Vol.2014-SE-186 No.13
2014/11/14

Event-B を用いたリアクティブシステムのモデリングケーススタディ

來間啓伸[†1]　　中島震[†2]

組み込みシステムの設計検証では，時間経過にともなう外部環境の変化と，それに対するシステムの反応をモデル化することが大切である．本稿では，架空の自動車ドアロックシステムの設計を題材とした，モデル化と検証のケーススタディを報告する．仕様アニメーションツールを使ったモデル実行による確認と定理証明支援ツールを使った検証を併用し，自動車がどのように加速／減速しても一定速度以上であればドアがロックされることを，形式手法 Event-B を用いて検証することができた．

Modeling Case-study of Reactive System in Event-B

Hironobu Kuruma[†1] and Shin Nakajima[†2]

To verify embedded system design, both modeling of environment which changes depending on time and modeling of system response to the change of environment are important. We present in this paper a case-study of modeling and verification of a simplified door lock system design of automobile. We used model checking method for verification by model animation and theorem proving method for verification by proof. As the result we verified that doors are locked by the system when the velocity exceeds the limit independent of the order of acceleration and deceleration.

1. はじめに

形式手法は 1970 年代に研究開発が開始され，その後も手法の発展と適用先の拡大が図られてきた．現在では様々な形式手法が提案されるとともに，記述された形式仕様の整合性を機械的に検証するツールの開発も進められ，ソフトウェア開発への実適用が報告されている [3]．

ソフトウェア開発で形式手法を使う主な目的として，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挙げられる．しかし近年では，より上流工程であるシステム分析への適用の試みもなされている [2]．これは，システムとそれが動作する環境を記述するとともに，整合性の検証を通じてそれらの基本性質を明確化し，その後にソフトウェアの機能仕様を抽出するものである．一般に機能仕様を正確に記述することは容易ではないが，システムと環境の記述から機能仕様を抽出することで，考慮漏れのない機能仕様を得ることが期待できる．

本稿では，Event-B を使ったリアクティブシステムのモデル化と検証のケーススタディを報告する．リアクティブシステムでは，システムは環境をモニタし，状況に応じて環境に働きかける．ケーススタディの対象として，自動車のドアロックシステムを取り上げる．ドアロックシステムは車速をモニタし，車速が規定値以上であればドアをロックする．この時，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，その間も車速は変化し続けるものとする．本稿では，このようなシステムをモデル化し，いかなる場合でも車速がある値以上であればドアがロックされていることを検証する．

Event-B を使ったリアクティブシステムのモデル化と検証では，信号制御システムを対象とした研究 [6]などがある．これらの研究と比較すると，本稿のケーススタディの対象では外部環境の一部である車速そのものはシステムの制御対象ではなく，アクチュエータの動作中にも刻々と変化する．このため，自律的に変化する外部環境のモデル化をはじめに行う点に特徴がある．

2. 形式モデリング手法 Event-B

2.1 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 [2]であり，EU の RODIN プロジェクトおよび DEPLOY プロジェクト [10]で開発が進められてきた．パリ地下鉄などへの適用で知られている B メソッド [1]は，機能仕様の形式記述と検証に裏付けられたプログラム導出を目的としている．これに対して，Event-B は B メソッドを継承しつつ，より上流工程である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，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．イベントは自発的に起こるできごとであり，イベントが起こるための必要条件をガード条件によって与える．図 1 に示すように，Event-B の形式モデルは，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．

†1 株式会社日立製作所　横浜研究所
Yokohama Research Laboratory, Hitachi, Ltd.
†2 国立情報学研究所／総合研究大学院大学
National Institute of Informatics／SOKENDAI

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，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， Event-B の形式モデルのデータ構造を与える．一方，抽象機械では変数と不変条件およびイベントを記述する．

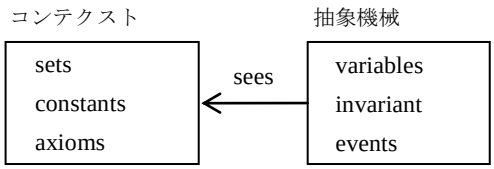

図１　Event-B 形式モデルの構成

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

any　x　where　G(x, v)　then　Q(x, v, v')　end

ここで，v は変数，イベントパラメタ x はイベント内で使われる一時変数，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，イベントによる変数値の変化を表す．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，前後条件と呼ばれる．

2.2 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．まず単純化した形式モデルを作成した後，リファインメントを通じて詳細を追加して目的とする形式モデルを作成する．これによって，形式モデルの記述と検証を段階的に行うことができ，モデル作成の労力が軽減される．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'が存在しなければならない．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のイベントである．糊付け不変条件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．

モデル記述の観点からは， Event-B のリファインメントではデータ詳細化に加えて重ね合わせ（superposition）もサポートされ，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可能である．

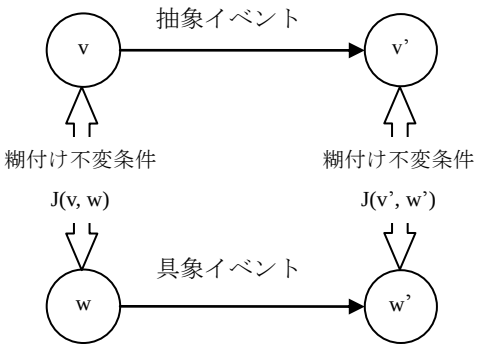

図２　リファインメント

2.3 モデルの整合性の検証

Event-B の形式モデルの整合性の検証では，不変条件が常に満たされることを検証する．検証手段として，定理証明による方法と，モデル検査による方法の２つが利用できる．

定理証明による方法では，Event-B のツールは，記述された形式モデルに対して不変条件が常に満たされることを検証するための証明課題を生成する．これは，次の３つである．

1. 変数の初期値が不変条件をみたすこと
   $Q(v') \Rightarrow I(v')$
2. ガード条件が満たされているイベントの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. アクションを実行した結果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，w はリファインされた抽象機械で導入された変数，H(x, v, w)と R(x, v, w, v', 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
   $I(v) \wedge J(v, w) \wedge H(x, v, w) \Rightarrow G(x, v)$
2. 抽象イベントのアクションが具象イベントのアクションを模倣すること
   $I(v) \wedge J(v, w) \wedge H(x, v, w) \wedge R(x, v, w, v', w') \Rightarrow Q(x\ v, v')$
3. イベント後の変数値の間に糊付け不変条件（gluing invariant）が成り立つこと
   $I(v) \wedge J(v, w) \wedge H(x, v, w) \wedge R(x, v, w, v', w') \Rightarrow J(v', w')$

証明課題が証明できた時，形式モデルは整合しているとみなせる．証明は定理証明支援ツールのサポートのもとで行うが，全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．

モデル検査による方法では，形式モデルを網羅的に実行して，全ての場合について不変条件が成り立つことを検証する．Event-B は一階の述語論理に基づくのでデータ構造が満たす性質を記述することができるが，モデル検査を行うためには実行可能となるように具体的なデータ値を与える必要がある．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具体化したインスタンスに関して実行した範囲内での検証に制限される．一方，実行によって，イベントが意図した順序で発生することを確認することができる．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，ProB[11]が知られている．ProB では，時相論理の式を与えて，イベントの発生順序に関する性質を検証することも可能である．

3. ケーススタディの題材

3.1 題材の概要

本稿では，モデリングの対象として，次の機能を持つ架空の自動車ドアロックシステムを考える．ここで，アンロック操作上限値はロック下限値より小さいとする．

- 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より高いとき
  乗員のアンロック操作を無視する．
- 車速がロック下限値以上のとき
  自動でドアをロックする．

簡単のため，進行方向の違いを無視して，車速を速度の絶対値で表す．これは 0 以上である．また，前進から後進および後進から前進する場合には，必ず一旦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ものとする．

1. センサは一定の時間間隔で周期的に車速を測定する．
2. 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．
3. アクチュエータはコントローラの指示にしたがってロック／アンロック動作を行う．

前提条件としてセンサの測定周期はコントローラの判断時間よりも長いとする．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いる．コントローラが判断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，次にセンサから車速値を受け取るときである．また，アクチュエータはセンサの測定周期よりも短い時間でコントローラの指示を受け取ることができるが，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ことがあり得る．

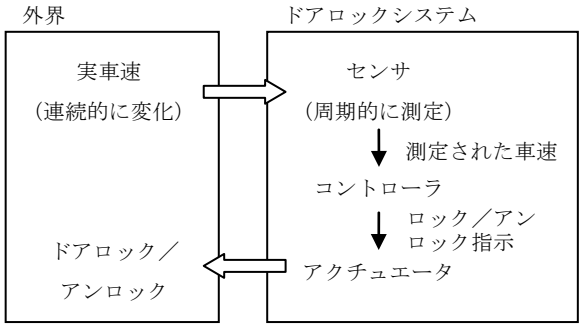

図３　ドアロックシステムの構成

3.2 モデル化の課題

上記の前提のもと，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検証する．モデル化と検証の目的は，次の 2 項目である．

1. 車が最大に加速したときでも，センサ値がロック下限速度までにロックが完了するよう，コントローラがロック指示を出す形式モデルを作成する．
2. 車速がどのように変化しても，センサ値が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を形式モデル上で検証する．

以下では，次の記号を使う．

- 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
- 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 の間にある時は，コントローラは現状維持を指示するものとする．

本稿では，「車速がどのように変化してもセンサ値がロック下限速度以上であればドアがロックされる」ことを，形式モデル上で確認したい．しかし，外界の要素である実車速は，システムの動作とは無関係に連続的に変化する．また，アクチュエータの動作時間が長く，モデル化にあたって無視することはできない．このような時間とともに変化する事象をどのように形式モデルで表現するかが，モデル化の要点となる．

また，上記のシステムでは，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一方，各要素が直接関与する情報は制約されている．例えばアクチュエータは車速センサ値を直接参照して動作するのではなく，センサ値からコントローラが判定したロック／アンロック

指示に基づいて動作する．また，コントローラの判定にはアクチュエータの動作状況は関与しない．このような要素間の情報の受け渡しを形式モデルに反映した上で，システムとして満たすべき性質を検証することがモデル化のもう一つの要点である．

4. モデル化

4.1 時間変化のモデル化

物理量としての時間や車速は実数であるが，Event-B は実数をサポートしていない．ただし，ここで取り上げる問題については，ロック上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ので，連続量に代えて離散量である整数による近似を使った粗いモデルでも十分であると考えられる．

また，システムが対象とするのはセンサの測定周期を単位とした車速の変化の変化とシステムの動作であり，物理的な時間をモデルに取り込む必要はない．そこで，センサの測定周期を時間の単位として，アクチュエータの動作時間をセンサ周期の整数倍で表現する．

このような考え方から，次の記号を導入する．

- １センサ周期内の車速変化の最大値（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．測定された車速が s_v である時，次の周期で測定される車速 s_v'は

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

であり，車速は最大で s_dv だけ減少または増加する．

なお，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 である時の実車速 v は，

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

となる．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．したがって，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

$$l\_limit + em + s\_dv \leqq v$$

と表される．

4.2 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，一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，これらの要素を順に繰り返し活性化することでモデル化する．

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

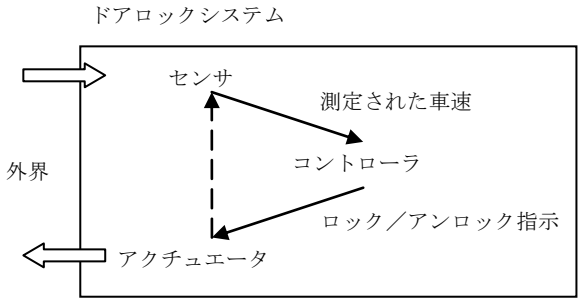

図４　ドアロックシステム動作のモデル化

イベント間の活性順序は，活性順序を格納する変数 phase を用意して制御している．例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時にのみセンサのイベントを起こす．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 を書き換えることとする[b].

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

c_mode は，次の２つの値を取る．

- 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

4.3 リファインメント戦略

リファインメントを利用して，センサ，コントローラ，アクチュエータの順に記述を追加することでモデルを構成した．図５に示すように，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の詳細化になっている．

1. モデル１：センサをモデル化
   センサ周期の間に起こり得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．

b 以下では，アクションの前後条件を代入文の形式で表す．この代入文は，phase' = cnt を意味する．

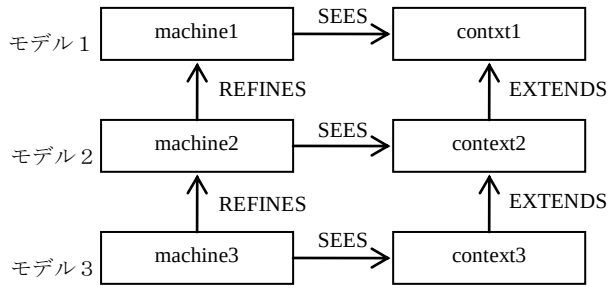

図５　形式モデルの構成

5. モデルの詳細

5.1 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 のときに起こることができ，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 の値は必ず存在するので，phase が sns であればイベント sensor は必ず起こる．

```
sensor =
any       dv
where
          phase = sns
          dv : INT
          dv ≦ s_dv
          - s_dv ≦ dv
          - s_v ≦ 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 =
where     phase = cnt
then      phase := act
end
actuator =
where     phase = act
then      phase := sns
end
```

5.2 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．コントローラの機能を表すイベントは，次のものである．

- controller_lockmode（ロック指示維持）: c_mode が c_lock のとき c_lock を維持する．
- controller_unlockmode（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．
- a_lock（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．
- m_lock（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．
- m_unlock（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，次のように振舞う．

- まだドアロックを指示していなければ，イベント a_lock または m_lock が起こってドアロックを指示する．
- ドアロックを指示していれば，イベント controller_lockmode が起こってドアロック指示を維持する．

自動車が最大加速しているとき，センサは一定周期で車速を計測しているので車速が c_high を越えた直後に計測が行われるとは限らないが，c_high + s_dv に達する前には必ず計測が行われる．したがって，コントローラがドアロックを指示するセンサ値は c_high から c_high + s_dv の間であり，車速が c_high + s_dv 以上の時には必ずドアロックを指示しているはずである．モデル２では，コントローラの指示が意図通りであることを，次の不変条件が成り立つことの証明によって検証できる．

$$c\_high + s\_dv \leqq s\_v \Rightarrow c\_mode = c\_lock$$

この不変条件により，最大加速時に限らず車速がどのように変化しても，上記の性質が成り立つことを示すことができる．このことから，次の関係を満たすように l_limit と

c_high を設定すれば，s_v が l_limit 以上である時にコントローラがドアロックを指示していることが証明できる．

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

5.3 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：アンロック動作を実行中の状態.
- a_locked：ドアのロックが完了した状態.
- a_locking：ロック動作を実行中の状態.

モデル上は，変数 a_mode がこれらの値のいずれかを取ることで，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２つの変数を用いる．

- a_dlock: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（センサ周期数）
- a_dunlock:アクチュエータがドア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．

- actuator_lockmode（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．
- actuator_unlockmode（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．
- start_locking（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．
- locking（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．
- locked（ロック完了）：c_mode が c_lock かつ a_mode が a_locking で a_dlock が a_delay のとき，a_mode を a_locked にする．
- start_unlocking（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_dunlock を 1 にする．
- unlocking(アンロック動作を継続)：c_mode が c_unlock で a_mode が a_unlocking かつ a_dunlock が a_delay より小さいとき，a_dunlock を 1 増やす．
- unlocked（アンロック完了）：c_mode が c_unlock かつ a_mode が a_unlocking で a_dunlock が a_delay のとき，a_mode を a_unlocked にする．

車速がどのように変化しても，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がロック動作中あるいはロック完了であることは，次の不変条件の証明によって示すことができる．

$$c\_high + s\_dv \leqq s\_v \Rightarrow$$
$$(a\_mode = a\_locking \text{ or } a\_mode = a\_locked)$$

アクチュエータは，ドアロックまたはアンロック動作を開始してから完了するまで，センサ周期で a_delay 回の時間がかかる．この間に車速は最大で a_delay * s_dv 変化するので，アクチュエータは最大で車速センサ値が

$$c\_high + s\_dv + a\_delay * s\_dv$$

になるまでドアロック動作を行う．いま，a_high を

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

とすると，車速がどのように変化しても a_high までにはロック動作が完了することは，次の不変条件が成り立つことの証明によって確認できる．

$$a\_mode = a\_locking \Rightarrow s\_v < a\_high$$

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言えるはずである．モデル上では，このことは次の不変条件が成り立つことの証明によって確認できる．

$$a\_high \leqq s\_v \Rightarrow a\_mode = a\_locked$$

したがって，次の関係を満たすように l_limit を設定すれば，s_v が l_limit 以上である時にアクチュエータによるドアロック動作が完了していることが証明できる．

$$a\_high \leqq l\_limit$$

6. 整合性の検証

作成した形式モデルは，まずモデル検査ツール ProB を使ってイベントが想定した順序で起こることを確認し，その後に定理証明支援ツールを使って不変条件を証明した．一般に，証明には人手の関与が必要であり手間が大きいだけでなく，形式モデルに表現されたシステムの動作が意図通りであることを直感的に理解し難い．例えば，イベント sensor の記述に誤りがあって s_v の値が常に初期値 0 のままであっても不変条件は満たされるので，不変条件を証明する過程で意図との乖離に気付くことは難しい．モデル検査ツールを使った形式モデルの実行によってこのような誤りを早い段階で取り除くことは，検証の作業効率を上げるうえで有効であった．

モデル検査ツールを使うためには形式モデルが実行可能でなければならず，実行可能にする過程でモデルを大きく改変すると実行する意味が失われてしまう．前述の形式モデルは基本的に整数を操作しているのみであり，Event-B の関数や関係を使った宣言的な定義によるデータ構造は用いていない．したがって，各定数にコンテクストの公理を満たす適切な整数値を与えることで，モデルを実行することができた．

しかし，不変条件の検証の観点からは，モデル検査では与えた個々の定数値に関する検証しか行えないため十分とは言えない．確認したいのは，制約を満たす定数値に対して，車速がどのように変化してもセンサ値が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．また，網羅的な検証を行うためには，変数 s_v が取り得る値の数を少数に限定する必要があり，この点からも検証としては不十分である．

定理証明による検証では，一階述語論理で表された制約に基づいて証明課題を証明するため，定数に具体的な値を与える必要がなく，変数が取り得る値の数を限定する必要もない．その一方，証明課題が定理証明支援ツールによって自動証明されるとは限らず，対話証明による人手の関与が必要になる．また，前述の形式モデルのようにイベントがセンサ，コントローラ，アクチュエータの３種に分かれて連携する場合，目的とする不変条件

$$l\_limit \leqq s\_v \Rightarrow a\_mode = a\_locked$$

は３種のイベントの連携の結果として証明できる．これは，センサ，コントローラ，アクチュエータの各々について不変条件を証明した後，それらから定理として目的の不変条件を証明することで可能である．しかし，証明課題の数が増えて，証明が煩雑になる．

記述した形式モデルの行数とツールが生成した証明課題の数を，下表に示す．ここで，詳細化後のモデルについては，詳細化前のモデルに追記した行数のみを示した．また，自動証明は定理証明支援ツールによって自動証明された証明課題の数を表し，対話証明は人間が対話的に指示を与えて証明した証明課題の数を表す．

モデル	行数	証明課題数	自動証明	対話証明
モデル１	36	2	2	0
モデル２	46	27	22	5
モデル３	83	107	97	10
合計	165	136	121	15

証明課題数に占める自動証明数の割合は，モデル１が 100%，モデル２が 81%，モデル３が 91%であり，全証明課題については 89%となった．割合が極端に低いモデルがないという意味で，検証の観点からは各モデルの複雑度はそろっており，リファインメント戦略は妥当であると考えられる．

7. まとめ

本稿では，時間経過にともなう外部環境の変化と，それに対するシステムの反応のモデル化と検証について，架空の自動車ドアロックシステムの設計を題材として述べた．形式モデル作成には Even-B を用い，求める性質を不変条件として記述した後，それが成り立つことを検証した．

作成した形式モデルは，まずモデル検査ツールを使ってイベントが想定した順序で起こることを確認し，その後に定理証明支援ツールを使って不変条件が成り立つことを証明した．これにより，車速がどのように変化しても，センサ値が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が，形式モデル上で検証できた．

本稿のケーススタディの範囲では，あらゆる場合について目的の性質が成り立つことを検証するためには，定理証明による検証が有効であった．一方，形式モデルの動作を確認し誤りを早い段階で取り除くには，モデル検査ツールを使った形式モデルの実行が有効であった．形式モデルの作成では，４節で述べたように仮定を置いて対象を抽象化している．検証目的に対して十分妥当な抽象化と考えるが，現実の問題を反映した抽象化となっているかについて，今後の評価が必要である．

参考文献

1) Abrial J.-R.: The B-Book, Cambridge University Press 1996.
2) Abrial, J.-R: Modelling in Event-B, Cambridge University Press, 2010.
3) Abrial, J.-R.: Formal Methods in Industry, in Procs. of ICSE 2006, pp. 761-767, 2006.
4) Abrial, J.-R., Su, W. and Zhu, H.: Formalizing Hybrid Systems with Event-B, in Procs. of ABZ 2012, 2012.
5) Hoang, T.-S., Kuruma, H., Basin, D., Abrial, J.-R.: Developing Topology Discovery in Event-B, Science of Computer Programming, Vol. 74, No. 11-12, pp. 879 – 899, 2009.
6) Hudon, S. and Hoang, T.-S.: Development of Control Systems Guided by Models of their Environment, Proc. B 2011, Electronic Notes in Theoretical Computer Science, Vol.280, pp.57-67, 2011.
7) 來間，中島：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.
8) 中島震: 形式手法入門，オーム社, 2012.
9) Romanovsky, A. and Thomas, M. (eds.): Industrial Deployment of System Engineering Methods, Springer, 2013.
10) Event-B.org, http://www.event-b.org/
11) The Pro B Animator and Model Checker, http://www.stups.uni-duesseldorf.de/ProB/