離散型生産システムのモデル化と検証における UPPAAL の適応範囲について

宮崎大学 工学部 教育研究支援技術センター
〇高塚佳代子

はじめに

離散型生産システムの代表的なモデル検査ツールの一つである UPPAAL は、現実的規模の問題に適用可能であり、ツールとしての完成度も高い。しかし、通信様式がハンドシェーク型であるため「出力側が受理側の都合で待たされる」といった問題点があり、表現可能な並列動作には制限がある。一方、この種のシステムの挙動表現モデルとして以前開発した拡張時間ステートチャートは、イベントの生成と受理を完全に独立に扱う並列動作処理機構を持つため、離散型生産システムで生じる全ての並列動作をそのまま表現することができる。本研究では、UPPAAL の適応可能な対象システムの範囲を、拡張時間ステートチャートとの比較の下で明らかにすると共に、適応例のモデル化実験を通して UPPAAL での考え方に準じたモデル化方法について検討した。

キーワード：離散型生産システム　モデル検査　UPPAAL　通信様式

1. 背景と目的

近年の離散型生産システムの巨大化・複雑化を受け、著者らでは、拡張時間ステートチャート(:ETSC)とモデル検査手法に基づく動作検証手法の開発を行ってきた [1)]。モデル検査を行う代表的なツールとしては、UPPAAL、SPIN、SMV 等 [2),3),4),5)]、良く知られたものがある。何れのツールでも共通していることは、状態遷移の実行過程をシミュレーションできるということであるが、それぞれに異なる特徴が存在する。中でも、著者らの研究では、SPIN の持つ充実した検証機能に着目し、ETSC ベースでの手法のサポートツールとして有効利用する方法を提案してきた [6)]。

今回着目した UPPAAL は、時間オートマトンのネットワーク形式の挙動表現モデルを持ち、通信を伴う並行処理や排他処理が得意で、時間を扱うことも可能である。そのため、モデル化に関して言えば、実用的な問題にも十分利用できる。また、利用し易い GUI を持つため、実行や検証のメカニズムを十分理解しないままでも動かすことが出来る。しかしながら、UPPAAL の通信様式は「ハンドシェーク型」であるため、モデルの見かけ上の並列性をそのまま実現できるわけではない。つまり、表現可能な並列動作には制限がある。

一方、著者らが以前開発した ETSC は、UPPAAL 同様の時間オートマトンのネットワーク構造に加え、UPPAAL では表現できない階層性の記述も可能である。しかも、イベントの生成と受理を完全に独立に扱う本質的な並列動作の処理機構を持つため、あらゆる並列動作をそのまま表現することが可能である。しかし、ツールとしての完成度は UPPAAL に劣る。以上のことから、UPPAAL は、その適応可能範囲を見極め、その範囲内で使用する限りにおいては、便利かつ有用なツールであると言える。

以上を踏まえ、本研究の目的は次の通りとする。

- SPIN や拡張時間ステートチャートに基づく手法との比較を行いながら、UPPAAL の適応可能範囲を明確にすること。
- その適応範囲内の対象である FA 実験装置を具体例とし、UPPAAL のモデル化方法の検討及び提案を行うこと。
- 提案するモデル化方法の妥当性を実験的に確かめること。

2. モデル検査ツール UPPAAL について

UPPAAL はリアルタイムシステムのモデリング、検証、及びシミュレーションのための統合ツール環境である。

モデル

UPPAAL は、時間オートマトンのネットワーク形式の挙動表現モデルを持つ。図 1 で示すように、UPPAAL での時間オートマトンとは、滞在可能条件を満たしている間は任意の時間滞在できるロケーションと、時間経過しない遷移のアークから成る。ここで、滞在可能条件は時間変数を用いた時間制約式で表し、遷移条件(ガード)は、時間変数による時間制約、同期通信のためのイベントの入出力、及び、時間変数初期化のためのリセット式で表す。また、同期通信のためのイベントのやり取りは！と？の組で表す。

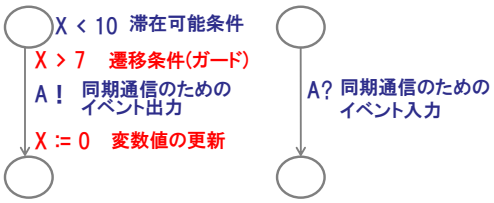


図 1　UPPAAL の時間オートマトン

モデル検査

UPPAAL のモデル検査では、“システムが満たすべき性質”を表す時相論理式(：CTL)を、モデルと共にツールへ入力すると、モデルが示し得る状態空間を網羅的に探索する検証方式を取るモデル検査アルゴリズムで性質検証し、その検証結果として true または false を出力する。また、結果が false の場合はその反例のパスも示す。

シミュレーション

シミュレーション機能によって、UPPAAL で作成された状態遷移モデルを直接実行し、動作の妥当性を確認することが可能である。シミュレーションの進行は、「メッセージ・シーケンス図」によりリアルタイムに表示される。このシーケンス図により、複数のプロセスの変化と、プロセス同士の通信の発生も理解しやすい。また、シミュレーション中は、状態やシステムで定義された変数の値の変化を確認できるため、シミュレーション中に起こった不具合などの原因を確かめることが容易に行える。

3. UPPAAL の適応範囲の検討

図 2 は、モデル検査を行うための手法やツールの違いを説明する図である。以下では、UPPAAL の適応範囲に最も大きな影響を与える通信様式の違いについて示し、UPPAAL の適応範囲を明らかにする。

	UPPAAL	SPIN	SMV	ETSC
動作記述モデル				
動作記述言語	時間オートマトン	Promela	SMV言語	時間ステートチャート
並行性	○	○	○	○
階層性	×	×	○	○
時間の扱い	○	×	×(△)	○
通信様式	1対1通信（ハンドシェーク型）	1対1通信（非同期型）	共有メモリ通信	ブロードキャスト通信
検証				
検証性質記述	時相論理式 (Timed Computational Tree Logic)			
可能性の検証	○	○	○	○
「正しさ」の検証（公平性・健全性 等）	×	○	×	×
GUI	○	○	×	×

図 2　モデル検査の代表的な手法と UPPAAL との比較

まず、UPPAAL の「1 対 1 通信様式(ハンドシェーク型)」だが、これは、メッセージの送信と受信が同期するため、メッセージの受け渡しが確実に行われるというメリットがある。しかし、送り手は受け手の都合で待たされるという問題があり、このことが、モデル上は並行動作するプロセス同士として示されているにも関わらず、実行段階ではその並列性が実現されないといった事態を引き起こす。

一方、SPIN は、UPPAAL 同様の 1 対 1 通信のチャネルを介した通信様式を持つが、キューやバッファリングによる非同期通信が扱えるため、送信側プロセスは受信側の都合で待たされることなく通信が可能である。従って、バッファの容量制限による取りこぼしに注意しさえすれば、並列性の実現能力は UPPAAL より高いと言える。

更に、ETSC では、1 対 1 通信ではなくブロードキャスト通信様式を取る。これは、理論上、モデルで示される並列性をそのまま実現することが可能であり、並列性の実現能力の高い手法といえる。具体的には、出力イベントは一律にイベントプールというイベント格納場所に登録され、各イベントは各々の発火可能なタイミングにその格納場所からシステム全体へまとめてブロードキャストされるという方式をとる。

以上のことから、全ての並列動作を扱える ETSC に対し、UPPAAL の適応範囲とは、ハンドシェーク同期に起因する待ち状態が現れないようなモデル化が可能な対象システムということになる。

[例 1] (ハンドシェーク型同期に起因する「隠れた待ち状態」の現れる例)

ここで対象とするのは、地理的に離れた生産拠点 A,B を持ち、拠点 B は拠点 A からのメッセージを受けて生産を行うといったシステムとなっている。具体的には、拠点 A、拠点 B はそれぞれ 3 工程から成り、拠点 A では第 1 工程終了後にメッセージを送り、そのメッセージを拠点 B では第 2 工程終了後に受理した後、第 3 工程を開始するようになっている。

実行結果は、図 3-(1)はハンドシェーク同期通信を行う UPPAAL による実行結果を表すガントチャートで、図 3-(2)はブロードキャスト通信を行う ETSC による実行結果を表すものとなっている。両者を比較してみると、拠点 A の方で UPPAAL の方の処理終了時間が ETSC の終了時間より遅くなっている。その理由は次の通りである。即ち、ETSC の方では、メッセージの出力と受理が独立に行えるため、生産拠点 A はメッセージ送信後、すぐに次の処理を行うことが出来るが、UPPAAL の方では、受け手である生産拠点 B がメッセージを受理しない限り送信側の A も次の処理を行うことができないということである。以上が、ハンドシェーク同期機構を持つ処理系の都合上起きる「隠れた待ち状態」の実態である。

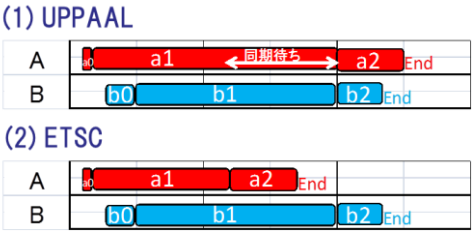


図 3　UPPAAL と ETSC の実行結果の比較

4. UPPAAL によるモデル化方法について

UPPAAL のモデル化は、時間システムの状態空間を、同じふるまいをする状態領域に分割して離散化した“プロセス”毎に記述することが一般的である。本稿では、以上のようなプロセス生成を、処理系の動作メカニズムによる処理の流れに沿った形で系統的に行うモデル化方法を提案する(後述(1))。更に、ハンドシェーク型同期に起因する「隠れた待ち状態」を回避する方法を考察する(後述(2))。以下、順に説明する。

(1)系統的モデル化方法

各ジョブに関し、当該ジョブを処理する上で必要な各種プロセスの定義を、次の手順で生成する。;

・まず、当該ジョブの処理手順を表す時間オートマトンを持つプロセスの定義を生成する。
・次に、ジョブの実行中に使用される各装置の操作手順を表す時間オートマトンを持つプロセスの定義を生成する。
・更に、装置使用上の競合解消ルール(優先関係や順序関係)を定義すると共に、プロセス間で必要なインタラクションも併せて導入する。

実行段階では、ジョブ発生の度に、上述の 3 種類のプロセスの実態を、順次生成し、ジョブ終了時に、当該ジョブに対して生成された全ての実態を消滅させる。以上の流れが、UPPAAL 処理系の動作メカニズムに合ったモデル化の流れとなっている。また、以上のような方法を採ることにより、対象システムにとって必要なプロセスとインタラクションの種類がもれなくピックアップでき、簡潔で分かり易いモデル化が可能となる。ただし、上述したように、UPPAAL の実行段階では、並行プロセスの数が不規則に変化していくといった実行形態になるため、システム全体の挙動の外部からの認識はし易いとは言い難い。このことは、UPPAAL に潜在的に存在する問題である。

(2)　「隠れた待ち状態」の回避

ハンドシェーク型同期に起因する「隠れた待ち状態」は、技巧的に回避することはある程度可能である。具体的には、そのような待ち状態が現れる部分に対し、メッセージを受信するプロセスが処理の流れを制御する「要求駆動型」としてモデリングすれば良い。しかし、その場合、実際のプロセスの振る舞いが直接的に表現されないため、モデルの分かり易さは損なわれるという問題が生じる。

以上のことから、UPPAAL の適応可能な対象とは、

・対象とシステムの振る舞いをそのまま記述しても「隠れた待ち状態」が現れない対象、或いは、
・モデルの分かり易さを損なわない範囲で技巧的な回避が可能な対象、

と言える。

5. 検証実験

今回の検証実験では、隠れた待ち状態が現れないような対象として、フローショップ型の FA 生産システムを対象に選び、モデル化実験を行った。実験の目的は、UPPAAL のモデル化方法の妥当性の確認である。なお、妥当性の確認方法として、次の 2 種類の方法を取った。;

・ETSC に基づく実機の動作結果との比較によるシミュレーションベースでの確認
・UPPAAL のモデル検査ツールによる動作検証による確認

5.1 対象システム

FA 実験装置は、図 4 で示すように、中央のターンテーブルを第 1 工程、右端のストッカー部分を第 2 工程とする、2 工程のフローショップ型生産システムに見立てたもので、各工程はそれぞれ 4 つの加工装置をもっている。パーツフィーダ上の処理対象は、ピック＆プレース装置によって搬入コンベア上に運ばれ、コンベアでの搬送後、アーム型のロボット（以降、“アームロボット”）によってターンテーブル上の装置 T1～T4 のいずれかに移される。ここで第 1 工程の処理が行われ、所定の処理時間が経過した後に、再度アームロボットによって搬出コンベア上に運ばれる。搬出コンベアでストッカーまで運ばれてきた処理対象は、押出シリンダーによって S1～S4 のいずれかの処理装置へ移される。ここで第 2 工程の処理が行われ、所定の処理時間経過後に製品が排出される。

製品と製造プロセスに関する情報

製品の種類は A,B の 2 種類とし、各装置での加工時間は、製品種に関わらず、T1→(30),T2→(50),T3→(60),T4→(40)とする。また、使用可能な装置や

ストッカーは、A→{T1,T2,T4}、B→{T3,T4}、及び、A→{S1,S2,S4}、Z→{S3,S4}とする。

制御方法

アームロボット、加工装置 T1～T4、及び、ストッカーS1～S4 の使用に関する制御としては、スケジューリング結果が持つ最適性を可能な限り保ちながら事象駆動での制御が可能な順序情報へ変換したものを制御方法として使用する。なお、今回使用する最適スケジューリングは、20～80 時間の間でばらつく発生間隔平均 47.1 時間でランダムに発生した 10 個の生産要求に対
する最適スケジューリングとする。

環境の不確定性について

生産要求発生間隔の不確定性は予め与える。それ以外にも加工時間の遅れ等の不確定要因も考えられるが、その種の不確定性は UPPAAL では扱いにくいため、今回は扱わない。

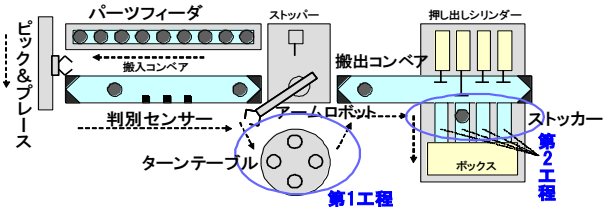


図 4　FA 実験装置

5.2 対象システムのモデル化

4 章の提案方法に従い、対象システムをモデル化した。具体的には、まず、不規則に発生する処理要求に対し、各ジョブの処理手順を時間オートマトンで定義し、
次に、ジョブの実行中に使用される各装置の操作手順を定義し、更に、装置使用上の競合解消ルール(優先関係や順序関係)を定義すると共に、プロセス間で必要なインタラクションを導入した。その結果、都合 19 種類のプロセスからなる並列オートマトンが得られた。その一部を図 5 に示す。

5.3 検証結果

シミュレーションベースでの検証結果

作成したモデルを UPPAAL のシミュレータによりシミュレーションし、その動作をトレースした。その結果、5.1 章で与えた製造レシピや制御方法や守りながら動作することを確認できた。以上のことから、提案したモデル作成法は妥当であることが確認出来た。

また、同じ工程を行う FA 実験装置の下で、モデルの正しさが既に確認済みの ETSC に基づく実機の動作と UPPAAL のシミュレータによる比較実験を行った。図 6-(1),(2)は、それぞれの実験の結果を示すガントチャートを表す。両者を比較し、ほぼ同じ結果を得られることが確認できた。この結果から、FA 装置では、UPPAAL の問題点である「ハンドシェーク型同期による待ち状態」は確かに現れないことが確認できた。

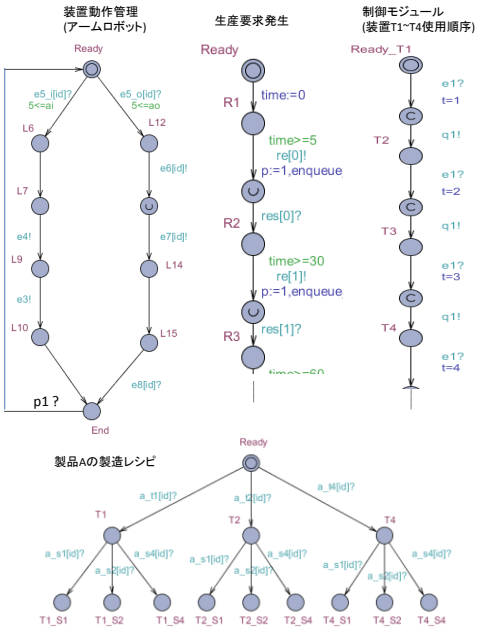
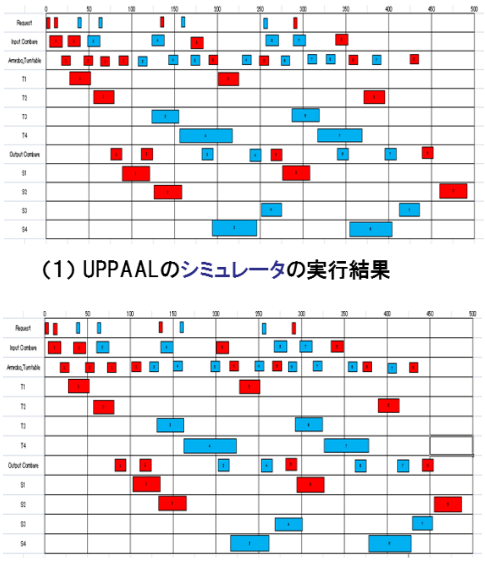


図 5　対象システムの UPPAAL によるモデル化



(1) UPPAALのシミュレータの実行結果

(2) ETSCに基づく実機の動作結果

図 6　UPPAAL のシミュレータの実行結果と
ETSC に基づく実機の動作結果との比較

モデル検査による検証結果

アームロボットのデッドロックや、4 つの加工装置の使用の公平性などの要求仕様に関する性質検証を行った。それらの結果がいずれも要求仕様を満足するという結果から、作成されたモデルの妥当性を確認した。以下、性質検証の実行結果を示す。;

(1) アームロボットのデッドロックに関する検証

A[ ]not(T1.L11andT2.L11andT3.L11 andT4.L11andArmrobo.L9) ⇒ true

⇔「全ての実行パスにおいて、常に、全装置が加工中であるにも関わらず、搬入コンベアからアームロボットがワークを運んでくることが起きないか？⇒起きない」

(2) 加工装置の選択の公平性に関する検証

E[ ] not (Armrobo.L10 and T1.L11)

E[ ] not (Armrobo.L10 and T2.L11)

E[ ] not (Armrobo.L10 and T3.L11)

E[ ] not (Armrobo.L10 and T4.L11) ⇒ true

⇔「特定の 1 つの装置(T1/T2/T3/T4)ばかりで加工を行うことが常に起こるようなパスは存在しないか？⇒ 存在しない」

以上の結果から、ハンドシェーク同期による「隠れた待ち状態」のない対象システムに関しては、提案するモデル化手法の正しさが確認できたといえる。

6. まとめ

・UPPAAL の通信様式であるハンドシェーク同期による「隠れた待ち状態」を持たない対象システムへ適用可能なモデル化手法を提案した。

・以下示す手順で、提案手法の妥当性を確認した。

(1) UPPAAL の適用が可能な（即ち、「隠れた待ち状態」を持たない）対象システムとして FA 実験装置を適応例とし、提案手法を用いてモデル化した。

(2)作成したモデルの正しさを、シミュレーションベース、及び、モデル検査による方法で確認した。

参考文献

1) Kayoko Takatsuka, Shigeyuki Tomita, Modelling of Discrete Manufacturing Systems having multiple jobs for Verification by Model-Checking, INDIN10, p1136-1141 (2010)

2) Gerard J. Holzmann: DEFINING CORRECTNESS CLAIMS, THE SPIN MODEL CHECKER, Published in Lucent Technologies Inc., Bell Laboratories (2004)

3) William Chan, Jon D. Reese, et al.: Model Checking Large Software Specifications, IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 24, NO. 7, JULY (1998)

4) Gerd Behrmann,Alexandre David,and Kim G. Larsen: A Tutorial on Uppaal, Department of Computer Science, Aalborg University, Denmark (2004)

5) Thomas Hune, Kim G. Larsen, Paul Pettersson: Guided Synthesis of Control Programs Using Uppaal, Nordic Journal of Computing (2001)

6) 高塚佳代子 冨田重幸: 離散型生産システムの挙動モデルにおける要制御部位の検出手法（－SPIN を活用した動的検出法の提案－）, 第 47 回 DES 予稿集 (2009)