクティビティ図としてフィードバックを行った．特別に扱いを定義した特殊ノードについても，正常に変換することができた．

現在，サブページの変換は実装中であるため，実装が完了次第，シグナル送信ノードなどのサブページを含む特殊ノードの変換の対応を行う．また，NXT 用 API を組み合わせた処理をページ毎に設計しておき，振る舞い呼び出しアクションで利用することで，NXT 用ライブラリとして使用することを検討している．UML では各ノードにタグ付き値と呼ばれる値を持たせることができるので，具体的な数値を含むシステムのカラーペトリネットへの変換を目指したい．

謝辞

本研究の一部は科学研究費 (23500174) の助成を受けたものである．

参考文献

[1] 児玉：UML モデリング入門，日経 BP 社，2008
[2] T.Murata：Petri Net: Propaties, Analysis and Applications, IEEE, Vol.77, No.4, 1989
[3] H.Störrle：Semantics of Control-Frow in UML 2.0 Activities, VLHCC'04, 2004
[4] 山口，葛崎：ネット理論の応用-ワークフローネットとプログラムネット-, Fundamentals Review, vol3, no.3, pp.52-63, 2010
[5] 野村，安田，新川：CPN による UML モデル間整合性検証，IEICE Technical Report，2011
[6] 松山, 堀内, 和崎：階層化時間ペトリネット設計解析ツール HiPS へのタイムラインシミュレーション機能の実現；平成 22 年度電気関係学会東海支部連合大会, D4-8, 2010