情報処理学会論文誌 Vol.61 No.2 407–416 (Feb. 2020)

演繹的開発手法と帰納的開発手法の結合に基づく機械学習適用ソフトウェアの形式検証とテスト

來間 啓伸[1,a)] 佐藤 直人[1] 中川 雄一郎[1] 小川 秀人[1]

受付日 2019年4月25日, 採録日 2019年11月7日

概要：計算機システムが実世界と密に連携して動作するためには，論理的に記述・分析できない不確実性に適合するソフトウェアが必要であり，未知の入力値に対して学習データからの推論により出力値を返す機械学習の適用が注目されている．一方，このようなソフトウェアは入力データ空間が定義できず出力値に予測不能性があるため，ソフトウェアの振舞いを確率的にしか把握できない．本稿では，機械学習適用ソフトウェアの高信頼化を目的に，段階的詳細化による演繹的な開発法と機械学習による帰納的な開発法の結合についてテスト・検証の観点から述べ，開発プロセスと制約充足性テスト方法を提案する．我々のアプローチは，演繹的モジュールと帰納的モジュールを分離し，それらをつなぐ部分仕様を設定するとともに，前者については部分仕様が満たされることを前提に論理的な検証を行う一方，後者についてはテストにより部分仕様の充足確率を評価し，論理的な検証結果に確率を付与する．これにより，帰納的に開発した機械学習適用モジュールと演繹的に開発した論理モジュールを，システムの信頼性評価のもとで整合的に結合する．形式手法 Event-B を用いたケーススタディにより，実現可能性を評価した．

キーワード：機械学習工学，ソフトウェア品質評価，ソフトウェア開発プロセス，ニューラルネットワークの頑健性

Formal Verification and Testing of Machine Learning Based Software Using Combination of Deductive and Inductive Development Methods

Hironobu Kuruma[1,a)] Naoto Sato[1] Yuichiroh Nakagawa[1] Hideto Ogawa[1]

Received: April 25, 2019, Accepted: November 7, 2019

Abstract: As computer systems are closely related to the real world, software is required to adopt the uncertain phenomena that cannot be described nor analyzed logically. Machine learning is expected to be a promising way to develop software that accommodate such uncertainty, since it makes outputs induced from the given learning data set even for unknown inputs. The input data space of machine learning based software cannot be defined and it is substantially impossible to predict its outputs. The behavior of such software can be analyzed only in a statistical way. In this paper, we describe a reliable software development method in which the machine learning based inductive development is combined with the refinement based deductive development at a point of testing and formal verification and propose a development process and conformance testing. Out approach is composed of: 1. separating deductive module and inductive module and placing a partial specification connecting them, 2. verifying the deductive module with the assumption that the partial specification is fulfilled, 3. evaluating the probability the inductive module fulfills the partial specification by conformance testing, 4. assigning the probability to the verified properties of the deductive module. We show a case study using Event-B formal method and discuss the feasibility of our approach.

Keywords: machine learning software engineering, software quality evaluation, software development process, robustness of neural networks

© 2020 Information Processing Society of Japan

1. はじめに

情報処理装置が発達し，高度なセンサを使った実世界データの収集とさまざまなアクチュエータを介した実世界への作用が可能になったことで，計算機システムは実世界と密接に結び付いて動作するようになった．ここで，実世界は論理的に記述・分析することが困難な不確実性を含んでおり，ソフトウェアによる論理的な処理のためには不確実性の評価に基づく近似が必要になる [12]．機械学習は，学習データからの推論を土台にして実世界の不確実性に適合するソフトウェアを構築する方法として，注目されている．教師ありの機械学習では，入力値とそれに対する正解出力値を規定する学習データセットを用意して学習処理させることで，ニューラルネットワークを作成する．作成したニューラルネットワークと推論機構（以下では，まとめて機械学習適用モジュールと呼ぶ）を組み込めば，人間による論理的な分析や記述を経ることなくソフトウェアが構成できると期待される．たとえば画像中の交通標識の種別判定では，画像データとその中で認識されるべき標識名の組を多数用意して学習させ，標識の種別を識別できるニューラルネットワークを作成する．天候や時刻，経年変化，周囲の建造物などの影響によって標識の見え方が多様に変化する点で標識の状態には不確実性があり，認識ソフトウェアの処理アルゴリズムを詳細に書き下すことは難しい．論理的に定義できない標識の状態に適合するためには，機械学習の利用は必然といえる．

対象を論理的に分析して仕様を記述し，仕様を満たすソフトウェアを実装する演繹的なソフトウェア開発手法では，仕様に定義されていない入力値に対する出力値や処理の停止性を保証する必要はなかった（design by contract）．これに対し機械学習は帰納的なソフトウェア開発手法であり，限られた数の学習データをもとに，学習データにない未知の入力値に対して学習データから推論した出力値を返すことが求められる．入力値に対して出力値が唯一に決まる確定的ニューラルネットワークは論理的に分析可能なソフトウェアであるが，開発時に未知の入力値に対する出力値を予測することは困難であり，演繹的なソフトウェア開発手法では考えられない種類の誤りが本質的に存在する [13]．このような入力空間が定義できないことから生ずる出力値の予測不能性を，実世界の不確実性と区別して，本稿では機械学習適用モジュールの振舞いの予測不能性と呼ぶ．入力空間が定義できないことが実世界の不確実性に由来する点で，両者は関連する．

機械学習適用モジュールのテスト・検証には従来の方法は適用できず，予測不能性を前提とした手法が求められる [15]．一方，入力空間が定義できないことを前提としたテスト・検証を機械学習適用モジュールを含むソフトウェア全体（以下では，機械学習適用ソフトウェアと呼ぶ）について行うことは，網羅性や効率の点で疑問がある．本稿は，機械学習適用ソフトウェアの高信頼化を目的に，演繹的開発手法と帰納的開発手法の結合についてソフトウェアのテスト・検証の観点から示す．ここでは，構造設計段階で機械学習を適用するモジュールとそうでないモジュールを分離するとともに，その間のインタフェースである部分仕様を設定する．後者については部分仕様が満たされることを前提に演繹的な検証を行う一方，前者については入力値に依存する統計的ゆらぎを考慮したテストにより部分仕様の充足性を評価し，演繹的な検証結果に確率を付与する．これにより，帰納的に開発された機械学習適用モジュールと演繹的に開発されたモジュールを，振舞いの確率的評価に基づいて整合的に結合する．

以下，2 章では演繹的なソフトウェア開発手法として段階的詳細化による構成的アプローチをとりあげ，機械学習を適用した帰納的な開発手法と結合するための本稿のアプローチについて述べる．3 章では，2 章のアプローチを具体化するための機械学習適用ソフトウェア開発プロセスと，リファレンス・モデルを使った機械学習適用モジュールの制約充足性テストを提案する．交通標識認識に基づく速度制御と連動した自動車のドアロックシステムを題材としたケーススタディを 4 章に示し，テストによって評価された機械学習適用モジュールの統計的な制約充足性と，論理的な証明によって検証されたシステムの性質の結合について 5 章で議論する．関連研究を 6 章に示し，7 章で結論を述べる．

2. 演繹的手法と帰納的手法の結合

2.1 演繹的開発手法

演繹的なソフトウェア開発手法では，開発対象についての包括的記述からはじめ，一定の規範にしたがって細部を具体化することで，ソフトウェアを構成する．最初に抽象度の高い仕様を記述し，細部の設計を加えて最終的にプログラムを得る段階的詳細化 [27] は，演繹的なソフトウェア開発手法を代表するものであり，大規模なソフトウェア開発に適応する．さらに，構成的なアプローチ [9] では，プログラムとその正しさの証明を連携させて開発を進める．構成的なアプローチを具体化した形式手法 B メソッド [1] や Event-B [2] では，仕様の整合性と詳細化の正しさを数学的基盤のもとに検証する方法が与えられている．

詳細化の各段階では，上位の記述はその段階で求められる抽象度でソフトウェアの動作を規定すれば十分であり，より詳細な動作の規定は下位の記述に任せることができる．このような設計上未定であることによる不確かさを残

[1] 株式会社日立製作所研究開発グループ
Hitachi, Ltd., Research & Development Group, Yokohama Research Laboratory, Yokohama, Kanagawa 244–0817, Japan
[a)] hironobu.kuruma.zg@hitachi.com

© 2020 Information Processing Society of Japan

しつつ，各抽象度で記述の整合性を検証可能にするしくみとして，非決定性が使われる．この場合下位の記述は，上位の記述に許容される範囲内で，非決定性が小さいか等しくなければならない．

2.2 帰納的開発手法

帰納的なソフトウェア開発手法では，開発対象に関わる具体的な事象の収集からはじめ，それらに共通する法則を発見し一般化することで，ソフトウェアを構成する．学習データセットから機械学習アルゴリズムにしたがってモデルを構築する機械学習の適用は，帰納的なソフトウェア開発手法に位置づけられる [21]．演繹的開発手法では初期段階で開発対象についての包括的理解が求められるが，帰納的開発手法では開発対象について分かっている範囲で学習データセットを用意し開発を進める．このため，開発対象の全体像が把握しにくい場合に有効である反面，学習データセットが開発対象を正しく反映していることの検証は困難であり，一般に開発対象の入力空間と学習データセットが表現する入力空間の間には「ずれ」が存在しうる．

機械学習アルゴリズムは，与えられた学習データに適合するようモデルのパラメタを調整することで，学習を進行させる．学習済みのモデルに求められるのは，学習データとは異なる入力値に対して適切な出力値を返す汎化能力であり，未知の入力値に対する出力値の正解率によって評価する．一般には，学習データセットを学習に用いる訓練データセットと汎化能力評価に用いる試験データセットに分け，モデルには未知である試験データセットについて正解率を計測する．正解率は未知のデータに関わる点で確率的であり，与えられたテストデータについて正しい動作を確立する演繹的開発のテスト・デバッグとは性格が異なる．入力空間が定義できないことから生じる確率的な予測不能性は，帰納的に開発されたソフトウェアの特徴である．

2.3 アプローチ

機械学習を適用した帰納的なソフトウェア開発では，詳細化の開始点になる仕様を明確に記述できず，詳細化に相当する学習の過程を厳密に検証する手段もないので，段階的詳細化を使ったプログラム導出が行えない．しかし，機械学習の適用を実装に向けたモジュール設計上の選択と位置づければ，ソフトウェア全体については要求定義から抽象仕様を記述し，機械学習を適用しないモジュール（以下，論理モジュールと呼ぶ）については段階的詳細化によってプログラムを導出することが可能と考えられる．このとき，機械学習適用モジュールは段階的詳細化の過程から除外されるが，その動作を非決定的に表現することで，論理モジュールについて仕様の整合性と詳細化の正しさを非決定性のもとで検証できる．この過程の概要を，**図 1** に示す．

非決定的に表現された機械学習適用モジュールの仕様は，

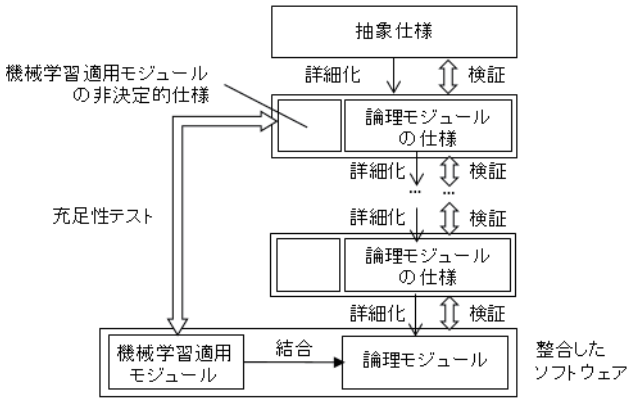

図 1 機械学習適用モジュールの結合
Fig. 1 Integration of ML based module.

他のモジュールとのインタフェースを規定する役割に限定された仕様であることから，以下では部分仕様と呼ぶ．部分仕様は，論理モジュールの仕様の整合性と詳細化の正しさを検証する際の，機械学習適用モジュールに関する前提を与える．一方，部分仕様は詳細化できないので，機械学習適用モジュールが部分仕様を満たすことは，利用状況を想定したテスト入力データを使うユースケーステストにより確認する．ここで，機械学習適用モジュールの予測不能性から，部分仕様の充足性は一般に確率的であり，論理モジュールと結合した際の影響を評価して，全体で整合したソフトウェアを構成する．本稿の演繹的手法と帰納的手法の結合アプローチの要素を，以下にまとめる．

- 抽象仕様段階での機械学習適用モジュールの切り分け
- 非決定性を用いた機械学習適用モジュールの部分仕様の記述
- 部分仕様充足性を前提とした論理モジュールの仕様の段階的詳細化と検証
- ユースケーステストを通じた機械学習適用モジュールの部分仕様充足性の評価
- モジュール結合にともなう部分仕様充足確率のソフトウェア全体への影響の仕様段階での評価

3. 機械学習適用ソフトウェア開発手法の提案

3.1 開発プロセスの概要

2.3 節のアプローチのもと，本稿では**図 2** に示す機械学習適用ソフトウェアの開発プロセスを考える．左下の枠内は機械学習適用モジュールの開発プロセスを表す．機械学習適用モジュールの要求定義段階では，機械学習適用モジュールが何をするのかを明確にする．機械学習適用モジュールの開発は学習データセットにしたがって帰納的に進行するので，達成できる要求は学習データの収集と学習を通じた試行錯誤によって決まり，ユースケーステスト段階からのフィードバックが必要になる．テストの結果，要求が満たされない場合は学習データセットを再作成するか要求自体を修正する必要があるが，後者の場合にはソフト

© 2020 Information Processing Society of Japan

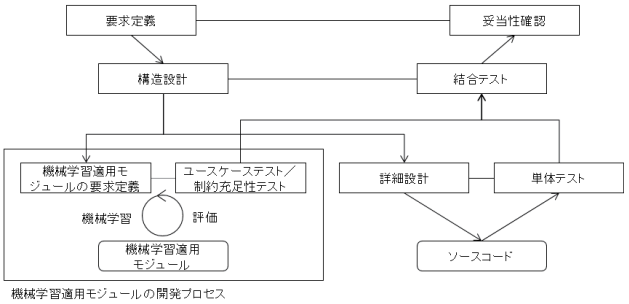

図 2 機械学習適用ソフトウェアの開発プロセス
Fig. 2 Development process of ML based software.

ウェアへの要求の修正にまで波及する可能性がある．このため，機械学習適用モジュールを含むソフトウェアの開発では，要求が満たされ得ることを機械学習適用モジュールの実験的な開発によって概念実証（Proof of Concept）し，その後にソフトウェア開発へと移行する手順がとられることが多い．

ソフトウェア開発では要求定義から構造設計へと進み，開発の最終段階も同様に結合テストから妥当性確認を行う．要求定義段階ではシステムと実世界の関わりを明確にし，実世界の中でソフトウェアが処理する項目とそれ以外を切り分ける．実世界の不確実性の評価も，この段階で行う．これは，学習データセットの十分性の評価基準になる．

構造設計ではソフトウェアをモジュールに分割し，要求を実現するために必要な機能をモジュールに割り当てる．機械学習の適用をモジュールの実装方法の 1 つと位置付けるので，実世界の不確かさに適合することが求められる機能については，概念実証の結果をふまえて，この段階で機械学習適用モジュールとして実装することを決定する．一方，適切な近似のもとで不確かさを除いた機能は，機械学習を適用しない論理モジュールに割り当てる．機械学習適用モジュールの動作をソフトウェアの動作時に監視し不適切な動作を抑制する機能を論理モジュールを付加する，フェイルセーフ化の決定もこの段階で行う．モジュールを整合的に結合して全体をソフトウェアとして機能させるため，機械学習適用モジュールには部分仕様を設定する．

論理モジュールの詳細設計では，論理モジュールの仕様を検証し，段階的に詳細化する．機械学習適用モジュールの部分仕様充足性が確率的であることから，確率的な振舞いが論理モジュール与える影響も見積もる．部分仕様を仲介として機械学習適用モジュールと論理モジュールが連携して動作することは，結合テストで確認する．

3.2 制約充足性テスト

制約充足性テストでは，機械学習適用モジュールが部分仕様を満たすことを，ユースケーステストにより評価する．利用状況を想定したテスト入力データについて十分な数のテストを実行することで，学習データセットが表現する入力空間に限定されず，機械学習適用モジュールが扱うべき入力空間を反映した評価が期待できる．ここで，学習データセットに含まれない入力値に対する出力値は一般に不明であり，入力値と出力値の対であるテストオラクルを構成できない．このような機械学習適用モジュール固有の課題に対して，本稿ではリファレンス・モデルとの照合によるテスト方式を提案する．機械学習適用モジュールの仕様を論理的に厳密に記述することはできないので，リファレンス・モデルが規定するのは入力値に対して許容できる出力値の範囲である．このテスト方式の特徴は，下記である．

- リファレンス・モデルを使った許容出力値範囲の生成
- 一連の入力値に対する出力値の履歴を使った許容出力値範囲の充足性の評価

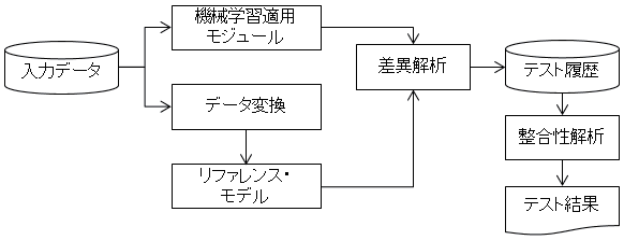

図 3 リファレンス・モデルを使った制約充足性テスト
Fig. 3 Conformance testing with reference model.

リファレンス・モデルを使った制約充足性テストの概要を，**図 3** に示す．このテストでは，機械学習適用モジュールのほかにリファレンス・モデルを用意し，テスト入力データを機械学習適用モジュールとリファレンス・モデルに入力する．このとき，データ変換系がテスト入力データをリファレンス・モデルが処理できる形式に変換する．差異解析では入力データに対する各々の出力を比較し，履歴として蓄積する．一致性判定は，履歴を参照してテストの結果を判定する．本稿では状態遷移に基づくリファレンス・モデルを考え，下記の記法で表記する．ここで，状態名はリファレンス・モデルの状態を表す識別子，条件はデータ変換系によって変換されたテスト入力データおよび機械学習適用モジュールの状態に関する条件式，出力値は条件が満たされたときに許容される機械学習適用モジュールの出力値である．記法中の「+」は 1 回以上，「*」は 0 回以上の繰り返し，「|」は「または」を表す．

<リファレンス・モデル> = <遷移>+
<遷移> = 状態名 ‘‘:=’’ <遷移先>
<遷移先> = 条件 ‘‘{‘‘<許容出力>‘‘}’’ ‘‘->’’ 状態名
| ‘‘->’’ 状態名
<許容出力> = 出力値*

制約充足性テストでは，機械学習適用モジュールの部分仕様に対応し，入力値に対して期待される出力値を機械学習適用モジュールの出力値と照合可能な形式で記述した，リファレンス・モデルを使う．差異解析は，機械学習適用ソフトウェアの出力値とリファレンス・モデルが規定する

© 2020 Information Processing Society of Japan

許容出力を比較し，前者が後者の範囲内にあることを判定して履歴として蓄積する．一致性判定は，履歴に基づいて部分仕様の充足性を統計的に評価する．このような仕組みにより，想定する機械学習適用モジュールの使い方に基づく入力データを与えて，部分仕様の充足性を統計的に評価するのに十分な数のテストを行う．

4. ケーススタディ

4.1 前提と検証目標

ケーススタディでは，交通標識認識に基づく速度制御と連動した自動車のドアロックシステムを題材として，機械学習適用ソフトウェアの開発における形式検証とテストを，機械学習適用モジュールを含まない場合 [19] と比較する．対象とするシステムには，運転者が切り替えることができる auto モードと manual モードがあり，車速が 0 のときは manual モードであるとする．manual モードでは運転者が車速を調整する．一方，auto モードではシステムが路上の交通標識を画像認識し，指定された速度になるよう加速あるいは減速する．ここで，あらかじめ定められた値 s_dv（$s\_dv > 0$）を超える急な加減速が auto モードで起こる場合には，標識の誤判定あるいは想定外の標識配置が存在するとして manual モードに切り替える，フェイルセーフ機能を持つ．なお，簡単のため後進は考慮せず，車速は 0 以上とする．

auto モードで車速がアンロック操作上限値以上になったとき，システムは乗員によるドアのアンロック操作を禁止する．さらに自動ロック下限値（自動ロック下限値 > アンロック操作上限値）以上では，ドアがロックされていなければロック動作を開始する．ただし，ロック動作を開始してから完了するまでの間も車速は変化しうるものとし，たとえばロック完了前に車速が下がった場合にはロック動作を取り消す．また，アンロック操作上限値以上の車速では，manual モードから auto モードへの切り替えを禁止する．

検証目標は，次の 2 項目である．

- auto モードのとき車速が一定値以上であれば必ずドアがロックされていること．
- ドアをロックするのに十分な時間 auto モードが継続すること．

4.2 構造設計

構造設計段階で定めたドアロックシステムのモジュール構成を，**図 4** に示す．交通標識種別判定モジュールは機械学習適用モジュールであり，周期的に交通標識の画像を認識し，現在の車速から次の車速を計算する．前述のように，車速の変化は s_dv 以内であることが期待されので，機械学習適用モジュールの部分仕様は「車速変化は s_dv 以内であること」とする．

一方，センサ，コントローラ，アクチュエータは機械学習を適用しない論理モジュールであり，交通標識種別判定モジュールと同期して周期的に起動される．センサは auto モードのとき機械学習適用モジュールが計算した車速変化を監視し，s_dv を超える場合および車速が 0 になった場合には manual モードに切り替える．コントローラは auto モードのときに車速が自動ロック下限値以上になればドアロックをアクチュエータに指示し，アクチュエータは一定の動作時間の後にドアロックを完了する．

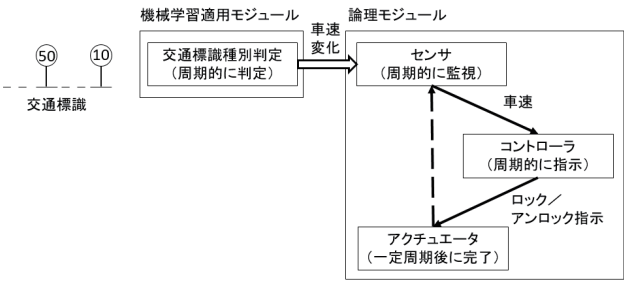

図 4 題材システムのモジュール構成
Fig. 4 Module structure of subject system.

4.3 機械学習適用モジュールの開発と制約充足性テスト

The German Traffic Sign Recognition Benchmark [28] の画像 51,839 枚のうち 39,209 枚を訓練データとして学習させて，43 種別の交通標識を判定する畳み込みニューラルネットワーク（Convolutional Neural Network）[7] を作成した．訓練データを除く 12,630 枚の画像を使って計測した判定の正解率は，97.7%である．

交通標識種別判定モジュールは周期的に標識判定を試み，標識があれば上限速度 v_lim を更新する．

- 速度制限標識：$v\_lim :=$ 標識に指示された速度
- 停止を指示する標識：$v\_lim := 0$
- 徐行を指示する標識：$v\_lim = 10$
- カーブを示す標識：$v\_lim = 20$
- その他の標識および標識がないとき：更新しない

さらに，現在の車速 s_v からの車速変化 dv を出力する．

$$dv := v\_lim - s\_v$$

制約充足性テストでは，機械学習適用モジュールが部分仕様を満たすことを，テスト入力データを与えて検証する．ここで，機械学習適用モジュールの振舞いには予測不能性があるので部分仕様がつねに満たされるとは限らず，テスト入力データに依存した経験的な充足確率が得られる．4.1 節で述べたように，機械学習適用モジュールが満たすべき部分仕様は

$$|dv| \leq s\_dv \tag{1}$$

であり，$s\_dv = 20$ として下記のリファレンス・モデルを作成し，ユースケーステストを行った．ここで，`s_v` は機械学習適用モジュールから得られる現在の車速，中括弧内は

© 2020 Information Processing Society of Japan

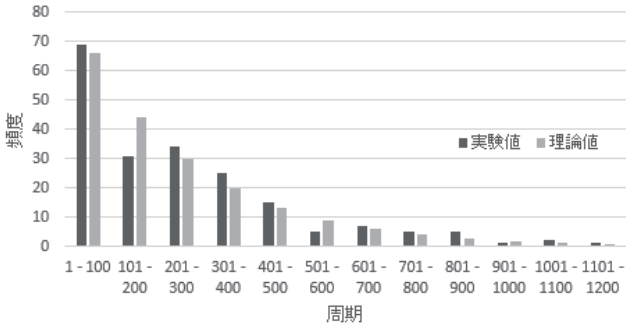

図 5 制約充足性テストから得た失敗頻度
Fig. 5 Frequency of failure obtained by conformance testing.

許容される速度変化を表し，ともに 10 km/h 刻みである．

```
S := s_v = 0 {0, 10, 20} -> S0
S0 :=  -> S
S := s_v = 10 {-10, 0, 10, 20} -> S10
S10 := -> S
...
S := s_v = 120 {-20, -10, 0} -> S120
S120 := -> S
```

正確な充足確率を得るには，テスト入力データは実世界の道路に沿った交通標識の並びであることが望まれるが，ケーススタディはテストの流れを確認することが目的であり，交通標識の画像を適切な順序で並べて作成した模擬データをテスト入力データとした．制約充足性テストでは，学習データからランダムに選択した画像を式 (1) が満たされる順序で並べたテスト入力データを 200 組用意して，各々について誤認識により式 (1) に対応するリファレンス・モデルが満たされなくなるまでの画像数を計測した．画像認識が周期的に行われることから，画像数は認識周期数で置き換えることができる．リファレンス・モデルが満たされなくなった周期数を横軸，満たされなくなった回数を縦軸として，制約充足性テストの結果を**図 5** の「実験値」に示す．グラフ化にあたり，横軸は 100 周期ごとにまとめた．たとえば左端の棒は，1 から 100 周期の間に式 (1) が満たされなくなったケースが，200 回中 69 回あったことを表す．

テストの結果，平均 256 周期後に式 (1) が満たされなくなる結果を得た．各周期の充足確率が一定であると仮定すると，5 章に示す議論から，1 周期あたりの充足確率は $1-1/256=0.996$ と推定できる．制約充足性は機械学習適用モジュールの内部状態や外部環境に依存し，それらは時間経過とともに変化することを考えると，厳密には各周期の充足確率は一定ではない．しかし，このケーススタディでは制約充足性を満たさない原因が誤認識であり，誤認識を引き起こす画像は事実上標識種別によらずランダムに存在することから，経験的には妥当な仮定であると考えられる．この仮定の下で計算した失敗（非充足）頻度の分布を，図 5 の「理論値」に示す．

4.4 論理モジュールの詳細設計と検証

論理モジュールは Event-B の段階的詳細化を使ってモデル化し，機械学習適用モジュールとの間に式 (1) が成り立つならば 4.1 節の 1 番目の検証目標が満たされることを論理的に検証した．Event-B では，自発的に起こるイベントとそれに対するアクションによって対象をモデル化する．イベントは，イベント内で使われる一時変数であるイベントパラメタ x，イベントが起こるための必要条件であるガード条件 G，イベントによる変数値の変化を表すアクション Q から構成され，下記のように表される．

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

ここで，v はアクション前の変数値，v' はアクション後の変数値を表す．x の値はガード条件によって定まり，複数の値が条件を満たす場合には，それらの中から非決定的に選択される．Event-B のモデルの検証では，基本的に不変条件がつねに満たされることを検証する．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．このためには，次の式が証明される必要がある．

$$I(v) \wedge G(x,v) \wedge Q(x,v,v') \Rightarrow I(v')$$

論理モジュールのモデルは，auto/manual モードの切り替えとフェイルセーフ動作にかかわるイベントの導入を除けば，機械学習適用モジュールを考慮しない既出のモデル [19] と同じであり，3 ステップの詳細化により構成した．以下では，変数 *mode* の値が *auto* であれば auto モードを，*manual* であれば manual モードを表す．変数 *phase* はセンサ，コントローラ，アクチュエータの起動順序を制御し，値が *sns* のときセンサモジュールの機能を表すセンサイベントを，*cnt* のときコントローライベントを，*act* のときアクチュエータイベントを活性化する．

機械学習適用モジュールの出力である車速変化 dv は，センサイベントでイベントパラメタを使って非決定的に表現した．次のイベント `sensor_normal` は auto モードで式 (1) が満たされる場合の動作を表し，車速 s_v を更新するとともに *phase* の値を *cnt* に変えてコントローライベントを活性化する[*1]．なお，$\mathbb{Z}$ は整数の集合である．

sensor_normal $\triangleq$
any dv
where $phase = sns \ \wedge\ mode = auto \ \wedge\ dv \in \mathbb{Z} \wedge$
$dv \le s\_dv \ \wedge\ -s\_dv \le dv \ \wedge\ s\_v + dv > 0$
then
$s\_v := s\_v + dv \ ||\ phase := cnt$
end

一方，`sensor_fail_safe` は，auto モードで式 (1) が満

[*1] 読みやすさの観点から，アクションでは述語表記 $v' = v + w$ に代えて代入表記 $v := v + w$ を用いた．

© 2020 Information Processing Society of Japan

たされない場合に manual モードに切り替える，フェイルセーフ動作を表す．

sensor_fail_safe $\triangleq$
any dv
where $phase = sns \ \wedge \ mode = auto \ \wedge \ dv \in \mathbb{Z} \ \wedge$
$(s\_dv < dv \ \vee \ dv < -s\_dv) \ \wedge \ s\_v + dv > 0$
then
$mode := manual \ || \ phase := cnt$
end

4.5 結果

4.4 節のモデル上では，次の不変条件が満たされることを証明できる．

$$mode = auto \wedge (c\_high + (a\_delay + 1) \times s\_dv) \leq s\_v \Rightarrow a\_mode = a\_locked \quad (2)$$

ここで，c_high は自動ロック下限値，a_delay はセンサ周期で測ったアクチュエータ動作時間であり，s_dv と同様に定数である．$a\_mode = a\_locked$ はアクチュエータ動作が完了してドアがロックされた状態を表す．すなわち，auto モードのとき，車速が自動ロック下限値と最大加速時のアクチュエータ動作時間 +1 周期の到達速度の和以上であれば，ドアはつねにロックされている．このように，4.1 節の 1 番目の検証目標が検証できた．

4.3 節のテスト結果から，平均して 255 周期の間は式 (1) が満たされ，イベント `sensor_normal` が起こり続けることが期待できる．この間は auto モードが継続されるので，平均 255 周期の間は式 (2) が満足されることがいえる．いい換えれば，運転者が manual モードに戻さない限り，auto モードに切り替えて n 周期目の車速が

$$c\_high + (a\_delay + 1) \times s\_dv$$

以上であれば，ドアは 0.996^n 以上[*2]の確率でロックされている．ここで，4.3 節で述べたように，1 周期あたりの充足確率 0.996 は各周期の充足確率が一定であることを仮定してテストから得た推定値であり，実世界の不確実性と機械学習適用モジュールの予測不能性にともなうゆらぎがあることに留意する必要がある．より正確な充足確率の推定には，制約充足性テストにおいてゆらぎに関する統計的評価が必要である．したがって，4.1 節の 2 番目の検証目標の検証は確率的であり，車速 0 から最大限加速してドアが自動ロックされるまでの周期数

$$k = \frac{c\_high}{s\_dv} + a\_delay + 1$$

に対し，この間 auto モードが継続される確率はテストの結果から 0.996^k と推定される．

[*2] auto モードが n 周期継続し自動ロックされる場合に加えて，n 周期目以前に自動ロックされた後 manual モードに切り替わる場合や，manual モードでも乗員が手動でドアロックする場合があるため．

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

	行数	証明課題数	自動証明数	自動証明率
本モデル	234	181	179	98.9%
既出モデル	175	134	131	97.8%

機械学習適用モジュールと結合して動作する論理モジュールのモデルと，速度制御と連動しないドアロックシステムの既出のモデル [19] の，記述量と証明課題数の比較を**表 1** に示す．ここで，証明課題数は不変条件が満たされていることを検証するために証明しなければならない式の数，自動証明数は証明課題のうちツールが機械的に証明できた式の数である．前述のように，本モデルと既出のモデルの間にはモデル化と詳細化ならびに検証の過程に大きな違いはなく，本モデルは既出のモデルにモードを導入しイベントを追加して作成することができた．追加にともない記述量と証明課題数が増えているが，自動証明数も増えていることから，多くはほぼ自明な内容の追加であるといえる．なお，既出のモデルで人手で証明した式の証明過程は本モデルに引き継がれ，ツールが証明過程を機械的に再現する．本モデルの方が自動証明できなかった式の数が少ないことは，証明過程を再利用できたことを示している．

5. 議論

本稿で我々は，統計的な振舞いをする機械学習適用モジュールと，離散論理に基づいて記述し検証できる論理モジュールを構造設計段階で分けて，両者のインタフェースとなる部分仕様を設定した．論理モジュールでは部分仕様を満たす/満たさないによって振舞いを場合分けして論理的に解析するとともに，部分仕様を満たす場合に機械学習適用モジュールの制約充足性テストから得た確率を付与することで，予測不能性のあるシステムの振舞いを統計的に検証した．一般に，複数の部分仕様が存在するときや，フェイルセーフ化のため複数の機械学習適用モジュールを利用するときであっても，部分仕様の充足性が各々独立に評価できるならば，論理モジュール側では複数の場合に分割して組み合わせることで同様な検証が行えると考えられる．一方，ある時点の充足性が過去の充足性の履歴に依存するときは，このような単純な手法では組合せが複雑になりすぎる．そのような対象についてはシステムの振舞いを確率的に記述し検証するしくみが必要であり，確率的な論理に基づく記述言語と検証方法が求められる．

ケーススタディでは，機械学習適用モジュールの振舞いの予測不能性から，部分仕様を満たさない場合が一定の確率で起こることを仮定した．これは，機械学習適用モジュールが誤作動を起こす入力データが，一定の確率で出現することに相当する．1 回の動作あたりの部分仕様を満

© 2020 Information Processing Society of Japan

たす確率を p（$0 \leq p \leq 1$）とすると，n 回目の動作で部分仕様を満たさなくなる確率 $Q(n)$ は $p^{n-1} \times (1-p)$ であり，部分仕様を満たさなくなるまでの平均動作回数は，

$$\sum_{n=1}^{\infty} n \times Q(n) - 1 = \frac{p}{1-p}$$

と計算できる．ここで，p はニューラルネットワークの正解率に依存するが，一般に正解率そのものではない．4.3 節では，さまざまな交通標識の並びを入力データとした制約充足性テストの結果から，p を算出した．第 1 回から連続する k 回の動作のいずれかで部分仕様を満たさなくなる確率は，下記である．

$$\sum_{n=1}^{k} Q(n) = \frac{1-p}{p} \times \sum_{n=1}^{k} p^n = 1 - p^k$$

テストから得られる p の値はテスト入力データに依存することから，利用環境を反映した入力データを選ぶ必要がある．たとえば，夜間や雨天などシーンによって正解率が大きく変わる場合には，シーンごとにテストし p を算出することも考えられる．リファレンス・モデルを使ったテストは，学習データセットに含まれない入力データについてテストできる点から，この目的に有効である．

ケーススタディの論理モジュールのモデル化には，形式手法 Event-B を使った．機械学習適用モジュールが部分仕様を満たしている/満たしていないことをモードの切り替えで表現するとともに，予測不能な振舞いをモードの非決定的な切り替えで表すことで，Event-B を拡張することなくモデルを記述し検証した．機械学習適用モジュールを考慮しない既出のモデル [19] とはモードの導入を除けばほぼ同一であり，本稿の手法が論理モジュールの開発過程に及ぼす影響は少ないと結論できる．また，段階的詳細化では下位（詳細化後）のモデルに上位（詳細化前）のモデルの性質が引き継がれるため，式 (2) は下位のモデルでも満足される．ケーススタディでは段階的詳細化による仕様レベルのモデル化と検証までを行い，プログラム導出は行っていないが，correctness by construction [4] に基づく開発手法が適用できると考えられる．

6. 関連研究

Tarasyuk ら [26] は，信頼性（与えられた時間範囲の間システムが正しく機能する確率）や反応性（要求されたサービスが与えられた時間範囲の間に完了する確率）など，確率的なイベントの停止性の検証を目的に，Event-B の非決定的な選択アクションに確率を付加する拡張を加え，段階的詳細化を再定義した．また，Probabilistic Event-B [3] では，段階的詳細化の最終段階に限られるが，アクションに加えてイベントやイベントパラメタの非決定的な選択にも確率を付加できる．これらは論理モジュールの設計に確率的振舞いを導入する研究であり，非決定的記述の設計詳細化として確率を付与する．このため確率は設計値として与えられるが，不変条件には確率が導入されておらず，確率的に満足される不変条件を記述し検証することができない．一方，我々は機械学習適用モジュールの確率的振舞いが論理モジュールに与える影響を分析するためモードを導入し，モードが維持される間は不変条件が満足されることを検証した．機械学習適用モジュールのテストによりモードが維持される確率を算出することで，不変条件が満足される確率を与えることができる．機械学習適用モジュールは論理モジュールの外部にあるため，モードが維持される確率は設計値ではなく統計的な推定値である．

ニューラルネットワークの品質評価の観点では，システムが対象とする実世界を学習データセットが十分に反映していること，未知の入力データに対してニューラルネットワークが十分に頑健であること，の 2 点が主な評価項目としてあげられる．前者に関しては，実世界の不確実性を認知構造上に配置し分類した研究 [8] などがある．本稿のリファレンス・モデルとの照合によるテストは，後者にかかわる．メタモルフィック・テスティング [5], [10] は，入力値の変化に対して期待される出力値の変化であるメタモルフィック関係を設定し，それを擬似正解値としてモデルの正しさをテストする．リファレンス・モデルを使ったテストと同様に出力値が不明な入力データを使ったテストを，より高精度で行うことが可能である．一方，リファレンス・モデルが処理の記述である部分仕様から導出されるのに対し，メタモルフィック関係は入力データに関する知見に基づいて設定されるため，本稿の部分仕様の充足性を検証する目的には適さない．ニューラルネットワークの入出力ノード間の充足性探索によるネットワークの正しさの検証 [16] や，入力データに対する微小な変換のネットワーク内伝播の解析によるネットワークの正しさの検証 [14] は，ネットワーク内部を参照した形式的かつ網羅的な検証方法である．これらは，論理的な充足可能性問題（SAT）を解くため検証できるネットワークの規模が制限され，利用状況が限られる．

7. 結論

本稿では，高信頼な機械学習適用ソフトウェア開発のための，演繹的開発手法と帰納的開発手法の結合による形式検証とテストについて述べた．一般に，機械学習適用ソフトウェアには入力データに依存する予測不能性があり，その振舞いは確率的にしか把握できない．我々のアプローチでは，機械学習適用モジュールと論理モジュールを分離するとともに，その間のインタフェースである部分仕様を設定する．論理モジュールについては部分仕様が満たされることを前提に論理的な検証を行う一方，機械学習適用モジュールについては入力データに依存する統計的ゆらぎを

© 2020 Information Processing Society of Japan

考慮したテストにより部分仕様の充足確率を評価し，論理的な検証結果に確率を付与する．このアプローチを具体化するため，機械学習適用ソフトウェア開発プロセスと，リファレンス・モデルを使った制約充足性テストを提案した．

ケーススタディでは論理モジュールのモデル記述に形式手法 Event-B を用い，テストによって計測された機械学習適用モジュールの確率的な制約充足性と，論理的な証明によって検証されたシステムの性質の結合の例を示した．Event-B を拡張する代わりにモードを導入し，モードが維持される間は不変条件が満足されることを検証するとともに，モードが維持される確率をテストから算出することで，不変条件が満足される確率を与えている．

機械学習の適用によりシステムの振舞いが確率的になることから，システムのモデル化と分析にも確率的な手法が求められる．より複雑な確率的振舞いを直接記述してシステム全体への影響を検証するための，確率的な論理に基づくモデル記述言語と不変条件の検証方法の検討が，今後の課題である．

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] Amine, M., Delahaye, B. and Lanoix, A.: Movint from Event-B to Probabilistic Event-B, *Proc. SAC 2017*, pp.1348–1355 (2017).
[4] Chapman, R.: Correctness by Construction: A Manifesto for High Integrity Software, *Proc. SCS 2005*, pp.43–46 (2005).
[5] Chen, T.Y., Juo, F.-C., Tse, T.H. and Zhou, Z.Q.: Metamorphic Testing and Beyond, *Proc. 11th STEP*, pp.94–100 (2004).
[6] Chu, L., Hu, X., Hu, J., Wang, L. and Pei, J.: Exact and Consistent Interpretation for Piecewise Linear Neural Networks: A Closed Form Solution, *Proc. KDD 2018*, pp.1244–1253 (2018).
[7] Credi, J.: *Traffic Sign Classification with Deep Convolutional Neural Networks*, Master's Thesis Complex Adaptive Systems, Chalmers University of Technology (2016).
[8] Czarnecky, K. and Salay, R.: Towards a Framework to Manage Perceptual Uncertainty for Safe Automated Driving, *Computer Safety, Reliability, and Security*, LNCS11094, pp.439–445 (2018).
[9] Dijkstra, E.W.: The Humble Programmer, *CACM*, Vol.15, No.10, pp.859–866 (1972).
[10] Ding, J., Kang, X. and Hu, X.-H.: Validating a Deep Learning Framework by Metamorphic Testing, *2nd International Workshop on Metamorphic Testing*, pp.28–34 (2017).
[11] Ehlers, R.: Formal Verification of Piece-Wise Linear Feed-Forward Neural Networks, arXiv: 1705.01320v3 [cs.LO] (2017).
[12] Garlan, D.: Software Engineering in an Uncertain World, *Proc. FoSER 2010*, pp.125–128 (2010).
[13] Goodfellow, I., Shelens, J. and Szegedy, C.: Explaining and Harnessing Adversarial Examples, arXive: 1412.6572 (2015).
[14] Huang, X., Kwiatkowska, M., Wang, S. and Wu, M.: Safety Verification of Deep Neural Networks, *Proc. CAV 2017*, LNCS 10426, pp.3–29, Springer (2017).
[15] 石川冬樹，来間啓伸，中島　震：不確かさを考慮したソフトウェア・テスティングおよび形式検証，情報処理，Vol.58, No.8, pp.693–695 (2017).
[16] Katz, G., Barrett, C., Dill, D., Julian, K. and Kochenderfer, M.: Reluplex: A Efficient SMT Solver for Verifying Deep Neural Networks, *Proc. CAV 2017*, LNCS 10426, pp.97–117, Springer (2017).
[17] Kohand, P.W. et al.: Understanding Black-box Predictions via Influence Functions, *ICML2017* (2017).
[18] 來間啓伸，中島　震：B メソッドによる形式仕様記述—ソフトウェアシステムのモデル化とその検証，近代科学社 (2007).
[19] 來間啓伸，中島　震：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，情報処理学会論文誌，Vol.57, No.8, pp.1690–1702 (2016).
[20] 來間啓伸，明神智之，佐藤直人，中川雄一郎，小川秀人：機械学習適用コンポーネントを含むソフトウェアの形式検証とテスト，IPSJ/SIGSE ウィンターワークショップ 2018・イン・宮島予稿集，pp.6–7 (2018).
[21] 丸山　宏：機械学習工学に向けて，日本ソフトウェア科学会第 34 回大会講演論文集 (2017).
[22] 中島　震，來間啓伸：Event-B：リファインメント・モデリングに基づく形式手法，近代科学社 (2015).
[23] Nakajima, S. and Bui, H.N.: Dataset Coverage for Testing Machine Learning Computer Programs, *Proc. APSEC 2016*, pp.297–304 (2016).
[24] Salay, R., Queiroz, R. and Czarnecki, K.: An Analysis of ISO 26262: Using Machine Learning Safely in Automotive Software, arXiv:1709.02435 (2017).
[25] Stallkamp, J., Schlipsing, M., Salmen, J. and Igel, C.: Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition, *Neural Networks* (2012).
[26] Tarasyuk, A., Troubitsyana, E. and Laibinis, L.: Integrating stochastic reasoning into Event-B development, *Formal Aspects of Computing*, Vol.27, pp.53–77 (2015).
[27] Wirth, N.: Program Development by Stepwise Refinement, *CACM*, Vol.14, No.4, pp.221–227 (1971).
[28] The German Traffic Sign Recognition Benchmark, available from ⟨http://benchmark.ini.rub.de/⟩.

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．

© 2020 Information Processing Society of Japan

佐藤 直人 （正会員）

2003 年東京工業大学情報工学科卒業．2005 年同大学大学院情報理工学研究科計算工学専攻修士課程修了．同年株式会社日立製作所入社．ソフトウェアテスト技術，形式手法，モデルベース開発技術等の研究開発に従事．博士（工学）．

中川 雄一郎

1999 年東海大学工学部航空宇宙学科卒業．2001 年同大学大学院工学研究科航空宇宙額学専攻修士課程修了．2005 年株式会社日立製作所入社．ソフトウェアテスト技術，形式手法等の研究開発に従事．

小川 秀人 （正会員）

1996 年名古屋大学大学院博士前期課程修了．1996 年株式会社日立製作所入社．2015 年北陸先端科学技術大学院大学博士後期課程修了．ソフトウェアテスティング，形式手法，ソフトウェア開発プロセス，ソフトウェアプロダクトライン等の研究開発に従事．博士（情報科学）．情報処理学会，電子情報通信学会，日本ソフトウェア科学会各会員．

© 2020 Information Processing Society of Japan