Vol.2019-SE-203 No.6
2019/12/12

Event-B に基づいた鉄道システムの実践的な形式化と検証

太田十字光 [1]　青木利晃 [2]

概要：地方の鉄道路線で一般的な単線路線の運行計画が危険を含むかどうかの判定を支援する形式化に基づいた手法を提示する．提示手法は単線路線一般で共通する性質を抽出して路線のモデルとそれを元に記述するスケジュール、そして安全性確保のため安全制約からなる参照モデルであり Event-B で記述する．参照モデル上に検証したい具体的な路線を記述して危険が含まれるかどうか検証する．具象路線を含めた参照モデル全体の証明責務を証明することで検証対象路線に危険が含まれないことを言う．提示手法は具体的な路線を個別に形式化するより少ないコストで実現できることを示し、コスト的な問題から形式化の導入が遅れている地方路線への形式手法導入に貢献することを示す．

キーワード：形式化，運行管理，スケジューリング，経路，駅，安全性制約，Event-B[**]

Practical formalization and verification of railway system based on Event-B

TOJIMITSU OOTA[†1]　TOSHIAKI AOKI[†2]

Abstract: This paper presents a formalization-based method that supports the judgment of whether or not a general single-track operation plan on a local railway line is dangerous. The presentation method is a reference model consisting of a route model, a schedule to be described based on it, and a safety constraint to ensure safety, and is described in Event-B. Describe the specific route you want to verify on the reference model and verify whether it contains danger. By proving the proof duty of the entire reference model including concrete routes, it means that the routes to be verified do not contain danger. It is shown that the proposed method can be realized with less cost than formalizing individual routes, and it contributes to the introduction of formal methods to local routes where the introduction of formalization is delayed due to cost problems. [**]

Keywords: Formalizaion, Transport controlling, scheduling, route, station, safety constraint, Event-B [**]

1. はじめに

我々は鉄道システムを 3 つの階層でとらえる．３つの階層とは連動制御、進路制御，スケジューリングである．物理層に近い連動制御は閉塞と転轍機を制御する．進路制御は列車が進む方向を制御する．スケジューリングは列車群の走行計画を管理する.安全の観点から３つの層を考える．最初に連動制御を考える．連動制御は閉塞（１つの列車を収容する十分な長さを持つ軌道の列）に２つ以上の列車を進入させないことで衝突を回避する．次に、進路制御は列車が進むべき方向に進ませることで衝突や追突の危険性を低減させる．スケジューリングは列車の走行計画の妥当性を保証することで走行計画に起因する危険を低減させる．

新規に敷設する路線ではこの 3 つの階層を統合して安全性を高める試みが可能であり、高い安全性を達成するために望ましい方法である．既設の路線では３つの階層が統合されていない場合が多い．これは小規模な改修を繰り返してきた結果、統合化を目指した改修が難しくなっているからである．

安全性を高める有効な手法の一つに形式手法があり様々な分野に適用されてきている．鉄道信号制御システムについても 1990 年代から適用事例が現れ現在も形式手法の適用と改善が続いている．鉄道信号制御システムへの形式手法の適用は 1990 年代の事例では連動制御に対するものが多く、より新しい事例では連動制御以外の層やホームドア制御を含み、検証する階層と領域が増えている．これは安全性を高める点から言って望ましい．

ところで我々が定義した３つの層の最上位であるスケジューリングについては形式手法の適用事例が少ない．特にスケジューリングだけを対象にした形式化の適用事例は見ない．既設システムに対する改修では予算が限られていることが多く３つの層全てに対して高安全性を求める改善を行いにくいことが理由である．そのため連動制御（稀に進路制御を含む場合がある）に対する形式化までの場合が多くなる．

1 北陸先端科学技術大学院大学
Japan Advanced Institute of Science and Technology.
2 北陸先端科学技術大学院大学
Japan Advanced Institute of Science and Technology.

我々の研究はスケジューリングを対象とし、スケジューリングの安全性を高めるために形式手法を適用する．個別のシステムに対する形式化は安全性に大きく貢献するがコストが高く既設路線を高安全性化する場合の問題となる．我々は、対象とする路線を小規模な単線路線に限定し、それらの路線を共通のモデルを使って記述する．このモデルを参照モデルと呼ぶことにし、個別の路線は参照モデル上に記述し安全性の検証が行えるようにする．これによって費用の問題を解決できる．

我々は形式化のための処理系として Event-B を採用した．Event-B はその言語仕様として段階的な詳細化を含み、処理系は機械的証明として証明責務の生成機能と証明器を備えている．モデルに対して生成される証明責務を証明することでモデルの正しさを言い、モデルの正しさを保ったままより詳細なモデルの記述が可能である．この特性は参照モデルの記述に適しているため、我々の研究では Event-B を使用する言語処理系として選択した．

2. 概要

我々の研究はスケジュールの安全性を形式的に記述することを目的にしている．この節では参照モデルを構成する路線のモデル、スケジュール、安全性のための制約について非形式的に示す．路線のモデルは中小規模の単線路線が持つ共通の性質を取り出したモデルであり、スケジュールは運行計画に含まれる個々の列車の走行計画のことである。そして安全性のためにスケジュールが保存すべき制約条件があり、これらから参照モデルは構成される．スケジュールを運行計画に追加したり削除したりすることをスケジューリングと呼ぶ。参照モデルによって、路線のモデルに従ったスケジュールを運行計画に追加したり削除したりした結果としての運行計画が安全性制約を満たすことを確かめることが出来る。

2.1 駅、経路と路線

路線のモデルは駅と経路で構成される。列車が計画上最初に出発する駅から最後に到着する駅までのパスが経路である．最初に出発する駅を始発駅、最後に到着する駅を終着駅と呼ぶと、経路は始発駅から終着駅までの間にある時間ベースで制御される設備が並んだものとみなせる．我々の研究ではこれら時間ベースで制御される設備を駅と呼ぶことにする．従って経路は始発駅から終着駅までの駅の並びとなる．路線は複数の経路で構成される．

2.2 スケジュール

列車が経路上を走行する計画がスケジュールである．列車が駅に到着または出発する時間のことを駅の使用時刻と呼ぶ．言い換えると駅の使用時刻は、駅が列車のために制御される時刻のことである．列車のスケジュールは始発駅から終着駅までの駅の並びと駅に対応する使用時刻の並びのことである．参照モデルにおいてスケジュールは路線のモデルを使って記述される。

現実の時間は連続量であり始まりも終わりもないものとして扱われるが列車のスケジューリングで扱う時間は離散時間であり時間は始まりと終わりを持っている．地方の単線路線では多くの場合始まりから終わりまでの時間範囲は 24 時間、つまり 1 日である．

2.3 安全性のための制約

スケジュールが危険を含まないことをスケジュールが安全であると定義する．最初にスケジュールにおける危険について考える．制御の誤りや操作の誤りに起因する危険はスケジューリングの範囲外であるため、我々の研究ではスケジュール通りに列車が運行した場合に追突または衝突が生じる場合をスケジューリング上の危険と考える．スケジューリング上の危険として追突と衝突（図 1）を扱う．追突と衝突を含まないことをスケジュールが満たすべき制約と定義する．スケジュールが制約を満たせば、スケジュールは危険を含まず安全であると考える．

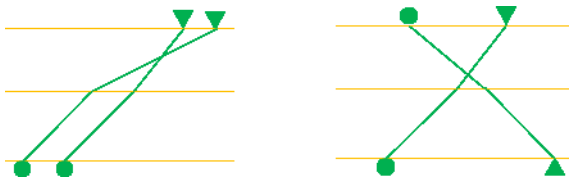

図 1 追突（左）と衝突（右）

2.4 運行計画

ある路線の 1 日のスケジュールを集めたものを、その路線の運行計画とする．従って、運行計画中のスケジュール群がその日の列車の運行となる．路線を構成する駅が有限個であることと、スケジュールが離散時間かつ範囲を持つことから、ある路線の 1 日の運行計画が取り得るスケジュールは有限個である．

3. 参照モデル.

2 節で参照モデルが路線のモデルと、路線のモデルを使って記述するスケジュール、スケジュール群が保存すべき制約条件である安全性制約から構成されることを説明した．本節では、路線のモデル、スケジュール、制約条件を形式的に定義する．本節では抽象モデルによって考え方を示し、4 節で詳細なモデルを提示する．

3.1 経路とスケジュールの形式化

参照モデルを構成するために、経路、スケジュール、運行計画を形式的に定義する．2 節で述べた通り、経路は駅

の列であるが、最初のモデルでは経路を単に集合として定義する．路線が取り得る経路はすべて ROUTE の要素である．

SET : ROUTE

Routes ∈ ROUTE

スケジューリングにおいて列車が始発駅から終着駅まで辿るパスを経路と定義した．物理的な列車は始発駅の前、終着駅の後にも存在するが、それらはスケジューリング上では別の列車であると考える．従って、１つの列車が複数の経路を取ることはなく列車から経路へマッピングを取ることが出来る．本稿では、列車を実際の列車を抽象化したものであるとみなす．運行計画上に存在し得る列車から経路へのマッピングを以下のように定義する．

SET : TRAIN

Trains ∈ TRAIN

trainToRoute∈ TRAIN ⇸ ROUTE (1)

本稿では異なる列車が同じスケジュールを持つことは考えない．スケジュールから列車へのマッピングを以下の通り定義する．

SET : SCHEDULE (2)

スケジュール ∈ SCHDULE

scheduleToTrain ∈ SCHEDULE ⇸ TRAIN

(1)と(2)から SCHEDULE から ROUTE へマップを取ることが出来る．

scheduleToRoute ∈ SCHEDULE ⇸ ROUTE (3)

3.2 スケジュール追加／削除イベントの形式化

参照モデルのスケジューリングについて定義する．1 日の運行計画には１つ以上のスケジュールがあり、１つのスケジュールは１つの列車の運行を計画している．運行計画に１つのスケジュールを追加することは列車の運行が１つ増えることを意味し、運行計画から１つのスケジュールを削除することは列車の運行が１つ減ることを意味している．運行計画に存在するスケジュールの一部を変更することを変更対象のスケジュールを運行計画から削除し、一部が変更されたスケジュールを追加することだと考えれば変更は運行計画に対するスケジュールの追加、削除操作と見なせる．従って、我々の研究ではスケジューリングは運行計画に対するスケジュールの追加・削除操作であると考える．

本稿が提案する参照モデルは運行計画が衝突と追突を含むかどうか検証することを目的としている．構成し得るスケジュールはすべて SCHEDULE 上に存在しており、その時々の状況に応じてその要素を取り出し、運行計画に加えるのだと考える．従って、個々のスケジュールの構成法については扱わない．実際の運用でスケジュールは頻繁に変更されるが、変更操作は削除操作と追加操作の組み合わせであるため本稿では追加操作と削除操作だけを扱う．

スケジュール追加は運行計画にスケジュールを１つ加えることであり、スケジュール削除は運行計画からスケジュールを１つ削除することである．スケジュール追加イベントを ADD_SCHEDULE、スケジュール削除イベントを DELETE_SCHEDULE とする．ADD_SCHEDULE において、運行計画に含まれているスケジュールを追加する意味はないため、イベントのガード条件として、追加するスケジュールが運行計画に含まれないことを指定する．運行計画を Schedules とするとスケジュール追加イベントは以下のように記述できる．

ADD__SCHEDULE :

WHERE grd : schedule ∈ SCHEDULE ∧ schedule ∉ Schedules

THEN act : Schedules ≔ Schedules ∪ {schedule}

DELETE_SCHEDULE において、運行計画に存在しないスケジュールを削除することはできないためイベントのガード条件として、削除するスケジュールが運行計画に存在することを指定する．スケジュール削除イベントは以下のように記述できる．

DELETE_SCHEDULE :

WHERE grd : schedule ∈ Schedules, Schedules ≠ ∅

THEN act : Schedules ≔ Schedules ∖ {schedule}

イベントの発生前、発生中、発生後のいずれの時点でも Schedules は SCHEDULE の要素の集まりでなければならない．これを不変条件として以下のように記述する．

inv : Schedules ⊆ SCHEDULE

3 節では、路線のモデルに従ってスケジュールを記述することを(1)(2)で例示し、スケジュールを運行計画 Schedules に追加／削除するスケジューリングを示した。安全性制約は 4 節で導入する。

4. 参照モデルの詳細化

ここまでで抽象的な参照モデルを記述した．4 節で参照モデルの詳細化を、5 節では 4 節で詳細化する参照モデルを詳細化して具象路線を記述する．参照モデルを構成する経路、スケジュール、スケジュール追加／削除イベントを詳細化することで、安全性の制約を追加することが可能になるため 4 節の最後で安全性制約を追加する．

4.1 経路の詳細化

前節で集合として定義した ROUTE を詳細化する．経路は ROUTE の要素であり駅の列と対応している．経路を構

成する駅の列は変わらず、異なる経路は異なる駅の列を持つ．従って、経路から駅の列へのマッピングをとることが出来る．経路と駅の列の関係を以下のように定義する．

routeToSeqOfStations ∈ ROUTE → (ℕ1 ⇸ STATION) (4)

経路の駅の列で最初の駅が始発駅、最後の駅が終着駅である．経路の駅の列が始発駅から終着駅へ向かって並んでいれば走行可能である．これは駅列内のｎ番目の駅と n+1 番目の駅が隣接していることと同じである．これを以下のように定義する．

isAdjacent ∈ ROUTE → ((STATION × STATION) → BOOL) (5)

(4)と(5)から経路内の時刻に基づく制御可能点である駅を進行方向に沿って並べることが出来る．

4.2 スケジュールとスケジュール追加／削除イベントの詳細化

参照モデルのスケジュールを詳細化する．本稿が提案する参照モデルは運行計画が衝突または追突を含むかどうかを検証するためのものである．衝突と追突の定義から、駅列の一部を共有する経路を持つ 2 つのスケジュールの時刻を検証する必要がある．そのためスケジュールと経路、スケジュールと時刻の関係を定義しなければならない．最初にスケジュールと経路の関係について説明する．列車が始発駅から終着駅まで走行する計画がスケジュールであるから、スケジュールと経路には対応関係がある．運行計画に含まれるスケジュールは、路線に存在する経路のいずれか一つと対応している．これを 3 節の(3)で定義した．(4)で経路と駅の列の関係を定義したが、(3)と合わせて考えると、スケジュールには対応する駅の列がある．スケジュールに対応する駅の列の最初の駅が始発駅であり、最後の駅が終着駅である．これを以下のように定義する。

scheduleToRoute ∈ SCHEDULE → (ℕ1 ⇸ STATION) (6)

スケジュールは列車が始発駅から終着駅まで走る時間に基づく計画であるからスケジュールに対応する経路の各駅に時刻をマッピングすることで、スケジュールを記述できる．これを以下のように記述する．

scheduleToSeqOfTimes ∈ SCHEDULE → (ℕ1 ⇸ TIME) (7)

スケジュールに対応する駅の列の各駅が１つの時刻に対応することを言うためには駅の列と時刻列が対応していることを言う必要がある．そこで、(6)の駅の列の並びと要素数と(7)の時刻列の並びと要素数について関連付ける．そして、(7)として定義したスケジュールに対応した時刻列と、(6)として定義したスケジュールに対応した経路の駅の列では要素数が同じであることが言える．

(6)でスケジュールと経路、(7)でスケジュールと時刻の対応関係が定義出来たので、スケジュールそのものを定義する．始発駅から終着駅まで並ぶ駅の列とそれに対応する時刻列がスケジュールには必要であり、これに計画上存在し得る列車を加えると、スケジュールは列車、始発時刻、終着時刻、駅の列、時刻列に対して１つ存在すると考えることが出来る．それを以下のように定義する．

generateSchedule ∈ TRAIN × ROUTE ×TIME×TIME×(ℕ1 ⇸ STATION) × (ℕ1 ⇸ TIME) ⇸ SCHEDULE (8)

ここまでで、Event-B 記法でスケジュール追加を記述することが出来るようになった．

次に、スケジュール追加／削除イベントを詳細化する．イベントパラメータであるスケジュールを(1)から(7)の定義を使って記述する．追加、削除されるスケジュールは抽象的なスケジュールではなく、

列車、経路、始発時刻、終着時刻、駅の列、時刻列で表される．スケジュール追加はこれらからスケジュールへのマッピングを運行計画に追加することである．ADD_SCHEDULE は以下のように記述できる．

ADD_SCHEDULE :
WHERE
grd1 : train ↦ route ↦ starttime ↦ finishtime ↦ seqOfStations ↦ seqOfTimes ∈ dom(generateSchedule)
grd2 : generateSchedule(train ↦ route ↦ starttime ↦ finishtime ↦ seqOfStations ↦ seqOfTimes) ∉ Schedules
THEN act : Schedules ≔ Schedules ∪ {generateSchedule(train ↦ route ↦ starttime ↦ finishtime ↦ seqOfStations ↦ seqOfTimes)}

4.3 安全性制約の定義

参照モデルに安全性制約を導入する．安全性制約は運行計画上の各スケジュールに追突と衝突がないことを示す制約である．詳細化したスケジュールの定義が含む駅の列と時刻の列を使って安全性制約を記述する．

4.3.1 追突

運行計画に含まれる異なる任意の２つのスケジュール

を考える．それぞれ schedule1, schedule2 として、schedule1 と schedule2 は経路の一部又は全体を共有しているものとする．共有していなければ追突することはない．追突は 1 組の隣接駅間で考えればよく isAdjacent(sta1, sta2)となる．sta1 で schedule1 < schedule2 の関係があり、sta2 で schedule1 < schedule2 であれば計画上の追突は生じない．それを以下のように記述する．i1_1,i1_2 は shedule1 の駅の列のインデックスを i2_1,i2_2 は shedule2 の駅の列のインデックスを表し、i1_1+1 = i1_2, i2_1+1 = i2_2 とする．Schedule1 の駅の列を seqOfTimes1、schedule2 の駅の列を seqOfTimes2 で表すことにする．

inv1:

routeToSeqOfStations(scheduleToRoute(schedule1))(i1_1) = routeToSeqOfStations(scheduleToRoute(schedule2))(i2_1) ∧
routeToSeqOfStations(scheduleToRoute(schedule1))(i1_2) = routeToSeqOfStations(scheduleToRoute(schedule2))(i2_2) ∧
seqOfTimes1(i1_1) < seqOfTimes2(i2_1) ⇒ seqOfTimes1(i1_2) < seqOfTimes2(i2_2)

4.3.2 衝突

追突に続いて衝突の安全性制約を考える．追突ではある sta1 と sta2 の間を同じ方向に移動する列車が検討の対象だった．衝突では異なる方向へ移動する列車が検討の対象である．運行計画に含まれる異なる任意の２つのスケジュールを考える．それぞれ schedule1, schedule2 とする．衝突も 1 組の隣接 Station 間で考えればよく isAdjacent(sta1, sta2)となる．schedule1 は経路の一部に隣接駅 sta1→sta2 を、schedule2 は経路の一部に隣接駅 sta2→sta1 含む．含まなければ軌道を共有しないため衝突することはない．Sta1 で schedule1 < schedule2 の関係がある時、sta2 で schedule1 < schedule2 であれば衝突は生じない．これを以下のように記述する．

inv2:

∀ dia1,dia2,seqOfTimes1,seqOfTimes2,m1,n1,m2,n2·{dia1,dia2} ⊆ Schedules ∧ dia1 ≠ dia2 ∧ {m1,n1,m2,n2} ⊆ ℕ1 ∧ m1 < n1 ∧ m2 < n2 ∧
seqOfTimes1 = scheduleToSeqOfTimes(dia1) ∧ seqOfTimes2 = scheduleToSeqOfTimes(dia2) ∧
1 < card(seqOfTimes1) ∧ {m1,n1} ⊆ dom(seqOfTimes1) ∧
1 < card(seqOfTimes2) ∧ {m2,n2} ⊆ dom(seqOfTimes2) ∧
1 < card(routeToSeqOfStations(scheduleToRoute(dia1))) ∧ {m1,n1} ⊆ dom(routeToSeqOfStations(scheduleToRoute(dia1))) ∧
1 < card(routeToSeqOfStations(scheduleToRoute(dia1))) ∧ {m2,n2} ⊆ dom(routeToSeqOfStations(scheduleToRoute(dia2))) ∧
routeToDirection(scheduleToRoute(dia1)) ≠ routeToDirection(scheduleToRoute(dia2)) ∧
routeToSeqOfStations(scheduleToRoute(dia1))(m1) = routeToSeqOfStations(scheduleToRoute(dia2))(n2) ∧
routeToSeqOfStations(scheduleToRoute(dia1))(n1) = routeToSeqOfStations(scheduleToRoute(dia2))(m2) ∧
seqOfTimes1(m1) < seqOfTimes2(m2)
⇒seqOfTimes1(n1) < seqOfTimes2(m2)

4 節では、詳細化した路線のモデルに従ってスケジュールを記述することを(1)…(7)で例示し、詳細化したスケジュールを運行計画 Schedules に追加／削除するスケジューリングを示した。運行計画 Schedules が安全であることを確かめる安全性制約を不変条件 inv1,inv2 として導入した。

5. 具象路線

本節では具象路線を記述する．経路、スケジュールをモデル化し、スケジュール追加／削除イベントを記述することでスケジューリングをモデル化した．さらにスケジューリングの結果として運行計画に追突と衝突が生じることを安全性制約で防止した．参照モデルは路線とスケジュールの性質を記述したもので具体的なインスタンスを含まない．現実世界のスケジューリングシステムでは、路線の定義をテーブルにインスタンスを記述することで行う場合が多い．具象路線では駅、経路、は実体を持っており、スケジュールは具体的な時刻と関連付いている．この路線を参照モデルの詳細化として記述する．現実の路線に近い検証路線を図 5 に示す．本稿では、図 2 で示す路線をインスタンス化することで記述する．

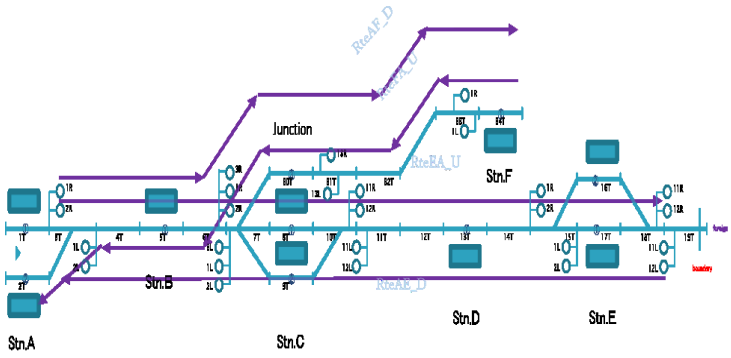

図 2 具象路線

図 2 の路線が持つ駅は StnA から StnF である．経路は StnA から StnE とその逆並び、StnC で F へ分岐できるため StnA から StnF とその逆並びの 4 つの経路がある．始発駅が StnA、終着駅が StnE の経路を RteAE と記述する．参照モデルの

経路と駅の列の関係からRteAEは以下のように記述できる.

{StnA, StnB, StnC, StnD, StnE, StnF} ⊆ STATION
routeToSeqOfStations(RteAE)={ 1 ↦ StnA, 2 ↦ StnB, 3 ↦ StnC, 4 ↦ StnD, 5 ↦ StnE}

経路の駅の列で列の添え字が隣り合う各 Station は隣接するためRteAEのStnAとStnBではisAdajcent(RteAE)(StnA ↦ StnB)は TRUE である．他も同様である.

参照モデルのADD_SCHEDULEとDEL_SCHEDULEを図2 の路線の要素を使って詳細化する．イベントのパラメータは具象路線のスケジュールであり、ガードも具象路線の要素で記述される．これは参照モデルのガードも満たす．運行計画に具象路線のスケジュールを追加する時、不変条件に違反すれば証明責務は証明されない.

ADD_SCHEDULE
REFINES ADD_SCHEDULE
ANY route, starttime, finishtime, seqOfTimes
WHERE route ∈ Routes
{starttime, finishtime} ⊆ TIME
route, startime, finishtime, seqOfTimes ∈ dom(generateSchedule)
generateSchedule(route ↦ starttime ↦ finishtime ↦ seqOfTimes) ∉ Schedules
THEN
Schedules := Schedules ∪ {generateSchedule(route ↦ starttime ↦ finishtime ↦ seqOfTimes)}
END

運行計画からスケジュールを削除するDELETE_SCHEDULEについても同様に記述する.

6. 具象路線の Event-B モデルと検証

6.1 参照モデルの Axiom、theorem

参照モデルと具象路線は５層の CONTEXT と５層の MACHINE で記述した．抽象モデルを 0 層の CONTEXT0 と MACHINE０として記述し、それを詳細化して参照モデルを CONTEXT3 と MACHINE3 として記述した．その上に CONTEXT4 と MACHINE4 として具象路線を記述した．各層の構成は表 5.1 のとおりである．CONSTANT には関数を含む．構成要素間の関係の定義は AXIOM で記述し THEOREM としての記述はなかった．表 1 に CONTEXT の表 2 に MACHINE の統計を示す.

表 1 CONTEXT の統計

	SET	CONSTANT	AXIOM/ THEOREM
CONTEXT0	4	10	22
CONTEXT1	2	11	31
CONTEXT2	0	1	3
CONTEXT3	1	20	22
CONTEXT4	0	24	14
合計	7	66	92

表 2 MACHINE の統計

	VARIABLE	INVARIANT	EVENT
MACHINE0	0	0	1
MACHINE1	0	0	1
MACHINE2	2	2	2
MACHINE3	5	3	2
MACHINE4	5	3	2
合計	12	8	8

6.2 参照モデルと具象モデルの Event-B 仕様記述

表 3 に CONTEXT の仕様記述と MACHINE の仕様記述の規模を示す．CONTEXT の仕様記述は 419 行で、参照モデルの記述が 80%、具象路線の記述が 20%だった. MACHINE の仕様記述は 1385 行で、参照モデルの記述が 84%、具象路線の記述が 16%だった.

表 3 仕様記述の規模

	規模(LOC)	割合(%)
CONTEXT0~3	337	80
CONTEXT4	82	20
小計	419	
MACHINE0~3	1163	84
MACHINE4	222	16
小計	1385	
合計	1804	

6.3 参照モデルと具象路線の証明責務の証明結果

参照モデルとその詳細化である具象路線のモデルに対して、証明責務生成器は 108 個の証明責務を生成した(表 4)．参照モデルに対して 103 証明責務、具象路線に対して 5 証明責務を生成した．すべての証明責務が証明されたが、そのうち 78 証明責務が自動証明、30 証明責務が対話証明により証明された. 参照モデルの証明責務のうち 73 証明責務が自動証明され、具象路線に対する証明責務はすべて自動証明された.

表 4 証明責務の統計

	自動証明	対話証明
参照モデル	73	30
具象路線	5	0
小計	78	30
合計		108

7. 関連研究

鉄道信号システムの高安全性の目標を列車の衝突回避、脱線回避に求める研究は多数あった．Abrial の研究では、列車が軌道上を移動し転轍機でその進路を変えて進行する性質から、同じ軌道内に複数列車が進入すること防止することで列車の衝突、追突を防げることが示唆された．これは列車の在線監視に基づくアプローチである．この研究ではタイミングの考察に基づく脱線回避についても言及されている．

Butler らの研究では、３つの時間制約をタイムクリティカルシステムに導入することで大部分のタイミング特性を形式化できることが示唆された．これらは連動制御層で直接的に衝突、追突、脱線の回避を目標にしている．どちらの研究においてもスケジューリングについては詳しく言及されていない．我々の研究は連動制御層を抽象化したものを基盤として使用し、運行計画に衝突、追突を誘発する危険なスケジュールがないことを示す．運行計画に危険なスケジュールが存在しなければ、考慮すべき危険要因を減らすことが出来るため鉄道信号システム全体の安全性向上に寄与すると考えられる．

近年、ローカルな鉄道信号制御システムを統合する場合の安全性が問題として認識されている．Haxthausen と Peleska は路線とシステムを分離し、TCC と Switch Box コンポーネントを導入することでローカル路線間の差異の影響を受けにくいシステム構築を提案した.我々の研究では、現場設備を直接記述せず、駅として抽象化しているためローカル路線間の接続も、単に路線を延長しているように扱うことが出来る．

8. 結論と課題

8.1 結論

参照モデルと具象路線を階層構造のモデルで記述した．抽象モデルでスケジューリングの概要を定義し、Event-B の詳細化を利用して付加する性質ごとに層を設けることで記述が容易になり詳細化の負荷が軽減した．

表 4 で示した路線全体の証明責務 108 個のうち参照モデルに対するものは 103 個で、実際に検証したい具象路線に対するものは 5 個だった．証明責務全体の 5%が具象路線に対するものであり 95%は参照モデルに対する証明責務だった．検証対象の具象路線が持つ性質は参照モデルに織り込まれており、本稿の具象路線では参照モデルに従ったインスタンス化のみで新たな関係を導入することがなかったため証明が容易になったものと考えられる．具象路線で新たな関係を定義することも考えられるが具象路線の記述の規模は参照モデルに比べて小さいため(表 3)具象路線の証明の負荷は参照モデルに比べて十分に小さくなるものと予想される．このことから参照モデルを構成し証明しておくことで、参照モデル上に構築する具象路線の証明作業が大幅に軽減できることが示せた．これらのことから提示する方法はコストの軽減に有効であると考えられる．一方で、全体の約 80%を占める参照モデルの記述自体はフロムスクラッチであり記述する手間が大きかった．

スケジュールを、路線の構成要素を単純化した駅の使用時刻で記述し、それらのスケジュールが制約を満たすことを示すことで衝突、追突による危険性を検出できるものと考えられる．我々の研究では、単純な構成要素を、段階的に詳細化していくことでより具体的な検証対象となるスケジュールに近づけていくことが出来た．このことから、さまざまな特例がある現実の単線路線について、より現実的で詳細なモデル化が可能であることが期待できる．

8.2 課題

本稿では実質的には出発制御時刻だけをモデル化している．駅間における衝突、追突に加えて、駅構内、駅への進入点、進出点での衝突、追突の検出ができるように、参照モデルを拡張して到着制御時刻のモデル化を行う．単線路線においても重要な駅では待避が可能な場合があるため、番線を導入して待避を含む運行計画において衝突、追突を検出できるようにする．

本稿があつかうスケジューリング上の危険は衝突および追突に限定していたが、列車事故では脱線も深刻な事態につながる．脱線の主な防止層は連動制御であるが、スケジューリング層でも脱線に対する予防措置を講じることが出来るため、駅間の速度勾配を参照モデルに取り込み脱線を扱えるように拡張する．

我々の研究は、駅を所与であるかのごとく扱った．これは本稿が既設路線のスケジューリングに関心を集中しているからである．新設の路線、あるいは既設路線を大幅に拡張する場合、駅を宣言することは簡単ではなく、安全性に大きく影響するが、本稿の範囲外とした．

参考文献

[1] Robert Abo, Laurent Voisin, The Unreasonable Effectiveness of B for Data Validation and Modelling of Railway Systems, LNCS 10598, 2017
[2] Jean-Raymond Abrial, Modeling in Event-B. CAMBRIDGE UNIVERSITY PRESS 2010.

[3] Jean-Raymond Abrial, The B-Book, CAMBRIDGE UNIVERSITY PRESS 1996.
[4] Zakaryae Boudi, Rahma Ben-Ayed, El Miloudi El Koursi, Simon Collart-Dutilleul, Thomas Nolasco, Mohamed Haloua, A CPN/B method transformation framework for railway safety rules formal validation, Eur. Transp. Res. Rev, 2017M. Butler1, D. Dghaym1, T. Fischer, T.S. Hoang1, K. Reichl, C. Snook1,and P. Tummeltshammer, Formal Modelling Techniques for Efficient Development of Railway Control Products, LNCS10598, 2017
[5] Anne E. Haxthausen and Jan Peleska, Formal Development and Verification of a Distributed Railway Control System, IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 26, NO. 8, AUGUST 2000
[6] Thierry Lecomte, Thierry Servat, Guilhem Pouzancre, Formal Methods in Safety-Critical Railway Systems, Symposium on Formal Methods, 2007.
[7] Gianluca Mando, Giovanni Giambene, LTE System Design for Urban Light Rail Transport, LNCS 10598, 2017
[8] Mohammad Reza Sarshogh, Michael Butler, Specification and refinement of discrete timing properties in Event-B, Electronic Communications of the EASST Volume 46, 2011.
[9] Neeraj Kumar Singh, Using Event-B for Critical Device Software Systems. Springer, 2013
[10] Jim Woodcock el al, Formal Methods:Practice and Expericence. ACM computing Surveys,Vol.41, No.4 October 2009
[11] Nancy G. Leveson, Safeware: System Safety and Computers (Series; 19), Addison-Wesley Professional, 1995.
[12] Nancy G. Leveson, Engineering a Safer World: Systems Thinking Applied to Safety (Engineering Systems), The MIT Press, 2016.
[13] John Fitzgerald, Peter Gorm Larsen, Paul Mukherjee, Nico Plat , Marcel Verhoef, Validated Designs for Object-oriented Systems, Springer, 2005.