情報処理学会論文誌 Vol.57 No.8 1690–1702 (Aug. 2016)

Event-Bを使った時間依存性のあるシステムのモデル化と動作分析のケーススタディ

來間 啓伸[1,a)] 中島 震[2,3,b)]

受付日 2015年11月18日, 採録日 2016年5月17日

概要：組込みシステムのソフトウェア設計では，環境に起こりうる変化と，それに対するシステムの反応の間の整合性を，網羅的に検証することが重要である．ここで，環境およびシステムの事象の間には，遅延など，時間間隔に関する制約である時間依存性が存在しうるので，それらを考慮した検証を行うことが課題になる．本稿では，自動車ドアロックシステムを題材とした動作分析のケーススタディについて報告し，時間依存性のある事象の間で局所的に時間経過をカウントするモデル化方法を提案する．形式モデリング手法 Event-B を用い，環境の変化とシステムの反応をイベントとして表現する一方，システムが満たすべき要件を不変条件として表した．環境やシステムと独立に進行する共通時間を明示的に導入することなく時間経過をカウントすることで，時間依存性に関わる検証が容易になる．定理証明支援ツールによる不変条件の充足性証明を使って，安全性要件である車速がどのように変化しても一定速度以上であればドアがロックされていることを，形式モデル上で検証することができた．一方，本手法が有効となる検証可能モデルを構成するための方法論の整備が，今後の課題である．

キーワード：Event-B，形式手法，定理証明，時間依存モデル

Modeling and Behavior Analysis Case-study of a Time Dependent System in Event-B

Hironobu Kuruma[1,a)] Shin Nakajima[2,3,b)]

Received: November 18, 2015, Accepted: May 17, 2016

Abstract: Proving the consistency between the system's responses and the changes that may occur in its environment is an important task in software design verification of embedded systems. Since the phoenomena in a system and its environment are generally related through the progress of time, modeling of time is a problem to be solved in constructing models of systems and environments. In this paper, we pick up a car door lock system and report a case study of formal modeling and behavior analysis. We propose a modeling method for time dependent systems in which the progress of time is counted locally by concerning events. The changes in the environment and the responses of the system are modeled as events and the properties which the system should maintain are modeled as invariants in Event-B. Since the time dependency between events are represented without introducing actual time, the verification processes of timing properties become simpler. We show a safety property, the doors are always locked when the velocity of the car exceeds a fixed value, is proved independent of the acceleration and deceleration patterns. Our future work is to develop a methodology for constructing verifiable time dependent models.

Keywords: Event-B, formal methods, theorem proving, time dependent model

[1] 株式会社日立製作所研究開発グループ
Research and Development Group, Hitachi, Ltd., Totsuka, Yokohama 244–0817, Japan
[2] 国立情報学研究所
National Institute of Informatics, Chiyoda, Tokyo 101–8430, Japan
[3] 総合研究大学院大学
The Graduate University for Advanced Studies, SOKENDAI, Chiyoda, Tokyo 101–8430, Japan
[a)] hironobu.kuruma.zg@hitachi.com
[b)] nkjm@nii.ac.jp

1. はじめに

ソフトウェア開発のための形式手法（Formal Methods）は 1970 年代に研究が開始され，その後も手法の発展と支援ツールの開発，適用先の拡大が進められてきた [27]．形式手法を使う主な利点として，機能仕様の厳密な記述およびプログラムの正しさの検証による信頼性の向上があげられる．機能仕様の形式記述と，形式検証に裏付けられたプ

ログラム導出の有効性は，B メソッド [1] を用いたパリ地下鉄システムの開発などにおいて実証された [5], [6]．また，上流工程であるシステムの動作分析への形式手法の適用も試みられている [2]．これは，システムが動作する環境を含む形式モデルを作成するとともに，検証を通じてシステムと環境の間の整合性を明確にした後，ソフトウェアの機能仕様を作成するものである．環境との整合性が明確になった記述から機能仕様を作成することで，考慮漏れのない機能仕様を得ることが期待できる．ここで，環境およびシステムの事象は，一般に時間経過を通して関連する．事象間の時間間隔に関する制約を，本稿では時間依存性と呼ぶ．すなわち，ある事象（トリガイベント）の発生から関連する事象（レスポンスイベント）の発生までの時間間隔に制約がある場合，時間依存性があるとする．時間依存性は，3 つのカテゴリに分類される [24]．

- 期限（deadline）
  トリガイベントの後，指定された時間内にレスポンスイベントが起こらなければならない．
- 遅延（delay）
  レスポンスイベントが起こるならば，トリガイベントの発生から指定された時間の後でなければならない．
- 終了（expiry）
  レスポンスイベントが起こるならば，トリガイベントの発生から指定された時間内でなければならない．

時間依存性はシステムの動作分析において無視できない要因となりうるので，モデル上でどのように表現するかは重要な課題である．

本稿では，形式モデリング手法 Event-B を使った，組込みシステムのモデル化と動作分析のケーススタディについて報告する．このケーススタディでは，時間依存性のうち，遅延の問題を扱う．ケーススタディの対象には，単純化した自動車ドアロックシステムをとりあげた [20]．このシステムは車速を周期的にモニタし，車速が規定値以上になればドアを自動的にロックする．このとき，ドアをロックするアクチュエータの動作時間は無視できない程度に長く，その間も車速はシステムとは独立に変化し続ける．

このようにシステム外部の環境から周期的に情報を得，その値に応じてアクションを行うシステムは，モニタ–コントロールシステム [25] と呼ばれ，実時間システムの標準的なシステムクラスに位置づけられる．このため，本ケーススタディから得られた知見は，組込みシステムの広い範囲をカバーすると期待される．一般に，実時間システムが要求された性質を満たすことの評価は，環境の事象の発生が予測できないことから困難であり，広範囲にわたるシステム評価とシミュレーションが必要であるとされる [25]．ドアロックシステムにおいても車速は環境によって決定され，システムの動作分析のためには起こりうるあらゆる車速変化を考慮しなければならない点で，本質的な困難さを持つ一方，適切な規模に単純化した例題と考えることができる．

本稿のケーススタディの目的は，時間依存性のもとで安全性要件の充足性を論理的に検証できる形式モデルが構成可能であることを示し，そのためのモデル化方法を提案することにある．すなわち，Event-B を使ってモデルを構成するとともに，車速がある値以上であればドアがロックされているという安全性要件をモデルが満足することを，論理的な検証によって示す．このことにより，従来は設計，実装，評価の繰返しによって開発されていた実時間システムの開発効率と信頼性を，システム開発の早い段階で形式モデルを作成し要件を満たすための条件を明確にすることで，向上させることが期待できる．

以下，2 章で本稿の時間依存性のモデル化方法の概要を示す．3 章では Event-B の概要を紹介し，4 章でケーススタディの対象について述べる．モデル化の方針については，5 章で触れる．モデルとその上での動作分析を 6 章に示し，7 章では得られた結果について述べる．関連研究を 8 章にあげるとともに，既存の時間依存性モデル化方法を使ったモデルと動作分析の概要を付録 A.1 に掲げ，本稿の時間依存性モデル化方法と比較する．

2. 時間依存性のモデル化

計算システムの時間に関する性質をモデル化し分析するため，さまざまな時間モデリングが提案されてきた [9]．時間モデリングの中でも，時間依存性のモデル化は主要なトピックである．Jackson ら [12] は，リアクティブシステムを環境からの刺激に対して次の刺激を受けるまでに反応するシステムと位置付け，リアクティブシステムでモデル化できない時間依存性のあるシステムを，イベントとイベントの間の状態に時間間隔を持たせることでモデル化する．この方法では，時間間隔に制約のあるイベントを単純なモデルのもとに表現できるが，時間にかかわる性質の解析においては，イベントに加えて状態を考慮する必要があり，検証が複雑になる．一方，Lamport [17] の実時間モデリングでは，時間を記録する実数型の変数を設けるとともに，時間を進めるイベント（以下では，Tick_Tock イベントと呼ぶ）をモデルに明示的に導入する．イベントの間の時間間隔に関する制約は，イベント間で共通に時間を参照し，その相対値を得ることで表現される．モデルの中で共通に参照される時間は，私たちが実世界の中で持つ時間の概念に近く，イベントと時間を結び付けやすい．その反面，各イベントが 1 つの時間軸上に配置されるので，時間を参照するすべてのイベントにとって適切なタイミングで時間が進むように，Tick_Tock イベントをモデルに記述しなければならない．また，イベント間の関係を共通時間からの相対値を介して解析するので，実数の加減算を含む解析になる．

ここで，システムの動作分析に関わる事象がモデル上にイベントとして表現されていることを前提とすると，動作分析において関心があるのは時間依存性のもとで発生するイベントの列であり，イベントが起こる時間の値そのものは分析に影響しない．また，時間依存性はトリガイベントとレスポンスイベントの間の時間間隔であることから，この 2 つの間でのみ時間を進行させれば十分である．すなわち，時間依存性のあるシステムの動作分析に関しては，実時間モデリングに現れる共通時間を進行させるだけの Tick_Tock イベントは，モデルから削除しても影響がない．本稿では，トリガイベントとレスポンスイベントの間の時間間隔を，両者の間で発生するイベントによってカウントする時間依存性のモデル化方法を提案する．このことによりモデルが簡潔になり，安全性要件の充足性検証が容易になることを示す．

トリガイベントとレスポンスイベントの間で時間が進行するという意味で，時間経過のカウントは局所的である．このため，別のトリガイベントとレスポンスイベントの組に対しては，両者の組に共通するイベントがあればそれらを使ってカウントすることで時間を共有し，共通するイベントがない場合には時間間隔に重なりがないので独立した時間を割り当てればよい．このモデルにはすべてのイベントに共通する時間はないが，時間の経過を通じた順序関係を持つイベントの解析により，システムの安全性検証が行える．イベントが局所的に時間経過をカウントする点では Jackson・Zave のモデルとも共通するが，状態はモデルに明示的には現れないので，記述を簡単化し検証の手間を軽減することができる．

3. 形式モデリング手法 Event-B

3.1 モデリングスタイル

Event-B はシステム分析を目的とした形式モデリング手法であり，EU の RODIN および DEPLOY プロジェクト [29] で開発が進められた．Event-B は B メソッドを継承しつつ，システム分析への形式手法適用を目的としている点に特徴がある．

Event-B では，自発的に起こるできごと（イベント）とそれに対する反応（アクション）に着目して，対象をモデル化する．ここで，イベントはアトミックであり，発生と終了の間に時間経過はない．**図 1** に示すように，Event-B の形式モデルは，対象の静的な側面を表すコンテクストと，動的な側面を表す抽象機械から構成される．抽象機械は，一般に複数のコンテクストを参照することができる．コンテクストは，データを構成する要素の集合（台集合）と定数の宣言およびこれらの間に成り立つ関係を表す公理からなり，形式モデルの「データ構造」を与える．一方，抽象機械は変数と不変条件およびイベントを持つ．不変条件（以下 I）は変数に対する制約条件であり，イベントによって変数の値が変化してもつねに満たされなければならない．また，コンテクストや抽象機械には，定理を記述することができる．定理は，公理や不変条件および他の定理から，論理的に証明されなければならない．

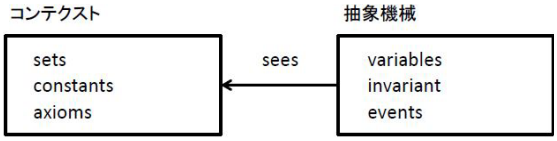

図 1 Event-B 形式モデルの構成要素
Fig. 1 Components of formal model in Event-B.

イベントの宣言は，下記のようにイベントパラメータとガード条件ならびにアクションから構成される．

$$\texttt{any}\ x\ \texttt{where}\ G(x,v)\ \texttt{then}\ Q(x,v,v')\ \texttt{end}$$

ここで，イベントパラメータ x はイベント内で使われる一時変数，v は抽象機械の変数，ガード条件 G はイベントが起こるための必要条件である．アクション Q は，アクション前の変数値 v とアクション後の変数値 v' の間の関係を表す論理式であり，イベントによる変数値の変化を表す．

3.2 リファインメント

Event-B は，複雑な形式モデルを作成するための道具としてリファインメントを用いる．すなわち，単純な形式モデルに，リファインメントを通じて詳細を付加し，目的とする形式モデルを構成する．形式モデルの記述と検証を段階的に行うことで，複雑な形式モデルを容易に作成できる．Event-B のリファインメントは前向き模倣（forward simulation）であり，**図 2** のようにリファインメント前の変数 v とリファインメント後の変数 w の間に糊付け不変条件（gluing invariant）が成り立つとき，具象イベント後の w' に対して糊付け不変条件が成り立つ抽象イベント後の v' が存在することが求められる．ここで，抽象イベントはリファインメント前の抽象機械のイベント，具象イベントはリファインメント後の抽象機械において抽象イベントに対応するイベントである．糊付け不変条件（以下 J）はリファインメント前後の変数の値を対応付ける不変条件であり，リファインメント後の抽象機械に記述する．

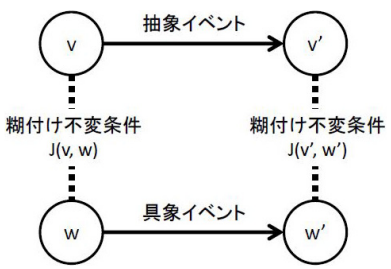

図 2 リファインメント
Fig. 2 Refinement.

Event-B のリファインメントでは，変数を置き換えるデータ詳細化だけでなく重ね合わせ（superposition）も可能であり，リファインメントの過程で新たな変数を加えてそれに作用するイベントを追加することができる．

3.3 モデルの整合性の検証

Event-B の形式モデルの整合性検証では，不変条件がつねに満たされることを検証する．Event-B のツールである RODIN では，検証手段として定理証明とモデル検査の 2 つが利用できる．

定理証明による検証では，記述された形式モデルに対して次の 3 つの証明課題を，RODIN が自動生成する．

(1) 変数の初期値が不変条件を満たすこと
$Q_{init}(v') \Rightarrow I(v')$

(2) ガード条件が満たされたときにアクションが実行可能であること
$I(v) \wedge G(x,v) \Rightarrow \exists v'.Q(x,v,v')$

(3) アクション実行後の変数値が不変条件を満たすこと
$I(v) \wedge G(x,v) \wedge Q(x,v,v') \Rightarrow I(v')$

イベントが具象イベントの場合には，さらに次の 3 つの証明課題が加わる．ここで，$H(x,w)$ と $R(x,w,w')$ は，各々具象イベントのガード条件とアクションである．

(1) 具象イベントのガード条件が抽象イベントのガード条件より強いこと
$I(v) \wedge J(v,w) \wedge H(x,w) \Rightarrow G(x,v)$

(2) 抽象イベントのアクションと具象イベントのアクションが模倣関係にあること
$I(v) \wedge J(v,w) \wedge H(x,w) \wedge R(x,w,w') \Rightarrow Q(x,v,v')$

(3) イベント後の変数値の間に糊付け不変条件が成り立つこと
$I(v) \wedge J(v,w) \wedge H(x,w) \wedge R(x,w,w') \Rightarrow J(v',w')$

また，コンテクストや抽象機械に定理を記述した場合には，それを証明することを求める証明課題が生成される．

証明課題が証明できたとき，形式モデルは整合していると見なせる．証明は定理証明支援ツールを使って行うが，すべての証明がツールによって自動的に行われるのではなく，証明過程で人間が対話的に指示を与えることが必要な場合もある．

モデル検査による検証では，形式モデルがとりうる状態を網羅的に探索して，不変条件と糊付け不変条件がつねに成り立つことを検証する．また，時相論理の式を与えて，イベントの順序に関する性質を検証することもできる．その反面，検証対象は状態空間が有限の形式モデルに制限される．一方，形式モデルの擬似的な実行（仕様アニメーション）にモデル検査メカニズムを使う際には，網羅的探索は必要ないので，有限の状態空間に限定されない．Event-B で記述された形式モデルのためのモデル検査ツールとして知られている ProB [18] は仕様アニメーションもサポートしており，イベントが意図した順序で発生することを確認できる．

定理証明による検証と比較すると，モデル検査による検証は，状態空間が有限であれば機械的な状態探索による自動検証が可能である．状態空間が有限でない場合には，形式モデルに制約を加えて状態空間を有限にしなければならない．状態空間が有限でない場合でも，仕様アニメーションは有効である．

4. ケーススタディの題材

4.1 題材の概要

本稿では，モデリングの題材として，次の機能を持つ自動車ドアロックシステムを考える．

- 車速がアンロック操作上限値以下のとき
  乗員の操作によって，ドアをロック/アンロックする．
- 車速がアンロック操作上限値を超えているとき
  乗員のアンロック操作を無視する．
- 車速が自動ロック下限値以上のとき
  ドアがロックされていなければ，ロックする．

簡単のため，進行方向の違いを無視して，車速を速度の大きさで表す．これは 0 以上である．また，前進と後進を切り換える際には，必ずいったん停止するものとする．

ドアロックシステムは，**図 3** のようにセンサ，コントローラ，アクチュエータから構成される．

- センサは一定の時間間隔で周期的に車速を測定する．
- コントローラは，センサからの車速値と乗員の操作によってロック/アンロックを判断し，アクチュエータに指示する．
- アクチュエータはコントローラの指示に従ってロック/アンロック動作を行う．

ここで，センサが出力する車速は整数値とし，コントローラは離散演算によりロック/アンロックを判断する．コントローラの判断にかかわる基本的な値として，以下では次の定数を使う．

- c_low：コントローラが乗員のドアアンロック操作を受け付ける上限のセンサ値
- c_high：コントローラが自動ドアロックを指示するセンサ値

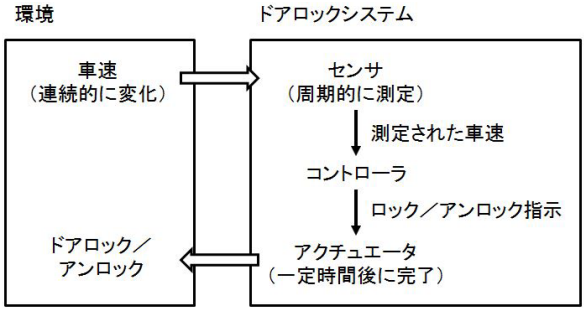

図 3 ドアロックシステムの構成（文献 [20] より）
Fig. 3 Door lock system.

- *l_limit*：ドアがロックされていることを保証する下限のセンサ値

なお，センサ値が *c_low* と *c_high* の間にあるときは，乗員のロック操作がない限り，コントローラは現状維持を指示するものとする．

前提条件として，コントローラの判断時間とアクチュエータへの指示伝達時間の和は．センサの測定周期に比べて十分短いとする．すなわち，コントローラはセンサから車速値を受け取って動作を判断し，次に車速値を受け取るまでには判断を終えてアクチュエータに指示を伝え終わる．一方，アクチュエータが動作してドアがロック/アンロックされるまでの時間は，機械的動作をともなうのでセンサの測定周期よりも長いとする．このため，アクチュエータが動作中に測定される車速値が変化し，それにともなってコントローラの指示が変わる場合がある．

4.2 モデル化の課題

このケーススタディでは，車速，ドアのロック状態，乗員によるロック/アンロック指示が環境に属する．要件は，ドアロック下限速度以上であればドアがロックされていることである．ここで，コントローラが判断し指示している間にも車速は変化するが，コントローラがそれを知るのは，次にセンサから値を受け取るときである．以下では，4.1 節の前提のもと，センシング周期による遅れとアクチュエータ動作時間を考慮してコントローラの動作をモデル化し，その動作を分析する．

このシステムの環境は実世界であり，環境を含む形式モデルを構成し分析するうえでは，物理量ならびに時間的変化をどのようにモデル化するかが課題となる．また，システムも物理的に独立した構成要素から構成され，それらの間には制約が存在する．モデル化の課題をまとめると，以下のようになる．

- 物理量のモデル化
  一般に物理量は実数であり，連続的に変化する．このケーススタディでは車速が物理量であり，ドアロックシステムの振舞いは車速によって決定される．
- 時間とともに変化する事象のモデル化
  車速は，システムの動作とは独立に，センサ周期の間も変化し続ける．また，アクチュエータは動作開始し一定時間経過してからロック/アンロックを完了する．
- 構成要素間の同期のモデル化
  このシステムでは，センサ，コントローラ，アクチュエータの各要素間で情報が受け渡される．各要素が直接関与する情報は制約され，たとえばアクチュエータはセンサ値を直接参照することはできない．

5. モデル化の方針

5.1 車速のモデル化

物理量としての車速は実数であるが，一般に実数を含むモデルに関する証明は整数のみのモデルに比べて困難であるため，実数を素朴にモデルに取り込む前に検討が必要である[*1]．

このケーススタディで取り上げる問題では，ドアロック下限速度までにドアが確実にロックされていることの検証に関心がある．ここで，コントローラはセンサ値に基づいてドアロックの要否を判断するので，車速の連続的な変化はシステムの動作に影響しない．また，システムの動作は車速に影響しないので，加速度を用いて車速の連続的な変化を計算する必要もない．このため，連続量に代えて離散量である整数で車速を近似したモデルでも十分であると考えられる．ただし，システムを構成する定数の間の関係を反映できる分解能を持った近似でなければならないことに注意する．たとえば，*l_limit* が 1 となる近似であれば，動作分析の意味がない．本稿では各定数に具体値を与えず，モデル化と検証を通じて定数間の関係を明らかにしてゆく．

ドアがロックされることが保証される実車速 v は，*l_limit* を使って

$$l\_limit + em \leq v$$

と表すことができる．ここで，em はセンサが車速を整数化することにともなう誤差の最大値である．

5.2 時間とともに変化する事象のモデル化

このケーススタディで取り上げる問題では，車速は外部環境によって決定される値であり，システムの動作とは独立に，物理時間の進行とともに変化する．一方，物理時間はシステム側では計測されず，システム内で表現されることもない．システムの動作分析では，時間経過にともなうシステムの動作順序が規定できれば十分であり，連続的に進む物理時間をモデルに取り込む必要はない．

そこで，センサが一定周期で車速を測定することに着目し，測定周期（測定から次の測定までの時間間隔）を時間の単位とすることで，時間による外部環境の変化とシステムの動作を関連付ける．これは，外部環境から周期的に情報を得るシステム一般について，外部環境の変化を表現するための手段として有効であると考える．

また，アクチュエータの動作時間が無視できない程度に長いことも特徴であり，動作分析のためには動作開始からの経過時間をモデル上で表現することが必要である．アクチュエータ動作中もセンサの車速測定が行われることと，測定周期がアクチュエータの動作時間に対して短いことから，センサの測定周期はアクチュエータ動作の経過時間の

[*1] Event-B/RODIN でも実数を扱うプラグインの開発が進められているが，Event-B 言語ではまだ実数はサポートされていない．

単位としても適切である．

次の定数を導入する．

- 1 センサ周期内に起こりうる車速センサ値の最大変化量（加減速ともに）を，定数 s_dv とする．
- アクチュエータの動作時間は一定であるとし，動作時間をセンサ周期で割った整数値を定数 a_delay とする．

定数 s_dv と a_delay は正の自然数である．測定された車速センサ値が s_v であるとき，次の周期で測定されるセンサ値 s_v' は

$$max(0, s\_v - s\_dv) \leq s\_v' \leq s\_v + s\_dv$$

であり，最大で s_dv だけ減少または増加する．

実車速はセンサ測定後も連続的に変化するため，センサ値が s_v であるときの実車速 v は，

$$max(0, s\_v - em - s\_dv) \leq v \leq s\_v + em + s\_dv$$

となる．

5.3 要素間の同期のモデル化

Event-B でモデル化するにあたって，センサ，コントローラ，アクチュエータを，1 つの抽象機械の異なるイベントとして表現した．ここで，センサ，コントローラ，アクチュエータの間の同期は，**図 4** のように，これらを順に繰り返し活性化することでモデル化する．このようなモデル化は，Event-B ではしばしば行われる [26]．活性化する要素は，次の 3 つの定数により指定する．

- sns：センサを活性化
- cnt：コントローラを活性化
- act：アクチュエータを活性化

イベント間の活性順序は，活性順序を格納する変数 $phase$ を用意して制御する．たとえばセンサのイベントでは，

$$phase = sns$$

をガード条件に加えることで，センサが活性化されたときにのみセンサのイベントが起きる．センサの次にコントローラを活性化するためには，センサのイベントのアクションで

$$phase := cnt$$

と $phase$ を書き換える．この代入文は Event-B の構文であり，$phase' = cnt$ を意味する．

センサ，コントローラ，アクチュエータの間の情報の伝達は，共通の変数を各イベントが参照，更新することで表現する．

- センサからコントローラ
  測定した車速を表現する変数 s_v
- コントローラからアクチュエータ
  ロック/アンロック指示を表現する変数 c_mode

ここで，c_mode は次の値をとる．

- c_unlock：アクチュエータにアンロックを指示
- c_lock：アクチュエータにロックを指示

6. モデル記述と分析

6.1 リファインメント戦略

このシステムは，車速をモニタし，判断を加え，アクチュエータを操作する．センサはシステムを流れるデータの起源でありコントローラとアクチュエータからの影響を受けないことから，センサからモデル化を開始し，図 4 の依存関係に従ってコントローラ，アクチュエータの順にモデル化する戦略をとった．

モデルの構成を，**図 5** に示す．モデル 1 から 3 の 3 つのモデルは，リファインメント関係にある．各モデルの位置付けは，以下である．

(1) モデル 1：センサをモデル化
センサ周期の間に起こりうる範囲で車速の測定値を変化させる．

(2) モデル 2：コントローラをモデル化
車速と乗員の操作に基づいてコントローラの指示を変化させる．

(3) モデル 3：アクチュエータをモデル化
コントローラの指示に基づいてアクチュエータの動作状態を変化させる．

各モデルは抽象機械とコンテクストから構成され，モデル 2 はモデル 1 の詳細化，モデル 3 はモデル 2 の詳細化である．ここで **EXTENDS** は，台集合，定数，公理の追加によるコンテクストの拡張を表す．各モデルの整合性は 3.3 節の証明課題を証明することで示せるので，もし不整

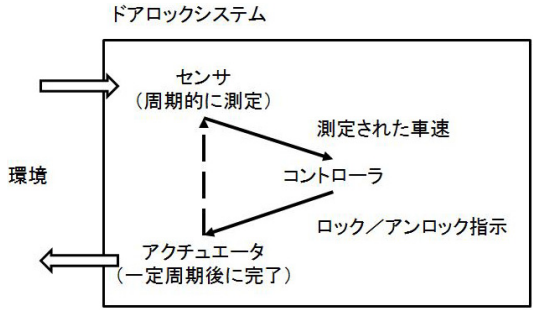

図 4 ドアロックシステム動作のモデル化（文献 [20] より）
Fig. 4 Door lock system model.

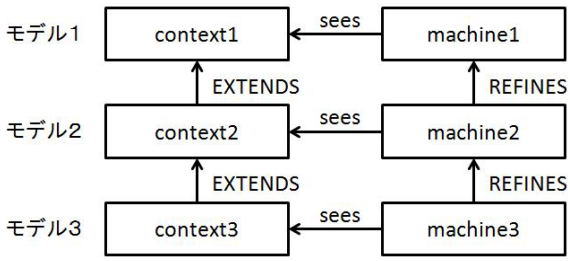

図 5 形式モデルの構成（文献 [20] より）
Fig. 5 Structure of formal model.

合があれば抽象機械あるいはコンテクストを修正する．また，モデル 2，3 は，それぞれモデル 1，2 から重ね合わせリファインメントで構成したモデルであり，抽象モデルで検証した性質は具象モデルに引き継がれる．このことは，3.3 節のリファインメントに関する証明課題を証明することで示せる．

6.2 モデル 1

モデル 1 では，センサが測定する車速と，その変化をモデル化する．車速の変化はシステム外部の要因によって起こるので，許容される範囲内の変化値が非決定的に選ばれるものとして車速の変化を表現した．車速は，変数 s_v に記録する．この変数は，コントローラが参照する．また，モデル 1 でセンサ，コントローラ，アクチュエータを順に活性化する．ただし，この段階ではコントローラとアクチュエータのイベントはダミーであり，後のリファインメントによって順次機能を加える．

センサのイベントを，次に示す．このイベントは，$phase$ が sns のときに起こり得て，車速 s_v を dv だけ変化させて，$phase$ を cnt に変える．ここで，車速の変化 dv は，$-s\_dv$（減速）から s_dv（加速）の間の値である．ただし，車速が負にならないよう，dv は $-s\_v$ 以上であるとする．これらの条件を満たす dv の値はつねに存在するので，$phase$ が sns であればイベント sensor のガード条件は必ず満たされる．

sensor $\triangleq$
any dv
where $phase = sns \wedge$
$dv \in \mathbb{Z} \ \wedge \ dv \leq s\_dv \ \wedge \ -s\_dv \leq dv \ \wedge \ -s\_v \leq dv$
then
$s\_v := s\_v + dv$
$phase := cnt$
end

一方，モデル 1 のコントローラとアクチュエータのイベントは，以下である．コントローラのイベントは，$phase$ が cnt のときに起こることができ，$phase$ を act に変える．アクチュエータのイベントは，$phase$ が act のときに起こることができ，$phase$ を sns に変える．このようにして，センサ，コントローラ，アクチュエータのイベントが循環して繰り返される．

controller $\triangleq$
where $phase = cnt$
then $phase := act$
end
actuator $\triangleq$
where $phase = act$
then $phase := sns$
end

6.3 モデル 2

モデル 2 は，コントローラをモデル化する．コントローラはアクチュエータにドアロック/アンロックを指示するが，モデル上ではこれを変数 c_mode の値で表現する．コントローラのイベントは，次のものである．

- ロック指示維持：c_mode が c_lock のとき c_lock を維持する．
- アンロック指示維持：c_mode が c_unlock かつ s_v が c_high より小さいとき c_unlock を維持する．
- 自動ロック：s_v が c_high 以上で c_mode が c_unlock のとき，c_mode を c_lock に変える．
- 乗員操作によるロック：c_mode が c_unlock のとき，c_mode を c_lock に変える．
- 乗員操作によるアンロック：s_v が c_low 以下で c_mode が c_lock のとき，c_mode を c_unlock に変える．

これらにより，コントローラは s_v が c_high 以上であるとき，次のように振る舞う．

- まだドアロックを指示していなければ，イベント「自動ロック」または「乗員操作によるロック」が起こってドアロックを指示．
- ドアロックを指示していれば，イベント「ロック指示維持」が起こってドアロック指示を維持．

6.4 モデル 2 の分析

モデル 2 では，センサ値が l_limit 以上であればドアロック指示されていることを表す，次の式を証明したい．

$$l\_limit \leq s\_v \Rightarrow c\_mode = c\_lock \tag{1}$$

しかし，システム動作中も車速は変化していることから，ドアロックが指示されていることを保証できる条件は $c\_high \leq s\_v$ ではない．実際，$phase$ の値で分割すると，次の 2 つの不変条件が満たされることは検証できる．

$$phase = sns \wedge c\_high \leq s\_v \Rightarrow c\_mode = c\_lock \tag{1.1a}$$

$$phase = act \wedge c\_high \leq s\_v \Rightarrow c\_mode = c\_lock \tag{1.1b}$$

一方，$phase = cnt$ はセンサ測定が終わってコントローラが起動される前を表すので，$c\_mode = c\_lock$ であることを保証できるのは，$c\_high \leq s\_v$ となった次の周期で測定されるセンサ値に対してである．この値は，自動車が最大加速していても $c\_high + s\_dv$ 以下であり，次の不変条件が満たされることが検証できる．

$$phase = cnt \wedge c\_high + s\_dv \leq s\_v \Rightarrow c\_mode = c\_lock \tag{1.1c}$$

$phase$ がとりうる値は sns，cnt，act のみなので，式 (1.1a)，

(1.1b)，(1.1c) から，センサ値が $c\_high + s\_dv$ 以上のときには必ずドアロックを指示していることを，定理として証明できる．

$$c\_high + s\_dv \leq s\_v \Rightarrow c\_mode = c\_lock \tag{1.1}$$

したがって，l_limit と c_high が次の関係を満たすように設定してあれば，式 (1) が証明できる．

$$c\_high + s\_dv \leq l\_limit$$

これは定数間の条件なので，コンテクストに公理として記述すればよい．

6.5 モデル 3

モデル 3 は，アクチュエータをモデル化する．アクチュエータは，コントローラの指示によりドアロック/アンロック動作を行う．アクチュエータの時間依存性，すなわちロック動作開始からロック完了までの遅延は，アクチュエータのイベントがカウントする．アクチュエータの動作状態は，次の 4 つとする．

- $a\_unlocked$：ドアのアンロックが完了した状態
- $a\_unlocking$：アンロック動作中の状態
- a_locked：ドアのロックが完了した状態
- $a\_locking$：ロック動作中の状態

モデル上は，変数 a_mode がこれらの値のいずれかをとることで，アクチュエータの動作状態を表現する．また，アクチュエータの動作時間を表現するため，次の変数を用いる．

- a_dlock：アクチュエータがドアロック/アンロック動作を開始してからの経過時間（センサ周期数）

アクチュエータの機能を表すイベントは，次のものである．

- ロック完了状態維持：c_mode が c_lock で a_mode が a_locked のとき，a_locked を維持する．
- アンロック完了状態維持：c_mode が c_unlock で a_mode が $a\_unlocked$ のとき，$a\_unlocked$ を維持する．
- ロック動作の開始：c_mode が c_lock で a_mode が $a\_unlocking$ または $a\_unlocked$ のとき，a_mode を $a\_locking$ にして a_dlock を 1 にする．
- ロック動作を継続：c_mode が c_lock で a_mode が $a\_locking$ かつ a_dlock が a_delay より小さいとき，$a\_locking$ を維持して a_dlock を 1 増やす．
- ロック完了：c_mode が c_lock かつ a_mode が $a\_locking$ で a_dlock が a_delay と等しいとき，a_mode を a_locked にする．
- アンロック動作の開始：c_mode が c_unlock で a_mode が $a\_locking$ または a_locked のとき，a_mode を $a\_unlocking$ にして a_dlock を 1 にする．
- アンロック動作を継続：c_mode が c_unlock で a_mode が $a\_unlocking$ かつ a_dlock が a_delay より小さいとき，a_dlock を 1 増やす．
- アンロック完了：c_mode が c_unlock かつ a_mode が $a\_unlocking$ で a_dlock が a_delay と等しいとき，a_mode を $a\_unlocked$ にする．

6.6 モデル 3 の分析

モデル 3 では，センサ値が l_limit 以上であればドアロックが完了していることを表す，次の式を証明したい．

$$l\_limit \leq s\_v \Rightarrow a\_mode = a\_locked \tag{2}$$

まず，モデル 2 のコントローラの振舞いから，センサ値が $c\_high + s\_dv$ 以上であれば，アクチュエータの状態はロック動作中あるいはロック完了であることが期待される．これは，$phase$ の値で分割した不変条件が満たされることを検証した後，次の定理を証明することで示せる．

$$c\_high + s\_dv \leq s\_v \Rightarrow (a\_mode = a\_locking \vee a\_mode = a\_locked) \tag{2.1}$$

アクチュエータがドアロック動作を開始してからの経過時間はセンサ周期 a_dlock 回であり，この間にセンサ値は最大で $a\_dlock \times s\_dv$ 増加するので，下記の 3 つの不変条件が満たされることが検証できる．

$$phase = sns \wedge a\_mode = a\_locking \Rightarrow s\_v < c\_high + a\_dlock \times s\_dv \tag{2.2a}$$

$$phase = cnt \wedge a\_mode = a\_locking \Rightarrow s\_v < c\_high + a\_dlock \times s\_dv + s\_dv \tag{2.2b}$$

$$phase = act \wedge a\_mode = a\_locking \Rightarrow s\_v < c\_high + a\_dlock \times s\_dv + s\_dv \tag{2.2c}$$

$phase$ が sns の段階ではまだ車速が測定されていないため，測定後のセンサ値を参照する cnt と act との間に，s_dv の差が生じる．

$a\_dlock \leq a\_delay$ がつねに成り立つことは，不変条件として検証できる．簡単のため a_high を

$$a\_high = c\_high + a\_delay \times s\_dv + s\_dv$$

とおくと，式 (2.2a)，(2.2b)，(2.2c) から次の定理が証明できる．

$$a\_mode = a\_locking \Rightarrow s\_v < a\_high \tag{2.2}$$

センサ値が a_high 以上のときはアクチュエータがロック動作を完了し，ドアがロックされた状態であることがいえるはずである．このことは式 (2.1) と (2.2) から次の定理を証明することで確認できる．

$$a\_high \leq s\_v \Rightarrow a\_mode = a\_locked \quad (2.3)$$

したがって，l_limit と a_high が次の関係を満たすようコンテクストに設定してあれば，式 (2) が証明できる．

$$a\_high \leq l\_limit$$

これはモデル 2 の条件を厳しくしたものであり，モデル 2 とは矛盾しない．

7. 結果

このケーススタディでは，まずイベントが想定した順序で起こることを仕様アニメーションツールを使って確認した後，定理証明支援ツールを使ってモデルが不変条件を満足することを証明する方法で，モデルの記述と動作分析を進めた．Event-B のモデリングスタイルでは，ガード条件が満たされたイベントが非決定的に起こるため，モデルに表現された振舞いが記述者の意図どおりであることを，直感的に理解し難い．また，一般に証明には人手の関与が必要であり，定理証明支援ツールを使った検証には手間がかかる．そのため，ProB [18] を使った仕様アニメーションによってモデルの振舞いを確認し，誤りを早い段階で除くことは，記述と分析の作業効率向上において効果的であった．

一方，定数の値を限定しても車速の増減パターンは無数にあり，それらすべてについて ProB による網羅的な検証を行うことはできない．モデル上で確認したいのは，公理を満たす定数値に対して，車速がどのように変化してもドアロック下限速度以上であればドアが確実にロックされていることである．この検証を行うには，定理証明法を使った検証が必要である．

定理証明法による検証は，次の手順で行う．

(1) RODIN を使って 3.3 節の証明課題を自動生成
(2) 定理証明支援ツールを使って証明課題を自動証明
(3) 自動証明できない証明課題がある場合，定理証明支援ツールを使って対話証明

モデルの記述量と RODIN が生成した証明課題数，定理証明支援ツールによる自動証明率を**表 1** に示す．表中のモデル 2 とモデル 3 の行数は各モデルで付加した行数であり，それぞれリファインメント前のモデルであるモデル 1，モデル 2 からの差分である．モデル 1，モデル 2，モデル 3 と機能が複雑になりイベントが細分化されるに従って，モデルの行数と証明課題の数が増加することが見て取れる．

表 1 モデル記述量と証明課題数
Table 1 Model size and number of proof obligations.

	行数	証明課題数	対話証明数	自動証明率
モデル 1	31	2	0	100%
モデル 2	52	27	5	81%
モデル 3	92	104	9	91%
合計	175	133	14	89%

6.4，6.6 節で示したように，センサ，コントローラ，アクチュエータの 3 種のイベントについて不変条件を設定し，それらからモデル全体について満たされる安全性要件を定理として証明するため，不変条件が細分化されて証明課題数が増加している．

一般に，物理量としての車速と車速の変化を表すためには，車速と時間は実数でなければならない．さらに，車速の変化は連続的に起こるので，離散的なイベントでは表せない．一方，このケーススタディで分析対象となるのは，センサが周期的に測定した車速値に対する振舞いであり，測定はイベントで表すことができる．このことに着目し，モデルではセンサ周期を時間の単位として，単位時間内に起こりうる車速変化を離散量で近似した．環境を周期的にモニタするモニタ–コントロールシステムのモデル化では，環境の物理量とシステムが処理するデータの境界において，このような近似が有効であると考える．

また，アクチュエータの時間依存性，すなわちロック動作の開始から完了までの時間的遅延，をモデル化するため，トリガイベントからレスポンスイベントまでの間で発生するイベントによって，時間経過をカウントした．このケーススタディでは，アクチュエータの動作時間がセンサ周期に対して長く，センサ/コントローラ/アクチュエータの同期のため 1 センサ周期内に必ずアクチュエータイベントが発生することから，経過時間のカウントにはアクチュエータイベントを用いた．このことによる Tick_Tock イベントと共通時間の除去は，モデルを単純化し検証を円滑に進めるうえで有効であった．比較を，8 章と付録 A.1 に示す．このケーススタディには表れないが，トリガイベントとレスポンスイベントの組が複数ある場合，共通するイベントがあればそれらを使って時間経過をカウントすることで，時間を共有することができる．共通するイベントによる時間経過のカウントは，並行動作する実体間で時間を共有する場合にも有効であると考えられる．

モデル上で証明できた安全性要件の間の関係を，**図 6** に示す．これは机上の分析結果とも一致し，作成したモデルは直感的にも妥当であると判断できる．一方，これらの関係が仕様アニメーションにより確認できるだけでなく，車速のあらゆる変化について成り立つことが検証できる点に，

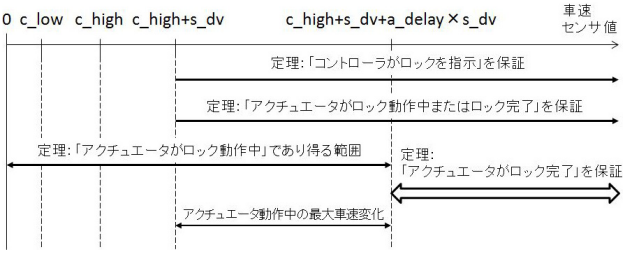

図 6 形式モデル上で分析したシステムの振舞い
Fig. 6 System behavior analized with formal model.

モデルを使った分析の利点がある．このことから，1 章で論じたケーススタディの目的は達成したと考える．

本稿では，検証可能なモデルを構成するために，5 章に述べた近似を行っている．これらは，このケーススタディにおいて選択した方針であるが，モニタ–コントロールシステムの検証可能モデル作成のための方法論として，一般化できる可能性がある．実際，付録 A.1 に示した Tick_Tock イベントを使うモデルも，5 章のモデル化方針のもとで構成し検証することができた．さらに，独立性の高いモデルを構成する一般的な方法が存在すれば，そのもとにモデルをモジュール化し，他のモデルを構成する際に再利用できる可能性がある．このケーススタディで得られた知見に基づく，これらの可能性についての検討は，今後の課題である．

8. 関連研究

Event-B を使った時間依存性のモデル化では，Sarshogh ら [24] があげられる．この論文では，期限，遅延，終了についてモデル化とリファインメントの方法が示され，これらを Event-B 記述に導入するための一般性のある手順が与えらえている．時間は共通に参照される整数値であり，Tick_Tock イベントによって一様に進行する．また，Cansell ら [8] は，Event-B における時間依存性のパターン化の問題を扱っている．時間は共通に参照される整数値であり，Tick_Tock イベントによって進むが，その進行は一様ではない．すなわち，時間は Tick_Tock イベントのたびに 1 増加するのではなく，時間制約に関わる時刻以下の値へ非決定的に増加する．Sarshogh らと Cansell らはともに時間依存性のモデル化方法を提案するが，時間依存性から導かれる安全性要件の検証については示していない．

5 章のモデル化方針のもとで Sarshogh らの時間依存性モデル化方法を適用したモデルと，安全性要件の検証過程を，付録 A.1 に示す．**表 2** は，このモデルの記述量と証明課題数，自動証明率である．イベント間の関係を共通時間からの相対値を介して解析するため，不変条件と定理が増加して，証明課題数が表 1 の時間依存性モデル化方法の 1.8 倍になった．対話証明は複雑になり，たとえば定理 (1.1)，(2.1)，(2.2) の証明では，Tick_Tock イベントを含む 4 つの *phase* についての場合分けが必要であった．このことから，同一の前提のもとで安全性要件を検証するうえでは，本稿の時間依存性モデル化方法が優位であるといえる．

表 2 Tick_Tock イベントを使ったモデル記述量と証明課題数
Table 2 Model size and number of proof obligations of Tick_Tock model.

	行数	証明課題数	対話証明数	自動証明率
モデル 1	39	4	0	100%
モデル 2	58	39	5	87%
モデル 3	99	202	9	96%
合計	196	245	14	94%

Zhang ら [28] は，Lamport の実時間モデリングの方法に基づき，時間をカプセル化する実時間モジュールを TLA+ に導入した．このモジュールを利用することで，TLA+を使って記述したモデルの中で実時間を扱うことができる．時間に関する性質は TLC モデルチェッカを使って解析できるが，時間は有限の離散時間モデルで置き換えられ，実数ではない．また，モデル検査なので，探索空間が爆発的に拡大することを避けるためには解析できるモデルが制限されるうえ，解析結果は与えた初期値のもとでの振舞いに限定される．本研究でも時間は離散値で表現するが，定理証明支援ツールを使って不変条件の充足性を証明することで，個々の初期値に依存しない網羅的な解析を行う．

Butler [7] は，制御システムを Event-B でモデル化するためのガイドラインの提供を試みている．この中では，モデル化は，コントローラ，センサ，アクチュエータの順に進行する．制御システムの役割はプラントを一定の状態に保つことにあり，はじめに制御対象とコントローラを一体にしたモデルを構成した後，センサとアクチュエータを導入して制御対象の物理現象とコントローラの内部表現を分離する．これに対し，Hudon ら [11] は環境のモデル化からはじめ，環境に作用するアクチュエータのモデル化，センサとコントローラのモデル化，効率的な制御のためのスケジューラのモデル化，の順で進める．環境とアクチュエータを先にモデル化することで制御システムの要件が明確になり，アクチュエータが必要とする情報と制御を提供するためにセンサとコントローラを導入するので，モデル化が容易になる．これらの研究では，制御システムは環境をモニタし，それを一定の状態に保つために環境に働きかける．システムは基本的にリアクティブシステムであり，環境からの刺激に対して時間依存性なく反応する．したがって，はじめに望ましい環境をモデル化することは，有効なモデル化方針である．これに対して本稿でとりあげたドアロックシステムは，アクチュエータの動作時間が無視できないことから，アクチュエータの動作状態が時間の影響下にある．すなわち，環境からの刺激を受けてアクチュエータがただちに動作を完了するのではなく，動作完了までの間に環境は変化し続ける．このような状況では，システムのリアクティブ性を前提とする上記関連研究のモデル化方針は有効に機能しない．時間依存性のもとでの安全性要件の検証に対応するため，環境との接点であるセンサをはじめにモデル化した点が，本稿のケーススタディの特徴である．

9. おわりに

自動車ドアロックシステムを題材とし，時間経過にともなう環境の変化とシステムの反応のモデル化と，動作分析のケーススタディを行った．モデリング手法 Event-B を使い，次の手順でモデル化と動作分析を進めた．

- 環境の変化とシステムの反応をイベントとして記述
- システムが満たすべき安全性要件を不変条件で表現し検証

ここで，時間依存性を持つイベント間の時間間隔を，両者の間で発生するイベントによってカウントした．このことは，モデルの単純化と検証の容易化に効果があった．

図 6 に示したように，分析の結果は妥当であると考えられる．一方，車速変化のあらゆる組合せに対して，この要件が満たされることがモデル上でいえる点で，Event-B を使ったモデル化と動作分析は有用であると結論できる．

今後の課題は，検証可能なモデルを系統的に構成するための方法論の開発である．これには，リファインメントの順序を立案するための方法と，モデルのモジュール化と再利用のための方法を含む．

参考文献

[1] Abrial, J.-R.: *The B-Book*, Cambridge University Press (1996).
[2] Abrial, J.-R: *Modeling in Event-B*, Cambridge University Press (2010).
[3] Abrial, J.-R.: Formal Methods in Industry, *Proc. ICSE 2006*, pp.761–767 (2006).
[4] Abrial, J.-R., Su, W. and Zhu, H.: Formalizing Hybrid Systems with Event-B, *Proc. ABZ 2012* (2012).
[5] Badeau, F. and Amelot, A.: Using B as a High Level Programming Language in an Industrial Project: Roissy VAL, *ZB 2005*, Treharne, H. et al. (Eds.), LNCS 3455, pp.334–354, Springer-Verlag (2005).
[6] Behm, P., Benoit, P., Faivre, A. and Meynadier, J.-M.: Météor: A Successful Application of B in a Large Project, *FM '99*, Wing, J. Woodcock, J. and Davies, J. (Eds.), Vol.1, LNCS 1708, pp.369–387, Springer-Verlag (1999).
[7] Butler, M.: Towards a Cookbook for Modelling and Refinement of Control Problems, working paper 108, University of Southampton (2009).
[8] Cansell, D., Mery, D. and Rehm, J.: Time Constraint Patterns for Event B Development, *B 2007*, LNCS 4355, pp.140–154, Springer-Verlag (2007).
[9] Furia, C.A., Mandrioli, D., Morzenti, A. and Rossi, M.: Modeling Time in Computing: A Taxonomy and a Comparative Survey, *ACM Computing Surveys*, Vol.42, No.2, Article 6 (2010).
[10] Hoang, T.-S., Kuruma, H., Basin, D. and Abrial, J.-R.: Developing Topology Discovery in Event-B, *Science of Computer Programming*, Vol.74, No.11–12, pp.879–899 (2009).
[11] Hudon, S. and Hoang, T.-S.: Development of Control Systems Guided by Models of their Environment, *Proc. B 2011 Workshop, Electronic Notes in Theoretical Computer Science*, Vol.280, pp.57–68 (2011).
[12] Jackson, M. and Zave, P.: Deriving Specifications from Requirements: An Example, *Proc. ICSE 1995*, pp.15–24 (1995).
[13] 来間啓伸：B メソッドと支援ツール，コンピュータソフトウェア，Vol.27, No.2, pp.8–13 (2007).
[14] 来間啓伸：B メソッドによる形式仕様記述，近代科学社 (2007).
[15] 來間啓伸，石川冬樹：形式仕様に基づくソフトウェア開発手法の紹介，コンピュータソフトウェア，Vol.29, No.4, pp.50–58 (2012)
[16] 來間啓伸，中島 震：Event-B：リファインメントに基づくシステム・モデリング，コンピュータソフトウェア，Vol.31, No.1, pp.43–48 (2014).
[17] Lamport, L.: *Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers*, Addison-Wesley (2002).
[18] Leuschel, M. and Bulter, M.: ProB: An automated analysis toolset for tbe B method, *International Journal on Software Technology Transfer*, Vol.10, No.2, pp.185–203 (2009).
[19] 中島 震：形式手法入門，オーム社 (2012).
[20] 中島 震，來間啓伸：Event-B：リファインメントに基づく形式手法，近代科学社 (2015).
[21] Nuseibeh, B. and Zave, P. (Eds.): *Software Requirements and Design: The Work of Michael Jackson*, Lulu.com (2010).
[22] Plagge, D. and Leuschel, M.: Seven at one stroke: LTL model checking for high-level specifications in B, Z, CSP, and more, *International Journal on Software Tools for Technology Transfer*, Vol.12, No.1, pp.9–21 (2010).
[23] Romanovsky, A. and Thomas, M. (Eds.): *Industrial Deployment of System Engineering Methods*, Springer (2013).
[24] Sarshogh, M.R. and Butler, M.: Specification and refinement of discrete timing properties in Event-B, *Proc. AVoCS 2011* (2011).
[25] Sommerville, I.: *Software Engineering, 6th Edition*, Addison-Wesley (2001).
[26] Su, W., Abrial, J.-R. and Zhu, H.: Complementary Methodologies for Developing Hybrid Systems with Event-B, *ICFEM 2012*, Aoki, T. and Taguchi, K. (Eds.), LNCS 7635, pp.230–248, Springer-Verlag (2012).
[27] Woodcock, J., Larsen, P.G., Bicarregui, J. and Fitzgerald, J.: Formal Methods: Practice and Experience, *ACM Computing Surveys*, Vol.41, No.4, Article 19 (2009).
[28] Zhang, H., Gu, M. and Song, X.: Specifying time-sensitive systems with TLA+, *Proc. COMPSAC 2010*, pp.425–430 (2010).
[29] Event-B.org, available from ⟨http://www.event-b.org/⟩.

付　　録

A.1 Tick_Tock イベントを使って記述したモデルの概要

Tick_Tock イベントによって進行する共通時間を使った時間依存性モデル化方法に基づいて作成した，ドアロックシステムのモデルの概要を示す．Sarshogh ら [24] の時間依存性モデル化方法をドアロックシステムのモデル化に適用するにあたり，5 章のモデル化方針を採用した．

モデル化方針が共通であるため 6 章のモデルとの差異はわずかであるが，時間依存性モデル化方法の違いにより，表 2 に示したように証明課題数は 1.8 倍になった．

A.1.1 モデル 1

システム動作の経過時間をカウントするためには，5.3 節の要素間の同期のたびに，少なくとも 1 回は Tick_Tock イ

ベントを起こして時間を進める必要がある．そこで *phase* の値となる 5 章の定数に *tck* を加え，アクチュエータイベントの後に Tick_Tock イベントが活性化されて共通時間 *time* が増えるよう，6.2 節のモデル 1 を変更した．これは，センサ周期数を数えるのと同等である．

actuator $\triangleq$
where $phase = act$
then $phase := tck$
end
tick_tock $\triangleq$
where $phase = tck$
then
$time := time + 1$
$phase := sns$
end

A.1.2 モデル 2

モデル 2 は，モデル 1 から引き継いだ変更部分のほかは，6.3 節のモデル 2 と同じである．*phase* に *tck* が加わったので，モデル 2 の分析では，次の不変条件が満たされることを検証する必要がある．

$$phase = tck \wedge c\_high \leq s\_v \Rightarrow c\_mode = c\_lock \quad (1.1d)$$

6.4 節の定理 (1.1) は (1.1a)–(1.1d) から証明することができ，この結果 (1) が証明できる．

A.1.3 モデル 3

共通時間を利用するため，アクチュエータがロック/アンロック動作を開始した時間を記録する変数 *start_at* を用意して，6.5 節のアクチュエータ動作時間を表現する変数 *a_dlock* を，$time - start\_at$ で置き換えた．これにともない，アクチュエータの機能を表すイベントのうち時間経過に関わるものは，次のように変更した．

- ロック動作の開始：*c_mode* が *c_lock* で *a_mode* が *a_unlocking* または *a_unlocked* のとき，*a_mode* を *a_locking* にして *time* を *start_at* に代入する．
- ロック動作を継続：*c_mode* が *c_lock* で *a_mode* が *a_locking* かつ *time* が $start\_at + a\_delay$ より小さいとき，*a_locking* を維持する．
- ロック完了：*c_mode* が *c_lock* かつ *a_mode* が *a_locking* で *time* が $start\_at + a\_delay$ と等しいとき，*a_mode* を *a_locked* にする．
- アンロック動作の開始：*c_mode* が *c_unlock* で *a_mode* が *a_locking* または *a_locked* のとき，*a_mode* を *a_unlocking* にして *time* を *start_at* に代入する．
- アンロック動作を継続：*c_mode* が *c_unlock* で *a_mode* が *a_unlocking* かつ *time* が $start\_at + a\_delay$ より小さいとき，*a_unlocking* を維持する．
- アンロック完了：*c_mode* が *c_unlock* かつ *a_mode* が *a_unlocking* で *time* が $start\_at + a\_delay$ と等しいとき，*a_mode* を *a_unlocked* にする．

tck に関わる次の不変条件を追加して検証することで，6.6 節の定理 (2.1) が証明できる．

$$phase = tck \wedge c\_high \leq s\_v \Rightarrow (a\_mode = a\_locking \vee a\_mode = a\_locked) \quad (2.1d)$$

一方，時間経過と車速の関係を表す不変条件は下記であり，モデル 3 で満たされていることが検証できる．

$$phase = sns \wedge a\_mode = a\_locking \Rightarrow s\_v < c\_high + (time - start\_at) \times s\_dv \quad (2.21a)$$

$$phase = cnt \wedge a\_mode = a\_locking \Rightarrow s\_v < c\_high + (time - start\_at) \times s\_dv + s\_dv \quad (2.21b)$$

$$phase = act \wedge a\_mode = a\_locking \Rightarrow s\_v < c\_high + (time - start\_at) \times s\_dv + s\_dv \quad (2.21c)$$

$$phase = tck \wedge a\_mode = a\_locking \Rightarrow s\_v < c\_high + (time - start\_at) \times s\_dv + s\_dv \quad (2.21d)$$

これらは 6.6 節の不変条件 (2.2a)，(2.2b)，(2.2c) に相当するが，経過時間を局所的にカウントする変数 *a_dlock* について不変条件 $a\_dlock \leq a\_delay$ を検証するのが容易であるのに対し，一様に増加する *time* の場合はつねに $(time - start\_at) \leq a\_delay$ が成立するわけではない．アクチュエータ動作時間の上限を規定するためには，さらに次の不変条件が満たされることを検証する必要がある．

$$phase = sns \wedge a\_mode = a\_locking \Rightarrow (time - start\_at) \leq a\_delay \quad (2.22a)$$

$$phase = cnt \wedge a\_mode = a\_locking \Rightarrow (time - start\_at) \leq a\_delay \quad (2.22b)$$

$$phase = act \wedge a\_mode = a\_locking \Rightarrow (time - start\_at) \leq a\_delay \quad (2.22c)$$

$$phase = tck \wedge a\_mode = a\_locking \Rightarrow (time - start\_at) < a\_delay \quad (2.22d)$$

(2.21a)–(2.21d) から，次の定理が証明できる．

$$a\_mode = a\_locking \Rightarrow s\_v < c\_high + (time - start\_at) \times s\_dv + s\_dv \quad (2.21)$$

また，(2.22a)–(2.22d) からは，次の定理が証明できる．

$$a\_mode = a\_locking \Rightarrow (time - start\_at) \leq a\_delay \quad (2.22)$$

6.6 節の定理 (2.2) は (2.21) と (2.22) から証明でき，これと定理 (2.1) から定理 (2.3) が，さらに定理 (2) が証明できる．

來間 啓伸 （正会員）

1981 年広島大学理学部卒業．1983 年同大学大学院理学研究科博士課程前期修了．1984 年株式会社日立製作所入社，主としてソフトウェア工学と形式手法の研究に従事．2006 年総合研究大学院大学複合科学研究科情報学専攻博士課程修了．博士（学術）．日本ソフトウェア科学会，IEEE，ACM 各会員．

中島 震 （正会員）

1979 年東京大学理学部物理学科卒業．1981 年同大学大学院理学系研究科修士課程修了．2004 年情報・システム研究機構国立情報学研究所教授，現在に至る．学術博士 (2000 年東京大学)．情報処理学会 2001・2015 年度山下記念研究賞，日本ソフトウェア科学会第 8 回論文賞受賞．ソフトウェア工学，形式手法，自動検証，サイバーフィジカルシステムに関する研究に従事．日本ソフトウェア科学会，ACM 各会員．