2023年度 修士論文

コントローラメイクスパン最小化フレームワークの適用範囲分割と段階的比較手法

2024 年 1 月 22 日 (月) 提出

指導：本位田　真一 教授

早稲田大学 基幹理工学研究科
情報理工・情報通信専攻
自律エージェント工学 本位田研究室

学籍番号：5122F049-1
清水 優希

目 次

第1章　はじめに

1.1　システム開発と離散制御器合成

システムの開発では，要求仕様を決定し，これに基づいてシステムの設計を行う．要件定義ではシステムを開発する目的をはじめ，システムが保証する諸条件を提示する．システムに求められる諸条件は要求として定義され，これは機能要求と非機能要求の 2 つに分類され，システム設計時に用いられる．

システム設計時には，事前に与えられた安全性をはじめとする要求を充足することを保証することが求められる．近年のシステムは様々なシステムが相互作用する環境に置かれ，その動作環境や動作状況を正確に把握することが難しくなっている．このような状況で要求充足に関する分野に強い関心が置かれており，その中でも特に安全性要求は飛行制御 [1][2]，産業ロボット [3]，自動運転 [4][5]，医療分野 [6] と幅広い分野で特に重要視されている．安全性要求の充足が損なわれるとシステムの停止や事故の原因となり，経済的損失はもちろんのこと，最悪の場合人的被害にまで影響する可能性がある．

既存の開発手法として，要求を充足するように動作仕様を策定し，これに基づいて設計したシステムをテストすることによって検証する方法や，動作仕様そのものをモデル化して要求の充足可能性を網羅的に検証する方法が存在する [7] ．代表的なモデル検査の手法は，動作環境を状態遷移モデルとしてモデル化，要求を線形時相論理を用いて形式化し，モデルが要求を充足できるかを検証する．

また，同じく既存の開発手法のひとつに制御器合成というものがあり，連続系のシステム制御 [8] と離散システム制御 [9] の両分野で研究が行われている．本研究で扱うのはそのうちの後者である離散制御器合成である．離散制御器合成は，動作環境のモデルと要求からシステムの動作仕様を自動的に生成する技術である．動作環境は離散事象システム [10] で与えられ，要求は線形時相論理式 [11] の集合から定式化される．離散制御器合成では安全性や到達性，活性といった要求の充足を保証する動作仕様の自動合成が可能であり，これを用いることで開発時だけでなく運用時にも適用することができる．

1.2 コントローラにおける非機能要求に着目した研究

離散制御器合成で扱われる安全性や到達性といった要求は機能要求に属し，一方でコストやメイクスパン (動作の実行時間) は非機能要求に属する．非機能要求を充足しないことによる損失は機能要求と比較して必ずしも致命的ではないが，時間的・経済的損失や運用時の性能に影響を与えるため，システムを開発するにあたり重要な要素の一つである [12].

非機能要求を扱う要求充足に関する研究も行われており [13] ，その中でもコストなどのパラメータを扱った研究では，定量的な比較手法 [14] と定性的な比較手法 [15] が存在する．前者は具体的な数値による正確な動作仕様の生成が可能な一方で，分析を行うための具体的な数値を獲得することが困難であるという欠点が存在する．システム開発にあたっては設計者がその動作仕様をを策定するが，その時点でシステムの挙動に関する具体的な情報を得ることは難しい．また，ここにおいて求められているのは，具体的な数値ではなく，どのような動作仕様であれば要求を充足するかであるため，このような具体的な数値を獲得することは望まれない.

定性的に比較を行う手法では，優先順位をはじめとした比較情報によって非機能要求の充足を分析する．このような方法では具体的なパラメータは不要で，2 つ (またはそれ以上) のパラメータに対してその優劣が判別できれば一定の分析ができる．例として Parametric Timed Automata のパラメータに対する記号計算 [16] と充足可能性モジュロ理論 (SMT) 解法 [17] を用いた網羅的分析がある．ただし，定性比較は比較情報による抽象的な比較であるため，複雑なシステムになると望んだ解を得られない可能性がある.

離散制御器合成においても非機能要求に関する研究は行われており，[18] では，動作仕様のメイクスパンに着目した最小化手法が提案されている.

1.3 先行研究の課題と本研究の目的

離散制御器合成によって生成される動作仕様は，その内部に不確実性を有している．これはモデル化されたシステム自身が持ち，実際にシステムを動かした際に持ちうるものである．この不確実性はメイクスパンの最小化を図る際に考慮しなければならず，先行研究 [18] では，この不確実性を制御する方法を確立した．また，動作仕様全体を網羅的に制御・分析することで，広く扱うことのできるフレームワークを提案している．一方でこの不確実性制御と網羅的な分析は計算時間の爆発を招いており，指数関数的に爆発する.

本論文では，先行研究で課題として挙げられている計算時間を削減することを目的とする．先行研究の手法はその課題によって適用可能なモデルが小・中規模に限られてしまう．また，複雑なシステムにおいてはモデル規模は必然的に大きくなることが想定されるため，この観点においても適用範囲が大きく狭まる.

この課題に対して，本論文では分析空間を小さくすることで計算時間を削減する方法を提案する．具体的には，先行研究で提案している不確実性制御と網羅的分析の 2 点に対して，分析空間を分割し，同時に不確実性制御を行う空間も縮小させることによって計算空間を削減し，アルゴリズム全体の計算時間減少を狙う．本論文では先行研究のメイクスパン最小化手法をベースに部分的分析を行うための条件を決定し，それに基づいた部分分析アルゴリズムを提示する．

1.4 本論文の構成

本論文の構成は次の通りである．まず 2 章では，本論文の背景技術として離散制御器合成の技術とこれを実現するための要素技術についての説明を行う．3 章では本論文で主に扱うケーススタディについて，その問題設定を説明する．4 章では，本研究の先行研究であるメイクスパン最小化手法 [18] について，重要な技術をそれぞれ説明する．5 章では，背景技術を踏まえコントローラのメイクスパン最小化アルゴリズムの計算時間を削減する方法を提案し，その具体的な技術詳細を提示する．6 章では，提案する手法が先行研究と比べどの程度の有用性があるかを検証し，同時に有効なドメインやモデルについて実験および考察する．7 章で本論文で扱う非機能要求の関連研究や制御に関連する研究について触れ，8 章で本論文の貢献や今後の展望についてまとめる．

第2章　背景技術

2.1　離散事象システム

システムのモデル化を行う方法のひとつに，離散事象システム [10] がある．離散事象システムはイベント (アクション) によって状態が変化するシステムであり，システムと環境との相互作用を離散的に表現することができる．例えば，ロボットの移動というアクションによってロボットが別の場所に移動するようなシステムは，移動前の状態から,「移動する」というアクションによって移動後の状態に変化する．現実世界におけるシステムは多くのものが連続的であるが，制御や検証といったシステムの内部での振る舞いに注目することで離散的に考えることが可能である．また，状態遷移のアクションはシステムの動作だけでなく，環境からの反応や入力などによっても定義されるため，現在ではデータベースや生産システムをはじめとした様々なシステムで用いられている．[10]

離散事象システムのモデル化にはオートマトンやペトリネット (図 2.1) が用いられる．離散事象システムを表現するオートマトンには状態遷移に対してイベントが与えられており，本論文では，これらのイベントをシステムや環境の振る舞いとしてアクションと呼ぶ.

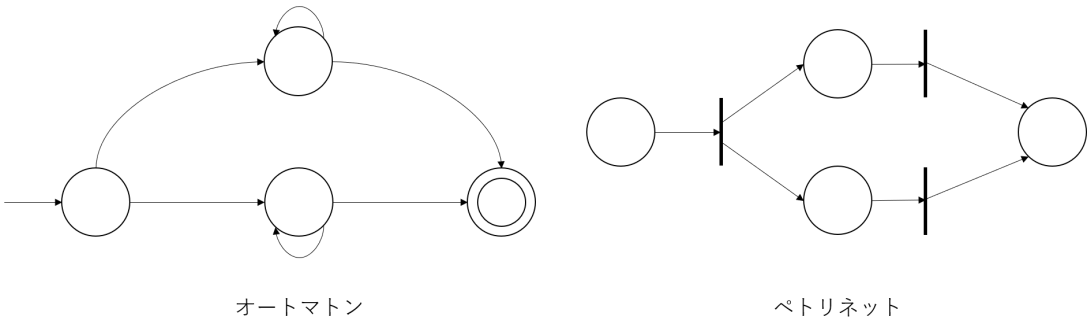

図 2.1: オートマトンとペトリネット

オートマトンはある状態においてアクションが与えられたとき，その遷移先の状態が複数存在する場合がある．このようなオートマトンは非決定論的であるといい，逆に遷移先の状態が最大 1 つである，すなわち遷移先が一意に決定するものを決定論的であるという (図 2.2)．本論文ではこれらのモデルのうち，後者の決定論的モデルに焦点を当てる．

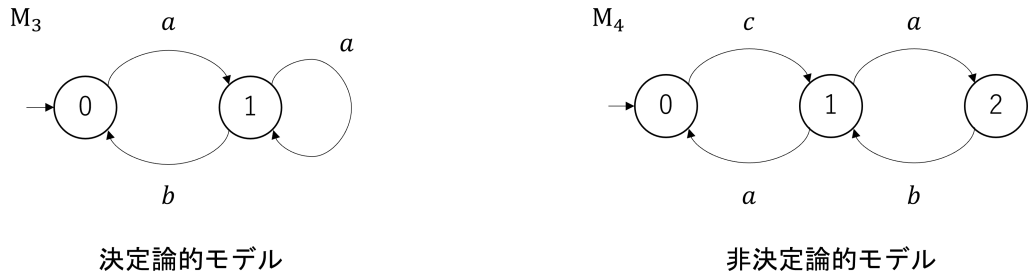

図 2.2: 決定論的モデルと非決定論的モデルの例

2.2　ラベル付き遷移システム

ラベル付き遷移システム (LTS)[11] は，離散事象システムを用いて振る舞いをモデル化するために使われる形式手法の一つである．LTS の各状態はアクションでラベリングされた遷移によって接続され，システムの動作を表現することが可能である．LTS M が与えられたとき，M は定義 2.2.1 のように定義される．遷移前の状態 q と遷移後の状態 q' に対して q から q' に遷移するアクション ℓ は $q \xrightarrow{\ell} q'$ または (q, ℓ, q') と表される．なお，これ以降，与えられた LTS に対してそれらの集合が明示的に定義されていない場合，それらはタプル名を用いて表現される．例として LTS M が与えられた場合，$M = (Q_M, \Sigma_M, \Delta_m, q_{M0})$ のように表される。

定義 2.2.1. (ラベル付き遷移システム)　ラベル付き遷移システム (LTS) はタプル (Q, Σ, Δ, q_0) であり，それぞれの構成要素は次のように与えられる．

- Q：状態の有限集合
- $\Sigma \subseteq Act$：観測されるアクションの集合 Act に対する部分集合
- $\Delta \subseteq (Q, \Sigma, Q)$：遷移の関係
- q_0：オートマトンの初期状態

振る舞いのモデリングに LTS を用いることで，システムの振る舞いを複数の要素に分解して考えることができ，更にこれらの構成要素を表す LTS は並列合成 [11] によってその相互作用をモデル化することができる．並列合成は構成されたモデルの非同期実行をモデル化する LTS として定義され，構成要素間に共有されている遷移を同期させると同時にそれ以外の非共有アクションをインターリーブする．また，2 つの LTS で共有するアクションには同期が求められる．例として，図 2.3 に 2 つの LTS M と N を並列合成した結果を示す．図 2.3 のそれぞれの LTS は，特定のツールを使うアクション $(Tool_A, Tool_B)$ を行い，その後にそれらのツールの使用を終了するようなモデルである $(Finish_A, Finish_B)$．これらは共有アクションを持たないため，並列合成することによって右のような LTS を得る．

定義 2.2.2. (並列合成)　2 つの LTS M と N の並列合成 $\|$ は $M\|N = (Q_M \times Q_N, \Sigma_M \cup \Sigma_N, \Delta_{M\|\|N}, (q_{M_0}, q_{N_0}))$ で表される対象演算子であり，$M\|N$ は以下を満たす最小関係である．

$$\frac{q_M \xrightarrow{\ell}_M q_M'}{(q_M,q_N) \xrightarrow{\ell}_{M\|N} (q_M',q_N)} \ell \in \Sigma_M \setminus \Sigma_N$$

$$\frac{q_N \xrightarrow{\ell}_N q_N'}{(q_M,q_N) \xrightarrow{\ell}_{M\|N} (q_M,q_N')} \ell \in \Sigma_N \setminus \Sigma_M$$

$$\frac{q_M \xrightarrow{\ell}_M q_M', q_N \xrightarrow{\ell}_N q_N'}{(q_M,q_N) \xrightarrow{\ell}_{M\|N} (q_M',q_N')} \ell \in \Sigma_M \cap \Sigma_N$$

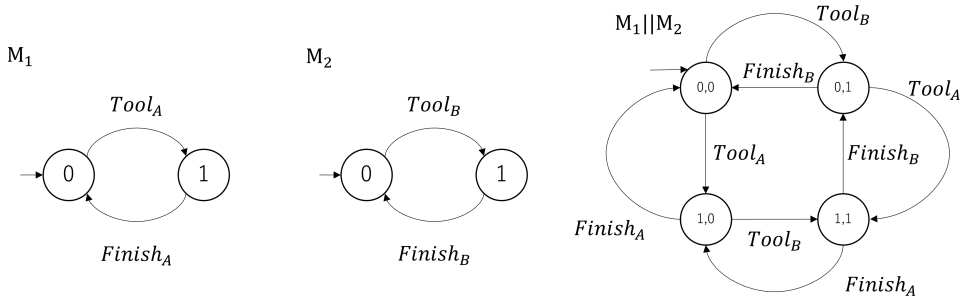

図 2.3: 2 つの LTS M と N の並列合成

LTS では，システムの実行中に起こりうる挙動の記述にはパスとトレースが用いられる．パスは LTS の有効な遷移の順列であり，状態とアクションの無限列で表される．トレースは LTS が実行可能なアクションの順列であり，これは LTS に対応するパスが存在することを意味し π で表記される．例として，図 2.3 の $M\|N$ を用いて考えると，初期状態 $q_{(0,0)}$ から $q_{(1,1)}$ に至るパスは $q_{(0,0)} \xrightarrow{Tool_A} q_{(1,0)} \xrightarrow{Tool_B} q_{(1,1)}$ が挙げられ，これのトレース π は $\pi = Tool_A, Tool_B$ となる．また，本論文においては LTS M において初期状態から状態 q に至ることのできる全てのパスの集合を $Path(M, q)$ と記述する．例えば図 2.3 では $q_0 \xrightarrow{Tool_A} q_1 \in Path(M_1, q_1)$ となる．同様に，LTS M のトレースの無限集合を $Tr(M)$ と記述する．本論文で扱うモデルは決定論的であるから，トレースが与えられると対応するパスは一意に決定する．

定義 2.2.3. (パス)　LTS $M = (Q, \Sigma, \Delta, q_0)$ 上のパス $q_0 \xrightarrow{\ell_0} q_1 \xrightarrow{\ell_1} ...$ は M の遷移の順列である．

定義 2.2.4. (トレース)　LTS $M = (Q, \Sigma, \Delta, q_0)$ 上にパス $q_0 \xrightarrow{\ell_0} q_1 \xrightarrow{\ell_1} ...$ が存在するとき，これのアクションの順列 $\pi = \ell_0, \ell_1, ...$ は M のトレースである．

2.3 Fluent Linear Temposal Logic

システムの動作の正しさに関わる特性を記述するために用いる形式言語のひとつに時相論理 [11] が存在する．時相論理は命題論理を拡張したものであり，システムが持つべき振る舞いを記述することができる．時間のモデル化の方法によって時相論理は線形と分岐の 2 種類に分けられる．これらの違いは可能な未来の唯一性であり，線形時間の始点からは可能な未来はただ一通りである一方で，分岐時間の始点からは複数通り存在する場合があるという点である．これらの特性は，すべての可能な未来で成立する必要のある性質を記述したり，特定の未来のみ成立しなければならない性質を記述する際に用いられる．本論文においては時間論理の中でも線形時間特性を記述するのに用いられている線形時相論理 (Linear Tenporal Logic, LTL)[11] を更に拡張した Fluent Linear Temporal Logic(FLTL)[19] を用いる．

FLTL はプロパティを記述するために使用する言語であり，状態ベースの命題の代わりにフルーエントを推論するための LTL である．フルーエント Fl は集合のペアとブール値で $Fl = \langle I_{Fl}, T_{Fl}, Init_{Fl} \rangle$ と定義される．ここにおいて，$I_{Fl} \subseteq Act$ は開始アクション、$T_{Fl} \subseteq Act$ は終了アクションを意味し，これらは $I_{Fl} \cap T_{Fl} = \emptyset$ を満たす．フルーエントにおける $Init_{Fl}$ は初期において $true$ または $false$ が与えられる．問題において与えられている全ての $\ell \in Act$ はフルーエント $\dot{\ell} = \langle \{\ell\{, Act \setminus \{\ell\}$, $false\rangle$ を生成する．FLTL は標準の LTL と同じ表現力を持つが，フルーエントを用いることで状態ベースの命題をイベントベースのモデルに適用することもでき、よりコンパクトな方法でプロパティを表現することが可能である．

定義 2.3.1. (フルーエント)　観測できるアクションの集合 Act に対するすべての可能なフルーエントの集合を F としたとき，FLTL は標準的なブール接続詞，時間演算子 X(next)，U(strong until) と 2 つの異なる FLTL 式 φ と ψ を用いて以下のように帰納的に定義される．

$$\varphi ::= Fl | \neg\varphi | \varphi \vee \psi | \mathbf{X}\varphi | \varphi \mathbf{U} \psi$$

これらの他に $\wedge$，$\Diamond$ (eventually)，$\Box$ (always)，W(weak until) なども用いる．FLTL 式はトレースとフルーエントに対して計算され，その満足度は標準的なものである．Act 上の無限トレースの集合を Π とする．トレース $\pi = \ell_0, \ell_1, ...$ が次の 2 式を満たすとき，位置 i においてフルーエント Fl を満たす．($\pi, i \models Fl$ と表記する)

- $Init_{Fl} \wedge (\forall j \in N \cdot 0 \leq j \leq i \rightarrow l_i \notin T_{Fl})$
- $\exists j \in N \cdot (j \leq i \wedge l_i \in I_{Fl}) \wedge (\forall k \in N \cdot j < k \leq i \rightarrow l_k \notin T_{Fl})$

したがって，フルーエントが位置 i で条件を満たすのは最初から条件を満たしている場合か，フルーエントの開始アクションが発生しており，かつ終了アクションが発生していない場合である．言い換えれば，この条件はフルーエントの開始アクションと終了アクションまでの間を示していることが分かる．また，トレース π においてフルーエント Fl が常に条件を満たす場合，$\pi \models Fl$ と表す．

次節では，2.2 節で説明した LTS と本節で説明した FLTL から，要求を満たす動作仕様 (コントローラ) を生成する手法について説明する．

2.4 コントローラ合成

コントローラ合成問題とは，形式的な仕様から自動的にその解を生成する問題である．この問題の解はコントローラと呼ばれ，LTS 制御問題はワールドマシンモデルに従ったイベントベース制御問題である [20]．LTS 制御問題では，環境の振る舞いを LTS で記述され，機能目標 (要求)[21] は FLTL 式で表現され，特に要求は本論文において安全性要求と到達性要求の 2 つによって構成される．LTS に記述されるアクションにはシステムが主体となるものとそうでないものに分けることができ，例えばロボットに与えられる入力は環境からロボットへの働きであり，その入力を受け取ったロボットが実行するアクションはロボットから環境への働きかけである．このようなアクションは，制御可能アクションと制御不能アクションの 2 種類に分類される．制御可能アクションはシステムが実行するアクションであり，移動やツールの実行などが挙げられる．もう一方の制御不能アクションは環境からシステムに働きかけるアクションであり，人間がシステムに与える入力や，システムが実行したアクションの結果などが挙げられる．LTS 上のアクションはこれらの 2 種類のアクションのいずれかに属する．環境ではそれぞれアクションの発生するタイミングが記述されており，コントローラは，システムが主体である制御可能アクションをいつ実行するかを決めることができるが，システムが主体でない制御不能アクションについてはその実行タイミングを制御することができない．コントローラ合成問題は、LTS 上のアクションのうち，制御可能なアクションのみを無効にすることで、要求仕様を満たすコントローラを生成する．要求仕様には様々なものがあるが，本論文におけるコントローラ合成問題ではそのうち，監視制御 [22] や不確実性を考慮した計画 [23] で注目される到達可能性と安全性の目標に焦点を当てる。

定義 2.4.1. (制御可能アクションと制御不能アクション)　LTS のアクションの集合 Σ が与えられたとき，Σ は制御可能アクションの集合 $\Sigma_c \subseteq Act$ と制御不能アクションの集合 $\Sigma_u \subseteq Act$ について次の条件を満たす．

・$\Sigma = \Sigma_c \cup \Sigma_u$
・$\Sigma_c \cap \Sigma_u = \emptyset$

また，上記の制御可能アクションと制御不能アクションのそれぞれによる遷移も同様に区別することができ，本論文では状態 q における遷移の有限集合 $\Delta(q) = \{\ell | q \xrightarrow{\ell} q'\}$ に対して以下のように記述する．

・$\Delta^c(q) = \Delta(q) \cap \Sigma_c$：制御可能アクションによる遷移の有限集合
・$\Delta^u(q) = \Delta(q) \cap \Sigma_u$：制御不能アクションによる遷移の有限集合

コントローラは過去に発生したアクションに基づいて制御可能なアクションの発生を制限する LTS としてモデル化され，このコントローラ C は環境 E と同時に実行される．C および E はともに LTS であるから，この実行は C と E の並列合成 $(E\|C)$ によってモデル化される．ただし，この C は定義 2.4.2 に基づいて生成される．まず，コントローラはシステムの要求を満たす動作仕様であるから，環境との同時実行の全てのトレースにおいて G を満たす必要がある．また，コントローラが制御できるのは制御可能アクションのみであるから，制御不能アクションをブロックしてはならない．この，制御不能なアクションをブロックしないコントローラの概念はインターフェイスオートマタ [24] の法的環境の概念に基づく．最後に，$E\|C$ はデッドロックフリーである必要がある．

定義 2.4.2. (LTS 制御問題)　LTS $E = (Q_E, \Sigma, \Delta_E, q_{E_0})$，FLTL で記述されたゴール，制御可能アクションの集合 $\Sigma_c \sqsubseteq \Sigma$ が与えられたとき，$\epsilon = \langle E, G, \Sigma_c \rangle$ は以下の 3 つを満たす決定論的な LTS $C = (Q_C, \Sigma, \Delta_C, q_{C_0})$ を探す制御問題である．

1．LTS C は E に対して合法的な LTS である
2．$E\|C$ はデッドロックフリーである
3．$Tr(E\|C)$ の全ての無限トレース π は G を満たす (すなわち $\pi \models G$)

定義 2.4.3. (合法的な LTS)　LTS $E = (Q_E, \Sigma, \Delta_E, q_{E_0})$，LTS $C = (Q_C, \Sigma, \Delta_C, q_{C_0})$，制御可能アクションの集合 $\Sigma_u \sqsubseteq \Sigma$ が与えられたとき，$E\|C$ の状態 $Q_{E\|C}$ について全ての $(q_E, q_C) \in Q_{E\|C}$ で $\Delta_{E\|C}((q_E, q_C)) \cap \Sigma_u = \Delta_E(q_E) \cap \Sigma_u$ を保証するならば C は E に対して合法的な LTS である．

第3章　ケーススタディ：産業オートメーション

本章では，本論文で主として扱うドメインの説明と，それらのモデル化や要求の記述，コントローラ合成について示す．

想定するモデルはある産業オートメーションにおいて，異なる種類の製品が事前に決定された製造手順に従って製造するものである．製品は 1 種類ないし 2 種類で与えられ，それぞれの製品に対して工場にあるツールを用いて加工・製造を行う．モデルの一例を図 3.1 に示す．図 3.1 では，製品は 2 種類で，製造機械には加工用の 6 つのツールと，加工した製品の品質を確認するための品質管理ツールと，故障がある製品に対して修理を施す修繕ツールが備えられている．また，この機械に対して図 3.2 のような製造手順と機能目標が与えられている．

機械の動作手順としてはまず，与えられた入力番号に応じた製品の加工・製造を開始する．加工および製造が終了したら，完成した製品に対して品質管理ツールを使用する．品質基準を満たない製品に対して修繕ツールを用いて修理を行い，製品を出力する．製造時の制約としては以下の 4 つが与えられており，a) 製品番号 1 に対してはツール 3 とツール 5 またはツール 6 を順に実行する．b) 製品番号が 2 の場合は，ツール 1 かツール 2 のどちらかを用いて加工を行い，その後ツール 4 を使用して加工を完了する．c) 機械が一度に処理する製品はたかだか 1 つであり，複数の製品を同時・平行に加工することはない．d) 機械は適切な場面以外において各種ツールを使用しない．

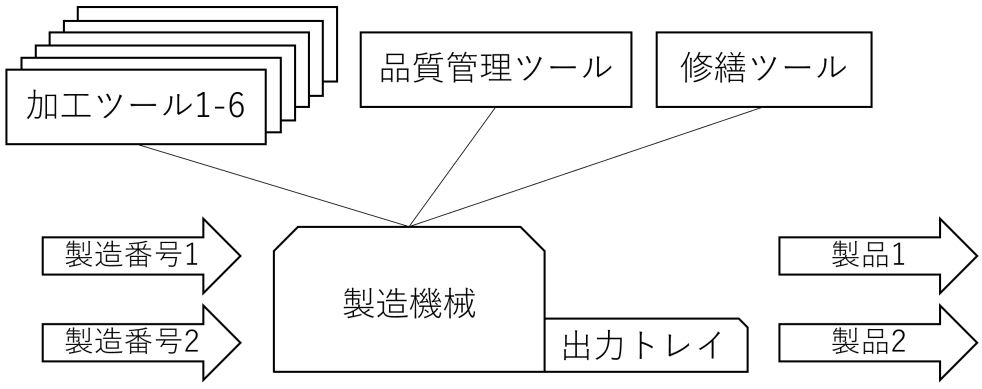

図 3.1: 産業オートメーションの機械イメージ図

製造手順

1. 製品番号が入力されたら，その番号の製造手順に従って製品の加工・製造を行う。
2. 加工・製造が終了後，品質検査を行い，基準を満たしていないものは再加工(修理)により基準値を満たすよう製造する.

製造における制約

1. 製造番号が1のとき，ツール3，ツール5または6の順に加工する.
2. 製造番号が2のとき，ツール1またはツール2による加工の後，ツール4によって加工を行う.
3. 一度に処理する製品は1つのみである.
4. 製造加工時以外においてツールを稼働させてはならない.

図 3.2: 製造における手順と制約

図 3.2 に沿ったシステムのフローを図示すると図 3.3 のようになる．各製品の加工ではそれぞれに使用するツールに選択肢が存在しており，どちらの加工ツールを用いても製品は加工することができる．一方，加工ツールにはそれぞれ特徴があることが想定され，例えばツール 1 よりもツール 2 の方が加工速度が速いが，ツール 1 の方がより高品質の製品を製造できるといった場合も考えることができる．本論文では，このようなシステムの動作において制約を満たす選択肢を複数持つ場合に，それぞれの選択肢が持つ特徴に着目した制御問題の研究を扱う.

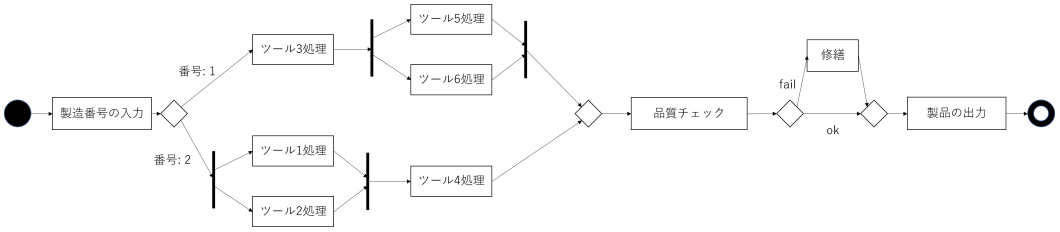

図 3.3: 産業オートメーションの流れ

制御問題の仕様は，環境の振る舞いを記述した LTS の集合 (図 3.4) と動作の制約を記述した FLTL 式の集合 (図 3.6) として与えられる．図 3.4 で与えられるように，振る舞いは複数の要素として分割して表現され，それらすべての並列合成によって表される．また，図 3.4 において制御可能アクションは実線，制御不能アクションは点線で表される．例えば $Process_i$ モデルはそれぞれのツールによる加工処理 $i \in \{1...6\}$ を行うモデルであり，プロセス開始を表す制御可能アクション s_{Ai} とプロセス終了を表す制御不能アクション e_{Ai} で与えられる．これはシステム側が任意のタイミングでプロセスを開始し，その結果として不定のタイミングで終了

アクションが発生することを表している．また，選択モデル $choice$ は製造部品の選択を決定する制御不能アクション $type_1, type_2$ とそれらの製造終了を表す制御可能アクション $done$ によって表現される．これは製造する部品の種類をシステム側で選択できない (環境からの入力に依存する) ことを意味している．同様に品質管理モデル QA，修理モデル fix もそれぞれ制御可能アクションと制御不能アクションによって表現されている．

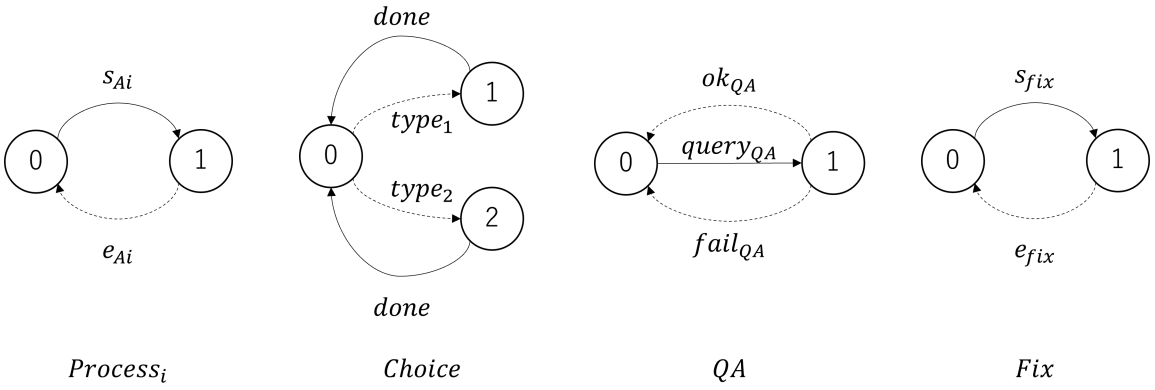

図 3.4: 環境の振る舞いを表した LTS

これらの環境モデルに対しフルーエントを図 3.5 のように定義する．これらの定義は FLTL で記述される制約に用いられ，実際に定義されたフルーエントを制約として記述したものが図 3.6 である．制約記述は左辺に対して右辺が必要条件であり，例えば $\ell \Rightarrow Fl$ はフルーエント Fl が真のときのみ ℓ が有効になることを意味する．

1. $TC_i = \langle \{type_i\}, \{done\}, false \rangle$
2. $AC_i = \langle \{e_{Ai}\}, \{done\}, false \rangle$
3. $QAchecked = \langle \{ok_{QA}, e_{fix}\}, \{type_1, type_2\}, false \rangle$
4. $QAfailed = \langle \{fail_{QA}\}, \{e_{fix}\}, false \rangle$
5. $Ongoing_i = \langle \{s_{Ai}\}, \{e_{Ai}\}, false \rangle$
6. $Inputchosen = \langle \{type_1, type_2\}, \{done\}, false \rangle$

図 3.5: 図 3.4 のモデルに定義したフルーエント

1. $s_{A4} \Rightarrow (AC_2 \vee AC_2)$
2. $s_{A5} \vee s_{A6} \Rightarrow AC_3$
3. $queryQA \Rightarrow \Big(\big(TC_1 \wedge AC_3 \wedge (AC_5 \vee AC_6)\big) \vee (TC_2 \wedge (AC_1 \vee AC_2) \wedge AC_4)\Big)$
4. $s_{fix} \Rightarrow QAfailed$
5. $done \Rightarrow QAchecked$
6. $AC_i(i = 1 \ldots 6) \Rightarrow Inputchosen$

図 3.6: 図 3.5 に基づいた制約記述

ここで記述したフルーエントや制約は図 3.2 に対応しており，例えば図 3.6 の 1 は，ツール 4 の使用開始 (S_{A4} はツール A またはツール 2 の加工の完了後であることを意味する．それぞれの TC_i, AC_i は各製造タイプの決定アクション $type_i$ および各プロセスの終了アクション e_{Ai} を開始アクション，製造終了アクション $done$ を終了アクションとしたフルーエントである．つまり，それぞれの製品番号が入力され，それに対応した製造プロセスが終了していてかつ製造自体が終了していない状態 $((TC_1 \wedge AC_3 \wedge (AC_5 \vee AC_6))$ または $(TC_2 \wedge (AC_1 \vee AC_2) \wedge AC_4))$ であることがアクション $query_{QA}$ の発生条件である，ということである．このように，フルーエントにおいて開始アクションと終了アクションを定義することで，指定したアクションとアクションの間において，特定の動作を行う (または行わない) といった制約を指定することができる．

これらのモデルおよび制約によって生成される合成問題は，製造の種類と品質チェックの結果が事前にわからないような環境において，システムの動作を制御して製造要求を満たすコントローラーを生成することである．この問題にはいくつかの解が考えられるが，ここで考えられる有効なコントローラは以下の 4 つである．

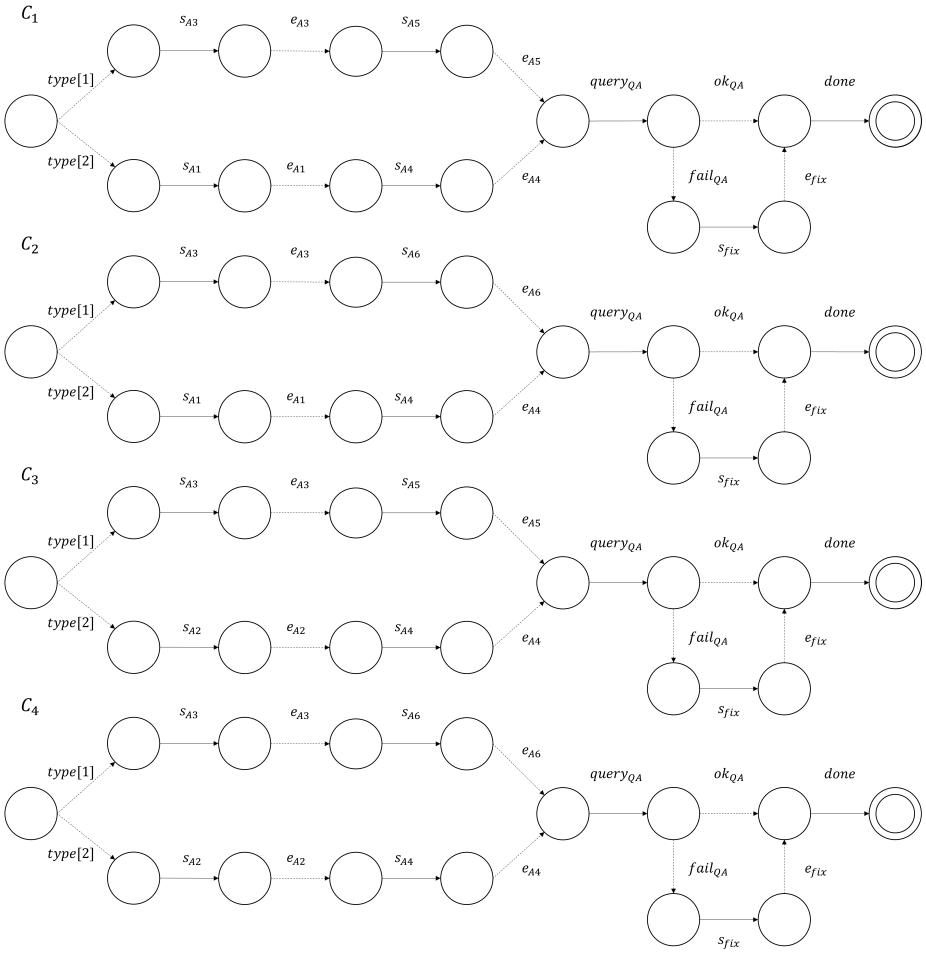

図 3.7: 有効なコントローラ

また，図 3.7 のように複数生成される解を全て包含するユニバーサルコントローラ [25] といい，図 3.8 のように表現される．ここで，コントローラはゴール状態からの遷移を取り除いたものであり，これは有向非巡回グラフ (DAG) の形を取る．本研究ではサイクルを持たないグラフを仮定しているため，この前提は重要である．

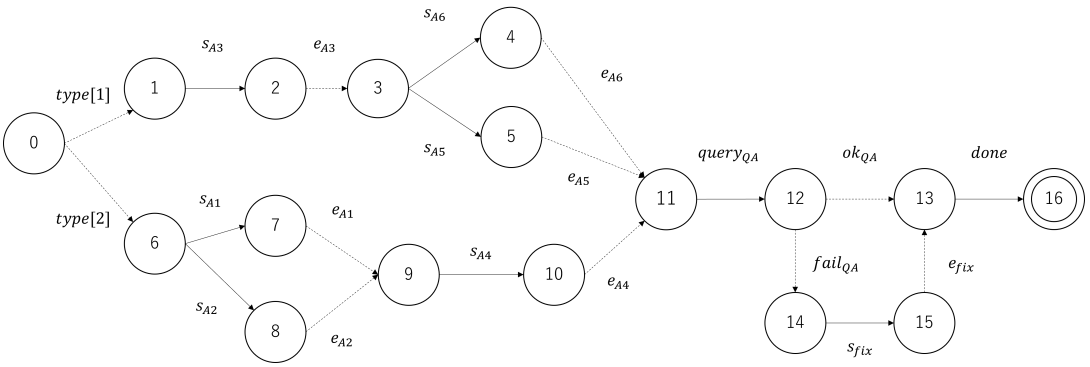

図 3.8: 4 つのコントローラを包含するユニバーサルコントローラ

第4章　コントローラ合成時のメイクスパン最小化手法

標準的なコントローラ合成では，FLTL で記述された機能目標を達成する LTS を生成することができる．機能目標は安全性や到達性に関するものであり，これらの要求は機能要求に属する．一方で，3 章で説明したように，機能要求を満たす複数のアクションがシステムが動作するある状況において選択肢として与えられているような場合がある．このような場合に，システムは複数ある選択肢のうちの 1 つを選択 (または発生を待つ) することで目標を達成するが，これらの選択肢にはそれぞれ特徴を持っている可能性があり，それは例えばコストや品質といった非機能要求に該当する要素である．これらの特徴が既知である場合，可能な限りより優れた特徴を備えた選択肢を選ぶことで，システムはより良い性能を発揮する．これに関する研究の一つに Ezequiel らによるメイクスパン最小化フレームワーク [18] がある．メイクスパンとは，システムが実行するアクションの実行時間を意味し，先行研究では，システムの稼働開始 (すなわち初期状態) から稼働終了 (終了状態) までのシステムの総実行時間をエンドツーエンドメイクスパンと定義している．ここで，このメイクスパンはあくまでシステムが実際に稼働するときの時間を意味していることに注意する．ユニバーサルコントローラに属する 2 つの異なるコントローラについて，エンドツーエンドメイクスパンを比較することによって劣性な一部の遷移を消去することでメイクスパンの観点でより優れたユニバーサルコントローラを生成することができる．図 4.1 は，メイクスパン最小化フレームワークの全体図である．

図 4.1: メイクスパン最小化フレームワーク全体図

4.1　コントローラのメイクスパン比較における考慮事項

3 章で生成されたコントローラは特定の状態で使用するツールに選択肢が与えられている．このような場合に，どちらのツールを選択することがメイクスパンの観点で有利かを考える必要がある．ここで考えなければならない要素として，制御不能アクションによる不確実性が挙げられる．制御不能アクションを持たない一般的な決定論的なオートマトンでは，計画法に基づいて各辺に設定されたパラメータが最小 (または最大) となるように辺を選択することで目標を達成することができる．しかしながらコントローラは制御不能アクションを持っており，この遷移はシステムが意図的に決定することができない．例えば，図 3.8 の q_{12} では，制御不能アクションによる結果に応じてその後の動作が異なっている．このような場合に，システムがどちらに遷移するか不明にも関わらず恣意的に優劣を決定することは非合理的であると言える．これはより一般的な場合にも言え，選択肢の先で制御不能アクションによって動作が変わるような場合に，その結果を恣意的に決定することで選択肢の優劣が決まるようなことがあってはならない．したがってメイクスパンを比較するにあたって制御不能アクションの発生に一貫性を持たせることが重要である．具体的に述べるのであれば，q_{12} の結果が異なることでそれ以前のツール毎のエンドツーエンドメイクスパンに影響を与えることによる比較結果の妥当性を脅かさないように可能な限り同じ結果を選択するということである．これを定義する方法は 4.3 節で説明する．

また，LTS における時間概念について考える必要がある．コントローラはアクションによって状態が遷移するモデルであり，特定のプロセスの実行中もまた状態で表現される．直感的には，場所 A と場所 B 間を繋ぐ辺に必要時間が設けられているようなグラフではなく，場所 A において「B に移動」というアクションに対して移動中を表現する状態があり,「B に到着」という結果を経て場所 B に到達する．ここにおいて時間の概念は移動中を意味する中継状態でのみ経過し，それぞれのアクションは瞬間的に発生していると理解できる．このように，LTS で表現されるシステムの制御器では時間は状態の上でしか経過しない．また，この特性上，時間経過をひとつのアクションでは表現できない場合がある．分かりやすい例が先述の移動であり，この例では「B に移動する」という一連のアクションは「B に移動」と「B に到着」の 2 つから成ることが直感的に理解できる．また，例えば B への移動中に別の作業を行っていた場合，$q_0 \xrightarrow{B\text{ に移動}} q_1 \xrightarrow{\text{別の作業開始}} q_2 \xrightarrow{\text{作業終了}} q_3 \xrightarrow{B\text{ に到着}} q_4$ というパスになるが，このとき直感的なメイクスパンは B への移動時間+作業時間になるのに対し，単純に「B に移動」というアクションと「B に到着」というアクションの間の時間経過によって B の移動時間を計測するのは不適切である．したがって，LTS 上の時間経過の概念について，新規の定義付けが必要である．Ezequiel らは，これをアクティビティと定義している．4.2 節にて述べる．

4.2　アクティビティ定義によるメイクスパン計測

システムの動作の中には，あるプロセスの始まりと終わりがアクションによって定義されているものがある．例えば，ツールの使用の開始や終了，品質チェックにおけるチェック開始とその結果出力がそれである．これらのアクション群はセットで考えられ，振る舞いの定義においても，一方のアクションが発生した後にもう一方のアクションが発生することによって元に戻るといった記述をされる．このアクションのセットを，先行研究 [18] ではアクティビティと定義している．アクティビティはその名称と，アクティビティの開始を表す 1 つの制御可能アクション，アクティビティの終了を表す 1 つ以上の終了アクションから成る．3 章で示したモデルでは，アクティビティ A_1 について，開始アクション s_{A_1} と終了アクション eA_1 で与えられる．また，QA のように，終了アクションが ok_{QA} と $fail_{QA}$ と複数存在する場合がある．通常，システムの動作の結果は成功と失敗で与えられ，失敗した場合には再度試みるなどが考えられるが，本論文で扱うモデルではサイクルを有さないという特徴から，システムの働きかけは成功するものとして定義している．QA における $fail_{QA}$ は，品質検査において一定の基準を満たさなかったという意味であることに注意する．これらの関数は各アクティビティ $\alpha \in A$ に対し定義される必要があり，観測される各アクションは最大でも 1 つのアクティビティに関係している．

定義 4.2.1. (アクティビティ定義)　LTS $M = (Q, \Sigma, \Delta, q_0)$ が与えられたとき，アクティビティ定義 $AD = (A, Start, Ends)$ は以下のように与えられる

- A：アクティビティを表現する記号の集合
- $Start$：$A \to \Sigma_c$
- $Ends$：$A \to 2^{\Sigma_u}$

アクティビティの開始アクションが発生すると，遷移先の状態では終了アクションが有効になる．アクティビティの開始アクションが発生してから終了アクションが発生するまでの間アクティビティは実行中の状態であり，状態 q における実行中のアクティビティの集合を $\zeta(q)$ と定義する．メイクスパンの観点では，アクティビティが実行中の間の経過時間を Parametric Timed Automata(PTA)[16] によって計算しており，開始したアクションは必ず終了する．したがって，コントローラのゴール状態おいて実行中のアクティビティは存在しないことに注意する．これらの考え方は，次節で説明するスケジューラや，本論文の提案手法において重要である．

定義 4.2.2. (実行中アクティビティ)　LTS $M = (Q, \Sigma, \Delta, q_0)$ と $\ell_k = Start(\alpha)$ であるアクティビティ α，パス $q_0 \xrightarrow{\ell_1} \ldots \xrightarrow{\ell_k} \ldots \xrightarrow{\ell_i} \ldots \xrightarrow{\ell_m} q_m$ が与えられたとき，$\forall i(k < i \leq m \wedge \ell_i \neq Ends(\alpha))$ であるならばアクティビティ α は q_m で実行中 ($\alpha \in$

$\zeta(q_m))$ であるという．M のゴール状態の集合を Q_G とすると，$\forall q \in Q_G(\zeta(q) = \emptyset)$ である．

4.3 スケジューラ

コントローラにおける制御不能アクションによる不確実性は，メイクスパンの計測と比較の結果に不合理をもたらす．これを解決する方法として，先行研究ではスケジューラ [26] を適用し，制御の一貫性をもたせることで合理的な比較を実現している．スケジューラはマルコフ型 [26] であり，システムの現在までの振る舞いに関係なく現在の状態のみに依存して決定することとし，任意のコントローラによって有効にされる可能性のあるアクションの集合 A を選択する関数 $\sigma : Q_E \times 2^{\Sigma} \to 2^{\Sigma}$ として定義する．

定義 4.3.1. (スケジューラ)　LTS $M = (Q, \Sigma, \Delta, q_0)$ が与えられたとき，スケジューラ $\sigma : Q_E \times 2^{\Sigma} \to 2^{\Sigma}$ はシステムの現在の状態のみに依存して次の条件を基にアクションを選択する．

1．$\sigma(q_E, A) \subseteq A \cap \Delta_E(q_E)$
2．$(|\sigma(q_E, A)| = 1 \wedge \sigma(q_E, A) \subseteq \Sigma_c \cup (\Sigma_u \setminus \Sigma_e)) \vee$
　$(\sigma(q_E, A) \cap \Sigma_e = \sigma(q_E, A))$
3．$(\sigma(q_E, A) \subseteq \Sigma_u) \Rightarrow \forall A'(\sigma(q_E, A') = \sigma(q_E, A))$
4．$\sigma(q_E, A) \subseteq \Sigma_c \Rightarrow \forall A' \subseteq \Delta_E(q_E)$
　$(\sigma(q_E, A) \subseteq A' \Rightarrow \sigma(q_E, A') = \sigma(q_E, A)) \wedge$
　$(\sigma(q_E, A) \not\subseteq A' \wedge A' \cap \Sigma_c \neq \emptyset \Rightarrow \sigma(q_E, A') \subseteq \Sigma_c) \wedge$
　$(\sigma(q_E, A) \not\subseteq A' \wedge A' \cap \Sigma_c = \emptyset \Rightarrow \sigma(q_E, A') \subseteq \Sigma_u)$

コントローラ上には制御可能アクションと制御不能アクションの 2 種類が存在するが，スケジューラにおいてはこれらに加えてアクティビティの終了アクションが分類として加わる．終了アクションは本論文において Σ_e と記述する．3 章のモデルでは，e_{A_i} や e_{Fix} などが Σ_e に属し，終了アクション以外の制御不能アクションには $type_i$ が含まれる．スケジューラは 2 つのコントローラのメイクスパンの比較にあたってシナリオの一貫性を持たせるために，一定の前提に従って状態におけるアクションを選択する．まず，スケジューラが選択するアクションはその状態 q において有効なアクションから選択される (1)．スケジューラが選択するアクションはアクションが制御可能アクションかアクティビティの終了アクション以外の制御不能アクションからただ 1 つであるか，アクティビティの終了アクションの集合で与えられる (2)．一方のコントローラが q で制御不能アクションを選択したとき，もう一方のコントローラは q で同じアクションを選択する (3)．一方のコントローラが q で制御可能アクションを選択した場合，そのアクションがもう一方のコントローラにおいて q で有効であればそれを選択し，そうでない場合に，もう一方のコントローラで制御可能アクションを選択可能であればその中

から任意に選択する．仮にもう一方が制御可能アクションすら選択できない場合には，制御不能アクションから選択する (4)．つまり，スケジューラは一方のコントローラに基本的に近しい挙動を示すようもう一方のコントローラのアクションを選択する．3 のモデルで考えると，比較するコントローラ A が q_{12} で ok_{QA} を選択した場合，条件 3 に基づいてもう一方のコントローラ B でも q_{12} で ok_{QA} を選択するということである．これによって，制御不能アクションによる不確実性を排除していることがわかる．

スケジューラによるアクションの選択はスケジューラ合成 $E\|_\sigma C$ によってコントローラに適用される．

定義 4.3.2. (スケジューラ合成)　LTS $E = (Q_E, \Sigma, \Delta_E, q_{E_0})$, LTS $C = (Q_C, \Sigma, \Delta_C, q_{C_0})$, スケジューラ σ が与えられたとき，スケジューラ合成 $E\|_{\sigma_C}$ は $\Delta_{E\|_\sigma C}$ を以下のように変更した並列合成で定義される非対象演算子である．

$$\frac{q_E \xrightarrow{\ell}_E q_E', q_C \xrightarrow{\ell}_C q_C', \ell \in \sigma(q_E, \Delta_C(q_C))}{(q_E, q_C) \xrightarrow{\ell}_{E||_\sigma C} (q_E', q_C')} \ell \in \Sigma_E \cap \Sigma_C$$

3 のモデルで q_0 において $type_2$ を選択するようなスケジューラを考えてみる．スケジューラは各状態において有効なアクションから選択するから，スケジューラ σ_1 は以下のものが考えられる．

- $\sigma_1(q_0, \{type_1, type_2\}) = \{type_2\}$
- $\sigma_1(q_6, \{s_{A_1}, s_{A_2}\}) = \{s_{A_2}\}$
- $\sigma_1(q_7, \{e_{A_1}\}) = \{e_{A_1}\}$
- $\sigma_1(q_8, \{e_{A_2}\}) = \{e_{A_2}\}$
- $\sigma_1(q_9, \{s_{A_4}\}) = \{s_{A_4}\}$
- $\sigma_1(q_10, \{e_{A_4}\}) = \{e_{A_4}\}$
- $\sigma_1(q_11, \{query_{QA}\}) = \{query_{QA}\}$
- $\sigma_1(q_12, \{ok_{QA}, fail_{QA}\}) = \{ok_{QA}\}$
- $\sigma_1(q_13, \{done\}) = \{done\}$

ここで注意することとして，q_6 において有効なアクションによっては選択するアクションが変化するということである．例えば，$\sigma_2(q_6, \{s_{A_1}\}) = \{s_{A_1}\}$ となる．

3.8 に対して，σ_1 と，q_6 以外が一致する σ_2 を適用すると，図 4.2 のようなコントローラを得ることができる．これらのコントローラは q_{12} における不確実性が排除され，シナリオにある程度の一貫性を持ちながらメイクスパンの比較を行うことができることがわかる．

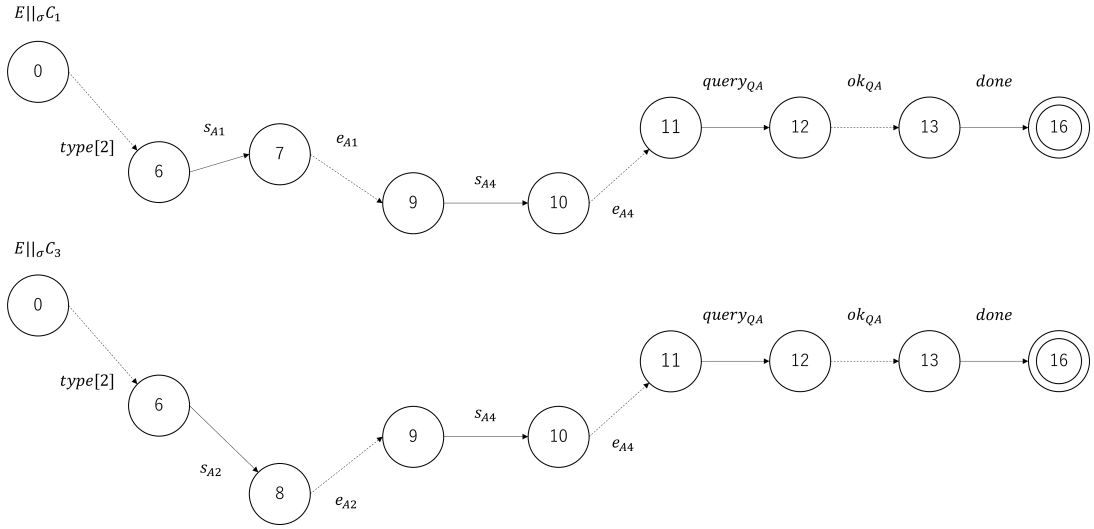

図 4.2: スケジューラを適用したコントローラの例

4.4 メイクスパン最小化コントローラの生成

スケジューラを適用して生成された 2 つのコントローラを生成できたら，これらのメイクスパンを比較してどちらが優れたコントローラかを判断する．この比較は PTA[16] を用いて時間パラメータを実装し，これに対して SMT ソルバーとして Z3[27] を用いることによって実現している．一方のコントローラを C，もう一方を C' としたときに，それらのエンドツーエンドのメイクスパンを計測し，その数値を比較する．このとき，メイクスパンの総計は各アクションやアクティビティのメイクスパンのパラメータの和として表現される．例として，図 4.2 の C_1 のエンドツーエンドメイクスパン T_{C_1} は，各アクティビティのメイクスパン p_α とアクションのメイクスパン p_A を用いて $T_{C_1} = p_{type_2} + p_{A_2} + p_{A_4} + p_{query_{QA}} + p_{done}$ と表すことができる．コントローラ同士のメイクスパンを比較したとき，C が C' に対してメイクスパンの観点で優れているとき，これらの関係を C が C' を支配していると言い，$C \triangleleft C'$ と表記する．C と C' の間の関係は 4 通りあり，一方がもう一方よりも明確に優れているとき，一方がもう一方を支配する．2 つを比較した結果両者がもう一方を支配しているとき，両者の関係は比較不可能であると言う．また，どちらも支配的でないとき，両者は等価であると言う．

メイクスパンを比較するためには，それらを比較するための情報が必要である．例えば，定量的なパラメータを用いれば具体的な数値での支配関係の比較が可能である．Ezequiel らは，定性的な情報を用いてこれらの支配関係を比較している．具体的には，アクティビティやアクションのメイクスパンに対して比較情報を入力として与える．図 4.2 においてアクティビティのメイクスパンの情報として，$A_1 < A_2$ といったものを与える．これは，メイクスパンの観点で A_1 が A_2 よりも小さい，す

なわち A_1 の方が優れていることを意味する．これによってメイクスパンの比較を行うと，入力情報を基にどちらのメイクスパンがより優れている (小さい) かを判定することができる．コントローラ同士を比較する機能は標準的なコントローラ合成には備わっておらず，これの実装にあたっては SMT ソルバーを介して行う．Ezequiel らの論文では Z3 SMT ソルバー [28] を用いており，パラメータと比較情報を入力としてコントローラ間の比較結果を返す．比較によって被支配的なコントローラは削除され，より優れたコントローラのみが残される．図 4.2 において $A_1 < A_2$ が与えられた場合は，A_2 の遷移を持つ C_2 が削除される．

表 4.1: 2 つのコントローラの関係

$C_2 \lhd C_1$	$C_1 \lhd C_2$	Result
Sat	Unsat	C_2 は C_1 を支配する
Unsat	Sat	C_1 は C_2 を支配する
Sat	Sat	比較不可能
Unsat	Unsat	等価

改めて，図3.8でこのメイクスパンの比較を考える．図3.8において，$Processing_i$ と Fix をアクティビティとして定義し，これらのアクティビティの比較情報として $A_1 > A_2, A_5 > A_6$ を入力として与える．これによって s_{A1}, s_{A5} の遷移が削除されたコントローラ (図 4.3) が生成される．

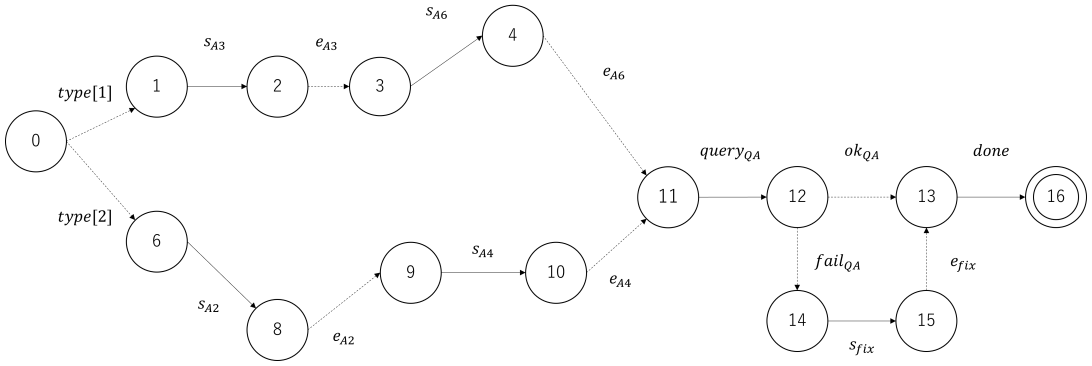

図 4.3: コントローラ比較により生成されるコントローラ

図 4.3 のような支配されていないコントローラを非支配コントローラと呼ぶ．非支配コントローラを生成するアルゴリズムはアルゴリズムの通りである．

アルゴリズム 1 では，まず最初にユニバーサルコントローラを生成する．次に，ユニバーサルコントローラをトポロジカルソートする．これは，コントローラが有効非巡回グラフであることによる．複数のアクションが有効で，制御可能なアクションを含むコントローラ上の状態において，アルゴリズムは各アクションの優劣比較を用いて，制御可能アクションのいくつかを無効にする．ここで，サブ

Algorithm 1 Non-dominated controller algorithm

Input: $E, \mathcal{AD}, G, \Sigma_c$

$\mathcal{U} =$ UNIVERSAL_SYNTHESIS(E, Σ_c, G)
$K = E||\mathcal{U}$
$[q_0, ..., q_n] =$ TOPOLOGICAL_SORT(K, Q_G)
for $q \in sorted([q_n, ..., q_0])$ **do**
  $Alt = \{\{c\}|c \in \Delta_K^c(q)\}$
  if $\Delta_K^u(q) \neq \emptyset$ **then**
    $Alt = Alt \cup \{\emptyset\}$
  if $|Alt| > 1$ **then**
    $\mathcal{D} = \emptyset$
    for $A \in Alt \wedge A' \in Alt \setminus \{A\}$ **do**
      if $K_{(q,A')}$ dominates $K_{(q,A)}$ **then**
        $\mathcal{D} = \mathcal{D} \cup A$
      $\Delta_K = \Delta_K \setminus \{(q, c, q')|(q, c, q') \in \Delta_K \wedge c \in \mathcal{D}\}$
return K

LTS $M_{\langle q,A\rangle} = (Q, \Sigma, \Delta', q)$ が定義され，優劣比較が行われる．サブ LTS$M_{\langle q,A\rangle}$ は初期状態が任意であり，$\Delta' = \Delta \setminus \{(q, \ell, q')|(q, \ell, q') \in \Delta^c(q) \wedge \ell \notin A\}$ である．ここで，A は q における制御可能アクションのいずれかが有効である集合か，制御不能アクションが q で有効である場合に空集合である．このような $M_{\langle q,A\rangle}$ について，M が選択しなかったアクションの集合 Alt を持つ $M_{\langle q,A'\rangle}(A' \in Alt \setminus \{A\})$ と比較する．

第5章　コントローラの分割と段階的比較手法

本章では，本研究で提案する定性的なメイクスパン比較手法を部分的に適用し，計算空間を削減することで実行時間を削減する手法について説明する．

まず，コントローラのメイクスパンを部分的に計測・比較するとはどういうことかを，5.1 節にて先行研究の限界を説明すると同時に述べる．その後，提案する手法について，要素技術毎に説明する．提案手法のアルゴリズム全体図を図 5.1 に示す．図 4.1 と比較すると，非支配コントローラアルゴリズムの前後に新規に 2 つのアルゴリズムが追加されていることがわかる．提案する手法は主に 3 つのフェーズから構成され，それぞれで 1 つのアルゴリズムを適用する．具体的には剪定フェーズ，部分比較フェーズ，同期フェーズから成る．剪定フェーズでは，一般的な離散制御器合成によって獲得できるユニバーサルコントローラにから部分的なコントローラを分割アルゴリズムによって取り出す．このフェーズには更に 2 つの詳細な手順があり，部分的なコントローラの開始状態と終了状態をそれぞれ特定・剪定する．5.3.1 節では，剪定フェーズのうち，部分的なコントローラの初期状態となりうる状態を特定する方法を説明し，5.3.2 節では終了状態を特定する方法について述べる．また，ユニバーサルコントローラから取り出す部分的なコントローラを 5.2 節で定義する．次に行う部分比較フェーズでは，取り出された部分的なコントローラに対して先行研究の手法である非支配コントローラアルゴリズムを実行する．これは Ezequiel ら [18] が提案したメイクスパン最小化手法そのものであり，比較部分において変更を加えていない．最後の同期フェーズでは，部分比較によって得られた結果をユニバーサルコントローラに同期させる．部分比較フェーズで削除される遷移をユニバーサルコントローラで削除することによって，部分的な比較結果をユニバーサルコントローラに同期させる．同期フェーズの詳細は，5.4 節で説明する．

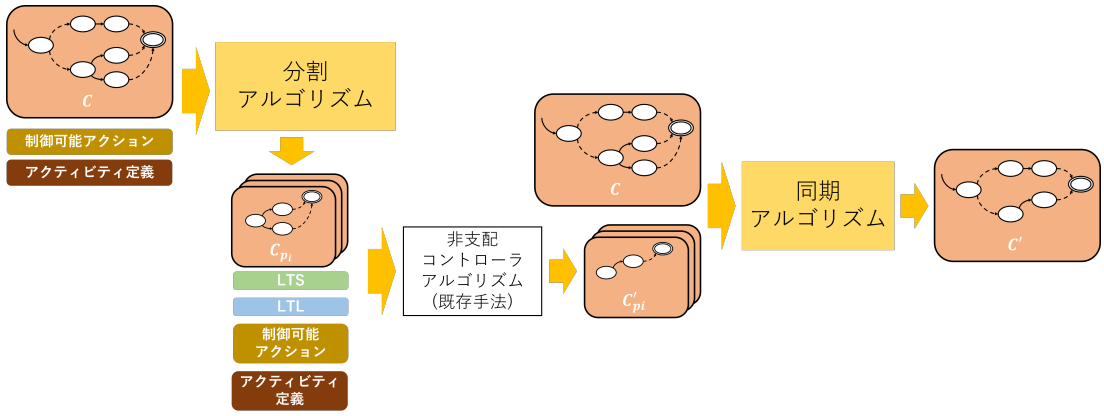

図 5.1: 提案手法の全体図

5.1　プロセスの部分的な比較

[18] が提案したメイクスパン最小化手法ではユニバーサルコントローラから 2 つのコントローラを選択し，その初期状態からゴール状態までの全プロセスについてメイクスパンを計測して比較することによって非支配コントローラを生成している．スケジューラの定義から，ここで比較するコントローラ同士には共通するプロセスが存在する．例えば，4.3 節によれば，共有するある状態においてコントローラ A の選択アクションが制御不能アクションだった場合はコントローラ B は必ず同じアクションを取る．また，$K4$ から，コントローラ同士は可能な限り同じ行動を選択する．つまり，2 つのコントローラの選択した行動が異なるのは，コントローラ A がある状態 q において制御可能アクションを選択し，コントローラ B が同じ状態 q においてその他の制御可能なアクションまたは制御不能アクションを選択していた場合である．このようなとき，それぞれのアクションを選択したコントローラ同士はその先に異なるパスを辿ってゴール状態に到達するため，メイクスパンを計測して比較することによって，どちらのアクションを選択することがメイクスパンの観点において合理的かを判断することができる．逆に，それ以外の場合は 2 つのコントローラは同じパスを辿るから，メイクスパンを比較することはない．したがって，コントローラにおいてプロセスを比較するのは，コントローラ A が制御可能アクションを選択し，コントローラ B がそれ以外の制御可能アクションまたは制御不能アクションを選択する状態 q と，その選択によってパスに違いが生じている部分のみを比較すれば良いと考えられ，それ以外の部分においては冗長なメイクスパン計測・比較であると言える．よって，このような状態 q を特定し，q によってパスの違いが生じている状態空間のみを取り出すことで，必要最小限の状態空間に対して計算を実行することができる．

本論文では,「コントローラ A がある状態 q において制御可能アクションを選択し，コントローラ B が同じ状態 q において他の制御可能アクションか，あるいは

制御可能アクションを選択する」ような状態を分岐と定義する．分岐は先述の特徴から，次のように定義される．

定義 5.1.1. コントローラ $C = \{Q_C, \Sigma_C, \Delta_C, q_0\}$ が与えられたとき，$q \in Q_C$ が $\Delta(q) > 1$ かつ $\Delta^c(q) > 0$ を満たすとき，状態 q は分岐と呼ぶ．

また，分岐によってパスが変化し得る状態空間について考える．本論文で扱うコントローラは有向非巡回グラフであり，トポロジカルソートが可能である．トポロジカルソートされたグラフでは，あるノード q_i において $i < j$ を満たす全てのノード q_j から q_i への有向辺は存在しない．また，$0 < h < i < j$ を満たすすべての h と j について $q_h \to q_j$ となる辺が存在しないとき，q_0 から q_i までの辺の重みの総和は q_i に到達した時点で確定する．一般的なグラフ理論では辺が重みを持ち，それらの総和などによって計画を行う．本論文で扱うメイクスパンも，疑似的な重み付き辺として扱うことができるため，このような条件を満たす q_i があるとき，q_i までのメイクスパンは確定すると言える．したがって，コントローラ上に分岐がある場合，それ以降の状態で q_i のような状態を発見することができれば，分岐によるメイクスパンを部分的に比較することができる．このようなノードを本論文では「収束点」と呼び，以下のように定義される．

定義 5.1.2. 有効非巡回グラフであるコントローラ $C = \{Q_C, \Sigma_C, \Delta_C, q_0\}(|Q_C| = n)$ が与えられたとき，$0 < i < j < k \leq n-1, 0 < k < j$ について以下が成り立つとき，q_k は分岐 q_i の収束点である．

１．q_i が分岐である．２．$(q_l, \ell, q_k) \in \Delta(q_l)$ となるような ℓ が存在しない．

以上から，非支配コントローラを生成するために必要なプロセスの比較は，分岐から定理**??**を満たす状態までの状態空間に対して行えば十分であることがわかる．ただし，コントローラには分岐が複数存在する場合がある．そのため，それらすべての分岐に対し，部分的な状態空間を取り出してプロセスを計測・比較する必要があることに注意する．また，収束点についても同様に複数存在する場合があり，特に，コントローラのゴール状態が単一であるとき，それは必ず収束点である．

5.2 部分的なコントローラ

本論文で提案するコントローラの部分的メイクスパン比較手法では，生成されたユニバーサルコントローラから部分的なコントローラを取り出し，これらに対して比較を行う．ユニバーサルコントローラから取り出す部分的なコントローラを，本論文では”部分コントローラ”と定義する．部分コントローラ C_p は，ユニバーサルコントローラ $U = \{Q_U, \Sigma_U, \Delta_U, q_0\}$ に対して，$Q_{C_p} \subseteq Q_U$ で与えられる状態の

有限集合 Q_{C_p}，$\Sigma_{C_p} \subseteq \Sigma_U$ で与えられるアクションの有限集合 Σ_{C_p}，$\Delta_{C_p} \subseteq Q_U$ で与えられる遷移の有限集合 Δ_{C_p}，$q'_0 \in Q_{C_p}$ を満たす初期状態 q'_0 から成るサブグラフである．

定義 5.2.1. コントローラ $C = \{Q_C, \Sigma_C, \Delta_C, q_0\}$ が与えられたとき，$Q_{C_p} \subseteq Q_C$，$\Sigma_{C_p} \subseteq \Sigma_C$，$\Delta_{C_p} \subseteq Q_C$，$q'_0 \in Q_{C_p}$ を満たすコントローラ $C_p = \{Q_{C_p}, \Sigma_{C_p}, \Delta_{C_p}, q'_0\}$ は C のサブグラフである．

また，メイクスパン比較を行うための部分コントローラとして，必要な条件は以下の 2 つである．

1．メイクスパンを比較するための分岐が存在する．

2．部分コントローラの初期状態から終了状態までのメイクスパンを計測することができる．

さらに，可能な限り部分コントローラの状態空間は小さいことが望ましい．これらを考慮すると，メイクスパン比較を行うための部分コントローラは次の 3 つを満たす．

定義 5.2.2. 部分コントローラ S がメイクスパン比較可能であるとき，S は以下を満たす．

1．部分コントローラ S は分岐を持つ．

2．分岐かそれの祖先かつ最も近い，実行中アクティビティがない状態を初期状態 q_{S_0} に持つ．

3．初期状態 q_{S_0} と終了状態 q_{S_G} において $\zeta(q_{S_0}) = 0$ かつ $\zeta(q_{S_G}) = 0$.

まず，3 について説明する．メイクスパン分析を行うコントローラにはアクティビティが存在し，アクティビティは 1 つの制御可能なアクションで与えられる開始アクションと 1 つ以上の制御不能アクションから与えられる終了アクションのセットで定義される．このアクティビティ制御可能なアクションが発生してから，対応する制御不能アクションのうち 1 つが発生することで 1 つの一連のプロセスとして定義している．ここで重要なことは，アクティビティが実行中な状態が存在することと，制御可能アクションの発生から制御不能アクションの発生までで 1 つのメイクスパンとして計測されることである．例として，アクティビティ $AD = (\alpha, Start, End), Start \in \Sigma_c, End \in \Sigma_u$ が与えられたとすると，$q \xrightarrow{Start} q' \xrightarrow{End} q''$ といったパスが考えられる．このようなパスにおいて，q' ではアクティビティ A が実行中であり，q から q'' のメイクスパンはアクティビティ A のメイクスパン p_A で与えられる．ここで，仮に $q' \to q''$ とするコントローラを部分コントローラとして取り出すと，アクティビティ A の終了アクションは存在するが存在アクションが存在しないというアクティビティの定義に反するコントローラが生成される．このようなコントローラにおいてはアクティビティが定義に反しておりメイクスパン

が計測できないため，メイクスパンを分析する対象のコントローラとしては扱うことができない．そのため，メイクスパンを分析する部分コントローラでは，アクティビティが分断されない，すなわち，初期状態と終了状態において実行中のアクティビティが存在しないことが条件である．

次に，部分コントローラの初期状態に関する条件について説明する．本論文の目的は分析空間の削減であるから，部分コントローラの状態空間は可能な限り小さいことが望ましい．また，メイクスパンの比較は分岐に対して行われるため，これらを考慮すると分岐を初期状態とする部分コントローラを生成することが最善である．しかしながら既に述べた初期状態の条件は実行中のアクティビティが存在しないことであり，場合によって分岐が実行中のアクティビティを有していることがある．以下に例を挙げる．

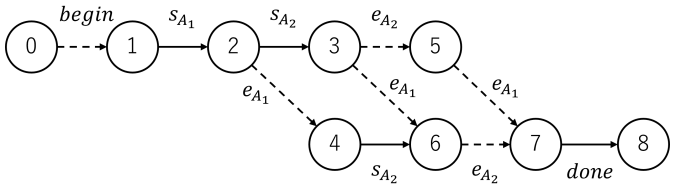

図 5.2: 分岐がアクティビティを実行中である例

図 5.2 の例ではまずアクティビティ A_1 を実行し，その後にアクティビティ A_2 を実行している．

このようなコントローラは「s_{A_1} が発生した後に s_{A_2} を実行する」が要求として与えられた場合に発生し，状態 q_2 は分岐の条件を満たしながらアクティビティを実行中である．このとき，分岐である q_2 を部分コントローラの初期状態とすることは認められないため，それよりも古い，アクティビティを実行中でない q_1 が部分コントローラの初期状態として認められる．

5.3 部分コントローラの生成

提案手法のうち，剪定フェーズでは定義 5.2.2 に基づいてユニバーサルコントローラから部分コントローラを取り出す．部分コントローラを取り出すには，大きく 2 つのプロセスが必要である．ひとつが，ユニバーサルコントローラ内の分岐を発見し，分岐を軸に初期状態を決定する前方剪定，もうひとつが，前方剪定されたコントローラについて，最初の分岐の収束点を特定し，これを終了状態とする後方剪定である．前方と後方では剪定の方法が異なるため，それぞれの状態の探索や剪定方法について，5.3.1 節と 5.3.2 節で説明する．

5.3.1 前方剪定

部分コントローラを生成するにあたり最初に必要なのは，部分比較を行うための分岐を探すことである．同時に，分岐を発見した場合に部分コントローラの初期状態として扱うことのできる状態 q_{S_0} を用意しておくことである．

分岐の特定は，その条件を基にユニバーサルコントローラを深さ優先探索することによって発見することができる．本論文における状態 q が分岐であることの定義は１．有効なアクションが複数あること ($|\Delta(q)| > 1$)，２．有効なアクションのうち，制御可能なアクションがひとつでも存在すること ($|\Delta^c(q)| > 0$) である．よって，これらの条件に該当する状態を探索によって特定する．また，部分コントローラの初期状態として扱う状態 q の探索は，分岐の探索と並行して実行することができる．部分コントローラの前方剪定手順は以下の通りである．

１．状態 q において $\zeta(q) = 0$ かどうかを判定し，これを満たす場合に q を部分コントローラの初期状態の候補 q_sub とする．
２．状態 q が分岐であるかを判定する．
３．q が分岐の場合，ユニバーサルコントローラから q_sub を初期状態とするコントローラを再生成する．
４．q が分岐でない場合は，$(q, \ell, q') \in \Delta(q)$ となる各 q' について再起的に繰り返す．

q_sub の初期値は q_0 であり，根からの探索で実行中アクティビティがない状態に到達する度に更新される．q が分岐でない場合には，q_sub を保存した状態で q' の判定を行うことで，分岐の祖先で最も近い初期状態候補を部分コントローラの初期状態とすることができる．

以上の手順によって初期状態が更新されたコントローラは，部分コントローラ生成フェーズにおいて前方剪定が完了したコントローラとなる．部分コントローラの後方剪定とメイクスパンの比較は分岐が存在しない場合には実行しないから，前方剪定手順の３の後に実行する手順である．

5.3.2 後方剪定

部分コントローラを生成するには，プロセスを比較する前提である分岐と，その分岐によって分かれた遷移が収束する収束点を特定する必要がある．収束点には，コントローラ全体における収束点と，その中の部分的な状態空間における収束点の 2 種類がある．図 3.8 において，ゴール状態と状態 q_{11} はグラフ全体の収束点ある．一方で，分岐 q_6 に着目すると，それらの収束点とは別に状態 q_9 が収束点であることがわかる．部分コントローラを生成する際に特定すべき収束点は，こ

こにおいて q_9 のような，特定の状態空間における収束点である．これは，5.3.1 節において分岐を基に部分コントローラの初期状態を決定したことによる．

5.3.1 節で生成したコントローラは前方剪定が完了し，分岐またはそれに最も近い実行中アクティビティを有さない状態を初期状態とする．再生成したコントローラにおける収束点は先述した特定の状態空間における収束点であるから，このコントローラを用いて分岐の収束点を特定する．コントローラにおける収束点を特定するために，本論文では関節点 [29] の考え方を用いる．

関節点とは，無向グラフにおいて，そのノードを削除することによってグラフが分割されてしまうノードを指す．以下に具体例を示す．図 5.3 のような無向グラフを考える．図 5.3 は連結グラフであり，ノード a からノード g まで全てのノードに到達可能である．ここにおいて，ノード c を削除すると，グラフはノード a，ノード b，ノード g によるグラフとノード d，ノード e，ノード f によるグラフの 2 つに分割されてしまう．このとき 2 つのグラフは連結成分を持たないので，非連結グラフとなる．一方，ノード b を削除する場合を考える．このとき，ノード b が削除されてもノード a からノード c にはノード g を介して到達可能であり，グラフは非連結とならない．このようなグラフにおいて，ノード c は関節点と呼ばれる．

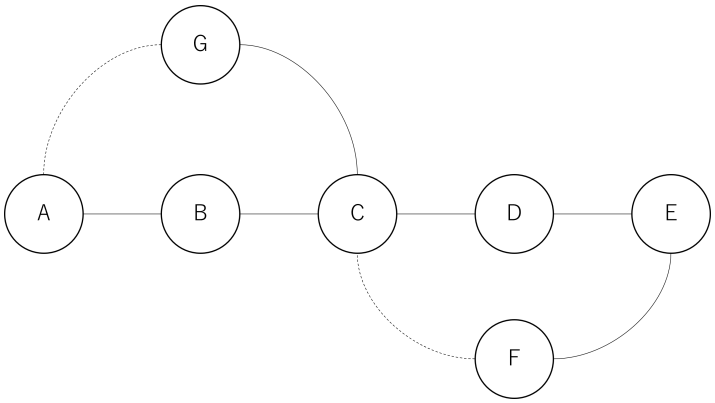

図 5.3: 関節点を持つグラフの例

本論文で扱うコントローラ C は有効非巡回グラフであるが，これをトポロジカルソートされたグラフはノード間の依存性が左辺から右辺への片方向であることに着目すると，無向グラフ C' として見なしても元の有向グラフ C に戻すことができる．そして無向グラフは以下の定理 5.3.1 を満たす．

定理 5.3.1. 無向グラフ G 上のある頂点 A について，A が関節点であるとき，以下は同値である．

1．A の子孫 B から A の祖先への逆辺が存在しない．

2．A の祖先から A の子孫へのパスにおいて A を必ず通過する．

特に，図 5.4 のようにコントローラのサブグラフに着目すると，分岐の収束点 q_9 は部分的にコントローラの関節点となる．

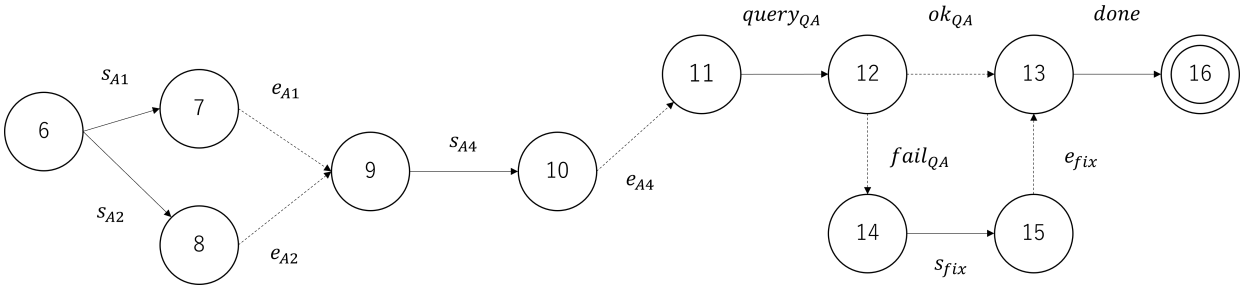

図 5.4: q_6 の分岐に着目したコントローラのサブグラフ

この特性に着目し，コントローラのサブグラフである部分コントローラの関節点を特定する．

関節点を特定するアルゴリズムとして，lowlink アルゴリズムが存在する．lowlink アルゴリズムはグラフにおける関節点問題を線形時間で解くことができるアルゴリズムである．

前方剪定を完了したコントローラから部分コントローラを生成するアルゴリズムをアルゴリズム 2 に示す．

Algorithm 2 Partial controller algorithm

Input: $S, \mathcal{AD}$

1: $[q_0, \cdots, q_n] = \text{TOPOROGICAL\_SORT}(S, Q_G)$
2: $Q_{S_G} = \text{Lowlink}([q_0, \cdots, q_n], \Delta_S)$
3: **for** $q \in Q_{S_G}$ **do**
4: **if** $\zeta(q) = \emptyset$ **then**
5: $\Delta_S = \Delta_S \setminus \Delta(q)$
6: break
7: **return** S

アルゴリズム 2 はコントローラ S とアクティビティ定義 AD を入力とした後方剪定アルゴリズムであり，S は前方剪定されたコントローラを想定している．コントローラ S の状態に対してトポロジカルソートを行い，ソートされた状態群について lowlink アルゴリズムを実行する (1 行目，2 行目)．3 行目以降では，lowlink アルゴリズムによって発見された各関節点 q について，そのうち最も若いかつ実行中アクティビティのない状態 q_j に対して遷移の削除を行う．状態 q_j は関節点であるから $\{(q_i, \ell, q_k) | i < j < k, \ell \in \Sigma\} = \emptyset$ を満たす．よって，ソートされた状態群の中で q_j 以降の状態は到達が不可能となり，後方剪定された S は q_{S_0} を初期状態，q_j を終了状態とする部分コントローラである．

5.4 部分的な非支配コントローラの生成とユニバーサルコントローラへの結果同期

双方の剪定によって生成された部分コントローラは，部分比較プロセスにおいて非支配コントローラアルゴリズムの入力として与える．部分コントローラは実行中のアクティビティが存在せず，メイクスパンを比較する分岐を有するため，メイクスパン比較手法を適用することによって部分非支配コントローラを生成することができる．ただし，非支配コントローラアルゴリズムの出力は非支配コントローラであり，部分比較プロセスでは部分非支配コントローラを獲得するのみである．したがって，部分コントローラに対してメイクスパン比較を行って獲得した結果をユニバーサルコントローラにも同期させる必要がある．

本研究では，この結果同期にあたり非支配コントローラアルゴリズムのプロセス中にある遷移の削除に着目する．4.4 によれば，非支配コントローラアルゴリズムは 2 つのプロセスのメイクスパンを比較し，明らかに劣る遷移を削除することによって非支配コントローラを生成している．アルゴリズム 1 では，13 行目でユニバーサルコントローラの遷移の集合 Δ_K から，メイクスパン比較によって特定された削除対象の遷移の集合の差を取ることで非支配コントローラが生成されていることが分かる．ユニバーサルコントローラ上で非支配コントローラアルゴリズムによって遷移が削除される可能性のある状態は必ずいずれかの部分コントローラに含まれるため，生成されたすべての部分コントローラに対する非支配コントローラを生成時の削除される遷移の集合はユニバーサルコントローラに対する非支配コントローラ生成時の削除される遷移の集合と同値である．すなわち，部分コントローラ C_{p_i} で削除される遷移の集合を $\Delta'_{C_{p_i}}$，ユニバーサルコントローラ U で削除される遷移を Δ'_U とすると，$\sum_{i=1}^{n} \Delta'_{C_{p_i}} = \Delta'_U$ と表すことができる．

定理 5.4.1. ユニバーサルコントローラ U から n 個の部分コントローラ $C_{p_i}(0 \leq i \leq n)$ を獲得できるとき，$\sum_{i=1}^{n} \Delta'_{C_{p_i}} = \Delta'_U$

以上のことから，各部分コントローラのメイクスパン最小化の結果を，削除される遷移の集合を集約しユニバーサルコントローラでも削除することで比較結果の同期が実現できる．

最後に，本論文で提案する段階的メイクスパン比較手法を実行するアルゴリズムをアルゴリズム 3 に示す．

アルゴリズム 3 は自身を再起的に呼び出すアルゴリズムであり，LTS で表現されるコントローラ C，現在の状態 q，部分コントローラの初期状態の候補 q_{sub}，アクティビティ定義 AD，制御可能アクション Σ_c を入力，ユニバーサルコントローラで削除する遷移の集合 D を出力とする．このアルゴリズムは本章の冒頭で説明したように，主に 3 つのフェーズに分けられる．各フェーズはそれぞれ，5 行目から 7 行目が剪定フェーズ，8 行目が部分比較フェーズであり，9 行目および本アルゴリズムの最終出力が同期フェーズに該当する．

Algorithm 3 Stepwise comparison algorithm

Input: $C, q, q_{sub}, \mathcal{AD}, \Sigma_c$

1: **if** $\zeta(q) = \emptyset$ **then**
2: $\quad q_{sub} = q$
3: $\mathcal{D} = \emptyset$
4: **if** $|\Delta(q)| > 1 \&\& |\Delta^c(q)| > 0$ **then**
5: $\quad S = C$
6: $\quad q_{S_0} = q_{sub}$
7: $\quad S = \text{PARTIAL\_CONTROLLER}(S, \mathcal{AD})$
8: $\quad K = \text{MINIMIZING\_MAKESPAN}(S, \mathcal{AD}, \Sigma_c)$
9: $\quad \mathcal{D} = \mathcal{D} \cup (\Delta_S \setminus \Delta_K)$
10: $\quad \mathcal{D} = \mathcal{D} \cup \text{STEPWISE\_COMPARISON}(C, q_{S_G}, q_{S_G}, \mathcal{AD}, \Sigma_c)$
11: **else**
12: $\quad$ **for** $q' \in \{q' | (q, \ell, q') \in \Delta(q)\}$ **do**
13: $\quad\quad \mathcal{D} = \mathcal{D} \cup \text{STEPWISE\_COMPARISON}(C, q', q_{sub}, \mathcal{AD}, \Sigma_c)$
14: **return** $\mathcal{D}$

アルゴリズムの具体的な操作は次の通りである．1 行目と 2 行目では，現在の状態 q において実行中のアクティビティが存在するかどうかを判定し，実行中のアクティビティがない場合は部分コントローラの初期状態の最も新しい候補として q_{sub} に代入する．3 行目では，出力 D の初期化を行う．4 行目からは，q に部分コントローラを生成する条件である分岐が存在するかの判定と，その結果に応じた処理を行っている．q が分岐を持つ場合，条件分岐を真として，5 行目から 10 行目の処理を実行する．5 行目と 6 行目は，部分コントローラの前方剪定の処理である．S は入力で与えられたコントローラのコピーであり，その S の初期状態 q_{S_0} を q_{sub} に更新することで分岐に最も近い実行中アクティビティを持たない状態 q_{sub} がコピーされたコントローラ S の初期状態となる．この操作によって前方剪定が行われたコントローラ S と AD を入力として，アルゴリズム 2 を実行する．アルゴリズム 2 は入力として与えたコントローラの後方剪定を行い，出力として部分コントローラを獲得することができる．7 行目の処理で獲得できたコントローラ S は、分岐 q についてのメイクスパン比較を行う部分コントローラである．そして，得られた部分コントローラ S に対して，非支配コントローラアルゴリズムを適用する (8 行目)．ここで獲得できるのは非支配コントローラであるので，9 行目の通り比較実行前後のコントローラ同士の差分を取ることでこの部分コントローラにおいて削除された遷移の集合を獲得することができる．この一連の操作によって q から q_{S_G} までのメイクスパン比較が完了したので，次は q_{S_G} 以降の状態について同様の処理を行っていく．10 行目では，これを行うために入力を C，q_{S_G}，q_{S_G}，AD，Σ_c としてこのアルゴリズムを再起的に呼び出し，その出力と D の和集合を取ることで q 以降のゴール状態までの状態における削除遷移を特定する．

また，q において分岐が存在しなかった場合には，12 行目と 13 行目にあるように，q で有効な各アクション ℓ によって遷移する状態 q' について同様にこのアルゴリズムを呼び出す．入力は，現在の状態を q' に更新したものである．以上の操作によって，最終的にユニバーサルコントローラに同期させるすべての遷移が D として獲得できる．このアルゴリズムを 1 の 3 行目から 13 行目に置き換えることで，ユニバーサルコントローラへの結果同期でき，部分比較によるメイクスパン最小化が実現できる．

第6章　評価

本章では，提案する手法の有効性を検証するための実験を行い，その結果について議論する．実験にあたる懸賞事項や実験に用いるモデル，ツールは6.1節で説明する．6.1節の設定の上で行った実験で獲得できた結果は6.2節で示され，この結果についていくつかの観点から議論を行う．最後に，本章で示した結果の妥当性を脅かす要因として考えられるものを、6.3節にて内的要因と外的要因から考察し列挙する．

6.1　実験設定

本論文で提案する手法を評価するにあたり，以下の2つの事項を検証する．

１．提案する部分比較手法が先行研究の手法と比較してどれだけの計算時間削減効果を持つか．
２．提案する手法が効果を発揮するドメインやモデルがどのようなものか，また，それらに共通する特徴はなにか．

検証事項１では，先行研究の計算時間の削減という本論文の目的に対して，提案手法がどれほど達成できているかを定量的に調査する．比較対象は先行研究のメイクスパン最小化アルゴリズムであり，それぞれのアルゴリズム全体の計算時間を計測し，比較する．
検証事項２では，提案手法が効果を発揮する条件を検証する．システムにはドメイン特有の特徴を持つものがあり，例としては，工場の生産ラインは逐次的な実行が多く，分散システムにおいては並列実行が多くみられる．また，動作環境の不確実性も扱うシステムにより多少が生じる．本論文では，いくつかのモデルに対して提案手法を適用することで，どのようなシステムやモデルに対して効果を発揮できるのか，また，有効でないモデルが持つ特徴がどのようなものかを調べる．

本論文の評価では，主に2種類のシナリオを扱う．1つは，本論文の例題モデルとして扱っている，産業オートメーションシステム (IA)[30] である．実験では，システムの入力として与えられる製品の種類やシステムが扱うことのできるツール

の数，製品加工時に用いるツールの要求を変化させて，モデル毎の計算時間を計測・比較する．以下に，実験で用いる IA モデルを示す．

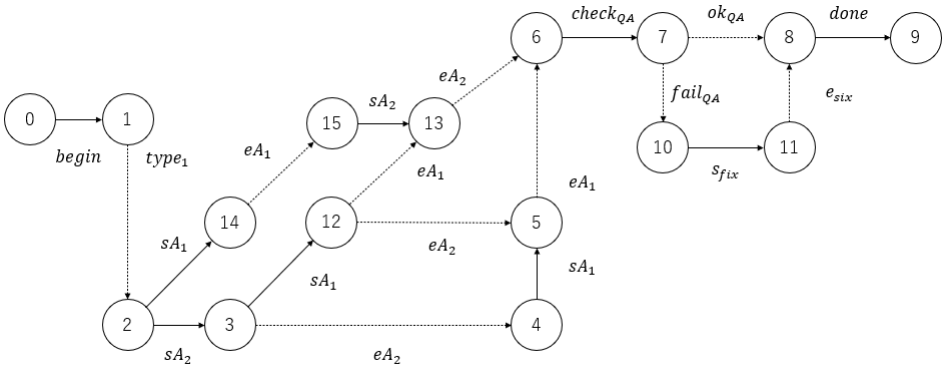

図 6.1: IA[2] モデルのコントローラ

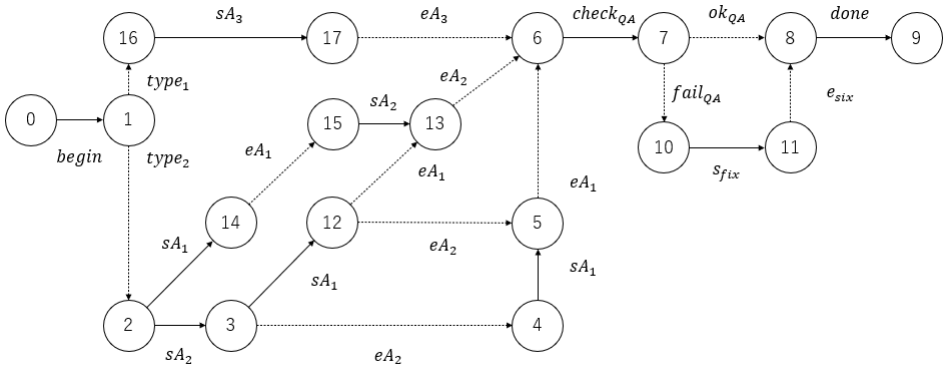

図 6.2: IA[3] モデルのコントローラ

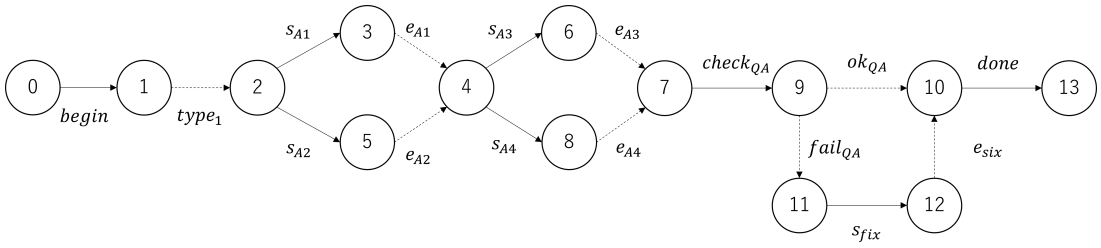

図 6.3: IA[4] モデルのコントローラ

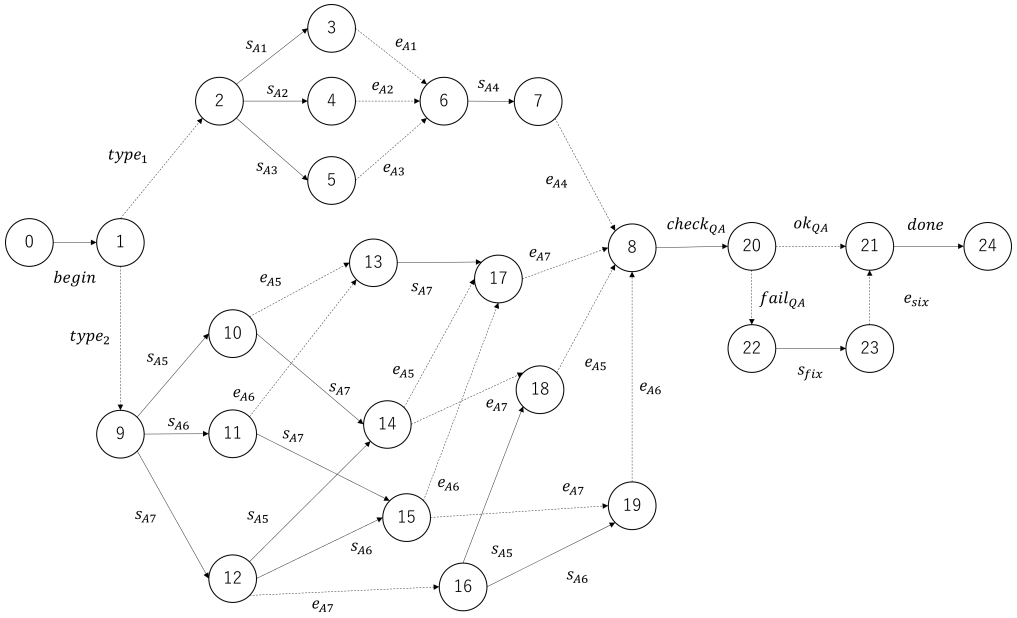

図 6.4: IA[7] モデルのコントローラ

IA モデルには，システムへの入力である加工製品の種類による不確実性が存在する．IA[2] は IA[3] で 2 種類ある入力のうち一方を削除し，不確実性を減らしたモデルである．実験においては，モデルの不確実性の多少がアルゴリズムの計算時間にどれほどの影響を与えるかを調査する．IA[4] および IA[6] モデルは，分岐が複数存在する場合に提案アルゴリズムが非支配ユニバーサルコントローラを生成いていることを確かめると同時に不確実性の多少や分岐の連続性が計算時間にもたらす影響を調べる．IA[7] は IA[6] より問題規模を複雑にし，分岐における選択肢が多い場合や，ひとつの分岐の問題規模が大きい場合に計算時間にどれほどの差が生じるかを検証するために用意した．これらのモデルは共通して，加工製品の入力 (type[1] または type[2]) が与えられるとそれに応じた加工プロセス (A[i]) を逐次または並列で実行し，加工が完了したら品質検査 (QA) を実行し，その結果に応じて修繕 (Fix) を行い製品を出力する．品質検査の結果は共通のため，モデルで異なるのは入力される種類の数と加工プロセスにおけるツールの数や要求のみである．

もう 1 つは，ジョブスケジューリング問題を扱うジョブタスクモデル (JOB)[31] である．このモデルは，システムが与えられた n 個のタスクを要求に従って逐次，または並列に実行するシステムである．
ここでは，システムは 4 つのタスクを与えられた場合を考える (JOB[4])．タスクはすべてアクティビティ $A[i]$ として定義され，それぞれ $sA[i]$ を開始アクション，$eA[i]$ を終了アクションとしている．また，このモデルでは要求として 2 つ与えられ，$sA[2]$ の発生は $A[1]$ の実行後，$sA[1]$ の発生は $A[3]$ または $A[4]$ の開始後である

必要がある．実験においては，タスクの数を変化させ，それに伴うタスクの実行順に関する要求を追加したものを扱う．

実験には，Modal Transition System Analyser(MTSA)[32] を用いる．MTSA は、動作環境をモデル化した LTS(環境モデル) と，要求を線形時相論理の集合によって形式化した FLTL 式を入力としてコントローラを自動生成する技術である．アプリケーションは JAVA 言語で記述されており，コントローラ合成問題において制御器の視覚化やロボット等のインターフェースの事前定義によりロボット上で生成コントローラ [33] を実行することが可能である．MTSA のインターフェースを図 6.5 に示す．

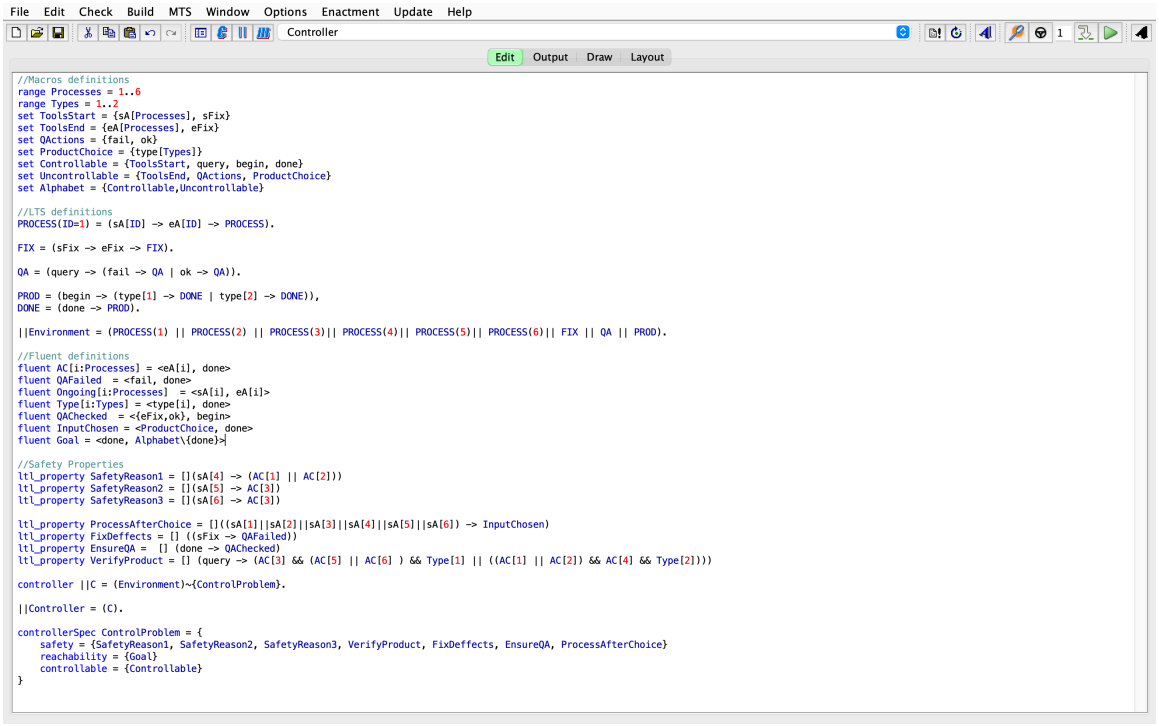

図 6.5: MTSA 上の入力情報

ここでは例題モデルとして扱っている産業オートメーションを例に入力を与えている．この制御問題は Finite State Process[34] というプロセス代数記法によって記述されている．例として，修繕プロセス Fix は図 6.5 において Fix = (sFix -> eFix -> FIX) と記述される．MTSA において複数の LTS は 2.2.2 節で説明した並列合成によって環境モデルとして統合することができ，図 6.5 ではそれぞれの振る舞いの LTS の並列合成を || Environment = (PROCESS(1) || PROCESS(2) || PROCESS(3) || PROCESS(4) || PROCESS(5) || PROCESS(6) || FIX || QA || PROD) として記述している．また，フルーエントは 3.5 と同様に記述される．コントローラ合成には環境モデルである LTS Environment とコントローラの仕様 controllerspec を

用いる．controllerspec には制御可能なアクションの集合 Σ_c や到達可能性ゴール (Reachability Goal)，要求などが含まれている．

これらの標準的な入力に加え，非支配コントローラアルゴリズムで使われるアクティビティ定義と，アクティビティやアクションのメイクスパンの比較情報が与えられる．アクティビティの定義は入力下部の Activity Definition でフルーエント形式で与えられ，比較情報は Activity relations で 2 つのアクティビティとそれらの間の関係性を入力とする．例として，アクティビティ A[1] とアクティビティ A[2] の間にメイクスパンにおいて A[1] の方が A[2] よりも優れている，すなわち A[1]<A[2] の関係が成り立つとき，Activity relations の入力として A[1]<A[2] を与える．これらの比較は，先行研究 [18] らが MTSA を拡張実装した Z3 ソルバー [28] の機能によって成り立っている．

6.2 実験結果と議論

6.1 節で述べた 2 つのモデルを用いて，先行研究の手法である非支配コントローラアルゴリズムと提案手法である部分比較アルゴリズムの計算時間を比較する．実験は，IA[2]-IA[4]，IA[6]，IA[7] の 5 つと，JOB[4]-JOB[6] の 3 つの計 8 つのモデルを対象に行った．各モデルに対して，非支配コントローラアルゴリズムと部分比較アルゴリズムの計算結果は表 6.1 の通りである．表に示した時間は，各モデルに対してアルゴリズムを 5 回実行して得られた計算時間評価するにあたり，提案手法による先行研究のメイクスパン最小化手法の計算時間の削減割合 R を，提案手法の計算時間 T_P，先行研究手法の計算時間 T_E として $R = 1 - \frac{T_P}{T_E}$ と定義した．

表 6.1: 各モデルにおける各手法の計算時間と時間削減の割合

モデル [N]	非支配コントローラ [ms]	部分比較 [ms]	計算時間の R 値 [%]
IA[2]	934.4	262.2	71.94
IA[3]	1358.6	277.0	79.61
IA[4]	644.0	248.0	61.49
IA[6]	878.6	239.2	72.77
IA[7]	18503.2	1639.0	91.14
JOB[4]	3147.0	3215.6	-2.178
JOB[5]	14641.8	14909.8	-1.830
JOB[6]	150887.8	153984.6	-2.052

まずシナリオ毎の結果を見てみると，IA モデルではいずれも計算時間の削減効果がある一方で，JOB モデルでは R 値が負の値となっている．これは非支配コントローラアルゴリズムと比較して計算時間が増加してしまっていることを示しており，提案手法が JOB モデルにおいて効果を持たなかったことを意味している．

次に，IA モデルにおいて削減率が異なる点に着目する．実験で扱った IA モデルでは，低いもので 61.49%，高いもので 91.14%の削減効果を示した．また，以下に各モデルで提案手法を適用した際の部分コントローラの問題規模を示す．Q_U はユニバーサルコントローラの状態数であり，$\sum_{i=1}^{n} Q_{C_{p_i}}$ は，各部分コントローラが持つ状態数の総和である．

表 6.2: 各モデルにおける状態数から見た問題規模

モデル [N]	Q_U	分岐数	$\sum_{i=1}^{n} Q_{C_{p_i}}$
IA[2]	16	1	9
IA[3]	18	1	9
IA[4]	14	2	8
IA[6]	16	2	8
IA[7]	25	2	17
JOB[4]	34	1	32
JOB[5]	53	1	51
JOB[6]	101	1	99

IA[2] と IA[3] は 6.1 節で述べたように不確実性が持つ計算時間の影響を検証するために使ったモデルであるが，表 6.1 および表 6.2 から，部分コントローラの状態空間が変わらない，すなわち本来扱うべき問題空間が同じ場合でも，加工製品の種類という不確実性によって非支配コントローラアルゴリズムは計算時間が大きくなってしまうことが確認でき，本論文の実験では，この不確実性の有無によって計算時間が 1.45 倍になってしまっていることが確認できた．一方で提案手法では部分比較を行う問題規模が同じであるため，先行研究の手法と比べて計算時間の増加は小さい．考えられる計算時間の増加は部分コントローラの生成フェーズにおける分岐や収束点の探索によるものであるが，これらの探索は線形時間で行えるため，指数関数時間で実行するメイクスパン最小化プロセスと比較してその影響は無視できると言える．

IA[4] と IA[6] では，非支配コントローラアルゴリズムでの計算時間が 1.36 倍になっている一方で部分比較アルゴリズムの計算時間は 0.965 倍になっている．これらも同じく部分コントローラの問題規模は等しいことから，提案手法での計算時間が概ね等しいことが確認できる．また，非支配コントローラアルゴリズムでの計算時間の差についてはツール数と不確実性の 2 つが挙げられ，提案手法は部分コントローラに関係のない状態空間におけるこれらの増減による影響を受けないことがこれらのモデルからわかる．

IA[7] は入力である加工製品 2 種類に対し，一方を 3 種類のツールから 1 つと指定されたツールの計 2 つによる加工，もう一方を，3 種類のツールをある程度していされた順番で用いる加工としたシステムである．非支配コントローラアルゴリ

ズムにおいて IA[7] 以外で最も計算時間が大きかった IA[3] のモデルと比べて単純な状態空間が状態数が 7，遷移数が 15 増加している．部分コントローラ外の不確実性は変わらないが，比較部の複雑性によって非支配コントローラアルゴリズムでは計算時間が 13.6 倍になっている．提案手法では同じ IA[3] と IA[7] において計算時間は 5.92 倍であり，計算時間の爆発は 2.3 倍の差が生じている．この比較によって，提案手法は部分比較部以外の不確実性や状態数・遷移数の増加の影響を限りなく小さくし実験モデルでは最大 91.1%の削減率を持つと同時に，部分比較部の複雑性が増加した場合においても最大 56.5%の計算時間抑制効果が確認され，当モデルにおける有効性が示された．

一方の JOB モデルについて，IA モデルとは対照的に計算時間の削減効果が見れなかったことについて議論する．表 6.2 を見ると，JOB タスクモデルにおいては部分コントローラの問題規模がユニバーサルコントローラとほぼ同等であることがまず確認できる．提案手法は非支配コントローラアルゴリズムを適用する状態空間を部分コントローラに限定することで計算時間を削減することを目的とした手法であり，非支配コントローラアルゴリズムの本質的な計算時間を改善しているわけではない．そのため，JOB モデルの結果で示されているような，部分コントローラの問題規模がユニバーサルコントローラに対し小さくできていないような場合には効果が発揮されないことがこの結果からも確認できる．

JOB モデルがこのような結果になった原因として第一に考えられるのは，分岐がひとつの非常に大きな問題であったことが挙げられる．IA モデルではシステムへの入力によっていくつかの製品加工に分けられるため，分岐が存在しても問題規模はユニバーサルコントローラと比較して必然的に小さくなった．JOB モデルでは制御不能アクションによっていくつかの状態に遷移する可能性があるような状態が存在せず，IA モデルであった問題規模の縮小が存在しなかった．このようなモデルの特徴は，コントローラを複数の小規模な問題に分けることのできない分散システムや，複数の小規模な問題から成り立つがそれらが並行して実行されるようなシステムが持つものであり，逆に，制御問題が複数のフェーズから成り立っていることで収束点が初期状態からゴール状態までのパスに存在したり，システムからの入力による場合分けといった小規模な問題が並べられているようなモデルでは効果が発揮できることが考えられる．

6.3 妥当性への脅威

本論文の評価にあたり行った実験とその結果について，その結果や考察の妥当性を脅かす要因を，外的要因と内的要因の 2 つに分けて列挙する．

6.3.1 内的妥当性への脅威

内的妥当性としては，実験に使用したコンピュータに由来する実験結果の偏りや，アプリケーションの特定が考えられる．本論文では実験に MTSA を使用した．MTSA は入力に従って要求仕様を生成するが，これを用いて直接システムを稼働させることができる．MTSA は環境の変化を考慮した合成プロセスを持つため，類似性のあるモデルについて連続で合成を行うと合成時間が短縮される事例がある．6 章ではこれを排除するために合成毎にアプリケーションを再起動して影響を小さくできるよう実験を行った．また，実験毎の偏りを考慮し，各モデルおよび各手法において複数回のコントローラ生成を行った上でその平均値を算出し評価対象とした．複数回の実行の結果を考えると先述の特性よる偏りはないと推測されるが，類似したアプリケーション由来の影響がある可能性がある．

6.3.2 外的妥当性への脅威

評価に対する外的妥当性を脅かすものとして，まず実験に用いたモデルが 2 種類しかないことが挙げられる．実験で用いたモデルである IA モデルと JOB モデルは先行研究に由来するものであるが，本研究の評価にあたっては問題設計として非常に極端な性質を持っている可能性がある．例えば JOB モデルは本研究において望ましい結果は得られなかったが，IA モデルと同様に一連のプロセスの後に品質チェックプロセスのような独立した工程が存在することで結果が変わってくる可能性がある．また，本研究は IA をモチベーション例として行ったため，IA モデルに特化した手法である可能性も考えられる．これらの懸念を解消するためにはより多くのモデルやシナリオでの実験が求められるので，将来研究のひとつとしたい．

第7章　関連研究

本章では，本研究に関連する研究についていくつか説明する．ここで主に挙げるのは 1 章にて既に言及している，非機能要求に関する研究である．また，本研究ではシステム稼働前にコントローラを合成するオフライン手法であるが，自動運転や飛行制御といった計画問題はオンラインでの実行に関心が寄せられている．これについても 7.3 節で挙げる．

7.1　定量評価手法

離散事象系制御問題の定量的手法では，コントローラにコスト関数を導入することによってコントローラの性能を測定し，そのコスト値を最適化しつつゴール状態に到達することを目標としている．具体的な研究事例として，[14] では目標状態のうちの 1 つに最適パスで到達するヒューリスティック探索アルゴリズムを提案している．Bloem ら [35] は辞書的平均ペイオフゲームを解くことによってリアクティブシステムの定量的特性を表現できることを示した．また，古典的計画 (Classical Planning) の観点では，Planning Domain Definition Language(PDDL) の第 3 版 [36] でプリファレンスが実装され，プリファレンス違反数と計画の長さといった方法により定量的なコントローラ比較を行っている．しかしながら，コントローラの性能を評価するための関数をユーザーのプリファレンスに合わせて設定することは難しい．また，コントローラのすべての評価対象に対して定量的な数値を設けることは，観測可能性や不確実性の観点から考慮しても不可能であると言える．

7.2　定性評価手法

システムの非機能要求への評価方法として，定性的なアプローチも積極的に取られてる [12][13]．本研究のベースとなった定性的メイクスパン比較フレームワーク [18] もその 1 つである．定性的な評価手法のアプローチの 1 つとして，時間的なプリファレンスを定性的に定義しており，ブール言語と算術演算子で表現している [15]．この方法ではアクション間の時間をモデル化することができ，本研究で扱ったようにプロセス A とプロセス B がそれぞれ T_A と T_B の時間で作業を完了するような場合に $T_A > T_B$ といった表現が可能である．これらのモデル化を考え

る際，ドメインによっては，2 者間の関係性が一意でない場合がある．例えば経路計画を考える際，通常であれば経路 A が好ましいと思われる一方で，天気が雨であったり渋滞が発生していたりといった条件下においては経路 B を選択することを好ましいと思う場合がある．条件付き選好ネットワーク (CP-net)[37] では，変数間の条件付き選好関係をユーザー側が指定することができ，これの拡張である TCP-net[38] ではこれに追加してプロセスの時間とコストといった変数間のトレードオフもモデル化することができる．また，本研究で扱った生成された制御器に対してプリファレンスを考慮する方法とは別に，プリファレンス自体を LTL 式で表現できるようにする方法がある [39]．プリファレンスはある種の要求としてみなすことができ，これを利用してアクションに関連するいくつかの要求のうち，どれだけを満足するかによってそのアクションの優位性を評価する．

7.3 リアルタイムにおける計画

本論文で扱った合成は主にオフラインにおける実行であり，システム稼働前に事前にコントローラを合成することができる．そのため，合成時間自体はそれほど重要ではなく，求められるのは終了時間である．一方で，ドメインによってはリアルタイム実行時に環境に変化が発生し，コントローラを再合成する必要が生じる場合があり [40][41]，本論文で扱わなかった自動走行やドローンといったドメインの場合には環境およびコントローラの更新が求められる．このような場合には環境の変化からの素早い適応が必要であり，すなわち短時間でのコントローラの再生成がしばしば求められる．Wang ら [40] は無人航空機のオンライン経路計画にあたり，計画効率の向上のために改良された遺伝的アルゴリズムを提案している．Wang らの提案した遺伝的アルゴリズムでは，母集団情報を駆使して集団内の一部の個体を分析し，計画空間内のさまざまな領域の検索値を判断することによって進化オペレーターの生成領域を合理的に制限している．オンラインのコントローラ再合成における非機能要求のサポートは自動走行などに求められるが，パラメータの取得に困難が残る．

第8章　おわりに

8.1　本論文のまとめ

本論文は，離散制御器合成によって生成された安全性と到達可能性を保証する単一の終了状態を持つコントローラに対して，メイクスパンに着目した定性的最小化フレームワークが持つ計算時間の限界を，分析空間を小さくすることで計算時間を削減することを目的とした．本論文では，目的を達成するために，標準的な離散制御器合成によって生成されたユニバーサルコントローラから，分岐と収束点という 2 種類の特徴を持つ状態を特定し，これらを基準にサブグラフである部分コントローラを生成して分析する方法を提案した．ユニバーサルコントローラには，メイクスパンを比較することによって明らかに劣った遷移を削除できる場合がある．このような削除可能な遷移を持つ状態は分岐として，メイクスパン比較の実行条件である．また，単一の終了状態を持つコントローラにおいては分岐による発散は必ず収束し，これは終了状態だけでなくそれまでのパス中にも存在する場合がある．収束点ではそれまでのメイクスパン比較結果も収束することから，分岐と収束点に着目して部分コントローラを生成し，部分コントローラに対してメイクスパン最小化アルゴリズムを適用することによって，分析結果が等価でありながら問題規模を縮小し，計算時間を間接的に削減することができる．提案手法は分析対象の状態空間の外に存在する不確実性や遷移に対しての影響を限りなく小さくし，同時に分析空間の複雑さによる計算時間の増加にも一定の抑止力があることが実験からわかった．一方で，ひとつの大きな分析問題である場合や，小規模なシステムが相互に作用し合って動作する場合には，コントローラを複数の小規模な分析問題に分割できず分岐や収束点の探索がオーバーヘッドになってしまうことが課題として挙げられた．

8.2　将来研究

本論文の提案手法は，グラフにおける計画問題の収束性に着目した分析空間の縮小である．先行研究ではメイクスパンに着目しており，システム内のプロセスの実行時間を計測するためにアクティビティを導入している．その他の非機能要求の例として，ロボットの燃料といったコストや，ドローンの飛行時における行動リスクといった要素が挙げられる．これらのパラメータも同様にアクティビティ

に導入できるため，本論文の実験と同条件で実行が可能である．従って，先行研究の手法の改良に適用できる可能性がある．

また，本手法の拡張性として，部分コントローラの再起的な生成が挙げられる．提案手法はユニバーサルコントローラ上の分岐に対して部分コントローラを生成し，部分分析を実行する．ただし，部分コントローラもまた小規模のユニバーサルコントローラとして考えることができるため，これに対して再度部分コントローラを生成することも可能である．本論文では削減効果を示さなかった，JOB モデルのような大規模な比較問題に対して，内部的に分割することで計算時間が削減できる可能性が考えられる．一方で，部分コントローラの分析結果はユニバーサルコントローラに戻されるため，無計画に再起生成することが計算時間の削減に繋がるかは疑問の余地がある．特に本論文の JOB モデルのように，それ以上分割できないようなモデルに対して実行した場合には，分岐や収束点の探索がオーバーヘッドとなってしまうため，このような場合の実行の中止ができる方法の確立が求められる．また，部分分析によって規模を小さくできる場合でも，前提となるコントローラの規模によっては大きな効果を持たないことも想定される．したがって，部分コントローラに対して再起的に部分分析を行うには，問題の規模や分割可能性といったグラフにおける特徴をはじめ，ドメインやシステムの特性にも考慮する必要がある．更に，本論文では 2 つのモデルを用いて提案手法が有効なモデル・ドメインの特定を調査したが，システムにはこれ以外にも多種多様な特性が存在する．実験で使ったモデル以外に対しての有効性を調査することで，提案手法の適用可能範囲やより有用な方法を発見することに繋がると考える．

謝辞

本論文の執筆および研究にあたり，指導教員として様々なご指導とご助力を賜りました，早稲田大学大学院 基幹理工学研究科 情報理工・情報通信専攻の本位田教授と東京工業大学大学院 鄭准教授に深謝申し上げます.

また，論文執筆および研究に関して多大な助言とご助力を頂きました，山内先輩や平野先輩を始めとした早稲田大学大学院 本位田研究室の皆様と，早稲田大学大学院 基幹理工学研究科 情報理工・情報通信専攻 深澤研究室，鷲崎研究室の皆様にも感謝申し上げます.

参考文献

[1] Pierre-Alain Bourdil, Bernard Berthomieu, and Eric Jenn. Model-checking real-time properties of an auto flight control system function. In *2014 IEEE International Symposium on Software Reliability Engineering Workshops*, pages 120–123, 2014.

[2] Chandrashekhar Singh, Jagadish Shivamurthy, and Asha Garg. Model based test framework for verification of flight control software. In *2023 International Conference on Computer, Electrical & Communication Engineering (ICCECE)*, pages 1–5, 2023.

[3] Markus Weißmann, Stefan Bedenk, Christian Buckl, and Alois Knoll. Model checking industrial robot systems. In Alex Groce and Madanlal Musuvathi, editors, *Model Checking Software*, pages 161–176, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[4] Johan Arcile, Raymond Devillers, and Hanna Klaudel. Verifcar: A framework for modeling and model checking communicating autonomous vehicles. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '20, pages 2126–2127, Richland, SC, 2020. International Foundation for Autonomous Agents and Multiagent Systems.

[5] Wei Xiao, Noushin Mehdipour, Anne Collin, Amitai Y. Bin-Nun, Emilio Frazzoli, Radboud Duintjer Tebbens, and Calin Belta. Rule-based optimal control for autonomous driving. In *Proceedings of the ACM/IEEE 12th International Conference on Cyber-Physical Systems*, ICCPS '21, pages 143–154, New York, NY, USA, 2021. Association for Computing Machinery.

[6] Jorge Badilla-Solórzano, Svenja Spindeldreier, Sontje Ihler, Nils-Claudius Gellrich, and Simon Spalthoff. Deep-learning-based instrument detection for intra-operative robotic assistance. *International Journal of Computer Assisted Radiology and Surgery*, 17(9):1685–1695, Sep 2022.

[7] Edmund M. Clarke, Orna Grumberg, and Doron A. Peled. *Model checking*. MIT Press, Cambridge, MA, USA, 2000.

[8] WS Levine and M. Reyhanoglu. *The Control Handbook*. 12 2010.

[9] P.J.G. Ramadge and W.M. Wonham. The control of discrete event systems. *Proceedings of the IEEE*, 77(1):81–98, 1989.

[10] Christos G. Cassandras and Stéphane Lafortune. *Introduction to Discrete-Event Simulation*, pages 593–651. Springer International Publishing, Cham, 2021.

[11] Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking (Representation and Mind Series)*. The MIT Press, 2008.

[12] Ganesh Ram Santhanam, Samik Basu, and Vasant Honavar. Representing and reasoning with qualitative preferences for compositional systems. *J. Artif. Int. Res.*, 42(1):211–274, sep 2011.

[13] Jorge Baier and Sheila Mcilraith. Planning with preferences. *AI Magazine*, 29:25–36, 12 2008.

[14] K.M. Passino and P.J. Antsaklis. On the optimal control of discrete event systems. In *Proceedings of the 28th IEEE Conference on Decision and Control,*, pages 2713–2718 vol.3, 1989.

[15] Torsten Schaub James P Delgrande and Hans Tompits. A general framework for expressing preferences in causal reasoning and planning. In *Journal of Logic and Computation*, pages 871–907, 2007.

[16] Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi. Parametric real-time reasoning. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing*, STOC '93, pages 592–601, New York, NY, USA, 1993. Association for Computing Machinery.

[17] Clark Barrett and Cesare Tinelli. *Satisfiability Modulo Theories*, pages 305–343. Springer International Publishing, Cham, 2018.

[18] Ezequiel Castellano, Victor Braberman, Nicolás D ' Ippolito, Sebastián Uchitel, and Kenji Tei. Minimising makespan of discrete controllers: A qualitative approach. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, pages 1068–1075, 2019.

[19] Dimitra Giannakopoulou and Jeff Magee. Fluent model checking for event-based systems. *SIGSOFT Softw. Eng. Notes*, 28(5):257–266, sep 2003.

[20] Michael Jackson. The world and the machine. In *1995 17th International Conference on Software Engineering*, pages 283–283, 1995.

[21] Nicolas D'Ippolito. Synthesis of event-based controllers: A software engineering challenge. In *2012 34th International Conference on Software Engineering (ICSE)*, pages 1547–1550, 2012.

[22] P. J. Ramadge and W. M. Wonham. Supervisory control of a class of discrete event processes. *SIAM Journal on Control and Optimization*, 25(1):206–230, 1987.

[23] Louise Pryor and Gregg Collins. Planning for contingencies: a decision-based approach. *J. Artif. Int. Res.*, 4(1):287–339, may 1996.

[24] Luca de Alfaro and Thomas A. Henzinger. Interface automata. 26(5):109–120, sep 2001.

[25] Kai Cai W. Murray Wonham. *Supervisory Control of Discrete-Event Systems.* Springer Cham, 2018.

[26] RICHARD BELLMAN. A markovian decision process. *Journal of Mathematics and Mechanics*, 6(5):679–684, 1957.

[27] Leonardo De Moura and Nikolaj Bjørner. Z3: an efficient smt solver. In *Proceedings of the Theory and Practice of Software, 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, TACAS'08/ETAPS'08, pages 337–340, Berlin, Heidelberg, 2008. Springer-Verlag.

[28] Z3 api documentation. http://z3prover.github.io/api/html/index.html.

[29] Robin J Wilson. *Introduction to graph theory.* John Wiley & Sons, Inc., USA, 1986.

[30] Nicolás D'ippolito, Victor Braberman, Nir Piterman, and Sebastián Uchitel. Synthesizing nonanomalous event-based controllers for liveness goals. *ACM Trans. Softw. Eng. Methodol.*, 22(1), mar 2013.

[31] Laurent Fribourg, Romain Soulat, David Lesens, and Pierre Moro. Robustness analysis for scheduling problems using the inverse method. In *2012 19th International Symposium on Temporal Representation and Reasoning*, pages 73–80, 2012.

[32] Nicolas D'Ippolito, Dario Fischbein, Marsha Chechik, and Sebastian Uchitel. Mtsa: The modal transition system analyser. In *2008 23rd IEEE/ACM International Conference on Automated Software Engineering*, pages 475–476, 2008.

[33] Víctor Braberman, Nicolas D'Ippolito, Nir Piterman, Daniel Sykes, and Sebastian Uchitel. Controller synthesis: from modelling to enactment. In *Proceedings of the 2013 International Conference on Software Engineering*, ICSE '13, pages 1347–1350. IEEE Press, 2013.

[34] Jeff Magee and Jeff Kramer. *Concurrency: State Models and Java Programs*. Wiley Publishing, 2nd edition, 2006.

[35] Roderick Bloem, Krishnendu Chatterjee, Thomas A. Henzinger, and Barbara Jobstmann. Better quality in synthesis through quantitative objectives. In Ahmed Bouajjani and Oded Maler, editors, *Computer Aided Verification*, pages 140–156, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[36] Alfonso Emilio Gerevini and Derek Long. Plan constraints and preferences in pddl 3 the language of the fifth international planning competition. 2005.

[37] C. Boutilier, R. I. Brafman, C. Domshlak, H. H. Hoos, and D. Poole. Cp-nets: A tool for representing and reasoning withconditional ceteris paribus preference statements. *Journal of Artificial Intelligence Research*, 21:135–191, February 2004.

[38] Ronen I. Brafman, Carmel Domshlak, and Solomon E. Shimony. On graphical modeling of preference and importance. *J. Artif. Int. Res.*, 25(1):389–424, mar 2006.

[39] Meghyn Bienvenu, Christian Fritz, and Sheila A. McIlraith. Planning with qualitative temporal preferences. In *Proceedings of the Tenth International Conference on Principles of Knowledge Representation and Reasoning*, KR'06, pages 134–144. AAAI Press, 2006.

[40] Xiaohai Wang and Xiuyun Meng. Uav online path planning based on improved genetic algorithm with optimized search region. In *2019 IEEE International Conference on Unmanned Systems (ICUS)*, pages 1–6, 2019.

[41] John Jackson, Luca Laurenti, Eric Frew, and Morteza Lahijanian. Synergistic offline-online control synthesis via local gaussian process regression. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pages 2232–2239, 2021.

[42] William Cushing, Subbarao Kambhampati, Mausam, and Daniel S. Weld. When is temporal planning really temporal? In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, pages 1852–1859, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.