博 士 論 文

時間，資源の制約を持つビジネスプロセスの形式検証

綿 引 健 二

主指導教員　平石 邦彦

北陸先端科学技術大学院大学
情報科学研究科

平成 26 年 3 月

要 旨

IT システムの開発においてビジネスプロセスのモデル化は非常に重要であり，その設計段階においてプロセスの正しさや妥当性を十分に検証しておく必要がある．特に，時間や資源に関する性質はボトルネック問題のようなビジネスプロセスの典型的な問題に関連しているため，注意深く分析する必要がある．本研究では，時間および資源に関する制約を持つビジネスプロセスの性質を，モデル検査技法を用いて形式的に検証する手法を提案する．まず，ビジネスプロセスの標準記法 BPMN に対して時間および資源に関する制約を付加できるように拡張を加える．さらに，拡張した BPMN で記述されたビジネスプロセスを，モデル検査ツールで検証可能な時間オートマトンに変換する方法を示す．本手法によって，プロセスの実行順序に関する問題だけでなく，時間と資源に関連した問題を初期の設計段階において排除し，ビジネスプロセスの設計品質の保証に役立てることができる．

目 次

第 1章

はじめに

1.1　背景

IT システムの開発においてビジネスプロセスのモデル化はますます重要になっている [24]．IT システムの開発初期では，現状のビジネスプロセスの性質を分析して問題を明らかにし，それを改善するための新しいビジネスプロセスを検討・設計する．IT システムはその新しいビジネスプロセスを前提に企画・開発されるため，新たに設計されたビジネスプロセスが妥当かどうか，すなわち，ビジネスにおいて求められる性質を満たすかどうかは，IT システムの有用性に大きな影響を与える．業務の順序関係に矛盾があったり，要求される時間的な制約を満たせないビジネスプロセスを前提として設計された IT システムは何の役にも立たないだろう．また，近年ではワークフローシステムなど，ビジネスプロセスそのものを支援するシステムが増加しているうえ，BPM(Business Process Management) システムなどビジネスプロセスを記述したモデルを直接入力して動作するシステムアーキテクチャも登場している．従って，ビジネスプロセスモデルの正しさや妥当性は，その設計段階において十分に検証しておくことが求められている．

1.2　問題点と本研究の目的

ビジネスプロセスの性質は，業務の直接的な実行順序に関する側面のほか，業務実行にかかる時間や待ち時間などの時間に関する側面，複数の業務に共有されて競合を発生させる資源に関する側面によって特徴づけられる．時間や資源に関する側面は，ボトルネック問題などのビジネスプロセスにおける典型的な問題と深く関連している．例えば，業務実

行のために必要な時間や同時利用が可能な資源の量などが制約となってボトルネック問題を引き起こし，結果としてビジネスから要求される性質，例えば業務全体のスループット性能が満たせなくなる，などである．よって，ビジネスプロセスの設計時には，上記の 3 つの側面を考慮して注意深く性質を検証する必要がある．

しかし，時間や資源の側面を考慮する場合，ビジネスプロセスの様々な状況を考慮する必要があるため，すべての状況の性質を確認することは容易ではなくなる．例えば，各業務の実行時間が一定ではなく実行ごとに幅がある場合，各業務の実行タイミングによって様々な状況が生まれるため，性質を確認すべきケースは膨大になる．さらに，競合が発生しうる資源を考慮すると，さらに状況は複雑化し，ビジネスプロセスがとりうる状況を列挙することも容易ではなくなる．

ビジネスプロセスの検証には，一般的にレビューやシミュレーションといった手法が用いられる．レビューはビジネスプロセスの意味的な妥当性を低コストで確認できる有効な手法だが，前述した様々な状況を網羅的に分析することは難しい．シミュレーションは，ビジネスプロセスを実際に模擬実行して性質を確かめる手法だが，すべての状況を模擬実行することは現実的ではない．よって，本研究では，網羅性という点において限界のある従来手法を補完することを目的として，ビジネスプロセスの性質を機械的に検証するための手法を提案する．網羅性は，人命に関わるような危険な状況にならないことを保証する必要があるビジネスドメイン，例えば，医療や介護のようなドメインにおいては特に重要である．

1.3　アプローチ

モデル検査は，対象のモデルが与えられた性質を満足するかどうかを，網羅的に検証することができる形式的な検証手法である．一般的には，システムやプロトコルなどの設計仕様を対象として，それらの仕様に矛盾がないことを網羅的に検査するための技法である．本研究では，これをビジネスプロセスの性質検証に適用する方法を考える．

モデル検査技法によってビジネスプロセスを検証するためには，ビジネスプロセスの振る舞いをモデル検査が扱うことができる形式モデルで表現する必要がある．一方で，ビジネスプロセスはビジネスユーザにとって馴染みのあるチャートで記述されることが多いため，それらを形式モデルに変換する必要がある．また，ビジネスプロセスの設計段階では，ビジネスプロセスの構造や時間，資源の各種制約を繰り返し変更，調節して性質を満たす

ビジネスプロセスを構築していくため，ビジネスプロセスを形式モデルに容易に変換するための系統的な手法が必要である．

これまでにも，サーベイ文献 [34][13] において整理されているように，ビジネスユーザ向けのビジネスプロセス表記法をモデル検査可能な形式モデルに変換し，ビジネスプロセスの性質を検証する手法が提案されている．ビジネスプロセスの実行順序に関する性質の検証にフォーカスした手法 [15][45] や，それに加えて時間に関する性質を考慮した手法 [46][22][33] が提案されているが，資源の側面までを考慮にいれてビジネスプロセスの性質を検証できる研究は少なく，特にビジネスプロセスの標準記法である BPMN（Business Process Model and Notation）[36] のように比較的複雑な振る舞いの要素を含む記法に対する手法は確立していない．

本研究では，ビジネスプロセスの標準記法である BPMN を用いて記述されたビジネスプロセスを対象とし，これをモデル検査可能な形式モデルに変換して，時間および資源の制約を考慮した検証を行うための手法を提案する．形式モデルとしては，本研究ではビジネスプロセスの検証において重要な時間制約を自然な形で記述でき，実用的なモデル検査ツールが利用できる時間オートマトン [11] を選択する．まず準備として，BPMN を用いて時間と資源に関する制約を記述できるように BPMN の記法を部分的に拡張する．さらに，拡張した BPMN によって記述されたビジネスプロセスモデルから時間オートマトンへ系統的に変換するための方針と手順を提案する．具体的には，BPMN の構成要素ごとに対応する時間オートマトンのテンプレート（部品）を用意し，モジュール指向で時間オートマトンモデルを構成できるようにする．変換後に得られる時間オートマトンモデルは，対応するモデル検査ツール，例えば UPPAAL[4] 等を用いて性質を網羅的に検査することが可能となる．

これにより，ビジネスプロセスの実行順序に関する問題だけでなく，時間と資源に関連した重大な問題を初期の設計段階において排除し，ビジネスプロセスの設計品質を保証し，さらにはそれをもとにして開発される IT システムの品質向上に役立てることができる．本研究では，看護・介護ドメインの問題を例に本手法を適用し，時間と資源に関連する性質を網羅的に検証し，クリティカルな要件の保証に役立てることができることを示す．

1.4　論文の構成

本論文の構成は下記の通りである．2 章では，ビジネスプロセス検証の問題点を整理し，提案手法のアプローチの詳細を説明する．3 章，4 章では，それぞれ提案手法で用いるビジネスプロセスの標準記法 BPMN，モデル検査技法とモデル検査ツール UPPAAL について説明する．5 章では，BPMN を拡張した拡張 BPMN を提案し，6 章では，拡張 BPMN をモデル検査技法を適用可能な形式モデルに変換する手法ついて説明する．7 章では，2 つの例題に提案手法を適用した結果を説明し，8 章で提案手法を評価する．9 章では，提案手法の関連研究について述べる．10 章では，まとめと今後の課題について述べる．

第 2章

ビジネスプロセス検証の問題とアプローチ

本章では，ビジネスプロセスの検証に関わる問題を整理し，本研究で対象とする問題を明らかにする．さらに，問題を解決するための提案手法のアプローチについて述べる．

2.1　ビジネスプロセス検証の問題

2.1.1　ビジネスプロセスの検証

ビジネスプロセスとは，製品やサービスの提供など，ビジネスの目的を達成するために必要な一連の作業や活動を定義したものである．IT システムの開発初期では，現状のビジネスプロセスの性質を分析して問題を明らかにし，それを改善するための新しいビジネスプロセスを検討・設計する．新しいビジネスプロセスは，導入予定の IT システムの前提となり，IT システム導入後のビジネスが適切に実行できるかどうかに関わるため，設計段階においてビジネスプロセスを注意深くモデル化し，要求される性質を満たしているかどうかを確認しておく必要がある．

通常，ビジネスプロセスはフロー形式のチャートを用いてモデル化される．一般的なフローチャート記法や UML におけるアクティビティ図 [3] のほか，近年はビジネスプロセスの標準記法である BPMN（Business Process Model and Notation）[2] が用いられることが多い．このようなチャートでモデル化されたビジネスプロセスを対象に検証すべき観点は，以下の 3 つに分類することができる．

1. 文法的な正しさ

2. 論理的な正しさ

3. 意味的な（業務的な）正しさ

文法的な正しさの検証では，ビジネスプロセスを表現したモデルが記法の定める構文に従っていることを確認する．例えば，記法の定める要素のみで構成されているか，要素間の接続ルールに準拠しているか，などを確認する．比較的容易に機械的な検証が可能であり，通常はビジネスプロセスのエディタ等に構文チェック機能（シンタックスチェッカ）が実装されている．

論理的な正しさの検証では，プロセスやタスクがいつまでも終了しない（デッドロックやライブロック），特定のタスクがいかなるケースでも実行されない（デッドタスク），など，業務に依存しない一般的な誤りがないかを確認する．

意味的（業務的）な正しさの検証では，ビジネスプロセスで実現される業務において，要求される要件が満たされるかどうかを確認する．例えば，出荷は入金後に行われる（入金前に出荷されることは決してない），注文後 24 時間以内に必ず出荷される，などである．

本研究では，上記の 3 つの観点のうち「文法的な正しさ」はシンタックスチェッカ等の技術で保証されているとの前提に立ち,「論理的な正しさ」と「意味的な正しさ」の検証をを対象とする．

2.1.2 ビジネスプロセス検証の問題

ビジネスプロセスの性質を検証する際には，その振る舞いを決定付ける下記の制約を考慮して，ビジネスプロセスがとり得る状況を洗い出し，要求される性質が満たされているかを確認する必要がある．

1. 構造制約

2. 時間制約

3. 資源制約

構造制約は，ビジネスプロセスを構成するアクティビティの実行順序関係を規定する制約である．例えば，制御フローや分岐，プロセスの親子関係（包含関係）等の構造によってアクティビティの実行順序が決定される．

時間制約は，ビジネスプロセスの実行の際の時間に関わる制約である．ビジネスプロセスの時間制約には，タスクの実行経過時間制約（Time Duration），連続した 2 つのアクティビティの間の遅延時間制約（Time Distance），アクティビティがある時点より前に必ず開始するという制約（Forced Start Time），アクティビティがある時点より前に必ず終了するという制約（Deadline），ある動作が特定日付や曜日で指定される制約（Fixed Date Constraint）などがある [29]．本研究では，これらのうち代表的な 3 つの時間制約，アクティビティの実行経過時間制約（Time Duration），連続した 2 つのアクティビティの間の経過時間制約（Time Distance），アクティビティがある時点より前に必ず終了するという制約（Deadline）を扱う．なお，Forced Start Time 制約は，直前のアクティビティに Deadline 制約を指定することで同等の制約が表現可能である．

資源制約は，ビジネスプロセスの実行の際に必要とする資源に関する制約である．本研究では，ビジネスプロセスの実行に影響を与える資源競合が発生しうる共有資源に対する資源制約を扱う．

上記の 3 つの制約をすべて考慮する場合，ビジネスプロセスにおいて発生する可能性のある状況を人手で網羅的に分析することは難しくなる．構造制約によって複数のプロセスが並行して実行されることがあるうえ，時間制約によって実行タイミングに様々なバリエーションが生まれる．さらに，資源制約によって資源の競合が発生し，実行の待ち時間が発生することも考慮に入れなければならない．

例として，看護・介護ドメインのビジネスプロセスを考える．本例題は，仮想的な病院内における看護・介護業務をモデル化したビジネスプロセスであり，文献 [49] に記載の業務を参考としたものである．

図 2.1 は，病院内のフロアマップであり，病室や浴室等の配置を図示したものである．本例題では，以下のようなビジネスプロセスを分析対象とする．

> 患者は全部で 6 名，各 3 名ずつの 2 グループに分かれており，以下に示す順序で食事やリハビリ，入浴を行う．
>
> - 全ての患者は，決められた時刻（正午）に一斉に昼食を開始する．
> - グループ 1 の患者は，食事をとり終えた順に各々リハビリ室に移動し，リハビリを行って病室に戻る．
> - グループ 2 の患者は，食事をとり終えた順に各々浴室に移動し，入浴を済ませて病室に戻る．

また，各活動や移動にかかる所要時間，サポートに必要な看護師の数，専用室の利用制限は下記の通りであり，これらがビジネスプロセスにおける時間と資源に関する制約に相当する．

- 食事には 30 分から 60 分かかり，グループ 1 では看護師が各患者に 1 名ずつ付いてサポートする必要がある．
- 病室とリハビリ室の間の移動には 3 分から 5 分かかり，看護師が各患者に 1 名ずつ付いてサポートする必要がある．
- リハビリ室では最大で同時に 2 名の患者がリハビリすることができる．
- 病室と浴室の間の移動には 1 分から 5 分かかり，看護師が各患者に 2 名ずつ付いてサポートする必要がある．
- 浴室は最大で 2 名の患者が入浴することができ，看護師が各患者に 1 名ずつ付いてサポートする必要がある．

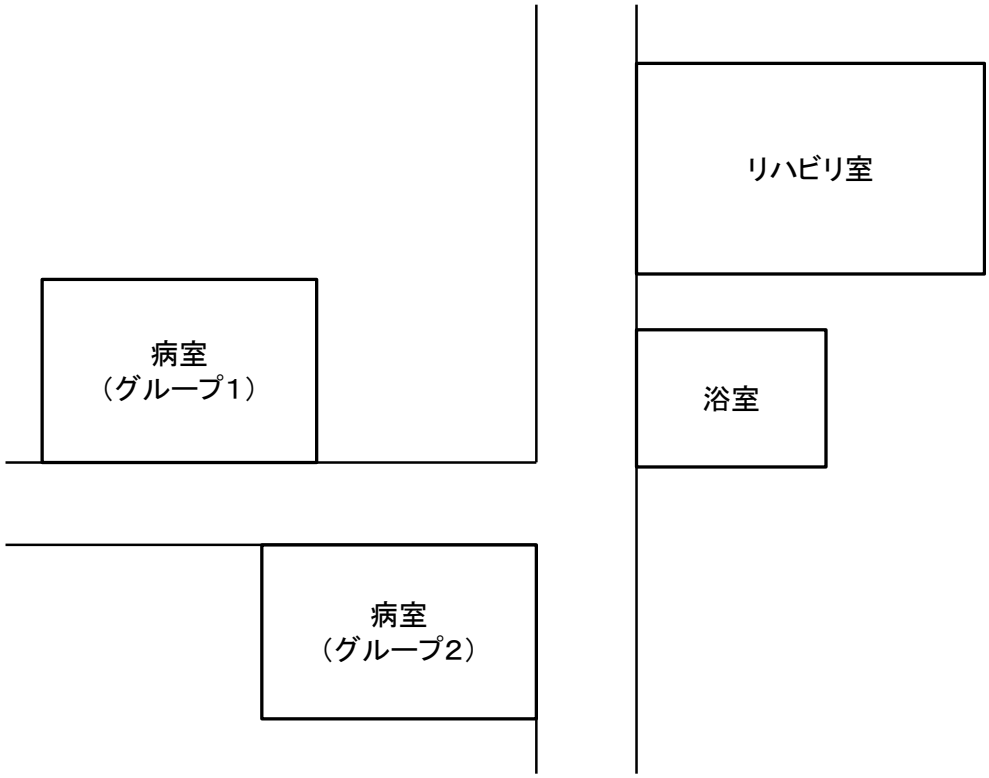

図 2.1: 病院内のフロアマップ

上記のビジネスプロセスにおいて，患者の安全上の要求から，以下に示す性質はいかなる状況においても満される必要があると仮定する．

1. 入浴を済ませた後に患者を浴室で 5 分を超えて待たせることがあってはならない．

2. 全員が食事を開始してリハビリまたは入浴を済ませて病室に戻るまでに 3 時間を超えてはならない．

ここで例えば，看護師の数を 5 人と設定し，上記の性質がいかなる状況においても満たされるかどうかを検証する場合を考えると，本例題は比較的シンプルなビジネスプロセスでありながら，時間や資源の制約条件によって様々な状況を考慮する必要があることがわかる．発生しうる状況を人手ですべて洗い出し，それらを一つ一つ分析をすることは容易ではないため，従来のレビュー手法やシミュレーションのような技術を使って網羅的に検証することは困難である．

本研究では，考慮すべき状況が複雑化する傾向にある，時間と資源の制約を持つビジネスプロセスを対象として，その性質を網羅的に検証する手法を提案する．これにより，ビジネスプロセスの実行順序に関する性質だけでなく，時間と資源に関連した重大な問題を初期の設計段階において排除し，ビジネスプロセスの設計品質を保証し，さらにはそれをもとにして開発される IT システムの品質向上に役立てることを目的とする．

2.2 アプローチ

モデル検査は，対象のモデルが与えられた性質を満足するかどうかを，網羅的に検証することができる形式的な検証手法である．数学を基盤として，IT システムの仕様記述，開発，検証を行う技術である形式手法の代表的な分野の一つであり，対象システムの振る舞いをすべて調べ上げて問題がないかを網羅的に確認する手法である．有限の時間の中ですべての振る舞いを洗い上げて確認するための理論やツールが研究されており，一般には，モデル検査は，ソフトウェアやプロトコルなどの動作仕様を対象として，それらの仕様の矛盾や問題の発見に応用されている．

一方，ビジネスプロセスは，業務における人やモノの振る舞いをモデル化したものであり，特定の動作仕様を持つ 1 つのシステムと捉えることができる．モデル検査手法を，前節で述べた時間，資源の制約を持つビジネスプロセスの検証に応用できれば，前述した検証の網羅性に関わる問題を改善できると考える．

ビジネスプロセスの性質検証にモデル検査技法を適用するためには，ビジネスプロセスの動作仕様を，モデル検査が入力として扱うことができる状態遷移系やプロセス代数といった形式モデルで記述しなくてはならない．一般に，ビジネスプロセスはビジネスユーザにとっての可読性が重視されるため，すでに述べたようにフローチャートやBPMNなどを用いて記述され，これを状態遷移系のような形式モデルに置き換えることは現実的ではない．よって，設計にはビジネスユーザとの親和性の高いBPMN等のチャートを用い，性質の検証の際に形式モデルへ変換する必要がある．また，ビジネスプロセスの設計段階では，ビジネスプロセスの構造や時間，資源の各種制約を繰り返し変更，調節して性質を満たすビジネスプロセスを構築していくため，ビジネスプロセスを形式モデルに容易に変換するための系統的な手法が必要である．

これまでにも，サーベイ文献 [34][13] において整理されているように，ビジネスユーザ向けのビジネスプロセス表記法をモデル検査可能な形式モデルに変換し，ビジネスプロセスの性質を検証する手法が提案されている．文献 [15][45] などでは，ビジネスプロセスモデルをそれぞれ形式モデルであるペトリネット，プロセス代数に変換する手法が提案されているが，時間や資源に関する制約は考慮されておらず，プロセスの実行順序関係に関する性質の検証に限定されている．文献 [46][22][33] などでは，時間制約を与えたビジネスプロセスを形式モデルに変換して分析を行う手法が提案されているが，資源の側面までを考慮にいれてビジネスプロセスの性質を検証できる研究は少なく，特にビジネスプロセスの標準記法であるBPMNのように比較的複雑な振る舞いの要素，例えばマルチインスタンスの実行やタイムアウトの例外処理など，を含む記法に対する手法は確立していない．関連研究の詳細については，9章に整理して示す．

図2.2は，本研究で提案する手法の概念図である．検証対象となるビジネスプロセスモデルは，本研究で想定しているビジネスプロセスの設計段階で標準的に利用される記法BPMN(ver. 2.0)[36] で記述されていることを前提とする．また，モデル検査を行うための形式モデルとしては，状態遷移系，プロセス代数，ペトリネットなどのモデル化技法が存在するが，本研究においては状態遷移系の一つ時間オートマトン [11] を選択する．他の技法でもビジネスプロセスをモデル化することは理論的に可能と考えるが，時間制約を変数として自然な形で記述でき，比較的高速で実用的なモデル検査ツールが存在することから時間オートマトンでのモデル化を行う．

提案手法は以下の2つで構成される．

(1)BPMN 記法の拡張 BPMNの標準仕様では，時間および資源に関する制約を十分に記

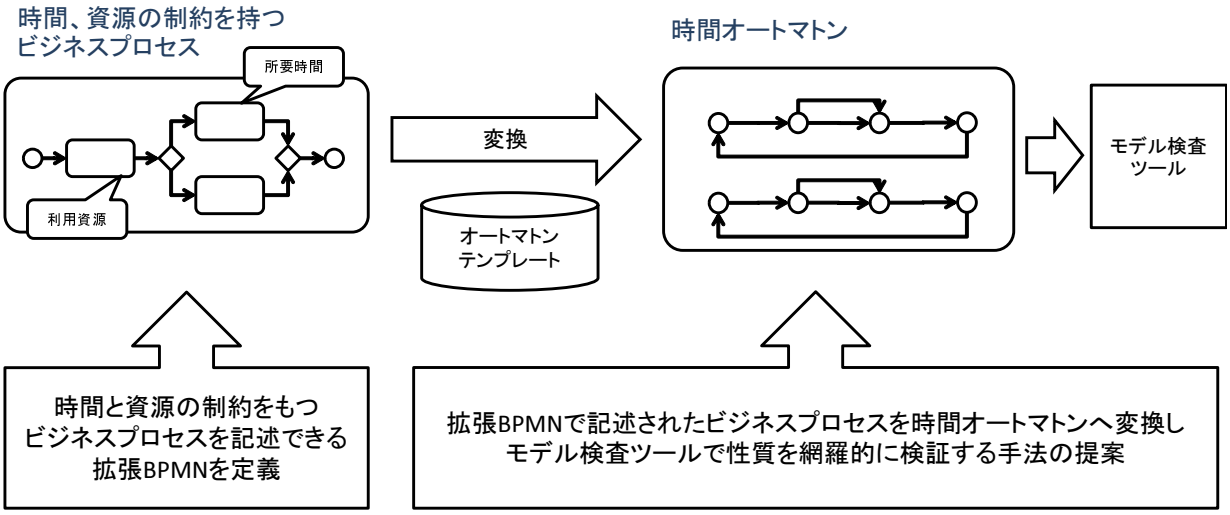

図 2.2: 提案手法の概念図

述することができないため，時間制約と資源制約を記述できるように記法を拡張した拡張 BPMN を定義する．拡張は BPMN が標準で備える拡張機能を用いる．

(2) 拡張 BPMN から時間オートマトンへの変換する手法 BPMN によるビジネスプロセスモデルが，タスクや分岐などの動作仕様をもつ要素で構成されていることに着目し，ビジネスプロセス中の各要素の組み合わせに応じて時間オートマトンモデルを構成できるようにする．具体的には，BPMN の要素の種類ごとに対応する時間オートマトンのテンプレートを用意しておき，要素の組み合わせ（接続関係）や動作仕様（時間制約や資源制約など）に応じて，時間オートマトンモデルを機械的に構成できるようにする．変換後の時間オートマトンモデルは，時間オートマトンを入力とするモデル検査ツールを用いて性質を網羅的に検査することが可能となる．本研究では，モデル検査ツールとして UPPAAL[10] を利用して性質の検証を行う．

第 3章

BPMN

3.1 BPMN

BPMN（Business Process Model and Notation）は，ビジネスプロセスのステップを表現するための標準図解記法である [2]．近年，ビジネスプロセスは組織内だけでなく組織をまたがる形で協調させることが可能となり，それによってビジネスプロセスのステークホルダーが増え調整がますます複雑になっている．BPMN の主要な目標は，ビジネスのステークホルダーが容易に理解できる標準表記法を提供することである．こうしたステークホルダーには，プロセスを開発，改善するビジネスアナリスト，プロセスの実装を受け持つ IT 開発者，プロセスを監視し管理する業務管理者が含まれており，彼らがビジネスプロセスに関する標準的なコミュニケーションを行えるように開発されている．これは，UML がソフトウェア・エンジニアリングの世界を標準化したのと同じ恩恵を，ユーザーにもたらすことになるものである．

3.2 BPMN の記法

BPMN では，ビジネスプロセスをフローチャートに類似した形式，すなわち，処理や分岐を表すフローオブジェクトとそれらの順序関係を表すシーケンスフローを用いたフロー形式でモデル化する．本研究では，図 3.1 に示す BPMN の主要要素を対象とする．これらの要素は BPMN の標準仕様のサブセットだが，基本的なプロセス構造をモデル化するために十分な記述能力を持ち，本研究で着目している時間および資源の側面に関連する要素を含んでいる．

フローオブジェクトは，イベント，ゲートウェイ，アクティビティに分類される．イベントはフローの開始，終了，中断を表し，それぞれ開始イベント，終了イベント，中間イベント（タイマー）と呼ぶ．中間イベント（タイマー）は，中断する時間を指定するための属性 $timeDuration$ を持つ．ゲートウェイはフローの分岐または併合を表し，並列したフローを作成する AND 分岐，複数のフローを同期化する AND 併合，複数の代替フローから一つのフローを選択する XOR 分岐，複数のフローのうち一つのフローを受け入れる XOR 併合を用いる．アクティビティはタスクとサブプロセスに分類される．タスクはプロセス内で実行される作業を表す原始的な要素である．サブプロセスはタスクと同じ図形だが，内部に詳細なプロセスを包含する複合要素である．サブプロセスは，マーカーや中間イベントを付加して特別な振る舞いを表現することができる．サブプロセス（タイムアウト）は，内包プロセスの実行時間が設定した時間を超えた場合に実行を中断し，中間イベントに接続されている代替のフローを進める．タイムアウト時間を指定するための属性 $timeDuration$ を持つ．サブプロセス（ループ）は，内包プロセスを指定された回数だけ直列に繰り返し実行する．サブプロセス（マルチインスタンス）は，内包プロセスを指定された数だけ並列に実行する．サブプロセス（ループ）およびサブプロセス（マルチインスタンス）は，それぞれ実行回数を指定するための属性として $loopCardinality$ を持つ．

3.3 記述例

2.1 に示した例題のビジネスプロセスを，BPMN を用いて記述すると図 3.2 の通りとなる．患者のグループごとに活動の内容が異なるためそれぞれをサブプロセスとして記述し，各グループでは 3 名の患者が独立して並列に活動するため，並列数 3 のマルチインスタンスとしている．

ただし，2.1 に示した所要時間や看護師数などに関する制約条件は，標準の BPMN では記述することができない．そこで 5 章では，時間と資源に関する制約をビジネスプロセスに与えられるよう標準の BPMN を一部拡張し，それらを考慮した性質検証が行えるようにする．

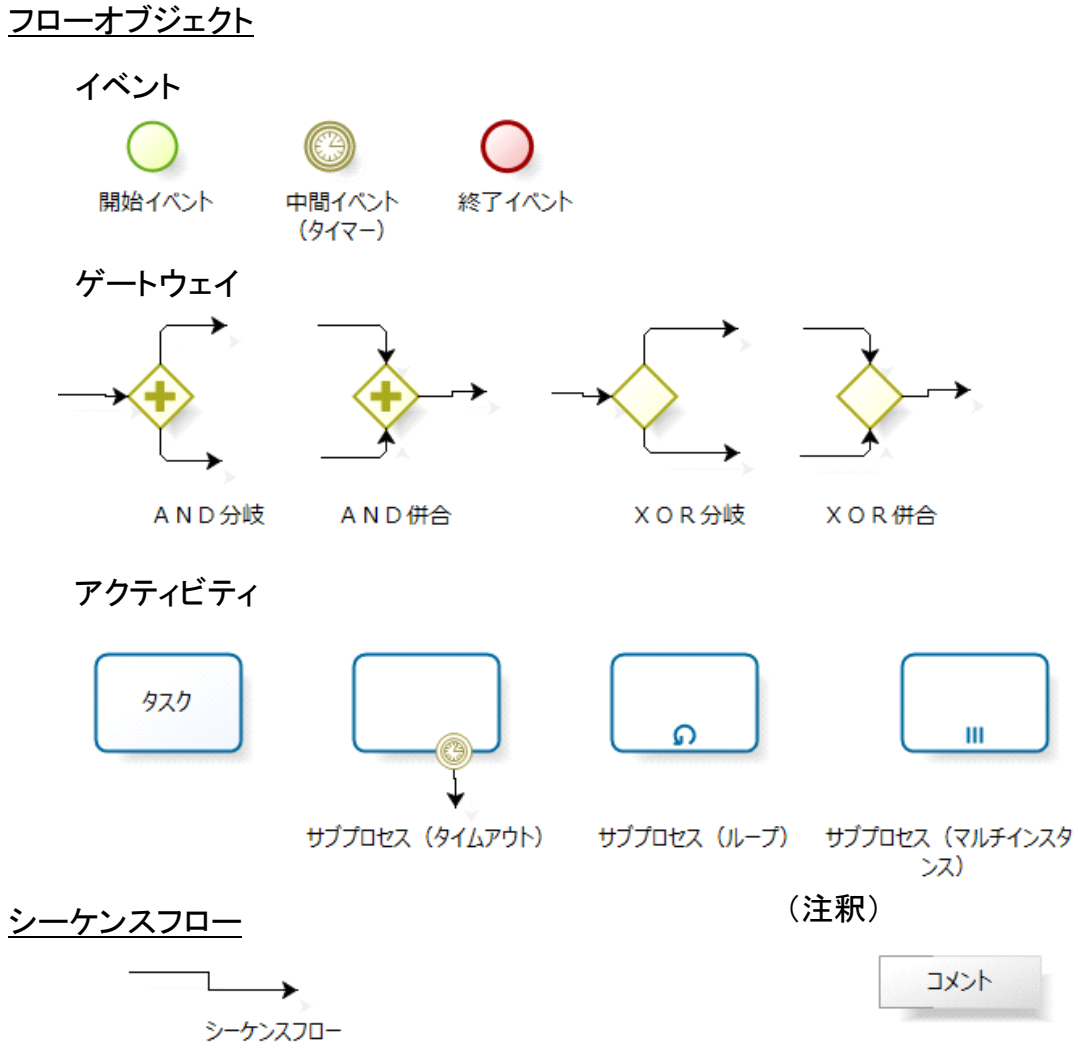

図 3.1: BPMN 主要要素

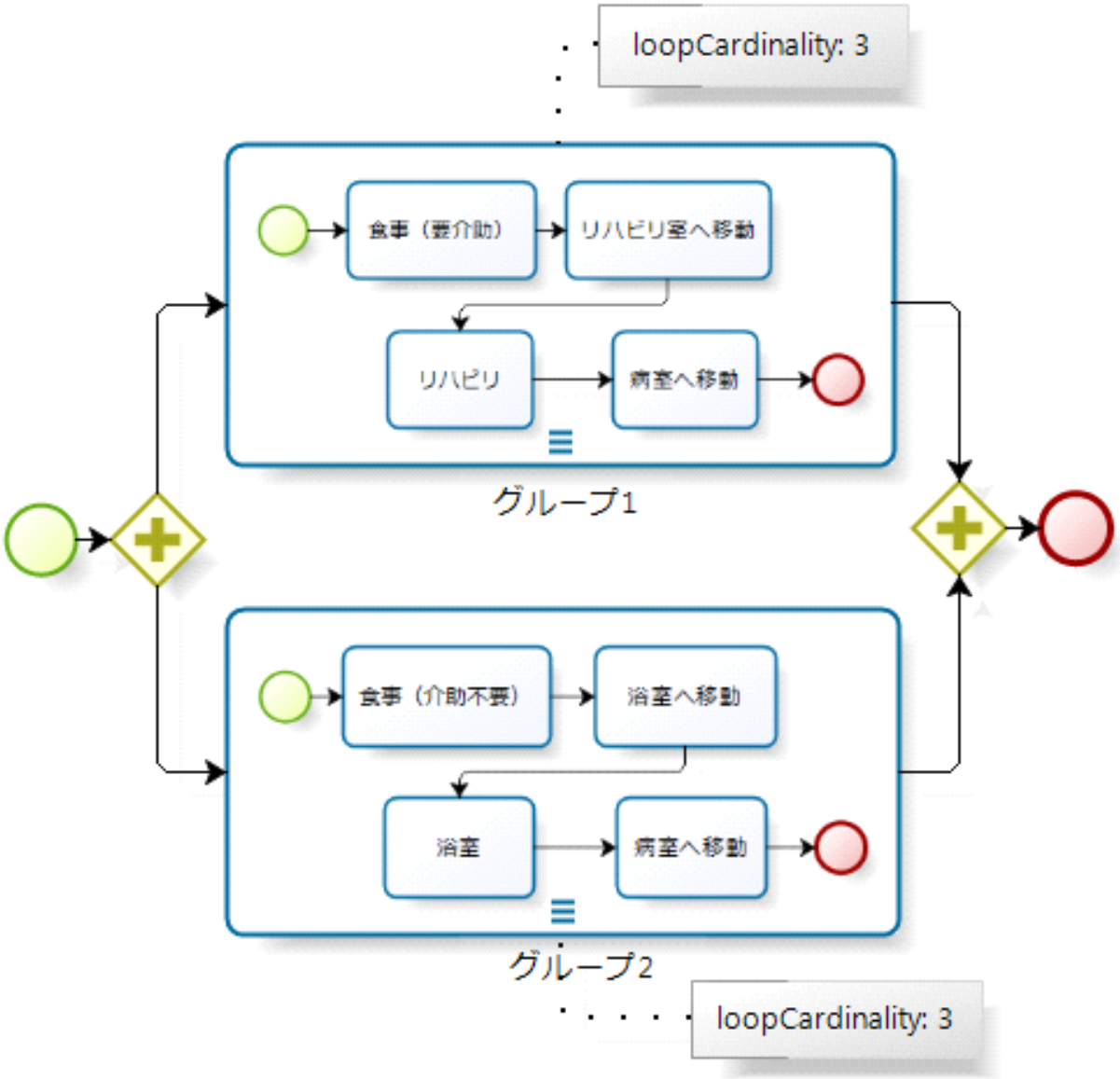

図 3.2: BPMN の記述例

第 4章

モデル検査

4.1　モデル検査技法

モデル検査は，対象のモデルが与えられた性質を満足するかどうかを，網羅的に検証することができる形式的な検証手法である．数学を基盤として，IT システムの仕様記述，開発，検証を行う技術である形式手法の代表的な分野の一つであり，対象システムの振る舞いをすべて調べ上げて問題がないかを網羅的に確認する手法である．有限の時間の中ですべての振る舞いを洗い上げて確認するための理論やツールが研究されており，一般には，モデル検査は，ソフトウェアやプロトコルなどの動作仕様を対象として，それらの仕様の矛盾や問題の発見に応用されている．

図 4.1 は，モデル検査の概念図である．モデル検査は，対象システムのモデルと検証性質を入力とし，対象システムが検証性質を満足するかどうかを網羅的に検証してその結果を返す．対象システムのモデルは，機械的に解析可能な形式モデルとして入力する必要がある．一般的には状態遷移系，プロセス代数，ペトリネットなどを用いてシステムを形式的に記述する．検証性質についても，機械的に理解可能な形式的な記述を行う必要がある．一般的には時相論理 [23] を用いた論理式で性質を記述する．

2 章で述べたように，本研究では対象システムのモデル化の方法として状態遷移系の時間オートマトン [8] を採用し，モデル検査ツールとしては UPPAAL[10] を用いる．UPPAAL では時間オートマトンを一部拡張した拡張時間オートマトンが入力の形式モデルとなる．検証性質は時相論理 TCTL[7] をベースとした UPPAAL の検証式で記述する．

以降の節では，時間オートマトン，UPPAAL による拡張時間オートマトンおよび UPPAAL の検証式について解説する．

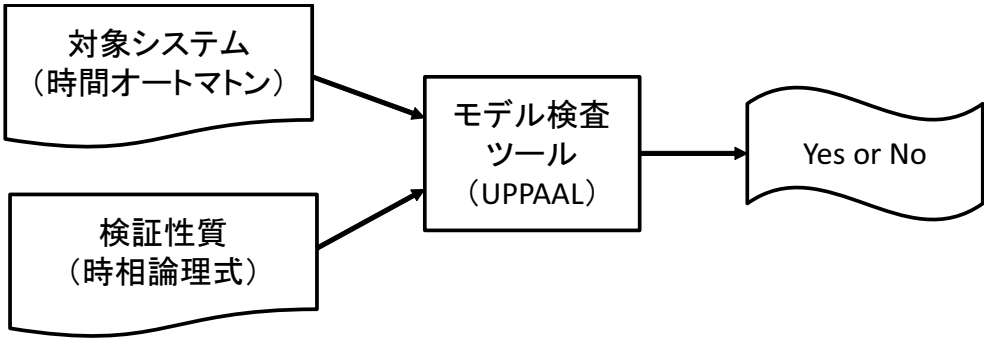

図 4.1: モデル検査技法の概念図

4.2 時間オートマトン

時間オートマトン [8] は，有限オートマトンの拡張であり，システムを離散的なイベントと連続的な時間経過の両方による状態遷移で記述するモデルである [1]．時間オートマトンでは連続的な時間経過を扱うため，一般にクロックによる同期のない物理的な制御対象との相互作用を自然な形で記述可能である．また，離散イベント間の時間制約を記述可能であり，時間的な挙動と離散的な状態遷移が相互に影響しあう動作を記述できる．

4.2.1 定義

時間オートマトンはクロック変数，ロケーション，不変式，ロケーション間の遷移，および，初期ロケーションの指定，から構成される．クロック変数は時間の経過と共に連続的に値が増加する実数型変数である．各ロケーションは，ロケーションの滞在中にクロック変数が満たすべき条件である不変式という属性を持つ．各遷移は，アクション，ガード条件，および，代入式という属性を持つ．ガード条件は遷移が実行可能となるための各クロック変数の現在値に対する条件式である．代入式は，遷移を実行後にゼロにリセットするクロック変数を指定するものである．

図 4.2 は時間オートマトンの表記例である．

[1] 本節は文献 [50] における時間オートマトンの解説を部分的に引用して構成したものである

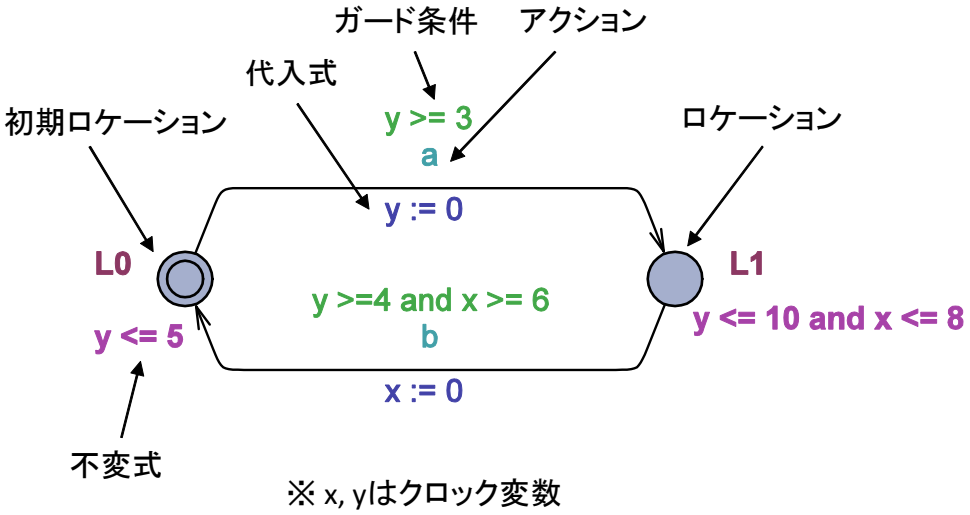

図 4.2: 時間オートマトンの表記例

4.2.2 操作的意味論

時間オートマトンの実際の状態はロケーションと各クロック変数の現在値の組で表現される．時間経過による遷移は，一つのロケーションに滞在中の遷移であり，ロケーションは変化せず，すべてのクロック変数は同じ値だけ連続的に増加する．ただし，不変式を満たす範囲内でしか時間経過は許されない．アクションによる状態遷移は，時間は経過せず瞬時にロケーションを変化させる遷移であり，現在のロケーションで実行可能な遷移のガード条件を満たすときに実行可能である．アクションによる遷移先状態では，クロック変数の代入式で指定されたものに関しては値がゼロにリセットされる．

図 4.2 の例では，ロケーション L0 の不変式（$y \leq 0$）とロケーション L1 への遷移上のガード条件（$y \geq 3$）より，y が 3 以上 5 以下の任意のタイミングで状態遷移が行われる．状態遷移の際には，代入式（$y := 0$）によってクロック変数の値はゼロにリセットされる．

4.2.3 並列結合

時間オートマトンでは，複数のサブシステムの時間オートマトンの並列合成として記述することにより，大きなシステム全体の動作を簡潔に分かりやすく表現することができる．時間オートマトンの並列合成はプロセス代数と同様の定義に基づく．まず，並行に動作する複数の時間オートマトンの組に対して，各時間オートマトンのロケーションの組を，並

列合成された時間オートマトンのロケーションとする．これは有限オートマトンにおける積オートマトンの構成法と同じ考え方である．次に，各時間オートマトン間の通信を表現するために次の概念を導入する．2 つ以上の時間オートマトンに共通に現れるアクションがあれば，そのアクションを同期アクションと呼び，それ以外を非同期アクションと呼ぶ．非同期アクションに関しては，並行に動作する各時間オートマトンにおいて独立に実行，すなわち，どれが先に実行されてもかまわない（任意の順序が可能）．一方，同期アクションに関しては，そのアクションを持つすべての時間オートマトンが同時に実行可能であるときのみ，そのアクションが実行可能である（すなわち，同期実行しなければならない）．

4.3 拡張時間オートマトン

4.3.1 時間オートマトンの UPPAAL 拡張

モデル検査ツール UPPAAL では，時間オートマトンに以下の特徴が追加されており，これを拡張時間オートマトンと呼ぶ [10, 11]．

- 定数．定数値は整数値をとり，変更はできない．
- 範囲指定付整数データ変数．上限値および下限値を指定できる整数変数．範囲指定は省略できる．ガード条件，不変式および代入式で利用できる．
- 1 対 1 同期通信．送信アクション “c!” の設定された遷移と受信アクション “c?” の設定された遷移が 1 対 1 で同期する．“c” をチャネルと呼ぶ．複数の同期ペアが存在する場合，ペアの選択は非決定的である．
- ブロードキャストチャネル．ブロードキャストチャネルによる同期通信では，一つの送信アクション “c!” と任意の数の受信アクション “c?” が同期可能である．同期可能な受信アクションはすべて同期し，同期可能な受信アクションが存在しなくても送信アクションがブロックされることはない．
- アージャントチャネル．同期可能になると時間の経過が許されないチャネル．
- アージャントロケーション．時間の経過が許されないロケーション．ロケーションの円の中に “U” と表記する．

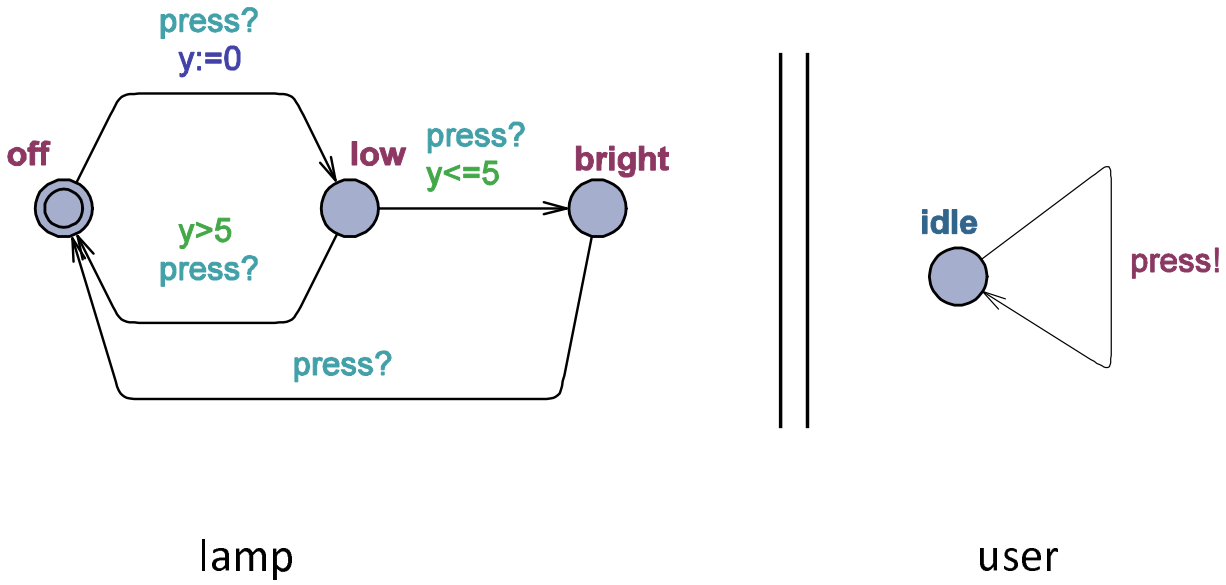

図 4.3: UPPAAL の検証モデル：拡張時間オートマトンのネットワークの例

- コミットロケーション．時間の経過が許されず，コミットロケーションからの遷移はコミットロケーション以外のロケーションからの遷移よりも優先される．ロケーションの円の中に “C” と表記する．

UPPAAL の検証モデルは，上記の拡張を加えた時間オートマトンを並列結合したものであり，これを拡張時間オートマトンのネットワークと呼ぶ．図 4.3 は，2 つの拡張時間オートマトンで構成されるネットワークの例である．一方はランプを，他方はランプの利用者をモデル化した拡張時間オートマトンである．ランプは，$off, low, bright$ の 3 つのロケーションを持つ．利用者がボタンを押す，すなわちチャネル $press$ によって両者が同期する（$press!$ が送信アクション，$press?$ が受信アクション）とランプは点灯（low）し，再び利用者がボタンを押すと消灯（off）する．ただし，利用者が素早くボタンを 2 度押した場合は明るく点灯（$bright$）する．利用者は任意のタイミングでボタンを押す可能性があり，全くボタンを押さないということもありうる．ランプのクロック変数 y は，利用者が素早くボタンを押したのか（$y \leq 5$），そうでないのか（$y > 5$）を判定するためのものである．

4.3.2 拡張時間オートマトンの抽象構文

BPMN から拡張時間オートマトンの変換を形式的に定義するための準備として，ここで拡張時間オートマトンの抽象構文を定義しておく [2] .

まず，拡張時間オートマトンの定義で用いる式をいくつか定義しておく．以降の定義において，$\mathbb{N}, \mathbb{Z}$ はそれぞれ自然数，整数の集合を示す.

まず，X をクロック変数の集合とし，$x, y \in X$ をクロック変数としたときにクロック制約式 $\varphi_{clk} \in \Phi(X)$ を以下のように定義する.

$$\varphi_{clk} ::= x \sim c \mid x - y \sim c \mid \varphi_{clk} \wedge \varphi_{clk}$$

$$\text{ただし,} c \in \mathbb{N}, \sim \in \{<, >, \leq, \geq\}$$

また，$v \in V$ を整数データ変数としたときに，整数式 $\psi_{int} \in \Psi(V)$ を以下のように定義する.

$$\psi_{int} ::= z \mid v \mid \psi_{int} \backsim \psi_{int}$$

$$\text{ただし,} z \in \mathbb{Z}, \backsim \in \{+, -, *, /\}$$

さらに，$\psi_{int1}, \psi_{int2} \in \Psi(V)$ を整数式としたときに，整数制約式 $\varphi_{int} \in \Phi(V)$ を以下のように定義する.

$$\varphi_{int} ::= \psi_{int1} \approx \psi_{int1} \mid \varphi_{int} \wedge \varphi_{int}$$

$$\text{ただし,} \approx \in \{=, <, >, \leq, \geq\}$$

最後に，X, V をそれぞれクロック変数，整数データ変数の集合としたときに，制約式 $\Phi(X, V)$ を以下のように定義する.

$$\varphi ::= \varphi_{clk} \mid \varphi_{int} \mid \varphi \wedge \varphi$$

$$\text{ただし,} \varphi_{clk} \in \Phi(X), \varphi_{int} \in \Phi(V)$$

次に，$x \in X, v \in V$ に対する値の代入式 $r \in R(X, V)$ を以下のように定義する.

$$r ::= x := 0 \mid v := \psi_{int} \quad \text{ただし,} \psi_{int} \in \Psi(V)$$

定義 1 (拡張時間オートマトン) 拡張時間オートマトンをタプル $A = (L, C, G, B, U, D, X, V, I, E, l_{ini})$ と表記する．ここで，

[2] 定義は文献 [19] を参考とした.

- L はロケーションの集合である．
- $C \subseteq L$ は，コミットロケーションの集合である．
- $G \subseteq L$ は，アージャントロケーションの集合である．ただし，$C \cap G = \varnothing$ である．
- B は，チャネルの集合である．
- $U \subseteq B$ は，アージャントチャネルの集合である．
- $D \subseteq B$ は，ブロードキャストチャネルの集合である．
- X は，クロック変数の集合である．
- V は，整数データ変数の集合である．
- $I : L \to \Phi(X, V)$ は，ロケーションにおいて必ず満たされるべき条件（不変式）を表す制約式を割り当てる関数である．
- $E \subseteq L \times B_{!?} \times \Phi(X, V) \times R(X, V)^* \times L$ は，有向枝の集合である．ここで，
  - $B_{!?} = \{a!|a \in B\} \cup \{a?|a \in B\} \cup \{\tau\}$ は，チャネル B に対応するアクションの集合である．$a!, a?$ は，それぞれチャネル B による送信側，受信側のアクション，τ は，チャネルによる同期を伴わない内部アクションである．
  - $\Phi(X, V)$ は，l から l' への遷移するための条件（ガード条件）を表す制約式の集合である．
  - $R(X, V)^*$ は，X に含まれるクロック変数と V に含まれるデータ変数に対する値の代入操作（代入式 $r \in R(X, V)$）のリストの集合である．

  l から l' の枝は，アクション α，ガード条件 φ，値の代入式のリスト $\vec{r}$ により $(l, \alpha, \varphi, \vec{r}, l')$ と表記できる．代入式の有限リスト $\vec{r}$ は，$\langle r_1, r_2, .., r_n \rangle$，特に代入を行わない場合は $\langle\rangle$ と表記することにする．
- $l_{ini} \in L$ は，初期ロケーションである．

記述例 1 (ランプの拡張時間オートマトン) 図4.3のランプの拡張時間オートマトンは，$A = (L, C, G, B, U, D, X, V, I, E, l_{ini})$ と記述できる．

$$
\begin{aligned}
L &= \{off, low, bright\} \\
C &= \varnothing \\
G &= \varnothing \\
B &= \{press\} \\
U &= \varnothing \\
D &= \varnothing \\
X &= \{y\} \\
V &= \varnothing \\
I &= \varnothing \\
E &= \{(off, press?, true, \langle y := 0\rangle, low), \\
&\quad (low, press?, x > 5, \langle\rangle, off), \\
&\quad (low, press?, x \leq 5, \langle\rangle, bright), \\
&\quad (bright, press?, true, \langle\rangle, off)\} \\
l_{ini} &= off
\end{aligned}
$$

4.4 検証式

検証対象のモデルが要求仕様（性質）を満たすことをモデル検査ツールで検証するためには，要求仕様を形式的な検証式として与える必要がある．形式的な記述には時相論理が用いられることが多く，UPPAAL では時相論理 TCTL(Timed Computation Tree Logic)[7] の部分論理を用いた検証式で性質を記述する [10].

4.4.1 TCTL

計算木論理 CTL[14] は，複数の経路を分岐した時系列と考えて性質を記述する分岐時相論理の一つである．TCTL[7] は CTL のリアルタイム拡張であり，構文は下記の通りである．ただし，f は原子命題またはクロック制約を表す.

$$\phi ::= f \mid \neg\phi \mid \phi \wedge \phi \mid AX\,\phi \mid EX\,\phi \mid E[\phi\ U\,\phi] \mid A[\phi\ U\,\phi]$$

X，Uは状態作用素であり次のような意味である．

- $X\ f$：経路の先頭の次の状態でfが成り立つ．
- $f_1\ U\ f_2$：f_2が成り立つまでf_1が成り立ち続ける．

E，Aは経路作用素であり次のような意味である．

- $E\ f$：現在の状態から分岐する経路のうち少なくともひとつの経路においてfが成り立つ．
- $A\ f$：現在の状態から分岐するすべての経路においてfが成り立つ．

また，CTL では次のような省略形の時相演算子を用いる．

- $AF\ f \equiv A[True\ U\ f]$：必ずいつかは$f$が成り立つ．
- $EF\ f \equiv E[True\ U\ f]$：いつかは$f$が成り立つ可能性がある．
- $EG\ f \equiv \neg AF\ \neg f$：$f$が常に成り立つ可能性がある．
- $AG\ f \equiv \neg EF\ \neg f$：必ず常に$f$が成り立つ．

4.4.2 UPPAAL の検証式

UPPAAL の検証式は，TCTL と同様にクロック制約を含む状態述語と時相演算子で構成される．ただし，TCTL とは異なり時相演算子を入れ子で用いることはできない．

状態述語はモデルの状態の集合を指定する述語であり，ロケーション名，制約式，および，これらを通常の論理演算子 (論理和，論理積，論理否定など) を用いて任意に組み合わせた式である．ロケーション名は，時間オートマトンが指定したロケーションに居る時に真となる述語である．モデルが複数の時間オートマトンの並列合成である場合は，その要素である個々の時間オートマトンが指定したロケーションに滞在している間は真となる．制約式は，時間オートマトンに定義されたクロック変数および整数データ変数を用いた条件式である．また，UPPAAL では *deadlock* という特別な状態述語が用意されており，すべての時間オートマトンにおいてロケーション間の遷移が不可能になった状態において真

となる．並列合成された時間オートマトンのロケーション名やクロック変数を指定する場合は，どの時間オートマトンのものかを明確にするために，それが属する時間オートマトンの名前を接頭語として付与する．例えば，図 4.3 のモデルにおいて，ランプが明るく点灯しており，クロック変数 y が 10 以内である状態を指定する熟語は，以下のように記述できる．

- $lamp.bright\ and\ lamp.y <= 10$

時相演算子は以下の 5 つが用意されている．ここで f, g は任意の状態述語である．

- $AG\ f$ （必ず常に f が成り立つ）
- $AF\ f$ （必ずいつかは f が成り立つ）
- $EG\ f$ （f が常に成り立つ可能性がある）
- $EF\ f$ （いつかは f が成り立つ可能性がある）
- $f \rightsquigarrow g$ （f が成り立てば，必ずいつかは g が成り立つ）[3]

時相演算子と状態述語と組み合わせることによって，モデル検査で検証される代表的な性質である到達可能性，安全性，活性を以下のように記述することができる．

到達可能性 「ある状態にいつかは到達する可能性がある」ことを表す性質．UPPAAL では $EF\ f$ で記述できる．

安全性 「決して悪い状態に到達しない」ことを表す性質．また,「悪い状態に到達しない可能性がある」ことも安全性のバリエーションである．UPPAAL では，f を良い状態として，それぞれ $AG\ f$，$EG\ f$ で記述できる．

活性 「必ずいつかは良い状態に到達する」ことを表す性質．また,「よい状態に到達する可能性がある」ことも活性のバリエーションである．UPPAAL では，f を良い状態として，それぞれ $AF\ f$，$EF\ f$ で記述できる．また,「状態 f に到達した場合に，必ずいつかは良い状態 g に到達する」ことは，$f \rightsquigarrow g$ で記述できる．

[3] $AG\ (f\ \text{imply}\ AF\ g)$ と同義だが，UPPPAL では時相論理式を入れ子で用いることができないため，同じ意味を表す時相演算子が用意されれている．

例えば，図 4.3 のモデルに対し,「ランプが 10（時間単位）以内で明るく点灯する可能性がある」という性質を記述すると以下のようになる.

- $EF\ (lamp.bright\ and\ lamp.y <= 10)$

第 5 章

拡張BPMNの提案

5.1 BPMNの拡張

有効なビジネスプロセスモデルを設計するためには，そのプロセスがビジネスの要件を満たしているかどうかを綿密に確認する必要がある．既に述べたとおり，時間，資源の概念はビジネスプロセスにとって重要な概念であり，それらを含むビジネス要件は，ビジネスプロセスモデルを基にしたIT システムの構築やそれを活用するビジネスの成否に関わる重要なものである．例えば，ビジネスプロセスの設計者は次のような性質を検証する．

- ビジネスプロセスは要求された時間内に終了するか.
- ビジネスプロセスは与えられた数の資源で要件を満たせるか.

上記のような性質を形式的に検証するためには，時間や資源に関する制約を明確に記述できる必要がある．時間制約は，2 章で述べたように以下の 3 つの時間制約を扱う．

(1) タスクの実行経過時間制約（Time Duration）

(2) 連続した 2 つのアクティビティの間の遅延時間制約（Time Distance）

(3) アクティビティがある時点より前に必ず終了するという制約（Deadline）

標準の BPMN では，上記のうち (2) と (3) の時間制約を記述することが可能である．それぞれ，中間イベント（タイマー），サブプロセス（タイムアウト）の要素において，時間制約を指定するための属性が定義されている．中間イベント（タイマー）では，遅延時間を指定する属性 *timeDuration* を，サブプロセス（タイムアウト）では，内包するプロセス

の上限実行時間（タイムアウト時間）を指定する属性 $timeDuration$ を持つ．しかしながら，もっとも典型的な時間制約 (1) の指定方法がないため，既に述べたようなビジネスプロセスの性質を検証するためには十分ではない．また，資源制約に関しては標準の BPMN で記述する方法は規定されていない．

そこで本研究では，標準の BPMN を拡張した “拡張 BPMN” を定義し，時間，資源に関する制約を考慮したビジネスプロセスの記述と検証が行えるようにする．BPMN の仕様では要素にカスタム属性を追加することができるため，この仕組みを用いて時間および資源に関する制約を表す属性を追加することで拡張を行う．

5.1.1 タスクの実行時間を指定する属性の追加

ビジネスプロセス全体の所要時間やタイミングに依存した性質を検証するためには，個々のタスクの実行時間を記述できるようにしておく必要がある．そのため，標準のタスク要素に属性として所要時間を指定するための最小実行時間 $MinTime$ および最大実行時間 $MaxTime$ を指定できるよう拡張する．これらの属性が指定されたタスクは，実行が開始されてから $MinTime$ 以上 $MaxTime$ 以下の時間で終了する．図 5.1 は，属性の指定例である．

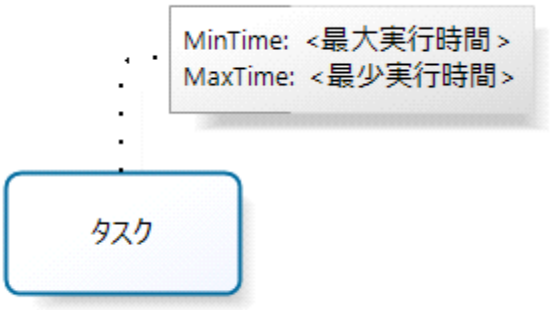

図 5.1: 実行時間制約を持つタスク要素

5.1.2　アクティビティの実行時に必要な資源を指定する属性の追加

ビジネスプロセスの資源には，アクティビティを実行する人的資源（人間）とアクティビティの実行中に一時的に利用する物的資源（モノやサービス）がある．これらはいずれも，アクティビティの開始から終了まで占有され，アクティビティの実行が終了すると別のアクティビティで利用可能となる排他資源とみなすことができる．資源は有限であり，最大数を超えた資源利用では競合が発生し，ビジネスプロセスの実行に影響を及ぼす．

本研究では，アクティビティの各要素に属性として利用資源 *Resource* およびその利用数 *Quantity* を指定できるように拡張する．これらの属性を指定されたアクティビティは，実行の開始から終了まで資源 *Resource* を利用数 *Quantity* だけ利用（占有）する．開始時点で利用数分の資源が確保できない場合，別のアクティビティが資源の利用を終了するまで待機する．図 5.2 は，属性の指定例である．

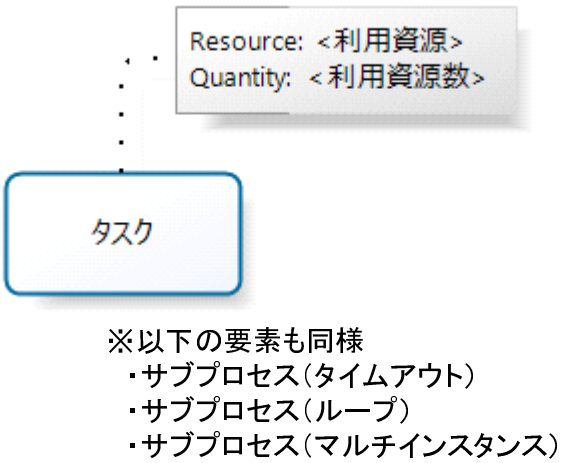

図 5.2: 資源制約を持つアクティビティ要素

一方，各資源の総数はビジネスプロセス全体の属性 *Resources* として指定できるようにする（図 5.3）．

5.2　拡張 BPMN の例

図 5.4 は，拡張 BPMN によるビジネスプロセスの記述例である．BPMN で記述した看護・介護のビジネスプロセス例（図 3.2）に対し，前節で導入した拡張属性を用いて時間，

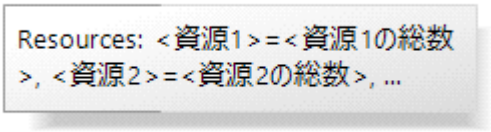

図 5.3: ビジネスプロセスで利用される資源の総数の指定

資源に関する制約条件を付加したものである．

例えば，グループ 1 のタスク「食事（要介護）」は，実行時間制約を表す属性 Min_Time，Max_Time によって，食事に要する時間が最短で 30 分，最長で 60 分であることが表現されている．さらに，資源制約を表す属性 $Resource$，$Quantity$ によって，看護師 1 名が食事を介助する必要があることが表現されている．なお，看護師の総数はビジネスプロセス全体の属性 $Resources$ に指定された 3 名である．

5.3 拡張 BPMN の抽象構文

拡張 BPMN では，サブプロセス要素を用いて，プロセスの階層構造を持つことができる．本研究では，各階層の個々のプロセスを拡張 BPMN プロセス，拡張 BPMN プロセスから構成される全体のビジネスプロセスモデルを拡張 BPMN モデルと呼ぶ．例えば，看護・介護のビジネスプロセスでは，3 つの拡張 BPMN プロセスによって，拡張 BPMN モデルが構成されている（図 5.5）．

拡張 BPMN から拡張時間オートマトンの変換を形式的に定義するための準備として，ここで拡張 BPMN の抽象構文を定義しておく．

5.3.1 拡張 BPMN プロセスの抽象構文

定義 2 (拡張 BPMN プロセス) 拡張 BPMN プロセスをタプル $\mathcal{P} = (\mathcal{O}, \mathcal{A}, \mathcal{E}, \mathcal{G}, \mathcal{T}, \mathcal{S}, \mathcal{S}^{\mathcal{T}}, \mathcal{S}^{\mathcal{R}}, \mathcal{S}^{\mathcal{MIP}}, \mathcal{E}^{\mathcal{S}}, \mathcal{E}^{\mathcal{IT}}, \mathcal{E}^{\mathcal{E}}, \mathcal{G}^{\mathcal{F}}, \mathcal{G}^{\mathcal{J}}, \mathcal{G}^{\mathcal{X}}, \mathcal{G}^{\mathcal{M}}, \mathcal{R}, \mathcal{F}, \mathcal{F}^{\mathcal{E}}, \mathsf{Duration}, \mathsf{Min}, \mathsf{Max}, \mathsf{Loop}, \mathsf{Res})$ と定義する．ここで，

- $\mathcal{O}$ はフローオブジェクトの集合であり，互いに素なアクティビティ集合 $\mathcal{A}$，イベント集合 $\mathcal{E}$ およびゲートウェイ集合 $\mathcal{G}$ に分割することができる．

- $\mathcal{A}$ は互いに素なタスク集合 $\mathcal{T}$ とサブプロセス集合 $\mathcal{S}$ に分割することができる．
- $\mathcal{S}$ は互いに素なサブプロセス（タイムアウト）集合 $\mathcal{S}^{\mathcal{T}}$，サブプロセス（ループ）集合 $\mathcal{S}^{\mathcal{R}}$，サブプロセス（マルチインスタンス）集合 $\mathcal{S}^{\mathcal{MIP}}$ に分割することができる．
- $\mathcal{E}$ はイベントの集合であり，互いに素な開始イベント集合 $\mathcal{E}^{\mathcal{S}}$，中間イベント（タイマー）集合 $\mathcal{E}^{\mathcal{IT}}$ および終了イベント集合 $\mathcal{E}^{\mathcal{E}}$ に分割することができる．
- $\mathcal{G}$ はゲートウェイの集合であり，互いに素なフォーク分岐ゲートウェイ集合 $\mathcal{G}^{\mathcal{F}}$，結合ゲートウェイ集合 $\mathcal{G}^{\mathcal{J}}$，排他ゲートウェイ集合 $\mathcal{G}^{\mathcal{X}}$ および併合ゲートウェイ集合 $\mathcal{G}^{\mathcal{M}}$ に分割することができる．
- $\mathcal{R}$ は資源の集合である．
- $\mathcal{F} \subseteq \mathcal{O} \times \mathcal{O}$ はシーケンスフローの集合であり，例外シーケンスフローの集合 $\mathcal{F}^{\mathcal{E}} \subseteq \mathcal{S}^{\mathcal{T}} \times \mathcal{O}$ は部分集合である．
- $\mathsf{Duration}: \mathcal{E}^{\mathcal{IT}} \cup \mathcal{S}^{\mathcal{T}} \to \mathbb{Z}_0^+$ は，中間イベント（タイマー）とサブプロセス（タイムアウト）に，それぞれ遅延時間とタイムアウト時間を割り当てる関数である．
- $\mathsf{Min}, \mathsf{Max}: \mathcal{T} \to \mathbb{Z}_0^+$ は，タスクに最小，最大処理時間を割り当てる関数である．ただし，$\forall t \in \mathcal{T}, \mathsf{Min}(t) \leq \mathsf{Max}(t)$ である．
- $\mathsf{Loop}: \mathcal{S}^{\mathcal{R}} \cup \mathcal{S}^{\mathcal{MIP}} \to \mathbb{N}$ は，サブプロセス（ループ）とサブプロセス（マルチインスタンス）に実行回数を割り当てる関数である．
- $\mathsf{Res}: \mathcal{A} \to \mathcal{R} \times \mathbb{N}$ は，アクティビティに対して利用資源とその利用数を割り当てる関数である．

ただし，$\mathbb{N}$，$\mathbb{Z}_0^+$ は，それぞれ自然数の集合，非負正数の集合である．

BPMN はビジネスドメインでの利用を意識した記法のため，様々な省略記法や代替記法が許されている．例えば，プロセスの開始点や終了点が明確な場合，開始イベントや終了イベントを省略することができたり，ゲートウェイを省略してアクティビティ要素に 2 つ以上の入力シーケンスフローを接続したりすることができる．本研究で扱う拡張 BPMN では，拡張時間オートマトンへの対応付けを単純にするために上記のような構文上の冗長性

を取り除いたプロセスを対象とする．ここで導入する制約は拡張 BPMN の記述能力を限定するものではなく，排除される記法は対象となる別の記法で代替可能である．

まず準備として，3 つの関数 $\mathsf{in}, \mathsf{out}, \mathsf{eout}$ を定義する．任意のフローオブジェクト $x \in \mathcal{O}$ に対し，$\mathsf{in}(x) = \{y \in \mathcal{O} | (y, x) \in \mathcal{F}\}$，$\mathsf{out}(x) = \{y \in \mathcal{O} | (x, y) \in \mathcal{F} \backslash \mathcal{F}^{\mathcal{E}}\}$ は，それぞれシーケンスフローで接続される直前のフローオブジェクト，直後のフローオブジェクトを返す関数である．また，任意のサブプロセス (タイムアウト)$x \in \mathcal{S}^{\mathcal{T}}$ に対し，$\mathsf{eout}(x) = \{y \in \mathcal{O} | (x, y) \in \mathcal{F}^{\mathcal{E}}\}$ は，例外シーケンスフローで接続される直後のフローオブジェクトを返す関数である．

本研究では，以下の条件を満たす拡張 BPMN プロセス $\mathcal{P}$ を対象とする．

- 拡張 BPMN プロセスは開始イベント，終了イベントをそれぞれ 1 ずつ含む．
  $|\mathcal{E}^{\mathcal{S}}| = 1 \wedge |\mathcal{E}^{\mathcal{E}}| = 1$

- 開始イベントの入力フロー数は 0，出力フロー数は 1，例外フロー数は 0 である．
  $\forall x \in \mathcal{E}^{\mathcal{S}}, |\mathsf{in}(x)| = 0 \wedge |\mathsf{out}(x)| = 1 \wedge |\mathsf{eout}(x)| = 0$

- 終了イベントの入力フロー数は 1，出力フロー数は 0，例外フロー数は 0 である．
  $\forall x \in \mathcal{E}^{\mathcal{E}}, |\mathsf{in}(x)| = 1 \wedge |\mathsf{out}(x)| = 0 \wedge |\mathsf{eout}(x)| = 0$

- 中間イベント (タイマー)，サブプロセス (タイムアウト) 以外のアクティビティの入力フロー数は 1，出力フロー数は 1，例外フロー数は 0 である．
  $\forall x \in \mathcal{E}^{\mathcal{IT}} \cup \mathcal{A} \backslash \mathcal{S}^{\mathcal{T}}, |\mathsf{in}(x)| = 1 \wedge |\mathsf{out}(x)| = 1 \wedge |\mathsf{eout}(x)| = 0$

- サブプロセス (タイムアウト) の入力フロー数は 1，出力フロー数は 1，例外フロー数は 1 である．
  $\forall x \in \mathcal{S}^{\mathcal{T}}, |\mathsf{in}(x)| = 1 \wedge |\mathsf{out}(x))| = 1 \wedge |\mathsf{eout}(x)| = 1$

- フォーク分岐ゲートウェイ，排他ゲートウェイの入力フロー数は 1，出力フロー数は 1 より大きく，例外フロー数は 0 である．
  $\forall x \in \mathcal{G}^{\mathcal{F}} \cup \mathcal{G}^{\mathcal{X}}, |\mathsf{in}(x)| = 1 \wedge |\mathsf{out}(x)| > 1 \wedge |\mathsf{eout}(x)| = 0$

- 結合ゲートウェイ，併合ゲートウェイの入力フロー数は 1 より大きく，出力フロー数は 1，例外フロー数は 0 である．
  $\forall x \in \mathcal{G}^{\mathcal{J}} \cup \mathcal{G}^{\mathcal{M}}, |\mathsf{in}(x)| > 1 \wedge |\mathsf{out}(x)| = 1 \wedge |\mathsf{eout}(x)| = 0$

5.3.2 拡張 BPMN モデルの抽象構文

次に，拡張 BPMN モデルの抽象構文を以下の通り定義する．

定義 3 (拡張 BPMN モデル) 拡張 BPMN モデルを以下のタプル $\mathcal{M} = (\mathcal{Q},\ top,\ \mathcal{S}^{\diamond}, \mathcal{R}^{\diamond},\ \mathsf{map}, \mathsf{HR}, \mathsf{RIni})$ と表記する．ここで，

- $\mathcal{Q}$ は拡張 BPMN プロセスの集合である．
- $top \in \mathcal{Q}$ はトップレベルのプロセスである．
- $\mathcal{S}^{\diamond} = \cup_{\mathcal{P} \in \mathcal{Q}} \mathcal{S}_{\mathcal{P}}$ は全てのサブプロセス要素の集合である．
- $\mathcal{R}^{\diamond} = \cup_{\mathcal{P} \in \mathcal{Q}} \mathcal{R}_{\mathcal{P}}$ は全ての資源の集合である．
- $\mathsf{map} : \mathcal{S}^{\diamond} \to \mathcal{Q} \backslash \{\ top\ \}$ は，サブプロセス要素から拡張 BPMN プロセスへの全単射関数である．
- $\mathsf{HR} = \{(\mathcal{P}_1, \mathcal{P}_2) \in \mathcal{Q} \times \mathcal{Q} | \exists_{\mathsf{s} \in \mathcal{S}_{\mathcal{P}_1}} \mathsf{map}(\mathsf{s}) = \mathcal{P}_2\}$ は親子プロセス関係の集合である．
- $\mathsf{RIni} : \mathcal{R}^{\diamond} \to \mathbb{N}$ は，各資源に対し利用可能数の初期値を割り当てる関数である．

ただし，$\mathcal{S}_{\mathcal{P}}, \mathcal{R}_{\mathcal{P}}$ はそれぞれ拡張 BPMN プロセス $\mathcal{P}$ で定義されたサブプロセス要素の集合，資源の集合である．

拡張 BPMN で導入した資源は集合 $\mathcal{R}^{\diamond}$ として定義している．RIni は，拡張 BPMN モデルが開始される時点で，各資源の初期数を割り当てる関数である．

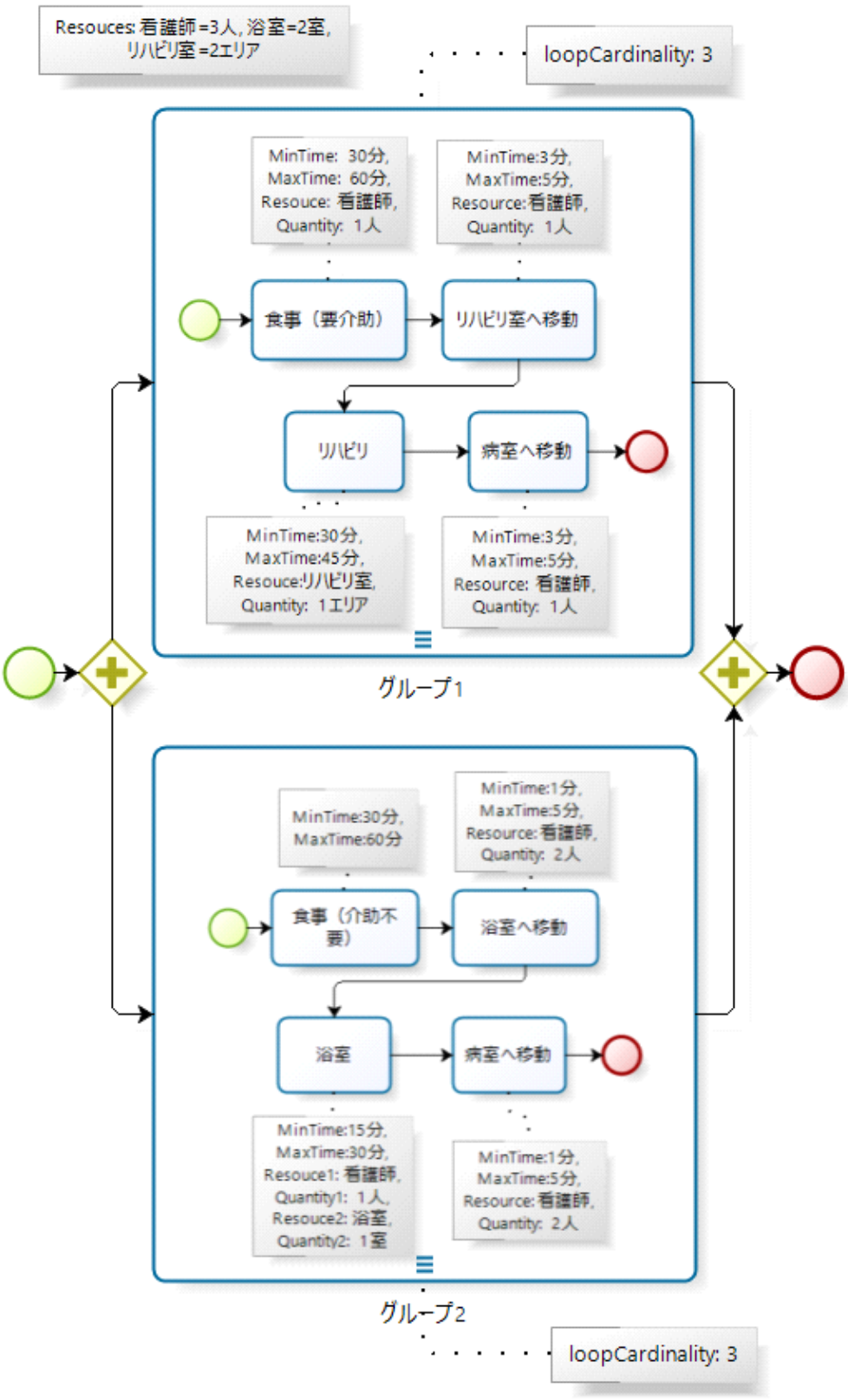

図 5.4: 拡張 BPMN の記述例

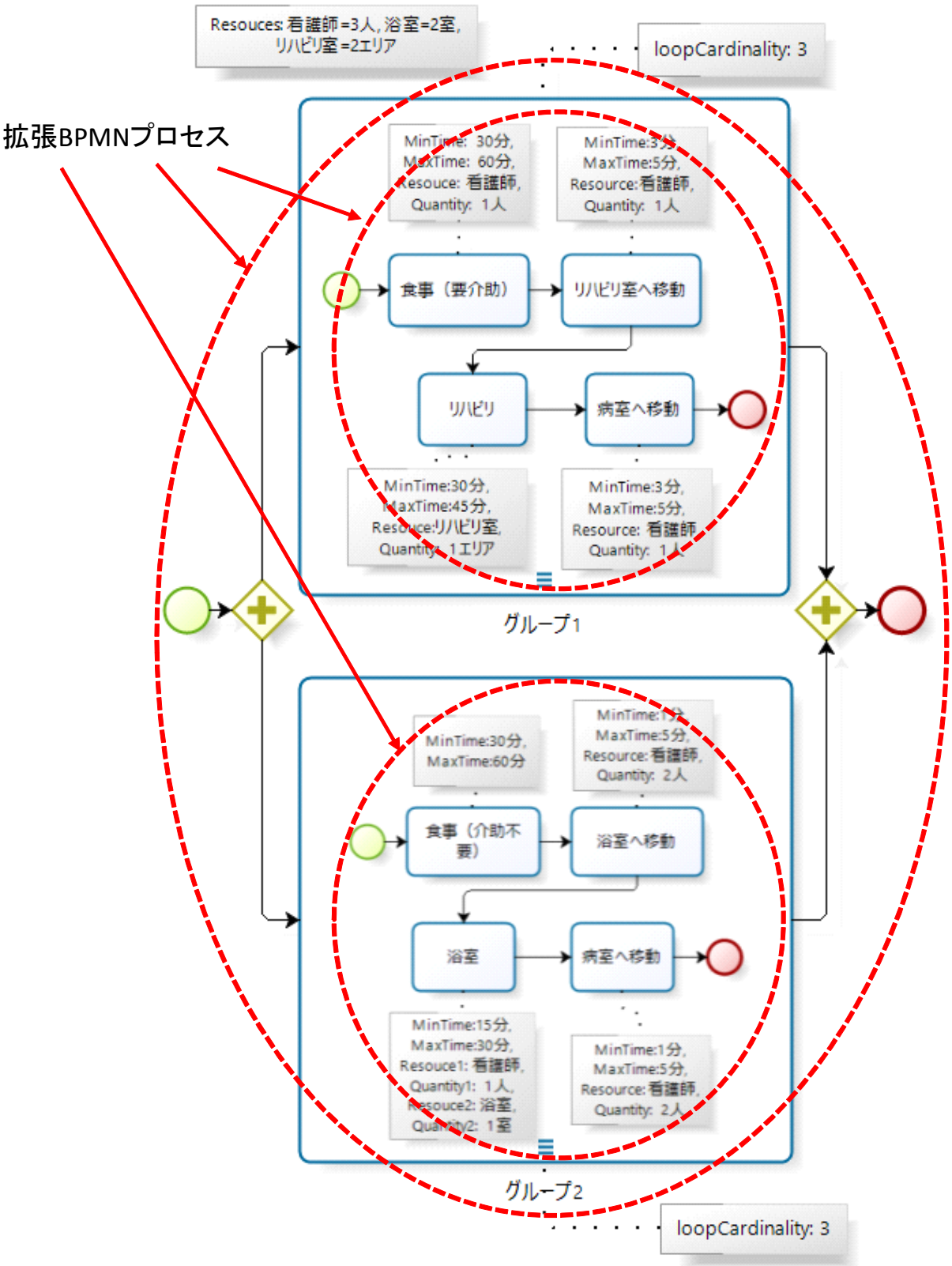

図 5.5: 拡張 BPMN プロセス

第 6 章

拡張BPMNから拡張時間オートマトンへの変換手法の提案

6.1 変換方針

拡張 BPMN におけるフローオブジェクトはビジネスプロセスの振る舞いを特徴づける主要な構成要素である．よって，本手法ではビジネスプロセスに出現するフローオブジェクトをそれぞれ 1 つの拡張時間オートマトンに対応付けてモデル化する．フローオブジェクトの持つ振る舞いや時間や資源に関する制約は，状態遷移の構造，チャネル同期を伴うアクション，クロック変数や整数変数を含む不変式，ガード条件および代入式によって表現する．一方，プロセスの構造，すなわちシーケンスフローやサブプロセスの呼出／復帰／タイムアウト割り込みによって決定される実行順序の制御は，順序関係を持つフローオブジェクト同士が同一のチャネルで送受信（同期）を行うことによって表現する．例えば，フローオブジェクト X とフローオブジェクト Y がシーケンスフローで接続されている場合，X に対応する拡張時間オートマトンの最後の遷移と，Y に対応する拡張時間オートマトンの最初の遷移を，同一のチャネルで同期させることでモデル化する．上記の方針により，ビジネスプロセスモデル全体は，出現するフローオブジェクトに対応する拡張時間オートマトンの集合をチャネル同期で結合したネットワーク，すなわち UPPAAL が検証可能なモデルの形式となる．

拡張 BPMN から拡張時間オートマトンへの変換は，ビジネスプロセスの構造や制約値の調整の度に行う必要がある．特に，プロセス構造や同時実行数などを調整する場合には，拡張時間オートマトン上の変数値の変更では対応できない．そのため，あらかじめフローオ

ブジェクトの種類ごとに拡張時間オートマトンのテンプレート（オートマトンテンプレートと呼ぶ）を用意しておき，プロセスに含まれるオブジェクトや構造に応じて，系統的な手順で変換できるようにする．フローオブジェクト同士の接続関係や具体的な制約値など，ビジネスプロセスの構成内容によって異なるチャネルや定数はパラメータとして定義しておき，変換時に具体的な値を設定できるようにする．

6.2 変換手順

フローオブジェクトごとに用意されたオートマトンテンプレートを用いて，拡張 BPMN によるビジネスプロセスモデルから拡張時間オートマトンモデルへ変換する手順は以下の通りである（図 6.1）．オートマトンテンプレートの詳細については次節で詳しく説明を行う．

(1) ビジネスプロセスモデル中のすべてのフローオブジェクトを，それぞれの種類に応じたオートマトンテンプレートに置換する．

(2) フローオブジェクト間の接続関係や制約値に基づいて，オートマトンテンプレートのパラメータに具体的なチャネルや定数値を割り当てて，完全な拡張時間オートマトン（のネットワーク）モデルにする．

(3) 最後にトップレベルプロセスの開始イベントと終了イベントを呼出／復帰させるための特別な拡張時間オートマトン（起動用オートマトンと呼ぶ）を追加する．

手順 (1)

まず，ビジネスプロセスに出現するフローオブジェクトを抽出し，それぞれ対応するオートマトンテンプレートに置換する．サブプロセス（マルチインスタンス）要素を含む場合，その内包プロセスに含まれるフローオブジェクトは，サブプロセス（マルチインスタンス）の同時実行数（属性 loopCardinality で指定される数）だけ，独立に並列して実行状態をもつため，対応するオートマトンテンプレートは同時実行数分だけ準備する必要がある．さらに，サブプロセス（マルチインスタンス）要素 S_A の内包プロセスの中に，さらに入れ子でサブプロセス（マルチインスタンス）要素 S_B が含まれる場合，S_B の内包プロセスに

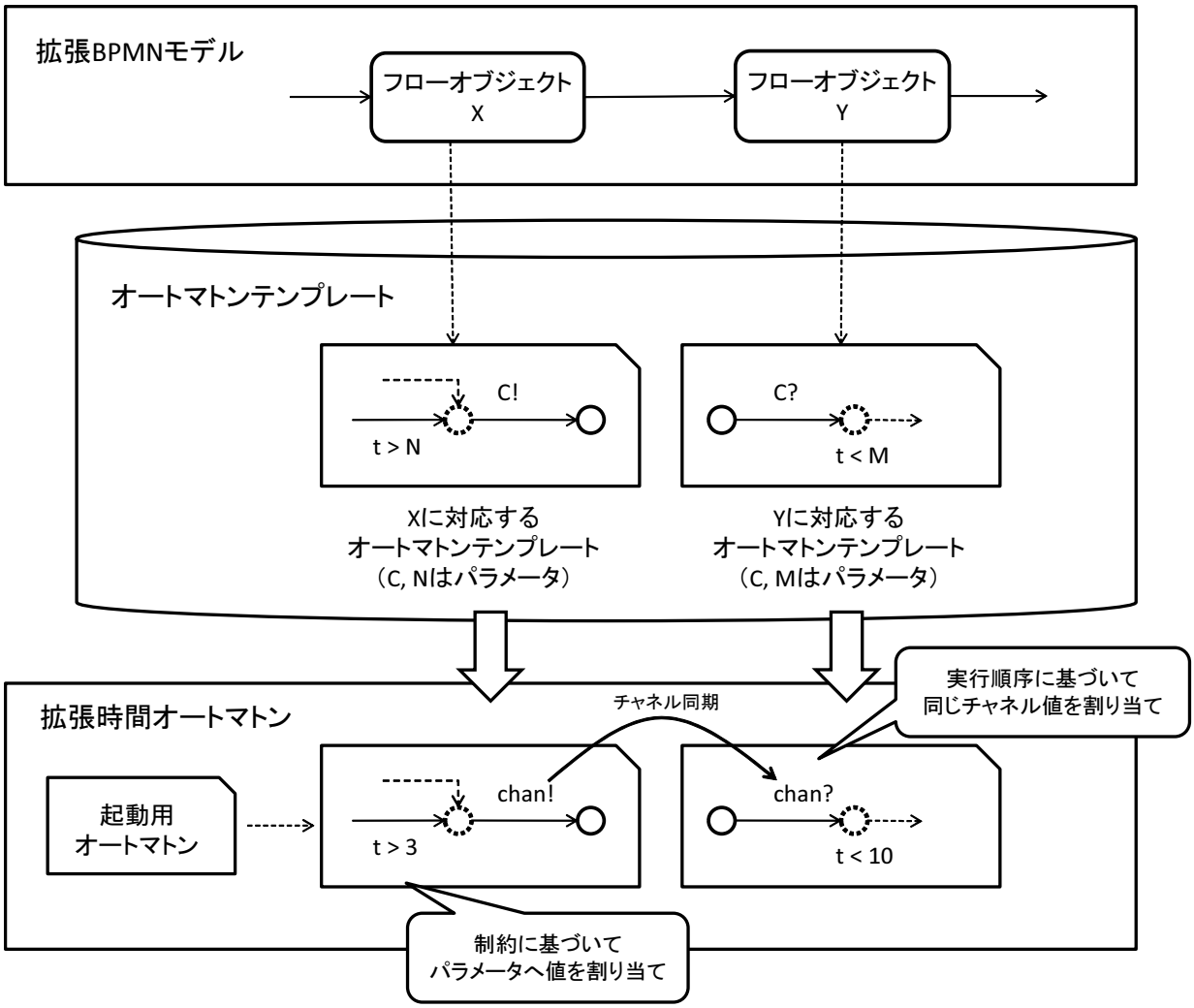

図 6.1: 拡張 BPMN から拡張時間オートマトンへの変換

含まれるフローオブジェクトに対応するオートマトンテンプレートは，S_A の同時実行数× S_B の同時実行数だけ準備する（図 6.2）.

手順 (2)

次に，ビジネスプロセスモデルの構成に基づき，手順 (1) で準備したオートマトンテンプレートのパラメータに，具体的なチャネルや値を割り当てる．オートマトンテンプレートにおけるパラメータは，チャネルと，不変式，ガード条件および代入式に出現する定数であり，次の方法で具体的な値を割りあてる．

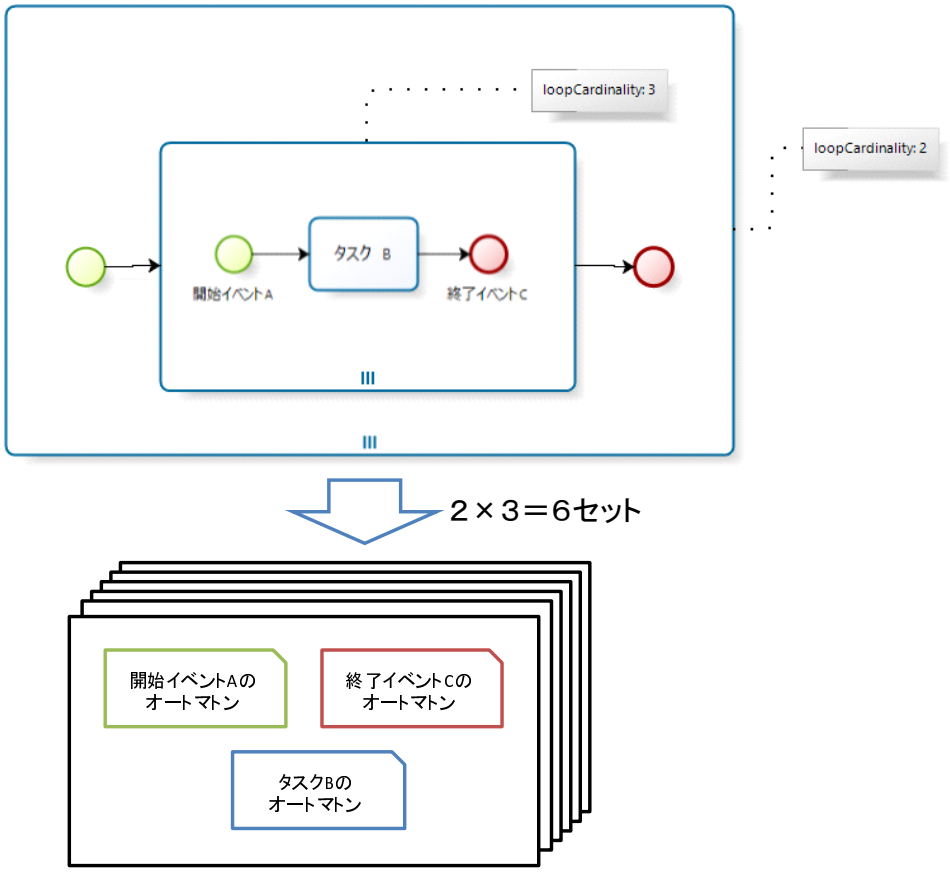

図 6.2: サブプロセス（マルチインスタンス）の内包プロセスに含まれるフローオブジェクトの変換方法

1. チャネル

   変換方針で示したように，フローオブジェクト間の実行順序を制御するものであり，シーケンスフローによる接続関係，サブプロセス要素による内包プロセスの呼出／復帰関係，タイムアウトの割込関係に基づいて，チャネル名を決定する．例えば，フローオブジェクト X とフローオブジェクト Y がシーケンスフローで接続されている場合，X に対応する拡張時間オートマトンの最後の遷移に指定されているチャネルパラメータと，Y に対応する拡張時間オートマトンの最初の遷移に指定されているチャネルパラメータには，同一のチャネルを割り当てる．

2. 定数

   フローオブジェクトの振る舞いを決定する属性値から定数値を決定する．例えば，タ

スク要素に対応するオートマトンテンプレートの最小実行時間には，タスク要素の属性 $MinTime$ の値を割り当てる．各オートマトンテンプレートのパラメータと，拡張 BPMN 要素の属性の対応関係は，次節で説明する．

手順 (3)

手順 (1) と (2) を実施すると，ビジネスプロセスの一番最初の開始イベントと一番最後の終了イベント以外のフローオブジェクトは，具体的なチャネルや値が割り当てられて完全な拡張時間オートマトンとなる．ただし，ビジネスプロセスの一番最初の開始イベントに対応するオートマトンテンプレートは，最初の遷移を行うためのチャネルパラメータが決定されていない．さらに，一番最後の終了イベントに対応するオートマトンテンプレートは，最後の遷移を行うためのチャネルパラメータが決定されていない．よって，最後に上記のチャネルと同期するための起動用オートマトンを追加して，開始イベント，終了イベントのチャネルと同期できるようにして，完全な拡張時間オートマトンモデルを得る（図 6.3）．

さらに，起動用オートマトンではビジネスプロセス全体の資源の初期化を行う．ビジネスプロセス全体の属性 $Resources$ の指定に従って，ロケーション $Start$ から $Ready$ への遷移に設定された資源数を表す変数の初期化を行う代入式に具体的な値（資源の総数）を割り当てる．

6.3 オートマトンテンプレート

BPMN は自然言語によって仕様が記述されており，明確なセマンティクスは与えられていない．よって，BPMN から別の形式モデルへの変換は一意ではなく，本研究で示す BPMN モデルから拡張時間オートマトンモデルへの変換の正当性を証明することはできない．従って，9 章に示す関連研究にも見られるように，本研究においても，BPMN における自然言語による性質の記述を注意深く解釈し，形式モデルである拡張時間オートマトンとの対応関係を示す．

図 6.4，図 6.5，図 6.6 および付録 A 章に示す拡張時間オートマトンは，拡張 BPMN に含まれるフローオブジェクト要素に対応するオートマトンテンプレートの一覧である．以下では，時間や資源に関する性質を持つフローオブジェクトのうち，タスク，サブプロセ

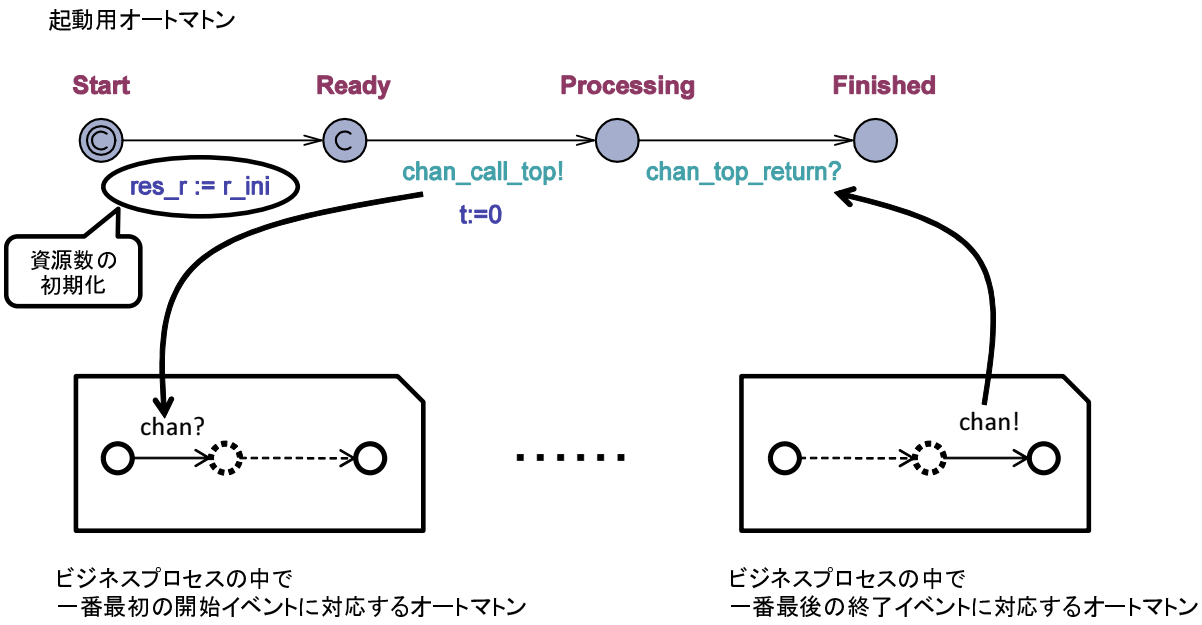

図 6.3: 起動用オートマトン

ス（タイムアウト），サブプロセス（マルチインスタンス）に対応するオートマトンテンプレートについて解説する．

6.3.1 タスク要素のオートマトンテンプレート

拡張 BPMN におけるタスク要素は，指定された実行時間の制約のもと，指定された資源を利用して作業を実行するアクティビティであり，以下の性質を持つ．

(1) 入力シーケンスフローを介して直前のフローオブジェクトから制御を受け取ると作業を開始し，作業が完了すると出力シーケンスフローを介して次のフローオブジェクトに制御を渡す．その後は，再び直前のフローオブジェクトからの制御を待つ．

(2) 作業は開始されてから属性 $MinTime$ で指定された時間以上，$MaxTime$ で指定された時間以下で完了する．

(3) 作業の開始から終了まで属性 $Resource$ で指定された資源を属性 $Quantity$ で指定された数量だけ利用（占有）する．資源を確保できない場合は，資源が利用可能になるまで作業開始を待機する．

(4) 自身を包含しているサブプロセス（タイムアウト）でタイムアウトが発生した場合は，作業の実行を即時停止し，その後は直前のフローオブジェクトからの制御を待つ．

上記の各性質を，図 6.4 に示すオートマトンテンプレートにおいて次のようにモデル化している．

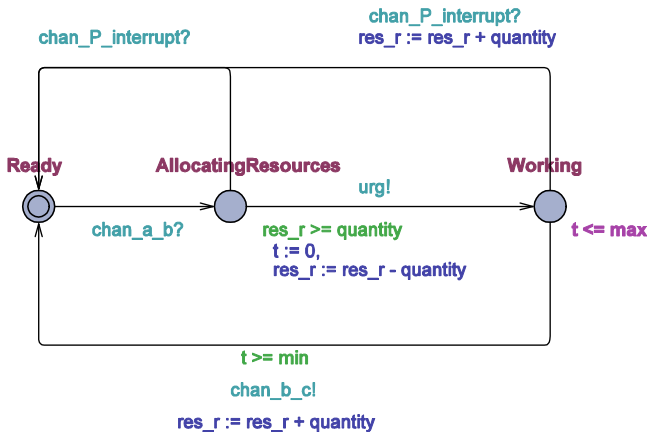

図 6.4: タスク要素のオートマトンテンプレート

タスクの状態を表すロケーションとして，直前のフローオブジェクトの完了を待っている状態（$Ready$），資源確保前の状態（$AllocatingResources$）および作業中の状態（$Working$）を用意する．

性質 (1) のモデル化のため，$Ready$ から $AllocatingResources$ の遷移と $Working$ から $Ready$ の遷移には，タスク要素の前後のフローオブジェクトと同期するためのアクション（$chan\_a\_b?$, $chan\_b\_c!$）を指定している．各チャネル（$chan\_a\_b$, $chan\_b\_c$）は，ビジネスプロセスの構成（フローオブジェクト間の接続関係）で決定されるパラメータであり，テンプレート利用時にはそれぞれ前後に接続されているフローオブジェクトと共有したチャネルが割り当てられる．

また，性質 (2) については，作業の実行時間を計測するクロック変数 t を用意し，実行時間が指定された最大時間以下であることは，$Working$ の不変式（$t \leq max$）によってモデル化している．同様に，実行時間が指定された最小時間以上であることは，$Working$ から $Ready$ の遷移上のガード条件（$t \geq min$）によってモデル化している．min，max は，テンプレート利用時にタスクの属性 $MinTime$，$MaxTime$ の値が割り当てられる定数パラメータである．なお，$AllocatingResources$ から $Working$ の遷移上の代入式（$t := 0$）は，作業開始時にクロック変数をリセットするためのものである．

さらに，性質 (3) については，作業の開始時すなわち $AllocatingResources$ から $Working$ の遷移上のガード条件（$res\_r \geq quantity$）および代入式（$res\_r := res\_r - quantity$）と，作業の完了時すなわち $Working$ から $Ready$ の遷移上の代入式（$res\_r := res\_r + quantity$）によってモデル化している．$AllocatingResources$ から $Working$ の遷移上のアクション（$urg!$）は，受信側の存在しないブロードキャストのアージェントチャネル送信であり，資源の確保後にこの遷移が即座に（時間の経過なく）行われるようにするための指定である．res_r および $quantity$ は，テンプレート利用時にタスクの属性 $Resource$ および $Quantity$ によって値が割り当てられる定数パラメータである．

最後に，性質 (4) のモデル化のため，$Ready$ 以外の状態から $Ready$ への遷移を用意している．遷移上に指定されているチャネル同期アクション（$chan\_P\_interrupt?$）は，タイムアウトの発生時にサブプロセス（タイムアウト）要素から送信されるブロードキャストチャネル通信を受信するためのアクションである．$chan\_P\_interrupt$ は，テンプレート利用時にサブプロセス（タイムアウト）との包含関係により決定されるパラメータであり，自身を包含するサブプロセス（タイムアウト）がタイムアウト時に送信するチャネルと同一のチャネルが割り当てられる．

6.3.2 サブプロセス（タイムアウト）要素のオートマトンテンプレート

サブプロセス（タイムアウト）は，実行時間にタイムアウトを設定して内包するプロセスを実行するためのアクティビティであり，以下の性質を持つ．

(1) 入力シーケンスフローを介して直前のフローオブジェクトから制御を受け取ると，内包プロセスを呼び出し，内包プロセスから復帰すると出力シーケンスフローを介して次のフローオブジェクトへ制御を渡す．その後は，再び直前のフローオブジェクトからの制御を待つ．

(2) 内包プロセスの呼出時には属性 $Resource$ で指定された資源を属性 $Quantity$ で指定された数だけ確保し，復帰時には資源を返却する．呼出時に資源を確保できなかった場合は，内包プロセスは呼び出されずに確保できるようになるまでブロックされる．

(3) 内包プロセスの呼出では内包プロセスの開始イベントへ制御が渡され，復帰では内包プロセスの終了イベントから制御を受け取る．

(4) 内包プロセスの実行が，属性 $timeDuration$ で指定されたタイムアウト時間を超過した場合，内包プロセスの実行を即時停止する．

(5) 自身を包含している別のサブプロセス（タイムアウト）でタイムアウトが発生した場合は，自身の実行を即時停止させる．また，本要素はサブプロセスの一種であるため，自信の内包プロセスの実行も即時停止する必要がある．

上記の各性質を，図 6.5 に示すオートマトンテンプレートにおいて次のようにモデル化している．

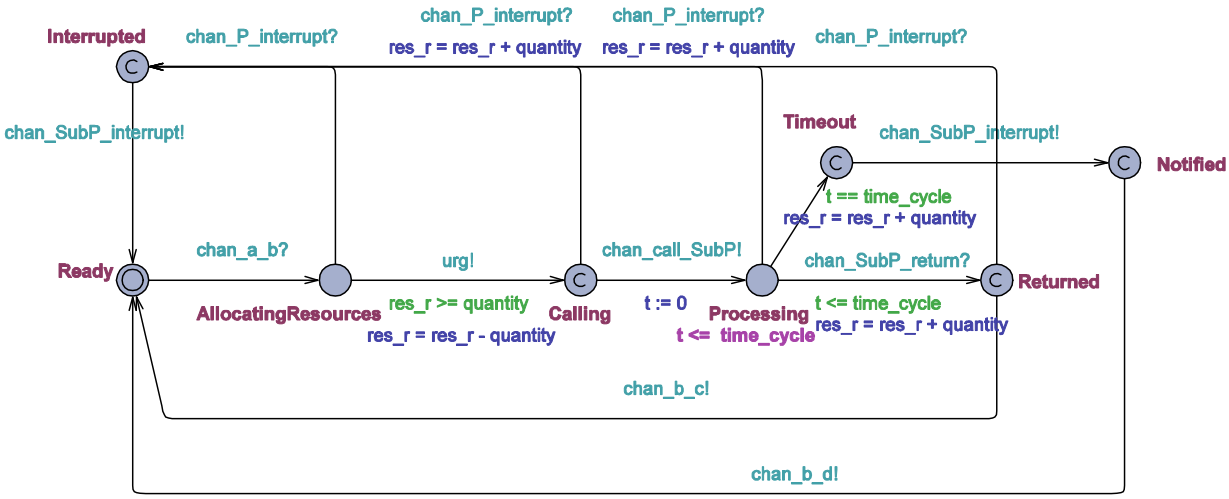

図 6.5: サブプロセス（タイムアウト）要素のオートマトンテンプレート

サブプロセス（タイムアウト）要素の基本的な状態を表すロケーションとして，実行前の状態（$Ready$），資源確保前の状態（$AllocatingResources$），内包プロセスを呼出している状態（$Calling$），内包プロセスが実行中の状態（$Processing$）および内包プロセスから復帰した状態（$Retruned$）を用意する．また，タイムアウトが発生した際の状態を表すため，発生直後の状態（$Timeout$）と自身の内包プロセスにタイムアウトを通知した状態（$Notified$）を用意する．さらに，自身を内包するサブプロセスからタイムアウト通知を受け取った状態（$interrupted$）を用意する．

性質 (1) については，$Ready$ から $AllocatingResources$ の遷移と $Returned$ から $Ready$ の遷移に，前後のフローオブジェクトと同期するためのアクション（$chan\_a\_b?$, $chan\_b\_c!$）を指定しており，タスク要素の場合と同様である．

性質 (2) については，内包プロセス呼出の直前すなわち $AllocatingResources$ から $Calling$ の遷移上のガード条件（$res\_r \geq quantity$）および代入式（$res\_r := res\_r - quantity$）と，内部プロセスの復帰時すなわち $Processing$ から $Returned$ の遷移上の代入式（$res\_r := res\_r + quantity$）によってモデル化している．$AllocatingResources$ から $Calling$ の遷移上のアクション（$urg!$）は，受信側の存在しないブロードキャストのアージェントチャネル送信であり，資源の確保後にこの遷移が即座に（時間の経過なく）行われるようにするための指定である．res_r および $quantity$ は，テンプレート利用時にタスクの属性 $Resource$ および $Quantity$ によって値が割り当てられる定数パラメータである．

性質 (3) については，内包プロセスの呼出を行うため，$Calling$ から $Processing$ の遷移には，内包プロセスに含まれる開始イベントの最初の遷移と同期するためのチャネル同期アクション（$chan\_call\_SubP!$）を指定している．一方，内包プロセスの復帰を行うため，$Processing$ から $Returned$ の遷移には，内包プロセスに含まれる終了イベントの最後の遷移と同期するためのチャネル同期アクション（$chan\_SubP\_return?$）を指定している．チャネル $chan\_call\_SubP$ および $chan\_SubP\_return$ はパラメータであり，利用時に前者は内包プロセスの開始イベントと，後者は終了イベントと同一のチャネルを割り当てる．

性質 (4) については，内包プロセスの実行時間を計測するため，クロック変数 t を用意し，内包プロセスの実行に上限があることを，$Processing$ の不変式（$t \leq time\_cycle$），およびタイムアウトが発生した状態（$Timeout$）への遷移のガード条件（$t == time\_cycle$）によってモデル化している．$time\_cycle$ は，利用時にサブプロセス（タイムアウト）の属性 $timeDuration$ の値が割り当てられる定数パラメータである．タイムアウト発生後は，内包プロセスの実行を停止するため，$Timeout$ から $Notified$（タイムアウトを通知した状態）への遷移に，内包プロセスに含まれるフローオブジェクトへブロードキャストチャネル送信を行うアクション（$chan\_SubP\_interrupt!$）を指定している．チャネル $chan\_SubP\_interrupt$ はパラメータであり，利用時に内包プロセスに含まれるフローオブジェクトがタイムアウト通知を受信するチャネルと同一のチャネルが割り当てられる．さらに，$Notified$ から $Ready$ の遷移においては，例外シーケンスフローに接続されているフローオブジェクトに制御を渡すためのチャネル同期アクション $chan\_b\_d!$ を指定している．チャネル $chan\_b\_d$ はパラメータであり，利用時に例外シーケンスフローに接続されているフローオブジェクトと同一のチャネルが割り当てられる．

性質 (5) については，自身を内包する上位のサブプロセス（タイムアウト）においてタイムアウトが発生した場合に，自身の実行を停止するため，$AllocatingResources$，$Calling$，

Processing および *Retruned* に *Interrupted* への遷移を用意し，タスクのオートマトンテンプレートと同様に，遷移上にはタイムアウトの通知を受けるためのブロードキャストチャネルの同期アクション（*chan_P_interrupt*?）を指定している．さらに，自身の内包プロセスの実行を停止するため，*Interrupted* から *Ready* の遷移に設定されたブロードキャストチャネルの同期アクション（*chan_SubP_interrput*!）により，内包プロセスに含まれるフローオブジェクトにタイムアウトの通知を伝播させている．*chan_P_interrupt* および *chan_SubP_interrput* は，プロセスの包含関係により決定されるパラメータである．

なお，サブプロセス（タイムアウト）において，時間の経過が発生する状態は実行前と内包プロセスの実行中のみであるため，*Ready* と *Processing* ロケーション以外は時間の経過が許されないコミットロケーションに指定している．

6.3.3 サブプロセス（マルチインスタンス）要素のオートマトンテンプレート

サブプロセス（マルチインスタンス）は，内包するプロセスを指定数だけ同時に実行するアクティビティであり，以下の性質を持つ．

(1) 入力シーケンスフローを介して直前のフローオブジェクトから制御を受け取ると実行を開始し，実行が完了すると出力シーケンスフローを介して次のフローオブジェクトへ制御を渡す．その後は，再び直前のフローオブジェクトからの制御を待つ．

(2) 内包プロセスの開始から終了まで属性 *Resource* で指定された資源を属性 *Quantity* で指定された数量だけ利用（占有）する．資源を確保できない場合は，資源が利用可能になるまで作業開始を待機する．

(3) 内包プロセスの呼出では内包プロセスに含まれる開始イベントへ制御が渡され，復帰では内包プロセスに含まれる終了イベントから制御を受け取る．属性 *loopCardinality* で指定された数だけ並列に内包プロセスの呼出を行い，すべての内包プロセスの実行が終了するまで待つ．

(4) 自身を包含しているサブプロセス（タイムアウト）でタイムアウトが発生した場合は，自身の内包プロセスの各要素にタイムアウトを通知（伝播）し，自身の実行を即時停止させる．

上記の各性質を，図 6.6 に示すオートマトンテンプレートにおいて次のようにモデル化している．ただし，サブプロセス（マルチインスタンス）のオートマトンテンプレートは，同時実行数を表す属性 $loopCardinality$ の値によってロケーションや遷移の数が異なる．ここでは同時実行数が 2 の場合を例に説明する．

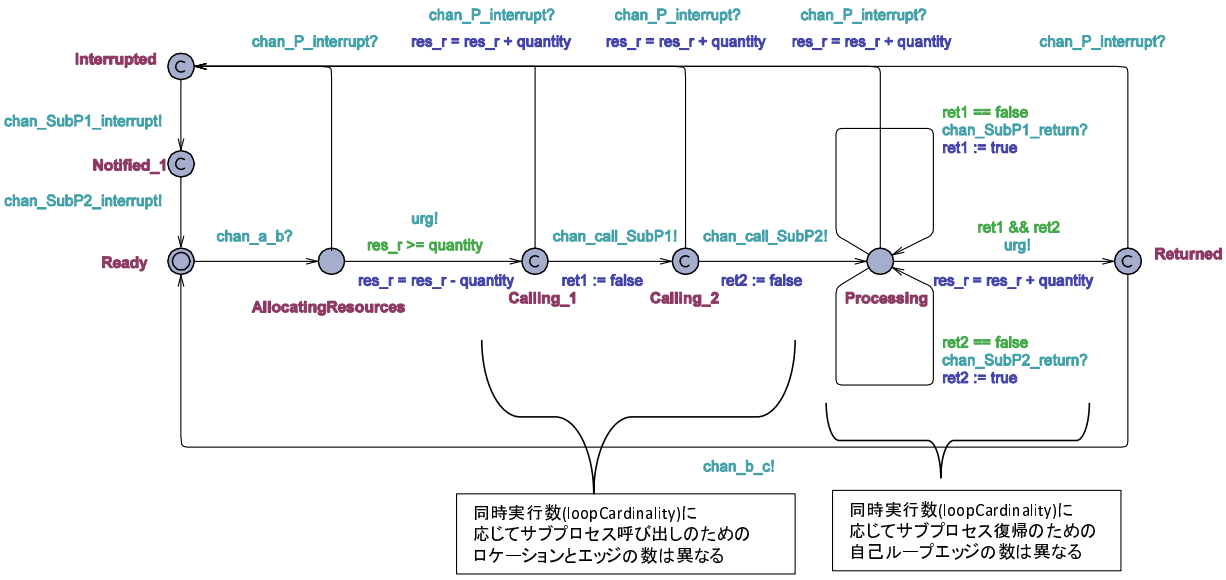

図 6.6: サブプロセス（マルチインスタンス）要素（同時実行数:2）のオートマトンテンプレート

サブプロセス（マルチインスタンス）の状態を表すロケーションとして，直前のフローオブジェクトの完了を待っている状態（$Ready$），2 つの内包プロセスを呼び出す前の状態（$Calling\_1$ および $Calling\_2$），内包プロセスの復帰を待っている状態（$Processing$），全ての内包プロセスが復帰した状態（$Returned$）を用意する．

性質 (1) については，$Ready$ から $Allocating\ Resources$ の遷移と $Returned$ から $Ready$ の遷移に，前後のフローオブジェクトと同期するためのアクション（$chan\_a\_b?, chan\_b\_c!$）を指定しており，サブプロセス（タイムアウト）要素の場合と同様である．

性質 (2) については，内包プロセス呼出の直前すなわち $Allocating\ Resources$ から $Calling\_1$ の遷移上のガード条件（$res\_r \geq quantity$）および代入式（$res\_r := res\_r - quantity$）と，内部プロセスの復帰時すなわち $Processing$ から $Returned$ の遷移上の代入式（$res\_r := res\_r + quantity$）によってモデル化しており，サブプロセス（タイムアウト）要素の場合と同様である．

性質 (3) については，$Calling\_1$ から $Calling\_2$ の遷移および $Calling\_1$ から $Processing$ の遷移に，それぞれの内包プロセス内の開始イベントと同期するためのアクション（$chan\_call\_Sub1!$，$chan\_call\_Sub2!$）を指定している．２つの内包プロセスは同時に呼び出す必要があるため，$Calling\_2$ はコミットロケーションに指定し，時間の経過を許さないようにしている．さらに，$Processing$ から $Processing$ への自己遷移において，2 つの内包プロセス内の終了イベントと同期するためのアクション（$chan\_Sub1\_return?$，$chan\_Sub2\_return?$）と代入式（$ret1 := true$，$ret2 := true$）を指定している．両方の内包プロセスの終了イベントと同期すると，$Processing$ から $Returned$ の遷移上のガード条件（$ret1$ && $ret2$）を満たし $Returned$ に遷移する．この遷移上には，ブロードキャストのアージェントチャネル送信（$urg!$）を指定しているため，両方の内包プロセスが終了すると時間の経過なしに遷移が行われる．チャネル $chan\_call\_Sub\{1,2\}$ と $chan\_Sub\{1,2\}\_return$ はパラメータであり，テンプレート利用時にそれぞれ内包プロセスの開始イベントおよび終了イベントと同一のチャネルが割り当てられる．

性質 (4) については，自身を内包する上位のサブプロセス（タイムアウト）においてタイムアウトが発生した場合に，自身の実行を停止するため，$Allocating\ Resources$，$Calling\_1$，$Calling\_2$，$Processing$ および $Retruned$ に $Interrupted$ への遷移を用意し，タスクのオートマトンテンプレートと同様に，遷移上にはタイムアウトの通知を受けるためのブロードキャストチャネルの同期アクション（$chan\_P\_interrupt?$）を指定している．さらに，自身の内包プロセス２つの実行を停止するため，$Interrupted$ から $Notified\_1$ への遷移と $Notified\_1$ から $Ready$ に設定されたブロードキャストチャネルの同期アクション（$chan\_SubP1\_interrput!$，$chan\_SubP2\_interrput!$）により，内包プロセスに含まれるフローオブジェクトにタイムアウトの通知を伝播させている．$chan\_P\_interrupt$，$chan\_SubP1\_interrput$ および $chan\_SubP2\_interrput$ は，プロセスの包含関係により決定されるパラメータである．

6.4 変換手続きの形式表現

拡張 BPMN から拡張時間オートマトンへの変換は，それぞれのモデルの抽象構文（4.3.2，5.3）を用いて形式的に記述でき，変換手続きは完全に自動化が可能である．

まず，既に述べたように拡張時間オートマトンのネットワークは，拡張 BPMN モデルに出現するフローオブジェクトに対応する拡張時間オートマトンと起動用オートマトンで構成されるため，以下のように記述できる．

定義 4 (対応関係) $\mathcal{M} = (\mathcal{Q}, top, \mathcal{S}^{\diamond}, \mathcal{R}^{\diamond}, \mathsf{map}, \mathsf{HR}, \mathsf{RIni})$ を拡張 BPMN モデル (定義 3) とすると，$\mathcal{M}$ に対応する拡張時間オートマトンのネットワークは，以下で定義される拡張時間オートマトンの集合 $A^{\diamond}$ である．

$$A^{\diamond} = \bigcup_{\mathcal{P} \in \mathcal{Q}} \{A_{(o,n)} | o \in \mathcal{O}_{\mathcal{P}} \wedge n \leq \mathsf{a\_ins}(\mathcal{P})\} \cup \{\mathcal{A}_{main}\}$$

ただし，

- $A_{(o,n)}$ は，フローオブジェクト o（の実行インスタンス）に対応する拡張時間オートマトンである．自然数 $n \in \{1, 2, ..., \mathsf{a\_ins}(\mathcal{P})\}$ は，サブプロセス（マルチインスタンス）に包含されることによって同時実行されるフローオブジェクトのインスタンスを識別するための番号である．一般には，サブプロセス（マルチインスタンス）は入れ子構造を成すことができるため，ある拡張 BPMN プロセス $\mathcal{P}$ が同時実行されるインスタンスの数 $\mathsf{a\_ins}(\mathcal{P})$ は，トップレベルのプロセスとの階層構造の間に存在するサブプロセス（マルチインスタンス）の同時実行数を掛け合わせた数となるため，以下の通り定義される．

  $$\mathsf{a\_ins}(\mathcal{P}) = \begin{cases} 1 \quad (\mathcal{P} = top \text{ の場合}) \\ \mathsf{Loop}(\mathsf{map}^{-1}(\mathcal{P})) * \mathsf{a\_ins}(\mathsf{par}(\mathcal{P})) \quad (\mathcal{P} \neq top \text{ かつ } \mathsf{map}^{-1}(\mathcal{P}) \in \mathcal{S}^{\mathcal{MIP}}_{\mathsf{par}(\mathcal{P})} \text{の場合}) \\ \mathsf{a\_ins}(\mathsf{par}(\mathcal{P})) \quad (\text{その他の場合}) \end{cases}$$

  ここで，$\mathsf{par}(\mathcal{P})$ は，全てのプロセス $\mathcal{P} \in \mathcal{Q}$ に対し親プロセスを返す関数とする．つまり，$(\mathsf{par}(\mathcal{P}), \mathcal{P}) \in \mathsf{HR}$ である．

- A_{main} は，起動用オートマトンである．

定義 4 の $A_{(o,n)}$，すなわち，フローオブジェクト o に対応する拡張時間オートマトンは，フローオブジェクトの種類別に抽象構文を用いて記述することができる．例えば，タスクに対応する拡張時間オートマトンは以下のように記述することができる．

定義 5 (タスクに対応する拡張時間オートマトン) 拡張 BPMN モデル $\mathcal{M}$ を構成する拡張 BPMN プロセス $\mathcal{P} = (\mathcal{O}, \mathcal{A}, \mathcal{E}, \mathcal{G}, \mathcal{T}, \mathcal{S}, \mathcal{S}^{\mathcal{N}}, \mathcal{S}^{\mathcal{T}}, \mathcal{S}^{\mathcal{R}}, \mathcal{S}^{\mathcal{MIP}}, \mathcal{E}^{\mathcal{S}}, \mathcal{E}^{\mathcal{IT}}, \mathcal{E}^{\mathcal{E}}, \mathcal{G}^{\mathcal{F}}, \mathcal{G}^{\mathcal{J}}, \mathcal{G}^{\mathcal{X}}, \mathcal{G}^{\mathcal{M}}, \mathcal{R}, \mathcal{F}, \mathcal{F}^{\mathcal{E}}, \mathsf{Duration}, \mathsf{Min}, \mathsf{Max}, \mathsf{Loop}, \mathsf{Res})$ に含まれるタスク $o \in \mathcal{T}$ に対応する拡張時間オートマ

トン $A_{(o,n)} = (L, C, G, B, U, D, X, V, I, E, l_{ini})$ は以下の通りである．

$$
\begin{aligned}
L &= \{Ready_{(o,n)}, AllocatingResources_{(o,n)}\} \cup \{Working_{(o,n)}\} \\
C &= \varnothing \\
G &= \varnothing \\
B &= \{chan_{((x,n),(o,n))} | x \in \mathsf{in}(o)\} \cup \{chan_{((o,n),(x,n))} | x \in \mathsf{out}(o)\} \cup \\
&\quad \{chan_{((\mathcal{P},n),interrupt)}, urg_{(o,n)}\} \\
U &= \{chan_{((\mathcal{P},n),interrupt)}, urg_{(o,n)}\} \\
D &= \{chan_{((\mathcal{P},n),interrupt)}, urg_{(o,n)}\} \\
X &= \{t_{(o,n)}\} \\
V &= \{res_{\mathsf{rid}(o)}\} \\
I &= \{(Working_{(o,n)}, t_{(o,n)} \leq \mathsf{Max}(o)\} \\
E &= \{(Ready_{(o,n)}, chan_{((x,n),(o,n))}?, true, \langle\rangle, AllocatingResources_{(o,n)}) | x \in \mathsf{in}(o)\} \cup \\
&\quad \{(AllocatingResources_{(o,n)}, urg_{(o,n)}!, res_{\mathsf{rid}(o)} \geq \mathsf{rquant}(o), \\
&\quad \langle t := 0, res_{\mathsf{rid}(o)} := res_{\mathsf{rid}(o)} - \mathsf{rquant}(o)\rangle, AllocatingResources_{(o,n)}) | \mathsf{x} \in \mathsf{in}(o)\} \cup \\
&\quad \{(Working_{(o,n)}, chan_{((o,n),(x,n))}!, t_{(o,n)} \geq \mathsf{Min}(o), \langle \mathsf{res}_{\mathsf{rid}(o)} := \mathsf{res}_{\mathsf{rid}(o)} + \mathsf{rquant}(o)\rangle, \\
&\quad Ready_{(o,n)}) | x \in \mathsf{out}(o)\} \cup \\
&\quad \{(AllocatingResources_{(o,n)}, chan_{((\mathcal{P},n),interrupt)}?, true, \langle\rangle, Ready_{(o,n)}), \\
&\quad (Working_{(o,n)}, chan_{((\mathcal{P},n),interrupt)}?, true, \langle res_{\mathsf{rid}(o)} := res_{\mathsf{rid}(o)} + \mathsf{rquant}(o)\rangle, Ready_{(o,n)})\} \\
l_{ini} &= Ready_{(o,n)}
\end{aligned}
$$

ただし，$\mathsf{rid}(o)$，$\mathsf{rquant}(o)$ はそれぞれ，アクティビティ $o \in \mathcal{A}$ が利用する資源およびその利用数を返す関数であり，$\mathsf{Res}(o) = (\mathsf{rid}(o), \mathsf{rquant}(o))$ である．

基本的に図 6.4 に示したオートマトンテンプレートの構造通りにロケーションや有向枝の集合を定義している．他のフローオブジェクトとの関係や制約によって決定されるチャネルや不変式，ガード条件，代入式に関する集合の定義には，拡張 BPMN プロセス $\mathcal{P}$ が持つ集合を用いている．例えば，チャネル集合 B は，拡張 BPMN プロセス $\mathcal{P}$ に定義された in や out，すなわちフローオブジェクト o の前後に接続されているフローオブジェクト集合を返す関数によって定義されるチャネルが含まれている．他の種類のフローオブジェク

トについても，同様の方法でチャネル集合の定義が行われているため，シーケンスフローで接続されているフローオブジェクト同士は，同じ名称のチャネルを共有することになる．また，不変式，ガード条件，代入式に関する集合 I や E については，拡張 BPMN プロセス $\mathcal{P}$ に定義された制約を表す属性値を返す関数，すなわち $\mathsf{Max}, \mathsf{Min}, \mathsf{Res}$ を用いて定義している．

同様にして，他の種類の拡張 BPMN 要素についても対応する拡張時間オートマトンを形式的に記述することができる．

第 7章

適用実験

7.1　ケーススタディ（1）：看護・介護のビジネスプロセスへの適用

7.1.1　看護・介護のビジネスプロセス

例題として，2.1.2 で導入した看護・介護ドメインのビジネスプロセスを対象に，ビジネスプロセスの性質をモデル検査ツール UPPAAL を用いて検証する．図 7.1 は，拡張 BPMN で記述した看護・介護ドメインのビジネスプロセスであり，フローオブジェクトに説明のための識別子（$a, b, c, ..., r$）を付記したものである．

患者は全部で 6 名，各 3 名ずつの 2 グループに分かれており，以下に示す順序で食事やリハビリ，入浴を行う．

- 全ての患者は，決められた時刻（正午）に一斉に昼食を開始する．
- グループ 1 の患者は，食事をとり終えた順に各々リハビリ室に移動し，リハビリを行って病室に戻る．
- グループ 2 の患者は，食事をとり終えた順に各々浴室に移動し，入浴を済ませて病室に戻る．

また，各活動や移動にかかる所要時間，サポートに必要な看護師の数，専用室の利用制限は下記の通りである．

- 食事には 30 分から 60 分かかり，グループ 1 では看護師が各患者に 1 名ずつ付いてサポートする必要がある．
- 病室とリハビリ室の間の移動には 3 分から 5 分かかり，看護師が各患者に 1 名ずつ付いてサポートする必要がある．
- リハビリ室では最大で同時に 2 名の患者がリハビリすることができる．
- 病室と浴室の間の移動には 1 分から 5 分かかり，看護師が各患者に 2 名ずつ付いてサポートする必要がある．
- 浴室は最大で 2 名の患者が入浴することができ，看護師が各患者に 1 名ずつ付いてサポートする必要がある．

7.1.2 拡張 BPMN から拡張時間オートマトンへの変換

拡張 BPMN で記述された図 7.1 のビジネスプロセスを，6 章に示した下記の手順に従って拡張時間オートマトンに変換する．

(1) ビジネスプロセスモデル中のすべてのフローオブジェクトを，それぞれの種類に応じたオートマトンテンプレートに置換する．

(2) フローオブジェクト間の接続関係や制約値に基づいて，オートマトンテンプレートのパラメータに具体的なチャネルや定数値を割り当てて，完全な拡張時間オートマトン（のネットワーク）モデルにする．

(3) 最後にトップレベルプロセスの開始イベントと終了イベントを呼出／復帰させるための起動用オートマトンを追加する．

手順 (1)

本手順では，図 7.1 に出現するフローオブジェクトを抽出し，それぞれフローオブジェクトの種類に応じたオートマトンテンプレートを準備する．表 7.1 は，ビジネスプロセスに出現するフローオブジェクトと用意したオートマトンテンプレートの対応関係を整理したものである．例えば，ビジネスプロセスに出現する開始イベント（a, g, m）には，それ

ぞれ開始イベントに対応するオートマトンテンプレートを準備する．ただし，サブプロセス（マルチインスタンス）に内包されている開始イベント（g, m）に関しては，その同時実行数だけオートマトンテンプレートを複製して準備する．例えば，開始イベント g に関しては，サブプロセス（マルチインスタンス）c の同時実行数（属性 $loopCadinarity$ の値）と同数（3）のオートマトンテンプレート（A_{g1}, A_{g2}, A_{g3}）を準備する．本節では，識別子 x のフローオブジェクトに対応するオートマトンテンプレートを A_{xn} という識別子で呼ぶことにする．n は，サブプロセス（マルチインスタンス）に内包されることで複製されたオートマトンテンプレートを区別する自然数の番号である．結果，ビジネスプロセスに出現する 18 のフローオブジェクトは，合計で 42 のオートマトンテンプレートに置換されることになる．

表 7.1: 出現するフローオブジェクトとオートマトンテンプレート一覧

No	フローオブジェクト		オートマトンテンプレート
1	開始イベント	a, g, m	$A_{a1}, A_{g1}, A_{g2}, A_{g3}, A_{m1}, A_{m2}, A_{m3}$
2	終了イベント	l, r, f	$A_{l1}, A_{l2}, A_{l3}, A_{r1}, A_{r2}, A_{r3}, A_{f1}$
3	AND 分岐	b	A_{b1}
4	AND 併合	e	A_{e1}
5	サブプロセス（マルチインスタンス）	c, d	A_{c1}, A_{d1}
6	タスク	h, i, j, k, n, o, p, q	$A_{h1}, A_{h2}, A_{h3}, A_{i1}, A_{i2}, A_{i3}, A_{j1}, A_{j2}, A_{j3}, A_{k1}, A_{k2}, A_{k3}, A_{n1}, A_{n2}, A_{n3}, A_{o1}, A_{o2}, A_{o3}, A_{p1}, A_{p2}, A_{p3}, A_{q1}, A_{q2}, A_{q3}$
	合計数	18	42

手順 (2)

本手順では，手順 (1) で準備したオートマトンテンプレートのパラメータに具体的なチャネルや定数値を割り当てる．例として，タスク i に対応するタスク要素用のオートマトンテンプレート A_{i1} のパラメータの具体化について説明する．A_{i1} のパラメータは次の通りである．

- チャネル
  - $chan\_a\_b$
  - $chan\_b\_c$
  - $chan\_P\_interrupt$
- 定数値
  - min
  - max
  - res_r
  - $quantity$

タスク i の前後に接続されるフローオブジェクトは，タスク h およびタスク j であるから，オートマトンテンプレート A_{i1} のチャネルパラメータ $chan\_a\_b$，$chan\_b\_c$ にはそれぞれ chan_h1_i1，chan_i1_j1 というチャネルを割り当てる．また，タスク i を包含するサブプロセス要素は c であるから，チャネルパラメータ $chan\_P\_interrupt$ は，chan_c_interrupt というブロードキャストチャネルを割り当てる．さらに，タスク i のもつ 4 つの属性 $MinTime$，$MaxTime$，$Resource$，$Quantity$ の値を，それぞれ定数値パラメータ min，max，res_r，$quantity$ に割り当てることで，A_{i1} は完全な拡張時間オートマトンとなる．

上記を手順 (1) で準備したオートマトンテンプレートすべてに行う．

手順 (3)

最後に，起動用オートマトンを追加し，起動用オートマトンのチャネルパラメータ $chan\_call\_top$ と開始イベント a に対応するオートマトン A_{a1} のチャネルパラメータ $chan\_call\_P$ に，同一のチャネル chan_call_a1 を割り当てる．同様に，起動用オートマトンのチャネルパラメータ $chan\_top\_return$ と終了イベント r に対応するオートマトン A_{r1} のチャネルパラメータ $chan\_P\_return$ に，同一のチャネル chan_r1_return を割り当てる．また，ビジネスプロセス全体の属性 $Resources$ に従い，資源（看護師数，浴室数，リハビリのエリア数）を初期化するための代入式を構成する．

ここまでの手順により，ビジネスプロセスの動作仕様を表現した UPPAAL の拡張時間オートマトンのネットワークモデルを得ることができる．

7.1.3 モデル検査ツール UPPAAL による性質の形式検証

前節で得られた拡張時間オートマトンのネットワークは，モデル検査ツール UPPAAL を用いて性質を形式的に検証することができる．図 7.1 の看護・介護のビジネスプロセスの例題では，以下の性質が満たされることが求められている．

(1) 入浴を済ませた後に浴室で 5 分を超えて待たせることがあってはならない.

(2) 全員が食事を開始してリハビリまたは入浴を済ませて病室に戻るまでに 3 時間を超えてはならない.

上記の性質は，UPPAAL の検証式として次のように記述することができる．

(1) $AG\ not\ ((A_{q1}.AllocatingResources\ and\ A_{q1}.wt > 5)\ or\ (A_{q2}.AllocatingResources\ and\ A_{q2}.wt > 5)\ or\ (A_{q3}.AllocatingResources\ and\ A_{q3}.wt > 5))$

(2) $AG\ not\ (Ma.Processing\ and\ Ma.t > 180)$

検証式（1）の A_{q1}, A_{q2}, A_{q3} は，グループ 2 の 3 人の患者の “病室へ移動” タスクに対応する拡張時間オートマトンである．wt は，入浴後に看護師が到着するまでの時間をカウントするための追加のクロック変数である（図 7.2）．検証したい性質に応じて，状態を監視するための補助的な変数を拡張時間オートマトンに追加し，それが 5（分）以上になることがないことを指定している．検証式（2）の Ma は起動オートマトンであり，トップレベルプロセスの呼出からの経過時間を計測するためのクロック変数 t を用いて 3 時間（180 分）以上かかることはないことを表している．

ここでは上記の性質を満たすために必要な看護師の人数を見積もるケースを考える．特に性質（1）は患者の健康状態に影響を与えるため，いかなる場合においても満たされる必要がある．本例題は非常に単純なビジネスプロセスだが，時間や資源を考慮すると，レビューやシミュレーションといった手法では網羅的な検証が困難である．本実験では，ビジネスプロセスをモデル検査ツール UPPAAL に上記の検証式を与え，看護師の数を変化させて性質が充足されるかどうかを網羅的に検証した．検証の結果，性質が充足されなければ，ビジネスプロセス上の看護師の数（または，必要であればプロセス構造や制約値）を変更し，それを提案した手順に従って再度拡張時間オートマトンモデルに変換して，繰り返し検証を実施することができた．具体的には，看護師数を 2 人から順に増加させてモデル検査を繰り返し，性質（1）と性質（2）がそれぞれ満足する看護師数を求めた．結果

として，性質（1）を満たすためには看護師が 8 人以上，性質（2）は 3 人以上必要であることが確認できた．

また，性質が満たされない場合に UPPAAL が出力する反例を分析することで，どのような悪い状況が起きうるかを確認することができた．例えば，看護師が 6 人の場合は，下記のような状況が発生する可能性があり，性質 (1) を充足しない．

- グループ 1 の 3 名は食事中．すなわち，3 名の患者に対し 3 名の看護師が介助中．
- グループ 2 の 3 名のうち 1 名 (患者 A) が浴室へ移動を開始，もう 1 名 (患者 B) は入浴を開始．すなわち，浴室への移動で 2 名，入浴で 1 名の看護師がサポート中．
- グループ 2 の残りの 1 名 (患者 C) は入浴を済ませて，病室への移動待ち（看護師が 2 名必要）．

この状況は，プロセスが開始されて 47 分後に発生しうる状況であり，患者 C の移動に必要な看護師 2 名が確保できない．グループ 1 の 3 名の食事，患者 B の入浴については，それぞれ最大で 13 分，30 分かかる可能性があり，患者 A の移動も最大で 5 分かかる可能性があるため，最悪のケースでは患者 C を入浴後に 5 分以上待たせてしまうことになる．

看護師が 7 名の場合は，上記の状況は回避できるものの，以下のような状況が発生する可能性があり，同じく性質 (1) を充足しない．

- グループ 1 の 3 名は食事中．すなわち，3 名の患者に対し 3 名の看護師が介助中．
- グループ 2 の 3 名のうち 1 名 (患者 A) が入浴中．すなわち 1 名の看護師がサポート中．
- グループ 2 の 3 名のうちの残り 2 名は，1 名 (患者 B) がちょうど入浴を済ませたところで，もう 1 名 (患者 C) が病室への移動を（少し前から）待っている．

この状況では，サポート可能な看護師が残り 3 名のため，入浴を済ませた 2 名を 1 名ずつ順番に移動させる必要があるが，誤って患者 B を先に移動させてしまうと，少し前から移動を待っている患者 C は最悪のケースで 10 分間（すでに待っていた時間 5 分＋患者 B の移動が終わる最大時間 5 分）移動を待つことになってしまう．この反例は，患者を浴室から病室に移動させる際に入浴を済ませた順に行う，という看護師の業務規則を設定すれば回避が可能であるが，規則が守られなかった場合の最悪のケースを考えれば看護師を 8 名配置することが望ましい．

表 7.2: ケーススタディ（1）：実験結果

性質	看護師数	検証結果	状態数	時間 (sec)	メモリ (MB)
（1）	7	False	22258	3.1	96.7
	8	True	856289	686.3	3047.2
（2）	2	False	767538	75.4	1409.4
	3	True	1417190	337.8	4105.9

モデル検査の実行は，Amazon Web Service EC2 M1 ラージインスタンス（4 ECU，7.5GB メモリ）[1]，Ubuntu 12.04 LTS（64bit），UPPAAL 4.1.13（64bit）の環境で行い，各検証における状態数，実行時間，使用メモリは表 7.2 の通りである．状態数に応じて実行時間や使用メモリが増加するが，本ケーススタディでは最大でも 12 分程度，メモリも 4GB 程度で検証できており，十分に実用的な範囲で計算できた．

7.2 ケーススタディ（2）：Discussion Cycle プロセスへの適用

7.2.1 Discussion Cycle プロセス

図 7.3 は，文献 [35] に BPMN の記述例として掲載されているビジネスプロセス “Discussion Cycle” の一部を改変し，拡張 BPMN で記述したものである．このプロセスは，管理者が複数の作業グループに議題をアナウンスするタスク “Announce Issues for Discussion” から開始される．管理者は各作業グループに専任の議長を割り当て，電子メールによる議論を開始させる（“Moderate E-mail Discussion”）．ここで，この組織には専任の議長が 2 名在籍しているものとする．“Moderate E-mail Discussion” は，サブプロセス（タイムアウト）要素によってタイムアウトが 7 日に指定されているため，議論は 7 日以内に終了するか，7 日経過時点で中断され，その時点で各作業グループの議長は管理者に状況を報告する．一方，管理者は議論が開始されてから 6 日経過した時点で，各作業グループに警告を出す（“E-mail Discussion Deadline Warning”）．上記の一連のプロセス，すなわち “Resolve Issues” サブプロセス（マルチインスタンス並列）は，同時に 2 並列で実行することができる．最後

に，管理者は各議題の状況を確認し，議題が解決されたかどうかを決定する（“Evaluate Discussion Progress”）．解決されていない議題は，次のサイクルで再度アナウンスされる．

7.2.2 拡張 BPMN から拡張時間オートマトンへの変換

ケーススタディ（1）と同様に，6 章に示した 3 つの手順に従って拡張時間オートマトンに変換する．結果として，表 7.3 に示す拡張時間オートマトンを含むモデルを得た．

表 7.3: 変換手続きの結果得られた拡張時間オートマトンの一覧

No	フローオブジェクト		拡張時間オートマトン
1	開始イベント	a, d, i, q	$A_{a1}, A_{d1}, A_{i1}, A_{i2}, A_{q1}, A_{q2}$
2	終了イベント	c, h, p, s	$A_{c1}, A_{h1}, A_{p1}, A_{p2}, A_{s1}, A_{s2}$
3	中間イベント（タイマー）	m	A_{m1}, A_{m2}
4	AND 分岐	j	A_{j1}, A_{j2}
5	AND 併合	o	A_{o1}, A_{o2}
6	サブプロセス（ループ）	b	A_{b1}
7	サブプロセス（マルチインスタンス）	f	A_{f1}
8	サブプロセス（タイムアウト）	k	A_{k1}, A_{k2}
9	タスク	e, n, g, r	$A_{e1}, A_{n1}, A_{n2}, A_{g1}, A_{r1}, A_{r2}$
10	-	-	Ma（起動用オートマトン）
	合計数	18	28

7.2.3 モデル検査ツール UPPAAL による性質の形式検証

Discussion Cycle ビジネスプロセスでは，以下の性質が満たされることが求められている．

(1) ビジネスプロセスが予期せず停止しないこと．

(2) 5 つの議題を 2 週間以内に解決できる可能性があること．

各性質は，UPPAAL の検証式としてそれぞれ下記の通り表現することができる．

(1) $AG\ (not\, deadlock\, or\, Ma.Done)$

(2) $EF\ (Ma.issue \geq 5\, and\, Ma.t \leq 14)$

検証式の中の Ma は起動用オートマトンである．起動用変数 $Ma.t$ はトップレベルプロセスの呼出からの経過時間を計測するためのクロック変数である．変数 $Ma.issue$ は，議題の解決総数を監視するためにモデルに追加した整数変数である．“Discussion” タスクに対応する拡張時間オートマトンに対し，その動作が完了するたびに変数 issue が増加するように代入式を追加している（図 7.4）．

UPPAAL に上記の検証式を与え検証を実行したところ，性質 (1)，(2) ともに満足しないことが報告された．性質 (1) の結果を分析すると，“Moderate E-mail Discussion” から出力される 2 つのシーケンスフローが AND 併合に入力されているが，これらのシーケンスフローが 2 つとも入力されることはなく，AND 併合が待ち状態となってデッドロックが発生していた．これは，ビジネスプロセスの構造上の問題であり，これを図 7.5 の通り修正した．

改めて検証を実行したところ，性質 (1) は満たされ，性質 (2) は満たされないことが報告された．すなわち，ビジネスプロセスは停止することなく最後まで動作するが，5 つの議題を 2 週間以内で解決できる可能性がないことが確認できた．

性質 (2) を満足させるため，“Resolve Issues” プロセスの同時実行インスタンス数を増加させることを考える．現在の同時実行数は 2 のため，“Resolve Issues” の属性 $loopCondition$ を 3 に増加させて再度検証を行う．6 章で示した手順に従って UPPAAL の検証モデルを再構築して検証を再実行すると，再び性質 (2) を満足しないという結果が報告された．UPPAAL の報告する実行トレースを分析すると，“Moderate E-mail Discussion” の実行時には，資源として議長（chair）が一人必要となるが，この組織が抱える議長は 2 人であるため，同時実行数を 3 としても実際には議論を 3 つ並行して実行することができていなかったことが分かった．すなわち，議論を並行して数多くこなすためには，それに応じて必要な資源（議長）を確保する必要があることを示唆しており，この場合では議長を 3 人確保することで性質 (2) を満足させることができることを確認した．

この問題は，同時実行数の増加に伴い資源数がボトルネックとなって発生しており，本研究で着目した時間，資源および同時実行数の各制約が互い関連しあって引き起こされる典型的な問題である．特にビジネスプロセスモデルが複雑な場合は，このような制約条件をすべて一貫して調整することは困難であり，確認も困難であるため問題が発生しやすい．

表 7.4: ケーススタディ（2）：実験結果

性質	同時実行数	議長数	検証結果	状態数	時間 (sec)	メモリ (MB)
(1)	2	2	True	2007	0.05	5.7
	3	3	True	26274	1.16	10.5
(2)	2	2	False	2047	0.02	6.0
	3	2	False	42807	1.23	15.0
		3	True	7515	0.19	8.5

モデル検査の実行は，Amazon Web Service EC2 M1 ラージインスタンス（4 ECU，7.5GB メモリ）[1]，Ubuntu 12.04 LTS（64bit），UPPAAL 4.1.13（64bit）の環境で行い，各検証における状態数，実行時間，使用メモリは表 7.4 の通りである．本ケーススタディではすべてのケースで数秒，メモリも数十 MB で検証できており，十分に実用的な範囲で計算できた．

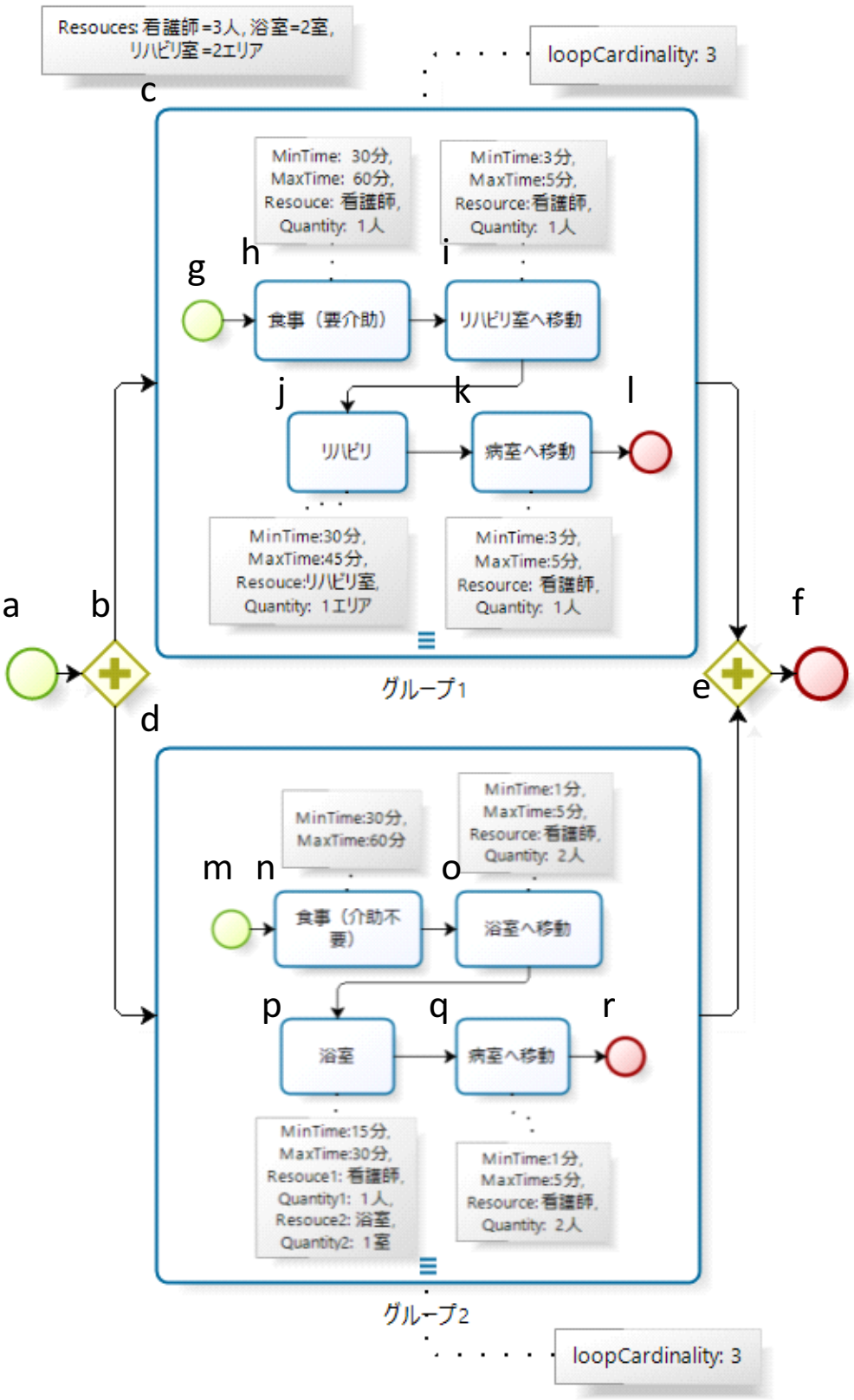

図 7.1: 看護・介護のビジネスプロセス

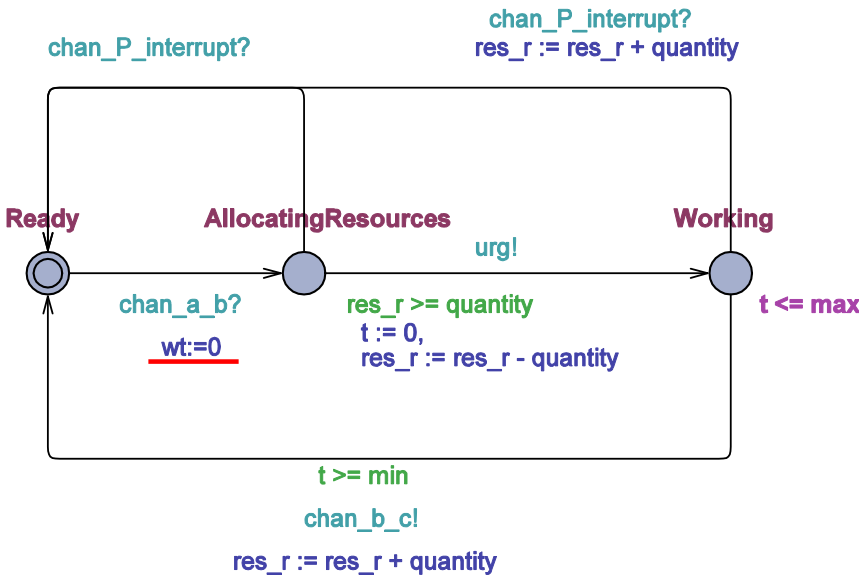

図 7.2: “病室へ移動” タスクに対応する拡張時間オートマトンへのクロック変数 wt の追加

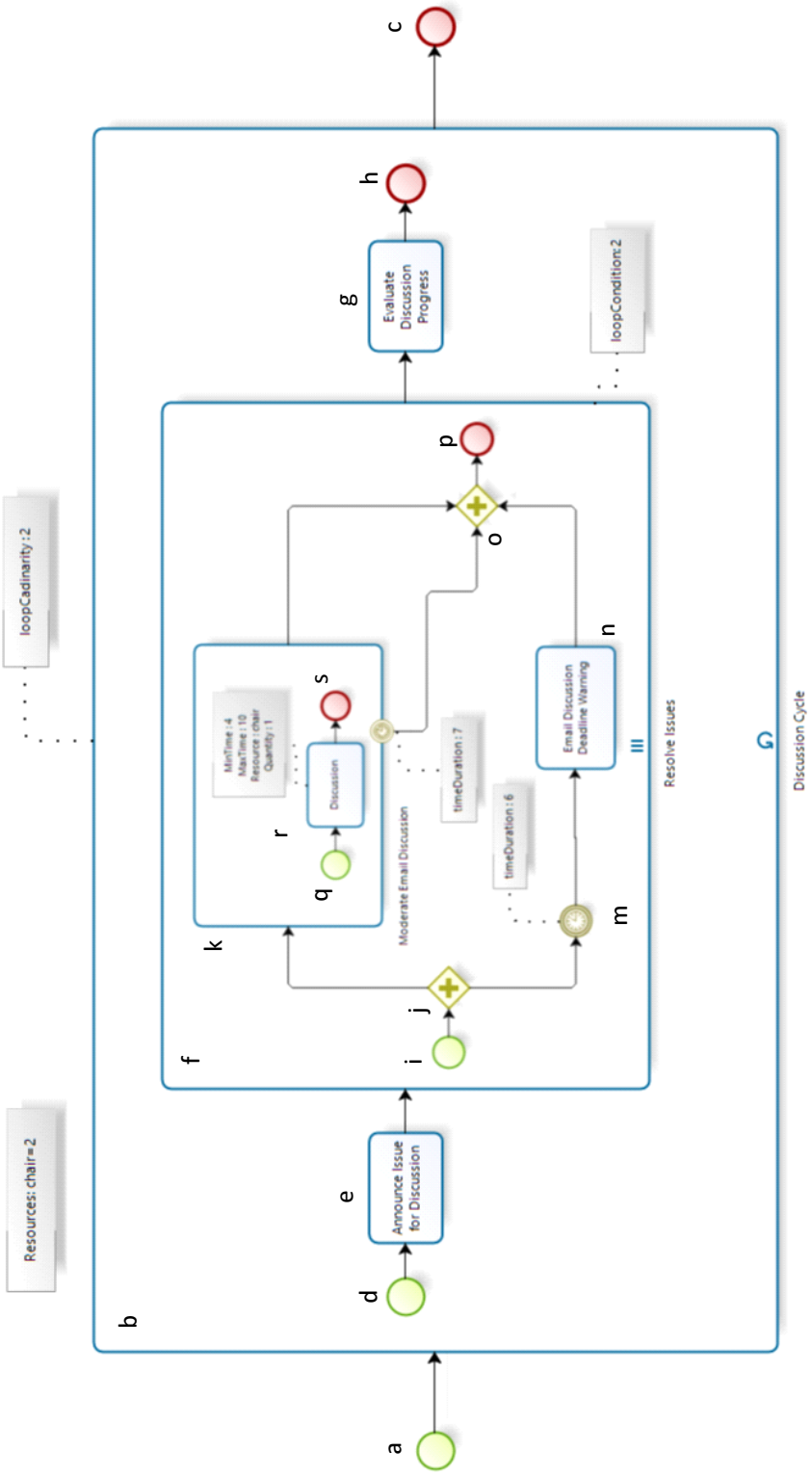

図 7.3: ビジネスプロセス：Discussion Cycle

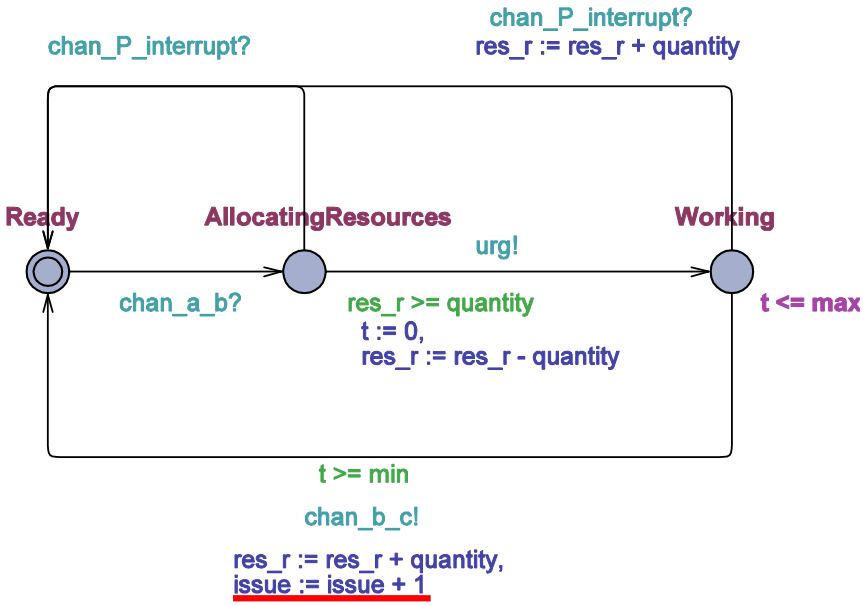

図 7.4: Discussion タスクに対応する拡張時間オートマトンへの代入式追加

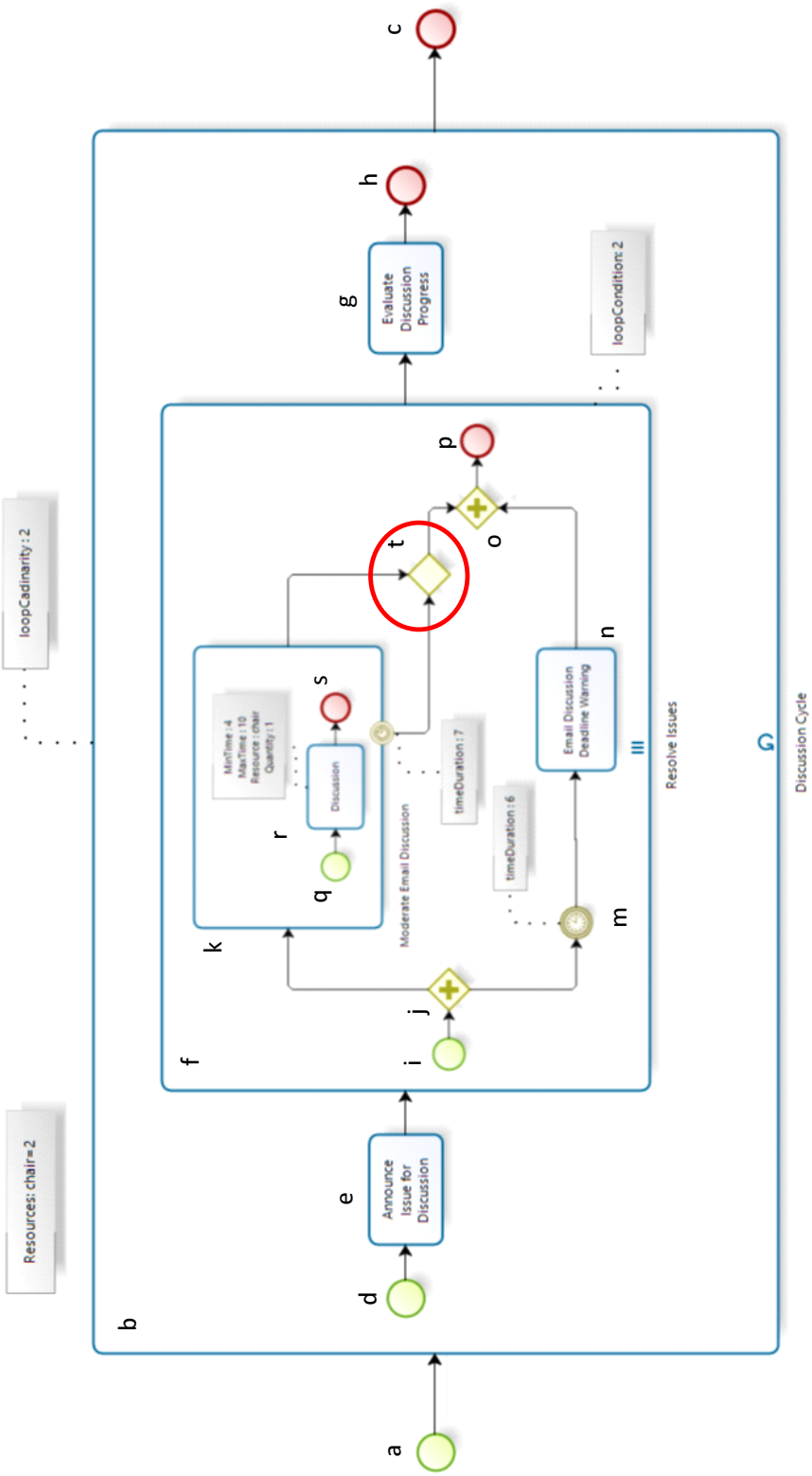

図 7.5: ビジネスプロセス：Discussion Cycle（修正）

第 8章

評価

8.1 BPMN から時間オートマトンへの変換

本研究では，モデル検査技法をビジネスプロセスに適用するために，ビジネスプロセスの標準記法である BPMN から時間オートマトンへの変換手法を提案した．BPMN が様々なタイプのフローオブジェクトを接続することでビジネスプロセスを構成していることに着目し，ビジネスプロセスに出現するフローオブジェクト毎に 1 つの時間オートマトンに対応付けてモデル化した．

一般に，ビジネスプロセスの設計段階では，プロセスの構造や制約の値を繰り返し変更して調整を行うことになる．この場合，ビジネスプロセスモデルの変更をモデル検査ツールが扱う検証モデルに繰り返し反映させなくてはならないが，ビジネスプロセスの構造を大きく変える場合，検証モデルの変更も大きくなるため手作業での変更は容易ではない．さらに，同時実行数などを調整する場合，ビジネスプロセスのモデル上では単なるパラメータ値の変更であっても，検証モデル上では単なる変数値の変更とはならず，モデル構造の変更やプロセス数の増減を伴うことになり，各所の整合性を保って正確に変更を加えることは困難となる．

本手法では，拡張 BPMN の要素に合わせて時間オートマトンをモジュール化し，あらかじめテンプレートとして用意しておくことで，検証モデルの導出を系統的に行えるようにした．フローオブジェクト同士の接続関係や具体的な制約値など，ビジネスプロセスの構成内容によって異なるチャネルや定数はパラメータとして定義しておき，変換時に具体的な値を設定できるようにした．

本研究で提案した時間オートマトンへの変換手続きは，拡張 BPMN と時間オートマトン

の両モデルの抽象構文を用いて形式的に定義され，それを変換ツールとして実装すれば完全に自動化が可能であり，さらなる効率化が期待できる．ただし，検証すべき性質に応じて，変換後の時間オートマトンモデルに時間や変数を監視するための時間オートマトンや変数の追加が必要となる場合がある．例えば，ケーススタディ（1）ではタスクに対応する時間オートマトンのクロック変数 wt を追加している．

8.2 モデル検査による性質検証

本研究では一般にリアルタイムシステムの検証に用いられるモデル検査技法およびツールを，ビジネスプロセスの検証に適用する手法を提案した．モデル検査においてシステムの正しさを形式的に定義する検証性質は，大きく安全性と活性に分類される．安全性 (safety) は「システムが決して悪い状態に到達しない」という性質である．活性 (liveness) は「システムがいつかは良い状態に到達する」という性質である．これらは，ビジネスプロセスのコンテキストにおいても，ケーススタディで検証したような重要かつ有効な性質に対応し，これらの性質をモデル検査の技法を用いて網羅的に検証できることは有用であると考える．

- 入浴を済ませた後に浴室で 5 分を超えて待たせることがあってはならない.（安全性）
- 5 つの議題を 2 週間以内に解決できる可能性があること.（活性）

本研究では，ビジネスプロセスにおける時間および資源に関する制約を扱い，リアルタイムシステム向けのモデル検査ツール UPPAAL を用いて，上記のような時間や資源を含む性質を検査することができることが確認できた．時間や資源の制約を扱うことによって，ビジネスプロセスの実行順序に関わる欠陥だけでなく，時間や資源に関する制約が関連して発生する，業務レベルのより本質的な欠陥を検出することができた．また，欠陥を排除するためにビジネスプロセスの構造や時間や資源に関するパラメータを調整して繰り返し検証を行うことで，無駄のない資源の配置計画などに役立てることができることを確認できた．

適用実験の結果からもわかるように，変換の結果得られる時間オートマトンの数は，ビジネスプロセスに含まれるフローオブジェクトの数と，サブプロセス（マルチインスタンス）の同時実行数に依存する．特に，サブプロセス（マルチインスタンス）に含まれる内部プロセスのフローオブジェクトは，各々が独立して別の状態を持つため，同時実行数分

表 8.1: ケーススタディ（2）の性質（1）検証の同時実行数に応じた状態数の推移

同時実行数	議長数	検証結果	状態数	時間 (sec)	メモリ (MB)
2	2	True	2007	0.05	5.7
3	3	True	26274	1.16	10.5
4	4	True	328837	20.8	55.0
5	5	True	3956528	339.9	533.8

だけ複製して用意する必要がある．時間オートマトンの数はモデル検査における計算量に直結する．例えば，表 8.1 はケーススタディ (2) において同時実行数を増加させた際の状態数や検証時間の推移であり，指数関数的に増加してしまう．しかしながら，BPMN のサブプロセス（マルチインスタンス）の内部プロセス群は，ほとんどの場合，互いに区別できない無名プロセスであり，半順序関係を利用した手法（Partial Order Method）[32] を適用すれば，状態空間を大きく削減することが可能である．また，本手法が対象とするビジネスプロセスの検証というコンテキストにおいては，同時実行数を含む制約パラメータの絶対的な値は重要ではなく，相対的に値を設定して検証できれば十分であることが多いと考える．また，設定値の精度についても細かな単位での設定は求められないことが多いため，本手法のもつスケーラビリティの範囲で目的を果たす局面は十分にあると考える．また，大規模なビジネスプロセスモデルに本手法を適用する場合には，サブプロセスの階層ごとに必要な性質の検証を行うことで規模の問題を回避することができると考える．

第 9章

関連研究

ビジネスプロセスの形式検証に関する研究は，文献 [34] や文献 [13] において調査，整理されている．古くから研究が行われている分野だが，より複雑な振る舞いをもつプロセスモデルへの対応や時間的な性質を考慮した検証については，現在でも重要な研究分野となっている [13]．ここでは，本研究の関連研究を二つに大別して整理する．一方は，主にビジネスユーザ向けの表記法で記述されたビジネスプロセスを，形式的なモデルに変換して厳密な意味を与え，性質を形式的に検証できるようにすることを目的とした研究である．他方は，ワークフローモデルの構造や性能に関する特定の性質を解析・検証することを目的とした研究である．

9.1 ビジネスプロセス表記法に厳密な意味を与えて形式検証を行う研究

文献 [20] では，BPMN の時間制約拡張（Time BPMN）が提案されている．アクティビティ間の様々な時間制約や依存関係が整理されているが，典型的な時間制約の一つであるアクティビティの実行経過時間制約（Time Duration）が含まれず，モデルの検証メカニズムも提供されていない．

文献 [15, 40] では，BPMN モデルをペトリネットに対応付けることによって厳密な意味を与えている．BPMN ツールが出力する XML を PNML(Petri Net Markup Language) に変換するツールを作成し，PNML を入力とする分析ツール ProM[16] を用いてビジネスプロセス内にデッドタスクがないこと，および不完全なプロセス実行がないことの 2 つの性質

を検証できることを示している．BPMN の比較的大きなサブセットを対象として形式的な意味を与えているが，ビジネスプロセスの実行順序の側面のみにフォーカスしており，時間や資源に関する性質を扱うことはできない．

文献 [12] では，BPMN モデルを Event-B[6] に変換する手法が提案されている．ビジネスプロセスの実行順序（制御フロー）だけでなく，データフローを対象としてモデル化し，それらの一貫性を検証することができるが，時間の性質や資源の競合を考慮して性質を検証することはできない．

文献 [47] では，BPMN をワークフロー記述言語 YAWL[42] に変換するためのアルゴリズムとツールを提供し，BPMN の形式的な意味を与えている．ただし，変換は制御フローにのみ着目した単純なマッピングであり，時間や資源の制約は扱われていない．

文献 [45] では，BPMN モデルをプロセス代数 CSP（Comunication Sequential Processes）[39] に対応付けて形式的な意味を与え，プロセスの実行順序に関する性質をモデル検査技法を用いて検証できるようにしている．さらに，[45] の筆者は文献 [46] において，タスクの実行間隔に関する時間制約（例：14 日〜20 日の間隔で実行する）を導入し，その制約のもとでタスク間の実行順序に関する性質が保証されるかどうかを検証できるように手法を拡張している．ここで扱われている時間制約は本研究で対象としている 3 つの時間制約（2.1.2）とは異なる．また，検証性質はプロセスの実行順序に限定されており，本研究のように時間の側面を含む性質を直接検証することはできない．

文献 [22] では，時間制約を含む BPMN モデルをペトリネットに対応付けて分析を行う手法が提案されている．モデル構造の問題として，デッドタスク，デッドロック，無限ループを検出できることに加えて，アクティビティに与えた実行経過時間制約（Time Duration）の間の矛盾を検出することができる．ただし，アクティビティの間の経過時間制約（Time Distance）を扱うことができない．

文献 [33] では，BPMN で記述されたビジネスプロセスに参加する作業者に着目し，それぞれの作業者の振る舞いを時間オートマトンでモデル化する手法を提案している．タスクの実行経過時間（Time Duration）を与えて UPPAAL を用いて検証することができるが，資源に関する性質は考慮されていない．

文献 [37] では，BPMN モデルをプロセス代数 COWS（Calculus of Orchestration of Web Services）[27] の確率拡張に変換する手法が提案されている．タスクの実行経過時間 (Time Duration) に関する制約を確率とともに与え，プロセス全体が指定の時間内に終わる確率をモデル検査ツール PRISM[26] で計算できることを示している．ただし，アクティビティ

間の経過時間制約（Time Distance）や競合資源に関する制約を与えて検証することはできない．

文献 [31] では，ワークフロー記述言語 YAWL に対して時間制約を与えるための拡張を行い，それを LTL モデル検査ツール DiVinE[9] のモデル記述言語 DVE に変換する手法が提案されている．YAWL を拡張し，タスクの実行経過時間制約 (Time Duration)[1] ，連続した 2 つのアクティビティの間の経過時間制約（Time Distance），デッドライン制約（Deadline）をビジネスプロセスモデルに与えて性質を検証することができるが，資源に関する制約を与えることはできない．

文献 [21] では，時間や資源に関する制約を付加したワークフローモデルを時間オートマトンに変換し，UPPAAL を用いて検証する手法を提案している．ただし，対象のワークフローモデルは BPMN と比較して極めて単純であり，ループやマルチインスタンスの同時実行，サブプロセスによる階層構造を扱えない．また，時間制約はタスクの実行経過時間制約 (Time Duration) のみであり，アクティビティ間の経過時間制約（Time Distance）やデッドライン制約（Deadline）を与えることはできない．

文献 [38] では，ビジネスプロセスの仕様化言語のひとつである PPML(Product Process Modeling Language) を UPPAAL の時間オートマトンに変換し形式的な意味を与えている．PPML には，処理の最大・最小時間（Time Duration）を与えることができるが，アクティビティ間の経過時間制約（Time Distance）やデッドライン制約（Deadline）は指定できない．また，PPML はデータフロー型のプロセスモデルであり，制御フロー型の BPMN とは異なる．

文献 [25] では，実行可能なビジネスプロセス言語 BPEL4WS の時間的側面をモデル化するために WSTTS（Web Services Timed Transition Systems）というフォーマリズムを提案し，モデル検査ツール NuSMV で検証を行う手法が提案されている．デッドロックの検出，指定された時間経過後に終了することの検証のほか，プロセス実行にかかる最小・最大時間を計算できる．ただし，BPEL4WS の時間的側面にフォーカスしており，用途が限定された手法である．

以上のように，これまでにも BPMN を含むビジネスユーザ向けのビジネスプロセス記述法を，オートマトン，ペトリネット，プロセス代数といった形式モデルに変換し，ビジネスプロセスの実行順序に関する性質や時間的な性質を形式的に検証しようとする手法が提案されている．しかし，それぞれの研究を細かく見ていくと，時間制約はアクティビティ内の

[1] 最大実行時間のみ．最小実行時間は与えることはできない．

時間制約（Time Duration）が中心であり，アクティビティ間の時間制約（Time Distance）やデッドライン制約（Deadline）を扱う研究は少ない．また，ビジネスプロセスの分析において重要な資源に関わる側面を，時間的な側面とともに考慮して検証を行う研究は少なく，特にBPMNのような比較的複雑な振る舞いの要素を含む記述法に対する手法は確立していない．本研究では，比較的複雑な要素を含むBPMNのサブセットを対象に，アクティビティ内の時間制約，アクティビティ間の時間制約，複数のアクティビティで構成されるサブプロセスのデッドライン制約の3つの時間制約と，アクティビティ間の資源の競合に関連する資源制約を与え，ビジネスプロセスの性質を形式的に検証できる手法を提案した．

9.2 ワークフローモデルの構造や性能に関する特定の性質を解析・検証する研究

文献[18]では，非循環グラフ構造でモデル化されたワークフローグラフに，アクティビティの間の経過時間に対する制約（Time Distance）やアクティビティの実行時刻に対する制約を与え，それらを満足するために必要な個々のアクティビティのデッドラインを計算するCPM（Critical Path Method）をベースとした手法が提案されている．

文献[28]では，各アクティビティに実行経過時間（Time Duration）に関する制約を持たせた時間付ワークフローグラフに，排他利用される資源の概念を追加し，複数のアクティビティの実行の重複可能性と資源への依存関係から，資源競合の可能性を検出する手法が提案されている．

文献[5]では，ワークフローの制御フロー（実行順序）の側面をペトリネットを用いてモデル化したWF-net[41]の健全性（soundness）を定義し，それを解析する手法が提案されている．健全性とは，ワークフローは必ず終了し，終了時に動作している不完全なアクティビティがなく，どのアクティビティも到達可能であるという性質であり，ワークフローの構造的な正しさを定義したものである．

また，文献[30]では，WF-netに対しタスクの実行経過時間制約（Time Duration）の概念を追加したTWF-netを提案し，健全性と時間的な安全性を定義・検証する手法が提案されている．さらに，文献[43]では，排他利用される資源を記述できるようWF-netを拡張したRCWF-netを提案し，同じく健全性を定義・解析する手法が提案されている．

文献[48]では，WF-netに時間と資源の概念を追加し，ワークフローの最大スループットを計算する手法を提案されている．

文献 [17] では，資源制約を持つ複数の並行プロセス間の時間的な違反を動的に検査する手法が提案されている．各プロセスを TWF-net でモデル化し，時間制約の違反のパスと解決方法を出力できる．扱われている時間制約は 1 つで，アクティビティ i が開始されて t 時間単位が経過する前にアクティビティ j が終了する，という k 形式で与えることができる制約である．

文献 [44] では，時間と資源の制約を含むワークフローをモデル化した PetriNet を検証する手法が提案されている．アクティビティの実行経過時間（Time Duration）に関する制約と複数のアクティビティが利用する排他資源を与え，各アクティビティの最早，最遅時刻を計算することができる．

これらの研究で対象としているワークフローモデルは，BPMN などでは記述要素として含まれるループやマルチインスタンスの同時実行などを要素として記述できない（容易に記述することができない）ため，ビジネスプロセスの検討段階で用いられるビジネスユーザ向けの表記法とは隔たりがあり，これらの手法をそのまま適用することは難しい．また，健全性やスループットなど，ワークフローモデルの構造や性能に関する特定の性質を改善することを目的とした解析手法であり，本手法のように TCTL ベースの検証式としてより一般的に性質を記述して検証することはできない．

第 10章

おわりに

10.1　まとめ

本研究では，IT システムの開発の重要なインプットであるビジネスプロセスモデルの設計品質を高めるために，ビジネスプロセスが満足すべき性質を，形式手法の一つであるモデル検査技法を用いて網羅的に検証する手法を提案した．特に，ボトルネックなどのビジネスプロセスの重要かつ典型的な問題を分析するために必要となる，時間および資源に関する制約を持つビジネスプロセスの検証にフォーカスした．

ビジネスプロセスモデルの国際的な標準記法である BPMN は，標準では時間および資源に関する制約を十分に記述できない．そのため本研究ではまず，BPMN の持つ拡張機構を用いて，アクティビティの所要時間や排他資源の利用を指定するためのカスタム属性を追加し，ビジネスユーザが時間および資源に関する制約を持つビジネスプロセスをこれまでと同じ作法で記述できるようにした．モデル検査技法によってビジネスプロセスを検証するためには，ビジネスプロセスの振る舞いをモデル検査が扱うことができる形式モデルで表現する必要がある．そこで，本研究では拡張 BPMN で記述したビジネスプロセスモデルを時間の扱いに優れた時間オートマトンモデルに変換する手法を考案した．BPMN が様々なタイプのフローオブジェクトを接続することでビジネスプロセスを構成していることに着目し，ビジネスプロセスに出現するフローオブジェクトをそれぞれ 1 つの時間オートマトンに対応付けてモデル化する方針とした．フローオブジェクトの持つ振る舞いや時間や資源に関する制約は，状態遷移の構造，チャネル同期を伴うアクション，クロック変数や整数変数を含む不変式，ガード条件および代入式によって表現した．一方，プロセスの構造，すなわちシーケンスフローやサブプロセスの呼出／復帰／タイムアウト割り込み

によって決定される実行順序の制御は，順序関係を持つフローオブジェクト同士が同一のチャネルで送受信（同期）を行うことによって表現した．モデルの変換を効率的に行えるようにするため，あらかじめフローオブジェクトのタイプごとに時間オートマトンのテンプレート（オートマトンテンプレート）を用意しておき，それらを用いて系統的な手順で変換できるようにした．フローオブジェクト同士の接続関係や具体的な制約値など，ビジネスプロセスの構成内容によって異なるチャネルや定数はパラメータとして定義しておき，変換時に具体的な値を設定できるようにした．

提案手法を評価するために，特定の状況がクリティカルとなる看護・介護ドメインのビジネスプロセスを含む 2 つのビジネスプロセスに適用する実験を行った．実験では，時間オートマトンを入力とする実用的なモデル検査ツール UPPAAL を用い，時間および資源に関連した性質を時相論理（CTL）で記述して検証を行った．ビジネスプロセスの構造や資源数などの属性を繰り返し変更して検証を行うことができ，例えば，クリティカルな状況にならないことが保証される最低限の資源の数の特定などを行うことができることを確認した．これより，従来の研究が主として対象としていたプロセスの実行順序に関する問題だけでなく，時間と資源に関連した重大な問題を初期の設計段階において排除し，ビジネスプロセスの設計品質を保証することができることを確認した．

10.2　今後の課題

6 章で述べたように，BPMN は自然言語によって仕様が記述されており，明確なセマンティクスは与えられていない．よって，BPMN から形式モデルへの変換は一意ではなく，その正当性を証明することはできない．すなわち，本研究で提案した BPMN のフローオブジェクトに対応する時間オートマトンの定義は，事例研究を重ねることによって妥当性の確認と洗練を行っていく必要がある．組織やドメインによって解釈が異なる場合には，それらに応じて時間オートマトンの定義を柔軟に変更するために，仕様の解釈と時間オートマトンの対応関係のパターン化などの手当てが必要となる可能性もある．

本研究で提案した拡張 BPMN から拡張時間オートマトンへの変換は，それぞれのモデルの抽象構文を用いて形式的に記述しており，変換手続きは完全に自動化が可能である．より大きなビジネスプロセス例題への適用や繰り返しの変換を効率化するためには，BPMN ツールの出力を入力とし，UPPAAL の検証対象システム定義（時間オートマトンモデル）を生成する変換ツールの実装が必要であり，今後の課題としたい．

モデル検査ツールによる検証では，検証性質を時間オートマトンモデルに基づくCTLなどの時相論理による検証式で与える必要がある．一方，ビジネスプロセスモデルにおける要求事項は一般に自然言語であり，それらを時相論理で表現するためには，変換後の時間オートマトンモデルを理解し，注意深く検証式を組み立てる必要がある．ビジネスプロセスモデルにおける要求事項（性質）の種類を整理し，検証式の記述・構成方法を対応付けてパターン化できれば，検証式の作成を効率化できると考える．また，モデル検査ツールが出力する検証結果や検証トレースをもとに，ビジネスプロセスモデル上の欠陥や改善個所を特定するためには，変換後の時間オートマトンモデルの理解が必要となる．本研究で提案した変換は，BPMNと時間オートマトンの対応は基本的に1対1となるため，時間オートマトンモデル上の実行トレースをBPMNによるビジネスプロセスモデル上でトレースすることはそれほど難しくないが，検証式の作成とともにビジネスユーザに対する支援が必要と考える．

謝辞

本研究を行なうに当たり, 終始御指導を賜わった平石 邦彦教授，国立情報学研究所の石川 冬樹准教授に深謝致します.

参考文献

[1] Amazon Web Service. http://aws.amazon.com/.

[2] Business Process Model and Notation. http://www.bpmn.org/.

[3] Unified Modeling Language. http://www.uml.org/.

[4] UPPAAL. http://www.uppaal.org/.

[5] Wil M. P. van der Aalst. Workflow verification: Finding control-flow errors using petri-net-based techniques. In *Business Process Management, Models, Techniques, and Empirical Studies*, pages 161–183, London, UK, UK, 2000. Springer-Verlag.

[6] Jean-Raymond Abrial. *Modeling in Event-B: System and Software Engineering*. Cambridge University Press, New York, NY, USA, 1st edition, 2010.

[7] R. Alur, C. Courcoubetis, and D. Dill. Model-checking for real-time systems. In *Logic in Computer Science, 1990. LICS '90, Proceedings., Fifth Annual IEEE Symposium on e*, pages 414–425, 1990.

[8] Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

[9] Jiri Barnat, Lubos Brim, and Petr Rockai. Divine 2.0: High-performance model checking. *2010 Ninth International Workshop on Parallel and Distributed Methods in Verification, and Second International Workshop on High Performance Computational Systems Biology*, 0:31–32, 2009.

[10] Gerd Behrmann, Re David, and Kim G. Larsen. A tutorial on uppaal. In *Formal Methods for the Design of Real-Time Systems*, Lecture Notes in Computer Science, pages 200–236, 2004.

[11] Johan Bengtsson and Wang Yi. Timed automata: Semantics, algorithms and tools. In *In Lecture Notes on Concurrency and Petri Nets*, Lecture Notes in Computer Science vol 3098, pages 87–124. Springer-Verlag, 2004.

[12] Jeremy W. Bryans and Wei Wei. Formal analysis of bpmn models using event-b. In *Proceedings of the 15th international conference on Formal methods for industrial critical systems*, FMICS'10, pages 33–49, Berlin, Heidelberg, 2010. Springer-Verlag.

[13] Saoussen Cheikhrouhou, Slim Kallel, Nawal Guermouche, and Mohamed Jmaiel. Survey on time-aware business process modeling. In *In Proceedings of the 15th International Conference on Enterprise Information Systems (ICEIS)*. ScitePress, 2013.

[14] E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Trans. Program. Lang. Syst.*, 8(2):244–263, April 1986.

[15] Remco M. Dijkman, Marlon Dumas, and Chun Ouyang. Semantics and analysis of business process models in BPMN. *Inf. Softw. Technol.*, 50(12):1281–1294, 2008.

[16] B.F. Dongen, A.K.A. Medeiros, H.M.W. Verbeek, A.J.M.M. Weijters, and W.M.P. Aalst. The prom framework: A new era in process mining tool support. In Gianfranco Ciardo and Philippe Darondeau, editors, *Applications and Theory of Petri Nets 2005*, volume 3536 of *Lecture Notes in Computer Science*, pages 444–454. Springer Berlin Heidelberg, 2005.

[17] YanHua Du, Pengcheng Xiong, Yushun Fan, and Xitong Li. Dynamic checking and solution to temporal violations in concurrent workflow processes. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 41(6):1166–1181, 2011.

[18] Johann Eder, Euthimios Panagos, and Michael Rabinovich. Time constraints in workflow systems. In *In Proc. of the 11 th International Conference on Advanced Information Systems Engineering (CAiSE' 99), Springer Verlag, LNCS 1626*, pages 286–300. Springer-Verlag, 1999.

[19] Henning Dierks Ernst-Ruediger Olderog. *Real-Time Systems: Formal Specification and Automatic Verification*. Cambridge University Press, 2008.

[20] D. Gagne and A. Trudel. Time-bpmn. In *Commerce and Enterprise Computing, 2009. CEC '09. IEEE Conference on*, pages 361–367, 2009.

[21] V. Gruhn and R. Laue. Using timed model checking for verifying workflows. In Jose Cordeiro and Joaquim Filipe, editors, *Proceedings of the 2nd Workshop on Computer Supported Activity Co ordination*, pages 75–88. Insticc Press, 2005.

[22] Wenjia Huai, Xudong Liu, and Hailong Sun. Towards trustworthy composite service through business process model verification. In *Ubiquitous Intelligence Computing and 7th International Conference on Autonomic Trusted Computing (UIC/ATC), 2010 7th International Conference on*, pages 422–427, 2010.

[23] F. Jahanian and A.K. Mok. Safety analysis of timing properties in real-time systems. *Software Engineering, IEEE Transactions on*, SE-12(9):890–904, 1986.

[24] Stefan Kätker and Susanne Patig. Model-driven development of service-oriented business application systems. *Proceeding of Wirtschaftsinformatik*, 2009:171–180, 2009.

[25] R. Kazhamiakin, P. Pandya, and M. Pistore. Representation, verification, and computation of timed properties in web. In *Web Services, 2006. ICWS '06. International Conference on*, pages 497–504, 2006.

[26] Marta Kwiatkowska, Gethin Norman, and David Parker. Prism 2.0: A tool for probabilistic model checking. In *QEST '04: Proceedings of the The Quantitative Evaluation of Systems, First International Conference*, pages 322–323, Washington, DC, USA, 2004. IEEE Computer Society.

[27] Alessandro Lapadula, Rosario Pugliese, and Francesco Tiezzi. A calculus for orchestration of web services. In Rocco Nicola, editor, *Programming Languages and Systems*, volume 4421 of *Lecture Notes in Computer Science*, pages 33–47. Springer Berlin Heidelberg, 2007.

[28] Hongchen Li, Yun Yang, and T. Y. Chen. Resource constraints analysis of workflow specifications. *J. Syst. Softw.*, 73(2):271–285, October 2004.

[29] Weiping Li and Yushun Fan. A time management method in workflow management system. In *Grid and Pervasive Computing Conference, 2009. GPC '09. Workshops at the*, pages 3–10, 2009.

[30] Sea Ling and H. Schmidt. Time petri nets for workflow modelling and analysis. In *Systems, Man, and Cybernetics, 2000 IEEE International Conference on*, volume 4, pages 3039–3044 vol.4, 2000.

[31] Ahmed Shah Mashiyat, Fazle Rabbi, Hao Wang, and Wendy MacCaull. An automated translator for model checking time constrained workflow systems. In *Proceedings of the 15th international conference on Formal methods for industrial critical systems*, FMICS'10, pages 99–114, Berlin, Heidelberg, 2010. Springer-Verlag.

[32] Marius Minea. Partial order reduction for model checking of timed automata. In *Proceedings of the 10th International Conference on Concurrency Theory*, CONCUR '99, pages 431–446, London, UK, 1999. Springer-Verlag.

[33] L.E.M. Morales. Business process verification using a formal compositional approach and timed automata. In *Computing Conference (CLEI), 2013 XXXIX Latin American*, pages 1–10, 2013.

[34] Shoichi Morimoto. A survey of formal verification for business process modeling. In Marian Bubak, GeertDick Albada, Jack Dongarra, and PeterM.A. Sloot, editors, *Computational Science - ICCS 2008*, volume 5102 of *Lecture Notes in Computer Science*, pages 514–522. Springer Berlin Heidelberg, 2008.

[35] Object Manegement Group. *Business Process Modeling Notation (BPMN) Version 1.2*. Object Manegement Group, 2009.

[36] Object Manegement Group. *Business Process Model and Notation (BPMN) Version 2.0*. Object Manegement Group, 2011.

[37] Davide Prandi, Paola Quaglia, and Nicola Zannone. Formal analysis of bpmn via a translation into cows. In *Proceedings of the 10th international conference on Coordination models and languages*, COORDINATION'08, pages 249–263, Berlin, Heidelberg, 2008. Springer-Verlag.

[38] German Regis, Nazareno Aguirre, and Tom Maibaum. Specifying and verifying business processes using ppml. In Karin Breitman and Ana Cavalcanti, editors, *Formal Methods and Software Engineering*, volume 5885 of *Lecture Notes in Computer Science*, pages 737–756. Springer Berlin Heidelberg, 2009.

[39] A. W. Roscoe, C. A. R. Hoare, and Richard Bird. *The Theory and Practice of Concurrency*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1997.

[40] Tsukasa Takemura. Formal semantics and verification of BPMN transaction and compensation. In *APSCC '08: Proceedings of the 2008 IEEE Asia-Pacific Services Computing Conference*, pages 284–290, Washington, DC, USA, 2008. IEEE Computer Society.

[41] Wil M. P. van der Aalst. The application of petri nets to workflow management. *Journal of Circuits, Systems, and Computers*, 8(1):21–66, 1998.

[42] W.M.P. van der Aalst and A.H.M. ter Hofstede. YAWL: yet another workflow language. *Information Systems*, 30(4):245 – 275, 2005.

[43] Kees van Hee, Natalia Sidorova, and Marc Voorhoeve. Resource-constrained workflow nets. *Fundam. Inf.*, 71(2,3):243–257, February 2006.

[44] Huaiqing Wang and Qingtian Zeng. Modeling and analysis for workflow constrained by resources and nondetermined time: An approach based on petri nets. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 38(4):802–817, 2008.

[45] Peter Y. Wong and Jeremy Gibbons. A process semantics for BPMN. In *ICFEM '08: Proceedings of the 10th International Conference on Formal Methods and Software Engineering*, pages 355–374, Berlin, Heidelberg, 2008. Springer-Verlag.

[46] Peter Y. H. Wong and Jeremy Gibbons. A relative timed semantics for BPMN. *Electron. Notes Theor. Comput. Sci.*, 229(2):59–75, 2009.

[47] JianHong Ye, Shixin Sun, Wen Song, and Lijie Wen. Formal semantics of bpmn process models using yawl. In *Intelligent Information Technology Application, 2008. IITA '08. Second International Symposium on*, volume 2, pages 70–74, 2008.

[48] Shingo Ymaguchi, Keisuke Kuniyoshi, Qi-Wei GE, and Minoru Tanaka. Analysis of resources and computation of maximum throughput in workflows. *IEICE technical report. Circuits and systems*, 102(426):67–72, 20021101.

[49] 崔 舜星, 平石 邦彦, and 内平 直志. 仮想フィールドを用いた看護・介護サービスにおける音声つぶやきコミュニケーションの評価実験について (システム数理と応用). **電子情報通信学会技術研究報告 : 信学技報**, 111(453):17–22, mar 2012.

[50] 中田 明夫. 時間オートマトンによる実時間システムの形式的検証. **計測と制御**, 48(11):803–809, 2009.

本研究に関する発表論文

[1] 綿引 健二，石川 冬樹，本位田 真一，平石 邦彦: “時間, 資源および同時実行数の制約をもつビジネスプロセスの形式検証”，情報処理学会ウインターワークショップ 2010・イン・倉敷論文集 (平成 22 年 1 月).

[2] 綿引 健二，石川 冬樹，平石 邦彦: “時間, 資源の制約を考慮したビジネスプロセスの検証”，電子情報通信学会ソサイエティ大会講演論文集 2011 年 基礎・境界, ”S-21”-”S-22”(平成 23 年 8 月).

[3] K. Watahiki, F. Ishikawa, and K. Hiraishi: “Formal verification of business processes with temporal and resource constraints”，IEEE International Conference on Systems, Man, and Cybernetics(SMC), pp.1173-1180(平成 23 年 10 月).

[4] 綿引 健二，石川 冬樹，平石 邦彦: “時間，資源の制約を持つビジネスプロセスの形式検証”，電子情報通信学会論文誌 (D) Vol.J96-D,No.8, pp.1878-1891(平成 25 年 8 月).

第 A 章

オートマトンテンプレート一覧

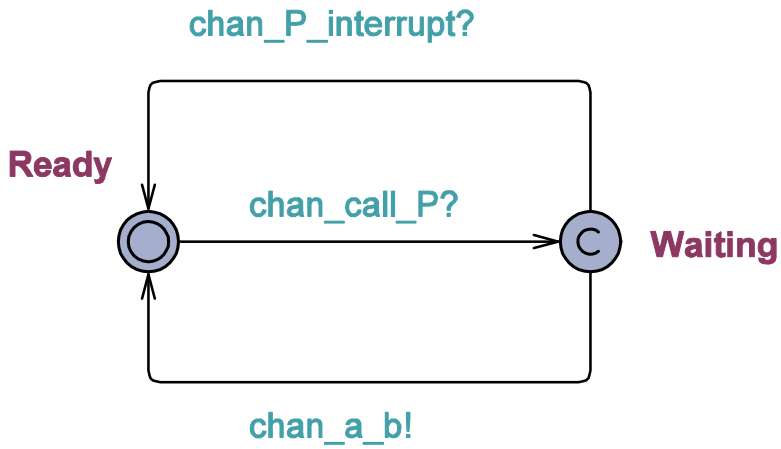

図 A.1: 開始イベントのオートマトンテンプレート

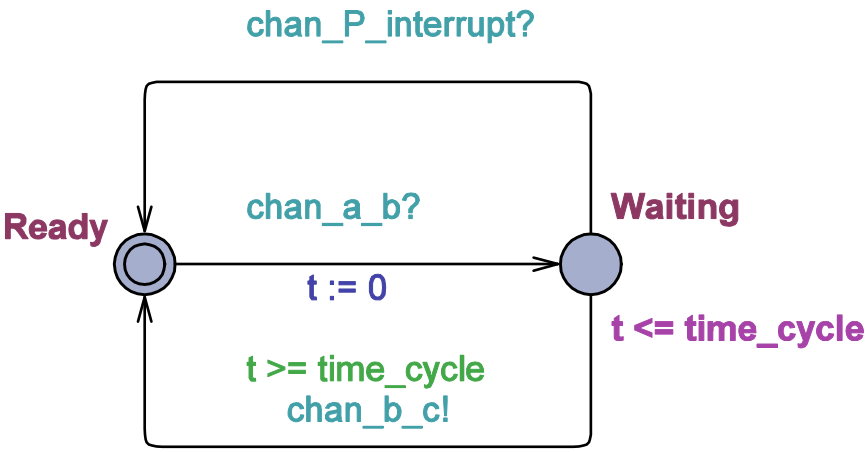

図 A.2: 中間イベントのオートマトンテンプレート

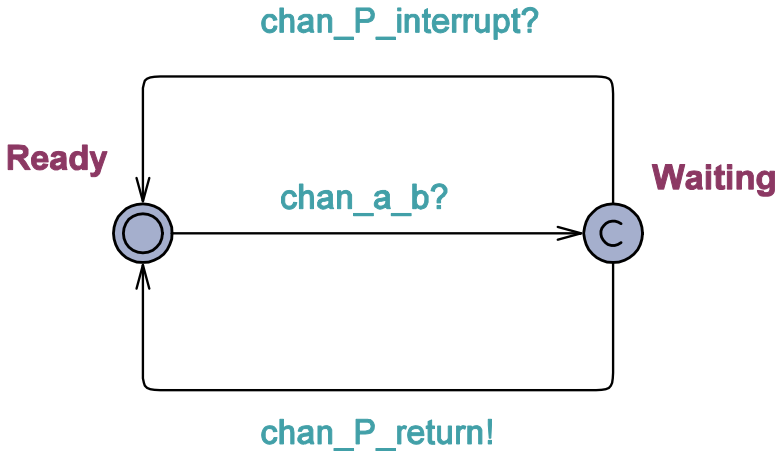

図 A.3: 終了イベントのオートマトンテンプレート

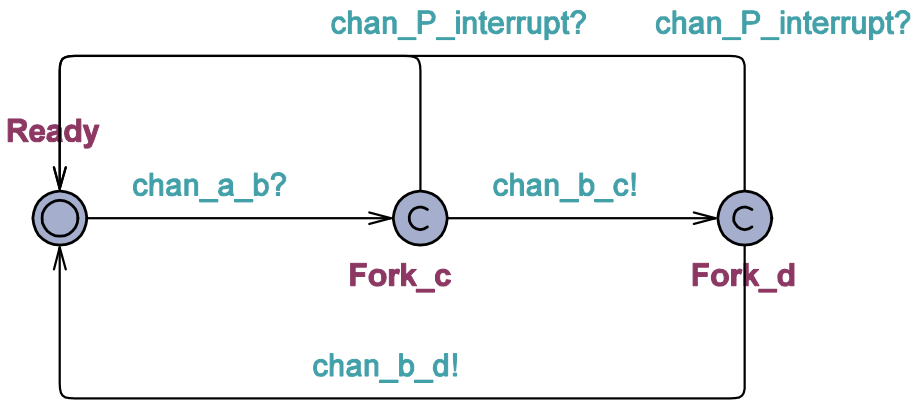

図 A.4: AND 分岐のオートマトンテンプレート

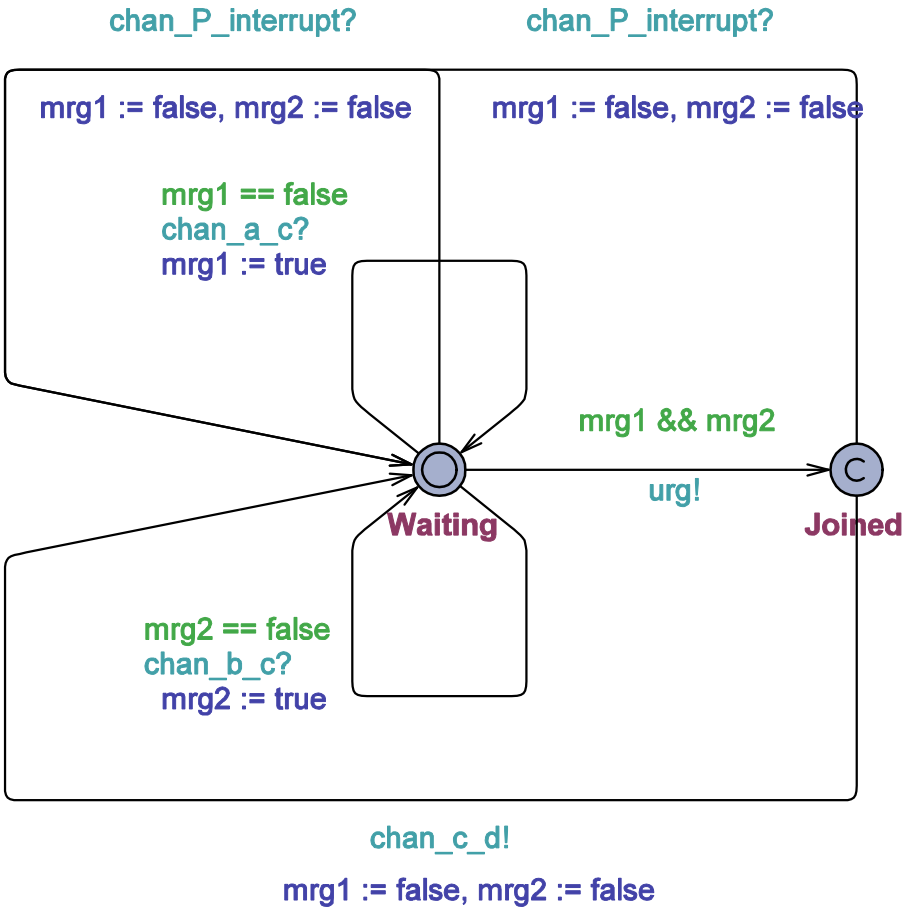

図 A.5: AND 併合のオートマトンテンプレート

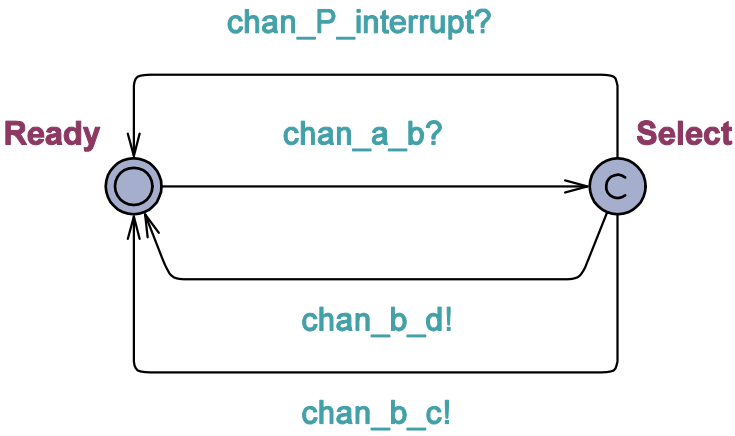

図 A.6: XOR 分岐のオートマトンテンプレート

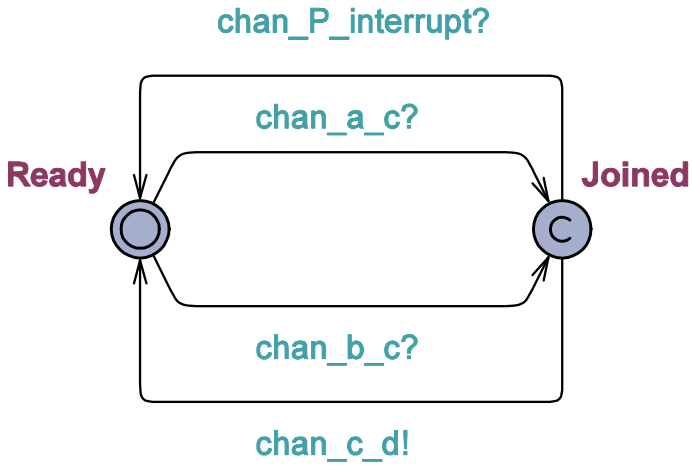

図 A.7: XOR 併合のオートマトンテンプレート

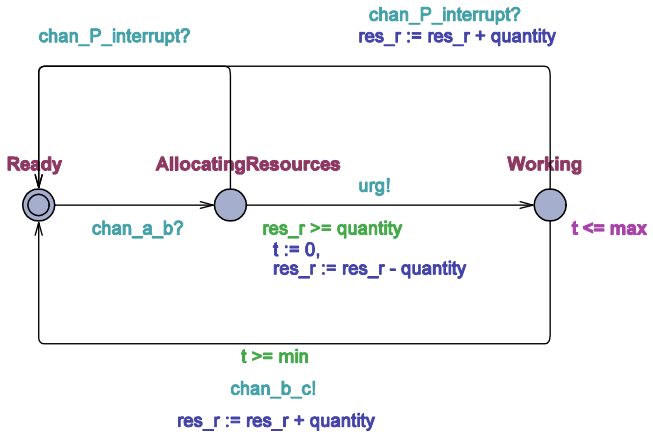

図 A.8: タスクのオートマトンテンプレート

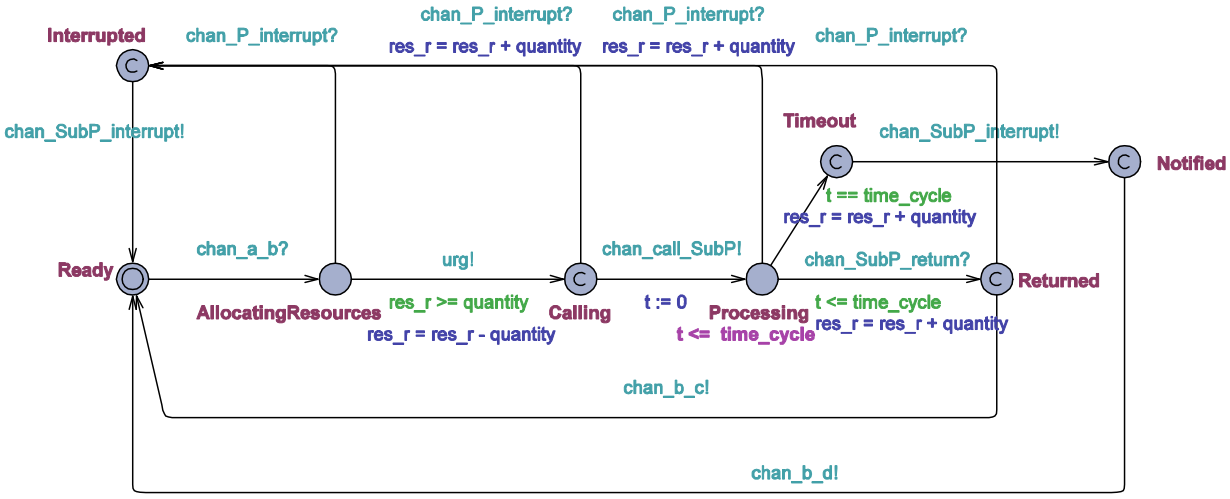

図 A.9: サブプロセス（タイムアウト）のオートマトンテンプレート

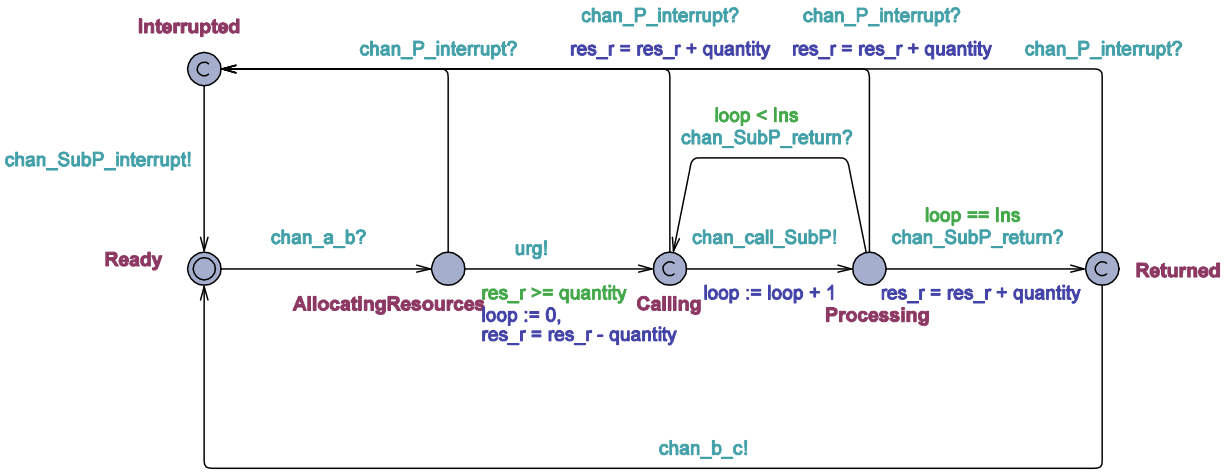

図 A.10: サブプロセス（ループ）のオートマトンテンプレート

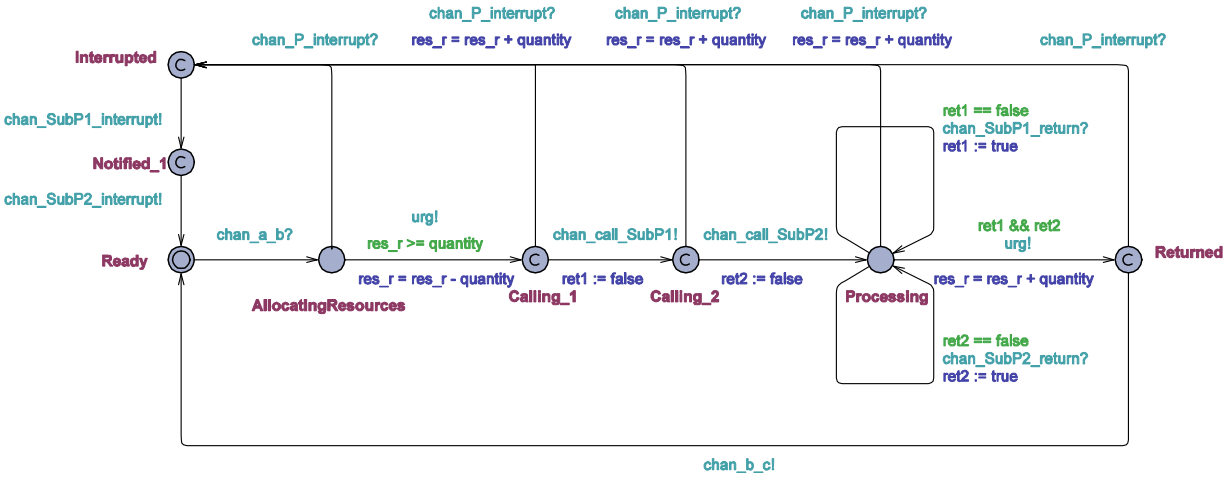

図 A.11: サブプロセス（マルチインスタンス）のオートマトンテンプレート

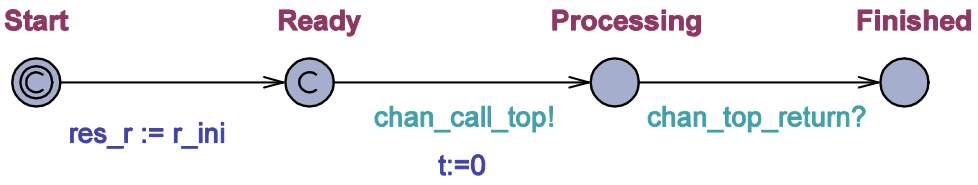

図 A.12: 起動用オートマトンテンプレート