名古屋大学 学位論文

時間的制約充足型統合制御に基づく
事象駆動型 6 脚移動ロボットの不整地歩行

名古屋大学 大学院 工学研究科

機械システム工学専攻

機械知能学講座

村田　勇樹

令和元年

目次

図目次

表目次

第 1 章

序論

これまで，多脚移動ロボットの研究においては，制御器をどのように設計するべきかが 1 つの課題として挙げられていた．多脚移動ロボットの制御器は，多くの場合で機能毎に分割したモジュール構造が採用される．その分割された機能のうち歩行に必要なものにはたとえば，姿勢制御や運動計画，接触センサを用いた局所制御などの複数の制御の要素 (本論文では制御要素と呼ぶ) が含まれている．いずれも制御要素間の相互作用を考慮しつつ，実装しなければならない．なぜなら，単純に制御要素同士を組み合わせるだけは相互の働きが衝突し，デッドロックに陥ることが予想されるからである．さらに，ある実験環境上で制御要素同士の干渉がないことが確認できていても，環境を変えた場合にも同じように制御要素同士の干渉が起こらないとは限らない．本論文では，環境を限定せずに複数の制御要素を含む制御系の設計を行う手法を提案する．本章では，研究背景として従来の多脚移動ロボットの従来研究や近年の実用化に向けての動きを紹介しながら，本研究で取り扱う問題とその解決手段の概要を述べる．

1.1　本研究の背景

人に不適応な環境や災害地を含む未知の不整地環境での移動を目的としたロボットが開発されてきた．1995 年の阪神・淡路大震災や地下鉄サリン事件をきっかけに日本での災害対策用の移動ロボットの開発が本格的に開始し [2]，2011 年 3 月の東日本大震災時では複数の災害対策ロボットが災害対応や復旧に利用された [3]．このとき，福島第一原発の瓦礫処理・原子炉内部環境の観測が必要な際に，作業者の被曝リスクを減らすために，複数の無人施行機械 (バックホウ・クローラダンプ・カメラ車・脚移動ロボット) が導入された [4]．このような災害用ロボットは移動環境や目的によって使い分けられる．比較的狭い通路での探索活動を行う際には小型のロボットが必要とされるし，複雑な瓦礫地帯で

あれば，移動面からの転倒を防げるものが望ましい．また，ロボットの運用には各々が有する技術的な課題も加味される．その一例が未知環境を移動するための環境認識・自己位置推定や耐故障性である．GPS を使用した自己位置推定は屋外であれば，精度良く使用することもできるが，屋内を移動する場合には SLAM のような自己位置推定技術が必要不可欠となる．故障については，常にロボットのセンサ類・アクチュエータ等の至る所で故障する恐れがある．あらゆる故障を網羅的に克服することはできないが，現実的な運用のためには部分的でも耐故障制御を導入する必要がある．たとえば，多脚移動ロボットであれば，脚一本が欠損したときに他の脚で歩行できるようにする等の手段が考えられる．さらに，このような多脚移動ロボットは多自由度構造を利用した姿勢制御による転倒回避が可能であるし，優れた不整地踏破能力を備えている．いっぽうで，冗長多自由度の脚構造を持つことから他の車輪型ロボットよりも制御器設計における複雑な課題が残されている．そこで本研究では上記の制御器設計の対象として多脚移動ロボットを使うことで，より高い不整地踏破能力を発揮する多脚移動ロボットの開発を行う．

1.1.1　多脚移動ロボット

多脚移動ロボットを使うメリット・デメリットがその用途や使用環境を基に車輪駆動・クローラ駆動ロボットと比較して述べられることがある．たとえば水平面・凹凸の少ない床などでの移動をさせる場合，車輪駆動の方が移動速度もエネルギー効率も良い．しかし，ある程度複雑な不整地になると車輪移動でのエネルギー効率が悪くなるだけでなく，移動環境を走破できない状況が生じる．さらに，車輪駆動で脚のような可変機構をロボットに搭載していないものについては姿勢制御をすることも困難になる．複雑な不整地環境での運用には，姿勢制御のための機構が必要となる．このことを考えた場合，姿勢を保ちつつ複雑な不整地環境上を移動する用途では，脚移動ロボットのようにロボット自身と移動環境との接触点 (脚先) の相対位置を自在に変えることのできる機構が必要となる．そのためには，状況に応じて脚先を 3 次元空間で自在に動かす必要があり，各脚に最低 3 自由度以上の機構が必要になる．さらに，6 脚を含む多脚ロボットの場合は，姿勢制御だけでなく状況に応じた歩容・歩行方法の切り替えや 1 脚が故障しても他の脚で静歩行を維持しつつ歩行させることも構造上可能である [5]．このことから本研究では各脚に 3 自由度を有する 6 脚移動ロボットを対象とする．

6 脚移動ロボットを含む多脚移動ロボットは国内外において研究・教育資源として頻繁に開発されている．しかしながら，その多自由度機構の制御の難しさから複雑な不整地上で人の作業を代替できるロボットの開発例は少ない．本研究のような各脚に 3 自由度以上を有し，運動計画に従って移動する 6 脚ロボットの最近の研究例としては，国内では新井

らによるリムメカニズムロボット ASTERISK[6]，藤田らの 6 脚クローラ型ロボット [7]，ホイールスパイダから着想を得た可変構造 6 脚ロボット [8]，国外では Weaver（オーストラリア連邦科学産業研究機構）[9]，Hector（Bielefeld University，ドイツ）[10]，LAURON（FZI Research Center，ドイツ）[11]，Ragno（Poznan University of Technology，ポーランド）[12] などが挙げられる (Fig. 1.1)．このように脚移動ロボットの開発は，世界各地で行われており，実際に不整地探索や一部の産業に用いられた事例もある．たとえば，上体にクレーン機構を有する The Walking Forest Machine は，フィンランドで実際の林業に用いられた [13]．Carnegie Mellon 大学の Dante は火口付近の探索に使用された [14]．東日本大震災でも，福島第一原発のプラント内の探索活動用に東芝の 4 脚ロボットが使用された [15]．しかし，これらは一時的に利用されたものであり，継続的に運用されている事例は確認されていない．

1.1.2　不整地歩行を目的とした多脚移動ロボットの課題

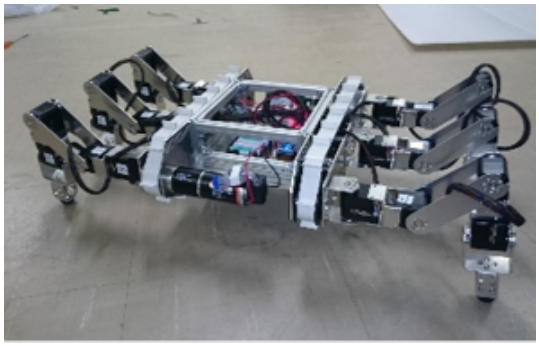
不整地上の移動や運搬作業を目的とした多脚移動ロボットには，姿勢制御のために各脚に 3 自由度以上を有する多自由度機構が必要であり，歩行環境に応じて各脚で任意の振舞い (脚を接地させる位置や遊脚の軌道等) を実行させる制御系が必要である．しかし，このようなロボットの実環境での継続的な運用は確認されていない．この原因として大きく分けて以下の 2 点が挙げられる．どちらも多脚ロボットの機構が冗長多自由度で複数脚を有することから生じる問題である．

1. 冗長多自由度機構の運動計画における計算コストの高さ
2. 冗長多自由度機構を制御するための制御器設計・制御パラメータのチューニングの難しさ

まず 1 つ目について，脚移動ロボットの運動計画の計算コストは，自由度・脚数に対して指数関数的に増加する．たとえば，ある多脚移動ロボットに関して各脚の状態が遊脚 (脚を上げる)，接地 (脚を地面につく) の 2 通りで与えられた場合，ロボット全体で捉えたときの遊脚・接地の組み合わせは，脚数が 4 脚のとき $2^4 = 16$ 通り，6 脚のとき $2^6 = 1024$ 通り，20 脚であれば $2^{20} = 1048576$ 通りと増大する．多脚移動ロボットの歩行制御および運動計画では各時刻でこの全ての状態の組み合わせの中から 1 つを選ばなければならないので，最先端のコンピュータを使う場合でも脚数が増加する毎に計算が困難になる．さらに，歩行制御法によっては，制御上で定義されている状態が上記の遊脚・接地の 2 通りより多い場合もあるし，各脚の自由度も増えるほど計算が複雑になる．

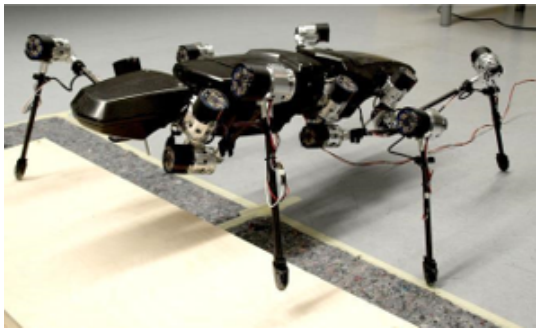
また，こういった制御上の状態 (遊脚・接地等) の選択以外にも姿勢制御に関する計算の問題もある．たとえば，上記で挙げた LAURON 等の 6 脚ロボットでは，各脚に 3 自由度以上の機構を持っており，少なくとも計 18 以上の多自由度関節を有している．こういった冗長多自由度機構の場合，脚先の位置をグローバルに固定したときにロボットが取り得る姿勢は一意でないため，ロボットの制御には単純に脚先の座標を決めるだけでなく，脚同士の相対位置とロボット自体の姿勢も考慮した手法が必要とされる．そのため，多脚移

(a)ASTERISK[6]



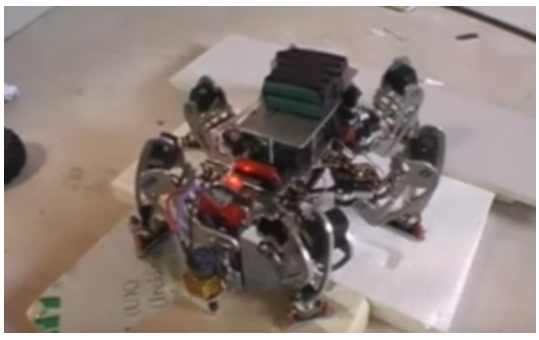
(b)Six legged crawler robot[7]

(c)Weaver[9]



(d)Hector[10]

(e)LAURON[11]



(f)Ragno[12]

Fig. 1.1　Example of six legged robot

動ロボットの運動計画や局所制御を集中制御的に行おうとすると自由度の数に応じて膨大な計算コストを必要とする．

次に 2 つ目の問題について説明する．たとえば，姿勢制御を加える場合でも，追加した制御に伴う脚の運動はそれぞれ異なる．姿勢制御による運動を実現するためには，制御器設計の段階で脚の運動に影響する制御則上のパラメータや制約をあらかじめ決定しておく必要がある．しかし，こういった脚の運動を決めるパラメータは脚数や自由度に応じて膨大になり，パラメータのチューニングが困難になる．この 2 つのデメリットにより，現状多脚移動ロボットの実用例は少ない．

1.1.3　多脚移動ロボットの歩行制御に関する研究例

脚移動ロボットの歩行制御に関する研究では，現在 Free gait に基づく最適化手法による歩行制御が数多く提案されている [16]〜[20]．これらの研究では，ロボットの姿勢・各関節角度等の状態だけでなくセンシングによる周囲環境も含めた運動計画が行われるため，前述したように計算コストが高くなる．そのため，最適化計算や脚の着き方・歩容に関して制約を与えている研究例が多く見られる．それでも自由度に応じて計算コストが増すことには変わりないので，制御対象としては多脚移動ロボットより自由度を抑えられる 4 脚移動ロボットを用いた研究例の方が多い [21]〜[27]．

いっぽうで 2 脚・6 脚・多脚ロボットを含めて中枢神経パターン生成器 (Central pattern generator：CPG) と呼ばれる手法も歩行制御に用いられている [28]〜[35]．CPG は生物の神経構造を模擬したニューラルネットワークにより動物の周期運動・歩容を再現する手法である．この CPG における制御のパラメータとは主にニューラルネットワークのパラメータ (各ニューロンの重み・バイアス等) であるため，これを同定する手法も提案されている．たとえば物理エンジン上での試行動作を基にした強化学習 [36]〜[39] や遺伝的アルゴリズム [40]〜[50] による方法がある．しかし，これらのパラメータは特定の条件下で決まったものであるため，学習時と異なる環境での歩行に使用できる保証はない．そこで，パラメータ同定の他にそれぞれの脚の位相を調整することで，歩行を維持する手法も提案されている [51]〜[56]．このように事前に決定したニューラルネットワークや歩容のパラメータでは対応できない状況を想定した局所制御の手法も提案されている．他にも移動環境に合わせて，遊脚相・接地相のデューティ比を変化させ，歩容を変化させたり [57]，一歩当たりのエネルギー消費を最小化するように歩行周期を更新したりする手法 [58] 等が提案されている．

また，ロボットの機構に基づいたアプローチとして，各脚を 1 自由度の板バネ構造 (Spring Loaded Inverted Pendulum (SLIP)) にした RHex[59] や歯車状の車輪脚を有す

る Whegs[60] などが提案されており，自由度を減らすことにより，前節の運動計画の計算コスト・制御器設計の難しさの問題を解決している．しかしながら，自由度が低いために姿勢制御ができず，運搬作業等には向かない等のデメリットが見られる．

1.1.4　多脚移動ロボットに対する分散システムの適用

以上で取り上げた多脚移動ロボットの歩行制御法の中でも，運動計画の計算コストを考慮する場合には，集中型の制御は避けた方が良い．運動計画の計算コストの問題に限らず，一極集中的に，ロボット全体の運動計画に沿って各関節の動きを一意に決めるためには，自由度の数に対応する拘束条件が必要になる．前述の Free gait に基づく最適化手法 [16]〜[20] の研究でも実装例はほぼ 4 脚ロボット等の自由度が抑えられたものに限定されている．これを 6 脚，あるいはさらに多脚・多自由度のロボットで制御に使うことは難しい．いっぽうで，この問題は冗長多自由度機構で，各脚の運動計画に関する解が一意に定まらないことから起きているため，各脚に関する解が一意に決まる制御手法であれば，逆に解決することができる．その手段としては Fig. 1.2 のようにロボットの制御系を局所的に分割し，分割されたサブシステム内における個々の運動計画の解を導くことで，全体のシステムとして目標とする制御が行われれば，少ない計算コストで現実的なロボットの運用が可能になる．このような制御手法は分散システムの分野で研究されており，大規模なシステムの制御に適用することが望ましい．このサブシステムは多脚移動ロボットの各脚に定義される．このサブシステムに分散型の制御器を定義し，サブシステム同士の通信・協調により，大域的な情報を使うことなく大規模システムを制御が可能である．前述した CPG による歩行研究もこの分散システムに含まれる手法である．次節から分散制御を使った歩行制御法に絞って，不整地移動に適した多脚移動ロボットの歩行制御法を選択する．

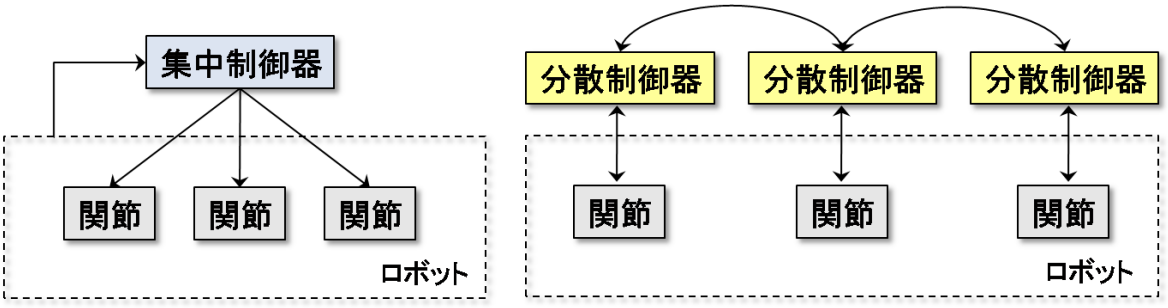


Fig. 1.2　Centralize and decentralized system

1.1.5　接地点追従戦略

多脚移動ロボットの研究では，上記の分散システムの導入に加えて，個々の脚における運動計画を削減する手法が提案されている．その 1 つが Follow-the-leader gait(FTL 歩容) [61] と呼ばれる歩容で，先頭脚を除く各脚を前脚が接地した地点 (接地点) またはその付近に接地させる手法である．これによって，ロボットが環境情報や自分自身の状態を十分に把握できていない状況においても少ない情報のみで不整地歩行が可能である．そのため，ロボットにこの手法を実装する場合には，先頭脚についてのみ接地点探索をし，続く脚については，運動計画を省くことができる．つまり，先頭以外の運動計画が先頭に集約される．この歩容は，H.Cruise らの Walknet[62] においてもニューラルネットワークを使う事で実現されている．しかしながら，この手法では前後脚間での接地点の交換を一脚ずつしか行えないという問題があった．また，ロボットの後方へ接地点を受け渡す際に，胴体が移動すると正確に後続脚が接地点を引き継げないという問題があった．接地点を正確に引き継ぐために胴体を停止させる必要があり，脚数が増えるほどロボットの前進が原理的に遅くなることが原因である．

そこでこの FTL 歩容を，前後の脚同士で接地点を共有する時間を設ける歩容に変えた事象駆動型分散制御「接地点追従法 (Follow-the-Contact-Point(FCP) gait control)」が稲垣ら [63] により提案された．この手法では，Fig. 1.3 のように上述の FTL 歩容を参考に各々の脚が，接地・脚を接地点から離脱・脚先を一時停止・次の接地点へ接近の 4 つの制御モードを有する制御オートマトンにより，自律分散的に制御される．この手法では，前後脚同士での接地点共有の時間が必ず設けられるため，脚先が移動面を滑らない前提においては，胴体が推進している間も正確に接地点を後方へ引き継ぐことができる．本研究では，多脚移動ロボットの制御における 1 つの課題である歩行制御の計算コストを削減したこの接地点追従法を対象に研究を行う．

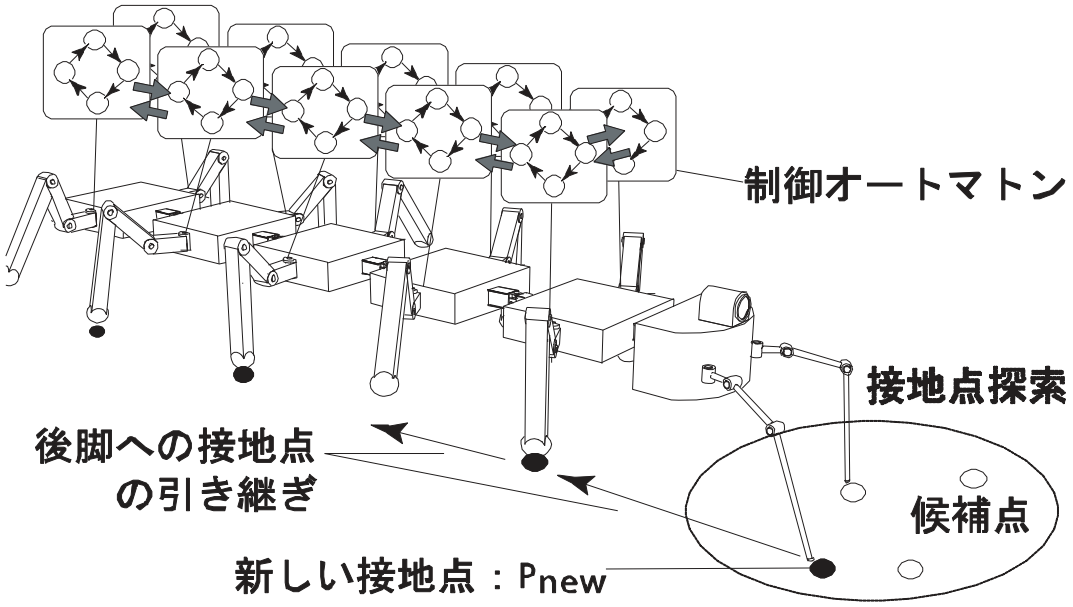


Fig. 1.3　Follow-the-Contact-Point gait control

1.1.6　歩行制御の従来研究における課題

前節の通り，数多くの多脚ロボットの歩行制御に関する研究がされてきたが，いずれの歩行制御法にも共通して，不整地で姿勢を維持するための姿勢制御や周囲環境を考慮した運動計画，接触センサを用いた局所制御等の複数の制御 (以下，制御要素と呼ぶ) が歩行制御法に付与され，ロボットの制御器が設計されている (Fig. 1.4)．ロボットの転倒回避能力や不整地踏破能力そのものに関してはこういった追加の制御要素によって向上できるだろう．

しかしながら，この制御要素の導入によって別の新たな問題が生じる．複数の制御要素を含むロボットの制御系は，当然ながらロボットの開発・実験環境に基づいて設計されたものであり，ロボットが異なる環境に置かれたときにこの制御系が想定通りに働かない可能性がある．たとえば，想定していなかった未知の不整地環境において，歩行制御法・運動計画・姿勢制御・局所制御等の制御要素同士が干渉し，デッドロックになる (歩行制御による歩行が止まる) ことも予想される．接地点追従法を例に挙げると，歩行を継続させるためには前脚の接地点を後続脚へ引き渡すために前方の脚先を後続脚の届く範囲へ運ぶ必要があるが，これが行えないような状況になった場合にデッドロックが起きる．このような状況は，制御要素として姿勢制御や接触センサを使った局所制御を制御系に導入した場合に，各脚の運動が環境やロボットの状況によって不規則に変わるため，十分起き得る．特に本研究では未知の不整地歩行を目的とした多脚移動ロボットを開発するため，この制御要素同士の統合を実現させ，ロボットの移動を継続させる新たな制御手法が必要となる．

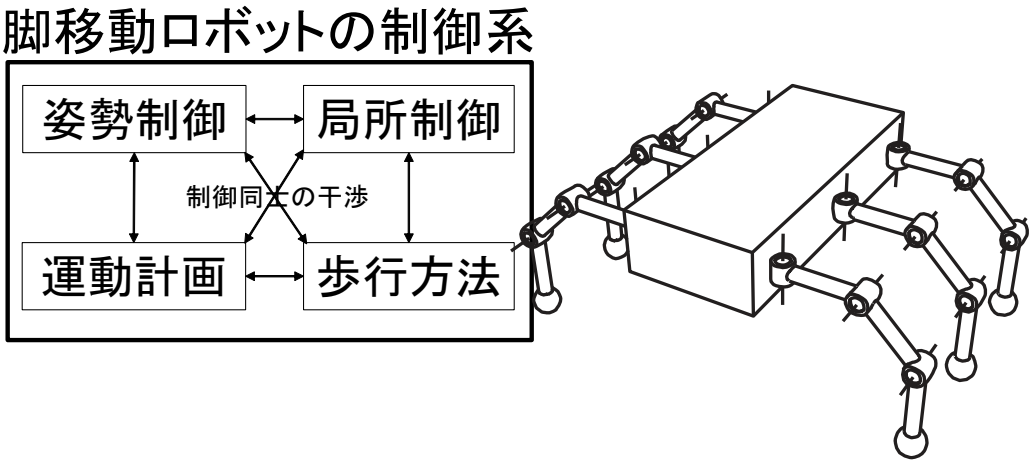


Fig. 1.4　Control system including multiple subsystems

1.1.7　形式的手法に基づく制御器設計

このようなシステムがデッドロックを回避するか否かを証明する手段として「モデル検査」と呼ばれる手法がある．モデル検査は数理モデルを対象として特定の仕様を満たすか否かを証明する手法であり，形式的手法というソフトウェア工学におけるシステムの開発・検証技術に含まれる．この手法を用いることで，歩行制御におけるデッドロックを回避するための制約が導出できると考えられる．本研究では上記の制御要素同士の干渉を防ぐ問題をこのデッドロックを回避するための制約充足問題を解くことで解決する．そのために，以降の各章において，Fig. 1.5 のように，歩行制御 (接地点追従法) 及び複数の制御要素を含む制御系によるロボットの動作をモデル検査用に抽象化し，この抽象化されたモデル (時間オートマトンと呼ばれる) を使って制約を導出する．その上で，導出された制約を保証する手法をシミュレーションまたは実機において実装するという段階を踏んでデッドロックの生じない制御系の実現をさせる．

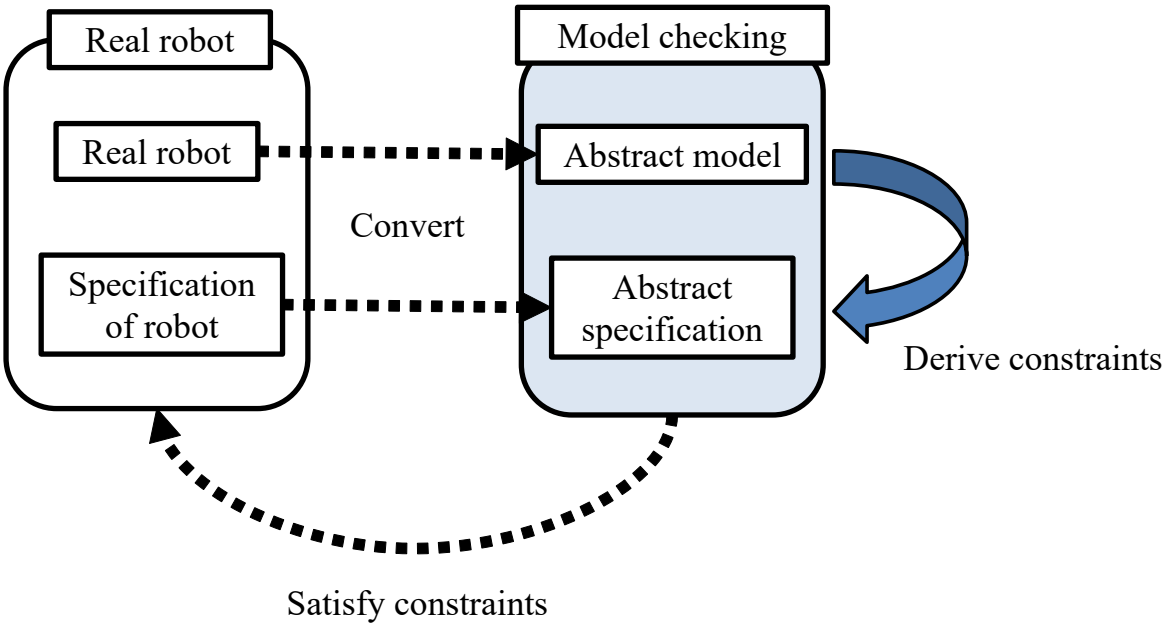


Fig. 1.5　Derivation of time constraints by using Model checking

1.1.8　制約充足の手法

本手法で導出される制約はロボットの動作に関する時間の集合である．この制約充足の方法として本研究ではまず，2 通りの方法を考案・施行した．

1. 制約を歩行制御のパラメータまたはロボットの物理的なパラメータ (ロボットの脚先座標や速度) の制約に対応させてから保証させる．
2. ロボットの状態を観測しつつ，制約を常に保証できるように制御する．

1 つ目の方法論は，前述の多脚移動ロボットの歩行制御におけるパラメータの同定方法に関する解決策として行ったものである．歩行制御法におけるパラメータ同定には，「歩行が継続すること (本研究ではデッドロックを起こさないこと)」を 1 つの評価や目標の動作仕様として扱う必要がある．そのため，パラメータが脚先座標の存在領域・速度等の物理的なパラメータであっても，リンク系のみの問題として扱い，厳密にデッドロックを防ぐようなパラメータを導出することは難しい．そこで，この方法では Fig. 1.6 のようにあらかじめモデル検査で得られた制約を基にロボットの物理的なパラメータ (脚先座標の存在領域・速度等) を導出した上で，それを制御に用いる．ただし，Fig. 1.6 におけるモデル検査のプロセスは手法の詳細は 4 章にて説明するが，この方法の場合，ロボットの歩行中に制約そのものを保証しているかを確認する必要はないが，ロボットの振る舞いが限定されるという問題がある．本研究での不整地歩行を目的とした移動の場合，不整地環境に対応した幅広い運動が求められる状況がある．

そこで 2 つ目の方法は，制約をロボットの移動に間に常に保証できているかを観測し，保証できない場合に，制約を保証できるように制御介入を行うものである．これは制約充足問題を用いた制御である．制約充足問題を用いた制御はシーケンス制御・通信・生産・交通・ロボットの分野で数多く研究されている [64]〜[68]．中でも本研究で扱う接地点追従法のような事象駆動型制御・離散事象システムを対象としたスーパーバイザ制御 [64], [68], [69] の研究は，スーパーバイザ制御は制御対象の状態を観測者 (オートマトン) が観測し，観測された状態が制約を満たしているかを観測者が判定し，満たしていない場合または制約から外れる直前に制約を保証できるように制御介入を行う．5 章では，Fig. 1.7 のように，このスーパーバイザ制御から着想を得た Timekeeper 制御という手法を使って制約を保証させる．

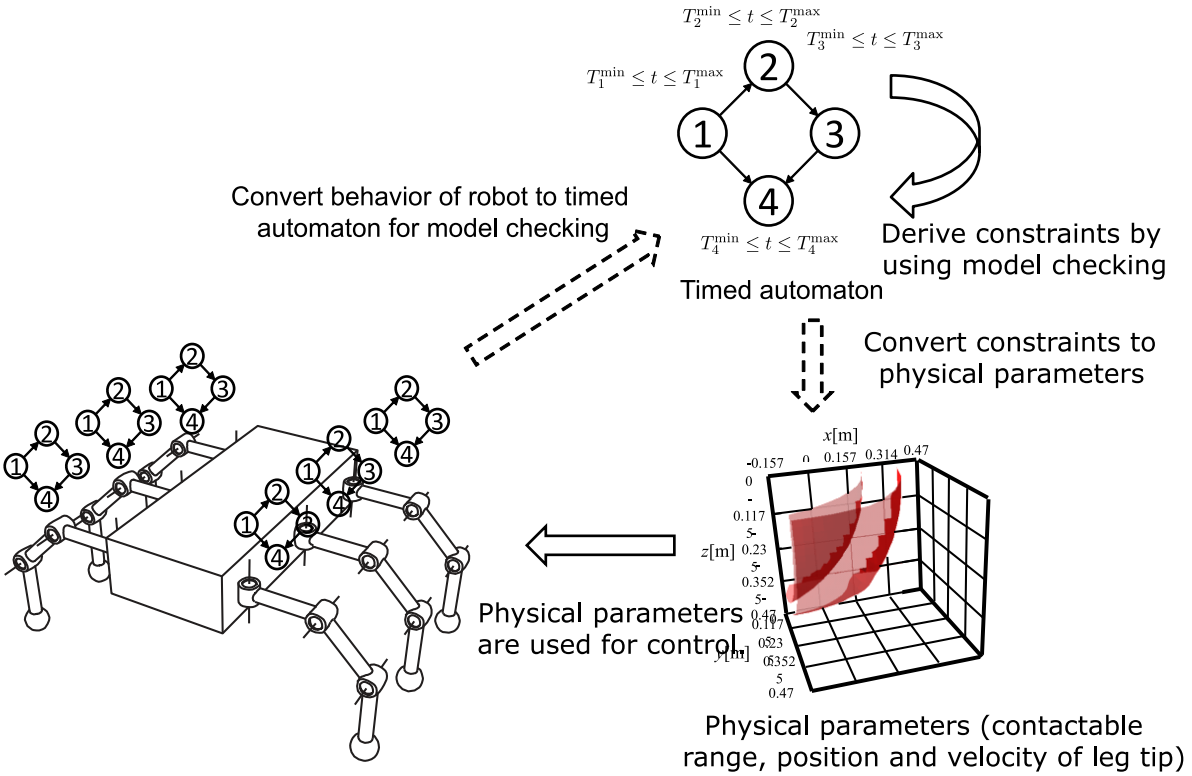


Fig. 1.6　1st method for constraints satisfaction problem

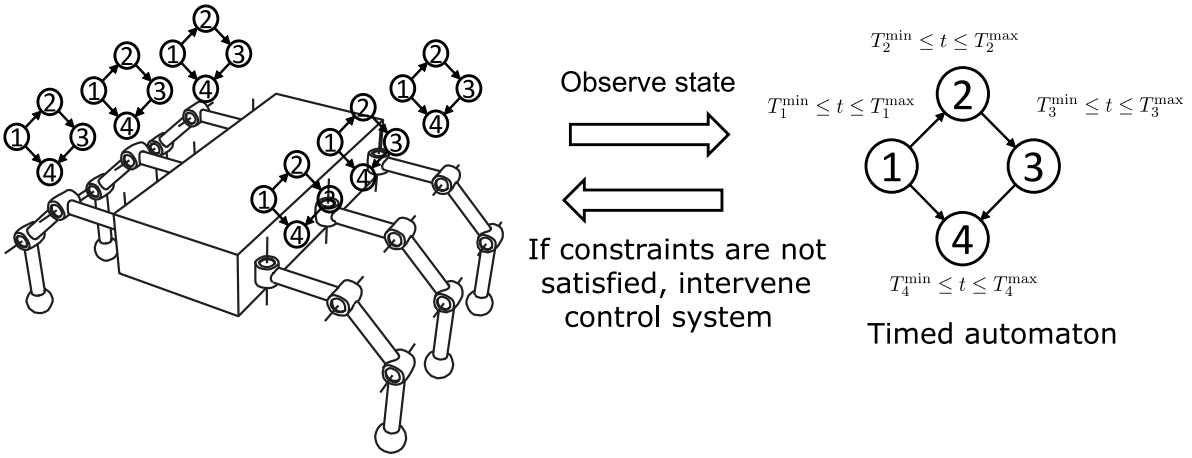


Fig. 1.7　2nd method for constraints satisfaction problem

1.2　本研究の目的

本研究の最終目標は，不整地歩行での実用が可能な多脚移動ロボットの開発であるが，その一歩として本研究の目的は，環境に依存しないロバストな制御器設計である．この意味は，ロボットが移動環境を変えても制御器に含まれる複数の制御要素 (歩行制御法・姿勢制御・運動計画・局所制御・・・) 同士の干渉を防ぐ手法を制御器に導入することである．これを実現するために本研究では 2 点を目的として設定する．

目的 1:複数の制御要素の導入

第一に，歩行制御において必要な複数の制御要素を導入する．この制御要素は転倒安定性に対応するもので，本論文では姿勢制御・脚の着き方・運動計画の手法を導入する．

目的 2:制御要素同士の干渉を防ぐ制約の導出

上記の制御要素を追加したことで，ロボットの転倒安定性は向上するが，いっぽうでこれらを含めた制御系をそのまま動かすと，制御要素同士の干渉 (歩行制御におけるデッドロック) が生じることがある．そこで，歩行制御を含めた制御要素同士の干渉を防ぐ制約の導出を行う必要がある．この制約を導出する上で，前述のモデル検査を用いる．これによって，デッドロックを必ず回避する制約が確認できる．そのために，この制御系によるロボットの振る舞いと目標仕様 (歩行し続ける・デッドロックを起こさない) をモデル検査用のモデル (時間オートマトン) に抽象化し，その上で，モデル検査を使った制約の導出方法を新たに提案する．また，制約充足の方法についても 2 種類の方法を提案している．

1.3　本論文の構成

本論文は 7 つの章により構成される．本論文の各章の流れを Fig. 1.8 に示す．2 章は本研究で用いる 6 脚用の接地点追従法の元となった稲垣ら [63] が提案したムカデ型ロボット用の接地点追従法に関する説明を行う．4〜6 章で後述するムカデ型ロボットの接地点追従法はこれを基に改良したものであるため，各章でどのような改良を施したかを 1 つずつ提示する．しかしながら，接地点追従法の概要として各脚に割り振られた制御オートマトンの概要については後述する 6 脚用の接地点追従法と共通するため，詳細に説明する．

次に 3 章では本研究においては制御器設計をする上での制約を導出するために用いるモデル検査の概要を説明する．モデル検査で扱う数理モデルは時間オートマトンと呼ばれる，接地点追従法に用いられる制御オートマトンとは異なるものであり，ロボットの制御則ではなく，制御に対応した振る舞いを時間で表現したものを扱う．過去の研究事例とし

てもハードウェアのデッドロック検査のために用いられた例がある [70]〜[74]．この章では制約を導出する段階ではなく，モデル検査そのものの説明をする．

次に 4 章では，モデル検査を使って得られた制約を基に脚先の接地可能領域を導出し，かつそれをロボットの歩行に利用することで間接的に制約を保証させる研究を説明する．ここではモデル検査を使って制御器における干渉を防ぐ制約を導出する手法を提案する．前述のハードウェアのデッドロック検査に用いられるケースは複数確認されているものの，今回のように制約を導出する目的で用いられるケースは確認されていない．そこで，一旦多脚ロボット・接地点追従法以外にも利用できる一般化した形で制約を導出するアルゴリズムを提示し，それで得られた制約をロボットの制御に取り入れる．なお，ここでの制約の導入方法は，制約を保証できるような脚の接地可能領域を算出し，それをロボットの制御に用いることで制約を間接的に保証させる手法を取っている．4 章末尾にはシミュレーションによりこの手法の効果の欠点についても述べる．

5 章では，4 章で導出した制約の充足方法を変えた研究を述べる．ここでは，時間制約を直接保証するために制約充足型統合制御「Timekeeper 制御」を提案する．ここで用いる時間制約は各状態 (脚を上げる・下げる・接地動作等) の滞在時間の集合になる．この時間制約には，滞在時間の最大値・最小値に規格化係数が掛かっているという特徴がある．この規格化係数の値が自在に変更可能であることを利用して，Timekeeper 制御ではスーパーバイザ制御を参考にロボットの各状態の滞在時間を観測し，滞在時間が時間制約を破りかけたときにこの規格化係数と滞在時間を調整するように制御介入を行うことで，時間制約を常に保証させる手法である．これによって，4 章の方法より制御器における制御要素同士の干渉を直接防ぐことができると考えられる．5 章末尾にはシミュレーションと実機実験両方の結果を表示し，どちらにおいても Timekeeper 制御の効果を確認する．

6 章では，5 章の研究における 1 つの問題点に取り組んだ研究を示す．それは 4，5 章においては全ての脚に共通の時間制約を設けていたことである．4，5 章では全ての脚に共通の時間制約を与えたことが原因で，ロボットの脚の動きを実際より限定していたのである．共通の時間制約を与えたことにより，稲垣らが提案した接地点追従法ではデッドロックの保証に加えて静歩行が維持できる時間制約が探索できなかったのである．この章では時間制約を脚毎に個別に定義することで，姿勢制御等の制御要素の追加はされているもののロボットの振る舞いとしては稲垣らが提案した接地点追従法に準拠したモデル (時間オートマトン) でデッドロックの回避と静歩行が維持できる時間制約が確認されたので，その導出手順及び Timekeeper 制御の改善について述べる．この改善点を述べた上で，最後に実機実験により 5 章で使った接地点追従法と 6 章で使う接地点追従法によるロボットの振る舞い・歩容の比較をし，時間制約の与え方による変化を述べる．7 章では，この研究の結論を述べる．

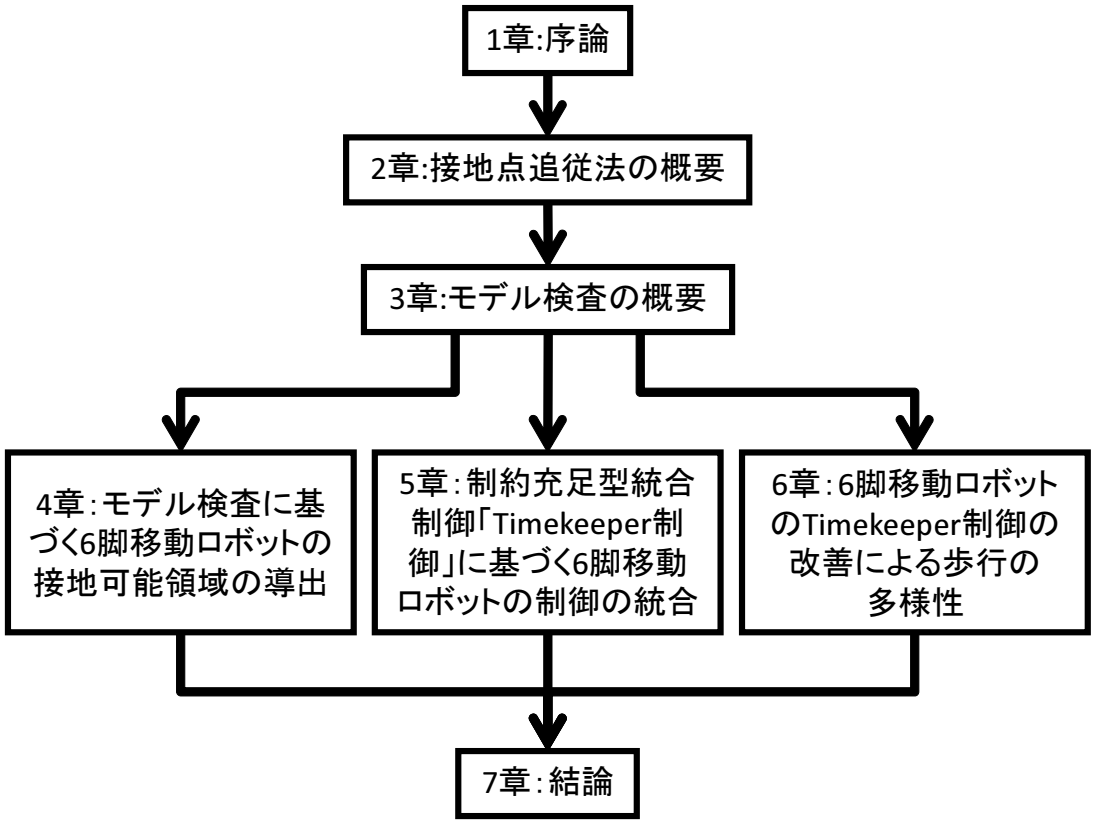


Fig. 1.8　Outline of this thesis

第 2 章

接地点追従法の概要

2.1　はじめに

本章では稲垣らが提案した接地点追従法を説明する．以降の章で使用される接地点追従法はこの章で説明されるものから 6 脚ロボット用に改良されているため，各章毎にどういった改良が施されたか, あるいは 1 章で触れた姿勢制御・運動計画等の複数の制御要素をどのように取り込んだかを後の章で 1 つずつ説明する．そのため，2.1 節では接地点追従法を説明するために必要な変数の定義を行い，2.2 節で稲垣らが提案したムカデ型ロボット用の接地点追従法の説明をする．2.3 節でこの初期の接地点追従法を基に行われたこれまでの先行研究とそれに対する本研究の立ち位置について述べる．2.4 節で本研究での接地点追従法の 6 脚ロボット用の改良を説明する．

2.2　セグメントと制御域の定義

接地点追従法は本来，Fig. 2.1 のような複数の体節と左右の脚，および前方の体節とつながるリンク (体節間リンク) を備えたセグメントを 3 つ以上前後につなげたムカデ型ロボットで提案された手法である．ここではセグメントを N 個連ねたムカデ型ロボットを対象として説明する．セグメント $j \in \{1, \cdots, N\}$ の左側を 0, と右側を 1 としたとき，それぞれの脚を $\mathrm{Leg}[i][j](i = 0, 1, j = 1, 2, 3)$ と表す．このとき，$\mathrm{Leg}[i][j]$ の左右反対側に相当する脚は $\mathrm{Leg}[1 - i][j]$ となる．また，以降の説明で述べる前脚とは $\mathrm{Leg}[i][j]$ に対する $\mathrm{Leg}[i][j - 1]$，後脚は $\mathrm{Leg}[i][j]$ に対する $\mathrm{Leg}[i][j + 1]$ を指す．そして，$\mathrm{Leg}[i][j]$ の付け根に原点を持つ直交座標座標系を $\Sigma_{i,j}$，脚先の可到達域を $\Omega_{i,j}$ と定義する．ただし，ここで述べる脚先とは $\mathrm{Leg}[i][j](i = 0, 1, j = 1, 2, 3)$ の 3 リンク目の末端，エンドポイントを指す．本稿で用いる脚先座標は $\Sigma_{i,j}$ 上のものとする．また，脚先の位置を

$P[i][j] = (x_{i,j}, y_{i,j}, z_{i,j})^T \in \Omega_{i,j}$ とする．さらに，座標系 $\Sigma_{i,j}$ において，Fig. 2.1 のように可到達域の中に制御域 $A_{i,j}$ $(\subseteq \Omega_{i,j})$ を定義する．制御域 $A_{i,j}$ は各脚の座標系 $\Sigma_{i,j}$ に固定された直方体型の領域であり，グローバル座標上ではロボットの進行にあわせて移動する．この $A_{i,j}$ には前後の領域 $A_{i,j-1}$，$A_{i,j+1}$ と重なる領域が存在する．この重なりを基準に $A_{i,j}$ を以下の 3 つの領域に分けて制御に使う．

Area 1: 制御域 $A_{i,j-1}$ と重なる領域；$A_{i,j} \cap A_{i,j-1}$,

Area 2: 前後の脚の制御域と重ならない領域,

Area 3: 制御域 $A_{i,j+1}$ と重なる領域：$A_{i,j} \cap A_{i,j+1}$.

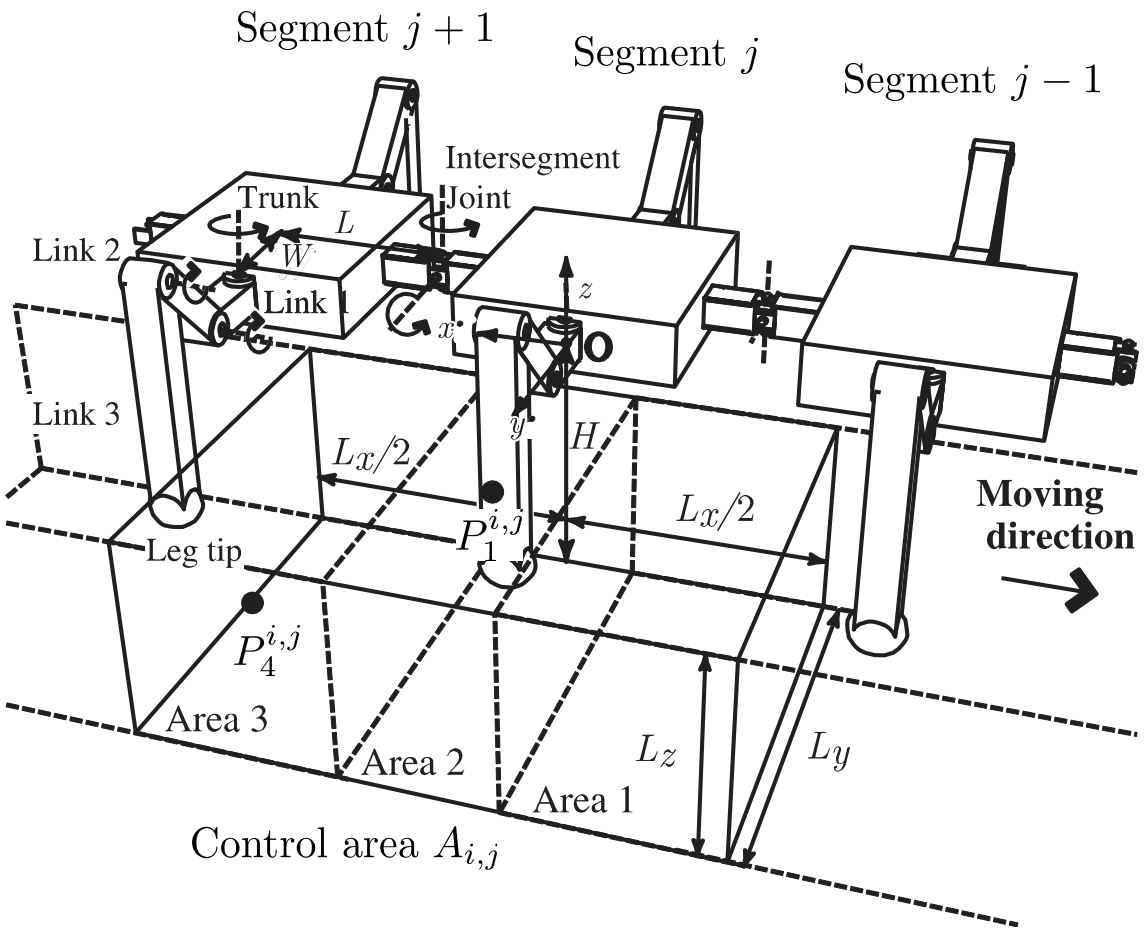


Fig. 2.1　Connection of segments and control area

2.3　接地点追従法：各脚の制御モード

以下にこの接地点追従法 [63] の制御を説明をする．ただし，本論文の 4 章以降では体節間関節を持つロボットを使わないため，ここでは，体節間関節の説明を省き，脚の制御を説明する．この手法では，各脚に定義した制御オートマトン (Fig. 2.2) を用いる．制御モードを用い，モード 1,2,3,4,1…と順番に遷移を繰り返す．以下に各制御モードを説明する．また，脚先の目標座標を $P^{i,j}$，モード 2 以外の各制御モードの終了時に到達する目標点をそれぞれ $P_1^{i,j}$，$P_3^{i,j}$，$P_4^{i,j}$ とし，モード i における脚先の移動速度指令値を v_i，制御周期を Δt, 各脚 $\mathrm{Leg}[i][j]$ の制御モードを $M^{i,j}$ とする．

制御モード 1

脚先を地面から離し，$\Sigma_{i,j}$ に定義される空中の中間点 $P_1^{i,j}$ に向かって速度指令 v_1 で動かす．この時，脚先の目標座標 $P^{i,j}$ は毎ステップ (2.1) 式で更新される．この式の $v_1\Delta t$ は 1 ステップあたりの $P^{i,j}$ の変化量の絶対値を表す．この絶対値に目標座標 $P^{i,j}$ から中間点 $P_1^{i,j}$ の方向の単位ベクトル $\frac{P_1^{i,j}-P^{i,j}}{|P_1^{i,j}-P^{i,j}|}$ をかけることで，脚先の目標座標 $P^{i,j}$ を $P_1^{i,j}$ に向けて等速直線運動で追従させる．そして，脚先が中間点 $P_1^{i,j}$ に収束したら，制御モー

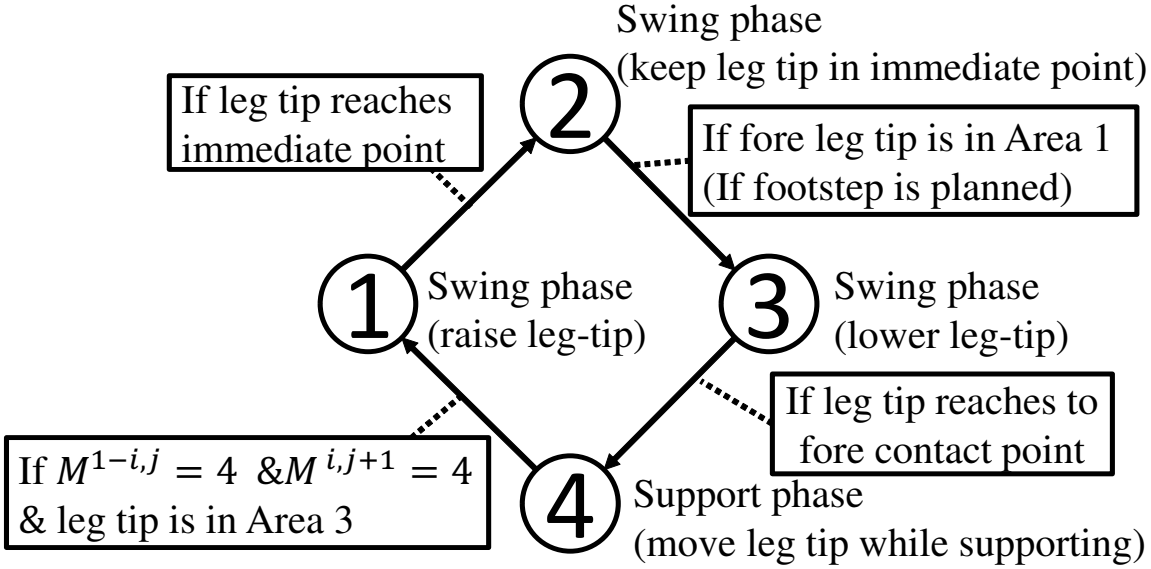


Fig. 2.2　Control automaton of each leg

ド 2 へ遷移する．ただし，← は毎ステップごとの更新を指す．

$$P^{i,j} \leftarrow \frac{P_1^{i,j} - P^{i,j}}{|P_1^{i,j} - P^{i,j}|} v_1 \Delta t + P^{i,j} \tag{2.1}$$

また，遷移条件は次式で記述される．

$$|P_1^{i,j} - P^{i,j}| < \epsilon \tag{2.2}$$

制御モード 2

速度指令を $v_2 = 0$ とし，脚先を中間点 $P_1^{i,j}$ に待機させる．同側前方脚 $\mathrm{Leg}[i][j-1]$ の脚先が自脚 $\mathrm{Leg}[i][j]$ の制御域における Area 1 に侵入したら，制御モード 3 に遷移する．先頭脚 $\mathrm{Leg}[i][1], i \in \{0, 1\}$ では，新しい接地目標点として設定された位置 $P_3^{i,j}$ が自脚の制御域における Area 1 に侵入したら制御モード 3 に遷移する．つまり，このモード遷移条件は次式で記述される．

ただし先頭脚以外の場合，この $P_3^{i,j}$ は前方の $\mathrm{Leg}[1-i][j]$ の脚先の位置に対し，脚先の太さ分だけ後方にずらした点である．先頭脚 $\mathrm{Leg}[i][1], i \in \{0, 1\}$ の場合，$P_3^{i,j}$ は制御モード 2 で新しく計画された接地目標点である．

$$P_3^{i,j} \in (A_{i,j} \cap A_{i,j-1}) \tag{2.3}$$

制御モード 3

脚先を新たな接地目標点 $P_3^{i,j}$ に向かって速度指令値 v_3 で動かす．$\Sigma_{i,j}$ 上では目標点 $P_3^{i,j}$ はロボットの進行と共に移動する．先頭脚以外では，前方の脚 $\mathrm{Leg}[i][j-1]$ から接地目標点 $P_3^{i,j}$ を常に獲得すると共に，先頭脚については計画した目標点 $P_3^{i,j}$ にセンサを用いて常に追跡する．この時，脚先の目標座標 $P^{i,j}$ は毎ステップ以下の式で更新される．脚先が $P_3^{i,j}$ に収束したら（接地点を受け継いだら），制御モード 4 に遷移する．

$$P^{i,j} \leftarrow \frac{P_3^{i,j} - P^{i,j}}{|P_3^{i,j} - P^{i,j}|} v_3 \Delta t + P^{i,j} \tag{2.4}$$

また，遷移条件は次式で記述される．

$$|P_3^{i,j} - P^{i,j}| < \epsilon \tag{2.5}$$

制御モード 4

脚先を接地させながら体節を推進させる．このとき脚先が地面を滑らないと仮定し，脚先位置をグローバル座標で固定する．いっぽう，$\Sigma_{i,j}$ 上で脚先を捉えると，接地点は胴体の推進に対して逆方向に移動する．このときの脚先座標の更新は毎ステップ次式で更新される．ただし，ここでの $P_4^{i,j}$ は各脚の限界目標点である．

$$P^{i,j} \leftarrow \frac{P_4^{i,j} - P^{i,j}}{|P_4^{i,j} - P^{i,j}|} v_4 \Delta t + P^{i,j} \tag{2.6}$$

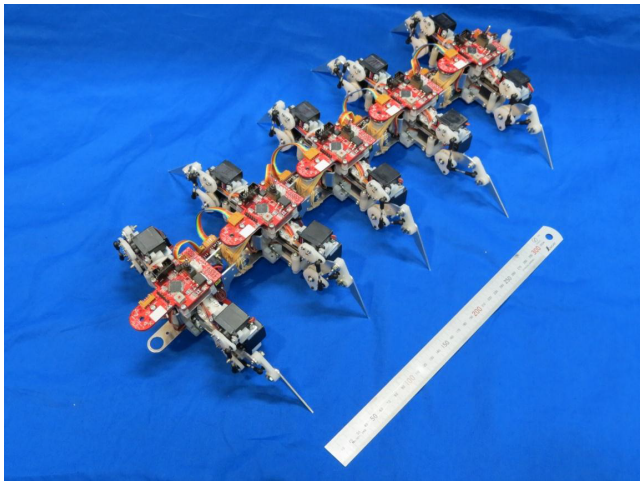
ここで，制御モード 1 への遷移条件の 1 つは脚先が Area3 に侵入していることである．2 つ目の遷移条件は他の脚との同期条件で，最尾脚か否かによって条件が異なる．最尾脚以外の場合 ($j \neq N$)，後方の $\mathrm{Leg}[i][j+1]$ が自脚 $\mathrm{Leg}[i][j]$ の接地点を受け継ぎ，かつ，反対脚 $\mathrm{Leg}[1-i][j]$ が接地しているときである．最尾脚（$j = N$）の場合は，後ろの脚がないので，反対脚 $\mathrm{Leg}[1-i][j]$ が接地していれば遷移する．つまり，遷移条件は次のように記述される．

$$\begin{cases} j \neq N: & P^{i,j} \in (A_{i,j} \cap A_{i,j+1}) \ \& \ M^{1-i,j} = 4 \ \ \& \ \ M^{i,j+1} = 4 \\ j = N: & M^{1-i,j} = 4 \end{cases} \tag{2.7}$$

2.4　接地点追従法に関する先行研究

最初に接地点追従法が提案されてから，不整地踏破のためのロボットの機構に関する課題，運動計画に関する課題，制御パラメータの設計に関する課題等への取り組みがされた．まず，丹羽ら [75], [76] により Follow-the-Leader gait(FTL 歩容) を基に事象駆動型分散制御を考案する研究から始まった．その後，高橋ら [77] によって，受動体節間関節を有するムカデ型多脚移動ロボットを対象に実装された．この時も新しく Fig. 2.3(a) のような実機が開発された．次に黄ら [78] によって，本研究でも扱う形式検証を用いた接地点追従法の動作検証を行う研究がされた．この研究では，ロボットの構造や制御に関する物理パラメータによっては接地点追従法においてデッドロックが生じることが確認されている．近藤ら [79] は接地点追従法に脚毎の反射制御を組み込むことで不整地歩行における脚の引っ掛かりの解消を試みた．次に野村ら [80] が形式検証を使って，接地点追従法による歩行が成立するための制御パラメータの導出の研究が行われた．岸ら [81], [82] はムカデ型ロボットの先頭脚の接地点計画に関して，触覚センサを使った手法と深度センサを使った手法の 2 通りで取り組んだ．この岸らの研究時にムカデ型ロボットだけでなく，構造の異なる 6 脚ロボット (Fig. 2.3(b)) でも接地点追従法の実装が可能であるかの確認が行われた．このとき，6 脚ロボットに接地点追従法を実装する際の課題が明らかになった．本研究は岸，野村らの研究における課題解決から始まる．

後に出島ら [83] は能動体節間関節のムカデ型ロボットにおける接地点座標系およびスプライン曲線を用いた目標軌道の生成に関する研究，鈴木ら [84], [85] は深度センサを用いた接地探索の手法としてヒューマンインザループ制御による遠隔操作と A*アルゴリズムによる自動探索の 2 通りの研究を本研究と並行して行った．2019 年現在は木俣ら [86] による平行リンク機構型 3 自由度体節間関節を有するムカデ型ロボットの開発がされている．藤井ら [87] は本研究の内容 (5 章の結果) を前提として脚先の反射制御の実装を，多良ら [88] は，本研究の 5・6 章の手法をムカデ型ロボットに実装する研究を行い，ムカデ型ロボットにおける頑強な制御器設計を目指している．本研究は，不整地においてもデッドロックを行さない頑強な制御器設計の開発を目的としてるため，前任者らが提案した運動計画・反射制御・ロボットの構造を接地点追従法の制御器に統合する上で必要な研究となる．



(a)Centipede robot developed by Takahashi

(b)Hexapod robot developed by Kishi and Kurita

Fig. 2.3　Multi-legged robot developed in previous reseaches

2.5　4 章以降における接地点追従法の改良

前述の前任者らの研究とは異なり，本研究を進める上で，必要となった接地点追従法の改良点を以下に列挙する．まず，接地点追従法はムカデ型ロボット用に提案されたものなので，本研究で扱う 6 脚ロボット用の改良を施す必要がある．各章において，ムカデ型ロボットとは異なる機構 (胴体の形状・脚の配置) での研究をしている．これはロボットの機構を変更しても提案手法が実装可能であることを確かめるためである．機構の変更により，各章毎に前後の脚同士の可到達域の重なり ($\Omega_{i,j} \cap \Omega_{i,j+1}$) も変わる．そのため，各章で制御域の定義が変わる．

4 章では，そもそも制御域 $A_{i,j}$ の一部である接地可能領域を設計することを研究内容として扱う．さらに姿勢制御の導入と遊脚条件，簡素な接地点計画の手法を追加することで，6 脚ロボットにおける接地点追従法の基礎を構築する．5 章では，4 章で設定した接地点計画をヒューマンインザループ制御による人の操作によるものに変更する．6 章では，4・5 章で与えた遊脚条件がロボットの歩容を限定していることからロボットの歩容を限定しないものを提案する．

- 4 章における接地点追従法の改良
  - ロボットの構造の変更.
  - 制御域 $A_{i,j}$ の拡張
  - 姿勢制御の導入.
  - 6 脚ロボットの静歩行を維持するための遊脚条件 (制御モード $4 \rightarrow 1$ の遷移条件) の追加.
  - 先頭脚の接地点計画の導入.
- 5 章における接地点追従法の改良
  - ロボットの構造の変更.
  - 制御域 $A_{i,j}$ の定義の変更.
  - 姿勢制御の変更.
  - 6 脚ロボットの静歩行を維持するための遊脚条件 (制御モード $4 \rightarrow 1$ の遷移条件) の追加 (4 章と同様).
  - 先頭脚の接地点計画をヒューマンインザループ制御による遠隔操作で代替.
- 6 章における接地点追従法の改良
  - ロボットの構造の変更.
  - 制御域 $A_{i,j}$ の定義の変更.

- 姿勢制御 (5 章と同様).
- 遊脚条件 (制御モード 4 → 1 の遷移条件) の変更.
- 先頭脚の接地点計画をヒューマンインザループ制御による遠隔操作で代替 (5 章と同様).

2.6　まとめ

本章では接地点追従法の概要を説明した．まず，基本的な接地点追従法における脚の運動を生成するための制御オートマトンと 4 章以降の説明で必要な変数を定義した．次に接地点追従法のみに限定したこれまでの先行研究について触れ，本研究の立ち位置・課題を提示した．さらに 2.5 節で 4 章以降に加える接地点追従法の改良点を列挙し，複数の制御要素を干渉させずに実装した制御器の開発をすることを示した．以降の章では各章で接地点追従法の改良点及び使用するロボットの機構を提示した上で，研究の内容を説明する．

第 3 章

モデル検査の概要

3.1　はじめに

本章では，接地点追従法のデッドロック検証に用いるモデル検査という手法の説明をする．まず，3.2 節では形式手法，3.3 節では形式検証，3.4 節ではモデル検査を順番に説明し，数々の形式手法の中でのモデル検査のフレームワークを述べる．そして，3.4.1 節ではモデル検査で使う抽象モデル (時間オートマトン)，3.4.2 節ではモデル検査器で扱う動作仕様の記述方法を述べ，3.4.3 節でこの 2 つを使ったモデル検査の仕組みを述べる．

3.2　形式手法 (Formal method)

形式手法は，数学的に厳密に意味づけられた言語 (「形式仕様記述言語」と呼ぶ) を用いてシステムの要求仕様や設計等を記述し，システムが要求仕様を論理的に推論する手法およびこれを用いたシステム開発・検証技術のことである．この手法を用いたシステム開発の対象は，ソフトウェアに限らずハードウェアで使用することもでき，交通・通信・航空を含む様々な分野での使用例がある [70]〜[74]．形式手法を用いる動機としては，以下のものが挙げられる．

- 数学的解析をするので，実験的・試行的な手段よりもシステム設計の信頼性を向上させることができる．
- システムを抽象化した上で開発・検証するのでコストが削減できる．
- 経済産業省「情報システム信頼性向上に関するガイドライン」[89] に従って厳密なシステム開発が求められる．

たとえば，Fig. 3.1 のようにシステムの要求仕様と設計仕様から直接システムを開発する際に，開発するシステムが本当に要求仕様を満たしているかをあらかじめ確認しなければならない場合がある．そこで要求システムと設計仕様を設計モデルに抽象化し，開発する前に設計の検証を行うことで，システムの動作検証と開発コストの削減を実現することができる．形式手法は形式言語・オートマタ理論・プログラム意味論・数理論学等の複数の理論計算機科学を応用した手法である．基準とする数学理論によって異なる多数の手法が存在し，それぞれ適用できる範囲や検証目的等の違いがある．形式手法は大きく分けて以下のものに分類される．

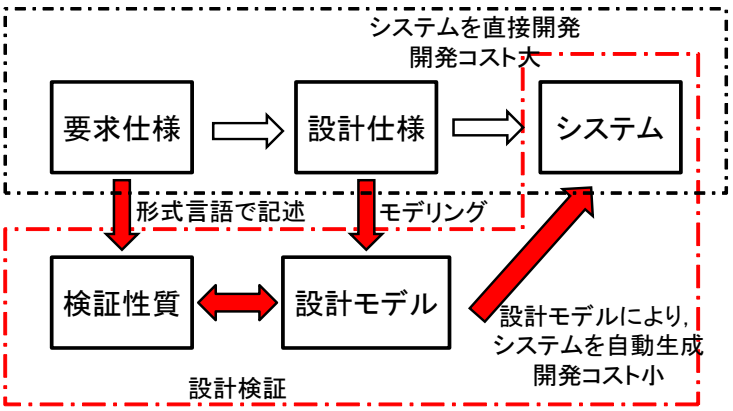


Fig. 3.1　Developing system by using formal method

形式仕様記述

システムの目標仕様を数学的な記述が可能である形式仕様記述言語を用いて記述する．システム自体を形式言語による抽象化をせずに検証する．この場合，システムは形式言語とは異なる数理モデルやシミュレーション上で表現される．1970 年代に欧州で誕生した手法で，後に「Z」「B」「OCL」「OBJ」「VDM」などの複数の対応アプリケーションが開発された．

形式検証 (Formal verification)

システムの目標仕様を詳細化することでシステムの開発，またはシステムの性質を証明する．たとえば，仕様記述からシステム開発を行うプロセスを詳細化と属性の証明を行う．高信頼のシステム開発を行う際に用いられる．この中には定理証明・モデル検査と呼ばれる手法が含まれる．

プログラム合成

システムの性質を自動的に検証する．自動定理証明により，完全に機械的な証明を行う．

このうち, ロボット等のシステム開発において，システムの振る舞いがモデル化可能な場合は形式検証を用いることで信頼性の高いシステム開発が可能となる．本論文では形式検証を使う．

3.3 形式検証 (Formal verification)

形式検証はソフトウェアおよびハードウェアのシステムにおいて，理論計算機科学を利用して，形式仕様記述で表現されたシステムの目標動作仕様を厳密に保証するか否かを証明する手法である．形式検証の適用例としては，組み合わせ回路，デジタル回路などのシステム，ソースコードで表現されるソフトウェアがある．これらのシステムの動作検証はシステムを抽象化した数理モデル上で行われる．この抽象化の前提として，数理モデルと実際のシステムの性質は一致しているものと考える．Fig. 3.2 のように形式検証自体も大きく 2 つに分類される [90].

モデル検査 (Model-checking)

Fig. 3.3 のように対象システムを状態遷移系によってモデル化し，検証したい動作仕様を時相理論で記述し，状態遷移系で想定し得る全ての状態を網羅的に検証し，時相理論で記述された仕様が満たされるかを検査する．自動で検証できるためソフトウェア設計で汎用されており，ポピュラーなアプリケーションとして「SPIN」「NuSMV」「UPPAAL」等が開発されている．いっぽうで，網羅的な検証を行う性質上，探索する状態数が収束しない場合 (状態爆発が起こる場合)，検査処理が終了しないというデメリットがある．

定理証明

システムを公理等の論理式の集合として記述し，公理と推論則に基づき対話的に証明を行う．証明の過程で人の操作が必要なケースがあり，モデル検査に比べて使用者は専門知識を必要とする．対応アプリケーションとして「B メソッド」「PVS」等が開発されている．

モデル検査は状態遷移モデルを対象としている事から，事象駆動型制御である接地点追従法に適用可能である．本論文では，モデル検査 (特にモデル検査用ツール UPPAAL) を使って，接地点追従法がデッドロックを回避するための制約を求める．

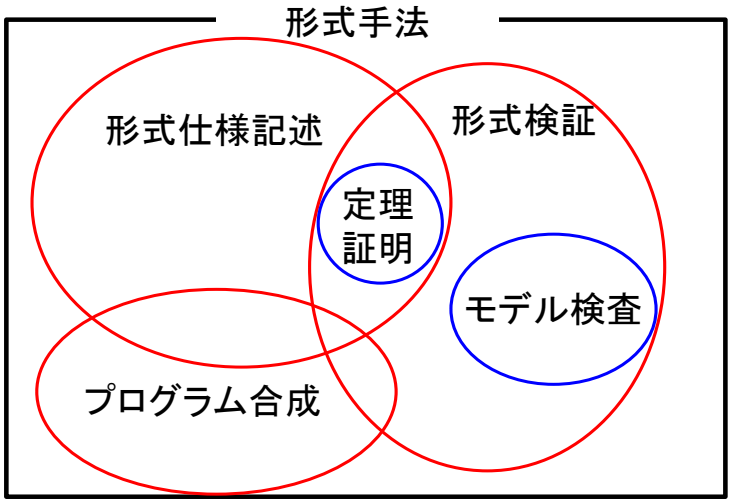


Fig. 3.2　Classification of model-checking in the formal method

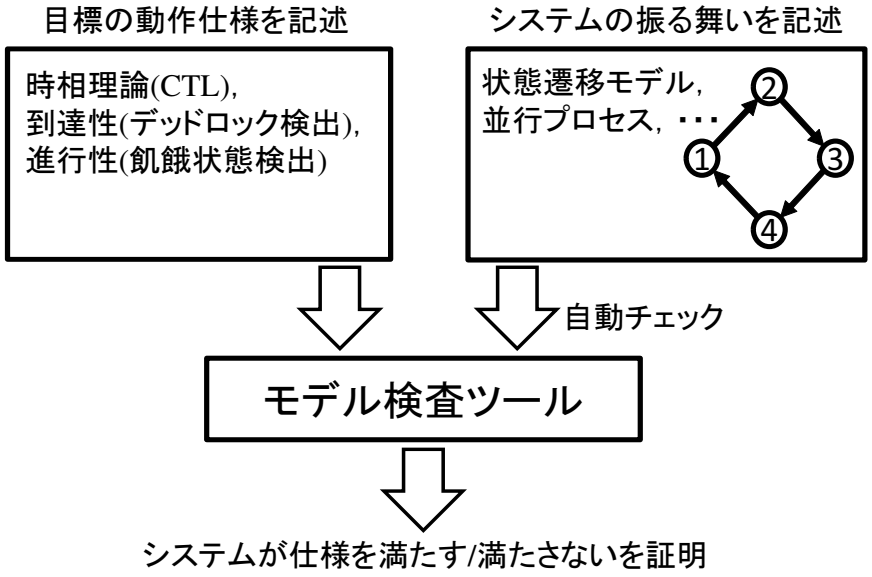


Fig. 3.3　Process of model checking

3.4　モデル検査

モデル検査による検証には，システムの振る舞いを網羅的に計算機上で展開する検査手法と，プロセス代数に基づく数学的な検査手法がある．前者では検査する際の内部表現として $Kripke$ 構造呼ばれる状態遷移モデルが使われることが多い．$Kripke$ 構造は，通常のオートマトンとは異なり，エッジではなく状態にラベルが定義されている．$Kripke$ 構造を使った検査方法にも，明示的モデル検査と記号的モデル検査の 2 つの分類がある [91].

明示的モデル検査
実行系列を明示的に算出して探索する．多くの明示的モデル検査では実行系列中に特定の性質が成立しているかを検証する．ループなどにより無限に動き続ける可能性があるシステムの振る舞いに対しても検証可能である．

記号的モデル検査
モデル検査を振る舞いモデルと検証項目の充足関係判定問題として捉え，問題解決を効率的に実行するアルゴリズムを適用する．代表的なアプリケーションとして「SMV」が挙げられる．

本研究では，このうち明示的モデル検査が実施できるツール「UPPAAL」を使用する．UPPAAL ではシステムが時相理論に基づき状態遷移系で抽象化される．この抽象化されたモデルを時間オートマトンと呼ぶ．また，システムの動作仕様は CTL(Computational tree logic) で記述される．以下にはこの時間オートマトンと CTL を説明する．

3.4.1　時間オートマトン

時間オートマトンの各状態には滞在条件，ガードには遷移条件，同期条件，更新式が記述される．ただし，ガードは各状態間の遷移関係を示す．

- 滞在条件…この条件式を満たす限り，そのノードに留まることができる．
- 遷移条件…条件式が成り立つときに遷移可能であり，クロック変数の条件，bool 型の変数の真偽などで記述される．同期条件が記述されている場合はそれも満たす必要がある．
- 同期条件…2 つのシステム間でそれぞれのあるガードでの遷移を同時に行うための条件（ハンドシェイクによる同期）．変数！と変数？と記述されたそれぞれのガードで同時に条件が満たされる．変数！は同期条件における同期信号の送信側で，変数？は同期信号

の受信側を示す．変数？が変数！からの同期信号を受け取ることで両方のオートマトンのガードで同時に遷移が行われる．

- 値の更新…特定の変数の値を変えたり，定義した関数を用いたりすることができる．

ただし滞在条件が成り立っていても，同期により遷移する場合は遷移しなければならない．また本研究で用いる同期条件は，遷移条件が成り立ち，変数！と変数？のペアが揃う場合，クロック変数の増加が起こる前に遷移する．具体例として Fig. 3.4 のようなオートマトン α, β がある場合を考える．それぞれに NodeA，B があり，NodeA の滞在時間は単位時間 4 以下，NodeB は単位時間 3 以下である．まず A から B への遷移条件は滞在時間に相当するクロック変数 t が 3 以上であることである．つまり，NodeA に到達してから NodeB へ遷移するまでの時間は 3〜4 になる．これは，NodeA におけるシステムの挙動が 3〜4 の間で自由に振舞っていることを表す．

次に NodeB における同期条件による遷移について説明する．オートマトン α の自己遷移では「Stay_B!」という同期条件，オートマトン β・NodeB のガードでは「Stay_B?」が与えられている．「Stay_B!」は同期条件における同期信号の送信側，「Stay_B?」の受信側を表しており，「Stay_B!」から「Stay_B?」への同期信号の送受信がされた瞬間にオートマトン α の自己遷移とオートマトン β・NodeB からの遷移が同時に行われる．Fig. 3.4 の同期条件による遷移の瞬間を Fig. 3.5 の 3 番目に表示している．ただし，Fig. 3.5 では，同期条件以外の遷移条件，滞在条件，更新を省略している．まず，この図では 1 番目にオートマトン α がすでに NodeB に滞在し，オートマトン β で NodeA から B への遷移が行われている瞬間を表示している．この次の瞬間どちらのオートマトンも NodeB に滞在する．このオートマトン α の自己遷移とオートマトン β・NodeB からのガードにはそれぞれ同期信号の送受信を行う Stay_B?，Stay_B! が与えらているため，次の瞬間 (3 番目) に同期条件による遷移を行うことができる．この遷移は同時刻に行われる．この同期条件の後 (4 番目) は，オートマトン α で NodeB，オートマトン β で NodeA の状態になる．システムをこのような時間オートマトンで抽象化した上でモデル検査を行うことで，動作仕様を保証するか否かを証明することができる．この証明の詳細は 3.4.3 節で述べる．

3.4.2　CTL

システムの動作仕様はモデル検査において，CTL(=Computational tree logic:計算機論理) で記述される．CTL では，特定の動作仕様が常に or いつか，ある実行経路で or 全ての実行経路で満たされるかを記述することができる．この「常に/いつか」,「ある実行経路で/全ての実行経路で」の意味の記号を以下に列挙する．

- A・・・常に
- E・・・いつか
- F・・・現在からのある実行経路で
- G・・・現在からの全ての実行経路で

たとえば，p という条件について CTL で「AGp」と記述した場合には「現在からの全ての実行経路で，常に p が成立する」という意味になる．逆に「EFp」と記述した場合には「現在からのある実行経路で，いつか p が成立する」という p の実現可能性の是非を示す仕様になる．次にこの p の記法について説明する．Fig. 3.4 のオートマトンについて，「Automaton α が NodeB に滞在する」という仕様は「(オートマトン名).(状態名)」次のように記述される．

$$\mathrm{Automaton}\alpha.\mathrm{NodeB} \tag{3.1}$$

デッドロック検査ではこの表記の CTL を使う．デッドロックを起こさないという仕様は，「全ての状態で常に状態遷移を繰り返す」と解釈して良い．これを上記の言葉で書き換えると，「現在からのある実行経路において，常に同じ状態に滞在し続ける実行経路は存在しない」と捉えることができる．この仕様は CTL で次のように記述される．

$$\mathrm{EFAG}\neg(X.Y), \forall X \in \{\text{All automata}\}, \mathrm{Y} \in \{\text{All states}\}. \tag{3.2}$$

この記述は後の章のデッドロック検査でも使用する．

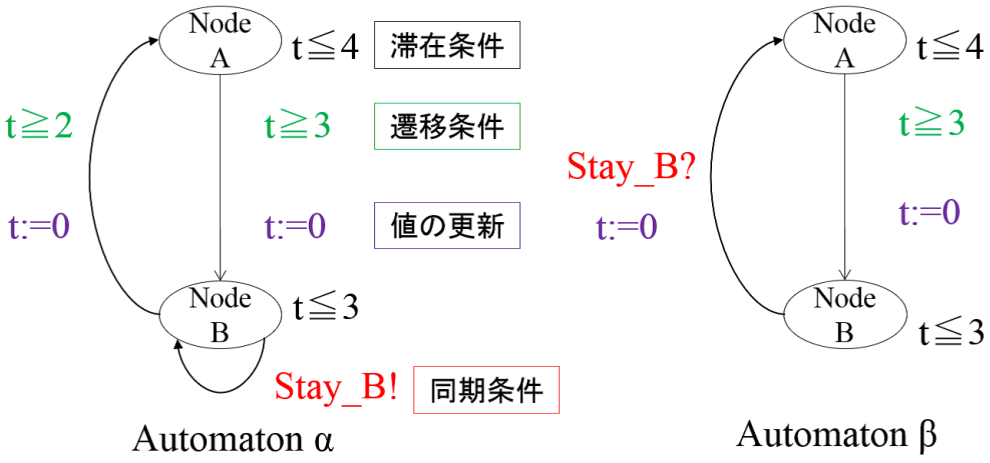


Fig. 3.4　Example of Automaton

3.4.3　モデル検査ツール UPPAAL による検証方法

以上のシステムを抽象化した時間オートマトンと動作仕様を記述した CTL を用意することで，モデル検査ツール UPPAAL でシステムがその動作仕様を満たすかを確かめることができる．Fig. 3.7 は UPPAAL におけるランダムシミュレーションの画面を表示しており，時間オートマトンを複数有するシステムの場合，現在の各状態と同期信号，状態遷移の履歴を確認することができる．検証の際には Fig. 3.4 で記述されているような遷移条件による滞在時間・遷移のタイミングを連続時間として扱い，想定される状態を網羅的に検証することができる．この仕組みを以下に説明する．

例として UPPAAL 内で時間変数 $0 \leq T_1 \leq 1$，$0 \leq T_2 \leq 1$ を使うとき，各時間は Fig.3.6(a) のような傾き 1 の方向に流れる [1]．このグラフを今度は (b) のように，$T_1 = T_2$，$T_1 > T_2$，$T_1 < T_2$，$T_1 = T_2 = 1$ 等の時間の大小関係・最大値 1・最小値 0 と

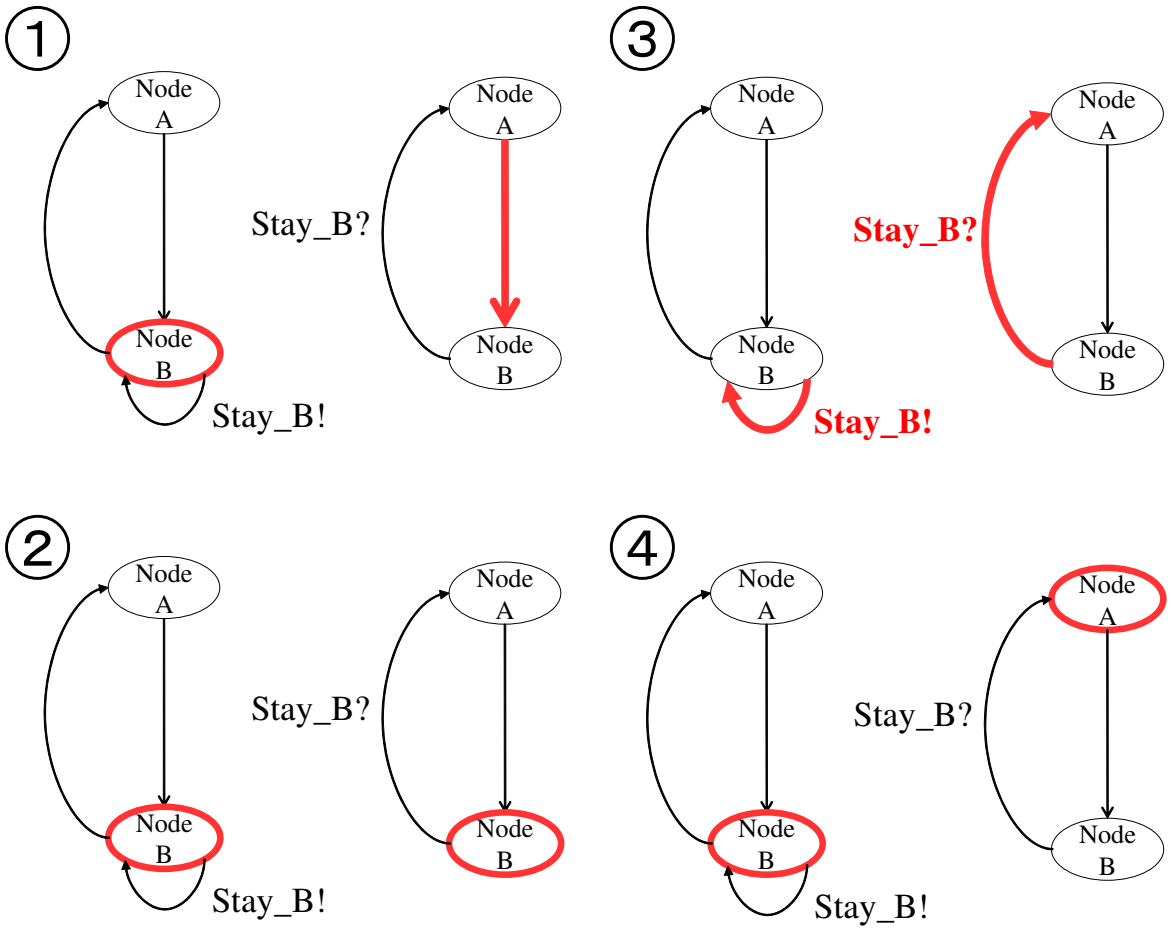
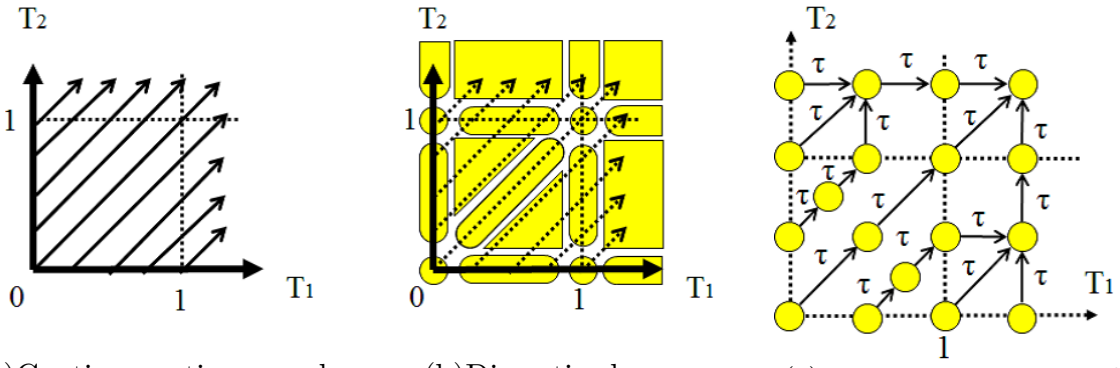


Fig. 3.5　Moment of synchronization between automata

の関係に基づいた成分に分ける．そして各成分を一つの状態として捉えた時間遷移グラフ (C) がモデル検査の計算上で使用される．これは UPPAAL の性質上，定数に整数しか使えないので，小数部分を抽象化する目的でも使われる．連続時間をこの時間遷移グラフに離散化し，(c) の各状態において仕様を満たすかを確かめることで，UPPAAL では網羅的な検証を実現している．また，このグラフでは $T_1 = 1$ に達した後，T_2 の値を維持しつつ $T_1 = 0$ にリセットされる．時間変数のリセットを繰り返す事で，$0 \leq T_1 \leq 1$，$0 \leq T_2 \leq 1$ 内で離散化された全ての条件を経由してモデル検査をすることができる．



(a)Continuous time graph　(b)Discretized area　(c)Time transition graph

Fig. 3.6　Verification with continuous time[1]

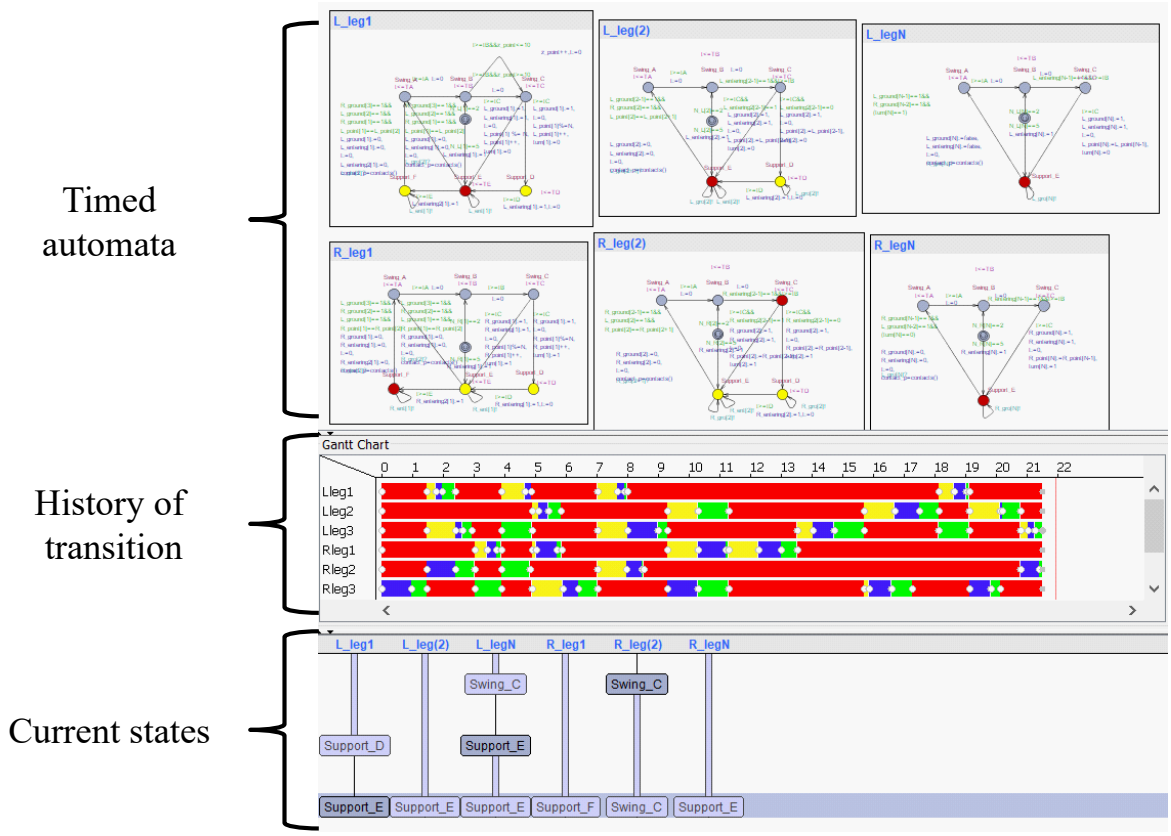


Fig. 3.7　Random simulation in UPPAAL

3.5　本論文でのモデル検査の使い方

モデル検査は，主にシステム開発の際に提案したシステムが特定の動作仕様を満たすか否かを確かめるために利用される [91]．その際，提案したシステムが動作仕様を満たさないことがモデル検査で分かった場合に，動作仕様を満たすシステムを再度考え直すという方法が取られる．この方法をとる 1 つのメリットは，ハードウェア開発において実機試験を行う手間を省き，設計図や抽象度の高い数理モデルの段階で，システムの破綻を早期に見つけることができることである．これによって，デッドロックの起こさないハードウェアの開発効率を大幅に向上させることができる．また，システムが特定の動作仕様を満たすかを証明するためにも用いられる．

本論文でも同じようにモデル検査をロボットの制御器設計に使うが，使用の前提が異なる．本論文では，設計対象となるシステム (歩行制御「接地点追従法」に複数の制御要素を追加したロボットの制御系) が既に決まっているものとしている．その前提で，このシステムで動作仕様を満たすための制約があると考え，その制約の導出とシステムへの適用を行う．その制約とはシステムの振る舞いを拘束するものである．

具体的には今回使用するモデル検査用ツール UPPAAL 上の時間オートマトンで使用する滞在時間の範囲が時間制約として導出される．この時間制約を導出する過程で，時間制約をある評価に沿って広げる操作と，モデル検査を繰り返す操作の両方を行う．4 章ではこの具体的な手法の流れを 6 脚ロボットの制御系のモデルと合わせて説明する．また，本論文の手法は，同じような特定のシステムに対して動作仕様を満たすためにどのような制約が必要かを求める問題にも適用できると考えられる．

3.6　まとめ

本章では，モデル検査による事象駆動システムの動作仕様の検証の方法について説明した．そのために，モデル検査が形式手法・形式検証のどの枠組みに含まれるのかを述べた上で，モデル検査をするために必要な時間オートマトンと CTL で記述する動作仕様の説明をし，最後にモデル検査においてどうやって網羅的な検証を実現しているのかを説明した．次章ではこのモデル検査を使って接地点追従法がデッドロックを起こさない制約を導出する．

第 4 章

時間オートマトンとモデル検査を用いた 6 脚移動ロボットの接地可能領域の導出

4.1　はじめに

脚移動ロボットの制御器設計における 1 つの課題として，現行の制御器のパラメータ設計の研究がされている．関連研究においては，CPG におけるニューラルネットの重みや [21]〜[27]，あらかじめ歩容を限定した制御に関しては歩行周期，遊脚・接地のデューティ比等のパラメータを遺伝的アルゴリズム [92]〜[94]，強化学習 [95]〜[97]，何かしらの最適化手法 [98], [99] に基づいて導出する手法が一般的に取られている．本章では制御器設計の課題のうち，このパラメータ設計に着目する．

接地点追従法の先行研究においても設計するべきパラメータが存在する．たとえば，岸ら [82] が提案した A*探索を用いた接地点探索 (運動計画) の手法においては，ロボットの脚が接地できる領域 (以後，接地可能領域と呼ぶ) を Fig. 4.1 のような直方体領域で与えていた．しかしながら，この領域は定量的な指標に基づいて設計されたものではないため，ロボットの歩行の弊害となる状況も存在し，場合によってはデッドロックになる．たとえば，接地点追従法は後続脚へ接地点を引き継ぐことで歩行が成立するが，この後続脚へ脚先が届かない状況が生まれるのである．他にも脚先の移動速度や接地点計画にかかる時間を含む歩行制御法に関する複数の制御パラメータを定量的な手法で導出する必要がある．そこで，本章では，ロボットが歩行し続ける (デッドロックを回避する) ような接地可能領域および，歩行制御のパラメータを導出する．

そこで，本章では，数学的根拠をベースにデッドロックを回避できるかを検証できるモ

デル検査 (3 章) を使う．このモデル検査そのものはモデルの動作仕様を確認するために用いられるのが一般的だが，本章ではモデル検査をパラメータ設計に応用するためのアルゴリズムを提案する．この手法は野村ら [80] の手法を参考にする．野村らはムカデ型移動ロボットの動作を時間オートマトンによりモデル化し，デッドロック回避および接地点追従法における先頭脚から最尾脚への接地点の受け渡しの動作に関するモデル検査を行った．このモデル検査を行う過程で，ロボットの動きに相当するモデル上のパラメータ (時間制約) を逐次更新することで，以上のデッドロック回避等の動作仕様を数学的根拠に基づき保証可能なロボットの振る舞いの集合を導出することができた．本章では，野村らのムカデ型ロボットではなく，6 脚移動ロボットを対象とした手法を提案する．

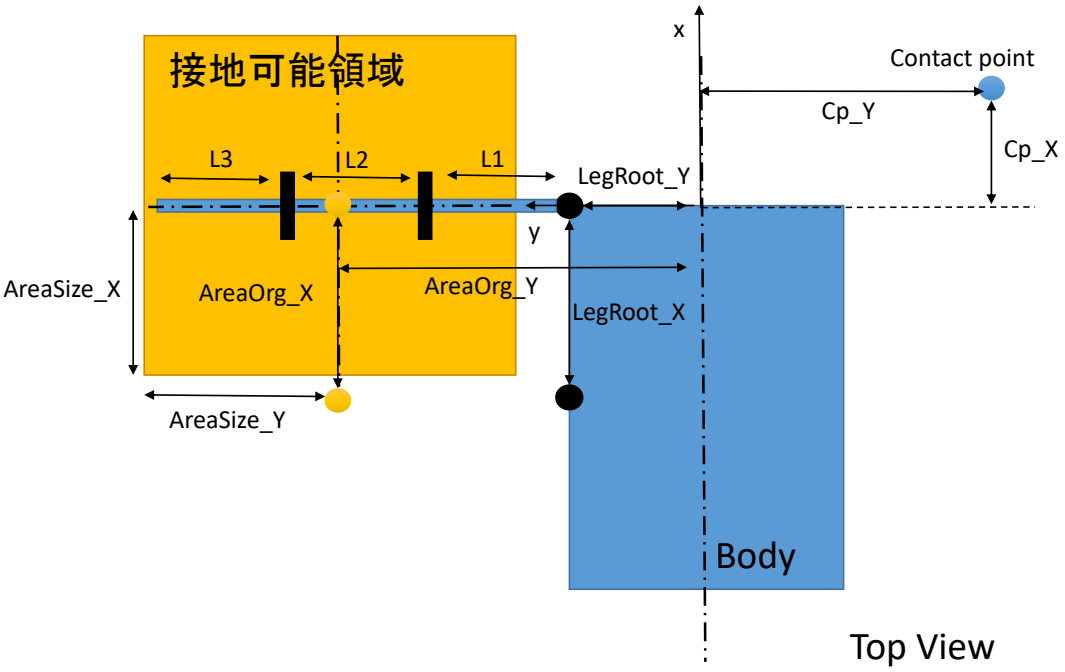


Fig. 4.1　Contactable area in Kishi's reseach

4.2　本章で用いる接地点追従法とロボットの構造

本章で用いる接地点追従法では主に制御モード 4 における脚の動作，制御モード 4 から 1 への遊脚条件，接地点計画の方法を追加している．ロボットの構造を前提に 1 つずつ述べる．

4.2.1　ロボットの構造

Figure. 4.2 のように，6 脚で構成されているロボットを考える．このロボットは 3 本の脚が左右対称に配置された構造になっている．このうち，胴体の正面 (カラーカメラ，深度センサの向き) にある 2 脚を先頭脚とし，左右の識別番号を右側で 0，左側で 1 とし，先頭から j 番目の脚を $\mathrm{Leg}[i][j](i = 0, 1, j = 1, 2, 3)$ とする．このとき，$\mathrm{Leg}[i][j]$ の左右反対側に相当する脚は $\mathrm{Leg}[1 - i][j]$ となる．また，以降の説明で述べる前脚とは $\mathrm{Leg}[i][j]$ に対する $\mathrm{Leg}[i][j - 1]$，後脚は $\mathrm{Leg}[i][j]$ に対する $\mathrm{Leg}[i][j + 1]$ を指す．そして，$\mathrm{Leg}[i][j]$ の付け根に原点を持つ直交座標座標系を $\Sigma_{i,j}$，脚先の可到達域を $\Omega_{i,j}$ と定義する．ただし，ここで述べる脚先とは $\mathrm{Leg}[i][j](i = 0, 1, j = 1, 2, 3)$ の 3 リンク目の末端，エンドポイントを指す．各脚先座標は $\Sigma_{i,j}$ 上のものとする．脚の構造は 3 リンク 3 自由度とし，

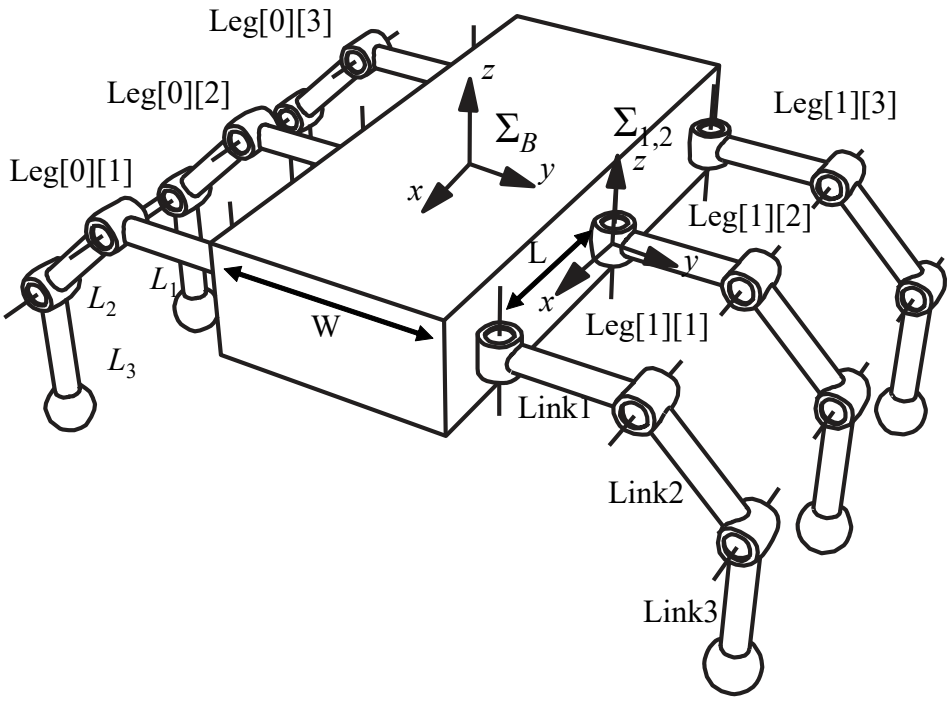


Fig. 4.2　Hexapod robot and local coordinate system on each leg

Link1 と胴体の左右に脚を 2 対配置し，一定距離毎に前後に 3 脚配置した構造になっている．

座標系 $\Sigma_{i,j}$ で捉えた脚先の座標を $(x_{i,j}, y_{i,j}, z_{i,j})^T$ としたとき，この脚 Leg$[i][j](i = 0, 1, j = 1, 2, 3)$ の運動学については，関節角度を脚と胴体の付け根から順に $q_{1_{i,j}}$，$q_{2_{i,j}}$，$q_{3_{i,j}}$ と定義したとき，$\Sigma_{i,j}$ を中心とした脚の先端座標が以下のように表される．ただし，各リンク長を L_1，L_2，L_3 とする．

$$\begin{aligned} x_{i,j} &= \{L_1 + L_2 \cos q_{2_{i,j}} + L_3 \cos(q_{2_{i,j}} + q_{3_{i,j}})\} \cos q_{1_{i,j}}, \\ y_{i,j} &= \{L_1 + L_2 \cos q_{2_{i,j}} + L_3 \cos(q_{2_{i,j}} + q_{3_{i,j}})\} \sin q_{1_{i,j}}, \\ z_{i,j} &= L_2 \sin q_{2_{i,j}} + L_3 \sin(q_{2_{i,j}} + q_{3_{i,j}}). \end{aligned} \tag{4.1}$$

ここでは実機の様々な構造に対応するように，$q_{1_{i,j}}$ の可動域が -90〜90 度，$q_{2_{i,j}}$，$q_{3_{i,j}}$ の可動域が -180〜180 度であると仮定する．接地点追従法には前後脚の可到達域の重なり具合が重要になる．この仮定のもと，可到達域を求めると次式と Fig. 5.2 のように表される．

$$\Omega_{i,j} = \{(x_{i,j}, y_{i,j}, z_{i,j}) | (L_2 - L_3)^2 \leq {z_{i,j}}^2 + \left(\sqrt{{x_{i,j}}^2 + {y_{i,j}}^2} - L_1\right)^2 \leq (L_2 + L_3)^2\}. \tag{4.2}$$

これに加えて後脚 Leg$[i][j+1]$ の可到達領域を $\Sigma_{i,j}$ にて表すと前後の脚間距離を L を用いて次式で表現される．

$$\Omega_{i,j+1} = \{(x_{i,j}, y_{i,j}, z_{i,j}) | (L_2 - L_3)^2 \leq {z_{i,j}}^2 + \left(\sqrt{(x_{i,j} + L)^2 + {y_{i,j}}^2} - L_1\right)^2 \leq (L_2 + L_3)^2\}. \tag{4.3}$$

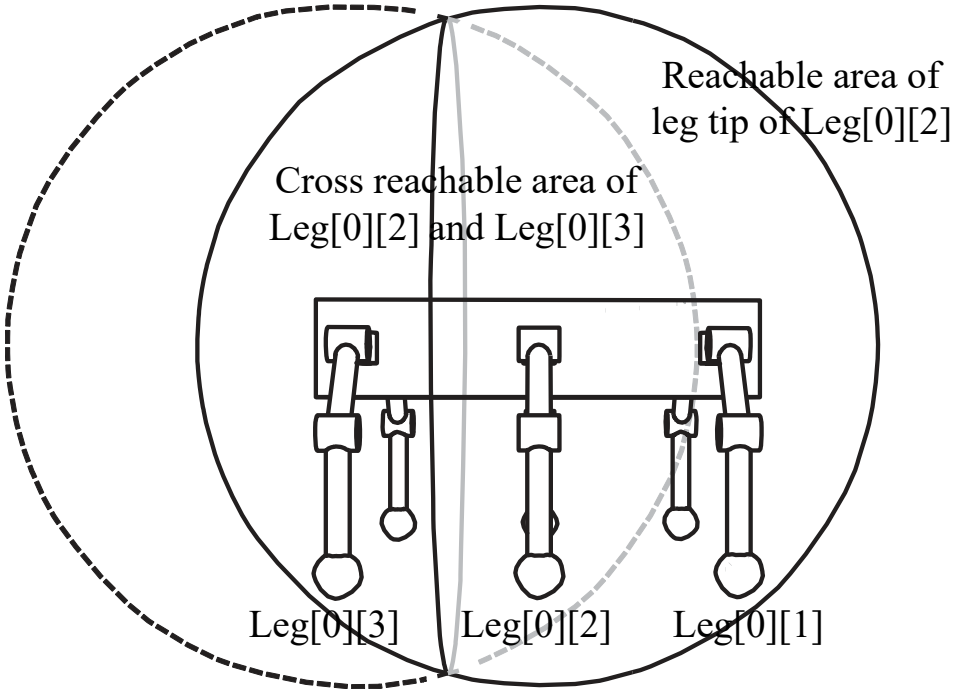


Fig. 4.3　Reachable area of center and rear leg-tip

接地点追従法では，ある脚から見て後脚との接地点交換は前述の制御域 $A_{i,j}$ の Area3 で行われる，この領域 (4.2),(4.3) が交差する領域に該当する．

前後の脚は次節で述べる歩行制御法により (4.2)，(4.3) 式が交差する領域内で接地点を変換する．

4.2.2　制御域の再定義

2 章では，ロボットの可到達域 $\Omega_{i,j}$ とは異なる領域である制御域 $A_{i,j}$ を定義した．本章末尾では，従来手法とは異なる接地可能領域が導出されるため，それに伴い，$A_{i,j}$ の領域も変更される．このことを想定し，本章では $A_{i,j}$ を可到達域 $\Omega_{i,j}$ と同じ領域まで広げ，その中で，接地可能領域や各制御モードにおける脚の動作範囲を導出する．よって，2 章で定義した制御域 $A_{i,j}$ は次の 3 つの領域に分けられる．

Area 1: 制御域 $A_{i,j-1}(=\Omega_{i,j-1})$ と重なる領域；$\Omega_{i,j} \cap \Omega_{i,j-1}$，
Area 2: 前後の脚の制御域と重ならない領域，
Area 3: 制御域 $A_{i,j+1}(=\Omega_{i,j+1})$ と重なる領域：$\Omega_{i,j} \cap \Omega_{i,j+1}$．

4.2.3　制御モード 4 における脚の動作

制御モード 4 における脚の動作はロボットの胴体の移動に基づき，運動指令を与えられる．この胴体の移動は本章では現在の胴体位置から胴体座標 Σ_B で定義されるサブゴール P_B への等速直線運動で与える．このサブゴールの位置を考える上でロボットの接地点や支持多角形を考慮したい．まず，そのために 6 脚ロボットが支持多角形を維持しながら歩行するための最低限の接地点数が 3 個であることを考える．ある 3 点のみを着目して歩行させることで，接地点の数が 4 個 5 個と変わっても同じ制御によりロボットを推進させることができる．いっぽう，ロボットの転倒安定性を考慮した場合には 5 点接地したままで一脚ずつ接地・遊脚を後退させる手段も取ることができるが，この手段では脚を一脚ずつ組み替えるため，歩行の効率が悪くなると考えられる．そこで，本章では最新の 3 つの接地点から成る三角形または，次の接地目標点と最新の 2 つの接地点による三角形のいずれか (Fig. 4.4) を考え，この 3 角形の重心から高さ h の地点をサブゴールとして与える．たとえば，Fig. 4.4 は右側先頭脚 Leg[0][1] が新しく接地したときのロボットの姿勢である．このときの最新の三角形は Leg[0][1]，Leg[0][2](Leg[0][1] から 1 つ後ろの接地点)，Leg[1][1](反対側の先頭脚) から生成される．

この三角形の各頂点を胴体座標系 Σ_B 上で C_a, C_b, C_c と定義する．また，ロボットには姿勢センサを搭載しているものとし，姿勢センサによって計測された姿勢角を元にグローバル座標から Σ_B への回転行列 $R(\hat{\theta}, \phi_g - \hat{\phi}, \psi_g - \hat{\psi})$($\phi_g$, ψ_g はグローバル座標系で姿勢センサにより観測された胴体のロール・ピッチ角，$\hat{\theta}$, $\hat{\phi}$, $\hat{\psi}$ はそれらの目標角) を求める．ただし，本手法では，ロボットが水平姿勢に保つことを目的として目標角は全て，$\hat{\theta} = 0$, $\hat{\phi} = 0$, $\hat{\psi} = 0$ とする．

これらの変数を用いて一歩毎の胴体のサブゴール P_B は Σ_B 上で次の式で与えられる．

$$R(\theta, \phi, \psi) = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{pmatrix} \begin{pmatrix} \cos\psi & 0 & -\sin\psi \\ 0 & 1 & 0 \\ \sin\psi & 0 & \cos\psi \end{pmatrix} \tag{4.4}$$

$$P_B = \frac{C_a + C_b + C_c}{3} + R^{-1}(\hat{\theta}, \phi_g - \hat{\phi}, \psi_g - \hat{\psi}) \begin{pmatrix} 0 \\ 0 \\ h \end{pmatrix} \tag{4.5}$$

ここで，h は支持脚から胴体までの目標の高さとして操作者によって与えられる．さらに胴体がサブゴールに到達するための一ステップごとの胴体の目標座標の移動量 ΔP_B は次

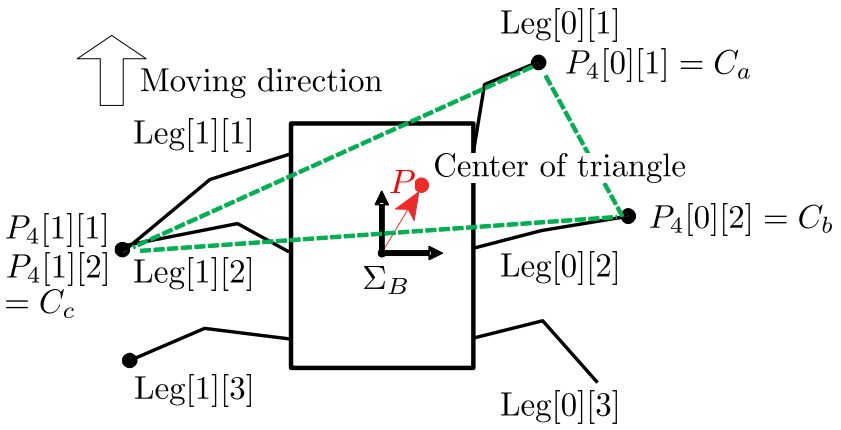


Fig. 4.4　The projected center of body and margined triangle made from three contacting points

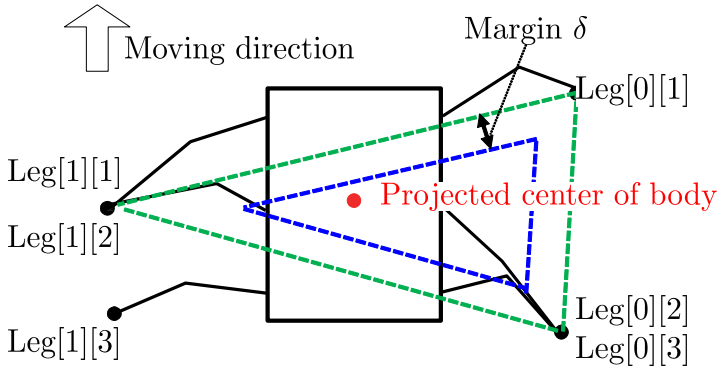


Fig. 4.5　Transition condition from control mode 4 to 1 with support triangle

の式で与えられる．

$$\Delta P_B = \begin{cases} \frac{P_B}{|P_B|} v_4 \Delta t & |P_B| > \varepsilon, \\ \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} & \text{otherwise} \end{cases} \tag{4.6}$$

ここで，Δt は制御周期，ε は胴体の移動を止める閾値とする．そして，この胴体の移動を実現するための支持脚 Leg$[i][j]$ の目標座標 $P_4[i][j]$ は次の式で導出される．これに脚先を追従させることで胴体がサブゴール P_B に追従するための脚の運動指令が生成される．

$$P_4[i][j] \leftarrow R^{-1}(\hat{\theta}, \phi_g - \hat{\phi}, \psi_g - \hat{\psi})(P_4[i][j] - \Delta P_B) \tag{4.7}$$

ただし，$\leftarrow$ は毎ステップ値が更新されることを指す．

4.2.4　制御モード 4〜1 への遷移条件

接地点追従法は，本来 10 脚以上を持つ，ムカデ型ロボットの歩行制御法として提案された [63]．この方法自体はロボットのバランスを考慮した制御を含んでいない．そこで，6 脚用の支持脚条件を導入する必要があると考えられる．Leg$[i][j]$ が脚を離す（制御モード 4 から 1 への遷移）条件として次のものが与える．

1. $j \leq 2$ の場合，後脚 Leg$[i][j+1]$ が自脚の接地点を引き継ぎ，反対側の脚 Leg$[1-i][j]$ が接地している．$j = 3$(最尾脚) の場合，反対側の脚 Leg$[1-i][j]$ が接地している．
2. 前述の新しい接地点 3 点 C_a, C_b, C_c が成す支持三角形を重力方向に投影して作られる三角柱の内部にマージンを取った領域に胴体の中心が含まれている．
3. 対角の 3 脚 Leg$[i][j \pm 1]$, Leg$[1-i][j \pm 2]$ が制御モード 4(接地相) である．

まず 1 つ目の遷移条件は従来のムカデ型ロボット用の接地点追従法にも備わっているものである．2 つ目は支持多角形の中に重心を維持するために追加した遷移条件である．このとき生成される三角形は Fig. 4.4 と同じように新しい順に生成された接地点 3 つから生成される (Fig. 4.5)．具体的にこの遷移条件の式は以下のように記述される．ただし，C_{ag}, C_{bg}, C_{cg} は C_a, C_b, C_c を水平座標に変換した上で，高さ成分を除いた 2 次元ベクトルである．この 2 次元ベクトルを表現している水平座標の原点が胴体の位置を写像した位置に相当する．よって，この原点が C_{ag}, C_{bg}, C_{cg} が成す三角形からマージン δ を取った領域に内包されているかを条件とすればよい．(4.11) 式は 3 角形の各辺から原点までの垂直距離がマージン δ 以上であるか，(4.12)，(4.13) 式は 3 角形が原点を内包しているかの条件を表現している．ただし，(4.11)〜(4.13) 式の「×」は座標ベクトル同士の外積を示す．

$$C_{ag} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} R^{-1}(0, \phi_g, \psi_g) C_a \tag{4.8}$$

$$C_{bg} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} R^{-1}(0, \phi_g, \psi_g) C_b \tag{4.9}$$

$$C_{cg} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} R^{-1}(0, \phi_g, \psi_g) C_c \tag{4.10}$$

$$\min \left\{ \frac{|C_{ag} \times\ C_{bg}|}{|C_{ag} - C_{bg}|}, \frac{|C_{bg} \times\ C_{cg}|}{|C_{bg} - C_{cg}|}, \frac{|C_{cg} \times\ C_{ag}|}{|C_{cg} - C_{ag}|} \right\} \leq \delta \tag{4.11}$$

$$\{(C_{ag} - C_{bg}) \times\ C_{ag}\} \cdot \{(C_{bg} - C_{cg}) \times\ C_{bg}\} > 0 \tag{4.12}$$

$$\{(C_{ag} - C_{bg}) \times\ C_{ag}\} \cdot \{(C_{cg} - C_{ag}) \times\ C_{cg}\} > 0 \tag{4.13}$$

3 つ目も 2 つ目の遷移条件と同様にロボットのバランスを維持するための遷移条件であり，これにより 3 点以上の支持が守られやすくなる．この遷移条件によって各脚の接地・遊脚のパターンは Fig. 4.6 のように 15 通りに限定される．Fig. 4.6 は 3 つ目の遷移条件に沿って，左側に少なくとも対角の 3 脚 Leg[0][1]，Leg[1][2]，Leg[0][3] が接地しているパターンを，右側に反対の 3 脚 Leg[1][1]，Leg[0][2]，Leg[1][3] を含む接地パターンを示しており，それぞれ 8 通り，そのうち重複する接地パターン (全ての脚が接地しているもの) が 1 通り存在する．ただし，接地点追従法では前後脚で接地点共有する状況が存在するため，3 脚以上が接地していても接地点共有を考慮して数えられる接地点の数が 3 点未満になり静歩行が維持されないケースも原理的に起こり得ることをここで注意しなければならない．

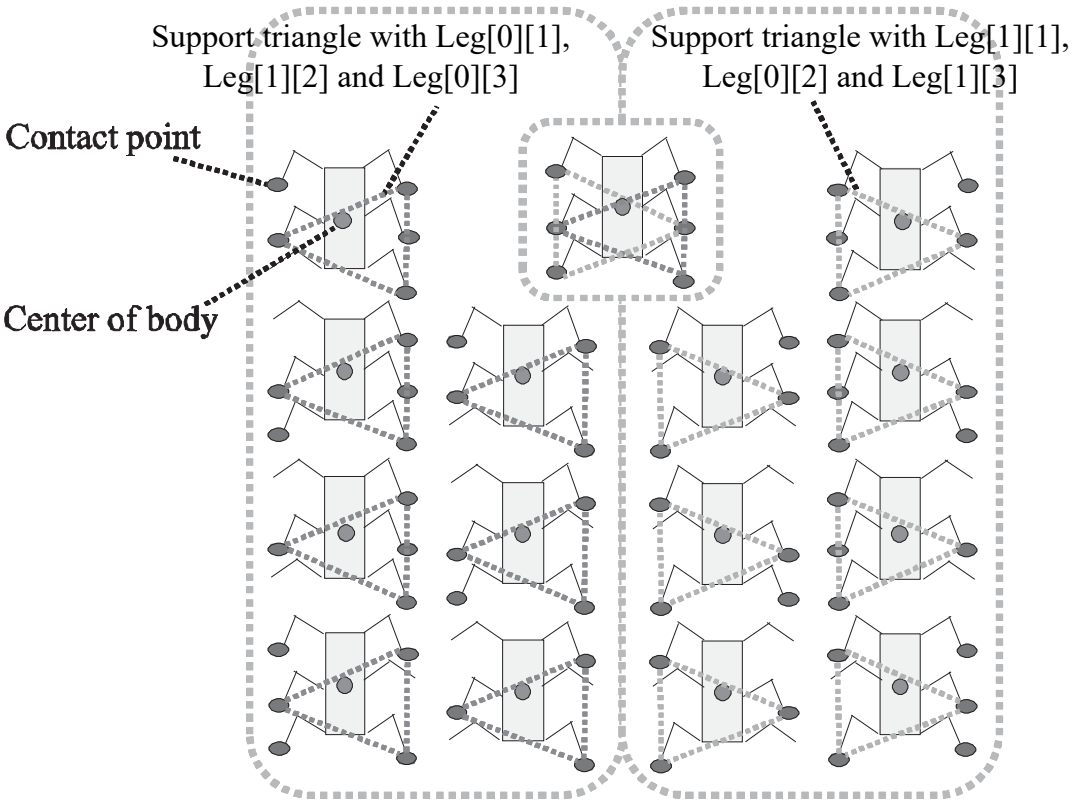


Fig. 4.6　Allowable contacting patterns under trandition condition from mode 4 to 1

4.2.5　先頭脚の接地点探索

本章では，岸ら [82] の手法を参考に深度センサを用いた接地点探索を行う．深度センサはロボット上部に配置しており，Fig. 4.7 のように先頭脚の接地点を計画する際に赤外線を前方の環境に照射し，反射光を捉えることで照射された地形を点群データとして扱うことができる．さらにこの点群の中で，4.4 節で導出する接地可能領域に含まれる点群から接地点の探索を行う．

ロボットのバランスを保つために支持多角形を最大化する接地点探索をさせたい．そこで，前述した次の接地目標点と最新の 2 つの接地点を水平写像した二次元ベクトル (C_{ag}, C_{bg}, C_{cg}) による三角形の面積が最大化される点を新しい接地点に決定する．この時の探索は接地可能領域に含まれる点群データに関して全探索を行っている．点群座標を $C_i (i$ は点群のラベル$)$ としたきに先頭の接地点 C_a は (4.14) 式で決定される．

$$C_a \leftarrow \arg\max_{C_i} \left| \left\{ \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} R^{-1}(0, \phi_g, \psi_g) C_i - C_{bg} \right\} \times \left\{ \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} R^{-1}(0, \phi_g, \psi_g) C_i - C_{cg} \right\} \right| \quad (4.14)$$

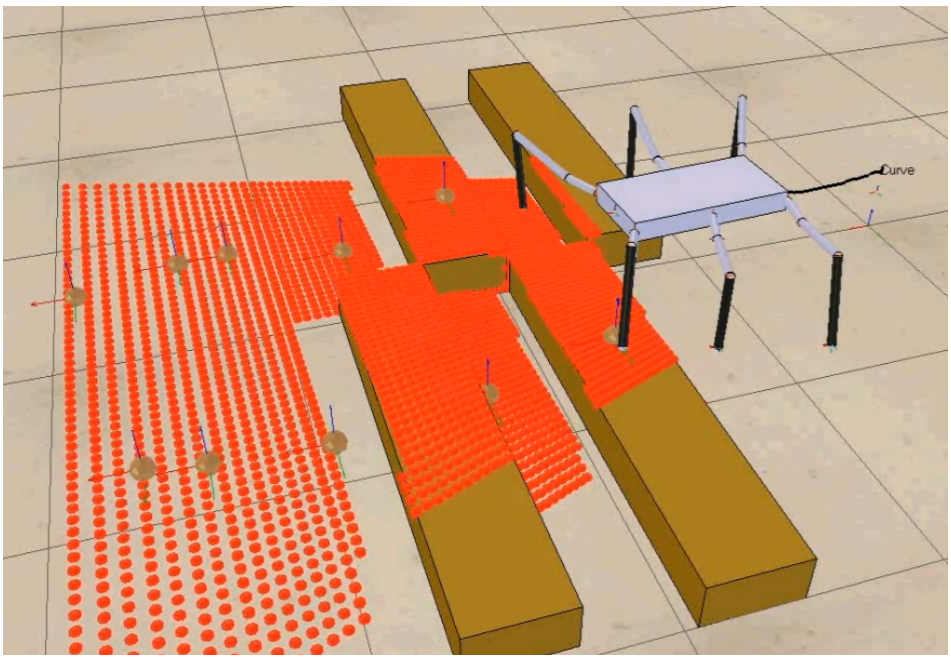


Fig. 4.7　Planning contact points of fore legs

4.3　モデル検査に基づく実機モデルのパラメータ導出

本章では，最終的に実機の制御に関するパラメータを，各脚の接地可能領域，各制御モードにおける脚先の移動速度，接地点計画時間 (制御モード 2 の滞在時間) の 3 つとして導出したい．いっぽう，時間オートマトン内で用いるパラメータとして本研究では時間オートマトン上の各状態の滞在時間の上下限値を扱う．以下，この時間オートマトン上のパラメータを実機と分けて「時間パラメータ」と呼び，以下にこの制御に関するパラメータの導出過程の概要を示す．

4.3.1　実機モデルのパラメータ導出の概要

本手法での接地点追従法から接地可能領域導出までの流れを Fig. 4.8 に沿って以下に示す．

Step1　実機モデルと満たすべき動作仕様を時間オートマトン・CTL に書き換える．
Step2　時間オートマトン・CTL でモデル検査を行い，動作仕様を満たす時間パラメータを導出する．
Step3　時間パラメータから実機の制御に関するパラメータへの対応を行う．
Step4　対応後のパラメータ空間（部分空間）に絞って，実機での検証を行う．

まず，Step1 で実機とその動作仕様を定義し，2 つをモデル検査用の時間オートマトン・CTL に抽象化する．本研究では接地点追従法に基づく 6 脚ロボットの脚の振る舞いを対象にこの抽象化を行う．これを元に Step2 で時間オートマトンの時間パラメータの探索を行い，Step3 で得られた時間パラメータを元の抽象化する前の設計対象（実機）に対応させる．最後に Step4 では Step1〜3 の操作が適切であることを確認するため，実機モデルに導出したパラメータを代入して検証を行う．ただし，この手順で得られるパラメータ集合は，実際のシステムにおいて動作仕様を満たすパラメータ集合に対して必要十分ではなく，十分条件になる．

特に Step2 の手順については Fig. 4.9 のフローに従って，時間パラメータの探索を行う．このフローは野村らが提案したものである．STEP2 の最初にモデルのテンプレートを作成する．このテンプレートとは，時間オートマトンに時間パラメータを代入していないプログラムのことを指し，どの時間パラメータを代入しても UPPAAL による動作検証ができるようにあらかじめ用意したものである．このテンプレートにロボットの具体的な振る舞いに対応する時間パラメータを導入するのが 2 番目の手順である．2 番目の手順で

は時間パラメータを 0〜1000 までの間の自然数 (離散値) として扱い，この中から時間パラメータの組み合わせを全て模索し，テンプレートに全ての時間パラメータが代入される．これにより，パラメータの組み合わせと同じ数だけモデルが作成される．作成された全てのモデルについて 3 番目でモデル検査を使って全て動作検証を行う．以降で，モデル検査を用いて接地点追従法による歩行が成立するためのロボットの振る舞いを導出する．

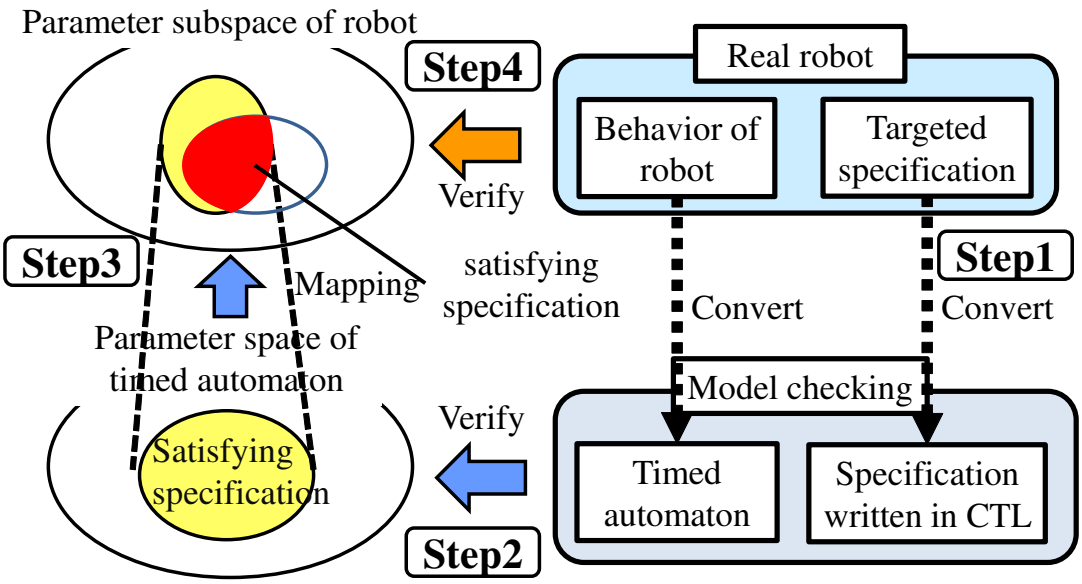


Fig. 4.8　Parameter design based on formal verification

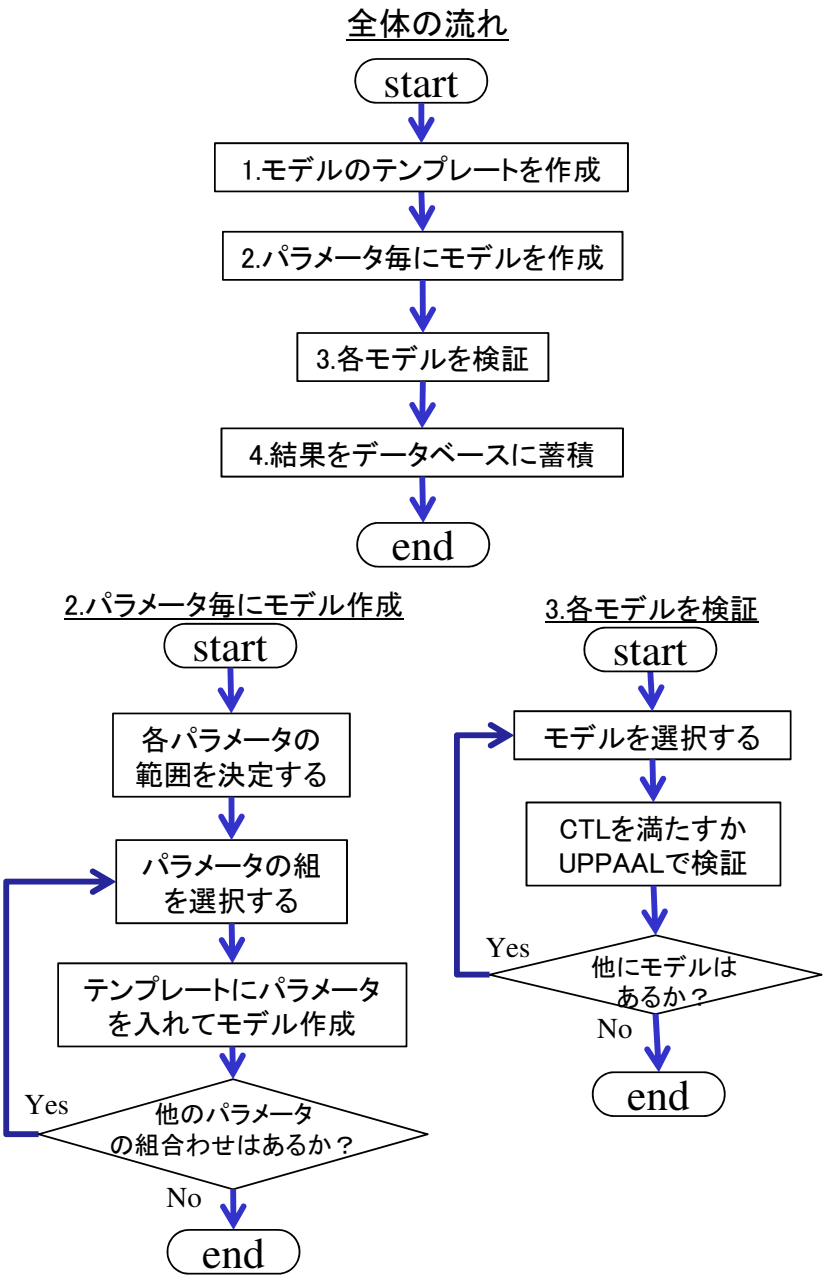


Fig. 4.9　Algorithm of derivation of parameter group in Nomura's research

4.3.2　接地点追従法に伴うロボットの振る舞いのモデル化

ここでは，ロボットの動きを Fig. 4.10 のように時間オートマトンにより表す．各ノードにおける滞在条件はノードの上部に記述する．脚先動作オートマトンはロボットの各脚先の動作を表わし，状態は時間もしくは隣接する脚のオートマトンの状態に基づいて切り替わる．脚先動作オートマトンにおけるクロック変数 t，u は，各オートマトンごとにローカルに定義されている．また，各ノードの滞在時間に区間を与え，その最小値，最大値をそれぞれ $T_X^{\min}$，$T_X^{\max}$ （X は各ノード名の語尾のアルファベット）と表記する．以下に脚先動作オートマトンの各ノードでの脚先の状態および遷移条件について述べる．ただし，本章では「同期を取る」という言葉を信号の送受信の 2 つの意味で使用する．受信側はほかの脚が特定の状態にあるという信号を自脚が受け取ってつぎのノードへ遷移するということ，送信側は逆に自脚が特定の状態にあるという信号を他脚に送りつぎのノードへ遷移させるという意味で使う．この時間オートマトンの状態は元々の各脚に割り振られた制御オートマトンに対応させて脚毎に定義する．この時間オートマトン上の各状態は脚の以下の動作に対応する．

SwingA:　遊脚相：モード 1 の動作（接地点からの離脱）
SwingB:　遊脚相：モード 2 の動作（中間点での待機）
SwingC:　遊脚相：モード 3 の動作（接地目標点への接近）
SupportD:　接地相：モード 4（体節の前進：後脚の可到達域外 $\Omega_{i,j} \cap \overline{\Omega_{i,j+1}}$）
SupportE:　接地相：モード 4（体節の前進：後脚の可到達域内 $\Omega_{i,j} \cap \Omega_{i,j+1}$）
SupportF:　接地相：モード 4（体節の前進：2 つ後ろの可到達域内 $\Omega_{i,j} \cap \Omega_{i,j+2}$）
SupportG:　接地相：モード 4（脚先が可到達域の境界で引き摺られた状態）

定義した変数は以下のとおりである．

- $\mathrm{Ground}_{i,j} \in \{0,1\}$: Leg$[i][j]$ の接地フラグ．
- $\mathrm{Grounding}_{i,j}$: Leg$[i][j]$ の接地時に発信する同期信号．
- $\mathrm{Entering}_{i,j} \in \{0,1\}$: Leg$[i][j]$ が後脚の接地可能領域 $\Omega_{i,j+1}$ に侵入しているかのフラグ．
- $\mathrm{Entering2}_{i,j} \in \{0,1\}$: Leg$[i][j]$ が 2 つ後の接地可能領域 $\Omega_{i,j+2}$ に侵入しているかのフラグ．
- $t \in \mathbb{Z}$: クロック変数．
- $T_x^{\min}, T_x^{\max} \in \mathbb{Z}, (x \in \mathcal{A} = \{A, B, C, D, DE\})$: 遷移時間の最大値，最小値（時間制約）．

- $\mathrm{Point}_{i,j} \in \mathbb{Z}$: それぞれの脚の接地点番号.

以下に脚先動作オートマトンの各ノードでの脚先の状態及び遷移条件について詳細に述べる.

SwingA: 遊脚相（接地点からの離脱）

脚先が接地点から離れ，空中の中間点 $P_1^{i,j}$ に向かって動いている状態である．滞在時間を $t_A[i][j]$ とすると，$T_A^{\min} \leq t_A[i][j] \leq T_A^{\max}$ の間で SwingB に遷移する.

SwingB: 遊脚相（中間点での待機）

脚先が中間点で待機している状態である．最前脚については接地点計画を行い，他の脚は前後との同期を取る．最前脚については滞在時間を $t_B[i][j]$ とすると，$T_B^{\min} \leq t_B[i][j] \leq T_B^{\max}$ の間に遷移する．$T_B^{\min}$ は接地点計画を要する最短時間に相当する．中間脚と最後脚は前脚が SupportE のときに発信するメッセージ $\mathrm{Entering}_{i,1}$，$\mathrm{Entering}_{i,2}$ をそれぞれ受け取って SwingC へ遷移する.

SwingC: 遊脚相（接地目標点への接近）

脚先が次の接地点 (前脚先端)$P_3^{i,j}$ に向かっている状態である．脚先が前脚の脚先に到達するまでの時間を $t_C[i][j]$ とすると，遷移条件は前の状態 SwingB と合わせた時間 t_C について $T_C^{\min} \leq t_C[i][j] \leq T_C^{\max}$ とする．この間に SupportD または E に遷移する．SupportD と E への遷移の違いは主に接地する位置の違いに基づく (Fig. 4.11). つぎに SupportD と E への遷移条件を記す．まず，先頭脚についてはつぎの接地点位置を SupportD と E のどちらの接地可能領域に計画したかを表現することができない．そこで，ランダムに接地点を決めても動作が継続するかを確認できるように，SwingB から C を経由して，SupportD または E のどちらにもランダムに遷移できるように記述している．中間脚では，Fig. 4.11 を参考に中間脚と先頭脚の可到達域 $\Omega_{i,j}$ の重なりから説明する．先頭脚の脚先が最尾脚の可到達域内にあるとき，つまり先頭脚にとって SupportF の領域に接地しているとき，中間脚が接地点交換できる領域は中間脚にとって SupportE の領域になる．いっぽう，先頭脚が最尾脚の可到達域外でかつ中間脚の可到達域内のとき（最前脚にとって SupportE の領域にいるとき），中間脚が接地点を引き継げるのは中間脚にとって SupportD の領域である．つまり，先頭脚が SupportF のとき中間脚は SupportE に遷移し，先頭脚が SupportE のとき中間脚は SupportD に遷移する．そのため，最前脚が SupportF のとき，2 つ後ろの可到達域に侵入しているという信号 $\mathrm{Entering2}_{i,1}$ を送り，中間脚はその受信によって SupportD と E の遷移を変えてい

る．最後脚については接地相に SupportD を設けていないのでそのまま SupportE に遷移する．

SupportD: 接地相（体節の前進：後脚の可到達域外）

後脚の可到達域外に脚先を接地させながら体節を前進させている状態である．$\mathrm{Ground}_{i,j} = \mathrm{true}$ を返し，後脚と反対側の脚と同期をとる．ただし，この $\mathrm{Ground}_{i,j}$ は制御オートマトンにおける制御モード 4 および時間オートマトンにおける接地ノード SupportD〜F に滞在していることを示す信号であって，力センサを用いて足場との接触を確認しているものではない．SupportD から SupportE への遷移時間についても $T_D^{\min} \leq t_D[i][j] \leq T_D^{\max}$ という遷移時間の範囲を与える．

SupportE: 接地相（体節の前進：後脚の可到達域内）

後脚の可到達域内で，脚先を接地させながら体節を前進させている状態である．後脚の可到達域に侵入していることを示すため，$\mathrm{Entering}_{i,j} = \mathrm{true}$ を返し，SupportD もしくは E の状態にある後脚と同期を取る．本章の接地点追従法では，追加した支持脚条件に従って，特定の 3 脚が接地したことを確認してから脚を離す．遷移時間を $t_E[i][j]$ として接地が継続する時間 $t_{DE}[i][j](= t_D + t_E)$ に対して遷移条件を $T_{DE}^{\min} \leq t_{DE}[i][j] \leq T_{DE}^{\max}$ とする．

SupportF: 接地相（体節の前進：2 つ後の可到達域内）

中間脚が接地する領域を SupportD と E とに分けるために設けられた最前脚のみに存在する状態である．SwingC で説明したとおり，中間脚の遷移については最前脚が SupportF のとき中間脚は SupportE に遷移し，最前脚が SupportE のとき中間脚は SupportD に遷移する．また，この状態における遷移時間を $t_F[i][j]$ として接地が継続する時間 $t_{DEF}[i][j](= t_D + t_E + t_F)$ に対して遷移条件を $t_{DEF}[i][j] \leq T_{DE}^{\max}$ とする．

SupportG: 脚先が可到達域の境界で引き摺られた状態

後脚と接地点交換する前に脚先が地を掻き切り，可到達域後方の境界に到達した状態である．実機上では脚先が可到達域を超えようとした場合に，関節のモーターに無理な負荷が掛かるのを防ぐため，脚の動作を停止させている．以後，脚先が引き摺られた状態になり，デッドロックし，接地点交換できなくなる．時間オートマトン上の SupportD，E，F ではこの可到達域に到達するまでの時間を $T_{DE}^{\max}$ とし，これを超えると SupportG に遷移するものとしている．

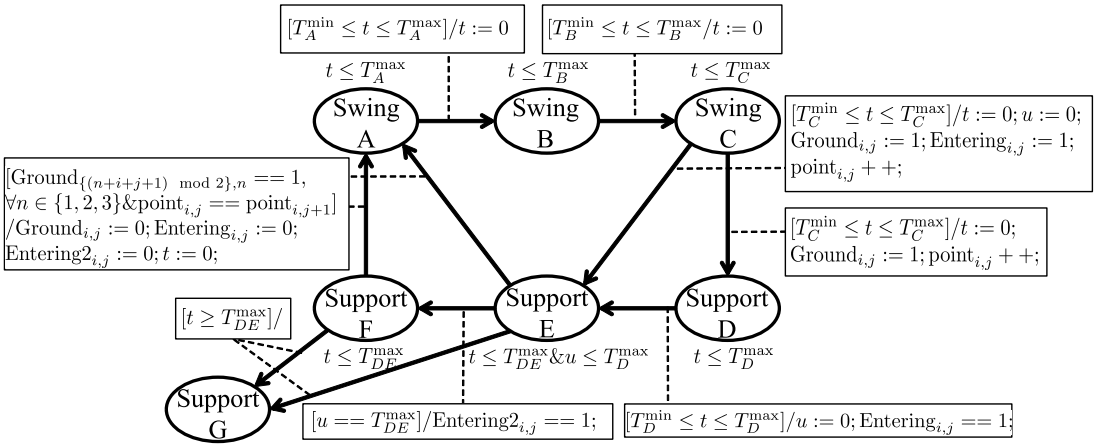
4.3.3　CTL(動作仕様)

モデル検査で使う動作仕様を定義する．本研究で使う「ロボットが歩行し続ける」という目標仕様は以下のものに対応する．

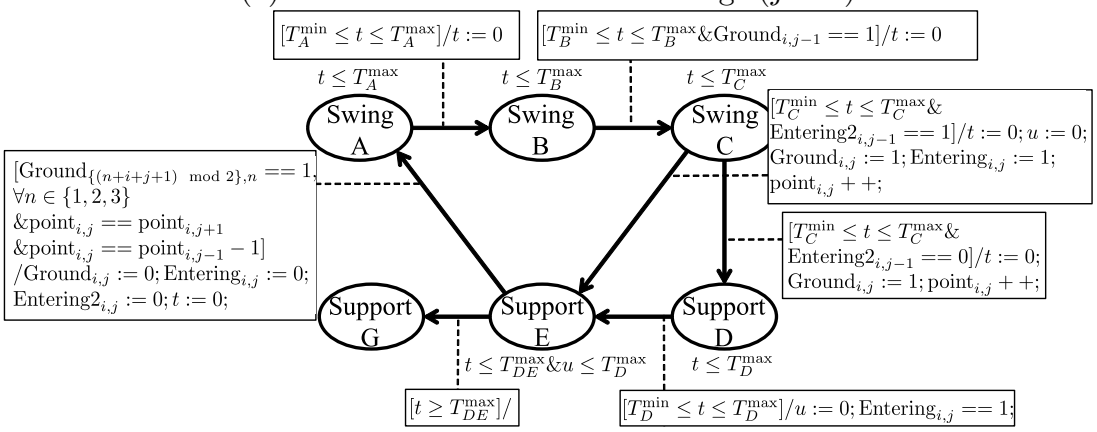
Spec.1:　デッドロックにならない．
Spec.2:　SupportG に遷移しない．
Spec.3:　接地している点の数が合計 3 個以上である (ただし，この点は先頭脚の SwingB でセンサを用いて計画した接地点を指す)．

1 つ目の仕様は前章より説明しているロボットの動作が止まっていないことを意味する．2 つ目は常に接地点交換するための仕様であり，接地点交換できなければ床に脚を引きずった状態で歩行に支障をきたす．また，Spec.2 以外にデッドロックが起こることがある．たとえば前後の脚について，前脚が SupportD，後脚が SwingB にいる状態で前脚が SupportD に長く接地している場合，後脚が接地できずに SwingB に滞在したままデッドロックすることがある．ほかにも SwingA や SwingB の滞在時間が長過ぎて，ほかの脚との協調が取れずにデッドロックになるようすが確認されている．3 つ目については接地点追従法が接地点を共有しながら前進する歩行であるため，たとえ脚が 3 本接地しているときでもロボットを支える接地点が 2 個になり転倒する場合がある．そこでこの仕様を足した．3 つの仕様は 3 章の CTL(Computational Tree Logic) の記述に従って以下のように記述できる．

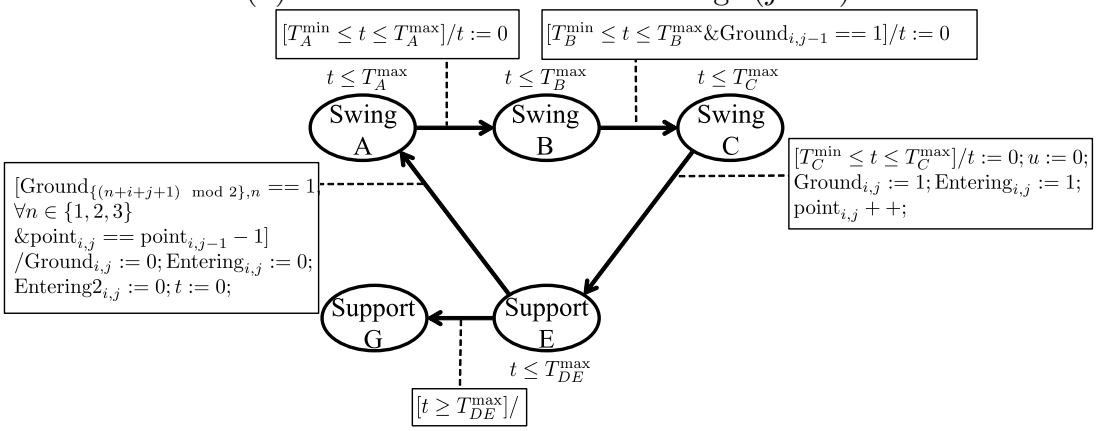
Spec.1:　$\mathrm{EFAG}\neg(\mathrm{Leg}[i][j].\mathrm{Swing}\alpha \cup \mathrm{Leg}[i][j].\mathrm{Support}\beta), \forall i \in \{0,1\}, j \in \{1,2,3\}, \alpha \in \{A,B,C\}, \beta \in \{D,E,F,G\}$.
Spec.2:　$\mathrm{AG}\neg\mathrm{Leg}[i][j].\mathrm{SupportG}, \forall i \in \{0,1\}, j \in \{1,2,3\}$.
Spec.3:　$\mathrm{AG} \sum_{\substack{i\in\{0,1\}\\ j\in\{1,2,3\}}} \mathrm{Ground}_{i,j} \prod_{l=1}^{j-1} (\mathrm{Point}_{i,j} \neq \mathrm{Point}_{i,l}\mathrm{Ground}_{i,l}) \geq 3$.



(a)Timed automaton of forward legs ($j = 1$)



(b)Timed automaton of middle legs ($j = 2$)



(c)Timed automaton of rear legs ($j = 3$)

Fig. 4.10　Timed automata of all legs

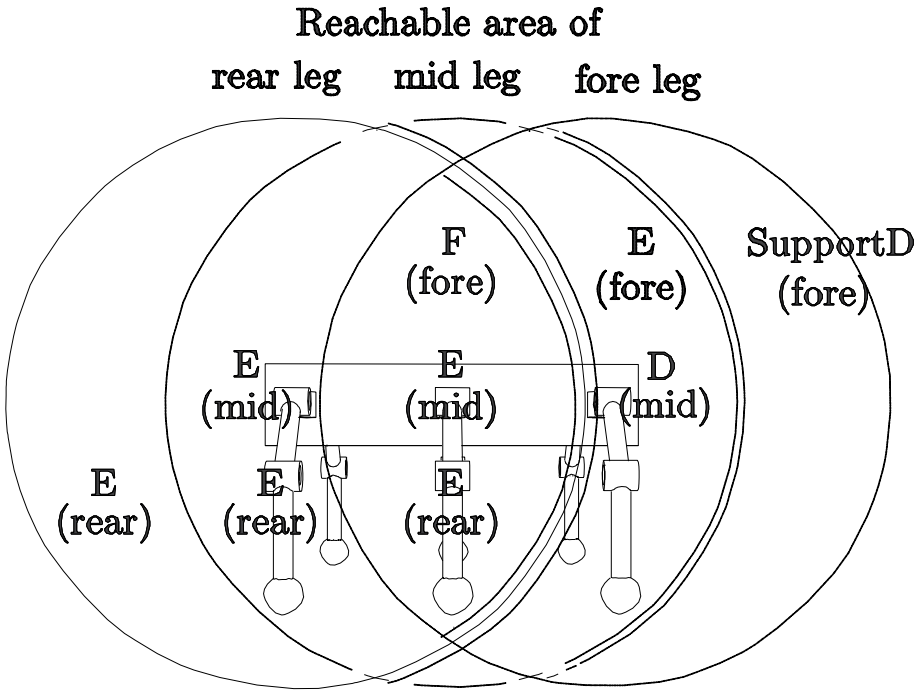


Fig. 4.11　Contact range of each leg & support node

4.3.4　導出パラメータの性質:先行研究との比較

それぞれのノードの滞在時間の最大値, 最小値 $T_X^{\max}, T_X^{\min}, X \in \{A, B, C, D, DE\} = \mathcal{A}$ の組み合わせはロボットのある動作のパターンを表わし，本章ではこれらを実機における制御のパラメータ (接地可能領域等) と区別して時間パラメータと呼ぶ．次節で述べる時間パラメータの導出方法は先行研究の野村ら [80] の手法を変更したものである．まず，野村らの手法で検証対象の時間パラメータを滞在時間の最小値と最大値ではなく，離散的な値としていた．たとえば，本章の SwingA での遷移条件は $T_A^{\min} \leq t_A \leq T_A^{\max}$ で記述されているが，野村らの手法では，$t_A = T_A$ と記述していた．これは，モデル検査を使って導出した時間パラメータをロボットの脚の運動に関するパラメータ (野村らの研究における実機のパラメータは接地点位置，脚先の移動速度，運動計画時間) に直接対応させるためである．たとえば，Fig. 4.12(a)(b) のような曲線で囲まれた領域が，システムの動作仕様を保証する時間パラメータの集合である場合を考える．このとき，野村らの手法で得られる時間パラメータは (a) における丸点 1 つ 1 つの集合になる．しかし，この手法の問題は各点間の滞在時間については検証結果が保証されていないことである．つまり，滞在時間の凸性が保証されないのである．そのためロボットの不整地歩行や本章で追加した姿勢制御のように逐次運動のパターンが変わり得る (取り得る時間パラメータが連続的に絶えず変化する) 場合には適用できない．

いっぽうで本章の時間オートマトンでは脚の軌道を限定せずに，時間パラメータを $T_X^{\max}, T_X^{\min}$ の区間で与えているため，得られる時間パラメータは点群ではなく，(b) のように最大値・最小値 $T_X^{\max}, T_X^{\min}$ で囲まれる領域全てを意味する．したがって，この領域内で許されるあらゆるクロック変数の変化が許容される．このクロック変数の変化に対応するのが，実際のロボットにおける多様な動作になる．また時間オートマトンに歩行環境の情報を含めた場合，膨大なパラメータ空間の探索が必要となるため，あえてロボットの脚の動作のみを表現し，さらに各時間を区間で与えることで環境情報やセンサの挙動に応じて動作時間が変わっても時間パラメータの区間内であれば，歩行が持続することが保証される．

4.3.5　時間パラメータの導出方法

時間パラメータ (最大値最小値) の導出方法を Algorithm 1 に記す．まず，1 行目に時間オートマトンの時間パラメータに代入する値の集合 $\{T_X^m\}$ を定義する．アルゴリズムでは，この値の集合 $\{T_X^m\}$ を変えながら時間オートマトンに代入・モデル検査の操作を

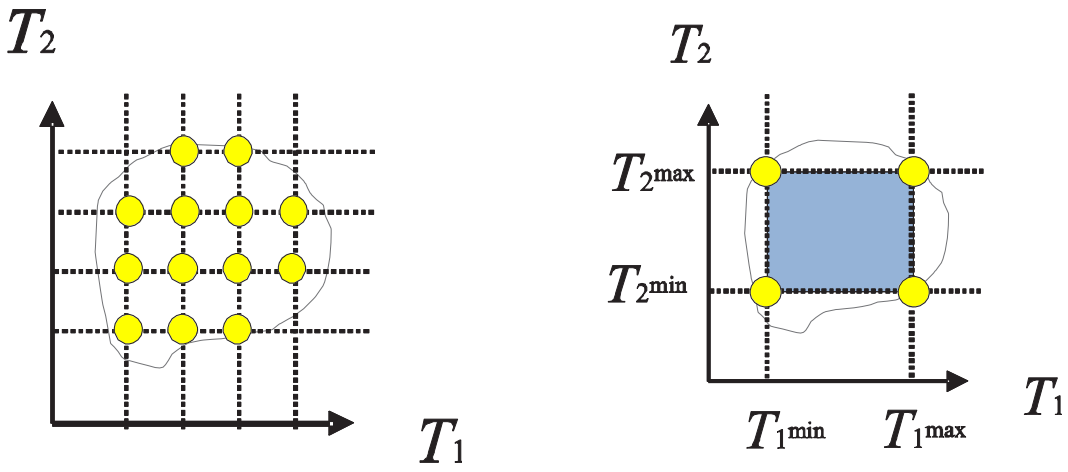
繰り返し，適切な $\{T_X^m\}$ の集合を最終的に導出する．その際，$\{T_X^m\}$ が代入された時間オートマトンは全ての動作仕様を満たしていなければならない．さらにこのアルゴリズムでは，各区間 $T_X^{\max} - T_X^{\min}$ の範囲をなるべく広げるものを最も適切な $\{T_X^m\}$ として選ぶ．そのため，各区間 $T_X^{\max} - T_X^{\min}$ の総乗を評価関数として与える．3〜14 行目では仕様を満たし，かつ評価関数を最大化する $T_X^{\max}$, $T_X^{\min}$ を導出する．ここで得られた時間は区間で導出されるので，たとえば Fig. 4.12(b) であればつぎの式で表現できる．

$$T_1^{\min} \leq T_1 \leq T_1^{\max}, T_2^{\min} \leq T_2 \leq T_2^{\max} \tag{4.15}$$

ここで時間オートマトンの特徴として，パラメータが状態間の遷移に伴う相対的な時間関係を表わしていることに着目する．(4.15) 式の時間パラメータを等しく倍にしたときを考えると，時間オートマトンでは各動作の時間の流れが等しく倍に長くなるだけでそれ以外の変化が生じないので，元のパラメータを代入したときと同様に仕様を満たすことができる．つまり，(4.15) 式と時間パラメータを倍にしても良いという利点から (4.16) 式のような規格化係数 k（正の実数）を用いて時間パラメータを導出することができる．

$$kT_1^{\min} \leq T_1 \leq kT_1^{\max}, kT_2^{\min} \leq T_2 \leq kT_2^{\max} \tag{4.16}$$

これが妥当であることを確認するため，Algorithm 1 の 13 行目以降は元の時間パラメータ $T_{X_{\text{temp}}}^{\min}$,$T_{X_{\text{temp}}}^{\max}$ に倍率 k を掛けて再度検証する．ここで規格化係数 k の値を変えるのはパラメータ区間の内部ではなく，パラメータ区間の両端 (最小値と最大値) の倍率を変えて検証するためであることに注意されたい．まず，15 行目のループで，k を 1〜1000 の間で更新し，15 行目で歩行に関する検証を行うが，一回ごとの検証中は k を固定している．本来は，$0 < k < \infty$ の範囲で値を変えて検証するのが望ましいが，現実的ではな



(a)Example of fixed parameters　　(b)Example of parameter range

Fig. 4.12　Derived parameters in two methods

い．そこで $0 < k < \infty$ の全範囲を調べる代わりに，本章では k の値を 1 から 1000 の間に限定してロボットが仕様を満たすかどうかを検証する．ここで，等価時間係数 k を任意の値に決定できることがわかれば，時間パラメータを実機の接地可能領域に対応させる際に，ロボットの性能や機構に基づいて k を決定し直すことができる．

また，モデル検査では，連続時間を状態にもつ時間オートマトンを 3.4.3 節の等価な有限状態オートマトンに変換した上で検証が行なわれる．本章のモデル検査に照らし合わせて改めてその仕組みを改めて説明すると，時間オートマトンの各ノードでの滞在条件および遷移条件に現れる時定数の組み合わせで連続時間を分割し，その時間区間を 1 つの状態として有限状態オートマトンにする．このように得られた有限状態オートマトン上で仕様を満たすかを検証することにより，各状態に割り振られた時間区間において元の時間オートマトンは任意の時間を取ることが許されるため，時間パラメータの凸性が保証されることになる．詳細は文献 [1] を参照されたい．

このアルゴリズムでも，対象とするシステムによっては動作仕様を保証する時間パラメータが見つからないケースもある．本手法は全探索を行っているため，動作仕様を満たす時間パラメータが見つからないということは，その集合がシステムの振る舞い上存在しないと見て良い．このような場合は，時間パラメータだけでなく対象とするシステムそのものを変更する必要がある．たとえば，本章で提案した動作仕様 (CTL) のうち，「3 点以上の支持脚が維持される」という動作仕様がどの時間パラメータでも保証されない場合には，システム自体にこの動作仕様を意図的に保証させるような制御要素を追加する必要がある．このシステムの修正は，3 章の形式手法の用途の 1 つに基づくものであり，システムの実装段階ではなく設計段階 (モデル検査のモデル上) でデッドロック可能性やエラーの早期発見を行い，システム開発のコストを削減できるという利点がある．

Algorithm 1 Finding of $T_X^{\min}, T_X^{\max}, X \in \mathcal{A}$

Define $\{T_X^m\} \equiv \{T_X^m | m \in \{\min, \max\}, X \in \mathcal{A}\}$
$T_X^{\min} \leftarrow 0, T_X^{\max} \leftarrow 1, \forall X \in \mathcal{A}$
Define function:$f \leftarrow \prod_{X \in \mathcal{A}} (T_X^{\max} - T_X^{\min})$
for $\{T_X^m\} \leftarrow 0$ **to** 1000 **do**
    Substitute $\{T_X^m\}$ into boundaries of transition time of timed automaton.
    Perform model checking for timed automaton by using UPPAAL.
    if all specifications are satisfied in model checking **then**
        if $\prod_{X \in \mathcal{A}} (T_X^{\max} - T_X^{\min}) > f$ **then**
            $f \leftarrow \prod_{X \in \mathcal{A}} (T_X^{\max} - T_X^{\min})$
            $\{T_{X_{\text{temp}}}^m\} \leftarrow \{T_X^m\}$,
        end if
    end if
end for
$Flag \leftarrow 0$
for $k \leftarrow 0$ **to** 1000 **do**
    $\{T_X^m\} \leftarrow \{kT_{X_{\text{temp}}}^m\}$
    Substitute $\{T_X^m\}$ into boundaries of transition time of timed automaton.
    Perform model checking for timed automaton by using UPPAAL.
    if all specifications are satisfied in model checking **then**
        $Flag \leftarrow 1$
    end if
end for
if $Flag == 1$ **then**
    $\{T_X^m\} \leftarrow \{T_{X_{\text{temp}}}^m\}$
else
    $\{T_X^m\} \leftarrow \{kT_{X_{\text{temp}}}^m\}$($k$ is arbitrary.)
end if

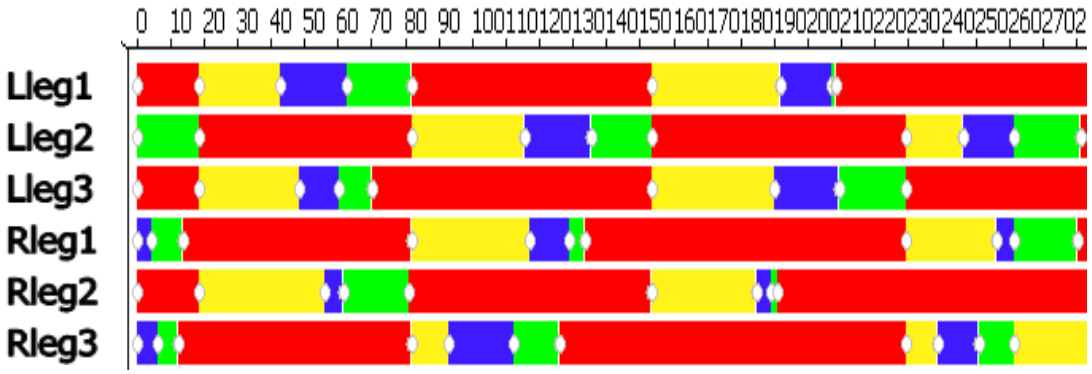
4.3.6　時間パラメータの探索結果

探索時に使用したツールを記述する．ここでは，モデル検査ツール UPPAAL のバージョンは Ver.4.1.8，使用 PC は CPU Intel Core$^{\mathrm{TM}}$i7-4770，3.40GHz，RAM64 ビットとした．この環境で前述の時間オートマトンと CTL に対して Algorithm 1 を適用したところ，計算時間は 3 時間 43 分 57 秒であった．時間パラメータの数は最小値・最大値含め 10 個であり，Algorithm 1 の 4 行目で扱う組み合わせは全部で 1000^{10} 組なので 1 組当たりの計算は十分に速いと言える．各脚の初期ノードは左脚前から順に「SwingB，SupportE，SwingB」，右脚前から順に「SupportE，SwingB，SupportE」に固定して，検証している．時間パラメータは (4.17) 式のように導出された．

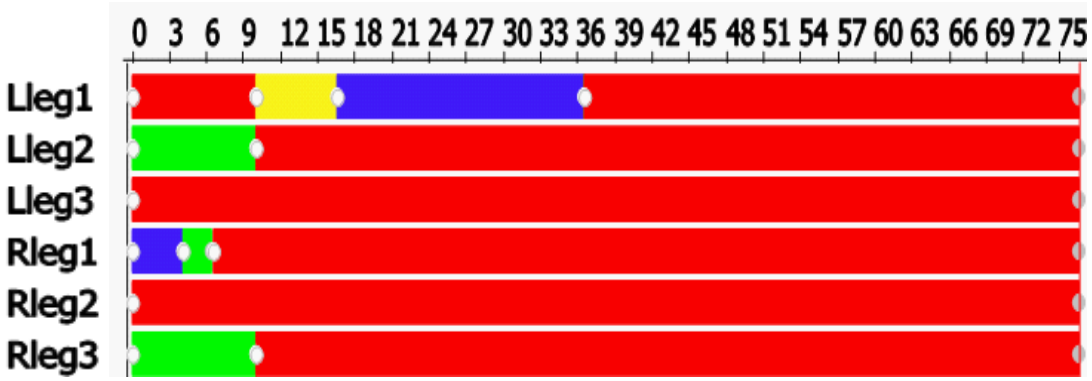
$$
\begin{gathered}
T_X^{\min} \le t_X \le T_X^{\max}, \\
\begin{pmatrix} T_A^{\max} & T_B^{\max} & T_C^{\max} & T_D^{\max} & T_{DE}^{\max} \\ T_A^{\min} & T_B^{\min} & T_C^{\min} & T_D^{\min} & T_{DE}^{\min} \end{pmatrix} = k \begin{pmatrix} 4 & 1 & 1 & 4 & 4 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix}. \qquad (4.17)
\end{gathered}
$$

ただし，k はロボットを動かす前に決める任意の正の定数である．たとえば，t_B について，ロボットを歩行させるときに $k = 2$ と決めれば t_B は歩行中に $0 \le t_B \le 2$ の範囲で状況に応じて自由に滞在時間を変化させるということになる．次元については $T_B^{\max} = 1$ 秒であり，k は無次元量である．時間オートマトンではロボットの具体的な幾何学モデルや環境モデルに関する具体的な情報を省略しており，各脚がどの動作を行なっているかを表現している．そのため，時間オートマトンの各ノードに対応する実機の状態や動作が存在し得る限り，この検証結果を適用することができる．たとえば今回使用したモデルは Fig. 4.10 のように最前脚の可到達域が 2 つ後ろの脚とも重なる (SupportF が存在する) 前提でモデルが作られているため，これを満たす 6 脚ロボットのみに適用できる．逆にこの前提が満たされないロボットの場合，SupportF に遷移できないので，今回の検証結果を適用できない．

また，この検証結果の時間パラメータを用いて，UPPAAL におけるランダムシミュレーションをさせ，ロボットのモード遷移の軌跡を確認した（Fig. 4.13）．導出した時間制約を用いた場合は，Fig. 4.13(a) のように 3 脚歩容を維持しながら歩行している様子が確認できる．いっぽうで，(b) では，全ての時間パラメータについて最小値 0，最大値 20(つまり各状態の滞在時間は $0 \le t_X \le 20$) とした場合に SupportG に遷移している様子が見られた．これは左前脚が遊脚している時間が長すぎて他の脚が地面を掻き切ったことでデッドロックが起こされている．導出した時間パラメータを制御に制御器設計に用いることでこのような状況は防げると考えられる．



(a)Simulation with derived time parameter



(b)Simulation without derived time parameter(Dragging leg tip)

Fig. 4.13 Random simulation in UPPAAL (Mode1:Yellow, Mode2:Blue, Mode3:Green, Mode4:Red)

4.4　時間パラメータの実機モデルへの対応

前節までに求めた時間パラメータと実機モデルの歩行に関する変数との対応関係を取り，接地可能領域を導出する．

4.4.1　実機モデルとの対応関係

実機モデルで扱う脚先の座標と速度，接地点計画時間の変数を以下に示す．ここでは各脚の根元に原点を置いた座標系 $\Sigma_{i,j}$ における座標および速度を扱う．

$P_A(x_A, y_A, z_A)$　：　遊脚相での中間点（SwingA の目標点），
$P_C(x_C, y_C, z_C)$　：　接地点（SwingC の目標点），
$P_{DE}(x_{DE}, y_{DE}, z_{DE})$：　接地相の離脱点，
v_A, v_C, v_{DE}　：　SwingA，C，SupportD〜F における脚先の速度，
t_B　：　接地点計画時間．

各座標間の脚先の移動を Fig. 4.14 のように等速直線運動に近似できると仮定したとき，次のような対応関係が得られる．

$$\mathrm{d}(P_A, P_C) = t_C v_C, \tag{4.18}$$
$$\mathrm{d}(P_C, P_{DE}) = t_{DE} v_{DE}, \tag{4.19}$$
$$\mathrm{d}(P_{DE}, P_A) = t_A v_A. \tag{4.20}$$

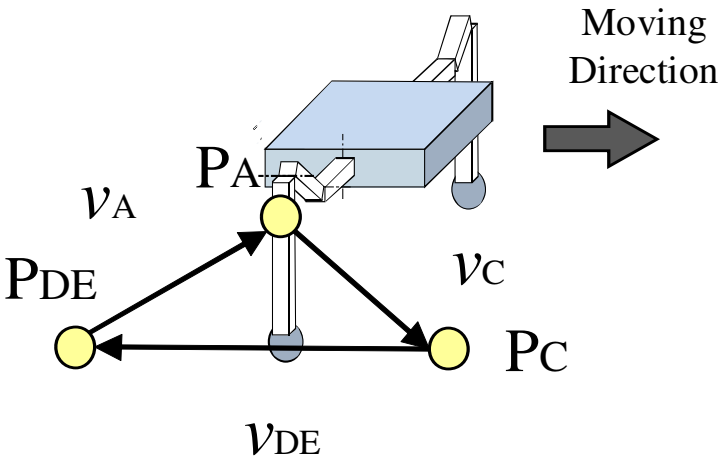


Fig. 4.14　Leg tip position and velocity in Σ_1^s

ここで，$\mathrm{d}(P_1, P_2)$ は等速直線運動の始点 P_1 と終点 P_2 間の距離を表す．また，本章では以下の手順で接地可能領域を導出する．

- (4.17) 式の時間パラメータのうち，実機で既に範囲が既知であるものを基準に等価時間係数 k を決定する．
- 残りの時間パラメータの区間を満たすような脚先の速度と移動距離の範囲 ((4.21)〜(4.23) 式) を導出する．
- (4.21)，(4.23) 式とロボットの構造に基づく可到達域 (4.2),(4.3) を満たしつつ接地可能領域を最大化する脚先の速度を導出する．

4.4.2　等価時間係数 k の決定

(4.18)〜(4.20) 式の対応を行う前にまず，実機の性能を基準に等価時間係数 k を求める．具体的にはセンサを用いた接地点探索時間 t_B，モーターの最大速度，リンク長などのロボットの構造がそれに相当する．本章では $t_B \leq 1\mathrm{s}$ であることが事前にわかっているものとして k の値を決める．つまり $T_B^{\max} = 1$ とする．このとき，(4.17) 式より $k = 1$ が求まる．k を固定したとき，(4.17)〜(4.20) 式から v_A，v_C，v_{DE} がいずれも 0 でないとき，(4.21)〜(4.23) 式を得る．時間の観点だけで見れば，この範囲で座標と速度を選ぶことで歩行が持続する．

$$0 \leq \frac{\mathrm{d}(P_A, P_C)}{v_C} \leq 1, \tag{4.21}$$

$$0 \leq \frac{\mathrm{d}(P_C, P_{DE})}{v_{DE}} \leq 4, \tag{4.22}$$

$$1 \leq \frac{\mathrm{d}(P_{DE}, P_A)}{v_A} \leq 4. \tag{4.23}$$

4.4.3　接地可能領域の導出

P_C, P_{DE} の存在領域を Ω_c, Ω_{DE} とする．(4.21)〜(4.23) 式より，接地時の振る舞いに最も影響する座標 P_C，P_{DE} の存在領域 Ω_c, Ω_{DE} を最大化するように速度 v_A，v_C，v_{DE} を決定する．まず，座標 $P_A(L_1 + |L_2 - L_3|, L_1 + L_3, 0)$ と速度 v_A，v_C，v_{DE} を動作中は一定であるものとする．これにより，(4.21)，(4.23) 式はそれぞれ P_A を中心とする P_C，P_{DE} の球領域として扱える．Ω_c, Ω_{DE} の存在領域を広げるとき，これと元の可到達域による領域の和集合を最大化すれば良いので，これを満たす速度 v_A，v_C を決定する．ただし，最大化には一般化簡約勾配 (GRG) 法 [100] を用いた．

$$\begin{aligned} \text{Find } v_C = & \arg\max_{v_C} \iiint_{A_C} dxdydz, \\ A_C = & \{P_C \mid (4.2), (4.21)\}. \end{aligned} \tag{4.24}$$

$$\begin{aligned} \text{Find } v_A = & \arg\max_{v_A} \iiint_{A_{DE}} dxdydz, \\ A_{DE} = & \{P_{DE} \mid (4.2), (4.3), (4.23)\}. \end{aligned} \tag{4.25}$$

いっぽう，(4.22) 式は P_C，P_{DE} の両方を含んでいるため，動作中の全ての時間で満たすべき領域を表現することはできない．そこで，(4.22) 式の動作に該当する接地相の速度 v_{DE} については，複数の v_{DE} から後出のシミュレーションコースを完走できたものを候補と

して選択した．得られたる v_{DE} については環境に依存したものになっている．そのため後出のシミュレーションコースについて環境依存性が無いように 3 つの区域を設定したシミュレーションの環境に依らない v_{DE} の導出方法は以降の課題となった．以上の操作で得られた P_C，P_{DE} の存在領域 Ω_C, Ω_{DE} は Figs. 4.15，4.16 の通りである．ただし，各動作の速度は $v_A = 0.185\text{m/s}$，$v_C = 0.217\text{m/s}$，$v_{DE} = 0.02\text{m/s}$ であり，$L_1 \leq |L_2 - L_3|$ を満たしているとし，Fig. 4.15(a) 中に値を記した．これを踏まえて，各リンク長と L の長さは Table.5.3，Fig. 4.17 と同じにした．

4.4.4　シミュレーションによる接地可能領域を用いた歩行の確認

最後に物理シミュレータにこれまで得られた接地点 P_C 等の座標を代入し，モデル検査によって得られた接地可能領域を含む実機パラメータで動作仕様が保証されるかを確認する．確認の方法として Fig. 4.18 のような 3 区画に分けたシミュレーションコースを，Table 5.3 と Fig. 4.17 に示した 6 脚ロボットを右から左のゴールへ完走させる．各地形の特徴は右から順に傾斜 10 度の斜面，小さな凹凸のある不整地，胴体の高さほどの急な段差であり，ロボットが転倒する要因は各環境によって異なる．

ただし，ロボットに自分の胴体の傾きを捉える姿勢センサと周囲に赤外線を照射してその反射光で地形を読み取る深度センサと姿勢センサを搭載しているとし，4.2 節で追加した接地点探索，姿勢制御を自律的に行うものとする．シミュレーションには以下の 2 パターンを用意し，2 種類の実験を行った．

Case.1　接地可能領域を可到達領域全体にしたもの（本手法を適用しなかったもので，可到達域全体から接地点探索が行なわれる）．

Case.2　モデル検査によって座標の存在領域を決定したもの．

Case1 は先行研究 [81] にしたがってパラメータ設計をせずに，接地点追従法によりそのまま歩行させている手法であり，接地可能領域を可到達域に置き換えて接地点計画をさせた．いっぽう，Case2 はパラメータ設計をした本手法である．この 2 つのケースの転倒要因の中にはデッドロックが起こる，3 点以上の支持が維持できない等の本章の CTL で定義した動作仕様を保証できない場合も予想されるが，それ以外にも時間オートマトン上で記述できなかった胴体の傾きやダイナミクス等も影響すると考えられる．このように時間オートマトンに記述できない転倒要因に関しては本手法では完全には防ぐことができない．そこで，Case2 についての時間オートマトンからの対応の妥当性については，主にデッドロック回避等の動作仕様が保証できるか否か，接地可能領域を含む実機のパラメータの設計に適しているかの観点から考察で述べる．

そのために，まず Case2 について歩行実験中の各モードの最長継続時間と最短継続時間を測定し，それぞれをモード持続時間の上限値，下限値とし，(4.17) 式を満たしているかを確かめた (Table 4.2)．つぎに，Case1，2 をそれぞれ 20 回ずつ違う経路で歩行させ，20 回中完走できた回数と完走できなかったときにモデル検査の仕様 (5.3 節 Spec.1〜3) を満たせなかった回数 (Table 4.2)，すべての試行で選んだ接地点座標 (Fig. 4.19) を記録した．ただし，Fig. 4.19 の (b)，(c) では接地点の立体的な分布を示すため，z 軸に沿って分布点の濃淡を変えて表示した．各 Case で仕様を満たせなかった回数を比較するのは，時間パラメータから接地可能領域への対応関係が適切であることを確かめるためである．前節で導出された時間パラメータは歩行が成立するための条件 Spec.1〜3 を満たすものであり，この時間パラメータと接地可能領域の対応が不適切であったなら，提案手法である Case2 でもそれらの仕様を満たせない．逆に環境からの外乱が加わっても Case2 で Spec.1〜3 を満たしているならば，対応関係が適切であると評価できる上に，導出された時間パラメータも適切であったと確認できる．

4.4.5　シミュレーション結果

まず，各モードの滞在時間の範囲は Table 4.2 の通りである．この時間幅は (4.17) 式を満たしているので時間オートマトンからの対応が妥当であったと言える．つぎに選んだ接地点の位置は，Fig. 4.19 の通りであり，Case.2 ではモデル検査によって得られた領域内で選ぶことができており，ここでも時間オートマトンからの対応が適切に行われていたことがわかる．最後に 20 回の歩行の結果は Table 4.3 の通りである．まず完走できた回数について比較すると Case1 は全て完走することが出来なかったが，Case2 ではほとんどの試行で完走できた．そして仕様についても Case2 はほとんどの仕様を満たしたのに対して，Case1 では全て Spec.1 を満たせずに転倒した．座標の存在領域を絞る必要があったとわかる．また，本実験では歩行環境に応じて急カーブ状の胴体軌道を生成し，旋回することがあった．特に本手法を適用した Case2 では，旋回後も歩行が持続するようすが見られ，運動の多様性を保証していることが確認できた．

4.4.6　考察

ただし Case2 にも完走出来なかったものが見られた．この要因の 1 つは時間オートマトンでは環境や胴体の傾きはモデル化していないことである．時間オートマトン上では与えられた領域内に必ず足場が含まれ接地できるという前提があったが，実機モデルでは胴体の姿勢が極端に後方に傾いた場合に接地可能領域内に足場が含まれない場合がある．その結果，Case2 においても接地点探索中に動きが止まるようすが見られた．ほかにも，Spec.3 が成り立つとしても支持多角形内に胴体の重心投影点が存在することは保証できないため，胴体の姿勢が極端に傾くと転倒することがあった．本章で求めた接地可能領域にロボットの脚先が存在することは，実際にロボットが歩行できることと等価ではなく，さらに胴体の姿勢や環境情報を考慮した制御が必要であることを示唆している．これらの制御の導入は今後の課題である．

Table 4.1　Charaferestics of robot

	Length[m]	width	height	Mass[kg]	Inertia[kgm^2]
Link1	0.10	0.04	-	0.80	0.96
Link2	0.12	0.04	-	1.26	0.50
Link3	0.25	0.04	-	2.62	0.91
Body	0.50	0.20	0.05	10.80	0.18

Table 4.2　Range of mode time in simulation

	t_A	t_B	$t_B + t_C$	t_D	t_{DE}
$T_i^{\max}$[sec]	2.54	0.95	0.97	3.65	3.82
$T_i^{\min}$[sec]	1.07	0.01	0.09	0.04	0.04

Table 4.3　Simulation result

	Reached to goal	Unsatisfied Spec.1	Spec.2	Spec.3
Case.1	0/20	20/20	15/20	1/20
Case.2	18/20	2/20	2/20	0/20

4.5　まとめ

接地点追従法によって制御される，6 脚多脚移動ロボットの各脚の接地可能領域の設計法を提案した．これはロボットの動作を時間オートマトンにより表現し，モデル検査を利用して動作仕様を満たす時間パラメータの範囲を導出する手法であった．そして，時間パラメータから各脚の接地可能領域を導出し，物理シミュレータで検証結果を確認しても歩行が持続した．実機での検証も今後の課題の 1 つである．

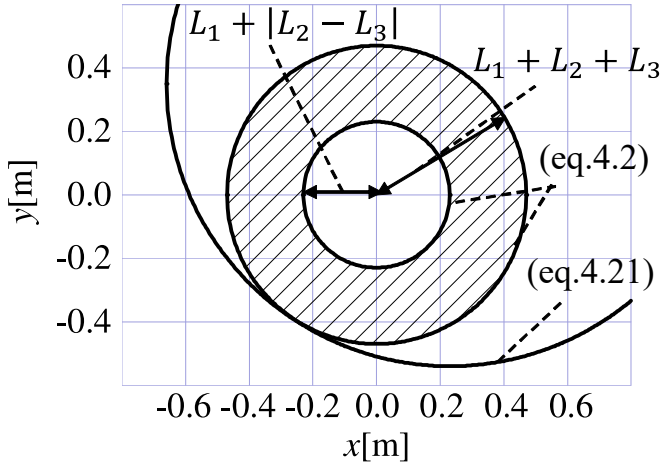
また，本章においてはモデル検査でのパラメータ導出の後にシミュレーションを用いてさらに検証したが，代わりに実ロボットを用いた検証も可能である．しかし，ハードウェアではシミュレーションからさらに考慮すべき特性がある．特に，制御コントローラの計算や通信に伴う遅延とアクチュエータの最大出力に起因した追従特性は，脚の各状態における経過時間に大きく影響をもつものであり，モデル検査からハードウェアへのパラメータの変換においては陽に考慮すべきである．これに対して，たとえば，状態の経過時間の中から制御コントローラの遅延を割り引く，脚先の移動速度に一次遅れの追従特性を入れることにより対応が可能だと考えられる．実ロボットへの適用は次章への課題である．

つぎに，実環境作業ロボットにおいては，不整地登破能力とともに，耐故障性も課題で

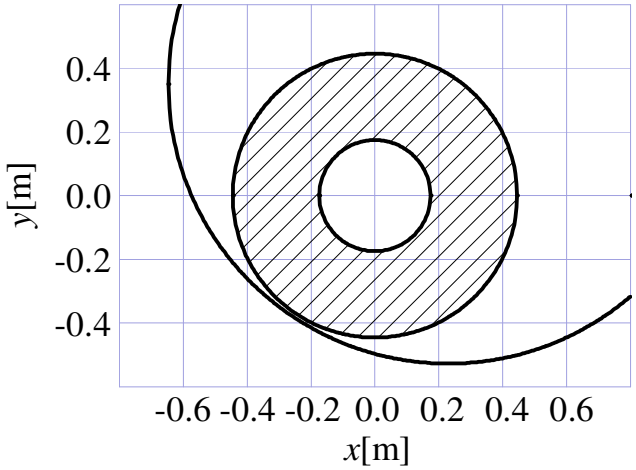
ある．特に 6 脚移動ロボットは脚の切断などの故障に対しても歩行を維持する性能が期待される．本提案手法を故障に対応させるには，まず故障時の脚の制御法を組込み，さらに各故障に対応したロボットの挙動を表わす時間オートマトンを導出する必要がある．本提案手法の耐故障性への適用も今後の課題である．

さらに，本提案手法の他制御手法への適用を考えたとき，接地点追従法と同様の挙動を実現できる Walknet[62] への適用が考えられる．その場合，ロボットの挙動を表現する時間オートマトンは本章と同様のものが使えると考えられるが，モデル検査から得られる時間パラメータとニューラルネットワークのパラメータの関連付けには検討が必要である．本章におけるモデル検査を使った手法が扱えるモデルは，時間オートマトンと呼ばれるハイブリッドオートマトンの一種であり，状態の挙動が単に時間経過のみで表現されている．状態変化の背景にある慣性や重力，衝突，変形などの物理的特性は無視されている．たとえば，今回扱ったモデルは脚間の衝突，胴体の位置，姿勢の挙動を考慮していない．したがって，このような特性の再現，検証を保証できない．そのため，モデル検査を使った実機モデルの制御に関するパラメータの導出に加えて，物理シミュレーションもしくは実ロボットを用いた実世界環境下での検証も必要となる．

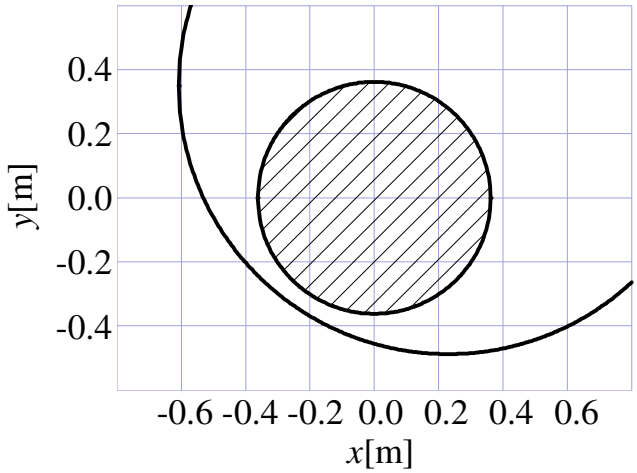
ただし，実機実験または物理エンジンによるシミュレーションを使った実機モデルのパラメータ探索を行う場合でも，本手法のようにモデル検査による仕様を満たした時間パラメータの範囲があらかじめ導出されている場合は，この時間パラメータに対応する領域を実機モデルのパラメータ必要条件として捉え，さらにダイナミクスや胴体の傾き等を加味した実機のパラメータ探索を行うことができる．これによって，通常の物理エンジンや実機のみを使った場合に比べて実機のパラメータ探索が容易になると期待できる．



(a) $z = 0$[m]

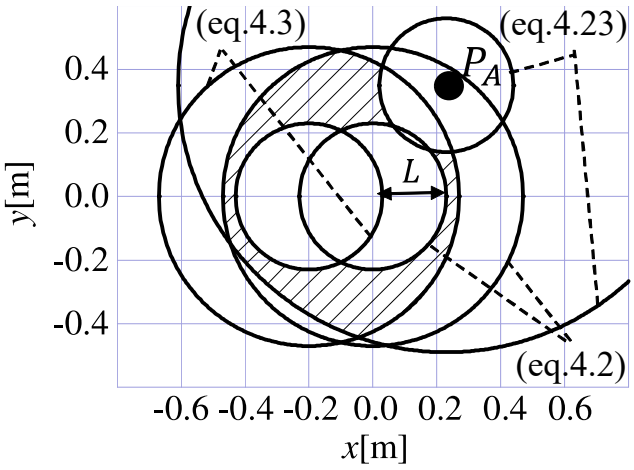


(b) $z = -0.15$[m]

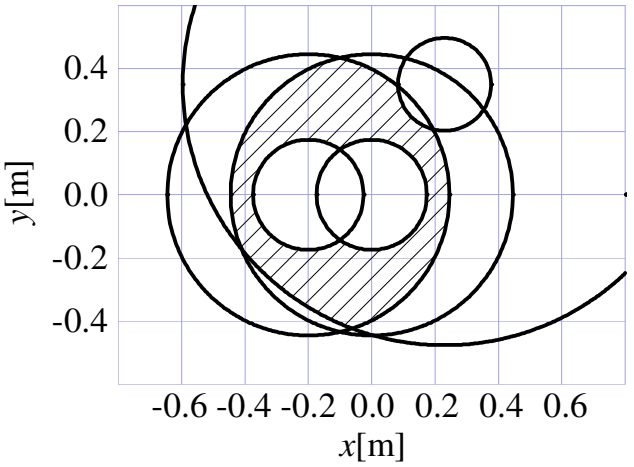


(c) $z = -0.30$[m]

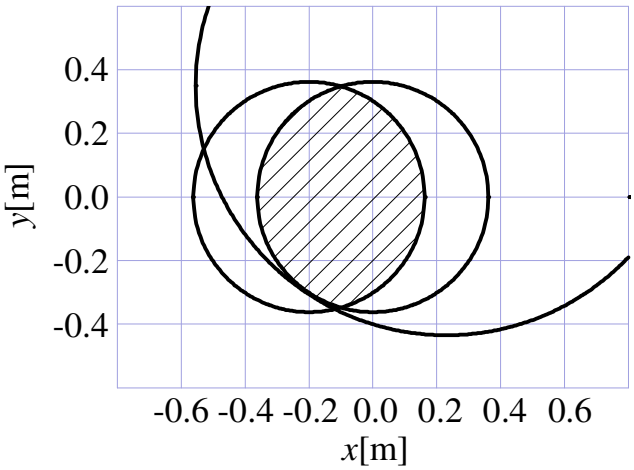
Fig. 4.15　Range of P_C



(a) $z = 0$[m]



(b) $z = -0.15$[m]



(c) $z = -0.30$[m]

Fig. 4.16　Range of P_{DE}

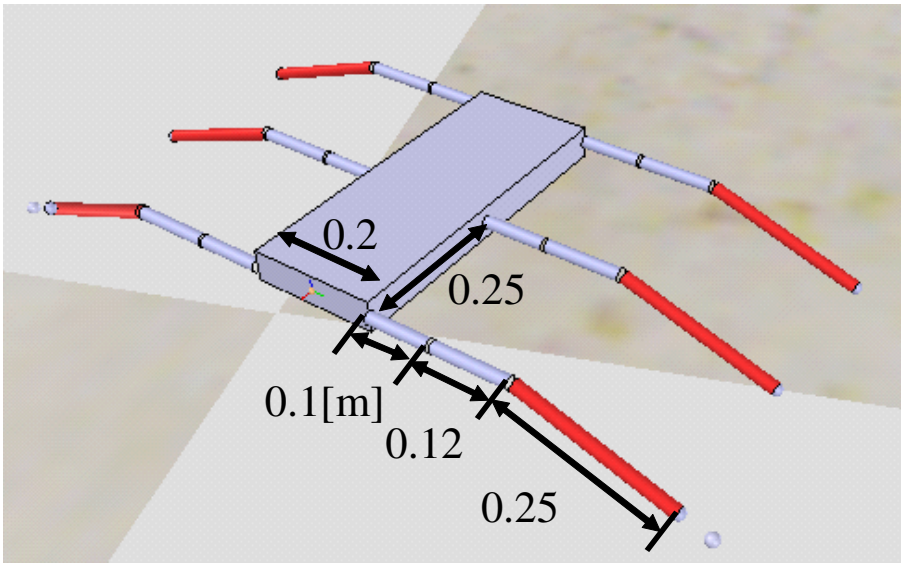
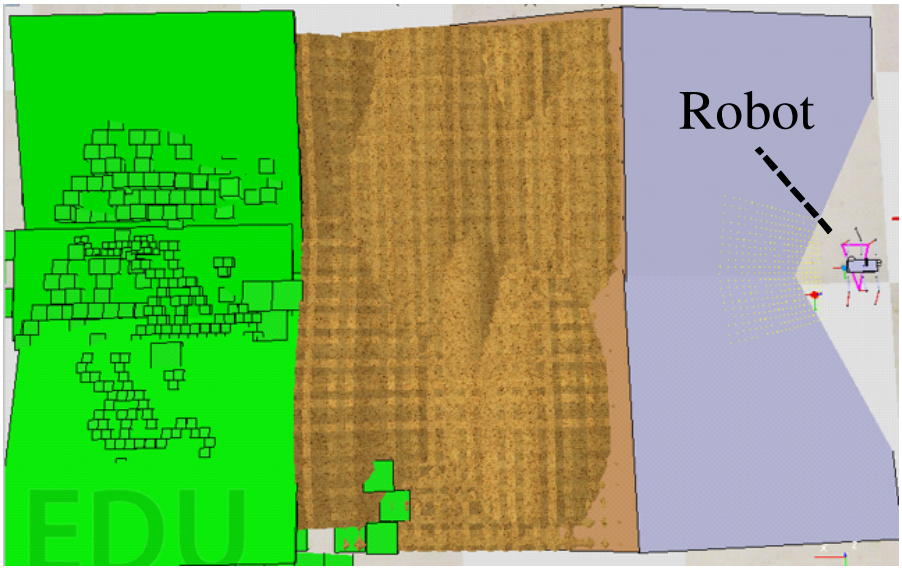
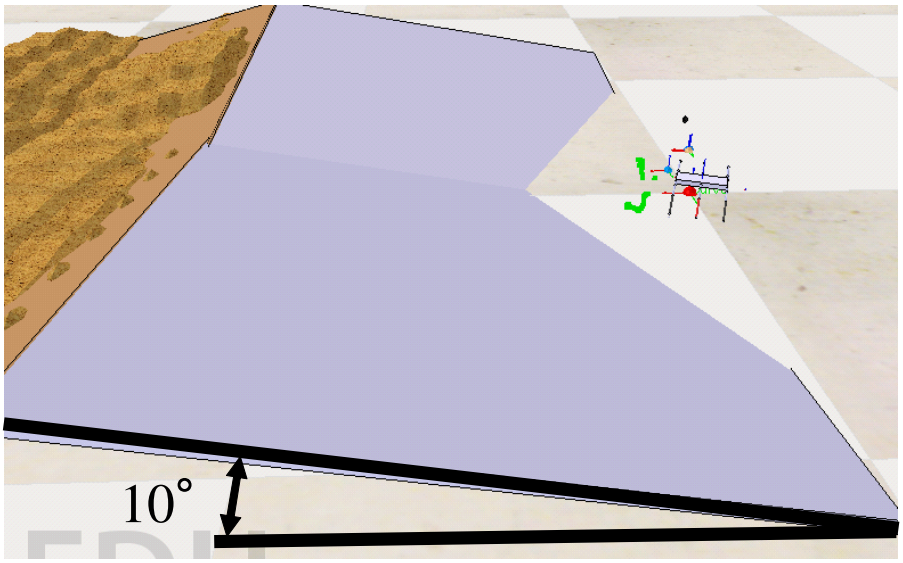


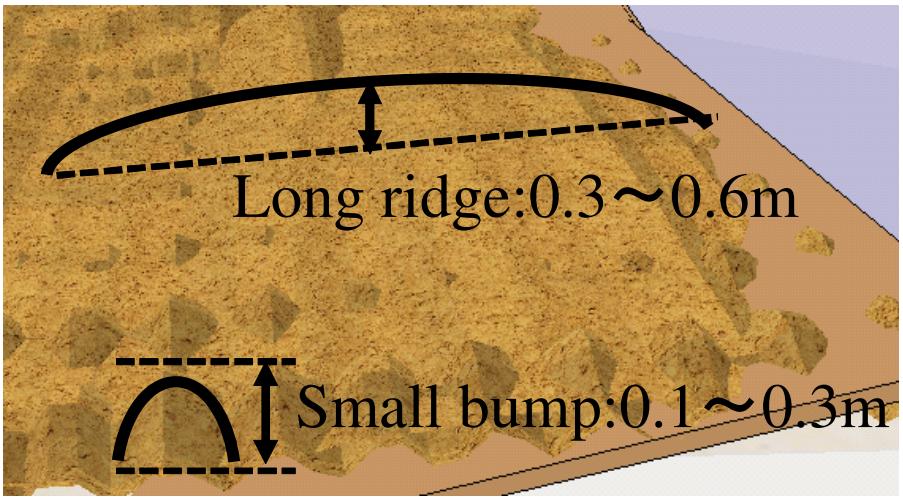
Fig. 4.17　Physical model of hexapod robot



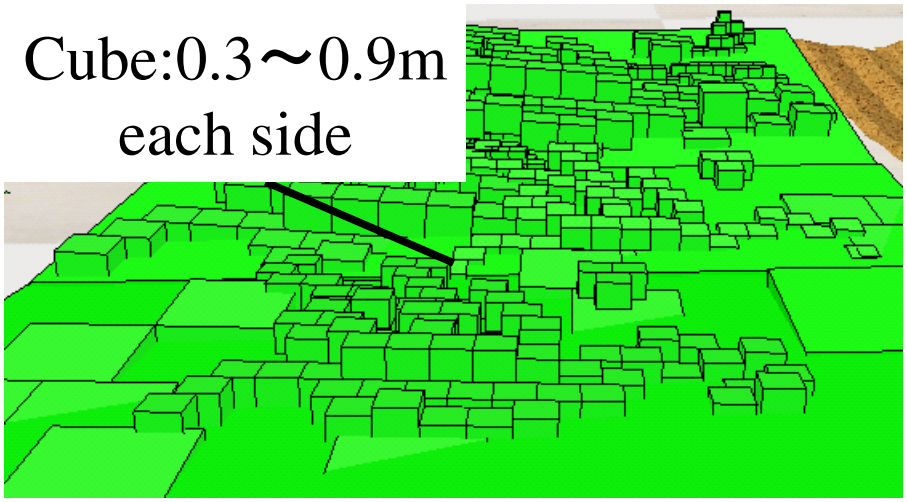
(a)3 types of ground



(b)1st ground: flat slope

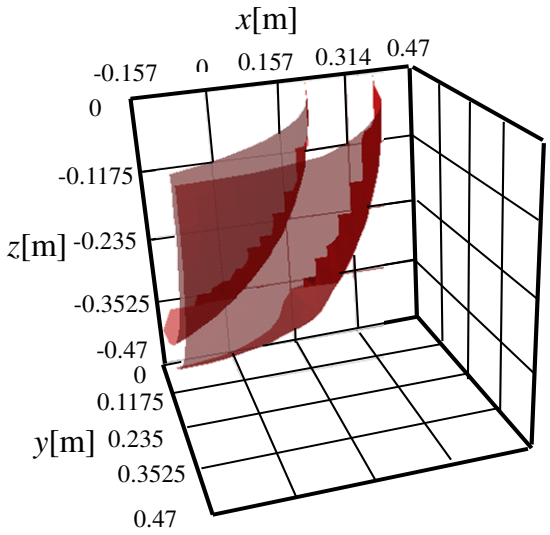


(c)2nd ground: small bumps

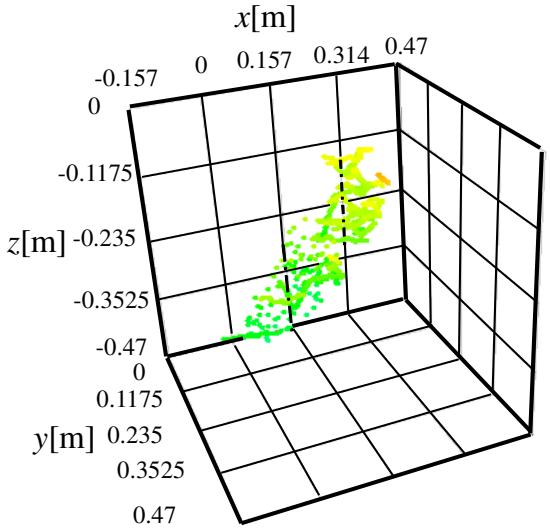


(d)3rd ground: high cubes

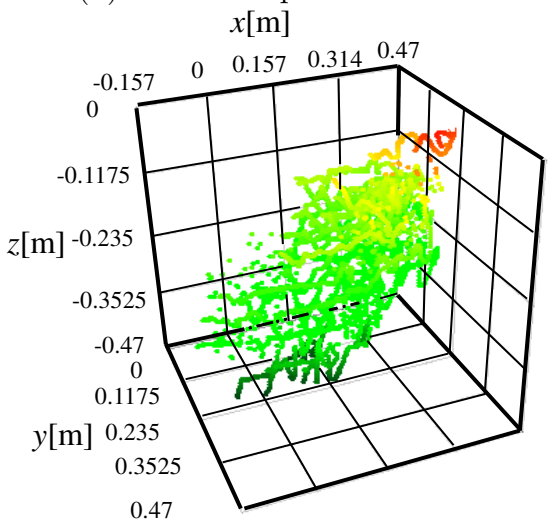
Fig. 4.18　Simulation Coarse



(a)Border of P_C range



(b)Simulation plots in Case2



(c)Simulation plots in Case1

Fig. 4.19　Contact points in simulation

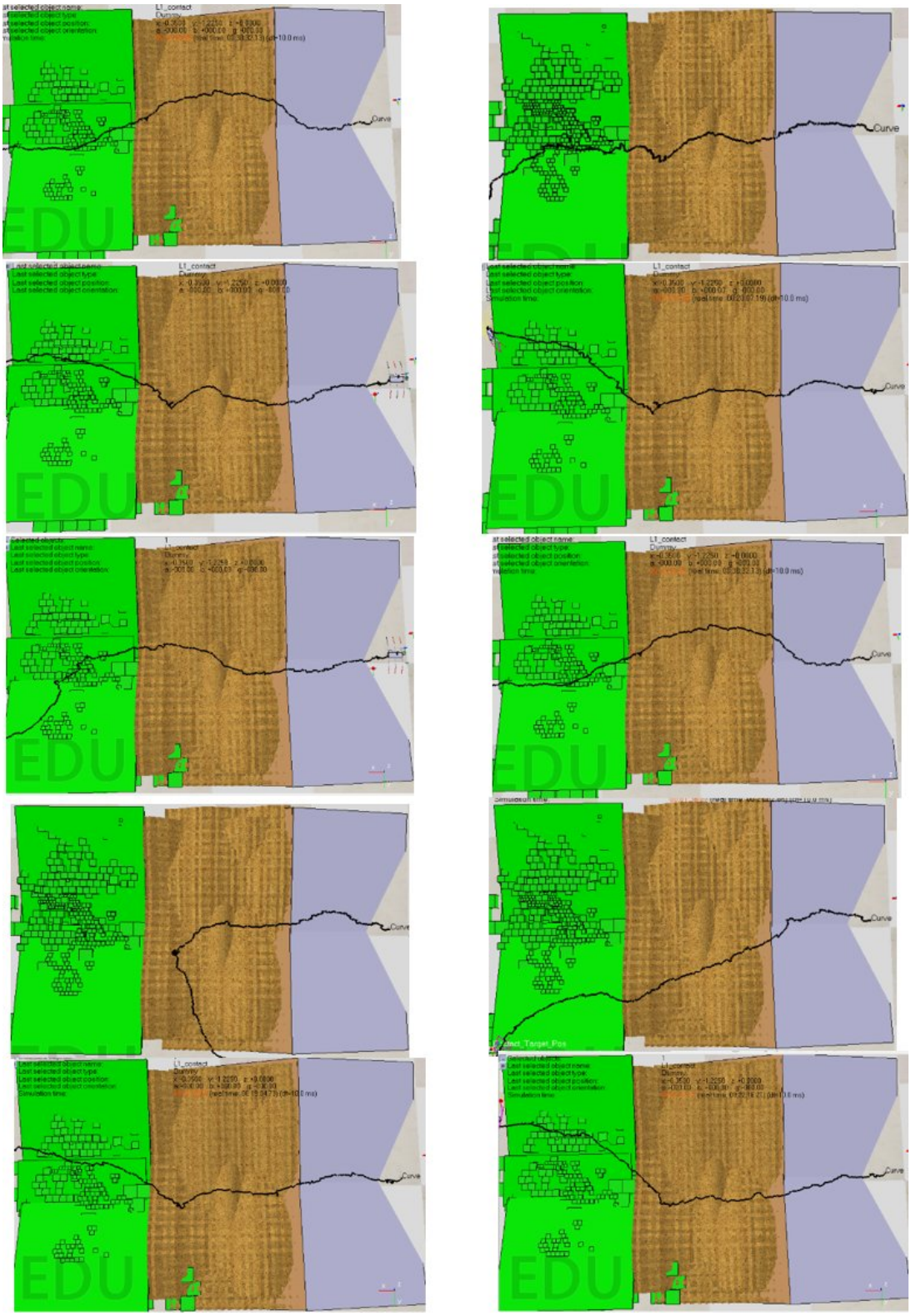


Fig. 4.20　Walking trajectory in simulation

第 5 章

制約充足型統合制御「Timekeeper 制御」による制御の統合

5.1　はじめに

4 章では，6 脚ロボットの制御器設計に必要な制約を接地点追従法における各状態の滞在時間の制約 (時間パラメータ) として導出した．そのつぎに時間パラメータを接地点追従法における制御パラメータ (接地可能領域等) に対応させてロボットの制御に用いた．この手法をとったのは，4 章の研究の当初の着眼点がロボットの制御器設計 (制御要素同士の干渉を防ぐ等) を目的とするものではなく，歩行制御の制御器におけるパラメータを数学的根拠に基づき導出することにあったからである．したがって，4 章では (4.17) 式の滞在時間の制約 (この章では，時間制約と呼ぶ) のことを実際の接地点追従法で用いる物理的なパラメータ (接地可能領域等) と対応させて「時間パラメータ」という言葉で説明していた．

しかし，この方法をとったことでロボットの歩行制御における 3 つの新たな問題が生じた．1 つは，時間制約を制御パラメータに対応させる都合上脚の遊脚運動を等速直線運動に近似できると考え，制御においても脚を現在の脚先から目標地点まで一定速度で追従させていた．しかし，不整地環境上では脚先は障害物に引っ掛かるなどして，等速直線運動に近似できない動きをすることもあり，想定した滞在可能時間と接地可能領域の対応関係が成立しなくなることがある．

2 つ目は，接地可能領域が脚の可到達域よりも狭い範囲で与えられたことである．脚の接地可能領域が狭いと，不整地上で狭い範囲から安全な足場を選択しなければならないので，安定した歩行が難しくなることが考えられる．しかし，実際には時間制約から接地可能領域への写像を行なわず，直接時間制約を保証する手法を提案することで，接地可能領

域を可到達域全体に広げることができると考えられる．

3 つ目は，(4.17) 式の縮尺に相当する等価時間係数 k を固定した前提で歩行制御のパラメータを決定したことである．この等価時間係数は全ての状態の滞在時間の最大値・最小値にかかっているため，k の値は各脚の歩行周期に大きく影響する．k を固定した場合，各状態の滞在時間だけでなく，歩行周期の振れ幅も限定される．ロボットの歩行周期は移動する不整地環境に応じて，任意に変えられることが望ましい．このような環境依存での逐次的なパラメータチューニングの研究も脚移動ロボットの歩行研究ではなされている．たとえば，CPG では特定環境で脚の位相を調整することで，適応的に歩行を維持する手法が提案された [35], [51]〜[56]．ほかにも歩行環境に合わせて，遊脚相と接地相のデューティ比を変化させ，歩容を変化する手法 [57] や，歩行環境に応じて一歩当たりのエネルギー消費が最小化されるように歩行周期を更新する手法 [58] も提案されている．これらの手法のように等価時間係数 k もロボットの状況・歩行環境によって変える必要がある．そこで，本章では時間制約に関する制約充足問題を解きつつ，ロボットの多様な運動を保証する Timekeeper 制御を提案し，6 脚移動ロボットの不整地突破能力のさらなる向上を図る．

本章ではまず，5.2 節で 6 脚移動ロボットの構造と接地点追従法の改良点を説明する．つぎに 5.3 節で 4 章の時間制約を保証させる制約充足の手法とそれを実現するための制約充足型統合制御「Timekeeper 制御」のアルゴリズムを説明する．5.4 節では Timekeeper 制御の細部の手法を決定する上で，実施した物理エンジン上のシミュレーションと実機実験を順番に述べる．これによって，本手法による動作仕様の保証と制御器内の制御要素同士の干渉が厳密に防がれることを確認する．

5.2　本章で用いる接地点追従法とロボットの構造

本章で用いる接地点追従法では主に制御モード 4 から 1 への遊脚条件の方法を 4 章から引き継ぎ，接地点計画と姿勢制御を新しく提案した．ロボットの構造の変化を前提に 1 つずつ述べる．

5.2.1　ロボットの構造

本章では，Fig. 5.1 のように，胴体に対し放射状に 6 つの脚を配置したロボットを考える．これは，ロボットの旋回運動を考慮して設計しており直方体構造より接地点を固定した際の旋回角度が広がると考え，構造を変えた．脚の番号は 4 章を引き継ぐ．進行方向側の二脚を先頭脚とする．この先頭脚は左右の識別番号を右側で 0，左側で 1 とし，先頭か

ら j 番目の脚を $\mathrm{Leg}[i][j](i = 0, 1, j = 1, 2, 3)$ とする．このとき，$\mathrm{Leg}[i][j]$ の左右反対側に相当する脚は $\mathrm{Leg}[1 - i][j]$ となる．ただし，ここで指す左右とは，進行方向に対するロボットの軌道を軸とした左右のことをいう．また，以降の説明で述べる前脚とは $\mathrm{Leg}[i][j]$ に対する $\mathrm{Leg}[i][j - 1]$，後脚は $\mathrm{Leg}[i][j]$ に対する $\mathrm{Leg}[i][j + 1]$ を指す．そして，$\mathrm{Leg}[i][j]$ の付け根に原点を持つ直交座標座標系を $\Sigma_{i,j}$，脚先の可到達域を $\Omega_{i,j}$ と定義する．各本稿で用いる脚先座標は $\Sigma_{i,j}$ 上のものとする．脚の構造は 3 リンク 3 自由度とであるが，脚の配置は放射状にするだけでなく，脚の付け根の関節を下方に 45 度傾けている．これは，各脚先の可到達域の重なり部分を Fig. 5.2 の $\Omega_{0,2} \cap \Omega_{0,3}$ のように胴体より下方の領域で広くするためである．これにより前後の脚同士で接地点交換しやすくなる．

5.2.2　4 章から引き継ぐ制御要素

1. 姿勢制御:制御モード 4 における脚の動作 (4.2.3 節)
2. 制御モード 4〜1 への遷移条件 (4.2.4 節)

4 章から引き継がれる制御要素は上記のものである．ただし，4.2.3 節と異なる点として，胴体の目標姿勢角 $\hat{\theta}, \hat{\phi}, \hat{\psi}$ が 4 節では常に 0(水平姿勢) で与えられたのに対して，5 章では不整地歩行での胴体の移動を考慮して，これらの目標姿勢が運動計画と同時に与えられる．また，4.2.4 節の遷移条件によって実現される脚の接地パターンに関してはロボット

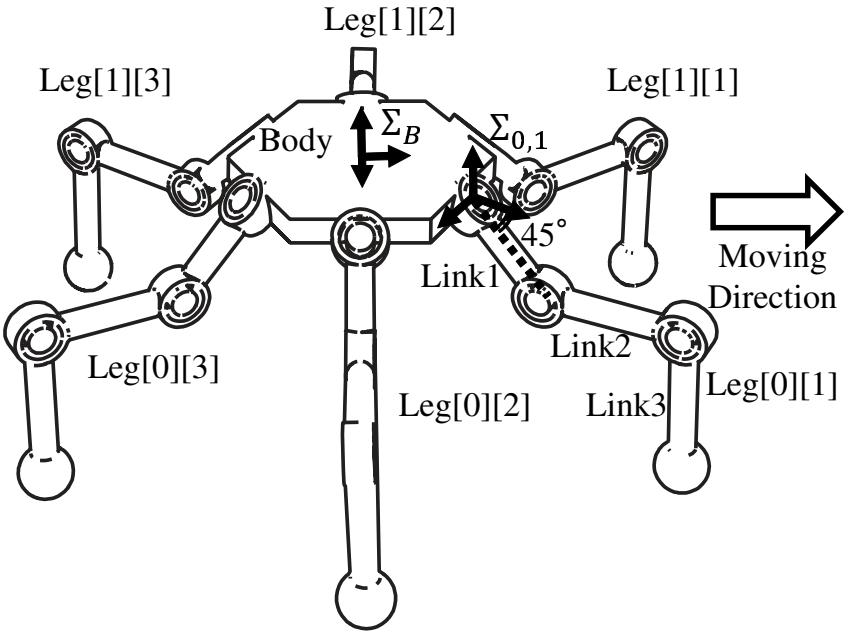


Fig. 5.1　Hexapod robot and local coordinate system on each leg

を円対称構造に変えたことから Fig. 5.3 のよう 4 つの姿勢に分類される．具体的には，1 つは対角の 3 脚が接地・それ以外の脚が遊脚している姿勢，2 つ目は 6 脚中 2 脚のみある 1 脚をまたいで遊脚している姿勢，ある 1 脚のみが接地している姿勢，全脚が接地している姿勢の 4 つのパターンに限定される．

5.2.3　制御域の再定義

4 章では，接地可能領域を導出する上で，制御域 $A_{i,j}$ の領域を一旦広げた上で，導出された接地可能領域に基づき $A_{i,j}$ も再更新された．本章では，目的の 1 つとしてロボットの振る舞い，運動の幅を広げることを目的としているので，一貫して制御域を次の定義として使用する．

$$A_{i,j} = \Omega_{i,j} \tag{5.1}$$

Area 1: 制御域 $A_{i,j-1}(= \Omega_{i,j-1})$ と重なる領域；$\Omega_{i,j} \cap \Omega_{i,j-1}$，
Area 2: 前後の脚の制御域と重ならない領域，
Area 3: 制御域 $A_{i,j+1}(= \Omega_{i,j+1})$ と重なる領域：$\Omega_{i,j} \cap \Omega_{i,j+1}$．

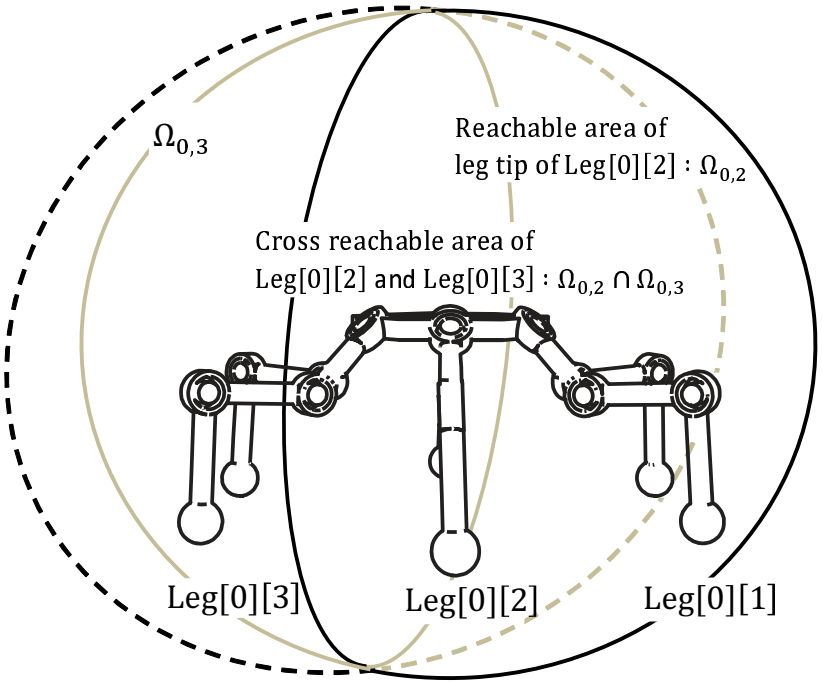


Fig. 5.2　Reachable area of center and rear leg-tip

5.2.4　先頭脚の接地点操作

本章のロボットでは Fig. 5.4 のように上部に深度センサが取り付けられている．深度センサは周囲に赤外線を照射してその反射光で環境情報を読み取る．また同じ箇所に操作者が接地点を観測するためのカメラを搭載している．鈴木ら [84] のヒューマンインザループ制御による接地点操作の手法に基づき，人がカメラを通して環境情報を受け取り，それを基に接地点目標点を一歩ずつ決めながらロボットを歩行させる．具体的には先頭脚 Leg[0][1]，Leg[1][1] の制御モード 2 のときに，深度センサから地面に赤外線を照射する．この照射した点群の座標を Fig. 5.6 のフローに基づき，遠隔の PC に送信する．受信した PC には Fig. 5.5 のような深度センサ・カメラから見た点群のようすである．この点群のうち，先頭脚が接地可能な点群がカメラスクリーン上で濃く表示される．このとき，接地点指令と同時に前節の姿勢制御におけるロール・ピッチ・ヨーの目標姿勢 $\hat{\theta}$, $\hat{\phi}$,$\hat{\psi}$ も指令値としてロボットに送ることができる．最後に操縦者がこの接地可能な点群の中から接地点を選択することで，接地点の指令がロボットに送られ，先頭脚のモード 3 へ遷移し，接地される．この操作を一歩ずつ行い，歩行を持続させる．ただし，ここでの接地可能な点群とは，各脚の可到達域と歩行環境の重なる範囲にある点群のことを指す．

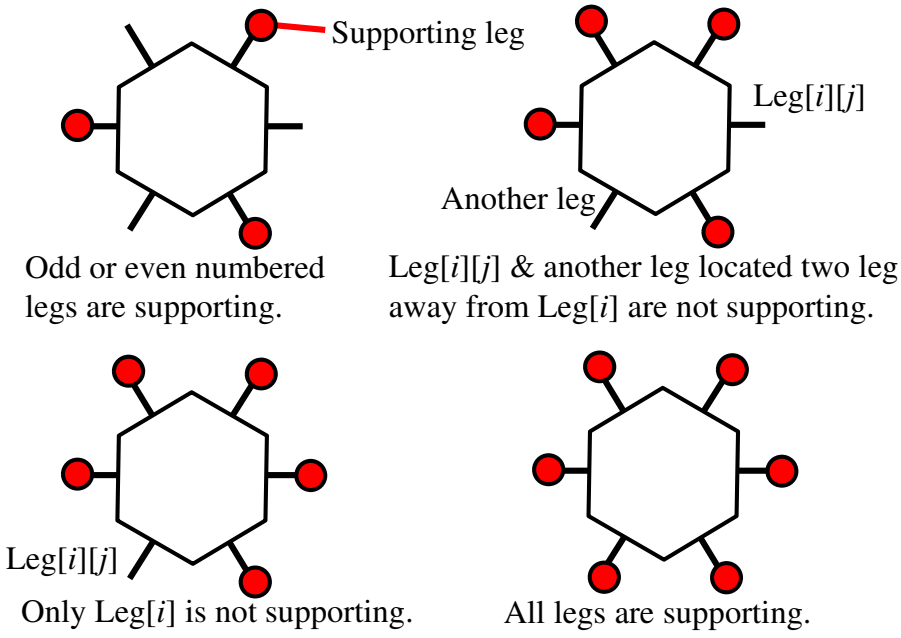


Fig. 5.3　Allowable contacting patterns under trandition condition from mode 4 to 1 in Chapter.5

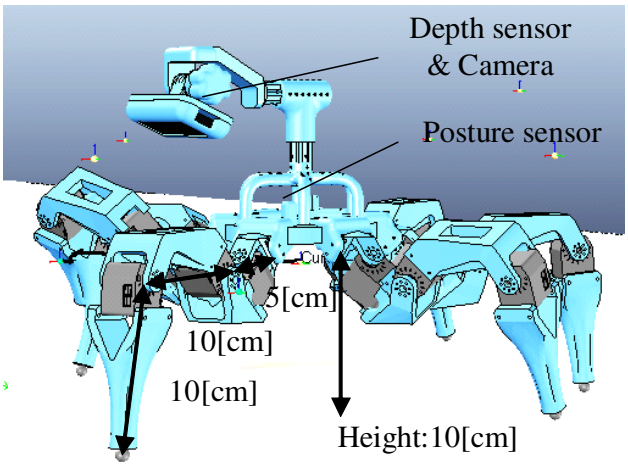


Fig. 5.4　Simulation model of hexapod

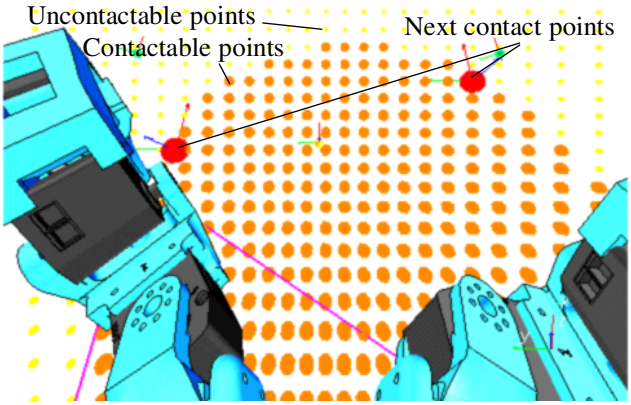


Fig. 5.5　Camera view and planning contact points

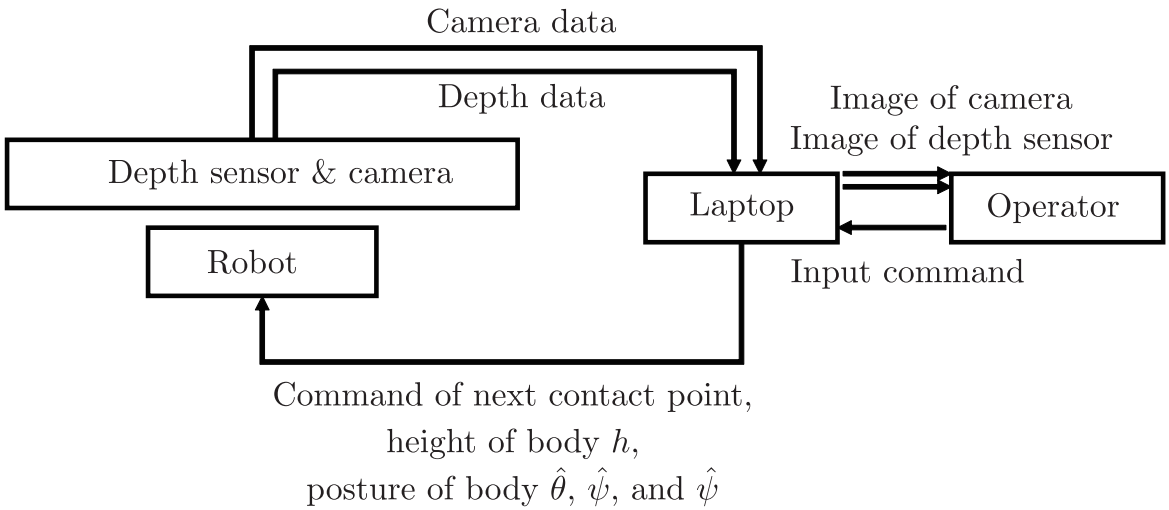


Fig. 5.6　Process of human-in-the-loop control for hexapod

5.3　時間制約と Timekeeper 制御

本節では，4 章で導出した時間制約を確認し，これを保証するための Timekeeper 制御を説明する．なおこの制御は前節で説明した接地点追従法と共に，制御周期ごとにリアルタイムに実行する．

5.3.1　等価時間係数を用いた時間の制約充足

4 章で導出した時間制約を再度ここに提示する．ここで，X は 4 章で使った時間オートマトンにおける状態のアルファベット (A, B, C, D, DE) を指し，t_X は各状態における滞在時間，$T_X^{\min}, T_X^{\max}$ はその最小値と最大値を指す．

$$
\begin{gathered}
T_X^{\min} \le t_X \le T_X^{\max}, \\
\begin{pmatrix} T_A^{\max} & T_B^{\max} & T_C^{\max} & T_D^{\max} & T_{DE}^{\max} \\ T_A^{\min} & T_B^{\min} & T_C^{\min} & T_D^{\min} & T_{DE}^{\min} \end{pmatrix} = k \begin{pmatrix} 4 & 1 & 1 & 4 & 4 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix}.
\end{gathered} \tag{5.2}
$$

また，各滞在時間 t_X を有する状態 SwingA〜C，SupportD〜E はロボットの次の制御モードに対応していた．

SwingA:　制御モード 1 の動作（接地点からの離脱）
SwingB:　制御モード 2 の動作（中間点での待機）
SwingC:　制御モード 3 の動作（接地目標点への接近）
SupportD:　制御モード 4（体節の前進：後脚の可到達域外 $\Omega_{i,j} \cap \overline{\Omega_{i,j+1}}$）
SupportE:　制御モード 4（体節の前進：後脚の可到達域内 $\Omega_{i,j} \cap \Omega_{i,j+1}$）
SupportF:　制御モード 4（体節の前進：2 つ後ろの可到達域内 $\Omega_{i,j} \cap \Omega_{i,j+2}$）

この各状態は全て制御オートマトンの制御と対応している．接地点追従法の制御において，この時間制約を用いるためには，この時間制約を各脚 $\mathrm{Leg}[i][j]$ の制御オートマトンによる滞在時間 $t_m[i][j], (m \in \{1,2,3,4\}, m$ は制御モード$)$ に対応させる必要がある．$t_m[i][j]$ と t_X の対応関係は次の式で得られる．

$$
\begin{aligned}
t_1[i][j] &\leftarrow t_A, \\
t_2[i][j] &\leftarrow t_B, \\
t_3[i][j] &\leftarrow t_C, \\
t_4[i][j] &\leftarrow \max\{t_{DE}, t_D\}.
\end{aligned} \tag{5.3}
$$

t_4 の対応については対応する滞在時間が t_{DE}, t_D の 2 つであるが，SupportD に属しているとき，$t_4 \leftarrow t_D, t_{DE} = t_D$ となり，SupportE,F に属しているとき，$t_4 \leftarrow t_{DE}, ($このと

き $t_{DE} \geq t_D$) となることから両方の滞在時間の最大値を取った値を対応させる．この対応に基づき，時間制約を次のように書き換える．ただし，$kT_m^{\max}, kT_m^{\min}$ は各制御モードの滞在時間の最大値，最小値である．

$$kT_m^{\min} \leq t_m \leq kT_m^{\max},$$
$$\begin{pmatrix} T_1^{\max} & T_2^{\max} & T_3^{\max} & T_4^{\max} \\ T_1^{\min} & T_2^{\min} & T_3^{\min} & T_4^{\min} \end{pmatrix} = \begin{pmatrix} 4 & 1 & 1 & 4 \\ 1 & 0 & 0 & 0 \end{pmatrix}. \tag{5.4}$$

この時間制約を保証し続ければ，4 章で定義した動作仕様が保証される．

1. デッドロックを起こさないこと．
2. 脚先が環境に引き摺られる，または引っかかることが原因でデッドロックを起こさないこと．
3. ロボットを支える接地点数が 3 点以上であること．

つぎに，(5.4) 式の滞在時間の制約充足を行ううえでの制御上で逐次的に調整することのできる変数について考える．まず，前述の通り，等価時間係数 k は任意の値を取ることができる．つぎに滞在時間 $t_m[i][j]$ を制御上で調整することはできるだろうか．たとえば，制御モード 1(脚を上げる動作) において $t_1[i][j]$ を大きくするためには，脚の運動を遅くし，$t_1[i][j]$ を小さくするために，脚の運動を速くするように事前に運動指令を決めることで，$t_1[i][j]$ が時間制約を保証できる場合もある．しかし時間には不可逆性があり，たとえばある制御モードにおいてモードの動作が始まってからの経過時間が既に時間 T に達した後に経過時間を T 未満に逆戻りさせることはできない．この経過時間は次のモードへの遷移のタイミングで滞在時間 $t_m[i][j]$ と同じになる．つまり，不整地上移動において滞在時間 $t_m[i][j]$ をとっさに変更する場合，$t_m[i][j]$ は次の制御モードに遷移するまでは上昇させることしかできない．

等価時間係数 k:　任意の正の値をとる．
滞在時間 $t_m[i][j]$:　次の制御モードに遷移するまでの間，上昇させることしかできない．

(5.4) 式をリアルタイムに保証させるために，上記の 2 つを調整するが，それぞれをどのように使用して制約充足を行うか決める必要がある．まず，滞在時間 $t_m[i][j]$ は減少させることができないことから，上限 ($t_m[i][j] \leq kT_m^{\max}$) を保証させるのに使うことはできないが，逆に下限 ($t_m[i][j] \geq kT_m^{\min}$) の保証には利用することができる．この考えを基に時間制約の上限と下限をそれぞれ異なる変数を使って保証させる．

上限 ($t_m[i][j] \leq kT_m^{\max}$):　等価時間係数 k を大きくすることで保証させる．
下限 ($t_m[i][j] \geq kT_m^{\min}$):　滞在時間 $t_m[i][j]$ を大きくすることで保証させる．

この 2 つの操作を行う手法を提案するが，ここで上限も下限も等価時間係数 k の更新の影響をうけることに着目したい．この k の更新そのものは上限側の制約を保証させる目的で使うが，いっぽうで，この k の更新によって下限 $kT_m^{\min}$ も変化し，滞在時間 $t_m[i][j]$ を調整する操作にも影響が出る．さらに，この k の更新も $t_m[i][j], \forall i, \forall j$ のうち最も大きな値の影響を受けることになる．つまり，ロボットの遷移のタイミングを時間制約により調整する上で $t_m[i][j]$ の上限値が最も支配的になる．

このような滞在時間が制約を超えるときに制御介入を行う制約充足の方法として 1.1.8 節で挙げたスーパーバイザ制御が参考になると考えられる．スーパーバイザ制御は制御対象の状態を観測者 (オートマトン) が観測し，観測された状態が制約を満たしているかを観測者が判定し，満たしていない場合または制約から外れる直前に制約を保証できるように制御介入を行う手法である．この観測される状態と制約が本章では各制御モードの滞在時間と時間制約に相当する．スーパーバイザ制御を参考にした本章の制約充足の概要を Fig. 5.7 に示す．まず，ロボットの制御オートマトンとは別にロボットの各制御モードの滞在時間を観測するための時間オートマトンを配置する．ただし，ここで定義する時間オートマトンは 4 章で用いたモデル検査用の時間オートマトンとは異なり，各制御モードの滞在時間が時間制約を保証しているか否かを判定するためにこの章で新しく定義する．最初に時間オートマトンにより現在時刻における全ての制御オートマトンにおける制御モードの滞在時間 $t_m[i][j]$ を観測する．そして，観測された $t_m[i][j]$ が (5.4) 式の時間制約が保証されているかを判定する．最後に制約が保証されない場合にロボットへの制御介入を行い，制約を保証させるように働く．この最後の制御介入の処理を以下に列挙する．

上限超過 ($t_m[i][j] > kT_m^{\max}$): 等価時間係数 k を上昇させる．
制約内 ($kT_m^{\min} \leq t_m[i][j] \leq kT_m^{\max}$): 等価時間係数 k を下降させる, または制御介入なし．
下限未満 ($t_m[i][j] < kT_m^{\min}$): 滞在時間 $t_m[i][j]$ を増加させるように脚の動きを遅らせる．

以上の手法はスーパーバイザ制御との共通点はあるが，リアルタイムに観測される滞在時間から逆に等価時間係数 k と時間制約そのものを変更させる．このことから他の制約充足制御とは区別して本章では「Timekeeper 制御」と名付け，その詳細を次節に示す．

5.3.2　Timekeeper 制御

Timekeeper 制御の具体的な Algorithm 2 に示す．まず等価時間係数 k とその下限値 $k^{\min}$，各脚・制御モードの滞在時間 $t_m[i][j]$ の初期値を与える．そして，6〜12 行目で全脚の各モードの滞在時間 $t_m[i][j]$ を計測する．この計測にはクロック変数を用いる．滞在時間 $t_m[i][j]$ は 7 行目の操作で 1 ステップ前からモード遷移が行われた場合に，0 に初期化される．つぎに等価時間係数 k を更新する．滞在時間 $t_m[i][j]$ の最大値が時間制約の上限値と一致するように k の値を更新する．これによって，滞在時間 $t_m[i][j]$ が滞在可能時間の上限を常に保証することになる．いっぽうで，時間制約の下限 $T_m^{\min}$ については制御モードの動作が早く終わり，$t_m[i][j]$ が $T_m^{\min}$ 未満である場合に，つぎのモードへの遷移をせず，一時停止または動きを遅らせることで $t_m[i][j]$ を延長させる．これによって，滞在可能時間の下限を保証させる．等価時間係数 k の推移は Fig. 5.8 のようになる．k の値が上下を繰り返しているが，これは時間制約の上限に沿って k が更新されているからである．k が上昇するときは観測される滞在時間 $t_m[i][j]$ に対して一定の勾配 $\frac{1}{T_m^{\max}}$ で上昇する．この上昇の途中で，各脚に関して制御モードが切り替わるとその脚の滞在時間が $t_m[i][j] = 0$ にリセットされるため，k の更新に使われる $\frac{t_m[i][j]}{T_m^{\max}}$ も 0 を返し，値が急降下する．

この k の値は全ての滞在時間 $t_m[i][j]$ の最大値によって更新される．この最大値に該当する制御モード・脚はロボットの動きに対応させると，最も脚の動きが遅れているものに

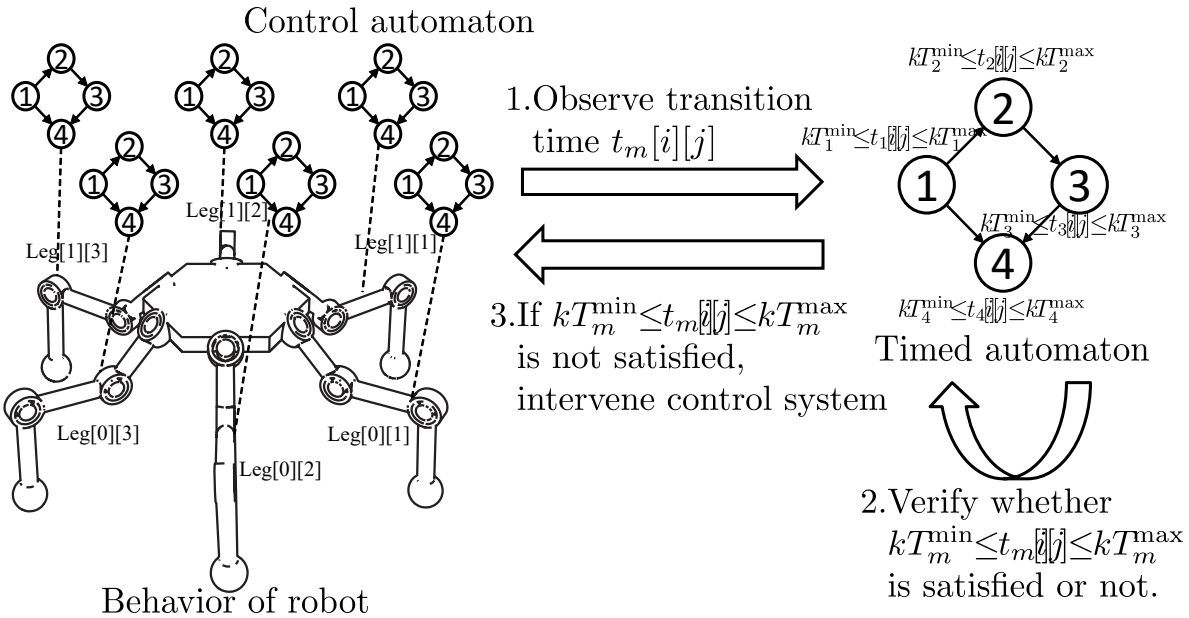


Fig. 5.7　Overview of Timekeeper control

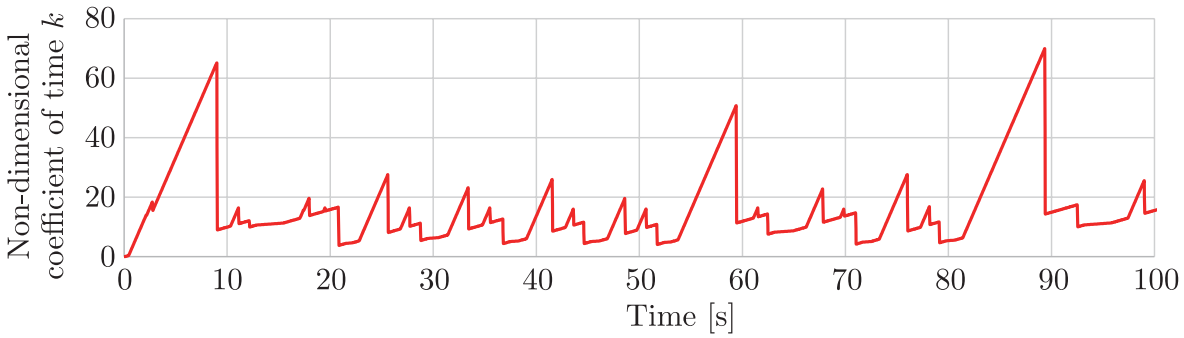


Fig. 5.8　Example of Time factor k

あたる．たとえば，不整地で地面に脚を引っ掛けたりした場合などが挙げられるが，この動きの遅れた脚に対して k が大きくなると時間制約のスケールも大きくなる．すると他の脚について動きの遅れた脚に沿って更新された時間制約を保証するように働くために全体として動きが遅くなる．そこで，全体の脚の動きを遅らせるために使われるのが，時間制約の下限 $kT_m^{\min}$ である．つまり，Timekeeper 制御による効果はつぎの順番で発揮される．

1. いずれかの脚の動きが遅れ，滞在時間 $t_m[i][j]$ が大きくなる．
2. 大きくなった $t_m[i][j]$ に沿って，時間制約の縮尺にあたる等価時間係数 k が大きくなる．
3. 下限値 $kT_m^{\min}$ も大きくなるので，他の脚の動きも遅くなる．
4. 動きが遅れた脚の制御モードが遷移したら $t_m[i][j] = 0$ になるので，k と時間制約が小さくなる．

ただし，ロボットの歩行で，Timekeeper 制御によって保証されるのは前述した 3 つの仕様である．Timekeeper 制御の歩行に関してこれ以外のことは保証できない．たとえば，上記の仕様を満たしても胴体の目標軌道と実際の軌道が一致するかどうかは保証できない．そこで，この仕様では記述されていない胴体姿勢，胴体軌道，接地点操作については，5.2 節のように複数の制御要素として導入した．

5.3.3　等価時間係数 k の更新に関する議論

Timekeeper 制御の k の更新 (Algorithm 2 の 13 行目) に関しては当初 k の変化によるロボットの動きの変化を滑らかにする目的で k の下降時には以下の更新則を使って研究

Algorithm 2 Timekeeper control

$k = 1$:Non-dimensional coefficient of time
$k^{\min} = 2$:Minimum value of k
$\begin{pmatrix} T_1^{\max} & T_2^{\max} & T_3^{\max} & T_4^{\max} \\ T_1^{\min} & T_2^{\min} & T_3^{\min} & T_4^{\min} \end{pmatrix} = \begin{pmatrix} 4 & 1 & 1 & 4 \\ 1 & 0 & 0 & 0 \end{pmatrix}$
$t_m[i][j] = 0, \forall i, \forall j, \forall m$:transition time of control mode m in Leg$[i][j]$
while robot is walking **do**
  for $i = 0$ to 1,$j = 1$ to 3 **do**
    if Control mode changed in current step **then**
      $t_m[i][j] := 0$
    else
      Record $t_m[i][j]$.
    end if
  end for
  $k = \max \left\{ \begin{array}{c} \max\limits_{\substack{i\in\{0,1\},\\ j\in\{1,2,3\},\\ m\in\{1,2,3,4\}}} \frac{t_m[i][j]}{T_m^{\max}}, \\ k^{\min} \end{array} \right\}$
  for $i = 0$ to 1,$j = 1$ to 3 **do**
    if $t_m[i][j] < T_m^{\min}$ and behavior of control mode m has been already finished **then**
      Stop behavior of leg.
    end if
  end for
end while

していた．

$$k = \max \left\{ \begin{array}{c} \max\limits_{\substack{i\in\{0,1\},\\ j\in\{1,2,3\},\\ m\in\{1,2,3,4\}}} \frac{t_m[i][j]}{T_m^{\max}}, \\ k - \alpha \\ k^{\min} \end{array} \right\}, \tag{5.5}$$

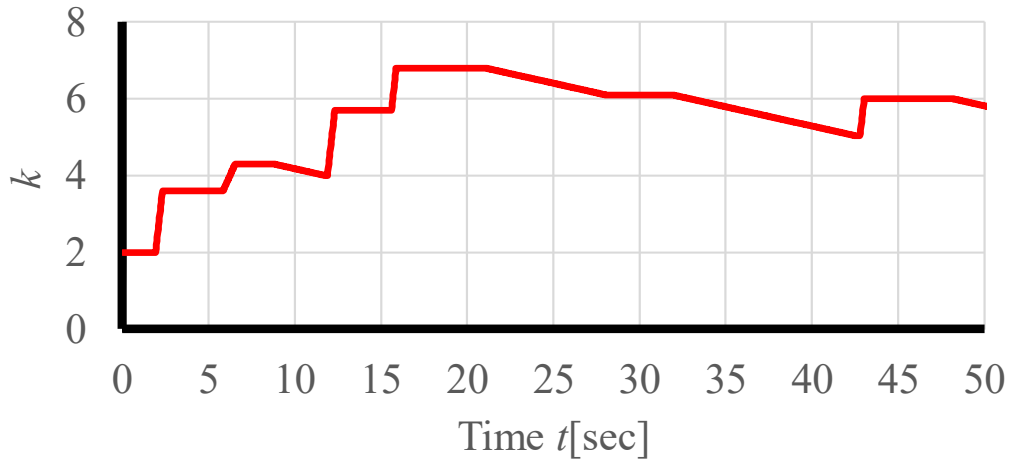
また，記録した滞在時間のリセットのタイミングについてもすぐに k が下降するタイミングを遅らせる目的で，当初は Algorithm 3 のように制御モードが終わったタイミングではなく，次のモードに遷移した後も滞在時間の値を維持し続け，一周し再度遷移したタイミングでリセットするようにしていた．この各滞在時間の記録方法と k の更新方法の差によって，Fig. 5.9 のような 3 つのパターンの k の推移を確認している．

Algorithm 3 Recording transition time

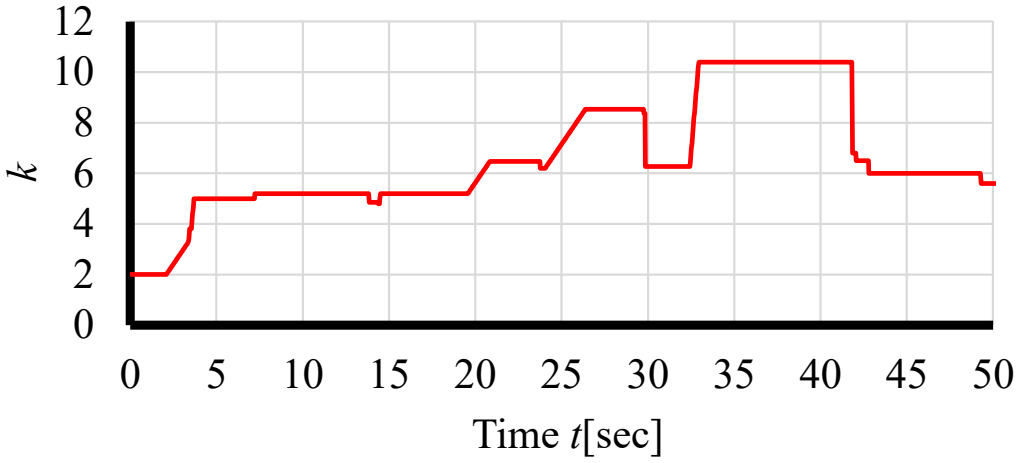
```
1: for i = 0 to 1,j = 1 to 3,m = 1 to 4 do
2:     if Control mode m starts in current step then
3:         t_m[i][j] := 0
4:     else
5:         Record t_m[i][j].
6:     end if
7: end for
```

(a).(5.5) 式による k の更新と Algorithm 3 の滞在時間の記録による Timekeeper 制御
(b).Algorithm 3 の滞在時間の記録を使った Timekeeper 制御
(c).Algorithm 2 による Timekeeper 制御

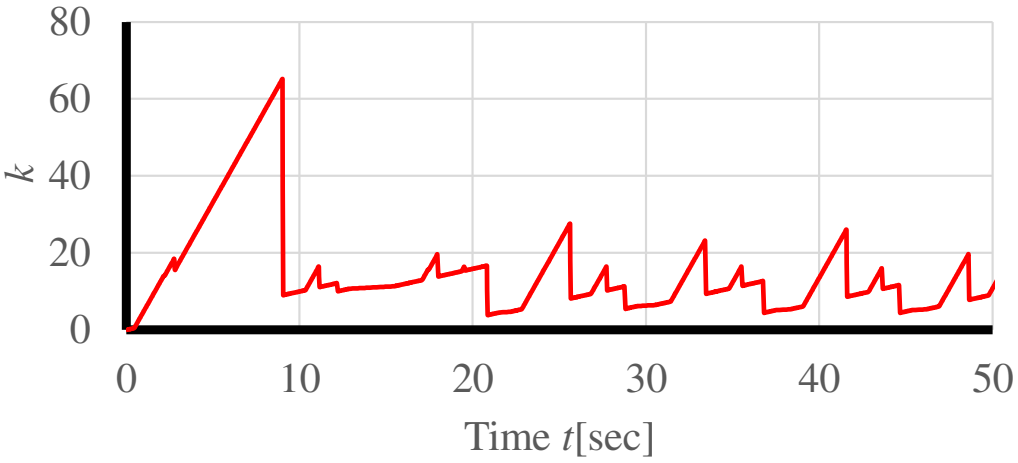
最終的に 3 つ目 (Fig. 5.9(c)) の手法を k の更新方法として採用したが，その理由としては，k が急降下しても時間制約のスケールが小さくなるだけで，k の急降下と同じタイミングで制約によりロボットの脚の動きが変わるとは限らないからである．ロボットの脚の動きは k の推移ではなく，時間制約の最小値未満である場合に，ロボットの脚の動きを遅らせるように働く．よって，k の変化を (5.5) 式に従って，滑らかな推移にする必要はない．このことを確認した経緯として Fig. 5.9(a)(b) に基づくシミュレーションと (c) の Timekeeper 制御に基づく実験を行ったのでそれぞれ順番に次節から説明する．



(a)Timekeeper control with eq.(5.5) and Algorithm 3



(b)Timekeeper control with Algorithm 3



(c)Timekeeper control

Fig. 5.9　Updation of Time factor k

5.4　シミュレーション・実機による実験

Timekeeper 制御による効果を確かめるため，物理エンジンによるシミュレーションと実機実験を行なった．以下にロボットの仕様とシミュレーション・実験環境について説明し，結果と考察を述べる．ただし，シミュレーション・実機実験の項目は 5.3.3 節で挙げた 3 つの k の更新則毎に分類して説明する．

5.4.1　シミュレーション (5.3.3 節の (a))

歩行環境として Fig. 5.10 のような 3 つの地形を用意し，それぞれ 20 回ずつ歩行させ，踏破回数を数える．それぞれの特徴は以下の通りである．ただし，胴体の高さ 10cm を基準に縮尺は決定した．

コース 1　:石垣状の地形．間に 10cm の溝がある．
コース 2　:ブロック群の地形．各 5〜20cm の高さ．
コース 3　:1 段 10cm の階段．

まず，コース 1 では石垣間の溝に，コース 2 では高い位置から脚を下ろす時に他のブロックに脚を引っ掛けやすい．そのため，コース 1 では接地相 (モード 4)，コース 2 では遊脚相での遅れが予想される．コース 3 はコース 1，2 の両方の要因で遅れが生じ，転倒しやすいために加えたコースである．

比較する試行は次の 2 つである．ただし，ロボットのスペックはいずれも Table 5.3 の通りである．

Case1　Time Keeper 制御を導入した本手法．
Case2　本手法を用いず，等価時間係数 $k = 1$ を固定し，Timekeeper 制御を用いない 4 章の手法 (従来手法として扱う)．

Case2 は Case1 に対して，Timekeeper 制御を適用せずに，従来手法によりあらかじめロボットのスペックから k を求め，固定したまま歩行させた試行である．ただし，この従来手法では k を決める上で参考になるロボットのスペックが「接地点計画時間 t_2 が 1 秒以下である」という前提のみであったため，(5.4) 式の上限値と対応させて，$k = 1$ に固定した．また，Case1,2 の接地点操作は同一の操縦者が操作する．

Table 5.1　Charaferestics of robot

	Length [cm]	Mass[kg]
Link1	5.00	0.15
Link2	10.0	0.30
Link3	10.0	0.30
Body	12.0	1.00

Table 5.2　Simulation result

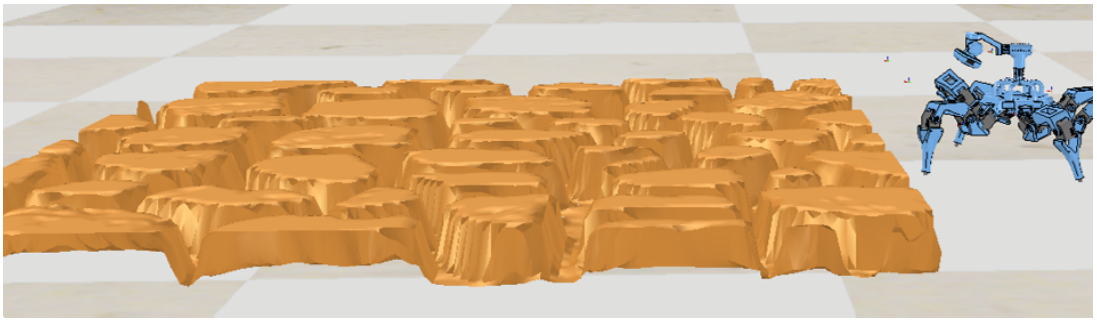
	Course1	Course2	Course3
Case1	18/20	14/20	13/20
Case2	14/20	3/20	1/20

これに加えて，Timekeeper 制御による挙動の変化を確認するため，Case1 の平地歩行において，接地操作の時間 $t_2[i][j]$ の時間を外乱により極端に長引かせたときの，各モードの持続時間と等価時間係数 k を Fig. 5.11 に，滞在時間を k で割り，無次元化した値 $\frac{t_m[i][j]}{k}$ の変動を Fig. 5.12 にプロットした．ただし，滞在時間 $t_m[i][j]$ は各モードが遷移する直前の持続時間の値とする．この場合，(5.4) 式より，$T_m^{\min} \leq \frac{t_m[i][j]}{k} \leq T_m^{\max}$ であれば良いので，$\frac{T_m[i][j]}{k}$ がこの関係式を満たすことを確認する．

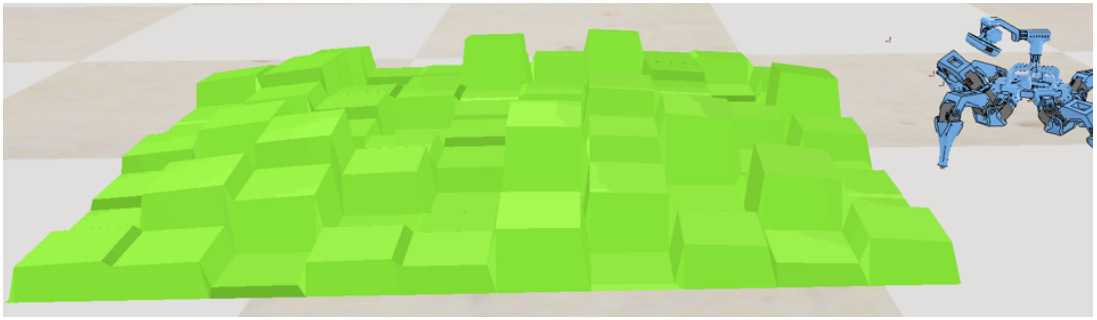
5.4.2　結果と考察

まず，Fig. 5.11(a) から外乱が挿入された後に各脚のモードの滞在時間が変動するが，モードの遷移が継続している，つまり歩行が継続されていることが分かる．また Fig. 5.11(b) から，外乱挿入後にモードの滞在時間の最大値と共に等価時間係数が変化しており，Timekeeper 制御が働いていることが分かる．ただし，本シミュレーションは (5.5) 式により k の急降下を防いでいるため，ロボットの動作が一度遅くなってから通常の歩行周期まで戻るまでの時間が非常に長くなっている．これはロボットの歩行において大きな弊害となる．次節以降のシミュレーションでは (5.5) 式を省いて再度シミュレーションを行う．

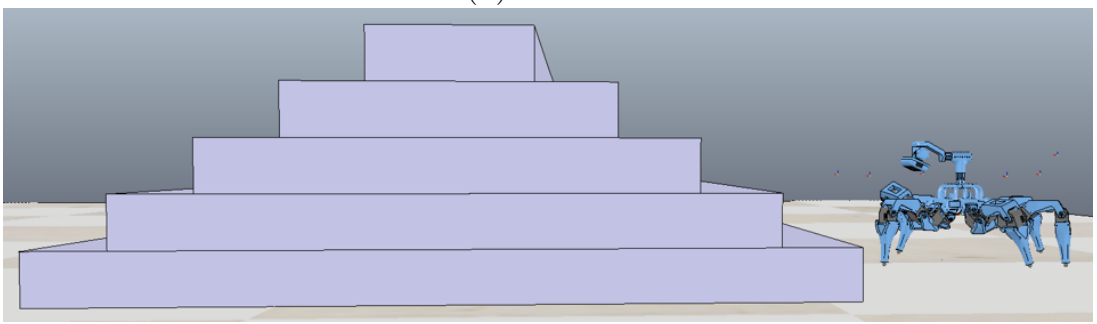
つぎに歩行性能について，Table. 5.4 のように，いずれの歩行環境でも Case2 より Case1 の踏破回数が多いことが確認できた．Case1 で走破できなかった試行としては，脚を上げる際に階段に脚を引っ掛け，転倒する場合があった．このような要因の転倒には Timekeeper 制御では対処できないが，Timekeeper 制御の効果を利用して，転倒回避用の制御要素を追加することができる．



(a)Course 1

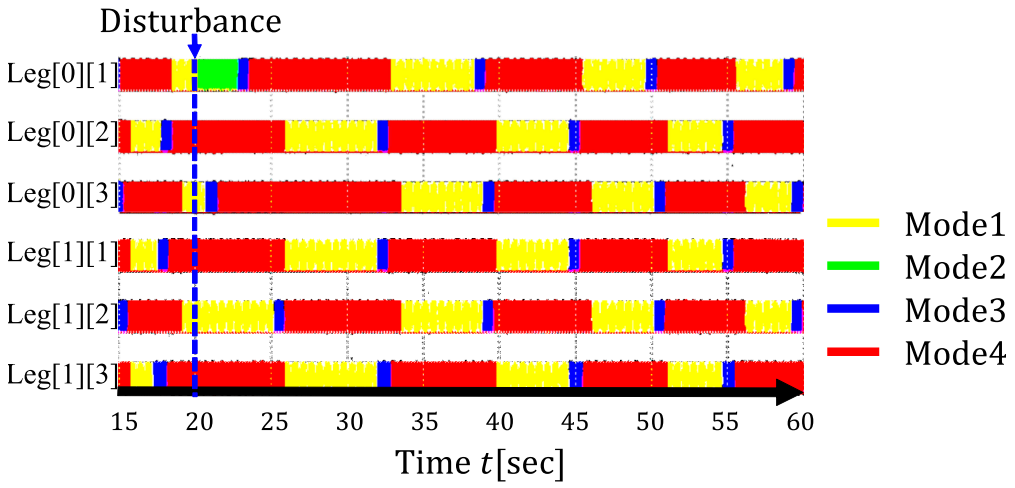


(b)Course 2

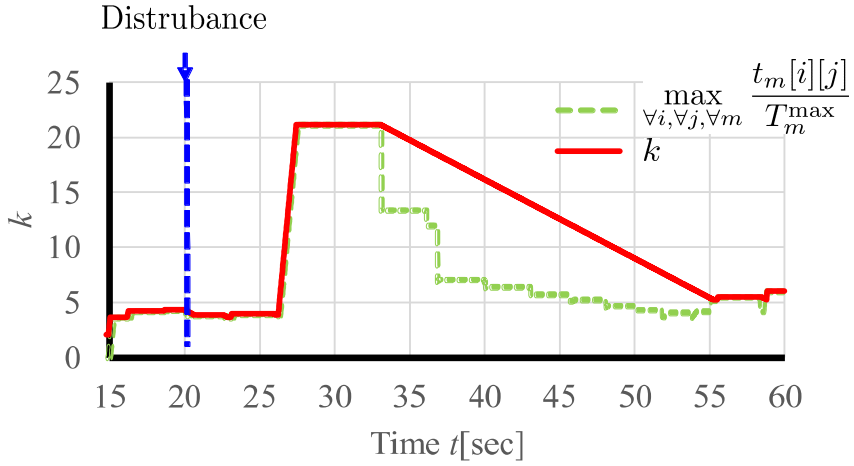


(c)Course 3

Fig. 5.10　Simulation course

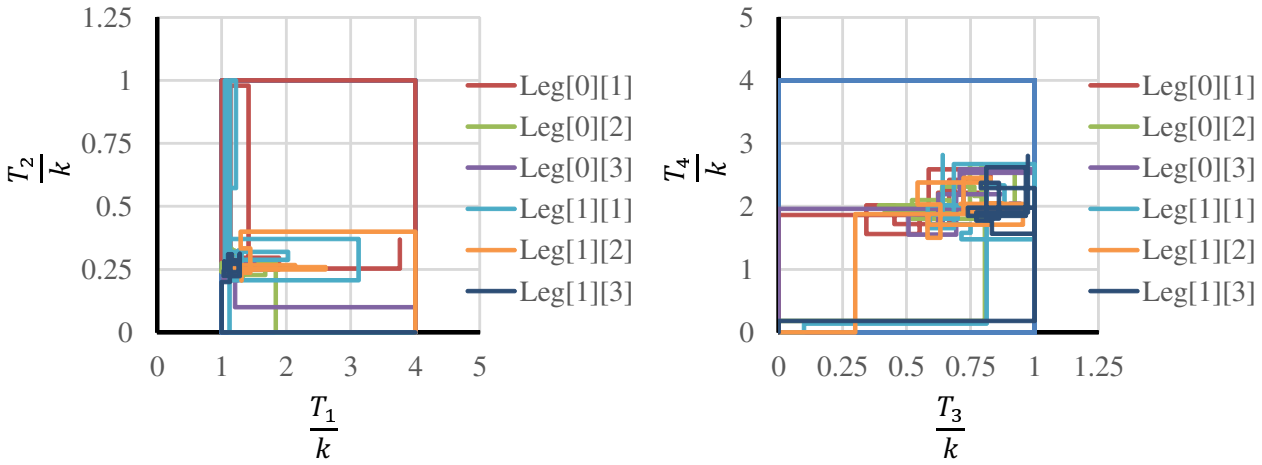


(a)Elapsed time of each mode



(b)Time factor k

Fig. 5.11　Control modes and time factor k in the proposed method, Case1



(a) $\frac{T_2}{k}$ vs. $\frac{T_1}{k}$　　(b) $\frac{T_4}{k}$ vs. $\frac{T_3}{k}$

Fig. 5.12　Locus of non-dimensional time $\frac{t_m[i][j]}{k}$

Table 5.3　Charaterestics of robot

	Length [cm]	Mass[kg]
Link1	5.00	0.15
Link2	10.0	0.30
Link3	10.0	0.30
Body	12.0	1.00

5.4.3　シミュレーション (5.3.3 節の (b))

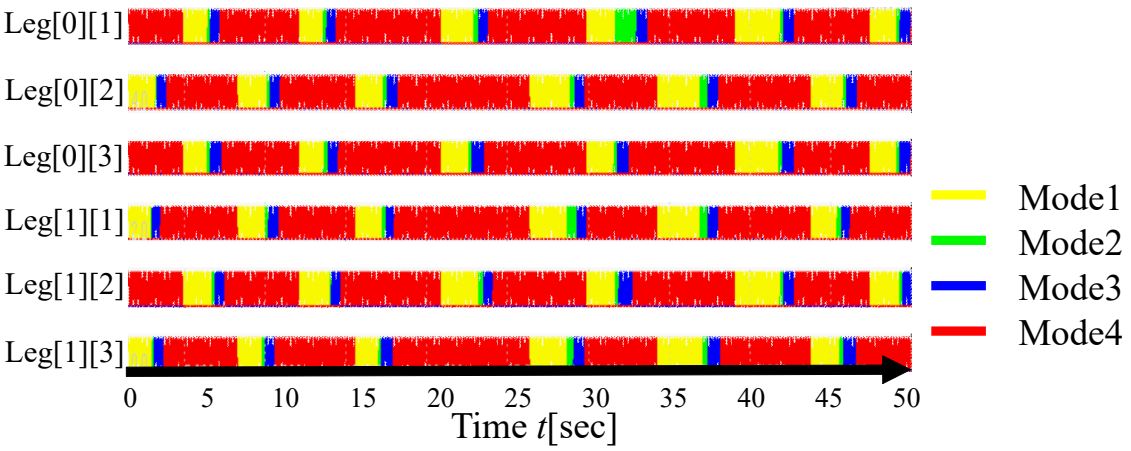
歩行環境として **Fig. 5.10** のような胴体より少し高い段差と，制御手法の異なる 2 つの試行 Case 1，2， ロボットの操縦者 6 人 (A, B, C, D, E, F) を用意する．操縦者が Case 1，2 のロボットを各 20 回ずつ操作する．このとき，操縦者の熟練度と先入観が結果に反映されないように，どちらが Case 1，2 かわからない状態で 2 つの試行を交互に 5 回ずつ操作するセットを 4 回繰り返す．操作したロボットが段差を乗り越え，その後すべての脚が段差から離れたときに歩行が継続している場合を踏破したと判断し，この踏破回数を記録する．また，比較する試行 Case 1，2 はつぎのとおりである．ただし，ロボットのスペックはいずれも Table 5.3 の通りである．

Case1　Timekeeper 制御を導入した本手法.

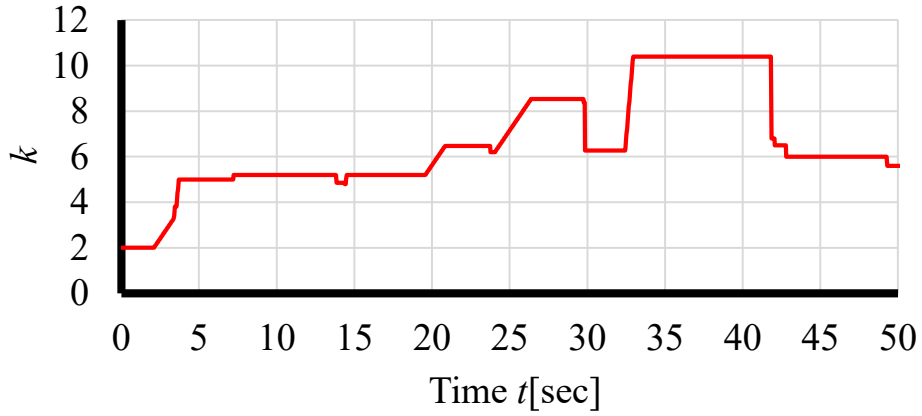
Case2　等価時間係数 $k = 1$ を固定し，Timekeeper 制御を用いない 4 章の手法 (従来手法として扱う).

Case 2 は Case 1 に対し，本手法を適用せず，従来手法によりあらかじめロボットのスペックから k を求め，固定したまま歩行させる試行である．k を決める上で「接地点計画時間 t_2 が 1 秒以下である」という前提から (5.4) 式の上限値と対応させて，$k = 1$ に固定した.

さらに，Case1 のある試行における各制御モードの滞在時間と等価時間係数 k を Fig. 5.13 に示す．また，歩行中の脚について先行研究で考慮されていなかった直線運動以外の多様な運動も歩行中に行われていることを確認するために，いずれかの脚が遊脚（制御モード 1・3）時に壁に引っかかったときの脚先の移動軌跡を Fig. 5.14 に示す.



(a)Elapsed time of each mode



(b)Time factor k

Fig. 5.13　Control modes and time factor k in the proposed method, Case1

5.4.4　結果と考察

まず，Fig. 5.13(a)(b) から，各滞在時間の値によって，等価時間係数 k が変化していることがわかる．いっぽうで，個々の制御モードの滞在時間の比についてはモード 1 の長さがモード 4 の $\frac{1}{4}$ 以上の値を，モード 2，3 がモード 1 以下の値を維持できていることがわかる．また，(b) における 30 秒前後では k が急増した．k の値が急増する主な要因は脚先が歩行環境に引っかかるか，引きずられて滞在時間が伸びることであり，このときも Leg[1][3] の脚先が壁に引きずられてモード 1，3 の遊脚における滞在時間が増加した．Fig. 5.14 には，このときの胴体中心座標系に基づく Leg[1][3] の脚先の移動軌跡を太線でプロットした．段差を登る様子を 6 コマに分けて表示しており，1，2 コマ目は制御モード 4 の軌跡，3，4 コマ目では制御モード 1 の軌跡，5 コマ目では制御モード 3 の軌跡，6 コマ目は制御モード 4 のまま段差を昇ったときの軌跡を示している．4 コマ目の軌跡を見

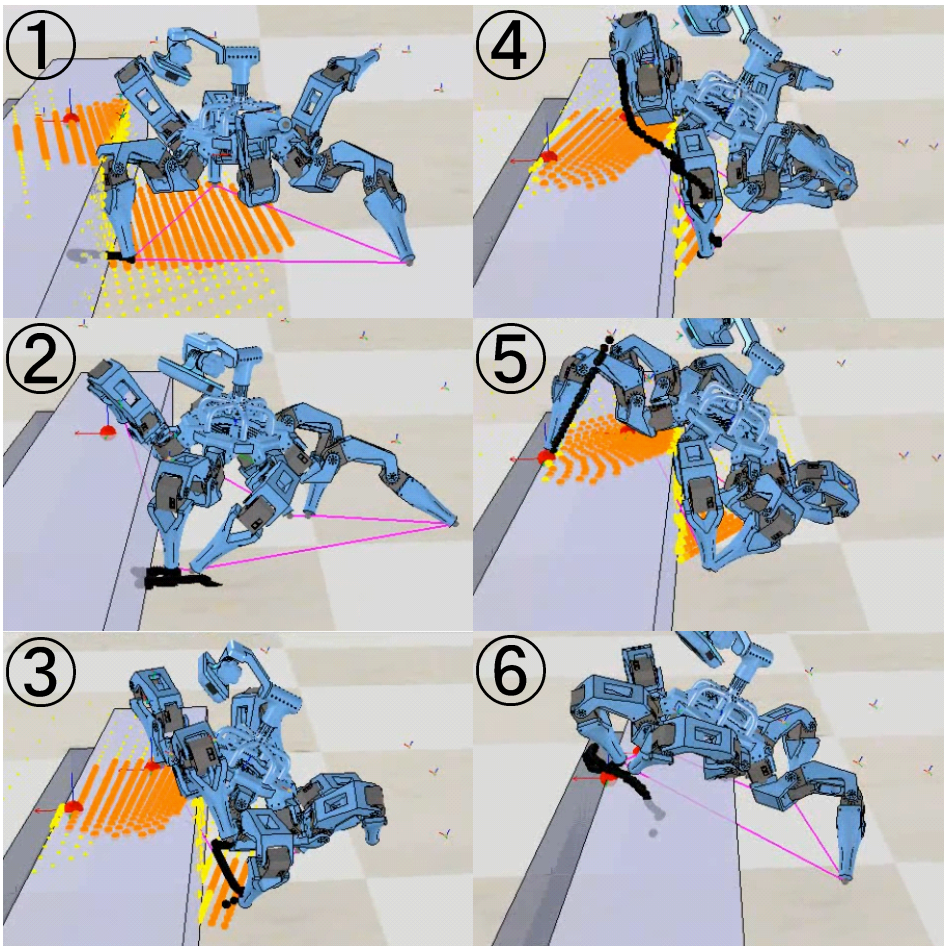


Fig.　5.14　Trajectory of Leg[4] looked from body coordinate system : ①②control mode4 ; ③④mode1 ; ⑤mode3 ; ⑥mode4

Table 5.4　Success rate in Simulation

	Operator for robot					
	A	B	C	D	E	F
Case1	18/20	19/20	15/20	16/20	17/20	18/20
Case2	6/20	3/20	6/20	5/20	4/20	7/20

ると，脚を上げる間，段差の側面に脚先が引きずられた影響で脚先が迂回しているようすが見られ，遊脚において直線運動以外の脚運動も行なわれていることが確認できる．いっぽうで，5 コマ目で脚を下す際には途中で障害物への衝突がなかったため，大きな迂回はせず，ほぼ直線軌道を描いている．このように本手法では，滞在可能時間を守る限り，4 章では考慮されていなかった多様な脚運動も保証されていることが確認できた．

つぎに各 Case の踏破回数を見ると Table 5.4 のように，いずれの操縦者も Case 2 より

Case 1 のほうが踏破回数が多いことが確認できた．つぎに，不整地の踏破回数について，Table 5.4 のように，いずれの操縦者も Case 2 より Case 1 のほうが踏破回数が多いことが確認できた．この理由は，Case 2 では脚を段差の側面に一度でも引っ掛け，ある制御モードの滞在時間が伸びた後，ほかの脚との協調が取れなくなり，デッドロックを起こすからだと考えられる．Case 2 では滞在可能時間の等価時間係数 k の調整をしていないので，一度脚の動きのタイミングがずれるとそのずれが修正できずに大きくなる．いっぽうで，Case 1 では脚を引っ掛け，いずれかの制御モードの滞在時間が長引いたときに，等価時間係数 k と滞在可能時間が大きくなり，これに応じてほかのすべての脚の滞在時間も，大きくなった滞在可能時間に収められるように調整される．つまり，ある脚の動作が遅れ，滞在時間が伸びたときに，ほかの脚の滞在時間もそれに協調できるように調整されるため，脚間の協調が維持され，歩行中にデッドロックすることがなくなる．これが Case2 より，Case 1 の踏破回数が多くなった理由であると考えられる．

ただし，前節のシミュレーションと同じように Case 1 でも走破できなかった事例がいくつかある．そのほとんどは脚間の協調は維持できているものの，脚を上げる際に段差の側面を脚先が押し出し，その反動で転倒することが原因であった．これは Timekeeper 制御による滞在時間の調整だけでは対処できないと考えられる．これを解決するためには，接触センサなどを利用し，遊脚時に脚先が障害物を回避する制御を追加する必要がある．これが今後の課題の 1 つとして考えられる．

5.5　実機実験 (5.3.3 節の (c))

本節では Fig. 5.15 の実機を使って Timekeeper 制御による不整地歩行の実験を行う．センサ・モーター・マイコン等の実機のスペックは Table 5.5 に示す．

まず，Fig. 5.19 のようなレンガでできた階段状と Fig. 5.20 の瓦礫の山の 2 つの歩行環境を用意する。レンガの一段の高さは 5cm，瓦礫の大きさはは 5〜15cm 程，それに対しロボットの胴体の高さの初期値は $h = 10$[cm] にする．遊脚時の各制御モード 1〜3 における脚先の目標移動速度はそれぞれ $v_1 = 10$[cm/s] and $v_3 = 30$[cm/s]，胴体の目標移動速度は $v_4 = 10$[cm/s] とする．接地点の操作には，操作者一人を用意し，Fig. 5.16 の操作画面を基に，先頭脚の接地目標点を一歩ずつ選んでもらい，踏破回数および踏破できなかった場合でのデッドロックの回数を数え，提案手法により仕様が保証されているかを確認する．

また，比較する試行 Case 1，2 はつぎのとおりである．ただし，ロボットのスペックはいずれも Table 5.3 の通りである．

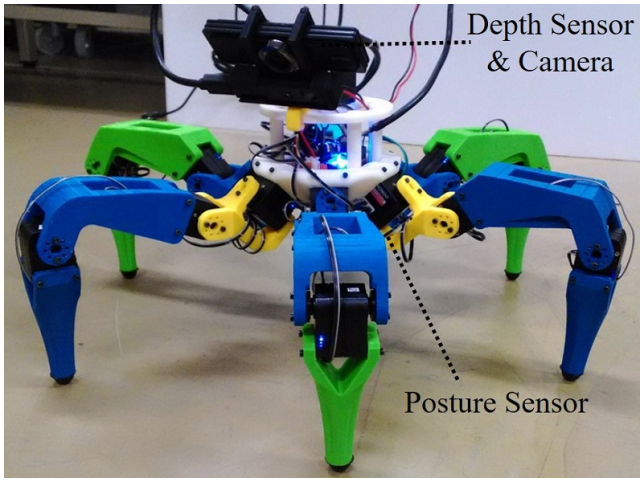


Fig. 5.15　Real hexapod robot

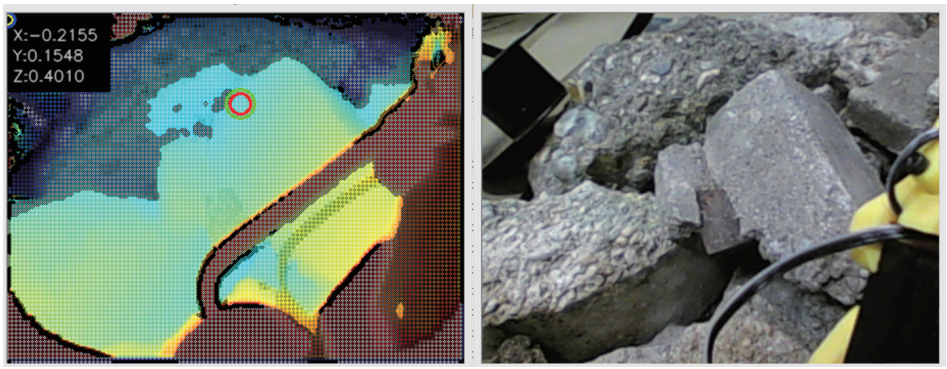


Fig. 5.16　View of depth sensor and color camera

Case1　Timekeeper 制御を導入した提案手法.

Case2　等価時間係数 $k = 1$ を固定し，Timekeeper 制御を用いない 4 章の手法 (従来手法として扱う).

Case2 は Case1 に対し，本手法を適用せず，従来手法によりあらかじめロボットのスペックから k を求め，固定したまま歩行させる試行である．k を決める上で「接地点計画時間 t_2 が 1 秒以下である」という前提から (5.4) 式の上限値と対応させて，$k = 1$ に固定した.

5.5.1　実験結果

Case1 で，環境を踏破したときの歩行の様子を Fig. 5.19 に示す．それぞれのコマには先頭脚が地面に接地した瞬間との接地点位置を示している．これを見ると，前脚が接地した接地点は後脚もほぼ正確に追従できており，接地点追従法による歩行が成立しているこ

Table 5.5　Specification of hexabot robot

Length of link	5[cm], 10[cm], 10[cm]
Total mass	1.34[kg]
DoF	leg:3, body:1 (to rotate depth sensor and camera)
Depth sensor	Cam board pico flexx, pmdtechnologies ag (224 × 171, 60 × 45[°])
Color camera	BSW200MBK, Baffalo inc. (1920 × 1080, 120 × 120[°])
Posture sensor	JY-901, DingDong
Microcomputer	8bit AVR ATmega644P, Microchip Technology Inc.
Servo motor	KRS-3304 ICS, KONDO KAGAKU CO. (Max 1.36[Nm], 0.16[s/60°])

とが確認できる．

Fig. 5.17 には Case1 の各制御モードの滞在時間と Timekeeper 制御における係数 k の値を表示している．k の値が常に更新されている様子が見られる．特に先頭脚がモード 2 で接地点計画が遅れている場合に k が大きくなり，他の脚がこの遅れた脚に同期して動くように Timekeeper 制御が働いている様子が確認できる．

また，姿勢制御の精度を確認するために，Fig. 5.18 には胴体のピッチ角とロール角の目標値との差分を表示した．この図のようにほとんどの場合は，目標値に収束している様子が確認できる．たびたび見られる大きな偏差は接地脚を組み合わせを切り替えたときに生じるものだと考えられる．

つぎに Case1，2 についての踏破回数の比較結果を Table 5.6，5.7 に示す．Case1 でのレンガおよび瓦礫地形での成功確率は 85〜90 %であったのに対し，Case2 では踏破回数は 50 %程になっている．このうち失敗した例について，Case1 で動作仕様が満たせず (デッドロック，3 点支持が維持できない等) に転倒した回数を見ると一度もなかった．いっぽうで失敗・転倒の要因としては，操作者によって歩行が安定しないような接地点が選ばれた試行が挙げられる．逆に Case2 を見ると，動作仕様を満たせずに歩行が止まった試行が複数ある．この比較から Timekeeper 制御を導入したことで明らかに動作仕様を保証する働きがあったことが確認できた．ただし，Case1・2 共に他の転倒要因も観られ，特に先頭脚の接地目標点が誤って段差の側面や蹴込みに選んだことにより，うまく接地で

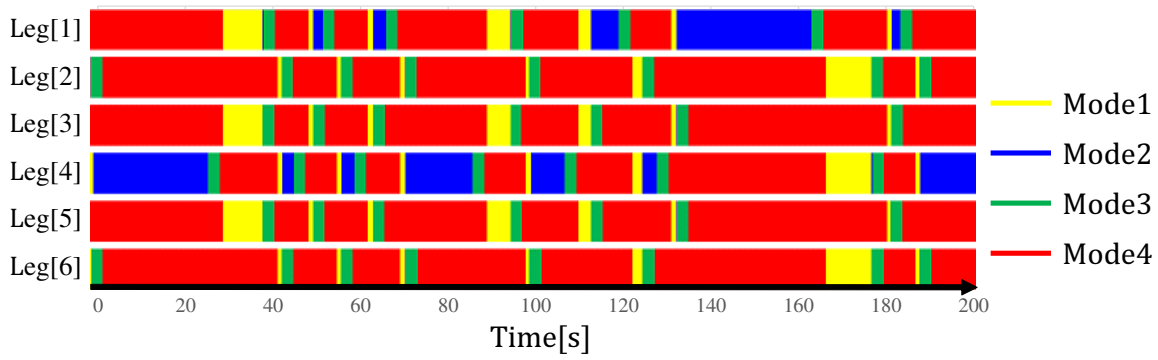
Table 5.6　Success rate in bricks

	成功回数	失敗例でのデッドロック回数	他の要因の失敗回数
Case1	34/40	0/6	6/6
Case2	22/40	11/18	7/18

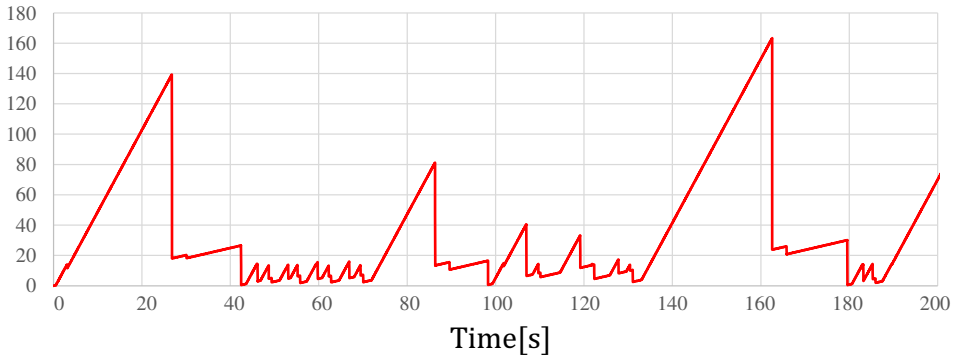
Table 5.7　Success rate in rubbles

	成功回数	失敗例でのデッドロック回数	他の要因の失敗回数
Case1	36/40	0/4	4/4
Case2	19/40	13/21	8/21

きずに転倒する場合があった．そこで，今後の課題としては，操作者自身の接地点操作の学習または接地点操作の自動化を行う必要があると考えられる．

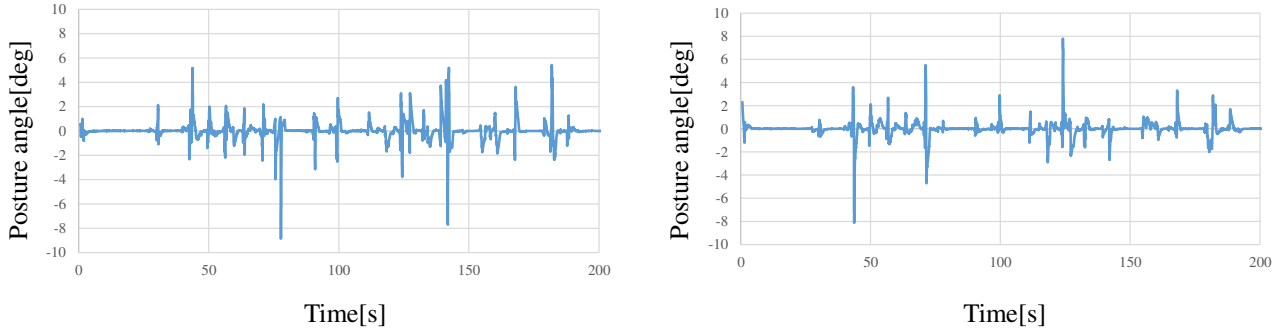


(a) Control mode



(b) Coefficient k

Fig. 5.17　Control mode and coefficient k in Timekeeper control



(a)Pitch angle $\phi_g - \hat{\phi}$　　(b)Roll angle $\psi_g - \hat{\psi}$

Fig. 5.18　Body posture in the experiment

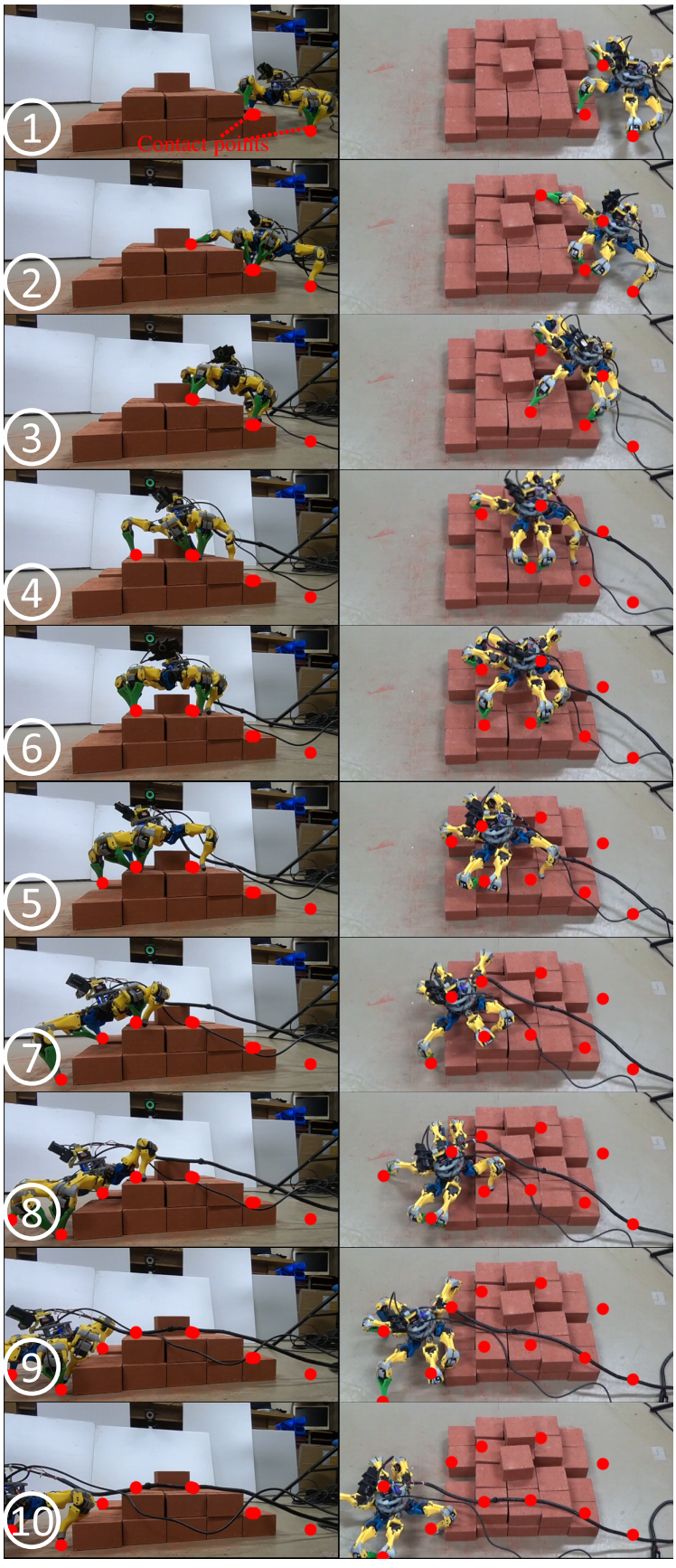
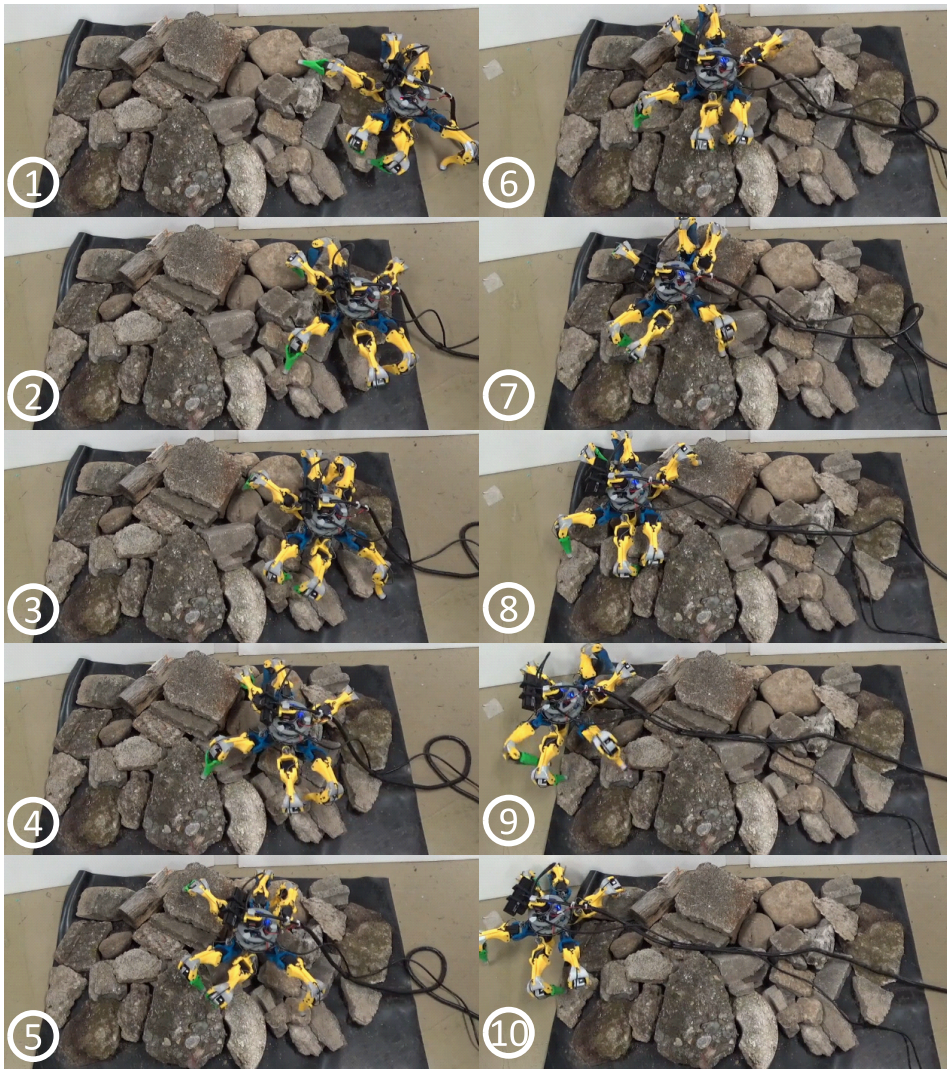
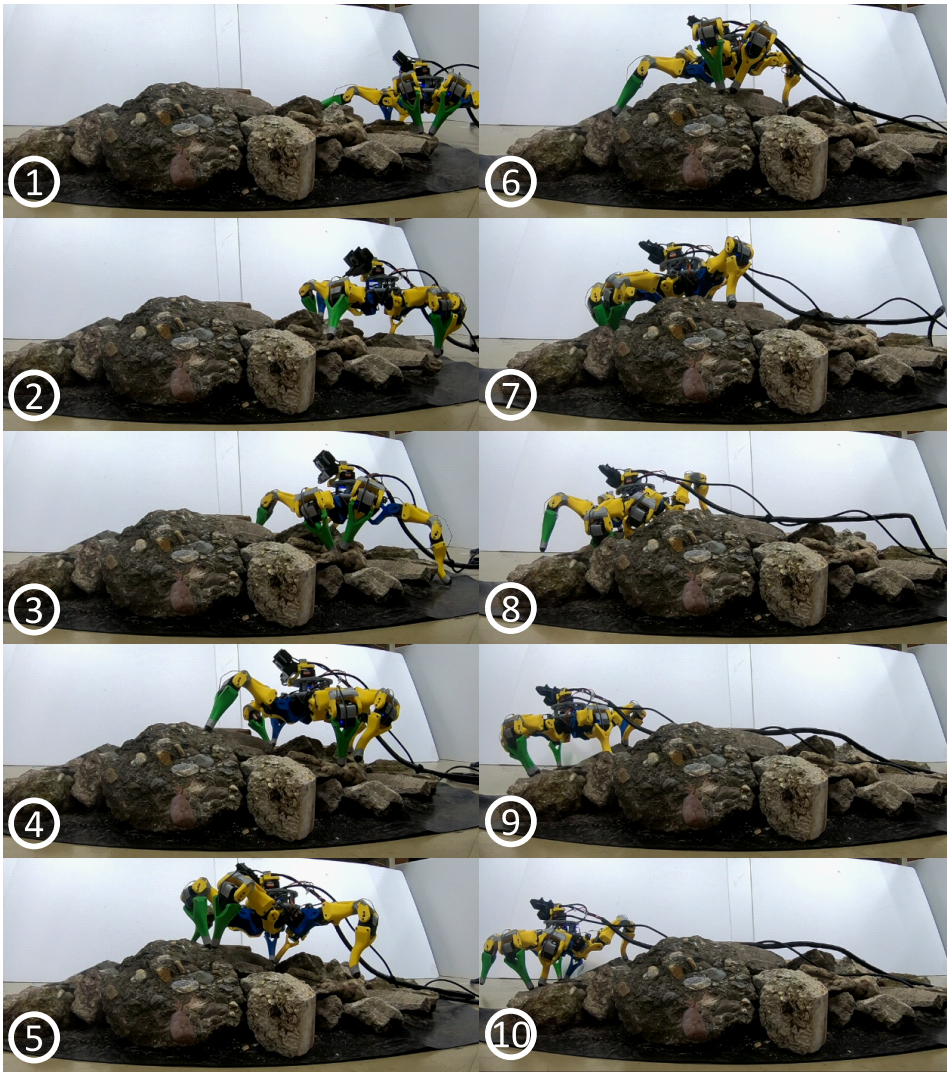


Fig. 5.19　View of experiment with contact point

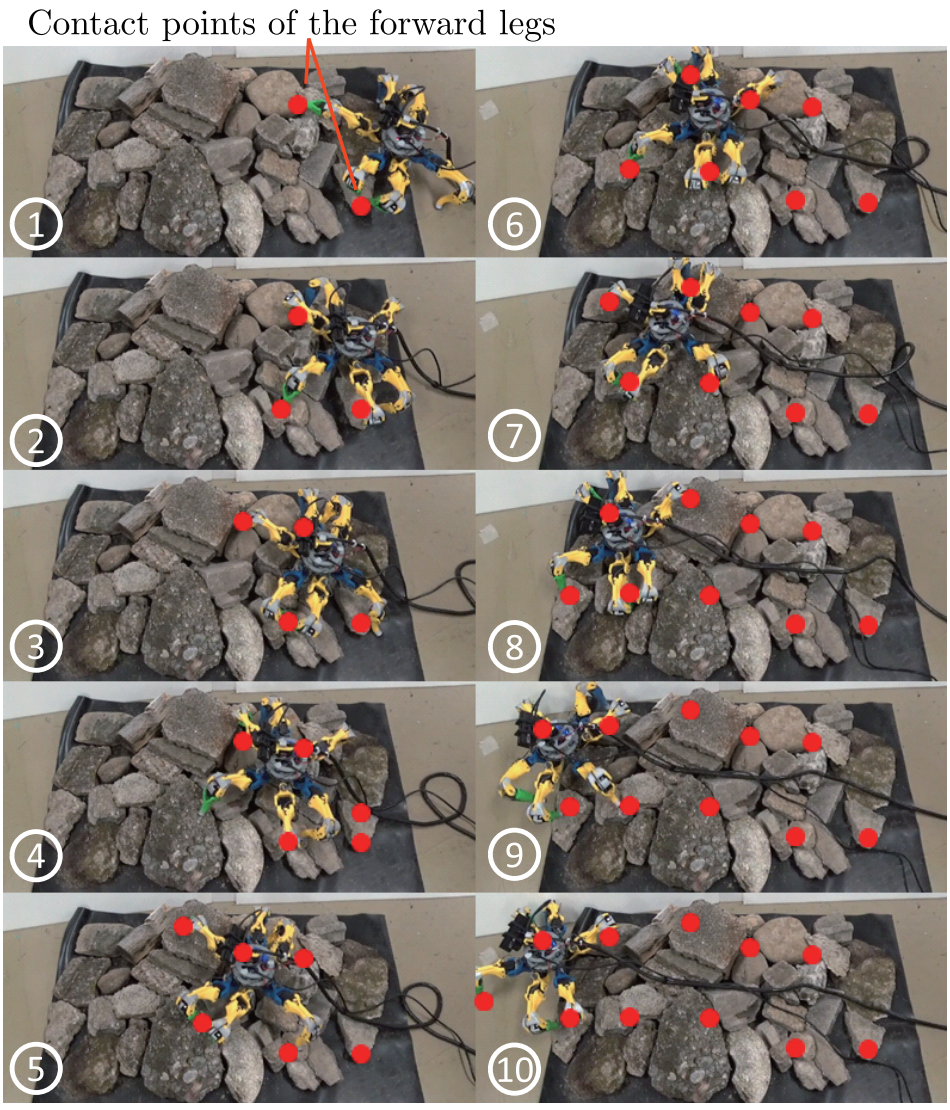


(a)Upper view

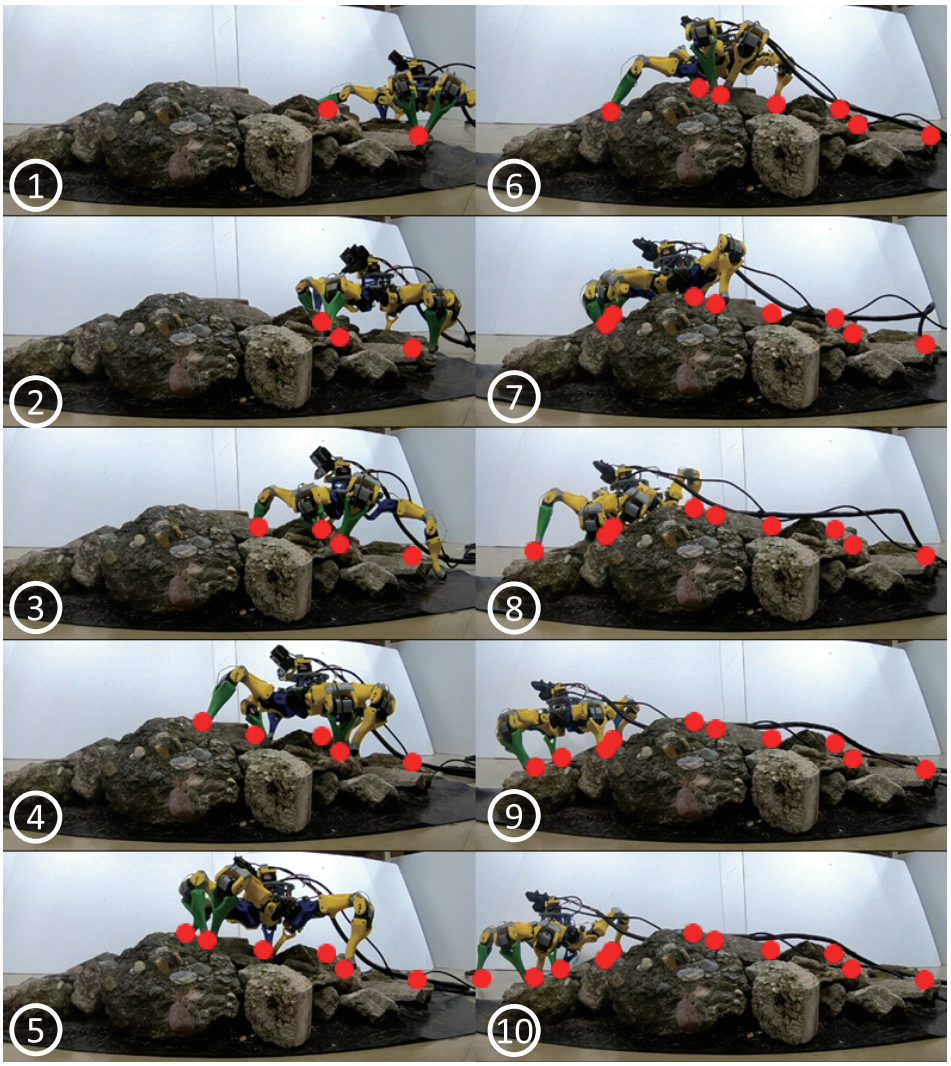


(b)Side view

Fig. 5.20　View of experiment



(a)Upper view



(b)Side view

Fig. 5.21　View of experiment with contact points

5.5.2　各々の実験における Timekeeper 制御の効果

この章では k の更新方法別に 3 つのシミュレーション・実機実験を行ったが，これらの実験環境すべてに，Timekeeper 制御なしでは時間制約を保証できずに動作仕様 (デッドロックを起こさない) を満たせなくなる要因がある．たとえば，5.4.1，5.4.2 節のシミュレーションコースには，石垣・ブロック・階段状の地形を用意しているが，いずれの地形にもロボットの胴体の高さ以上の大きさの段差を設定している．5.4.3 節の実機実験においても様々な大きさの瓦礫を置いた地形を用意している．このような地形を乗り越える場合，各脚の制御モードの滞在時間，遊脚時間，接地時間は一歩ごとに大きく変化する．たとえば，遊脚時・接地時に段差や溝に脚を引っ掛けた場合，その滞在時間も伸びるので，Timekeeper 制御なしでは，時間制約を破る状況が生まれ，デッドロックになる可能性がある．実際に 5.4.3 節では Timekeeper 制御なしの場合で，デッドロックになるケースがあったのに対して，Timekeeper 制御を導入した場合ではデッドロックになったケースが一度もなかった．つまり，この実験では，デッドロックになる要因 (脚の動きが遅れ，時間制約が保証できなくなる等) がある不整地環境上でも，Timekeeper 制御を使うことでデッドロックを完全に回避しながら歩行することができることが確認されたのである．

また，この Timekeeper 制御の時間制約そのものが，特定の歩行環境下ではなく環境情報を省略した数理モデル (モデル検査における時間オートマトン) を使って導出されたものであることに着目したい．このモデルでは，環境情報が直接記述されていない代わりに，脚の動きに対応する滞在時間に関して最小値から最大値までの時間制約を与えている．したがって，動作仕様 (デッドロックを回避する・3 点支持を維持する等) を保証するために時間制約をロボットに適用する上で，歩行環境は直接的に限定されないが，脚の動き (各制御モードの滞在時間) に関しては時間制約により拘束されることになる．つまり，Timekeeper 制御が提案されたことによる 1 つのメリットは制御要素同士を干渉させずに統合させる制御 (Timekeeper 制御) が歩行環境を考慮せずに導入することができる事を意味する．その代わり，Timekeeper 制御では脚の動きの方が，歩行環境および時間制約の縮尺 k によって調整される．ただし，この手法によって保証されるのは制御要素同士の統合と上記で定義されたロボットの動作仕様の保証のみであるため 、転倒安定性に関しては保証されない．転倒安定性に関しては Timekeeper 制御によって統合される制御要素 (姿勢制御・運動計画・局所制御等) を追加することで向上させる必要がある．

5.6 まとめ

本章では，接地点追従法における 3 つの仕様 (デッドロック回避，脚を引きずらない，支持多角形の維持) を保証する時間制約が与えられているときに，これをロボットの制御中に保証させる「Timekeeper 制御」を提案した．Timekeeper 制御では，時間制約の等価時間係数 k を歩行環境，ロボットの置かれた状況に応じて変更すること，滞在時間を時間制約に収めることの 2 つの操作を行ない，少なくとも 3 つの仕様が保証できないことによる歩行の停止，転倒を回避できるようになった．これは制御器内の複数の制御要素同士の干渉を防げるようになったことを意味する．

また，本章では 6 脚移動ロボットを対象として Timekeeper 制御を実装したが，ロボット以外でも動作仕様を保証するための時間制約が与えられたハードウェアに対して適用可能であると考えられる．ただし，適用の条件としてシステムが離散事象システムであるか，その振舞いが状態遷移系で記述でき，等価時間係数 k が可変である必要がある．さらに状況に応じて制御全体の動作の速度を遅くしたり，速くしたりすることが望ましいシステムには十分適した制御であると考えられる．

また，本章では，時間の不可逆性により各制御モードの滞在時間を減少させられない前提で Timekeeper 制御を実装したため，滞在時間が大きくなった場合に等価時間係数 k も大きくなる場合が見られ，歩行速度が遅くなるケースが生じてしまう．この問題は滞在時間が大きくなる要因を回避することで解決することができると考えている．たとえば，脚先の移動速度をできるだけ速めたり，脚の動きが遅れる要因の 1 つである障害物への衝突・脚先の引っ掛かりを回避しながらの運動をさせることで滞在時間が短くなり k の値も小さな値で抑えることができると考えられる．

第 6 章

6 脚移動ロボットの Timekeeper 制御の改善による歩行の多様性

6.1 はじめに

5 章では，接地点追従法を対象とした制約充足型適応制御「Timekeeper 制御」を提案した．この手法はモデル検査により得られた時間制約の縮尺が無次元時間係数で表現されていることを利用して，無次元時間係数と各モードの滞在時間 (=脚の振舞い) を調整することで時間制約を保証させている．これにより，制御要素同士の統合とそれによる不整地歩行を実現させた．

しかし，この研究には他の問題が残っている．4・5 章で追加した 6 脚ロボット用の遊脚条件 (4.2.4 節:追加した制御モード 4 → 1 の遷移条件) により，ロボットの歩容が意図的に tripod gait になるように動きの制限が与えられていたことである．この遊脚条件はロボットの支持多角形を維持する目的で追加されたいっぽうで，これを追加したもう 1 つの動機として，この遊脚条件を用いない 2 章の接地点追従法に基づく 6 脚ロボットに関して 4 章のモデル検査を用いた時間制約の探索を行った際に，「3 点以上の支持脚を維持する」という仕様を満たす時間制約が見つからなかったという経緯がある．このときは，3 章のモデル検査に基づくシステム開発・動作検証の考え方に基づき，どの時間制約 (ロボットの振る舞い) をシステム (接地点追従法による動作) に与えても動作仕様 (3 点以上の支持脚を維持する) を満たせなかったことから，システムそのものを変更する必要があると考えた．そこで，4 章では支持多角形を維持するための遊脚条件をロボットの制御に加えざるを得なかった．これによって，稲垣ら [63] による接地点追従法では，wave gait による歩容の中で，幅広い遊脚・接地のタイミングの調整ができたのに対して，4・5 章では tripod gait のみに限定されてしまった．

その後，研究を継続する中で，4 章の接地点追従法による脚の振舞いをモデル化した時間オートマトンに関するある問題点が明るみになった．それは，2 章の接地点追従法は分散歩行制御であり，各脚にはほぼ同じ制御オートマトンが割り振られているが，先頭脚，最尾脚，それ以外の脚についてはモード間の遷移条件が異なることである．たとえば，先頭脚での制御モード 2 は前方脚の接地点を待つ状態ではなく，接地点計画を行う状態であるし，最尾脚の制御モード 4 の遊脚条件は他の脚と異なる．この細部の制御の違いにより，動作仕様 (デッドロックを回避，3 点以上の支持脚を維持等) を保証するために必要とされる脚の振る舞い (時間制約) も脚毎に異なる可能性が高いのである．つまり，各脚に個別の時間制約を与えることで，tripod gait を強いる遊脚条件を追加せずに，ロボットの動作仕様を保証する時間制約が導出できるのではないかと考えられる．

ただし，この個別の時間制約を与える上で，モデル検査を使った 4 章の手法を直接使うことは探索する時間制約の数を比べると現実的ではない．4 章では，時間制約の最大値，最小値を $T_X^{\max}, T_X^{\min}, X \in \{A, B, C, D, DE\}$ としており，全部で 10 個の変数を探索していた．これを 6 脚ロボットで脚毎に導出する場合，扱う変数は 60 個となり，4 章に比べて膨大な探索時間が必要となる．5 章の Timekeeper 制御についても変数の増加は計算遅れを招く可能性が高い．このことを加味して本手法では，モデル検査を用いた時間制約の探索法と Timekeeper 制御の手法も脚毎に異なる時間制約を与えた場合の接地点追従法に使うために改良する．

本章では，6.2 節で 6 脚移動ロボットの構造と接地点追従法に追加した制御要素を説明をする．6.3 節でモデル検査を用いた各脚の時間制約の導出 (4 章) の改良をし，6.4 節で Timekeeper 制御の改良を行う．6.5 章で実機実験を行い，手法の改良の効果とそれによるロボットの振る舞いの変化を確認する．

6.2　本章で用いる接地点追従法とロボットの構造

ここでは，ロボットの構造とそれに伴う接地点追従法に追加した制御の説明を行うが，制御域 $A_{i,j}$ の定義，ヒューマンインザループ制御による運動計画と姿勢制御の手法は 5 章と同じものを扱うのでここでは省略する．

6.2.1　ロボットの構造

本章では，Fig. 6.1 のような 6 脚ロボットを用いる．5 章とは異なり，3 本の脚が左右対称に配置された構造になっている．このうち，胴体の正面 (カメラ，深度センサの向き) にある 2 脚を先頭脚とする．脚のラベル $\mathrm{Leg}[i][j](i = 0, 1, j = 1, 2, 3)$ 及び脚の座標系 $\Sigma_{i,j}$，脚先の可到達域を $\Omega_{i,j}$ は 2・4・5 章と同じ定義で扱う．脚の構造については，5 章と同じく 3 リンク 3 自由度とし，各脚先の可到達域の重なり部分を胴体より下方の領域で広くするために Link1 と胴体間の関節は 45 度傾けている．

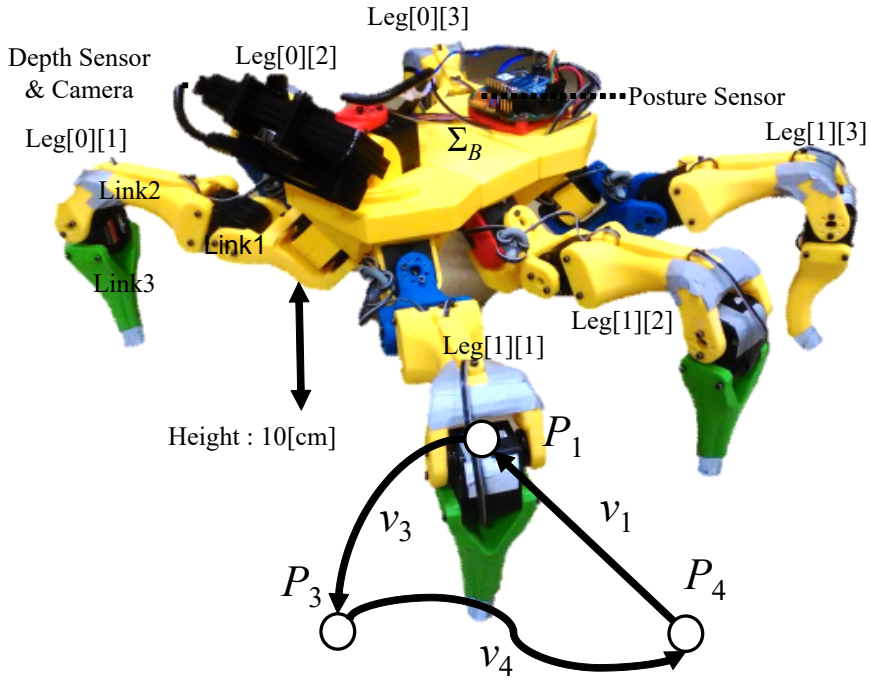


Fig. 6.1　The coordinates of each leg

6.2.2　制御モード 4〜1 への遷移条件

6 脚用の支持脚条件は 4 章を引き継ぐ．その内容を改めて提示する．Leg$[i][j]$ が脚を離す（制御モード 4 から 1 への遷移）条件として次のものが与える．

1. $j \leq 2$ の場合，後脚 Leg$[i][j+1]$ が自脚の接地点を引き継ぎ，反対側の脚 Leg$[1-i][j]$ が接地している．$j = 3$(最尾脚) の場合，反対側の脚 Leg$[1-i][j]$ が接地している．
2. 最尾脚 $j = 3$ についてのみ，ロボットが生成した新しい接地点 3 点 C_a, C_b, C_c が成す支持三角形を重力方向に投影して作られる三角柱の内部にマージンを取った領域に胴体の中心が含まれている．

まず 1 つ目の遷移条件は従来のムカデ型ロボット用の接地点追従法にも備わっているものである．2 つ目は支持多角形の中に重心を維持するために追加した遷移条件である．5.2.4 節で与えた 2 つ目の遊脚条件を最尾脚のみに加えたものである．最尾脚以外 $(j \neq 3)$ の脚に関しては，1 つ目の遊脚条件により遊脚後も後続脚が自脚が最後に踏んだ接地点に追従していることが保証されるため，遊脚のタイミングを拘束する必要はない．

6.3　モデル検査を用いた各脚の時間制約の導出

4 章と同じように，接地点追従法に基づくロボットの振る舞いのモデル化，動作仕様の記述を行うが，ここでは，脚の遊脚条件と時間制約の定義が 4 章と異なることからモデルの再定義を行う．

6.3.1　時間オートマトンによる各脚の動作のモデル化

時間オートマトンを Fig. 6.2 に示す．各ノードにおける滞在条件はノードの上部に記述する．時間オートマトンはロボットの各脚の動作を表わし，状態は時間もしくは隣接する脚のオートマトンの状態に基づいて切り替わる．時間オートマトンにおけるクロック変数 t は，各オートマトンにローカルに定義されている．また，各脚・ノードの時間制約では，最小値，最大値をそれぞれ $T_X^{\min}[i][j]$，$T_X^{\max}[i][j]$ （X は各ノード名の語尾のアルファベット）と表記する．以下に時間オートマトンの各ノードの状態を列挙する．

SwingA　モード 1 の動作：脚先を上げる．
SwingB　モード 2 の動作：脚先を停める．
SwingC　モード 3 の動作：脚先を降ろす．
SupportD　モード 4：脚先が $\Omega_{i,j} \cap \overline{\Omega_{i,j+1}}$ に存在する．
SupportE　モード 4：脚先が $\Omega_{i,j} \cap \Omega_{i,j+1}$ に存在しつつ，滞在時間が時間制約の下界未満の状態．
SupportF　モード 4：脚先が $\Omega_{i,j} \cap \Omega_{i,j+1}$ に存在しつつ，滞在時間が時間制約の下界以上の状態．
SupportG　モード 4：脚先を引きずりデッドロックを起こしている状態．

定義した変数は以下のとおりである．

- $\mathrm{Ground}_{i,j} \in \{0,1\}$: $\mathrm{Leg}[i][j]$ の接地フラグ
- $\mathrm{Grounding}_{i,j}$: $\mathrm{Leg}[i][j]$ の接地時に発信する同期信号
- $\mathrm{Entering}_{i,j} \in \{0,1\}$: $\mathrm{Leg}[i][j]$ が後脚の接地可能領域 $\Omega_{i,j+1}$ に侵入しているかのフラグ
- $t \in \mathbb{Z}$: クロック変数．
- $T_x^{\min}[i][j], T_x^{\max}[i][j] \in \mathbb{Z}, (x \in \mathcal{A}, \mathcal{A} = \{A, B, C, D, DE, DEF\})$: 遷移時間の最大値，最小値（時間制約）．

- $\mathrm{Point}_{i,j} \in \mathbb{Z}$: それぞれの脚の接地点番号.

以下に時間オートマトンの各ノードでの脚先の状態及び遷移条件について詳細に述べる.

SwingA: 遊脚相（接地点からの離脱）

脚先が接地点から離れ，空中の中間点 $P_1^{i,j}$ に向かって動いている状態である．滞在時間を $t_A[i][j]$ とすると，$T_A^{\min}[i][j] \leq t_A[i][j] \leq T_A^{\max}[i][j]$ の間で SwingB に遷移する.

SwingB: 遊脚相（中間点での待機）

脚先が中間点で待機している状態である．先頭脚については接地点計画を行い，他の脚は前後との同期を取る．先頭脚のについては滞在時間を $t_B[i][j]$ とすると，$T_B^{\min}[i][j] \leq t_B[i][j] \leq T_B^{\max}[i][j]$ の間に遷移する．$T_B^{\min}[i][j]$ は接地点計画を要する最短時間に相当する．中間脚と最後脚は前脚が SupportE のときに発信するメッセージ $\mathrm{Entering}_{i,1}$, $\mathrm{Entering}_{i,2}$ をそれぞれ受け取って SwingC へ遷移する.

SwingC: 遊脚相（接地目標点への接近）

脚先が次の接地点 (前脚先端)$P_3^{i,j}$ に向かっている状態である．脚先が前脚の脚先に到達するまでの時間を $t_C[i][j]$ とすると，遷移条件は前の状態 SwingB と合わせた時間 t_C について $T_C^{\min}[i][j] \leq t_C[i][j] \leq T_C^{\max}[i][j]$ とする．この間に SupportD または E に遷移する．今回のモデルでは 4 章と異なり，SupportD または E への遷移は，どちらでも構わないものとする．この場合，UPPAAL での動作検証では，どちらに SupportD，E どちらに遷移した場合も網羅的に検証される.

SupportD: 接地相（体節の前進：後脚の可到達域外）

後脚の可到達域外に脚先を接地させながら体節を前進させている状態である．$\mathrm{Ground}_{i,j} = \mathrm{true}$ を返し，後脚と反対側の脚に同期信号 $\mathrm{Grounding}_{i,j}!$ を送信する．ただし，この $\mathrm{Ground}_{i,j}$ は制御オートマトンにおける制御モード 4 および時間オートマトンにおける接地ノード SupportD〜F に滞在していることを示す信号であって，力センサを用いて足場との接触を確認しているものではない．SupportD から SupportE への遷移時間についても $T_D^{\min}[i][j] \leq t_D[i][j] \leq T_D^{\max}[i][j]$ という遷移時間の範囲を与える.

SupportE: 接地相（体節の前進：後脚の可到達域内）

後脚の可到達域内で，脚先を接地させながら体節を前進させている状態である．ただし，制御モード 4 における滞在時間が下界未満の状態である．後脚の可到達域に侵入していることを示すため，$\mathrm{Entering}_{i,j} = \mathrm{true}$ を返し，SupportD もしくは E の状態にある後脚と同期を取る．本章の接地点追従法では，追加した支持脚条件に従って，特定の 3 脚が接地したことを確認してから脚を離す．遷移時間を $t_E[i][j]$ として接地が継続する時間 $t_{DE}[i][j](= t_D + t_E)$ に対して遷移条件を $T_{DE}^{\min}[i][j] \leq t_{DE}[i][j] \leq T_{DE}^{\max}[i][j]$ とする．

SupportF: 接地相（体節の前進：後脚の可到達域内）

後脚の可到達域内で，脚先を接地させながら体節を前進させている状態である．ただし，制御モード 4 における滞在時間が下界以上の状態である．この状態では，遊脚条件に従って，反対脚，後脚との同期をとって遊脚させる．ただし，この遊脚のタイミングは遷移条件を満たしつつ，同期条件による同期信号の受信が可能なタイミングである．この遷移条件には 6.2.2 節の制御モード 4 から 1 への遷移条件を記述しており，同期条件には後方または反対側の脚が接地している場合に同期信号を受信する条件 $\mathrm{Grounding}_{i,j+1}$？，$\mathrm{Grounding}_{1-i,j}$？ が記述されている．この同期条件による他脚 (後側または反対側) の接地の確認は遷移条件にも記述されているが，遷移条件とは状態遷移のタイミングを調整する上での役割が異なる．同期条件の場合は，接地が確認できる (同期信号を受信できる) 場合にすぐに遷移させるという役割を持つ．つまり，この同期条件と遷移条件を用いることで，この SupportF から SwingA への遷移は，制御モード $4 \to 1$ の遷移条件を満たせば「すぐに」実行される．これは実際の接地点追従法における制御モードの遷移のタイミングを再現していると言える．逆に，この同期条件が無い場合は，時間制約と遷移条件を満たす限りでの網羅的な検証が行われる．これが 4 章で使った時間オートマトンとは異なる．4 章で用いた時間オートマトンでは逆に同期条件を省略していたため，制御モード 4 から 1 への遷移条件を満たしていても遊脚のタイミングが遅れる場合も同時に検証されていたため，時間制約の導出結果が過保証であったと考えられる．

また，この状態における遷移時間を $t_F[i][j]$ として接地が継続する時間 $t_{DEF}[i][j](= t_D + t_E + t_F)$ に対して遷移条件を $t_{DEF}[i][j] \leq T_{DEF}^{\max}[i][j]$ とする．

SupportG: 脚先が可到達域の境界で引き摺られた状態

後脚と接地点交換する前に脚先が地を掻き切り，可到達域後方の境界に到達した状態である．実機上では脚先が可到達域を超えようとした場合に，関節のモータに無理な負荷が

掛かるのを防ぐため，脚の動作を停止させている．以後，脚先が引き摺られた状態になり，デッドロックし，接地点交換できなくなる．時間オートマトン上の SupportD，E，F ではこの可到達域に到達するまでの時間を $T_{DEF}^{\max}[i][j]$ とし，これを超えると SupportG に遷移するものとしている．

6.3.2　CTL(動作仕様)

モデル検査で使う動作仕様 4 章と同じように次のものを定義し，下にはモデル検査用の記述 CTL(Computational tree logic) を示す．4 章のものと同じであるため，説明を省略する．

Spec.1:　デッドロックにならない．

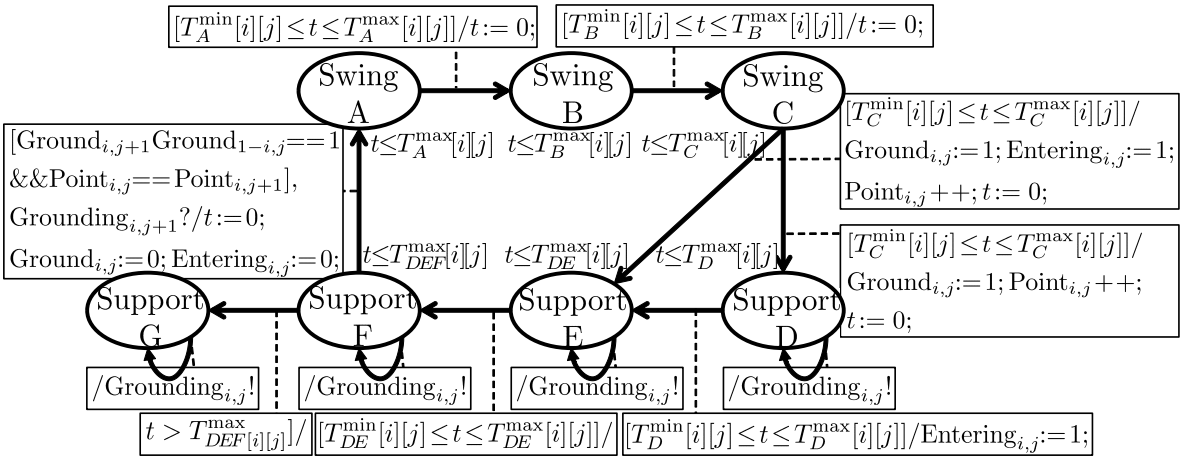
Spec.2:　SupportG に遷移しない．

Spec.3:　接地している点の数が合計 3 個以上である (ただし，この点は先頭脚の SwingB でセンサを用いて計画した接地点を指す)．

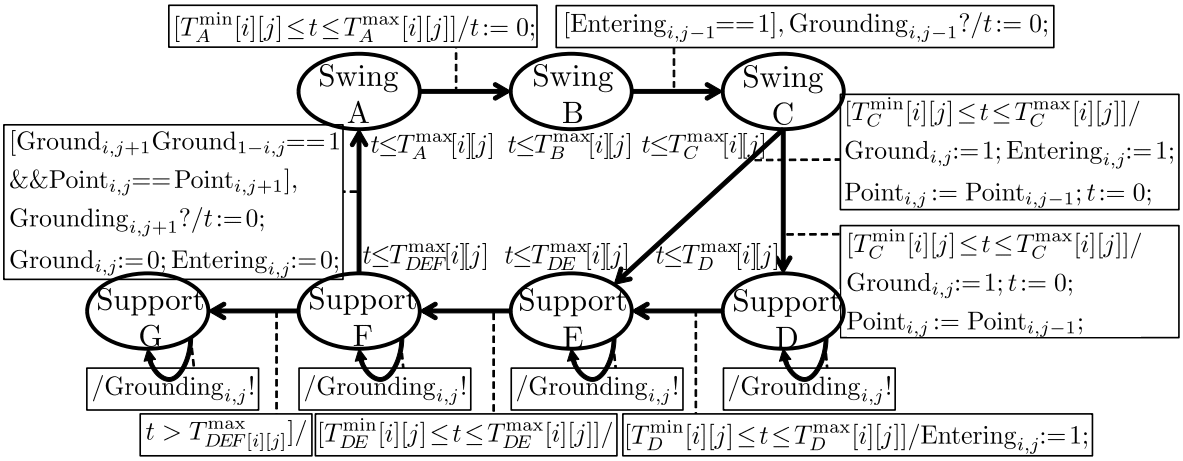
Spec.1:　EFAG¬(Leg$[i][j]$.Swing$\alpha \cup$ Leg$[i][j]$.Supportβ), $\forall i \in \{0,1\}, j \in \{1,2,3\}, \alpha \in \{A,B,C\}, \beta \in \{D,E,F,G\}$.

Spec.2:　AG¬Leg$[i][j]$.SupportG, $\forall i \in \{0,1\}, j \in \{1,2,3\}$.

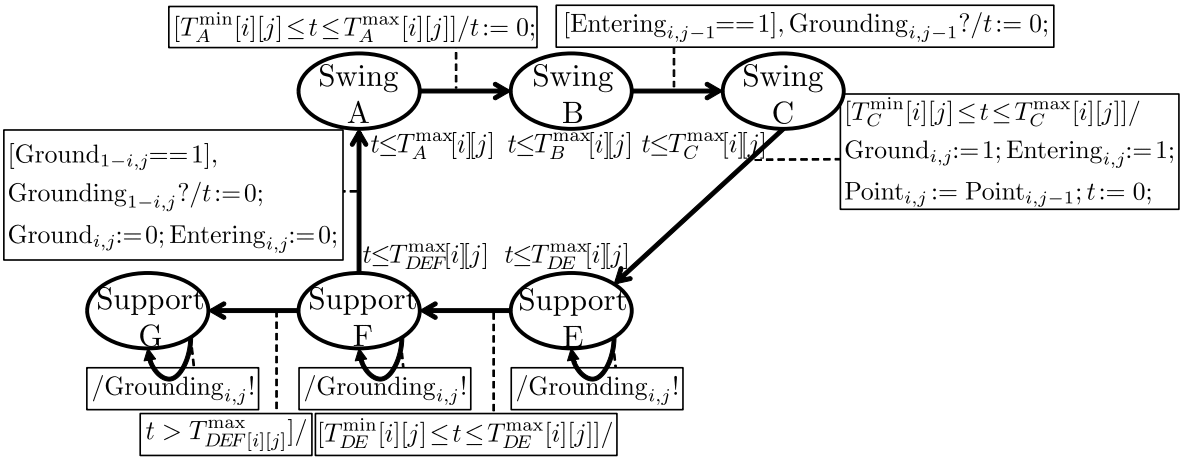
Spec.3:　$$\mathrm{AG} \sum_{\substack{i\in\{0,1\}\\ j\in\{1,2,3\}}} \mathrm{Ground}_{i,j} \prod_{l=1}^{j-1} (\mathrm{Point}_{i,j} \neq \mathrm{Point}_{i,l}\mathrm{Ground}_{i,l}) \geq 3.$$



(a)Timed automaton of forward legs ($j = 1$)



(b)Timed automaton of middle legs ($j = 2$)



(c)Timed automaton of rear legs ($j = 3$)

Fig. 6.2　Timed automata of all legs

6.3.3　時間制約の導出法の改良

次に時間制約の探索方法を説明する．4 章からの大きな改良点として，探索方法を全探索ではなく，ダイクストラ法 [101] を参考にした手法に変えている．4 章での探索方法は全ての時間制約について，0〜1000 の値を順番に代入し，仕様をみたしつつ評価が最大となる時間制約を探索した．探索する時間制約の数が 66 個の今回のケースにこの手法を用いると，1001^{66} 通りの時間制約に関するモデル検査が必要になり，現実的ではない．そこで本手法では時間制約に代入する自然数の組み合わせを 1 つのノードとして捉え，ダイクストラ法と同じようにある初期ノードを決めてから隣り合わせのノードのみについてモデルに代入・検証をし，評価が最大化される最適解までを辿る．そのため，実際にモデル検査を行う回数も抑えられる解の導出までの計算時間は 4 章より格段と速くなる．また，ダイクストラ法を使った理由は 2 つあり，1 つは，UPPAAL 上で時間制約として使える変数が自然数に限定されているため，連続関数を前提とした最適化手法が使えないことである．2 つ目は最適解への収束性だけでなく，動作仕様が厳密に保証されることも考慮するため，初期値から隣接するノード (時間制約の集合) でのモデル検査および評価関数の計算の両方が行える手法が必要だったからである．

手法の具体的な流れを Algorithm 4 を示す．まず，4〜8 行目で時間制約を探索するための初期値を決める．この初期値は仕様を満たす値である必要があるので，時間制約 $\{T_x^m[i][j]\} \equiv \{T_x^m[i][j] | m \in \{\min, \max\}, x \in \mathcal{A}, i \in \{0,1\}, j \in \{1,2,3\}\}$ にランダムな値を代入し仕様を満たすかを確かめ，満たさない場合はもう一度この操作を繰り返す．この操作の後得られた $\{T_x^m[i][j]\}$ を探索の初期値とする．次に $\{T_x^m[i][j]\}$ に隣接する値 $\{T_x^m[i][j]\}(= \{T_x^m[i][j]-1\}, \{T_x^m[i][j]\}, \{T_x^m[i][j]+1\})$ から仕様を満たしつつ評価関数を最大化する値をグラフ探索する．$\{T_x^m[i][j]\}$ を代入したモデルが仕様を満たすかを検証する．仕様を満たさない場合は $\{T_x^m[i][j]\}$ を直接 Closed list に格納し，仕様を満たす場合には Open list に格納する．Open list に格納された $\{T_x^m[i][j]\}$ のうち，評価関数 $f_{(\{T_x^m[i][j]\})}$ を最も大きくする $\{T_x^m[i][j]\}$ を Open list に残し，新しい $\{T_x^m[i][j]\}$ とする．ここで通常のダイクストラ法と異なり，Open list の残りの $\{T_x^m[i][j]\}$ を全て Closed list に移す．通常のダイクストラ法の場合，ゴールまでの最短経路を導出するため，最後まで Open list に格納するノードは複数あるが，今回は動作仕様を満たしつつ評価を最大化する時間制約の集合 1 つのみを求めるため，探索の過程で，評価を最大化できる 1 点のみを Open list に残している．

$\{T_x^m[i][j]\}$ に隣接する値に関して上記の操作を繰り返すことで，得られた時間制約は上記の仕様を必ず保証するだけでなく，最も脚の動きの自由度を広げるものになっている．

Algorithm 4 Finding of $T_X^{\min}[i][j], T_X^{\max}[i][j], X \in \mathcal{A}$

Define $\{T_X^m[i][j]\} \equiv \{T_X^m[i][j] | m \in \{\min, \max\}, X \in \mathcal{A}, i \in \{0,1\}, j \in \{1,2,3\}\}$
Define function:$f_{(\{T^m_{X[i][j]}\})} \leftarrow \prod_{\substack{m\in\{\min,\max\},X\in\mathcal{A}\\ i\in\{0,1\},j\in\{1,2,3\}}} (T_X^{\max}[i][j] - T_X^{\min}[i][j])$
Define Open list and Closed list.
while not all specifications are satisfied in model checking **do**
  $\{T_X^m[i][j]\} \leftarrow <$ random value $>$
  Substitute $\{T_X^m[i][j]\}$ into boundaries of transition time of timed automaton.
  Perform model checking for timed automaton by using UPPAAL.
end while
Store $\{T_X^m[i][j]\}$ into Open list.
flag=0
while flag$\neq$ 1 **do**
  for $t_X^m[i][j] = T_X^m[i][j] - 1$ **to** $T_X^m[i][j] + 1$ **do**
    if $\{t_X^m[i][j]\}$ is not included in Closed list **then**
      Substitute $\{T_X^m[i][j]\}$ into boundaries of transition time of timed automaton.
      Perform model checking for timed automaton by using UPPAAL.
      if all specifications are satisfied in model checking **then**
        Store $\{t_X^m[i][j]\}$ in Open list.
      else
        Store $\{t_X^m[i][j]\}$ in Closed list.
      end if
    else
      Store $\{t_X^m[i][j]\}$ in Closed list.
    end if
  end for
  if $\{T_X^m[i][j]\} < \arg\max_{\{t_X^m[i][j]\}\in\text{Open list}} f_{(\{t_X^m[i][j]\})}$ **then**
    $\{T_X^m[i][j]\} \leftarrow \arg\max_{\{t_X^m[i][j]\}\in\text{Open list}} f_{(\{t_X^m[i][j]\})}$
  else
    flag:=1
  end if
  Other than $\{T_X^m[i][j]\}$ in Open list are moved to Closed list.
end while
$\{T_X^m[i][j]\} \leftarrow \{kT_X^m[i][j]\}$ (k is arbitrary.)

6.3.4　時間制約の導出結果

導出した時間制約の最大値，最小値は Table. 6.1 に列挙する．ただし，時間制約そのものは Table. 6.1 の最大値，最小値を用いて (6.1) 式で表される．この時間制約の最大・最小値には 4・5 章に従ってこの時間制約には任意の正値をとる等価時間係数 k が掛けられている．

$$kT_X^{\min}[i][j] \leq t_X[i][j] \leq kT_X^{\max}[i][j], \forall i, \forall j, \forall X \tag{6.1}$$

まず，左右 ($i \in \{0, 1\}$) での時間制約を比較すると差は全くなかった．接地点追従法では，左脚と右脚で異なる制御則を与えているわけではないので，これは理にかなった結果と言える．今回は 6 脚全てに個別の時間制約を与えたが，このように制御則や振舞いが左右対称のシステムに関しては，左右で同じ時間制約を与えて，探索空間と探索時間を削減する 1 つの手段として考えられる．次に前後の脚 ($j \in \{1, 2, 3\}$) で比較すると，先頭脚と中間脚 ($j = 1, 2$) では同じ時間制約が得られたのに対して最尾脚 ($j = 3$) の遊脚相 ($X = A, B, C$) のみ全く異なる時間制約が得られた．6．1 節で挙げた仮説の通り，最尾脚については後方へ接地点を引き継ぐ制御を行わないため，動作仕様を保証するには他と

Table 6.1　Upper/lower boundaries of elapsed time in timed automaton

	$i=0$(Right side)			$i=1$(Left side)		
	$j=1$	$j=2$	$j=3$	$j=1$	$j=2$	$j=3$
$T_A^{\max}[i][j]$	2	2	3	2	2	3
$T_A^{\min}[i][j]$	0	0	2	0	0	2
$T_B^{\max}[i][j]$	2	2	3	2	2	3
$T_B^{\min}[i][j]$	0	0	2	0	0	2
$T_C^{\max}[i][j]$	2	2	3	2	2	3
$T_C^{\min}[i][j]$	0	0	2	0	0	2
$T_D^{\max}[i][j]$	6	6	–	6	6	–
$T_D^{\min}[i][j]$	0	0	–	0	0	–
$T_{DE}^{\max}[i][j]$	6	6	6	6	6	6
$T_{DE}^{\min}[i][j]$	0	0	0	0	0	0
$T_{DEF}^{\max}[i][j]$	22	22	22	22	22	22

は異なる脚の振る舞いが必要となる．また，この時間制約に基づいてモデル検査ツール UPPAAL 上のランダムシミュレーションを行い，制御モードの遷移の軌跡を Fig. 6.3 にプロットした．Fig. 4.13(a) とは対照的に遊脚のタイミングが後ろから前方へ伝搬されるような wave gait が確認され，遷移のタイミング 4 章よりも束縛されていないことが確認された．

これらの時間制約を次節の Timekeeper 制御に使うために，次の対応をとる．

$$
\begin{aligned}
T_1^{\max}[i][j] &\leftarrow T_A^{\max}[i][j],\\
T_1^{\min}[i][j] &\leftarrow T_A^{\min}[i][j],\\
T_2^{\max}[i][j] &\leftarrow T_B^{\max}[i][j],\\
T_2^{\min}[i][j] &\leftarrow T_B^{\min}[i][j],\\
T_3^{\max}[i][j] &\leftarrow T_C^{\max}[i][j],\\
T_3^{\min}[i][j] &\leftarrow T_C^{\min}[i][j],\\
T_4^{\max}[i][j] &\leftarrow T_{DEF}^{\max}[i][j],\\
T_4^{\min}[i][j] &\leftarrow T_{DE}^{\min}[i][j].
\end{aligned}
\tag{6.2}
$$

この対応に基づき，各時間制約の最大値・最小値は Table 6.2 で表される．ただし，Timekeeper 制御に用いる時間制約も次の式で与えられており，Table 6.2 に示すのは，単位時間 ($k = 1$ のとき) における最大値・最小値である．

$$
kT_x^{\min}[i][j] \leq t_x[i][j] \leq kT_x^{\max}[i][j], \forall i, \forall j, \forall x \tag{6.3}
$$

Timekeeper 制御では $T_x^{\max}[i][j], T_x^{\min}[i][j], x \in \{1, 2, 3, 4\}$ を用いる．

6.4　Timekeeper 制御の改良

5 章の Timekeeper 制御と異なり，時間制約を各脚に与えるので，これを考慮してアルゴリズムを改良する．まず，1〜4 行目では変数を定義する．ここでは全体扱う k を k_{all}

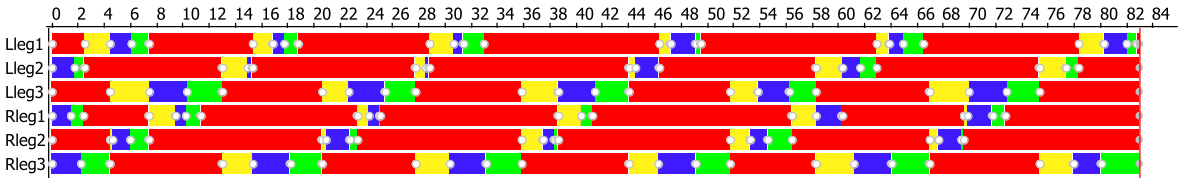


Fig. 6.3 Random simulation in UPPAAL (Mode1:Yellow, Mode2:Blue, Mode3:Green, Mode4:Red)

Table 6.2　Upper/lower boundaries of staying time in each control mode

	$i=0$(Right side)			$i=1$(Left side)		
	$j=1$	$j=2$	$j=3$	$j=1$	$j=2$	$j=3$
$T_1^{\max}[i][j]$	2	2	3	2	2	3
$T_1^{\min}[i][j]$	0	0	2	0	0	2
$T_2^{\max}[i][j]$	2	2	3	2	2	3
$T_2^{\min}[i][j]$	0	0	2	0	0	2
$T_3^{\max}[i][j]$	2	2	3	2	2	3
$T_3^{\min}[i][j]$	0	0	2	0	0	2
$T_4^{\max}[i][j]$	22	22	22	22	22	22
$T_4^{\min}[i][j]$	0	0	0	0	0	0

と表記する．前節で求めた時間制約から係数 k_{all} を省いた値を $T_x^{\max}[i][j], T_x^{\min}[i][j]$ 定数として扱う．$k[i][j]$ は滞在時間が時間制約の上限値 $T_x^{\max}[i][j](= k_{\mathrm{all}} T_x^{\max}[i][j])$ を上回らないように k_{all} を更新するために用いられる．以下，5 行以降はロボットの制御周期毎に繰り返される操作である．このループ内の処理は，各脚での分散処理と，$k[i][j]$ の値を全て受け取った上で k_{all} を更新する集中処理の 2 つに分かれる．分散処理においては，各脚の滞在時間が時間制約の下限値に満たない場合に脚の動きを遅らせる操作を行う（8 行目）．これにより，時間制約の下限値が保証される．次に滞在時間と，関数 $k[i][j]$ を更新する．この $k[i][j]$ がプログラムの中枢に送信され，滞在時間が上限値を保証するような k の値が得られる (16 行目)．また，上記の通り，Timekeeper 制御を実行するために各脚で送受信される変数は $k[i][j]$ だけである．今回の場合，この $k[i][j]$ の数は脚の数と同じ 6 本であるため，今回のように時間制約の数を拡大した場合でも，少ない計算量・通信量で Timekeeper 制御を実現することができる．

Algorithm 5 Timekeeper control

$t[i][j] = 0$:transition time in Leg$[i][j]$
$k_{\text{all}} = 1$:Non-dimensional coefficient of time
$T_x^m[i][j] = \frac{T_x^m[i][j]}{k_{\text{all}}}$:Time constraints without k_{all}
$k[i][j] = \frac{t[i][j]}{T_x^{\max}[i][j]}$:Function used for calculating k_{all}
while robot is walking **do**
  ▷ Distributed control on each leg
  for all $i \in \{0, 1\}, j \in \{1, 2, 3\}$ **do**
    if $t[i][j] \leq k_{\text{all}} T_x^{\min}[i][j]$ **then**
      Delay the movement of Leg$[i][j]$.
    end if
    Record transition time $t[i][j]$
    $k[i][j] \leftarrow \frac{t[i][j]}{T^{\max}_{(\text{Control mode of Leg}[i][j])[i][j]}}$
  end for
  ▷ Centralized calculation
  Receive $k[i][j]$ from all legs
  $k_{\text{all}} \leftarrow \max_{\substack{i \in \{0,1\}, \\ j \in \{1,2,3\}}} k[i][j]$
end while

Table 6.3　Success rate in experiment

	成功回数	失敗例でのデッドロック回数	他の要因の失敗回数
提案手法	35/40	0/5	5/5
従来手法	31/40	0/9	9/9

6.5　実機実験

実機実験により，今回の Timekeeper 制御の改良の効果および 5 章との歩容の変化を見る．Fig. 6.1 のように使用する実機上部には 3 つのセンサを取り付けている．遠隔操作用の深度センサとカメラ, 姿勢センサであり，前者 2 つのセンサ画像を元に遠隔 PC からロボットの先頭脚の接地点を選択する．前述の通り，先頭脚の接地点探索は人が代替する．比較対象としては，以下の 2 つを用意し，不整地環境上での歩容・k の縮尺の推移の違いを見る．前提として，どちらも Timekeeper 制御が実装されている．

- 提案手法：本手法の接地点追従法で脚毎に個別の時間制約を与えている．
- 従来手法：5 章の 6 脚用に改良した接地点追従法で全脚共通の時間制約を与えている．

本手法で使う接地点追従法は 6 章の各節で説明した追加の制御および時間制約によって制御される．比較対象の従来手法は，5 章の追加の制御および時間制約によって制御されるものである．この 2 つの大きな差は，接地点追従法の遊脚条件と時間制約，Timekeeper 制御の手法である．歩行環境及び，本手法の接地点追従法による歩行の様子は Fig. 6.4 に示す．

実験結果と考察

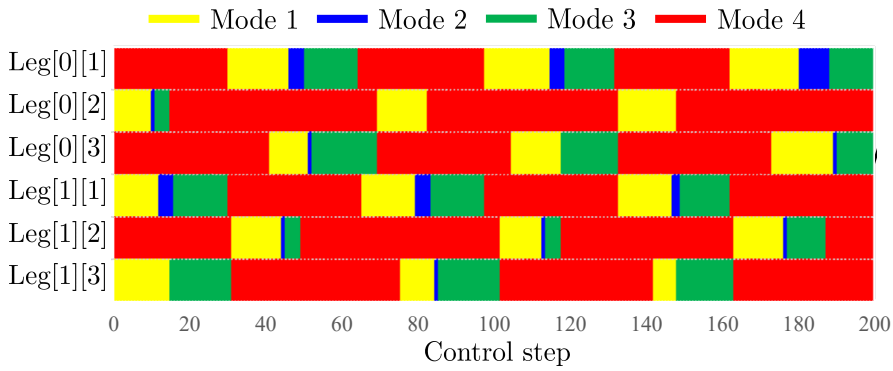
提案手法と従来手法での歩容 (モード遷移図) と時間制約の等価時間係数 k の推移を，Fig. 6.5 と Fig. 6.6 にそれぞれ示す．まず，歩容については本手法の場合，後方から前方の脚へ順番に遊脚している様子が見られ,wave gait に近い歩容になっており，基本的に 4 点以上の接地点を確保した歩容が見られる．いっぽうで，6 脚の接地点追従法の場合，遊脚のタイミングが 3 脚ずつで一致しており，結果的に tripod gait が生成されており，3 点支持が基本の歩行になっている．そのため，支持脚数の多い本手法の wave gait の方が不整地上を安定して歩行することができると考えられる．また従来手法の tripod gait の場合，3 脚の遊脚のタイミングをそろえる影響もあり，提案手法に比べて相対的に歩行周期が遅れる要因があると考えられる．その様子が特に顕著に見られるのが縮尺 k の推移

Fig. 6.4 Upper and side view of experiment with original FCP gait and Timekeeper control

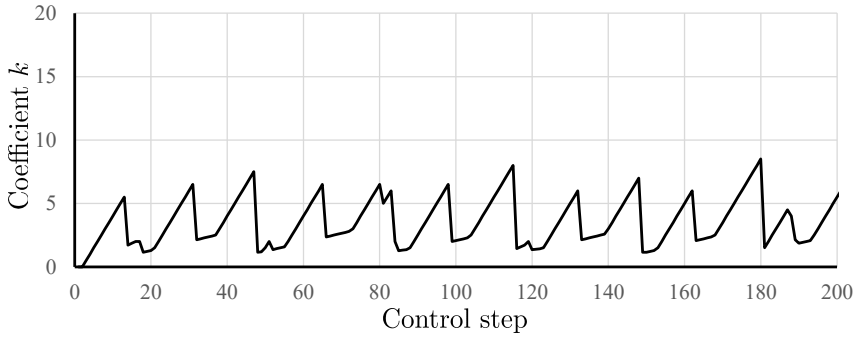
(b) であり，たとえば，本手法における k のピークは，各脚モード遷移のたびにだいたい $k = 6$∼8 の間の均等な値をとっている．いっぽうで，6 脚の接地点追従法におけるピークは遊脚の場合 $k = 16$，接地相の場合 $k = 8$ 前後を反復している．これは，遊脚と接地時とで滞在時間の k への影響が大きく異なることを示す．逆に提案手法 (元の接地点追従法，各脚の時間制約) では遊脚と接地時とで滞在時間の k への影響に差がないため，いずれかの制御モードの遅れのみで全体の動きが遅れる (k が大きくなる) という状況が従来手法に比べて避けられる傾向にある．これは本章の時間制約を探索する段階で，固定した k に対して時間制約ができるだけ広い幅 (Algorithm 1, 評価関数 $f_{(T_x^m[i][j])}$) を取れるように探索されたためである．

また，この不整地歩行を 40 回試行し，不整地を踏破した回数と転倒した回数の中で仕様を満たせなかった回数を数えたところ Table 6.3 のようになった．両者共に Timekeeper

制御の効果で動作仕様を満たせずに転倒することは一度もなかったが，従来手法に比べてわずかに提案手法の踏破回数が多くなった．主な理由としては，提案手法では 4 つ以上の接地点による歩容が主であるため，支持多角形をより広く取りやすく転倒を防ぎやすくしていたのである．このことから，提案手法の方が 6 脚ロボットの特徴である複数脚によるバランス維持能力が発揮されていると捉えることもできる．

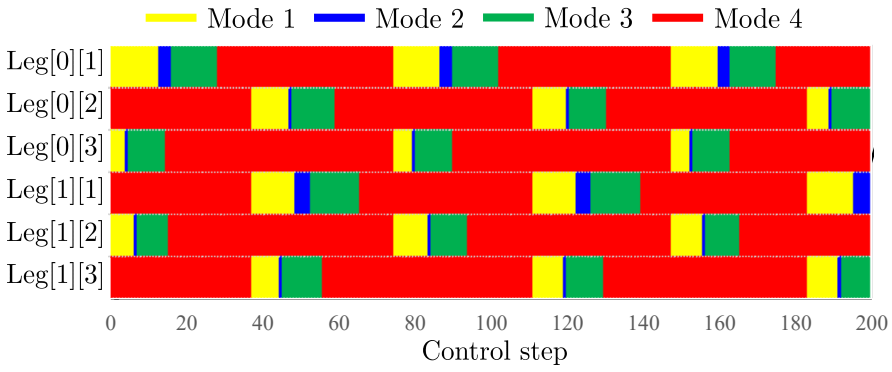


(a)Control mode

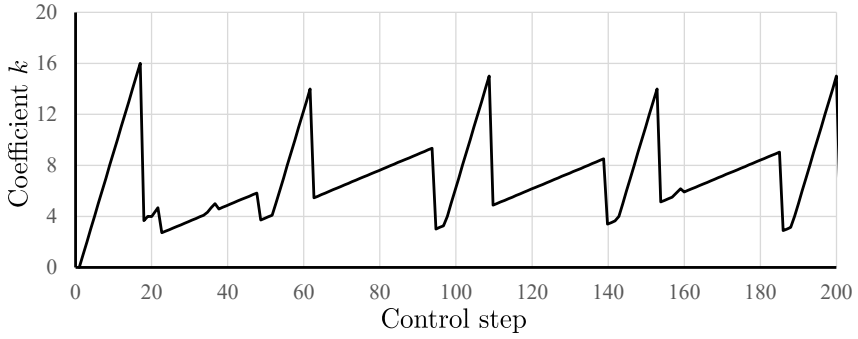


(b)Coefficient k

Fig. 6.5　Control mode and coefficient k in proposed method (1 control step:10[ms])



(a)Control mode



(b)Coefficient k

Fig. 6.6　Control mode and coefficient k in conventional method (1 control step:10[ms])

6.6　まとめ

本章では 6 脚移動ロボットに脚毎に個別の時間制約を与えた Timekeeper 制御を実装することで，元の接地点追従法による適応歩行を実現した．これにより，先行研究に比べて 4 点以上の支持脚を維持できるようになった上に，各制御モードの滞在時間の範囲 (=脚の動きの自由度) をより広くとれるようになった．同時に Timekeeper 制御の効果を利用することで，歩行に必要な追加の制御 (バランス制御・運動計画) を他の制御アーキテクチャと矛盾なく導入することができた．今後の課題としては Timekeeper 制御の改良案を模索し，適応的かつ効率的な歩行を実現させる．

また，今回は 6 脚ロボットが用いたが，さらに脚数の多いロボットにおいて Timekeeper 制御を適用した場合も同じ効果が見込める．全脚共通の時間制約を与えた場合と，各脚個別の時間制約を与えた場合とでは等価時間係数 k の推移に大きな違いが生じ，後者の方が等価時間係数 k を必要以上に大きくせず (全体の脚の動きを遅らせず)，歩行させることができると推測される．また，k を大きくしないことによって，不整地環境でも動作仕様を保証するロバストな制御器の開発を実現するだけなく，歩行運動の遅れを防ぐなどの歩行性能の向上が期待される．

第 7 章

結論

7.1　本研究の成果

本研究では，不整地環境において動作仕様を保証させる 6 脚移動ロボットの制御器設計の開発を実現した．不整地上の歩行を目的とした脚移動ロボットには，基本となる歩行のルール (歩行制御法)，制御センサ類による周囲環境の認識，それに対応する運動計画の手法，姿勢制御，接触センサを用いた緊急回避や局所制御等の複数の制御要素を統合した制御器の設計が必要になる．この制御器による歩行の動作検証はロボットの開発環境で行われるため，他の歩行環境でロボットを動作させた場合，制御器に含まれる制御要素同士の干渉が起こる可能性がある．この制御要素同士の干渉によってはロボットの歩行制御法が止まる (デッドロックを起こす) こともある．本研究では，この「デッドロックを起こさない」という目標の動作仕様とロボットの歩行制御によるロボットの振る舞いをモデル検査手法に基づき解析することで，デッドロックを起こさず，かつ静歩行を維持 (3 点以上の接地点を維持) するロバストな制御器の開発手法を提案した．

その過程でまず，ロボットの動作仕様とロボットの動作を形式言語によりモデル化し，動作仕様を保証させる時間制約を導出した．そしてこの時間制約には縮尺に関する自由度が任意の値をとるという特徴があった．このことを利用して，時間制約の縮尺を歩行の状況によって変更しつつ保証させる Timekeeper 制御を提案することで，環境依存の無い動作仕様の保証と制御要素同士の統合が実現された．

また，本手法は 6 脚移動ロボットを対象として研究を行ったが，同じような事象駆動型システム・離散事象システムで，振る舞いと動作仕様が時間オートマトンと CTL により記述・解析できる場合に同じ手順での実装が可能であると考えられる．ただし，手法の適用方法についてはシステムの性能によって 4 章の接地可能領域の導出・5 章の Timekeeper 制御のどちらを参考にするべきかを考慮する必要がある．システムの時間制約に関して，

絶対時間に関する制約がある場合には，4 章のように時間制約の縮尺を固定する必要があるし，そうでない場合は 5 章の手法に沿った実装が望ましい．また，6 章においてもエージェント毎に異なる時間制約を与えられる場合にのみシステムへの実装が可能である．

本論文では以下の手順に従って議論を展開し，ロボットの制御器開発手法を提案した．

1. モデル検査手法を応用したシステムが動作仕様を保証するための時間制約の導出（3，4 章）
2. 時間制約を常に保証させる制約充足型統合制御「Timekeeper 制御」の提案（5 章）
3. エージェント毎の振る舞いの違いを考慮した Timekeeper 制御の改善（6 章）

2 章では，本手法の制御器に関する問題に取り組む前に使用するシステムに使用する事象駆動型分散歩行制御法接地点追従法の説明をした．2.2 節で，ロボットの行動に必要な時間制約を導出し，本論文全体で使う変数の定義 (脚のラベル等) を説明した．この変数定義はロボットの構造が，左右 2 脚を有するセグメントを複数連ねた脚移動ロボット全てを対象とした一般化した形で記述した．2.3 節で接地点追従法における具体的な制御則，制御オートマトンの説明をした．この制御オートマトンは後章でも使われる．2.4 節では接地点追従法に着手した先行研究・関連研究を説明した．この節では，本研究の前任者だけでなく，現行の共同研究者の紹介も行い，彼らの研究と本研究の関係性についても説明している．2.5 節では，本研究が進むにつれて接地点追従法の細部が変更されていることを述べ，4〜6 章でどのような改良が施されているかを述べている．以降の章では接地点追従法およびそれに追加される制御を 2 章で述べた内容に付与する形で説明した．

3 章では，本研究で用いるモデル検査の定義について説明している．2.2，2.3 節ではモデル検査を包含する形式検証，形式手法の説明をした．形式手法は理論計算科学を基盤としたハードウェア・ソフトウェアシステムの仕様記述，開発検証技術全般を指すのに対し，その中の形式検証は数学理論に基づきシステムの動作に関する証明を行う手法，さらにモデル検査は実際のソフトウェア・ハードウェアシステムの設計において導出されたモデルが形式仕様を満たすかを証明する動作であり，本研究のように 6 脚ロボットのシステムの動作検証開発に使う場合はモデル検査が適している．2.4 節でそのモデル検査を検査用ツール UPPAAL に限定して説明した．UPPAAL で使うモデルは時間オートマトンと呼ばれるシステムの振る舞いを時間で抽象化したものである．そして動作仕様は CTL(Computational tree logic) により記述され，システムの動作検証に用いられる．4 章以降で使う仕様の説明のためにここでは，時間オートマトンと CTL の記述方法も述べている．4 章以降はこの説明を基にシステムのモデル化を行う．

4 章では，モデル検査に基づきシステムが動作仕様を保証するための時間制約を導出した．このシステムとはここでは，接地点追従法を含む複数の制御要素による 6 脚ロボッ

トの制御系，動作仕様はこの制御系がデッドロックを起こさない，ロボットが支持多角形を維持するなどのことを指す．4.2 節で具体的なロボットの制御系の内容を説明した．4.3 節ではこの制御系を時間オートマトンでモデル化し，時間制約を導出する手法を述べた．また，この節では野村ら [80] の手法との比較としてロボットの多様な振る舞いを時間オートマトン内の時間制約として記述していることも述べた．4.4 節では導出された時間制約を制御で用いる 1 つのアプローチとして時間制約を脚の運動に関する物理的なパラメータに対応させ，それを制御に使うことを提案した．この対応の際には，時間制約の縮尺をロボットのスペックから逆算した上で，脚の接地可能領域やロボットの動きに対応させ，最後にシミュレーションにより動作仕様が保証されつつロボットが歩行できているかも確認している．

5 章では，4 章の時間制約を脚の運動に対応させる都合上ロボットの動きを極端に制限してしまったことを 1 つの問題として取り上げ，別の手法としてスーパーバイザ制御的にシステムに時間制約を保証させる制約充足型統合制御「Timekeeper 制御」を提案した．5.2 節では 4 章の流れと同様にロボットの制御系に関する説明をしている．5.3 節で Timekeeper 制御の内容とその実装方法について述べた．Timekeeper 制御はモデル検査によって得られた時間制約の特徴と時間の不可逆性を利用した制約充足手法であり，ロボットが時間制約を破る際に時間制約の縮尺を大きくするか，脚の動きを遅らせるかの 2 択の制御介入を行う．これによって，システムの遅延等にも対応した制約充足が可能となった．5.4，5.5 節では Timekeeper 制御によるシミュレーションと実機実験の内容を提示している．各々 k の更新則を変えた手法を使っており，手法毎にメリット・デメリットが生じているが，いずれの場合もシステムの動作仕様 (デッドロック回避，3 点以上の接地点) を常に保証した歩行が実現できていることを確認した．

6 章では，5 章の制約充足型統合制御において残された課題を取り上げた．それは時間制約をすべての脚で共通のものとして扱っており，これにより 1 つの問題が生じたことである．4 章の方法で，共通の時間制約を求めようとしたが，動作仕様を満たす時間制約の集合が得られなかったことから動作仕様を満たすために接地点追従法にロボットの運動を制限する遊脚条件を加えてしまい，歩容が tripod gait に制限されてしまったことである．この運動の制限は不整地歩行において多様な運動・緊急回避をとることの弊害となるので，除去する必要があった．そこで，この 6 章では時間制約を脚毎に分けるように 4 章・5 章の Timekeeper 制御を改良することで，ロボットの振る舞いがどのように変更されるかを解析した．6.3 節で 4 章に対応するモデル検査を用いた時間制約の導出方法および Timekeeper 制御の改善を行い，6.4 章で実機実験を通して本改良によるロボットの振る舞いの変化を確認した．脚の振る舞いは主に遊脚のタイミングが制限されなくなり，運動の自由度が増えていることが確認されただけでなく，Timekeeper 制御において時間制

約の縮尺 k も必要以上に大きくせずに歩行できることも実証している．k を大きくするとロボットの全体の動きが遅くなることから，k の上昇を抑えられることはロボットの歩行効率や速度を向上させる上でも重要な事である．

7.2　今後の課題と展望

以上のような研究から，事象駆動型システムを対象とした動作仕様を保証させる制約充足型統合制御「Timekeeper 制御」を確立することができた．本論文は 6 脚ロボットを対象とした研究を行ったが，他の事象駆動型システムに適用した例はない．そのため，本研究には他の離散事象システムでの Timekeeper 制御の実装検証の他，膨大な課題が残されている．このような課題の一部と，本研究の今後の展望について，以下に述べる．

1. **転倒安定性のための制御要素の更なる導入**　本論文では，制御器内の制御要素同士を統合させる (干渉を防ぐ) 手法として Timekeeper 制御を提案したがその制御要素の導入は不十分である．たとえば，接触センサを使った脚の回避運動や接地力向上，耐故障制御，歩行環境を考慮した歩容の切り替えなど本手法で提案した Timekeeper 制御の基，制御要素を導入する余地は十分にあると考えられる．ただし，接触センサを使った脚の回避運動は共同研究者の藤井ら [87] が現在進行している．
2. **多脚ロボットの脚欠損時の耐故障制御の提案**　以前，6 脚ロボットの場合，脚 1 つまたは 2 つが欠損してもロボットが歩行できるような接地点追従法の改良ができるのではないかと共同研究者と議論していた．本研究当初の 3 つ目の研究計画はこの 6 脚ロボットの脚欠損の耐故障制御法の提案とこれに関する 4・5 章の手法の適用であったが，途中経過で 6 章の手法を提案したために先延ばしになった．後任の研究者には是非この課題に取り組んでいただきたい．
3. **実機における先頭脚の接地点探索の自動化**　本手法では接地点探索をヒューマンインザループ制御により半自律的に遂行したが，共同研究者の鈴木らは，A*アルゴリズムを使った接地点探索の手法を提案している．しかし，この手法の有効性はシミュレーション内でしか確認されていない．実機実装をし，実環境における自動化の検証をしなければならない．
4. **他の事象駆動型システムへの Timekeeper 制御の適用**　前述したようにこの手法の適用事例は 6 脚ロボット以外にない．まず，多脚ロボットの枠組みでは 4，8 脚およびそれ以上の脚数に関しての適用例を作り，6 脚ロボットへの実装時には確認できなかった問題点が無いかを確かめる必要がある．ロボットとは異なる他の離散事象システムについても実装上の新たな問題が見つかる可能性がある．たとえば 5 章で取り上げたような時間制約の縮尺 k の更新方法に関する問題も挙がる事だろう．

5. **時間制約の探索方法に関する改善**　6 章で最終的にダイクストラ法を参考にして時間制約の探索を用いて行っているが，これが最善の手法とは限らない．また，別の用途・ツールであるが，モデル検査のそのものの自動化に関する研究もされている [102], [103]．モデル検査ツール UPPAAL での時間制約は自然数しか使うことができないので，使用できる探索手法・最適化手法は限定されるが，モデル検査ツールの汎用性を高めるためにさらに効率的な時間制約の探索手法を提案する必要がある．
6. **Timekeeper 制御における等価時間係数 k を抑える方法**　5 章以降では，時間の不可逆性から各制御モードの滞在時間をある時刻から減少させられない前提で Timekeeper 制御を提案した．そのため，時間制約自体は保証できるものの，滞在時間が伸びた場合に等価時間係数 k が大きくなり，全体の動きが極端に遅くなる様子も確認された．こういった k の上昇を抑えるためには，各脚でできる限り速く次の制御モードに遷移させ，滞在時間を短くすればよい．そのためには，周囲環境を加味しながらどのような脚の動きが滞在時間を最小化できるかを予測する必要ある．なぜならこの滞在時間の上昇の原因の 1 つとして，脚先の障害物の衝突等の歩行環境の影響を大きく受けることが考えられるからである．
   たとえば，整地歩行の場合は，脚の動きが遅れる要因が少ないため，ロボットのスペックの許容範囲で移動速度と制御モードの遷移の速度を上げて滞在時間と k を最小化することができる．いっぽうで，複雑な未知の不整地環境であれば脚の動きを速めるにしても，障害物と脚との衝突可能性も考慮する必要がある．接地点追従法では，この周囲環境の認識・運動計画を先頭脚で行っているため，先頭脚で滞在時間を最小化する運動計画を行い，これを後続脚へ引き継ぐことで k を抑えたままの歩行ができると考えられる．

参考文献

[1] 磯部祥尚, 櫻庭健年, 田口研治ほか. ソフトウェア科学基礎—最先端のソフトウェア開発に求められる数理的基礎 (トップエスイー基礎講 座), 2008.

[2] 松野文俊. 阪神淡路大震災を振り返って. 日本ロボット学会誌, Vol. 28, No. 2, pp. 138–141, 2010.

[3] 淺間一. 災害時に活用可能なロボット技術の研究開発と運用システムの構築. 日本ロボット学会誌, Vol. 32, No. 1, pp. 37–41, 2014.

[4] 松野文俊. レスキューロボットの現状と課題 (特集 消防・防災と人工知能 (ai)). 消防防災の科学, No. 133, pp. 16–20, 2018.

[5] F. Delcomyn, M.E. Nelson. ”architectures for a biomimetic hexapod robot”. *Robotics and Autonomous Systems*, Vol. 30, No. 1, pp. 5–15, 2000.

[6] Yasushi Mae Tomohito Takubo, Kenichi Ohara and Tatsuo Arai. ”adaptive gait for dynamic rotational walking motion on unknown non-planar terrain by limb mechanis robot asterisk”. *Journal of Robotics and Mechatronics*, Vol. 25, No. 1, pp. 172–182, 2013.

[7] Toyomi Fujita and Tiga Sasaki. ”consideration on a crawler robot with six legs”. *In Proceedings of The 2016 International Conference on Artificial Life and Robotics (ICAROB 2016)*, Vol. 25, No. 1, pp. 88–91, 2016.

[8] 三浦偉志, 畠山省四朗, 高橋佑平, 岩瀬将美, 根本琢磨. 側転移動を可能とする可変構造クモ型ロボットの開発 : 滑らかな変形の考察. システム制御情報学会研究発表講演会講演論文集, Vol. 62, p. 4, 2018.

[9] M. Bjelonic, N. Kottege, P. Beckerle. ”proprioceptive control of an over-actuated hexapod robot in unstructured terrain”. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2042–2049, 2016.

[10] Axel Schneider, Jan PaskarbeitMalte, SchillingJosef Schmitz. ”hector, a bio-inspired and compliant hexapod robot”. *Living Machines 2014: Biomimetic and Biohybrid Systems*, pp. 427–429, 2014.

[11] A. Roennau, G. Heppner, M. Nowicki, R. Dillmann. "lauron v: A versatile six-legged walking robot with advanced maneuverability". In *2014 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, pp. 82–87, 2014.

[12] Dominik Belter, Piotr Skrzypczyński. "population-based methods for identification and optimization of a walking robot model". In *Robot Motion and Control 2009. LNCIS*, 第 396 巻, pp. 185–195, 2009.

[13] Marcel Bergerman, John Billingsley, John Reid, and Eldert van Henten. Robotics in agriculture and forestry. In *Springer handbook of robotics*, pp. 1463–1492. Springer, 2016.

[14] John E Bares and David S Wettergreen. Dante ii: Technical description, results, and lessons learned. *The International Journal of robotics research*, Vol. 18, No. 7, pp. 621–649, 1999.

[15] 上原拓也, 菅沼直孝. 4 足歩行ロボット開発までの環境変化. 日本ロボット学会誌, Vol. 32, No. 2, pp. 139–140, 2014.

[16] P. Fankhauser, M. Bjelonic, C. Dario Bellicoso, T. Miki, and M. Hutter. Robust rough-terrain locomotion with a quadrupedal robot. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5761–5768, 2018.

[17] Yuan Tian and Feng Gao. Efficient motion generation for a six-legged robot walking on irregular terrain via integrated foothold selection and optimization-based whole-body planning. *Robotica*, Vol. 36, No. 3, pp. 333–352, 2018.

[18] Ruiqin Li, Hongwei Meng, Shaoping Bai, Yinyin Yao, and Jianwei Zhang. Stability and gait planning of 3-upu hexapod walking robot. *Robotics*, Vol. 7, No. 3, p. 48, 2018.

[19] Yue Zhao, Xun Chai, Feng Gao, and Chenkun Qi. Obstacle avoidance and motion planning scheme for a hexapod robot octopus-iii. *Robotics and Autonomous Systems*, Vol. 103, pp. 199–212, 2018.

[20] Baoling Han, Xiao Luo, Rui Zhao, Qingsheng Luo, and Guanhao Liang. The optimization algorithm for gait planning and foot trajectory on the quadruped robot. In *International Conference on Geometry and Graphics*, pp. 1274–1279. Springer, 2018.

[21] H. Igarashi, T. Machida, F. Harashima, and M. Kakikura. Free gait for quadruped robots with posture control. In *9th IEEE International Workshop on Advanced Motion Control, 2006.*, pp. 433–438, 2006.

[22] SHUAISHUAI ZHANG, YIBIN LI, RUI SONG, XUEWEN RONG, and BIN LI. A free gait planning method based on the foothold search strategy for quadruped

robot. In *ASSISTIVE ROBOTICS: Proceedings of the 18th International Conference on CLAWAR 2015*, pp. 461–468. World Scientific, 2016.

[23] JingYe He, JunPeng Shao, GuiTao Sun, and Xuan Shao. Survey of quadruped robots coping strategies in complex situations. *Electronics*, Vol. 8, No. 12, p. 1414, 2019.

[24] Shaoping Bai, Kian Hsiang Low, Gerald Seet, and Teresa Zielinska. A new free gait generation for quadrupeds based on primary/secondary gait. In *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No. 99CH36288C)*, Vol. 2, pp. 1371–1376. IEEE, 1999.

[25] Shaoping Bai, Kin Huat Low, and Teresa Zielinska. Quadruped free gait generation based on the primary/secondary gait. *Robotica*, Vol. 17, No. 4, pp. 405–412, 1999.

[26] Shaoping Bai, KH Low, and Teresa Zielinska. Quadruped free gait generation for straight-line and circular trajectories. *Advanced Robotics*, Vol. 13, No. 5, pp. 513–538, 1998.

[27] Y Fukuoka and H Kimura. Dynamic locomotion of a biomorphic quadruped 'tekken' robot using various gaits: walk, trot, free-gait and bound. *Applied Bionics and Biomechanics*, Vol. 6, No. 1, pp. 63–71, 2009.

[28] Jonas Buchli and Auke Jan Ijspeert. Distributed central pattern generator model for robotics application based on phase sensitivity analysis. In *International Workshop on Biologically Inspired Approaches to Advanced Information Technology*, pp. 333–349. Springer, 2004.

[29] Jiang Shan, Cheng Junshi, and Chen Jiapin. Design of central pattern generator for humanoid robot walking based on multi-objective ga. In *Proceedings. 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000)(Cat. No. 00CH37113)*, Vol. 3, pp. 1930–1935. IEEE, 2000.

[30] Shinkichi Inagaki, Hideo Yuasa, Takanori Suzuki, and Tamio Arai. Wave cpg model for autonomous decentralized multi-legged robot: Gait generation and walking speed control. *Robotics and Autonomous Systems*, Vol. 54, No. 2, pp. 118–126, 2006.

[31] Chengju Liu, Qijun Chen, and Jiaqi Zhang. Coupled van der pol oscillators utilised as central pattern generators for quadruped locomotion. In *2009 Chinese Control and Decision Conference*, pp. 3677–3682. IEEE, 2009.

[32] Chengju Liu, Danwei Wang, and Qijun Chen. Central pattern generator inspired control for adaptive walking of biped robots. *IEEE Transactions on Systems,*

Man, and Cybernetics: Systems, Vol. 43, No. 5, pp. 1206–1215, 2013.

[33] Erick Israel Guerra-Hernandez, Andres Espinal, Patricia Batres-Mendoza, Carlos Hugo Garcia-Capulin, Rene De J Romero-Troncoso, and Horacio Rostro-Gonzalez. A fpga-based neuromorphic locomotion system for multi-legged robots. *IEEE Access*, Vol. 5, pp. 8301–8312, 2017.

[34] Matthew D Berkemeier and Kamal V Desai. Control of hopping height in legged robots using a neural-mechanical approach. In *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No. 99CH36288C)*, Vol. 3, pp. 1695–1701. IEEE, 1999.

[35] S. Ito, Yuuichi Sahashi, and Minoru Sasaki. Learning scheme of multiple-patterns in quadruped locomotion using cpg model. In *2008 SICE Annual Conference*, pp. 132–137, 2008.

[36] Norikazu Sugimoto and Jun Morimoto. Phase-dependent trajectory optimization for cpg-based biped walking using path integral reinforcement learning. In *2011 11th IEEE-RAS International Conference on Humanoid Robots*, pp. 255–260. IEEE, 2011.

[37] Kazuya Nishigai and Kazuyuki Ito. Control of multi-legged robot using reinforcement learning with body image and application to a real robot. In *2011 IEEE International Conference on Robotics and Biomimetics*, pp. 2511–2516. IEEE, 2011.

[38] Winfried Ilg and Karsten Berns. A learning architecture based on reinforcement learning for adaptive control of the walking machine lauron. *Robotics and Autonomous Systems*, Vol. 15, No. 4, 1995.

[39] Kazuyuki Ito and Fumitoshi Matsuno. A study of reinforcement learning for the robot with many degrees of freedom-acquisition of locomotion patterns for multi-legged robot. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, Vol. 4, pp. 3392–3397. IEEE, 2002.

[40] M Anthony Lewis, Andrew H Fagg, and Alan Solidum. Genetic programming approach to the construction of a neural network for control of a walking robot. In *Proceedings 1992 IEEE International Conference on Robotics and Automation*, pp. 2618–2623. IEEE, 1992.

[41] Jose Hugo Barron-Zambrano, Cesar Torres-Huitzil, and Bernard Girau. Configurable embedded cpg-based control for robot locomotion. *International Journal of Advanced Robotic Systems*, Vol. 9, No. 3, p. 92, 2012.

[42] Wenlu Li, Weihai Chen, Xingming Wu, and Jianhua Wang. Parameter tuning

of cpgs for hexapod gaits based on genetic algorithm. In *2015 IEEE 10th Conference on Industrial Electronics and Applications (ICIEA)*, pp. 45–50. IEEE, 2015.

[43] M Anthony Lewis, Andrew H Fagg, and George A Bekey. Genetic algorithms for gait synthesis in a hexapod robot. In *Recent trends in mobile robots*, pp. 317–331. World Scientific, 1993.

[44] Manuel F Silva, Ramiro S Barbosa, and JA Tenreiro Machado. Development of a genetic algorithm for the optimization of hexapod robot parameters. In *Proceedings of the IASTED International Conference on Applied Simulation and Modelling (ASM'09)*, pp. 77–82, 2009.

[45] Makoto Dohi, Tateshi Fujiura, Noriaki Ishizuka, and Kazuyoshi Nonami. Gait control by genetic algorithm for agricultural hexapod walking robot. *IFAC Proceedings Volumes*, Vol. 33, No. 29, pp. 89–93, 2000.

[46] Gary B Parker. Evolving gaits for hexapod robots using cyclic genetic algorithms. *International journal of general systems*, Vol. 34, No. 3, pp. 301–315, 2005.

[47] Ya-guang Zhu, Bo Jin, Wei Li, and Shi-tong Li. Optimal design of hexapod walking robot leg structure based on energy consumption and workspace. *Transactions of the Canadian Society for Mechanical Engineering*, Vol. 38, No. 3, pp. 305–317, 2014.

[48] Dennis Krupke, Norman Hendrich, Jianwei Zhang, and Houxiang Zhang. Gait optimization based on physics simulation of 3d robot models with a modular robotic simulation system. In *ASSISTIVE ROBOTICS: Proceedings of the 18th International Conference on CLAWAR 2015*, pp. 612–619. World Scientific, 2016.

[49] Sonia Chernova and Manuela Veloso. An evolutionary approach to gait learning for four-legged robots. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, Vol. 3, pp. 2562–2567. IEEE, 2004.

[50] Gary B Parker and William T Tarimo. Using cyclic genetic algorithms to learn gaits for an actual quadruped robot. In *2011 IEEE International Conference on Systems, Man, and Cybernetics*, pp. 1938–1943. IEEE, 2011.

[51] Takeshi Kano, Dai Owaki, and Akio Ishiguro. A simple measure for evaluating gait patterns during multi-legged locomotion. *SICE Journal of Control, Measurement, and System Integration*, Vol. 7, No. 4, pp. 214–218, 2014.

[52] L. Teng, X. Wu, W. Chen, and J. Wang. Central pattern generators of adaptive frequency for locomotion control of quadruped robots. In *IEEE 10th International Conference on Industrial Informatics*, pp. 586–590, 2012.

[53] L. Matthey, L. Righetti, and A. J. Ijspeert. Experimental study of limit cycle and chaotic controllers for the locomotion of centipede robots. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1860–1865, 2008.

[54] S. Aoi, T. Tanaka, S. Fujiki, K. Senda, and K. Tsuchiya. Experimental verification of cusp catastrophe in the gait transition of a quadruped robot driven by nonlinear oscillators with phase resetting. In *5th IEEE RAS/EMBS International Conference on Biomedical Robotics and Biomechatronics*, pp. 825–830, 2014.

[55] Kotaro Yasui, Kazuhiko Sakai, Takeshi Kano, Dai Owaki, and Akio Ishiguro. Decentralized control scheme for myriapod robot inspired by adaptive and resilient centipede locomotion. *PLOS ONE*, Vol. 12, pp. 1–12, 2017.

[56] Yasuhiro Fukuoka, Hiroshi Kimura, and Avis H. Cohen. Adaptive dynamic walking of a quadruped robot on irregular terrain based on biological concepts. *The International Journal of Robotics Research*, Vol. 22, No. 3-4, pp. 187–202, 2003.

[57] Weihai Chen, Guanjiao Ren, Jianbin Zhang, and Jianhua Wang. Smooth transition between different gaits of a hexapod robot via a central pattern generators algorithm. *Journal of Intelligent & Robotic Systems*, Vol. 67, No. 3, pp. 255–270, 2012.

[58] 梅枝真守, 梶原秀一, 田中孝之, 金子俊一. 周期入力制御を用いた準受動的歩行ロボットの斜度・外乱に対するロバスト性. 計測自動制御学会論文集, Vol. 42, No. 8, pp. 974–981, 2006.

[59] Uluc Saranli, Martin Buehler, and Daniel E Koditschek. Rhex: A simple and highly mobile hexapod robot. *The International Journal of Robotics Research*, Vol. 20, No. 7, pp. 616–631, 2001.

[60] Robert T Schroer, Matthew J Boggess, Richard J Bachmann, Roger D Quinn, and Roy E Ritzmann. Comparing cockroach and whegs robot body motions. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, Vol. 4, pp. 3288–3293. IEEE, 2004.

[61] Shin-Min Song and Byoung Soo Choi. A study on continuous follow-the-leader (ftl) gaits: an effective walking algorithm over rough terrain. *Mathematical Biosciences*, Vol. 97, No. 2, pp. 199 – 233, 1989.

[62] Holk Cruse, Thomas Kindermann, Michael Schumm, Jeffrey Dean, and Josef Schmitz. Walknet: a biologically inspired network to control six-legged walking. *Neural Networks*, Vol. 11, No. 7, pp. 1435 – 1447, 1998.

[63] S. Inagaki, T. Niwa, and T. Suzuki. Follow-the-contact-point gait control of centipede-like multi-legged robot to navigate and walk on uneven terrain. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5341–5346, 2010.

[64] T. Yamaguchi, T. Jumpei, H. Okuda, Y. Tazaki, T. Suzuki, T. Ito, and K. Muto. Driver assistance control based on model predictive computation of constraint satisfaction. In *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, pp. 1304–1310, 2015.

[65] Pim van Leeuwen, Henk Hesselink, and Jos Rohling. Scheduling aircraft using constraint satisfaction. *Electronic Notes in Theoretical Computer Science*, Vol. 76, pp. 252 – 268, 2002. WFLP 2002, 11th International Workshop on Functional and (Constraint) Logic Programming, Selected Papers.

[66] R. E. Young, R. Giachetti, and D. A. Ress. A fuzzy constraint satisfaction system for design and manufacturing. In *Proceedings of IEEE 5th International Fuzzy Systems*, Vol. 2, pp. 1106–1112, 1996.

[67] D. Ambrosino and Anna Sciomachen. A constraint satisfaction approach for master bay plans. 1970.

[68] Thomas B. Sheridan. Telerobotics, automation, and human supervisory control. 1992.

[69] Benoit Gaudin and Hervé Marchand. Supervisory control of concurrent discrete event systems. 2004.

[70] Panagiotis Kouvaros and Alessio Lomuscio. Automatic verification of parameterised multi-agent systems. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems*, AAMAS '13, pp. 861–868. International Foundation for Autonomous Agents and Multiagent Systems, 2013.

[71] Louise Dennis, Michael Fisher, Marija Slavkovik, and Matt Webster. Formal verification of ethical choices in autonomous systems. *Robotics and Autonomous Systems*, Vol. 77, pp. 1 – 14, 2016.

[72] M. M. Quottrup, T. Bak, and R. I. Zamanabadi. Multi-robot planning : a timed automata approach. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, Vol. 5, pp. 4417–4422, 2004.

[73] L. Li, Z. Shi, Y. Guan, C. Zhao, J. Zhang, and H. Wei. Formal verification of a

collision-free algorithm of dual-arm robot in hol4. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1380–1385, 2014.

[74] A. Classen, M. Cordy, P. Schobbens, P. Heymans, A. Legay, and J. Raskin. Featured transition systems: Foundations for verifying variability-intensive systems and their application to ltl model checking. *IEEE Transactions on Software Engineering*, Vol. 39, No. 8, pp. 1069–1089, 2013.

[75] Tomoya Niwa, Shinkichi Inagaki, and Tatsuya Suzuki. Locomotion control of multi-legged robot based on follow-the-contact-point gait. In *2009 ICCAS-SICE*, pp. 2247–2253. IEEE, 2009.

[76] SHINKICHI INAGAKI, TOMOYA NIWA, and TATSUYA SUZUKI. Navigation control and walking control on uneven terrain for centipede-like multi-legged robot based on fcp gait control. In *Emerging Trends In Mobile Robotics*, pp. 656–663. World Scientific, 2010.

[77] Ryo Takahashi and Shinkichi Inagaki. Walk control of segmented multi-legged robot based on integrative control of legs and 2-dof active intersegment joints. *Advanced Robotics*, Vol. 30, No. 20, pp. 1354–1364, 2016.

[78] 黄可濱, 稲垣伸吉, 鈴木達也ほか. 時間オートマトンによる多脚歩行ロボットの動作検証. 平成 22 年電気学会産業応用部門大会講演論文集, p. 71, 2010.

[79] 近藤悠太, 稲垣伸吉, 鈴木達也. 反射を組み込んだ接地点追従法によるムカデ型多脚歩行ロボットの分散歩行制御〜引っ掛かりの解消による不整地踏破能力の改善〜. 第 23 回自律分散システム・シンポジウム, pp. 269–274, 2011.

[80] Keisuke Nomura and Shinkichi Inagaki. Cutting a parameter space for a multi-legged robot based on model checking. *SICE Journal of Control, Measurement, and System Integration*, Vol. 10, No. 4, pp. 317–323, 2017.

[81] Shundo Kishi and Shinkichi Inagaki. Graph-search based footstep planning for multi-legged robots on irregular terrain by using depth-sensor. In *Mobile Service Robotics*, pp. 417–424. World Scientific, 2014.

[82] 岸俊道, 栗田啓喜, 稲垣伸吉. 2a2-o02 接地点追従法と接地点探索に基づく 6 脚移動ロボットの不整地歩行と実機検証. ロボティクス・メカトロニクス講演会講演概要集 2015, pp. _2A2–O02_1. 一般社団法人 日本機械学会, 2015.

[83] 出島貴将, 村田勇樹, 稲垣伸吉. 対側脚間の非同期制御に基づくムカデ型ロボットの歩行性能の向上. ロボティクス・メカトロニクス講演会講演概要集 2016, pp. 1A2–07a1. 一般社団法人 日本機械学会, 2016.

[84] 鈴木義久, 村田勇樹, 稲垣伸吉. 接地点追従法と接地点指令による 6 脚移動ロボットの歩行. ロボティクス・メカトロニクス講演会講演概要集 2017, pp. 1A1–B10. 一般

社団法人 日本機械学会, 2017.

[85] 鈴木義久, 村田勇樹, 稲垣伸吉. 接地点追従法に基づく 6 脚移動ロボットの接地点計画. 第 31 回自律分散システム・シンポジウム, pp. 1C3–2, 2019.

[86] 木俣岳志, 出島貴将, 稲垣伸吉. 平行リンク機構型 3 自由度体節間関節を持つムカデ型ロボットの実機開発. ロボティクス・メカトロニクス講演会講演概要集 2018, pp. 2P2–I06. 一般社団法人 日本機械学会, 2018.

[87] 藤井海斗, 村田勇樹, 稲垣伸吉, 鈴木達也. 3d プリンタを用いた分布型接触センサの開発と事象駆動型 6 脚移動ロボットの反射制御. ロボティクス・メカトロニクス 講演会 2019.

[88] 多良氣, 木俣岳志, 村田勇樹, 稲垣伸吉. ムカデ型ロボットにおける接地点追従法の時間制約の導出. 計測自動制御学会システム・情報部門学術講演会 2019.

[89] 情報システムの信頼性向上のためのガイドライン. `https://www.meti.go.jp/policy/it_policy/softseibi/index.html`.

[90] 中島震. オブジェクト指向デザインと形式手法. コンピュータ・ソフトウェア, 2001.

[91] 吉岡信和, 田辺良則, 田原康之, 長谷川哲夫, 磯部祥尚. モデル検査による設計検証. コンピュータ ソフトウェア, Vol. 31, No. 4, pp. 4_40–4_65, 2014.

[92] Takemasa Arakawa and Toshio Fukuda. Natural motion trajectory generation of biped locomotion robot using genetic algorithm through energy optimization. In *1996 IEEE International Conference on Systems, Man and Cybernetics. Information Intelligence and Systems (Cat. No. 96CH35929)*, Vol. 2, pp. 1495–1500. IEEE, 1996.

[93] Filipp Seljanko. Hexapod walking robot gait generation using genetic-gravitational hybrid algorithm. In *2011 15th International Conference on Advanced Robotics (ICAR)*, pp. 253–258. IEEE, 2011.

[94] Zoltán Pap, István Kecskés, Ervin Burkus, Fulop Bazso, and Péter Odry. Optimization of the hexapod robot walking by genetic algorithm. In *IEEE 8th International Symposium on Intelligent Systems and Informatics*, pp. 121–126. IEEE, 2010.

[95] Nate Kohl and Peter Stone. Policy gradient reinforcement learning for fast quadrupedal locomotion. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, Vol. 3, pp. 2619–2624. IEEE, 2004.

[96] Erik Schuitema, Daan GE Hobbelen, Pieter P Jonker, Martijn Wisse, and JG Daniël Karssen. Using a controller based on reinforcement learning for a passive dynamic walking robot. In *5th IEEE-RAS International Conference on*

Humanoid Robots, 2005., pp. 232–237. IEEE, 2005.

[97] Hajime Kimura, Toru Yamashita, and Shigenobu Kobayashi. Reinforcement learning of walking behavior for a four-legged robot. *IEEJ Transactions on Electronics, Information and Systems*, Vol. 122, No. 3, pp. 330–337, 2002.

[98] István Kecskés and Péter Odry. Optimization of pi and fuzzy-pi controllers on simulation model of szabad (ka)-ii walking robot. *International Journal of Advanced Robotic Systems*, Vol. 11, No. 11, p. 186, 2014.

[99] Jack M Wang, David J Fleet, and Aaron Hertzmann. Optimizing walking controllers. In *ACM Transactions on Graphics (TOG)*, Vol. 28, p. 168. ACM, 2009.

[100] J.L.D. Fac. A generalized reduced gradient algorithm for solving large-scale discrete-time nonlinear optimal control problems. *IFAC Proceedings Volumes*, Vol. 22, No. 2, pp. 45 – 50, 1989. 8th IFAC Workshop on Control Applications of Nonlinear Programming and Optimization 1989, Paris, France, 7-9 June 1989.

[101] Dijkstra and Edsger W. A note on two problems in connexion with graphs. *Numerische mathematik*, Vol. 1, No. 1, pp. 269–271, 1959.

[102] 高田沙都子, 森奈実子, 村田由香里. モデル検査自動化ツールの開発 検査自動化と反例解析効率化 . 第 74 回全国大会講演論文集, Vol. 1, pp. 269–271, 2012.

[103] 池田彩恵, 松浦佐江子. 要求分析段階におけるモデル検査技術を用いた設計制約検証の自動化. 第 80 回全国大会講演論文集, Vol. 1, pp. 211–212, 2018.

謝辞

本研究の遂行，本論文の編纂のみならず，終始多岐に渡る並々ならぬ御指導，御鞭撻を賜りました名古屋大学大学院工学研究科機械システム工学専攻 稲垣伸吉准教授には，心より格別の御礼を申し上げます．

本論文をまとめるにあたり，数々の御指導，御助言を賜りました，名古屋大学大学院工学研究科機械システム工学専攻 鈴木達也教授，名古屋大学大学院工学研究科機械システム工学専攻 東俊一教授，岐阜大学工学部機械工学科知能機械コース 伊藤聡教授に，厚く御礼申しあげます．

加えて，本研究内容を議論するにあたり，同 奥田裕之助教，同 川島明彦特任助教，同山口拓真特任助教，同研究員 Anh Tuan，TRAN 氏，神戸大学大学院工学研究科機械工学専攻 田崎勇一准教授に，ご指導を頂きましたことを心より感謝申し上げます．

また，本研究を進めるにあたりまして，本学後輩諸氏であり，共に 6 脚・多脚ロボットの研究開発に従事した出島貴将氏，鈴木義久氏，木俣岳志氏，藤井海斗氏，多良氣氏，同講座の一期生として共に研究に励んだ陳 ヒョン兌氏，さらには名古屋大学工学研究科技官の方々には，多大なるご助力賜りましたことを深く感謝いたしますとともに，日々助言や励ましを頂きました名古屋大学の数々の先輩，友人，後輩諸氏に感謝いたします．

最後となりましたが，筆者の我侭に対して，長きに渡り暖かく，そして惜しみなく支援して下さいました父，母に，何よりの感謝を込めつつ，本論文の締めくくりとさせていただきます．