情報処理学会論文誌　プログラミング　Vol.7　No.2　20–35 (June 2014)

信号装置仕様の検証を通じた Bメソッドにおける仕様記述法の検討

寺田 夏樹[1,a)]

受付日 2013年11月11日, 採録日 2014年2月28日

概要：高信頼性が要求される分野において形式手法を用いることでソフトウェアの信頼性を向上させることが期待されている．我々は鉄道の信号保安制御に関わる装置のうち，単線区間の運転方向を制御する自動閉そく装置について，形式手法の 1 つである B メソッドによりモデル化を行い，定理証明による検証を行った．B メソッドでは検証済みコードの生成が可能であるが，実際にシステム仕様の記述を行うと，記述の正当性を保証するために証明すべき条件である証明責務が大量に生成され，検証作業が困難となる場合が多い．本報告においてもシステムが複雑になるにつれて証明責務の生成数が多くなり，対話的証明の実行数（User Pass）が増えていくことを示す．この結果を受け，本報告では証明責務や User Pass の数と仕様記述の関係を考察し，条件分岐により証明責務の生成数が増えることを示す．また一方で，実行される演算結果が満たすべき条件に，システム変数間の制約として記述される不変条件を含めたり，不変条件の表現に条件分岐の条件を近づけたりすることで，自動証明される証明責務の割合が増え，User Pass を減らせることを示す．これに基づき，証明責務や User Pass の数を減らすための仕様記述法を検討した．この手法を単線区間向け閉そく装置の仕様記述に対して適用した結果，証明作業の負担を軽減できたことを報告する．

キーワード：形式手法，自動閉そく装置，B メソッド，証明責務，User Pass

Study on Expression of Specification in B through the Verification of Railway Signalling Systems

Natsuki Terada[1,a)]

Received: November 11, 2013, Accepted: February 28, 2014

Abstract: Formal methods are expected to increase reliability of software, including that of railway signalling systems. We modeled the specifications of automatic block systems for single lines with B, which is one of the formal methods, and verified the model with theorem proving. Although we can obtain verified code with B, more proof obligations are generated to assure correctness of the specification as the specification becomes complicated, which makes the verification difficult. We show that the numbers of proof obligations and user passes to prove them increase as the system becomes complicated. Then we studied the relationships between the expression of the specifcations and the numbers of proof obligations and user passes. We show that use of conditional branch increases proof obligations, while it reduces the number of user passes to approximate the conditions satisfied by operations results to the invariant which is the conditions among system variables. In consideration of the relationships, we have studied how to improve specifications to reduce the number of proof obligations and user passes. We applied the method to the specification of automatic block systems for single line, and reduced the verification process.

Keywords: formal methods, automatic block system, B-mehod, proof obligations, user pass

1. はじめに

鉄道において列車の衝突や追突を防ぐための信号装置に

[1] 公益財団法人鉄道総合技術研究所
Railway Technical Research Institute, Kokubunji, Tokyo 185–8540, Japan
[a)] terada.natsuki.08@rtri.or.jp

は高度の安全性とフェールセーフ性が要求される．従来はリレーの非対称故障誤り特性を利用し，故障時に安全側に制御されるようにシステムを構築してきた．近年はマイクロエレクトロニクスが導入されているが，この電子化機器については，多重系構成とフェールセーフ性を持つ比較器とで装置を構成することによりハードウェアとしてフェールセーフの確保が実現され，現在広く利用されるに至っている．今後も電子化機器においては処理能力の高度化が期待されるが，その場合，ハードウェアに搭載されるソフトウェアの安全性をいかに確保するかが重要となってくる．

信号装置用のソフトウェアにおいては安全性を確保するために，ソフトウェアダイバーシティ（N バージョンプログラミング）や，シングルスレッド・定周期起動方式の使用など，多くの技術が適用されているが，そもそもプログラムが誤っている場合には信頼性が確保できない．

我々はプログラムの信頼性を向上させる手法として形式手法（formal methods）に着目してきた [1], [2]．形式手法とは，システムやプログラムを何らかの数学的な背景を持つ形式で記述し，それを数学的に検証したり，計算機支援により妥当性や正当性を検証したりする手法である．

形式手法は海外での適用事例が見受けられるものの，国内ではあまり採用されていない．形式手法の鉄道信号分野への普及のためには，様々な信号装置について形式手法を用いたモデル化を行い，例示することが有効と考えられる．この方針に基づき，鉄道の単線区間の運転方向を制御する自動閉そく装置について，形式手法によるモデル化を実施し，モデルの検証を実施した．

ここで我々は，仕様を満たす検証済みコードの生成が可能という点に着目して B メソッド [3] を使用した．B メソッドとは Abrial が提唱したソフトウェア開発手法であり，仕様の段階的詳細化と証明により，プログラムを生成する手法である．B メソッドについては海外では実用事例 [4], [5] が報告されているが，証明を行うために記述の柔軟性が低くなっており，実際に B メソッドを扱うと，仕様の記述の困難さに直面する．また，単に仕様を工夫なしに記述していくと，システムが大規模になるにつれ，大量の証明責務が生成され，処理が難しくなっていく．

そこで本報告では，B メソッドにおける証明という観点からの仕様記述の改良についても検討を行った．まずは似通ったシステムでありながら少しずつ特徴が異なるシステムのモデル化とそれに対する証明責務の様相を考察し，証明責務の生成規則に立ち返ることで，仕様記述と証明責務の生成数との関係を明らかにする．次に，実際の検証の手間が対話的証明を実行する時間によることから，より多くの証明を自動で完了させるための仕様記述法について考察し，提案を行う．最後に提案手法を適用することで，証明の負担が軽減されることを報告する．

本報告の構成は以下のとおりである．まず，2 章において対象となる単線区間向けの自動閉そく装置を 3 種類紹介するとともに，今回使用した B メソッドについて解説する．3 章では 3 種類の中で最も単純な自動閉そく式（特殊）と呼ばれる方式についてモデル化を行い，自動生成される証明責務に対して自動証明および対話証明を実施した事例を紹介する．引き続いて 4 章では，駅間で複数列車が走行可能な単線自動閉そく式と呼ばれる方式，5 章では駅間の列車の在線をチェックイン・チェックアウト方式で検出する特殊自動閉そく式と呼ばれる方式について同様にモデル化と検証を紹介するが，システムが複雑になるにつれて，検証が難しくなっていくこともあわせて示す．

3〜5 章のモデル化においては，特に証明を意識せずに仕様の記述を行った．それに対し，6 章において，証明責務の生成規則に立ち返り，どのように仕様の記述を行えば証明の負担を減らせるか検討を行った．検討結果をもとに，自動証明を考慮に入れた仕様記述法を提案し，適用例を示す．

2. 単線自動閉そく装置の仕様検証

2.1 単線区間向けの閉そく装置

信号装置の代表としては駅構内の信号機やポイントを制御する連動装置や，列車の速度を制御する ATS（Automatic Train Stop，自動列車停止）装置，ATC（Automatic Train Control，自動列車制御）装置などが考えられるが，ここでは連動装置ほどは複雑でないものの，連動装置と同じようにリレー論理で制御される単線区間向けの自動閉そく装置をモデル化および検証の対象とした．鉄道では列車の衝突や追突を防ぐために，**図 1** のように線路を一定区間ごとに区切り，その区間内に 1 本の列車しか進入させないようにしている．この区間のことを閉そくと呼び，各閉そくに列車を 1 本しか入れないように，列車の在線状況に応じて制御する装置を自動閉そく装置と呼んでいる．さらに単線区間では衝突を防ぐために列車の運転方向を一定とするが，この制御部分も閉そく装置に含まれる．

運転方向の制御のうち，要となる方向回線と呼ばれる部分の制御に着目し，共通点が多く見られる単線自動閉そく式（図 1），自動閉そく式（特殊）（**図 2**），特殊自動閉そ

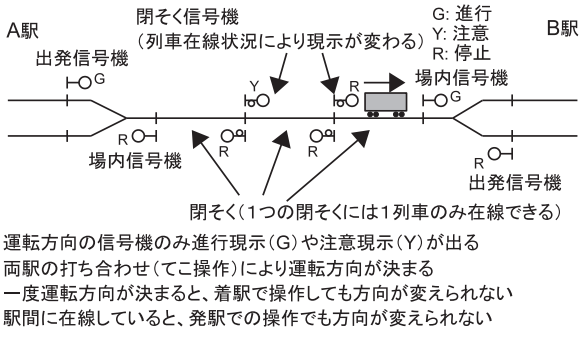

図 1　単線自動閉そく装置

Fig. 1　Automatic block system for single line.

図 2　自動閉そく式（特殊）

Fig. 2　Simplified automatic block system.

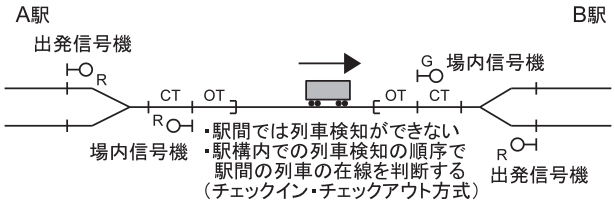

図 3　特殊自動閉そく式（軌道回路検知式）

Fig. 3　Semi-automatic block system.

く式（軌道回路検知式）（**図 3**）の各装置を対象に仕様のモデル化を実施した．実際には閉そく装置は単独で存在するのではなく，連動装置の一部として実装されており，継電（リレー）連動装置の標準結線図 [6], [7] にも閉そく機能に関する記載があるが，ここでは実績のある結線図を検証するのではなく，閉そく機能だけを取り出して装置化した場合に安全性が検証できるかという視点でモデル化を行った．そのためリレーを 1 対 1 で変数に対応させるのではなく，リレーが実現している機能に着目してモデル化した．

閉そく機能の中心となるのは方向回線（駅間に設備され，運転方向の制御を司る 1 対もしくは 2 対の回線）と運転方向てこ（てことは設定を行うためのレバーのこと）である．上記 3 種の装置において，詳細は異なるが，基本は以下のとおり共通している．

- 両駅の方向てこの向きを一致させることにより方向回線が構成される（取扱上は着駅側から取り扱う）．構成時において方向回線は到着側から電力が送電される．また運転方向リレーと呼ばれるリレーの状態が方向に応じて決定される．
- 1 度方向回線が構成されると到着側の方向てこを扱っても運転方向は変わらない．
- 出発側の方向てこは，出発進路（進路とは列車に対し駅での通行の仕方を規定するもので，駅から駅間へ列車が進むための進路が出発進路である）の設定状況や，駅間の在線などにより鎖錠，すなわちロックされる．この機能は運転方向鎖錠リレーと呼ばれるリレーが落下（無励磁状態となる）することで実現される．この間に方向てこを扱っても運転方向は変わらない．

今回は，両駅が列車を送出する条件にならないこと，着駅でのてこ操作が無効になるなどの要件についての検証を行うこととした．

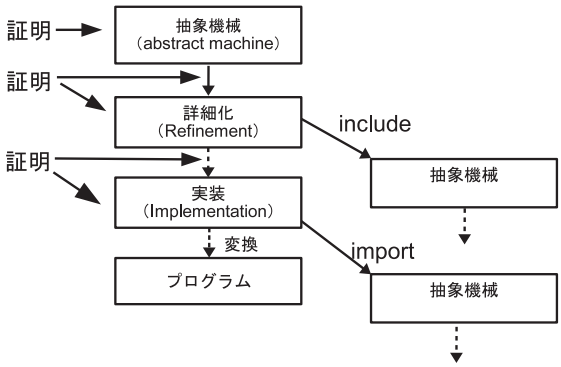

図 4　B メソッドによるプログラム開発

Fig. 4　System development with B.

2.2　B メソッドの解説

仕様を検証する場合，テストによる検証が通常行われるが，テストでは任意の入力パターンに対して正常に動作するかどうかを保証するには限界がある．これに対し，仕様に対する命題を証明することで，任意の入力パターンに対する正当性を保証できる．そのため定理証明を行うことは，仕様の信頼性を保証するうえで大きな意味を持つ．今回使用した B メソッドとは検証済みプログラムの生成に重きを置いた形式手法の 1 手法であり，仕様から生成される定理の証明と，仕様を段階的にプログラムまで詳細化するという概念を持ったものである．

B メソッドでのシステム開発イメージを**図 4** に示す．ここでは最初に抽象機械（abstract machine）と呼ばれる仕様を記述する．記述に使用する言語を B 言語，あるいは単に B と呼ぶが，Z 記法 [8] と強く関連しており，集合に関する記法が細かく定義されているという特徴がある．抽象という言葉が示すとおり，ここでの仕様記述は，演算結果やアルゴリズムを非決定的に記述することが可能である．この抽象機械を詳細化（refinement）していく．この詳細化とは，演算がより決定的，つまり演算結果の値域がより狭くなるということを意味する．詳細化においては，変数や演算結果が詳細化前の条件を満たしていることを証明する必要がある．最終段階は実装（implementation）段階と呼ばれ，プログラムに直接変換可能な形としての記述となる．ここでの演算は決定的なアルゴリズムとする必要があり，使用できる記述も B のサブセット（B0 言語）となる．この実装段階からコード生成器によりコードを得ることができる．

B の抽象機械モジュールの概略を**図 5** に示す．先頭の `MACHINE` が抽象機械の宣言であり，名前が `MA` であることを宣言している．`SET` は集合の宣言であるが，集合としては図 5 の `X` のような列挙型のものか，自然数の集合が定義可能である．`VARIABLES` は抽象変数（abstract variable）の宣言である．その型については，`INVARIANT` の中で定義する．型は通常の整数型や，集合の要素であるという定義だけでなく，集合の積や関数型といったものも定義できる．ここで「抽象」といっているのは，仕様の定義のために使

表 1　B における一般化代入の例
Table 1　Part of generalized substitutions in B.

種類	記述例	概要
恒等代入	`skip`	何もしない
等価代入	`a := b`	単純な値の代入
要素代入	`a :: A`	集合の 1 要素を非決定的に代入
充足代入	`a :(X(a))`	`X(a)` を満たす値を非決定的に代入
有限の非決定的選択	`CHOICE S1 OR S2 OR ... Sn END`	`S1,...,Sn` の 1 つを非決定的に選択
条件選択	`SELECT P1 THEN S1 WHEN P2 THEN S2 ... ELSE T END`	成立する `Pi` に対応する `Si` を非決定的に選択
IF 条件	`IF P1 THEN X1 ELSIF P2 THEN X2 ... ELSE Y END`	通常の IF～THEN と同様
非決定的選択	`ANY X WHERE P(X) THEN S END`	`P(X)` を満たす `X` を用いて `S` を実行する
逐次代入	`A; B`	`A` を実行した後，`B` を実行する
同時代入	`A || B`	独立した代入を同時に行う

```
MACHINE
  MA
INCLUDES
  SS.Minc
SET
  A, X = {x1, x2}
VARIABLES
  aa, xx
INVARIANT
  aa : NAT & xx : X
INITIALISATION
  aa :: NAT || xx :: X
OPERATIONS
 Op_A == aa :: NAT || xx :: X;
 Op_B == SS.Op_C
END
```

図 5　B メソッドの抽象機械モジュールの概要
Fig. 5　Outline of abstract model in B.

用するということであり，実装される形態とは異なっていてよい．実装される具象変数（concrete variable）も定義できるが，これについては列挙型か自然数型，あるいはその配列のみが許されている．

`INVARIANT` は不変条件と呼ばれ，前述のように変数の型を宣言するほか，変数の値の間の関係についても宣言可能である．ここで : は左辺が右辺の集合の要素であることを示している．ここで宣言された不変条件は，`INITIALISATION` に示されるインスタンス生成時の初期化実行後，また `OPERATIONS` の中に記述される operation 実行後にはつねに満たされている必要がある．

初期化や operation の中で記述できるのは一般化代入（generalized substitution）と呼ばれ，代入の概念を拡張したものである．**表 1** に主なものを示す．大きく分けると，等価代入のような基本的な代入と，条件選択のように複数の一般化代入を組み合わせて大きな一般化代入にするための代入がある．図 5 の `aa :: NAT` は `aa` に `NAT`（有限の自然数）の 1 要素を非決定的に代入するものとなる．また `||` は 2 つの代入を同時に行うことを意味している．

また，構造化の仕組みとしては include リンクと呼ばれるものがある．抽象機械や詳細化段階では図 5 で示したように外部モジュールの抽象機械を `INCLUDE` 文で宣言することにより，外部モジュールのインスタンスを部品として利用することが可能になる．include するモジュール `Minc` の前に `SS` と付けることで，この operation や変数について図 5 の `SS.Op_C` のように `SS` を付けて参照が可能となる．インスタンスの値の変更は外部モジュールで定義された operation を通じて実施する必要がある．また operation での振舞いはあくまでも include する抽象機械の定義によるため，その詳細化段階で加えられたアルゴリズムには立ち入ることができない．

2.3　モデル化と検証の方針

B メソッドはフランスを中心に適用事例が多数報告されている [4], [5]．しかし B では証明を意識しているために記述の柔軟性が低い．たとえば抽象機械の段階においては逐次代入が使用できず，その代わりに同時代入しか使用できない．また，モジュール内での補助的な演算については，たとえ内部状態に影響を与えないものであっても，サブルーチンとして定義することは実装段階でしか許されていないため，マクロとして記述するか，他の抽象機械の operation として記述するしかない．しかも抽象機械においては逐次代入ができないという制約がゆえに，同一の外部モジュールの operation を同時に呼び出すことができない．たとえばモジュール B に `Op_A` と `Op_B` が定義されていた場合に，これらを他のモジュール A から呼ぼうとして `Op_A || Op_B` と記述することが許されていないのである．これらの B の制約に慣れていないと仕様の記述が難しい．そのため，たとえモデル化の例を B で提供しても，それに倣った記述を行うのが容易でない．そこで比較的記述が容易で，日本語が使用可能な VDM [9] を併用した．VDM は仕様の記述とその妥当性検証に重きを置いた手法である．VDM のうち，オブジェクト指向対応の VDM++で記述を行った．

VDM++では基本的なモジュールの構成単位はクラスと

なり，クラスの内部にインスタンス変数および function（引数のみを用いて，返り値を得る演算）および operation（引数のほかにインスタンス変数を参照，変更できる演算）を定義する．ここで function をサブルーチンとして定義可能であり，B よりも柔軟な記述を可能にしている．VDM++での変数の型付けは B とは異なり，変数の宣言と同時に行うが，VDM++ではインスタンス変数間の制約として記述される不変条件のほかに，型の一部として不変条件を記述できる．VDM++ではレコード型（record type）と呼ばれる複数の変数の組で定義される型が使用しやすく，項目間に成立する条件として不変条件を宣言できるほか，レコード型の項目に不変条件付きのレコード型を用いることも可能で，operation や function の引数の型としても不変条件の付きのレコード型が利用できる．このレコード型の使いやすさが VDM++の記述のしやすさにつながっている．

基本的なモデル化および検証の流れは以下のとおりである．

- システムを方向回線，起点方駅装置，終点方駅装置の 3 つに分け，その上に 3 つのシステムを統合する全体システムを置く．なお，単線自動閉そく装置についてはさらに軌道回路と呼ばれる部分を追加した．
- これらを VDM++で記述する．
- VDM++記述を B に変換する（手作業で実施）．
- B のツールで生成される証明責務を証明する．
- B の詳細化を実行する．

VDM++および B では不変条件が定義できるが，システムとしてつねに不変条件を満たす必要がある．このように記述の正当性を示すために証明すべき定理を証明責務と呼ぶが，これは仕様から自動的に生成される．この証明責務を計算機上で証明するのが今回実施した検証の主要な部分となる．実際に検証したのは 2.1 節に記述された内容のうち

要件 1　両駅が同時に列車を送出する条件にならない，
要件 2　着駅でのてこ操作が無効となる，
要件 3　列車出発後，到着までの間，運転方向が固定される，

といった項目である．これにより自動閉そく装置の機能が実現され，列車が安全に運行できるかどうかを確認できる．

3. 自動閉そく式（特殊）のモデル化と検証

3.1 自動閉そく式（特殊）の VDM モデル化

まず，図 2 に示す自動閉そく式（特殊）の装置についてモデル化を実施した．このシステムでは駅間の閉そくは 1 つに限定される．駅間の列車の在線については有無が分かればよく，位置は問わない．実際にはレールに流れる電流によって列車を検知する軌道回路と呼ばれる装置が設備されている．歴史的に後述する単線自動閉そく式を簡略化したものであるために特殊と名前が付いているが，3 つの装置の中では最も単純な構成となり，運転方向の制御としては最も基本的といえる．

```
inv
not (運転方向 = <下り> and 反対側方向回線 = <上り>) and
(運転方向 = <上り> =>
  (進路鎖錠 = <解錠> and 運転方向鎖錠 = <解錠>)) and
(運転方向 = <下り> =>
  ((方向回線在線 = <在線> or 進路鎖錠 = <鎖錠> or
    進路てこ = <設定>) =>
      運転方向鎖錠 = <鎖錠>))
```

図 6　起点方駅装置の VDM++モデル（不変条件）
Fig. 6　Invariant of origin station equipment model in VDM++.

駅装置については，それぞれに運転方向リレーに相当する変数「運転方向」と，運転方向鎖錠リレーに相当する変数「運転方向鎖錠」，方向てこに相当する「方向てこ」などを記述した．両側の駅装置間では役割が共通であるリレーが多く見られる．そこで VDM++の記述では駅装置クラスを作成し，その継承として起点方および終点方駅装置クラスを記述した．起点方駅装置における不変条件を**図 6** に示す．< > で囲まれた値は列挙型の値を示している．VDM++では bool 型の true，false といった値を使うこともできるが，日本語の列挙型を使うことで意味を明確にできる．最初の行は自分自身の「運転方向」の設定が下り，かつ「反対側方向回線」（反対駅での方向回線設定）が上り，という状態にはならない，つまり要件 1 を示している．次の 2 行では「運転方向」が上り，つまり到着側となっている場合には「進路鎖錠」（自駅から駅間に向かう進路が設定されていると鎖錠となる）が解錠され，「運転方向鎖錠」が解錠される（少々分かりにくいが，リレーの制御では回路構成上解錠となる．運転方向が上りであれば基本的には運転方向の設定ができないため，問題はない）ことを示している．最後の 4 行は下りの場合の要件 3 の静的条件として「運転方向鎖錠」が鎖錠される条件であり，列車が在線しているか，「進路鎖錠」が鎖錠されて列車が出発できる状態であるか，「進路てこ」が設定されている（進路の設定を要求している）場合には，「運転方向鎖錠」が鎖錠されていることを示している．このように日本語が使えることで，可読性が増しており，これは考えを整理するうえで重要な点である．

これに，てこの操作や列車の移動に対応する operation や function を加える．VDM++ の operation には 2 種類の記法が提供されている．1 つは値の代入や他の operation 呼び出しを逐次的に実行する explicit（陽）な定義であり，もう 1 つは operation 実行前の引数と状態変数の間の制約（事前条件）と実行後の引数，状態変数の間の制約（事後条件）を記述し，具体的なアルゴリズムを記述しない implicit（陰）な定義である．本報告では基本的には explicit な定義を行った．**図 7** に下りに方向てこを設定する行為に相当する operation を例として示す．ここでは「方向てこ」を下りにしたうえで，「運転方向」が上り，かつ「方向回線在線」（方向回線での列車の在線状態の判定結果を示す）が非

```
operations
public 下り設定 : () ==> 方向回線`方向型
下り設定 () ==
  (方向てこ := <下り>;
  if 運転方向 = <上り> and 方向回線在線 = <非在線> and
     反対側方向回線 = <下り> then
     (運転方向 := <下り>;
      運転方向鎖錠 :=
        (if 進路てこ = <設定> then <鎖錠> else <解錠>));
  return 運転方向);
```

図 7　起点方駅装置の VDM++モデル（下り方向への設定）
Fig. 7　Operation corresponding to setting the direction down in VDM++.

```
INVARIANT
   not(direction = down & oppositedirect = up) &
     (direction = up =>
        (routelock = unlock & directlock = unlock)) &
     (direction = down =>
        (((existence = detect or routelock = lock
        or routelever = set) => directlock = lock) &
        ((existence = nondetect & routelock = unlock &
           routelever = unset)=> directlock = unlock)))
```

図 8　B における不変条件の記述例
Fig. 8　Example of invariants in B.

在線，かつ「反対側方向回線」が下りの場合に，「運転方向」を下りにし，「運転方向鎖錠」の状態については，「進路てこ」が設定されている場合は鎖錠とし，そうでない場合は解錠としている．なお，図 7 では遷移条件を省略したが，基本的には遷移条件は事後条件として記述可能で，post というキーワードに続けて記述する．

一方，方向回線は 3 本の回線で構成される 2 対の回線であり（1 本を共通の負極としている），1 対は方向てこを制御するための回線，もう 1 対は軌道回路の条件を照査して運転方向表示リレー（FKR）を動作させるための回線である．これに対応して両駅のてこの状態と軌道回路の状態で方向回線を記述した．

3.2　自動閉そく式（特殊）の B モデルへの変換と証明

次に VDM++の記述を B に変換する．まずは変数を変換する．本報告ではほとんどレコード型は使っていないが，VDM++のレコード型は B においては複数の変数として宣言し，別途不変条件を宣言する．また，VDM++のレコード型に対応して 1 つのモジュールを宣言してしまうのも 1 つの方法である．

そして不変条件も変換する．基本的に VDM の列挙型は B の列挙型集合に置き換えることができる．**図 8** に図 6 に対応する不変条件を示す．direction が図 6 の「運転方向」，oppositedirect が「反対側方向回線」，routelock が「進路鎖錠」，directlock が「運転方向鎖錠」，existence が「方向回線在線」，routelever が「進路てこ」である．

VDM の operation のうち，explicit な定義については，

```
xx <-- DownSet =
  ANY dir WHERE
    dir : DIRECTION &
    ((direction = up & directlock = unlock)
              => dir = down) &
    ((direction = down or directlock = lock)
              => dir = direction) &
    transition(direction, directlock, oppositedirect,
              dir, directlock, oppositedirect)
  THEN
    directlever := down ||
    direction := dir ||
    xx := dir
  END;
```

図 9　B の当初の抽象機械モデルの例（起点方駅装置の operation）
Fig. 9　An operation of first abstract machine in B.

B の抽象機械で逐次代入が使えないため，まずは同時代入できる形式としたうえで B の記述に変換する．たとえば

```
if a = b then a := c else a := 0;
b := a * 2;
```

のように条件分岐が途中にあるような場合には

```
if a = b then (a := c; b := c * 2)
else (a := 0; b := 0);
```

として，各分岐の中で実行される演算の順序に関係なく同一の結果が得られるようにする．また implicit な定義の場合，B において直接事後条件を表現できないので，表 1 の非決定的選択を用い，その条件にある P(X) に事後条件を記述するという手法をとる．

VDM++の function については，マクロとして定義するか，別途モジュールを定義し，function を適用している箇所については関数型として宣言された抽象変数を使って記述する方法をとる．この抽象変数はこの別モジュールを詳細化することで実装される．

図 7 に対応する B の抽象機械における記述例を**図 9** に示す．ここで非決定的選択を使用し dir という変数を宣言している．dir : DIRECTION は型宣言であり，その次の 2 行は図 7 の if 条件が成り立つ場合，dir の値を下りに相当する down とすることを述べている，その次の 2 行の条件は図 7 の if 条件の否定を意味しており，if 条件が成立しないときは dir は direction の値と等しいということになる．

その次の 3 行中に transition という記述があるが，これは事前の状態を記述する変数と事後の状態を記述する変数の間の条件を記述したマクロであり，ここで動的な制約を記述する．マクロということで operation 間で共有可能である．具体的には**図 10** に示される．transition の最初の 3 変数は演算適用前の direction，directlock，oppositedierct を引数とし，後の 3 変数は適用後の direction，directlock，oppositedierct を引数とする．前半の 3 行は「適用前の運転方向設定が下りかつ反対側の運転方向設定が下りであれば，運転方向が鎖

```
DEFINITIONS
   transition(dir, d_lo, op, dir2, d_lo2, op2) ==
  ((dir = down & op = down) =>
  (((d_lo = lock & d_lo2 = lock) => dir2 = dir) &
    op2 = op)) &
  ((dir = up & op = up) => dir2 = dir)
```

図 10　状態遷移条件の記述例
Fig. 10　Example of condition in state transitions.

錠されつづけていれば運転方向設定は変わらないし，反対側の運転方向も変わらない」ということで，これは要件 3 の動的な部分に対応する．最後の 1 行は「適用前の運転方向が上りかつ反対側の運転方向が上りであれば，運転方向設定は変わらない」ということで，要件 2 に対応する．

operation の最後の部分に等価代入が並んでいるが，ここで条件を満たす dir を実行後の direction の値としている．最後の代入部分に directlock，oppositedirect がなく，これらの変数は値が変わらないため，transition において事前事後の両方の値として使用している．なお，実際のモデルにおいては最後の代入部分を別の抽象機械を include し，その operation を使用しているが，図 9 では，説明上それを展開している．

証明責務は，Atelier B [10] などの B メソッドの開発ツールを用いれば，仕様から自動で生成されるが，大部分は自動的に証明される．残りはツールを使って対話的に証明する必要がある．本報告では Atelier B（4.1.0）を使用した．

証明責務は $A_1, \cdots, A_n \vdash (X \Rightarrow Y)$ の形の命題として与えられるが，対話的証明は，これを deduction（$A_1, \cdots, A_n \vdash (P \Rightarrow Q)$ を $A_1, \cdots, A_n, P \vdash Q$ と変形する），modus ponens（$(P \Rightarrow Q), P \vdash Z$ の場合に $(P \Rightarrow Q), P \vdash (Q \Rightarrow Z)$ と変形する），補題の追加などといった命令の組合せで実施し，最終的に $A_1, \cdots, A_n, Z \vdash Z$ の形にするものである．また，$A_1, \cdots, A_n \vdash (X \wedge Y)$ の場合，$A_1, \cdots, A_n \vdash X$ と $A_1, \cdots, A_n \vdash Y$ に命題が分割され，それぞれを証明することとなる．

証明責務は最初からすべて証明できるわけではなく，記述に不足や矛盾がある場合には証明ができない（不成立とも判定されない）．このような場合，証明責務が成立するように記述を修正する必要があるが，そのたびに証明責務は生成し直す．図 9 は最終的には**図 11** のような形となり，不変条件や遷移条件に対応して条件が追加されただけではなく，directlock についても値を変更する必要が生じている．

3.3　B モデルの詳細化と検証結果

前節における抽象機械の仕様記述に対して，これを詳細化した演算を記述する．概要で述べたとおり，詳細化段階においては operation 実行結果の値域がより狭く，なおかつ実行結果が詳細化前の段階の条件を満たすような記述が求められる．最終的にはプログラミング言語に変換可能

```
xx <-- DownSet =
  ANY dir, dir_l WHERE
    dir : DIRECTION & dir_l :LOCK &
    ((direction = up & existence = nondetect &
            oppositedirect = down) =>
      (dir = down &
        (routelever = set => dir_l = lock) &
        (routelever = unset => dir_l = unlock))) &
    ((direction = down or existence = detect or
         oppositedirect = up) =>
      (dir = direction & dir_l = directlock)) &
    transition(direction, directlock, oppositedirect,
       dir, dir_l, oppositedirect)
  THEN
    directlever := down ||
    direction := dir ||
    directlock := dir_l ||
    xx := dir
  END;
```

図 11　最終的な B の抽象機械の例（図 9 の operation に対応）
Fig. 11　Final abstract model corresponding to Fig. 9.

```
xx <-- DownSet =
IF direction = up & existence = nondetect &
   oppositedirect = down THEN
     xx := down;
     IF routelever = set THEN
        directlever := down ||
        direction := down ||
        directlock := lock
     ELSE
        directlever := down ||
        direction := down ||
        directlock := unlock
     END
ELSE
    xx := direction;
    direction := down
END;
```

図 12　B の最終的な実装モデルの例（図 11 の operation に対応）
Fig. 12　Final implemenation in B, corresponding to Fig. 11.

な形にまで詳細化し，それを変換してコードが得られる．図 9 や図 11 では非決定的選択を使用しているが，実装段階では値が決定的に選択される必要があるため，このような記述は行えない．そこで IF 条件を用いて**図 12** のように記述した．図 11 の段階で，ある条件が成立する場合の関係式，成立しない場合の関係式という記述の仕方をしている．そのため，これを IF 条件に変更するには，その条件を IF 以下の条件式として，THEN 以降に成立する場合，ELSE 以降に成立しない場合の値の代入を記述すればよい．

最終的には今回の仕様記述に対しては証明責務をすべて証明することができた．実際に証明した数を**表 2** に示す．この数は最初の抽象的な段階での証明から実装段階の証明までを含んだものであり，その大小は仕様の複雑さを示す目安となる．なお，合計については表に記していないモジュールを含むので各行の和とは一致しない．表 2 におい

表 2　証明の数（自動閉そく式（特殊））
Table 2　The number of proofs for simplified automatic block systems.

module	行数 (B)	証明責務	自動証明	対話的証明	User Pass
Station	209	99	85	14	13
Direction Control	60	21	17	4	4
Whole System	207	243	195	48	24
summary	761	462	382	80	56

ては最初の段階での B の記述行数についても示した．今回はおよそ 8 割の証明責務が自動的に証明されたことになる．

複数の証明責務に対して，同一の命令の列で証明が完了する場合がある．証明において実行した命令列を Atelier B では User Pass と呼んでいるが，その数を表 2 に加えた．実際の証明の実行の手間は User Pass の数に応じて増えるが，この例では全体システムを除くとほぼ対話的証明の数に一致している．この User Pass は保存可能であり，仕様に修正があった場合でもこれを実行することで，再検証の手間を省略することが可能である．モデルの再利用を考えた場合，User Pass を含めて提供を行えばよい．

なお，B においては推論規則（rule）を追加することにより，自動証明できる数が増えるが，rule は検証しない形で追加可能であり，場合によっては証明系の整合がとれなくなる．そのため，本報告においては追加を行っていない．

次章以降においては，表 2 の証明責務の数を基準に他のシステムを記述したときの証明責務の数がどのように変化するかを，この後検証していく．

4. 単線自動閉そく式のモデル化と検証

4.1 単線自動閉そく式とは

続いて図 1 に示す単線自動閉そく式のモデル化および検証を実施した．単線自動閉そく式では，自動閉そく式（特殊）と比べると，駅中間に閉そくが複数あることを想定しており，同一方向であれば異なる閉そくにおいて別の列車を走行させることが可能となっている．軌道回路は閉そくごとに設備される．複数列車の間隔を制御するため，列車が在線している閉そくについては，閉そく入口の信号機を停止現示とし，さらに 1 つ外側の閉そく入口の信号機を注意現示，つまり速度を落とすように指示している．この現示情報を制御するため，列車を検知する軌道回路の送電方向を変えたり（列車に向かって電流を流す），極性を変えたり（前方の閉そくに在線しているときに極性を反転する）することで，レールに情報伝送の役割を担わせている．標準結線図 [6] では下り用の軌道回路送受信器と上り用の軌道回路送受信器が設備され，方向回線が設定された方向の送受信器が使用される．方向回線は設定された方向の軌道回路のいずれかが列車を検知すると鎖錠される．この現示制御を含めてモデル化を実施することとした．

```
!ii.(ii : TC => (
(ex(ii) = detect => as(ii) = red) &
(tr(ii) = tstop => ex(ii) = detect) &
  ((tr(ii) = t90 & ex(ii) = nondetect)
     => as(ii) = green) &
  ((tr(ii) = t45 & ex(ii) = nondetect)
     => as(ii) = yellow)))
 !ii.(ii : TC => ((ii + 1 : TC & tr(ii) /= tstop) =>
      ((tr(ii + 1) = t45 => as(ii) = red) &
       (as(ii) = red => tr(ii + 1) = t45))))
```

図 13　駅間の閉そくのモデル（不変条件の一部）
Fig. 13　Part of invariants for blocks between stations.

4.2 単線自動閉そく式のモデル化

モデル化においては軌道回路部分について別のモジュールとした．送信の極性を表す `tr`，閉そくの現示を示す `as`，閉そくにおける列車の在線を示す `ex` を用いた場合，軌道回路における送信制御や現示に関わる不変条件を B では**図 13** のように記述できる．ここで `!` は全称限定子 ∀ の意味である．`TC` は閉そくの番号を示す集合，`t90` および `t45` とは極性が通常か，反転しているかを示す．`tr`，`as`，`ex` は後ろに括弧がついているが，この 3 つの変数は `TC` からそれぞれ極性，現示，在線状態への関数として定義されており，括弧の中はその引数となる．したがって図 13 の 2 行目は `ii` 番目の在線状態が検知状態であれば，`ii` 番目の現示が赤であるという意味，3 行目は `ii` 番目の軌道回路が送信停止状態であれば，列車は検知状態になるという意味となる．4，5 行目では現示が緑になる条件，6，7 行目では現示が黄になる条件を示している．最後の 3 行は信号現示と隣接する軌道回路の送信との関係を示しており，`ii` 番目が赤の現示であれば，`ii+1` 番目では反転して送信するということになる．そうすると，`ii+1` 番目に列車がいないときは黄色が現示されることになる．

方向回線に関するモデルについては，いずれかの閉そくで列車が検知されると，方向回線が鎖錠されるという程度の違いであり，自動閉そく式（特殊）と大きな違いはない．

4.3 証明責務の数

このモデルに対する証明責務の数を**表 3** に示す．これらはすべて証明することができたが，軌道回路のモデルが複雑となった結果，軌道回路モデル単独で自動閉そく式（特殊）全体以上の User Pass の数となっている．表 3 では段階ごとに証明責務の数を示した．この増加に対する考察については，6 章において加えていく．

5. 特殊自動閉そく式のモデル化と検証

最後に特殊自動閉そく式（軌道回路検知式）である．ここでは装置の内部状態が複雑になるとどうなるかを示す例となっている．

表 3　証明の数（単線自動閉そく式）
Table 3　The number of proofs for normal automatic block systems.

module	行数 (B)	証明責務	自動証明	対話的証明	User Pass
Station	217	51	29	22	14
(refinement)	188	77	77	0	-
(implementation)	297	29	29	0	-
Direct Control	69	15	13	2	2
(refinement)	75	31	27	4	4
(implemenataion)	50	2	2	0	-
Track Circuit	118	21	18	3	1
(refinement)	128	134	71	63	51
(implementation)	134	282	162	120	71
Whole System	333	1,716	1,586	130	53
(implementation)	211	6,914	6,208	706	54

5.1　特殊自動閉そく式の概要とモデル化

特殊自動閉そく式（軌道回路検知式）とは自動閉そく式（特殊）の設備をさらに削減した装置であり，図 3 に示されるシステムである．この設備では駅間に軌道回路が設備されないため，駅間の列車の存在を直接検知できない．代わりに，駅間への列車の出入りを検出する CT と OT と呼ばれる 2 つの短小軌道回路を両駅に設け，方向鎖錠保持リレーと呼ばれるリレーにより，出発側の CT 進入時から到着側の CT 通過時までの間，方向回線を鎖錠する．すなわちチェックイン・チェックアウトにより鎖錠が行われる．この方向鎖錠保持リレーの挙動は構内の軌道回路の列車検知の順序に大きく依存する．方向回線は 2 本で構成されている．列車の到着時には到着側から方向回線に電力を送出する電源の極性を一時的に反転させて出発側に伝達する仕組みを採用している．さらに列車が退行することを考慮し，列車が停車場内に残った状態から退行後に扱う退行解錠ボタンや停車場外に進出した後に退行するために扱う場内代用てこといったものが設備され，取扱いの手順も決められている．出発信号が出ていないにもかかわらず列車が出発しようとするのを検出する誤出発検知リレーも設備される．駅間に列車が在線していないことを保証するために，駅構内のリレーの数はむしろ増えているのが特徴である．

モデル化では両駅の OT，CT や駅構内の着発線の列車を検知する軌道回路（ホームトラックと称される）と，進路上の軌道回路を記述した．軌道回路が複数あるので抽象機械では変数 `tc` を宣言し，`home`[*1]，`ot`，`ct`，`route` を適用すると，それぞれの在線状況を取得できるようにした．また，実際には様々なリレーの状態によって表現される列車位置を以下のような集合の値で表現することとした．

```
POSITION = {inexistent, home, route, ct, otct,
            ot, between, approach, arrival}
```

[*1] ホームトラックに対応するが，ホームトラックの「ホーム」はプラットホームの略であり，その綴りは本来は platform である．

```
CTNondetect =
PRE tc(ct) = detect
THEN
  tc := tc <+ {ct |-> nondetect} ||
  IF dir = dep THEN
    IF r_lev = unset & r_lock = lock &
       tc(route) = nondetect THEN
      r_lock := unlock ||
      IF s_lev = set THEN s_relay := set END
    ELSIF r_lev = set & r_lock = lock & s_relay = unset
          & (pos = home or pos = inexistent) &
          tc(route) = nondetect & tc(ot) = nondetect &
          mis = nondetect & o_mis = nondetect
    THEN r_aspect := green END ||
    IF tc(route) = nondetect & o_mis = nondetect THEN
      IF r_lev = unset & r_lock = lock &
         pos /: {ct, otct, ot, between, arrival} THEN
         d_lock := unlock
      ELSIF s_relay = set & pos = otct &
            tc(ot) = nondetect & mis = nondetect
      THEN d_lock := unlock END
     END ||
     IF mis = nondetect THEN
       IF s_relay = unset & pos = otct THEN pos := ot
       ELSIF s_relay = unset & pos = ct &
             tc(ot) = nondetect & tc(route) = nondetect
       THEN pos := between
       ELSIF s_relay = set & pos = otct &
             tc(ot) = nondetect & tc(route) = nondetect
       THEN
       IF tc(home) = detect THEN pos := home
             ELSE pos := inexistent END
       END
     END
   ELSIF pos = ct & mis = nondetect THEN pos := between
   END
END;
```

図 14　特殊自動閉そく式における複雑な分岐の例
Fig. 14　Example of complicated conditional branches in semi-automatic block systems.

ここで `otct` は OT と CT に列車がまたがっている状態，`between` は駅間に列車が存在する状態，`approach` は到着駅に列車が接近している状態，`arrival` は列車が到着したことを示している．各列車位置において各軌道回路の列車検知状態が変化した場合に，装置が新たな列車位置をどう認識するかを，結線図をもとに網羅的に調べ，状態遷移表にまとめた．この中には装置として想定していない動作も含まれる．この場合，事前条件を用いて排除するか，何らかの列車位置を割り当てて，動作を規定する必要があるが，今回は後者の方針をとり，その割当をもとに仕様記述を行った．

その結果，列車検知状態の変化に対応する operation に関しては，列車位置などの状態によって複雑な条件分岐をすることとなった．**図 14** に B の記述例を示す．詳細は省略するが，CT から列車が抜けて非在線状態になったときの状態遷移に対応したものである．また不変条件について

表 4　証明の数（特殊自動閉そく式）
Table 4　The number of proofs for semi-automatic block systems.

module	行数 (B)	証明責務	自動証明	対話的証明	User Pass
Station	604	4,979	4,485	494	128
(implementation)	577	6,341	5,386	955	136
Direct control	103	44	26	18	15
(implementation)	95	68	59	9	6
Whole system	342	17,502	17,100	402	61
(implemenataion)	255	8,677	8,644	33	12

表 5　駅構内モデルに対する証明責務の数（特殊自動閉そく式・抜粋）
Table 5　The number of proof obligations for some operations in the station model of semi-automatic block systems.

operation	Machine		Implementation	
	証明責務	分岐	証明責務	分岐
HomeTrackDetect	39	2	20	2
HomeTrackNondetect	39	2	20	2
RouteTrakcDetect	85	4	61	5
RouteTrackNondetect	1,884	137	3,835	245
CTDetect	283	28	606	46
CTNondetect	1,317	82	1,051	92
OTDetect	106	9	78	11
OTNondetect	210	13	160	13

表 6　全体モデルに対する証明責務の数（特殊自動閉そく式・抜粋）
Table 6　The number of proof obligations for some operations in the whole model of semi-automatic block systems.

operation	Machine			Implementation		
	証明責務	分岐	内部含む	証明責務	分岐	内部含む
OriginRoute-TrackNonDetect	102	1	137	3	1	1
OriginCTDetect	2,446	3	364	1,056	2	46
OriginCTNondetect	4,442	2	738	1,534	2	92
OriginMis-DepartureRelease	1,248	3	131	1,108	2	13

も，運転方向の鎖錠条件が増えるだけでなく，位置に応じた変数の制約が加わるため，記述量が増えている．

5.2　証明責務の生成状況

証明責務の数は**表 4** のとおりである．駅装置や全体システムで証明責務の数が非常に多い．これは内部状態による条件分岐が増えたためと考えられる．そこで，**表 5** に駅構内装置モデルの列車検知に関する operation とその中での条件分岐の数を示した．条件分岐の数に対しておおむね 10〜15 程度の証明責務が生成されている．この傾向は単線自動閉そく式の軌道回路モデルと同様の傾向である．

また，全体システムにおいては，**表 6** に示すとおり，一部の operation において極端に多くの証明責務が生成されている．全体システムの仕様記述には表 6 の「分岐」に示されるように条件分岐はほとんどない．しかしここでは呼び出した外部 operation の内部で多くの条件分岐が行われている．これを「内部含む」に示した．このように外部 opeartion を呼び出す場合には，呼び出した外部 operation 内での条件分岐も考慮する必要があることが分かる．

6.　証明の負担を減らす仕様記述法の検討

3〜5 章に示したとおり，3 種類いずれのモデルにおいても生成された証明責務をすべて証明することができた．しかしながら，仕様が複雑になるにつれ，証明責務の数が膨大となっている．連動装置などさらに複雑なシステムを考えた場合，やみくもにモデル化を行っても証明しきれない可能性が高くなる．そこで，証明の負担を減らすためにどうすればよいかを，前章までに記述したモデルを対象に検討を行った．まずは単線自動閉そく式での検討をもとに，どのような仕様記述により証明の負担が減るかを検討し，その知見をもとに，さらに多数の証明責務が生成されていた特殊自動閉そく式の仕様記述の改善を図った．

6.1　仕様記述と証明責務の数との関係

B の抽象機械において，operation に対する証明責務は，I を不変条件，S を一般化代入としたとき，

$$I \Rightarrow [S]I \tag{1}$$

と記述できる [3]．ここで $[S]I$ とは S によって I を変換することを意味する．たとえば S を $x := E$ とし，I を $x = y$ とすると $[S]I$ は $E = y$ となる．ここで $I = I_1 \wedge I_2 \wedge \cdots \wedge I_n$ とすると，今回使用した証明責務生成器は

$$I_1 \wedge I_2 \wedge \cdots \wedge I_n \Rightarrow [S]I_k (k = 1, \cdots n) \tag{2}$$

という n 個の証明責務を分割して生成する．

なお，I_k が $P \Rightarrow Q$ のような形をしていた場合，

$$I_1 \wedge I_2 \wedge \cdots \wedge I_n \wedge [S]P \Rightarrow [S]Q \tag{3}$$

というように P を左辺に移動できる．ここで $Q = Q_1 \wedge Q_2 \wedge \cdots \wedge Q_m$ と書ける場合，これらも分割される（ただし，自明と判断された場合は除く）ので，証明責務の数を見積もる場合，式 (2) における n の値は，個々の節に含まれる含意の右辺についてもさらに分解したうえで決定することとなる．

実際の証明責務を分析すると，今回の証明責務生成器では以下の条件によりさらに分割されていることが分かった．

- S の内部で条件分岐が行われた場合，たとえば IF P THEN S_1 ELSE S_2 END という記述がなされると

$$I_1 \wedge I_2 \wedge \cdots \wedge I_n \wedge P \Rightarrow [S_1]I_k$$
$$I_1 \wedge I_2 \wedge \cdots \wedge I_n \wedge \neg P \Rightarrow [S_2]I_k \tag{4}$$

表 7　条件分岐と証明責務の数（単線自動閉そく式・抜粋）
Table 7　The numbers of conditional branches and proof obligations for some operations of automatic block systems.

operation	自明	証明責務	計	分岐	M_i 最大	N_{PO}
OriginDownSet	102	197	299	4	1	392
OriginUpSet	52	53	105	4	0	212
DTrackCDown	345	204	549	8	2	896
DTrackCUp	142	147	289	3	1	280
OriginRouteSet	41	0	41	1	0	56
OriginRoute-Release	52	53	105	2	2	162

※不変条件における結合節の数 (N_1) = 14
※ include した機械における不変条件の論理和の数 (N_2) = 2

のように，証明責務が 2 つに分割される．$I_1, \cdots, I_n$ のすべてで分割されるから，IF を 1 回用いるごとにおよそ 2 倍の証明責務が生成される．

- `INCLUDE` 文で他の抽象機械を使用した場合，include する抽象機械の不変条件が左辺に加わるが，これを $J = P \vee Q$ とすると，

$$I_1 \wedge I_2 \wedge \cdots \wedge I_n \wedge P \Rightarrow [S]I_k$$
$$I_1 \wedge I_2 \wedge \cdots \wedge I_n \wedge Q \Rightarrow [S]I_k \quad (5)$$

の 2 つの証明責務が生成される．したがって include した抽象機械の不変条件などで論理和が 1 回使われるごとに証明責務が 2 倍となる．
- 非決定的選択（ANY X WHERE P THEN S END）を用いたときに $P = P_1 \vee P_2$ のように書かれた場合，include される抽象機械に論理和を使用したときと同様に，証明責務が分割される．

これらの検討から各 operation（初期化を含む）における証明責務の数は以下のように見積もることができる．

$$N_{PO} = N_1 \cdot 2^{N_2} \cdot \sum_i 2^{M_i} \quad (6)$$

N_1　不変条件における結合節の数
N_2　include した機械の不変条件内での論理和の数
M_i　operation 内の条件分岐の枝 i の中での論理和の数

単線自動閉そく式について，この 3 つを考慮して証明責務の数を見積もった．結果を**表 7** に示す．ここでは枝ごとの M_i を示す代わりに目安として M_i の最大値を示した．また証明責務生成器が自明と判断したものを含めた生成数を同時に示した．一部で数が見積りより多いが，おおよその生成数を見積もれることが分かった．条件分岐の多い operation において証明責務の生成数が多いことが分かる．

ここで，不変条件における論理和の数が 2 となっていることが，証明責務の数を増やしていること，さらにこの不変条件を一般化代入の条件に使用したことで証明責務の数が増えたことが読み取れる．そこで，論理和の使用を減らすことを考える．具体的には $(A \vee B) \Leftrightarrow (\neg A \Rightarrow B)$ であるから，これによって書き換える．その結果を**表 8** に示す．全体システムの証明責務が 1/4〜1/6 になった．このように証明責務の数が条件分岐の枝の数と不変条件内の論理和に大きく影響されることが分かった．

表 8　論理和の除去による証明責務の数の違い（単線自動閉そく式）
Table 8　Difference in the number of proof obligations when logical disjunctions are removed.

(a) 除去前

module	自明	証明責務	自動証明	対話的証明	User Pass
TrackCircuit	107	21	18	3	1
(refinement)	95	134	71	63	51
Whole System	1,901	1,716	1,586	130	53
(implementation)	N/A	6,914	6,208	706	54

(b) 除去後

module	自明	証明責務	自動証明	対話的証明	User Pass
TrackCircuit	79	13	10	3	1
(refinement)	89	128	64	64	48
Whole System	374	399	327	72	53
(implementation)	2,893	1,036	811	225	42

なお，全体システムの実装部分も大きく減っているが，これは実装部であっても外部モジュールについては抽象機械の表現を使用するためである．したがって，外部から利用されるモジュールにおいては，論理和を不変条件に使用する回数を積極的に減らすとよいことが分かる．

6.2　User Pass を減らすための記述法の検討

証明責務の数は条件分岐と不変条件内の論理和の数が影響しており，これを少し検討し直すだけでも証明責務が劇的に減る場合があることを示した．ただし，改めて表 8 を見ると，証明責務の数の減少と比べて User Pass の数はほとんど変わりない．一方で大量の証明責務が生成されているにもかかわらず，大部分が自明と判定される例もある．

Atelier B の処理はモジュールごとの証明責務が増えると，証明責務の整理に時間を要し，処理能力が低下する．たとえば自動証明の処理を 1 度に実行できるのは 3,000 程度であり，証明責務が 10,000 を超えると生成処理にも支障をきたしてくる．また，証明責務が増えると，証明しようとする内容の見通しが悪くなるため，証明責務を減らす検討は必要である．しかし実質的な手間は User Pass の数に比例するので，証明の手間を減らすには別の検討が必要である．

証明の手間を減らすためにはどうすべきかを考察する．実際に証明すべき式は，前節で示したとおり，$I \Rightarrow [S]I_k$ であって，不変条件の各項を一般化代入で変換したものである．それが不変条件の一部に一致すれば自明となる．ま

た，$S = (P \Rightarrow Q)$ のような場合は $\neg P$ が I に含まれていればよい．こう考えると User Pass を減らすためには，$[S]I_k$ を不変条件 I の一部に極力一致させればよいことが分かる．このためにはいくつかの方法が考えられる．

方法 1　$[S]I_k$ を I の一部に一致させるために，積極的に変換後の変数が満たす制約として不変条件式の全部あるいは一部を与える．それには非決定的選択を用いればよい．この一般化代入は $\forall X.(P \Rightarrow [S]I)$ を意味するが，ここで P に $[S]I$ を加えれば $[S]I \wedge \cdots \Rightarrow [S]I$ とできる．ただし P を満たす値が存在する必要があり，その範囲で条件を加えないと，詳細化段階で証明不能となることに注意が必要である．基本的には一般化代入で変換する変数が関係する不変条件式を加えればよい．なお，外部モジュールの変数を用いて不変条件を記述した場合は，外部モジュールの変数の変更を operation を通じて行う必要があるため，非決定的選択を用いた条件の追加は困難である．

方法 2　不変条件が $P \Rightarrow Q$ と記述できる場合は IF〜THEN〜ELSE などの条件式に P を加える．そう考えると，基本的には条件分岐を使う場合は条件式を極力不変条件の含意の左辺に合わせる方がよい．

方法 3　表明（assertion）を与える．
条件を表明として与える方法がある．表明とは不変条件から導出可能な式を証明の補題として宣言するもので，仕様中に ASSERTIONS と宣言して記述する．一般化代入内で補題を与える表明付き代入というものもある．IF 条件や非決定的選択とあわせて使用ができ，IF 条件の条件式から補題を作ることも可能である．

ここで，補題を与えることにより証明が成功する可能性の高い書き換え規則例を示す．

```
(tc <+ {route |-> detect})(ct) = tc(ct)
```

これは特殊自動閉そく式で使用した記述で，列車検知状態を示す写像 `tc` に `route` から `detect` への対応を上書きするものである．`ct : dom(tc)` かつ `ct /: dom({route |-> detect})`（`/:` は左辺が右辺に属さないの意）であれば成り立つ命題であるが，自動ではこの書き換えはなされない．不変条件に `tc(ct)` という記述がある場合に `tc := tc <+ {route |-> detect}` という代入を行うと，この不変条件の `tc` を `tc <+ {route |-> detect}` に書き換えたものが証明責務の一部となるが，この部分も書き換えられない．

そこで，これを表明として与えることにより，書き換えが可能になる．これにより証明される証明責務が多く存在する．表明に記述された命題の証明は別途必要であるが，各証明責務で毎回証明するのではなく，表明を記述した箇所で証明すればよい．そのほかにも `xx : {ot, ct, ...}` というような集合を使用した関係式や，自然数を使った数式にも表明の使用が有効と考えられる．

今回の実施例について，仕様記述と実際の User Pass の中身を再検討すると，仕様を以下のとおりに記述していた．

- 個別モジュールにおいては，非決定的選択で演算を記述し，不変条件そのもの，あるいはその一部を条件に加えていた．これは不変条件を満たすものを明示的に与える意図で行っているが，結果的に方法 1 を実践していたことになる．表 3 によれば，個別モジュールでの証明が少ないため，方法 1 を使用すると，少なくとも抽象機械の証明の手間は省けることが分かる．
- 外部モジュールを取り込んで構成した全体モジュールで，証明責務が多い．ここでは operation 呼び出しを用いて演算を記述すると，非決定的選択を使用できないため，その外部 operation 内の演算において値が書き換わるかどうかに着目して条件分岐を記述していた．
- 外部モジュールを取り込んだ場合に，対話証明を行った証明責務の多くは，条件分岐における条件と不変条件内の含意の左辺の一致が判定できないために，自動証明できないものが多かった．したがってこれらの一致が判定できれば自動証明できる数を増やすことができる．

このような結果から，仕様記述の改善には前述の方法 1〜3 をさらに進めればよいと判断される．外部 operation が関連する変数については，その外部モジュール内での条件分岐に着目し，条件分岐を記述する．条件分岐と表現が一致させられない場合には，分岐前後に ASSERT 文を挿入することで表現を一致させるようにする．そして，ひととおり仕様記述した後は，いったんツールにより証明責務の生成状況や，自動証明で残った証明責務を確認することにより，足りない条件式をチェックし，追加していく．

単線自動閉そく式について，軌道回路モジュールでは，対話的に証明が必要な命題のみが残っていて，改良の余地が少ないため，駅装置のモジュールを中心に修正を行った．結果を**表 9** に示す．駅装置モジュールの証明責務は 51 から 31 にまで減少し，対話的証明は 22 から 0，つまりすべて自動で証明されるに至った．全体モジュールについてもあわせて修正を行った，証明責務の数は 399 から 340 に減少し，対話的証明は 72 から 28 にまで減少させることができた．

一方でいずれも自明とされる証明責務の数は増えている．条件分岐を使うことは，場合分けを仕様記述において明示することにより証明を容易にしている面があることを意味しており，条件分岐の使用などによる証明責務の増大を気にしすぎる必要がないことを示している．

6.3　詳細化（実装）段階での証明責務の数と戦略

抽象機械をいかに証明の手間を省きつつ，詳細化し，実装

表 9　対話的証明の数を減らす改良の適用結果（単線自動閉そく式）
Table 9　Effect of proposed method to reduce the proofs for the specificaiton of normal automatic block systems.

(a) 改良前

モジュール	自明	証明責務	自動証明	対話的証明	User Pass
Station	108	51	29	22	14
TrackCircuit	79	13	10	3	1
Whole System	374	399	327	72	53
(implementation)	2,893	1,036	811	225	42

(b) 改良後

モジュール	自明	証明責務	自動証明	対話的証明	User Pass
Station	193	31	31	0	-
TrackCircuit	※改良前に同じ（修正なし）				
Whole System	764	340	312	28	20
(implementation)	2,893	1,212	1,085	127	18

に変換するかについては，多くの議論がある．特に自動で詳細化を行う技法も研究されており，文献 [5] の事例では自動詳細化を併用していることが紹介されているほか，現在では Atelier B にも自動詳細化を行う BART（B Automatic Refinement Tool）と呼ばれるツールが付属する．このツールは，詳細化のルールとパターンマッチングを行い，それに従って詳細化後のモジュールを自動生成するものである．

抽象機械の表現を詳細化する技法については，既存の議論に譲るが，ここでは，抽象機械で非決定的選択を用いて記述したものをいかに決定的なものとしていくかという点に絞って議論を行う．

詳細化段階の証明の負担を減らすためには，抽象機械とは別の戦略が必要である．最終的には演算を決定的な記述にする必要性から，非決定的選択を用いて条件を仮定に追加するという戦略はとれない．明示的な条件分岐を使用していく必要がある．

ここで詳細化段階の証明責務は

$$I \Rightarrow [S_n]\neg[S_{n-1}]\neg I_n \tag{7}$$

と記述される [3]．ここで I および I_n は詳細化後のモジュールで宣言される不変条件を示しており，実際には詳細化前の変数と詳細化後の変数を結び付ける関係式である．

ここで S_n で IF 条件を使用すると証明責務が 2 倍となる．また，S_{n-1} においても IF を使用していた場合，さらに証明責務が 2 倍となる．しかし，条件が一致していれば，抽象機械のときと同じ理屈で自明な証明責務が増えるから，可能な範囲で条件分岐を詳細化前後で一致するようにすることで，証明責務の生成数を抑えることが可能となる．

例として，全体システムに記述されている DTrackCUp という operation 内で使用されている次の文を考える．

```
IF ran(DN.existence <+ {xx |-> nondetect})
   = {nondetect} THEN
```

実装段階ではこのままではプログラムに変換できないため，

```
DN.existence :=
  DN.existence <+ {xx |-> nondetect};
IF detect : ran(DN.existence)
THEN ex := detect ELSE ex := nondetect END;
IF ex = nondetect THEN ...
```

と置き換えられている（実際には operation 呼び出しなどを行っているが，ここではそれを展開している）．その結果，実装段階最後の行の IF 文の条件は以下と等価となる．

```
not(detect :
  ran(DN.existence <+ {xx |-> nondetect}))
```

しかし，これでは抽象機械の IF 文の条件と等価とはただちに判定できないので，抽象機械，実装の両方の条件で場合分けされてしまう．そこで実装段階の 3，4 行目を

```
IF ran(DN.existence) = {nondetect}
THEN ex := nondetect ELSE ex := detect END;
```

と書き換える．そうするとこの DTrackCUp の証明責務は 229（そのほかに自明なもの 428）であったものが，142（自明なもの 168）となった．ほかにもこれにより証明責務が減ったものがあるため，この 1 文のわずかな変更だけで表 3 の証明責務が 1,036（自明なもの 2,893）から 822（自明なもの 2,491）に減った（これらの数字は表にはまとめていない）．

このように詳細化段階においては，条件分岐を一致させることで，証明責務を減らすことが可能である．表現で一致できない場合には前節で説明したように表明を挿入することで意図した条件分岐に合わせる方法が有効である．

6.4　特殊自動閉そく式への仕様記述改善法適用結果

特殊自動閉そく式の仕様に対し，単線自動閉そく式をもとに検討した 6.2 節の改善法を適用した．駅装置のモジュールにおいては，まず，不変条件を整理し，項ごとにどの変数が影響されるかを明らかにしたうえで，個別の演算の修正を行った．詳細化段階については，条件分岐を極力抽象機械と合わせるとともに，条件分岐ごとに表現が異なるものについて表明を加えた．全体モジュールについては，極力分岐を外部呼び出しの内部条件に合わせ，適宜表明を挿入していった．実装段階については大幅な修正は行わず，条件分岐箇所における表明の挿入程度にとどめた．

結果を**表 10** に示す．抽象機械について見てみると駅装置の証明責務は改良前の 27%，全体システムでは 21%となった．対話的証明を行った証明責務は抽象機械において駅装置で 41，全体システムで 14 であるが，ほとんどは表明として与えた補題の証明であり，自動証明するのは困難と考えられる．実装段階については抽象機械での条件分岐の削減にともなって生成数が減っているが，さらに表明を加えることにより証明責務の生成数を減らすことができた．

表 10　対話的証明の数を減らす改良の適用結果（特殊自動閉そく式）
Table 10　Effect of proposed method to reduce the proofs for the specification of semi-automatic block systems.

(a) 改良前

module	証明責務	対話的証明	User Pass	size
Station	4,979	494	128	58.6 kb
(implementation)	6,341	955	136	270.5 kb
Whole System	17,502	402	61	7.4 kb
(implementation)	8,677	33	12	3.3 kb

(b) 改良後

module	証明責務	対話的証明	User Pass	size
Station	1,353(27%)	41	40(31%)	9.2 kb
(implementation)	3,217(51%)	28	23(17%)	5.1 kb
Whole System	3,759(21%)	30	14(23%)	1.8 kb
(implementation)	5,270(61%)	21	12(100%)	2.6 kb

ただし割合としては抽象機械ほどは減少はしていない．

User Pass の数については，抽象機械では駅装置で改良前の 31%，全体システムで 23%となっている．User Pass の保存ファイル（テキストのコマンド列）のサイズは 58.6 kbyte から 9.2 kbyte と 1/6 以下となり，1 証明あたりの入力テキストでは半分となっている．これは単に証明の数が減っただけではなく，補題の導入により少ないステップ数で証明ができたことを示している．また実装段階については，駅装置では元の 17%であり，保存ファイルのサイズに至っては 270.5 kbyte から 5.1 kbyte と劇的に減少している．全体システムでは数は変わっていないが，サイズが小さくなっており，ステップ数を減少させることができたことが明らかとなった．

実際には表明を追加するということは証明の作業の一部を仕様に記述する形で実施しているということになる．しかし，いったん導入してしまえば，再証明する際にも少ない手間で証明できるし，完全に自動証明されればより再利用性が高まるといえる．

6.5　まとめ

単線区間向け閉そく装置の B 仕様記述について，証明責務の生成状況を観察し，証明責務の生成規則に立ち戻って考察を加えることにより，どのように仕様記述を行えば証明の負担が軽減するかを検討し，実際に改善を得ることができた．本検討で得られた知見は以下のとおりである．

- 不変条件内での論理和の使用，非決定的代入における論理和の使用により証明責務の数が増える．したがって生成数が多い場合には，これらの表現を少なくすることで証明責務の生成を減らすことができる．また条件分岐の使用は証明責務の数を多くする傾向にある．
- 不変条件に用いる含意の左辺と条件分岐の条件の表現を一致させることで，証明責務が自動証明される可能性が高くなる．また非決定的選択を用いて一般化代入適用時の条件に不変条件を取り込むことで自動証明される証明責務を増やすことができる．
- 論理式の種類によっては，自動での書き換えができず，一般化代入で変換した不変条件が，不変条件式を満たすかどうかを判定できないが，これには表明が利用できる．これにより，証明の作業を部分集約することが可能で，負担軽減につながる．

この結果はツールに依存する部分も多く，一般性を持つと結論するのにはまだ十分でないが，今回得られた知見を参考とすることはできると考えている．また，本報告での検討対象は，元々がリレー論理で記述できるものであり，列挙型に属する変数が主となっている．これに対し自然数を用いた数式や集合演算を多用した場合などで本報告における知見を適用した場合，異なる傾向が見られる可能性はあるが，これらの検討についても今後の課題である．

7.　関連研究

国内において，実用システムに形式手法を使用した事例としては VDM を使用した IC チップの設計の例 [11] が有名だが，この例は形式記述によるあいまいさの排除と VDM のラピッドプロトタイピングを活用した事例である．また，モデル検査法を利用した事例が多く見られる．鉄道でもモデル検査の適用事例 [12] が報告されている．

一方，証明責務を生成して，定理証明により検証を実施する事例はアルゴリズムやプロトコルなどに多く，組み込みシステムなどへの応用にはあまり見られない．文献 [13] ではモニタリングシステムを証明を使って検証しているが，ここでは B をシステム開発向けとした Event-B を使用している．

これに対し，海外では B メソッドの鉄道への適用事例として，冒頭で紹介したパリ地下鉄 14 号線（Météor）[4] や空港シャトル（VAL）への適用 [5] が有名であり，単に鉄道分野への適用ということにとどまらず B メソッドの大規模システムへの適用例として紹介されている．ただし，文献 [4] については，システム開発報告の色彩が強く，プロジェクトの進め方については詳しいが，具体的なモデル化手法には触れていない．文献 [5] においてはモジュールの構成の考え方やモデルの記法について言及されていて，B を使ううえでの示唆を含んでいる．しかしながら本報告の特殊自動閉そく式は日本独自のものであり，列車検知が連続的でなく，1 つの列車検知デバイスの動作による状態遷移が列車位置に関する内部状態によって複雑な影響を受けるところが異なる．

また文献 [4], [5] では，自動証明率を上げるために推論規則を追加する手法を採用しているが，本報告では仕様記述の改良による新規のアプローチをとっている．推論規則追加後の自動証明率は文献 [4] では 92%，文献 [5] では 97%と

報告されているが，本報告における改良後の自動証明率は単線自動閉そく式で 90%，特殊自動閉そく式で 99%である．対象の違いや，使用ツールのバージョンの違いを考慮すると単純に自動証明率の比較だけで優劣は語れないが，仕様記述の改良だけで推論規則追加と同程度の自動証明率を実現できることを示せたことには意義があると考える．

その他の B メソッドのアプリケーションとしてはスマートカードへの適用事例 [14] などがあるが，これらの文献でも証明を考慮した仕様記述法への言及はない．

B メソッドにおける証明の困難さへの対応として，別のツールの使用が報告されている [15]．この文献では実装される定数値が不変条件を満たすかどうかを ProB [16] と呼ばれるツールを使って検査している．これらのツールはモデル化や実装の検証で補助的に使用するには有効と考えられるが，定理証明を必要とする部分は依然として残る．

また，鉄道における B メソッド以外の形式検証事例としては SAT ベースの検証事例 [17] やモデル検査の適用事例 [18] などがある．これらは連動装置において具体的な結線をもとに検証を行っており，一定の成果を得ていることが報告されているが，装置一般の安全に関する要求を考える場合には定理証明は依然として有効であり，これらの検討を進めることには一定の意義があると考える．

8. おわりに

鉄道の単線区間向けの閉そく装置の B メソッドによるモデル化と検証を行い，信号設備への形式手法の適用を目指した．さらに，仕様記述の違いによる証明責務の生成数の違いや自動証明される割合の違いを手がかりに，記述したモデルについて証明の負担を減らすための仕様記述改良の検討を行った．条件分岐や論理和の使用により証明責務が増えることが分かったが，実際の証明の手間としては，いかに個々の演算において不変条件と一致した式を記述できるかに依存しており，非決定的代入や表明を利用することで User Pass を減らすことができた．

この知見を他の記述に適用した場合，集合演算や自然数の演算を多用した場合などで，証明の負担に関して違う傾向が見られる可能性があるが，これについては今後の課題である．ただし，本考察は証明責務の生成規則に立ち戻って行っており，傾向に大きな違いはないと考えている．また，本考察は適用分野を限定していないため，ここで得られた知見は，鉄道以外の分野にも適用可能であると考えられる．今後はこの知見をもとに他のシステムの記述を試みるだけでなく，モデル検査などの様々な検証手法の適用についても試みることで，鉄道信号システムに用いられるソフトウェアの安全性向上に寄与したいと考えている．

謝辞　本報告をまとめるにあたり，貴重なご意見をいただいた九州大学の荒木啓二郎先生に，この場を借りて感謝申し上げます．

参考文献

[1] 寺田夏樹：鉄道信号システムへのフォーマルメソッド適用，鉄道総研報告，Vol.16, No.7, pp.33–38 (2002).
[2] 寺田夏樹：段階的詳細化に基づく鉄道信号へのフォーマルメソッド適用法，鉄道総研報告，Vol.21, No.11, pp.41–46 (2007).
[3] Abrial, J.-R.: *The B-Book: Assigning programs to meanings*, Cambridge University Press (1998).
[4] Behm, P., Benoit, P., Faivre, A. and Meynadier, J.-M.: Météor: A Successful Application of B in a Large Project, *Proc. FM '99*, LNCS 1708, Vol.I, pp.369–387 (1999).
[5] Badeau, F. and Amelot, A.: Using B as a High Level Programming Language in an Industrial Project: Roissy VAL, *Proc. ZB2005*, LNCS 3455, pp.334–354 (2005).
[6] 継電連動装置標準結線図（JR グループ監修），日本鉄道電気技術協会 (1987).
[7] 継電連動装置（CTC 関連及び ARC）標準結線図（改訂版），日本鉄道電気技術協会 (2000).
[8] Potter, B., Sinclair, J. and Till, D.: *An Introduction to Formal Specification and Z*, Prentice Hall (1991).
[9] Larsen, P.G. and Fitzgerald, J.: *Modelling Systems – Practical Tools and Techniques in Software Development*, Cambridge University Press (1998).
[10] Atelier B, Clearsy (online), available from ⟨http://www.atelierb.eu⟩ (accessed 2013-06-10).
[11] 栗田太郎：フォーマルメソッドの新潮流：PartII：産業界への応用：3. 携帯電話組込み用モバイル FeliCa IC チップ開発における形式仕様記述手法の適用，情報処理学会誌，Vol.49, No.5, pp.16–23 (2008).
[12] 川村　正，藤井英明，土田勝紀，高橋和子：鉄道信号システムの連動装置の形式的検証向けモデル化と検証環境構築，電子情報通信学会論文誌，Vol.J88-D-I, No.12, pp.1727–1739 (2005).
[13] 佐藤直人，タイソンホアン，デビッドベイジン，來間啓伸：Event-B による列車監視システムのモニタリング要件の検証，情報処理学会論文誌，Vol.54, No.6, pp.1738–1750 (2013).
[14] Gomes, B., Déharbe, D., Moreira, A. and Moraes, K.: Applying the B Method for the Rigorous Development of Smart Card Applications, *Proc. ABZ2010*, LNCS 5977, pp.203–216 (2010).
[15] Romanovsky, A. and Thomas, M. (Eds.): *Industrial Development of System Engineering Methods*, pp.27–43, Springer (2013).
[16] ProB, available from ⟨http://www.stups.uni-duesseldorf.de/ProB/index.php5/Main_Page⟩.
[17] Borälv, A.: Case Study; Formal Verification of a Computerised Railway Interlocking, *Formal Aspects of Computing*, Vol.10, pp.338–360 (1988).
[18] Haxthausen, A.E., Kjær, A.A. and Bliguet, M.L.: Formal Development of a Tool for Automated Modelling and Verification of Relay Interlocking Systems, *Proc. FM2011*, LNCS 6664, pp.118–132 (2011).

寺田 夏樹

1971 年生．1994 年東京大学工学部計数工学科卒業．1996 年同大学大学院修士課程修了．同年財団法人（現在，公益財団法人）鉄道総合技術研究所入社．ATC，軌道回路をはじめとする鉄道信号システム，およびその形式手法適用に関する研究に従事．電気学会，計測自動制御学会各会員．