CB-002

自然言語処理モデルを利用したソフトウェア検証分野における反例予測
Prediction of Counter Examples in the Field of Software Verification by Using NLP Models

大野 亮祐[1)]　張江 洋次朗[2)]
Ryosuke Ono　Yojiro Harie

1　イントロダクション

モデル検査 (model checking) は，アルゴリズムやシステムの設計を状態空間として表現し，システムに期待される性質を満たすかどうかを全数検査する形式手法の一つである．状態空間に対して，検査したい性質のことを仕様と呼ぶ．モデル検査ツールに検査対象となる設計と仕様を記述し実行することで，その設計が仕様を満たしているかどうかを検査することができる．検査結果は真偽で示され，偽の場合は変数の状態遷移が反例として与えられる．ユーザは得られた反例や仕様を参照し，バグの原因を読み解く必要がある [1].

モデル検査には多くの利点があるが，長大な状態空間においては現実的な時間で全数探索をすることが不可能になるという，状態爆発問題 (state explosion problem) が存在している．この状態爆発問題を回避する手法として，Zhu ら [2] や張ら [3] により，Random Forest などの機械学習を用いた検査結果予測手法が提案されている．状態空間に対して全数探索を行う代わりに，機械学習を用いて検査結果を予測することで，検査時間を短縮することができる．しかし，これらの既存手法は仕様が満たされているかどうかの真偽判定の予測に留まり，反例トレースの予測はできていない．

Transformer モデルは，RNN や CNN などを一切使わずに Attention 機構だけを使うことで, 入力と出力の内容の広範囲な依存関係を捉えられる機械学習モデルである．さらに Ruoss らは，大規模な Transformer を単なる「統計的パターン認識器」とみなす従来の考え方から，一般的なアルゴリズム近似のための強力な手法とみなすというパラダイムシフトが起きていることに言及している [4].

このような背景から，本研究は Transformer の代表的なモデルである BERT を活用し，仕様の真偽判定だけでなく，仕様を満たしていない場合の反例予測手法を提案する．言い換えれば，本研究は検証アルゴリズム及びグラフ到達可能性解析アルゴリズムの Transformer による近似モデルの構築法を示すことであり，ソフトウェア検証と機械学習の融合を図る研究である．

2　既存研究

本節では本研究で用いるモデル検査や機械学習の概念，用語および既存研究の手法の詳細について説明する．

2.1　モデル検査

2.1.1　概要

モデル検査は，クリプキ構造を用いて定義された状態空間と仕様から，状態空間が仕様を満たしているかを検査する技術である．状態空間について，ここで用いられるクリプキ構造は非決定性有限オートマトンの一種であり，ノードはシステムの到達可能状態を，エッジは状態遷移を表している．モデル検査を行った結果として，設計が仕様を満たしているかどうかを True または False で返し，False の場合は初期状態からのパスを仕様に対する反例として出力する．

2.1.2　時相論理

モデル検査では，時間とともに内部状態が非決定的に変化するシステムを扱うため，仕様は時相論理を用いて記述される．時相論理は，論理積や論理和，否定などの論理演算に加えて時相演算子 (temporal operator) や経路限定子 (path quantifier) を導入したものである．時相演算子には「次の状態で」を表す X や「現在から始まる未来のある状態で」を表す F,「現在から始まる未来のあらゆる状態で」を表す G などがある．また，経路限定子には,「現在の状態から始まる，ある経路において」を意味する E や,「現在の状態から始まる，すべての経路において」を意味する A がある [5]．通常，経路限定子と時相演算子を組み合わせて時相論理は記述され，先に経路限定子を，後に時相演算子を記述する．

$$EF(x = 3)$$

上記のような時相論理があるとする．これは,「現在から始まるある経路が存在して，その未来で $x = 3$ が成り立つ状態が存在する」ことを意味する．これは,「$x = 3$ はいずれ成り立つ場合がある」と言い換えられる．

代表的な時相論理には CTL(Computation Tree Logic) と LTL(Linear-time Temporal Logic) があり，LTL の論理式は経路限定子が先頭の A だけであるような式である [6].

2.1.3　NuSMV

NuSMV[7] はモデル検査ツールの一種であり，SPIN[8] などのほかのモデル検査ツールに比べると大規模で複雑なシステムのモデル検査に適している．また，CTL と LTL の両方に対応している．NuSMV ではテキストデータとして状態遷移モデルと仕様を記述したファイルを実行することで設計が仕様を満たしているかを検査することができる．検査の結果，設計が仕様を満たしていれば True を，満たしていなければ False を返す．また，False の場合は反例が生成される．NuSMV における反例は，変数の状態遷移が記述された文字列である．以上より，反例を含む検査結果とその仕様が 1 対 1 に対応していることが分かる．

2.2　モデル検査結果の予測

Zhu ら [2] は，機械学習の二項分類を設計の真偽判定に適用する手法を提案している．この手法では，まずランダムに生成された複数の設計と LTL で記述された仕様に対してモデル検査を行った．そして，その結果をラベル付けして学習モデルを作成することで未知の設計と仕様に対する検査結果の真偽を推定している．また，Boosted Tree, Random Forest, Decision Tree, Logistic Regression の 4 つの機械学習のアルゴリズムをそれぞれ用いることでモデル検査の結果を予測し，選択するクリ

1)　金沢学院大学経済情報学部: ei22n009@kanazawa-gu.ac.jp
2)　金沢学院大学情報工学部: harie@kanazawa-gu.ac.jp

プキ構造によってはその精度が 99%から 30%まで大きく変動することを示した．

さらに，張ら [3] の研究では，Zhu らの研究と同様の手法をとった上で，訓練セットとテストセットでの仕様には一定の類型があると考えられることに着目した．そして訓練に使用する仕様のパターンをそうした類型にあてはまるものに限定することでより良い精度を得た．

これらの手法は全数探索を行わずに検査結果を予測するため，状態爆発問題を完全に回避し，検査時間を短縮することが出来るという利点がある．一方で，偽陽性 (False positive)・偽陰性 (False negative) が予測結果に含まれるため， どれだけ高精度でも，LTL 検査近似手法はモデル検査技術を代替することは不可能であることに言及している．

2.3 BERT

BERT は GLUE ベンチマークを含む様々な自然言語処理タスクに対応可能な機械学習モデルとして，2018 年に Google により提案された [9]．BERT の学習は，大規模な文章コーパスから汎用的な言語のパターンを獲得する「事前学習」と，個別のタスクのラベル付きデータを用いてそのタスクに特化させるよう学習する「ファインチューニング (fine-tuning)」の 2 つの段階で構成される [10]．BERT は文章をトークン毎に分解し，トークン列の一部を $[MASK]$ トークンに置き換え，$[MASK]$ で隠された箇所にどのトークンが配置されるのかを予測するよう学習を進める．また，トークン列の先頭に特殊トークン $[CLS]$ を，末尾に特殊トークン $[SEP]$ を加えることで，複数の文章を連結させて処理させることが可能である．BERT の特徴の 1 つは，ファインチューニングの際の汎用性が高いことが挙げられ，テキストを多クラスに分類するテキスト分類，文章要約，文章生成など多岐にわたる言語処理領域のタスクを扱うことができる．

3 提案手法

Zhu らや張らの研究では，設計の検査結果が偽である場合の反例を予測することはできていない．一方，モデル検査の反例はトレースの可視化を通じて設計の不備を特定し，設計の修正を容易にする重要な情報を提供する．本研究では，真偽判定と反例予測の 2 段階で BERT モデルを適用する反例予測手法を提案する．

図 1 に提案手法の全体像を示す．提案手法では，設計と仕様が記述された smv ファイルとその実行結果を入力にとり，次の手順を経て最終的に仕様の真偽と反例の推定結果を出力する．

①smv ファイルに記述された設計と仕様を抽出する．
②smv ファイルを実行し，仕様の真偽を抽出する．また，偽の場合はその反例を抽出する．
③抽出したものを設計，仕様，真偽，反例の列からなる 1 つの csv ファイルに出力し，それをトークナイズしてトークン列を得る．
④仕様の真偽判定用の BERT モデルを用いて仕様の真偽判定を行う．
⑤仕様が偽と判定された場合，トークン列を再度用いて仕様の反例出力用の BERT モデルを用いて反例を出力する．本研究はここに大きな新規性がある．

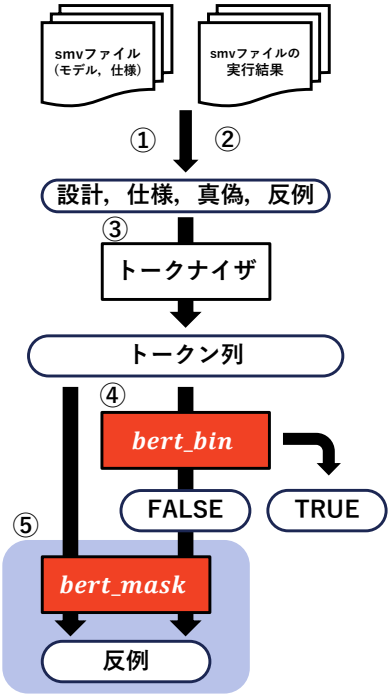

図 1: 反例予測手法の概略図

3.1 データセットの構築

我々は，NuSMV 公式ホームページと AIT[11] で公開されている smv ファイル群のうち，実行可能な 73 ファイルを実験対象とした．smv ファイル記述やその実行結果を抽出し csv ファイルへ整形する正規表現を用いたシェルスクリプトを実装し，コメントアウトなどのノイズ除去の前処理も同時に実行した．再現性を確保するためランダムシードを 42 に設定し，設計と仕様を 1 対 1 で対応させ，最終的に 220 個のデータからなるモデル検査データセットを構築した．さらに，構築したモデル検査データセットを，学習用データと検証用データで 8:2 の割合で分割した．

3.2 BERT モデルの設計

本研究では，HuggingFace が公開している事前学習済みモデル *bert-base-uncased* を利用し，トークナイザ，マスクスケジューリング及びファインチューニングの実装を行った．

3.2.1 トークナイザへの登録

BERT では，入力する文章をトークン列へ変換するトークン化 (tokenize) の処理が必要である．提供されているトークナイザには入力した文字列をトークンへ分割する機能が備わっている．しかし，時相論理記号をトークンとして提供トークナイザに事前に登録すると，それらの文字列を含むモジュール名や状態名が正しくトークン化されないという問題が生じる．そこで，NuSMV 形式の設計記述，LTL 式と CTL 式による仕様記述，そして検査結果の文字列をトークン化する正規表現を用いた字句解析器を実装した．

NuSMV の設計と仕様である検査式を記述する SMV 形式では，*VAR* や *ASSIGN* などの予約語が存在している．設計・仕様記述に関する予約語を含む代表的な 31 個のトークンを事前にトークナイザに登録し，加えて各設計仕様記述を前述の字句解析器でトークンに区切り，

Algorithm 1: *bert_bin* と *bert_mask* の同期型訓練法

Data: Pretrained BERT models *bert_bin* and *bert_mask*
Data: Training data *dataloader*
Data: Optimizer *AdamW* and loss function *CrossEntropyLoss* as *CEL*

for $epoch \leftarrow 1$ **to** num_epochs **do**
  for *each batch* ***in*** *dataloader* **do**
    $x \leftarrow Extract(batch)$;
    $y \leftarrow label(batch)$;
    $\hat{y} \leftarrow bert\_bin(x)$;
    $loss_{bin} \leftarrow CEL(\bar{y}, y)$;
    Reset gradients for *bert_bin*;
    $L_{bin}.backward()$;
    $optimizer_{bin}.step()$;
    $indices_f \leftarrow argmax\_indices(\hat{y})$;
    if $len(indices_f) > 0$ **then**
      $f\_input\_ids \leftarrow batch[indices_f]$;
      $x\_f \leftarrow Extract(f\_input\_ids)$;
      $y_f \leftarrow label(f\_input\_ids)$;
      $z \leftarrow bert\_mask(mask(x\_f))$;
      $L_{mask} \leftarrow CEL(z, y_f)$;
      Reset gradients for *bert_mask*;
      $L_{mask}.backward()$;
      $optimizer_{mask}.step()$;
    end
  end
end

トークン列に含まれる未知語をトークナイザに登録する手順を取った．

設計 $M = [w_{m_1}, w_{m_2}, \ldots, w_{m_j}]$，仕様 $S = [w_{s_1}, w_{s_2}, \ldots, w_{s_k}]$，真偽判定 $T \in \{[\mathrm{TRUE}], [\mathrm{FALSE}]\}$，検査結果列 $R = \left[w_{r_1}, w_{r_2}, \ldots, w_{r_l}\right]$ と表す．ここで，w_i はトークン列中の i 番目のトークンである．

トークン列を設計，仕様，検査結果の真偽値，偽の場合の反例の 4 項目を連結させたトークン列 $[[SEP], M, [CLS], S, [CLS], T, [CLS], R, [CLS]]$ からデータセットを作成した．実際に学習に用いるデータ形式はトークン列ではなく，トークナイザの辞書に登録されたトークンと対応する整数値の列である．また，BERT モデルが 1 度に扱うことのできる最大トークン長は 512 長という制限がある．今回使用した設計の中には最大長を超えるトークン列もあり，設計記述を分割しデータセットへ登録した．

3.2.2 ファインチューニング

同一の学習済み BERT モデルを 2 つ用意し，それぞれ「クラス分類タスク」と「反例予測タスク」の 2 種のタスクの学習を行う．

クラス分類タスク学習モデル *bert_bin* は，設計と仕様を記述したトークン列を入力として与えられると，その仕様を設計が真に満たしているか否かを 2 値で分類する．*bert_bin* への入力 $x_j = [[SEP], M, [CLS], S, [CLS]]$ に対して，x_j に対応する真偽結果のラベル $y_j \in \{0, 1\}$ を割り充てることで，2 クラス分類問題とみなすことができる．*bert_bin* モデルの学習は，入力 x に対する予測値 $\hat{y} = bert\_bin(x)$ について，次のクロスエントロピー損失 L_{bin} の最小化によりパラメータを最適することである．

$$L_{bin} = -\sum_j \left(y_j \log \left(\frac{e^{\hat{y}_j}}{\sum e^{\hat{y}}} \right) + (1 - y_j) \log \left(1 - \frac{e^{\hat{y}_j}}{\sum e^{\hat{y}}} \right) \right)$$

続いて，反例予測タスクでは，一部の単語を $[MASK]$ に置き換えて，その部分に入るトークンを予測するマスク付き言語モデル (MLM) としての訓練を行う．反例予測学習モデル *bert_mask* に対して，入力 $x' = [[SEP], M, [CLS], S, [CLS], R, [CLS]]$ で構成する．この x' の R に相当する部分にマスク処理を行うが，反例部のマスキングの具体的な方針については 3.2.3 節で記述する．*bert_mask* の出力に softmax 関数 σ を適用し得られる確率分布 $z_j = \sigma(bert\_mask(x_j^{mask}))$ に対して，マスクされたトークンに対する損失関数を次のように定めた．

$$L_{mask} = -\sum_{j=1} \sum_{i \in x_j^{mask}} y_j^i \log(z_j^i)$$

ここで x_j^{mask} を入力列 x'_j のマスクされたトークンの添え字集合，y_j^i を実際のトークンとなるワンホットエンコードされた正解ラベルとする．

本研究では，*bert_bin* と *bert_mask* に対して，両モデルの非同期型の学習と同期型の学習の比較を行う．同期型では Algorithm 1 のように学習を行う．11 行目で，*bert_bin* の予測結果が偽判定だった場合に条件分岐し，*bert_mask* の学習ステップへ進む．*bert_bin* には偽陰性の結果を含むが，反例を含まない入力を *bert_mask* への学習ノイズとして混入させる．非同期型と同期型で学習結果を比較し，学習ノイズによるモデルのロバスト性への影響について検証を行う [12].

3.2.3 マスクスケジューリング

BERT では，入力データに含まれる 15% のトークンに対して，ランダムに $[MASK]$ トークンを割り当て，そのうちのさらに 10% に別のトークンを割り当てる．Transformer モデルでは，マスクのスケジュールを制御することで，その精度を向上させた事例がある [13]．一方で，LTL（線形時間論理）における演算子 U（until）や X（next）は，現在の状態を前提として未来の状態を表す論理を記述するために使用される．反例予測には状態間の因果関係を双方向的に捉える必要があるものの，ある地点での状態を前提として仕様オートマトンの遷移が決定することから，我々は状態パスの生成手順は過去から未来へ自己回帰的な生成を考慮する必要があると考えた．そこで，ファインチューニング時のマスクスケジューリングは，図 2 のように下方から $[MASK]$ トークンを置き換える関数を定義した．

4 実験

我々はバッチサイズ 4，4000 ステップで学習を設定し，Google Colaboratory 上で，表 1 の環境の元，実験を行った．ライブラリには，Adam オプティマイザー（Adaptive moment optimization），PyTorch2.3.0+cu121，TensorFlow2.15.0 を使用した．

作成した 220 個のデータから，非同期型訓練として *bert_bin* の評価を行い，その結果を表 2 に示す．

4.1 考察

Zhu らの実験では，対象モデルと仕様の組み合わせによっては判定精度が非常に高くなることが指摘されてお

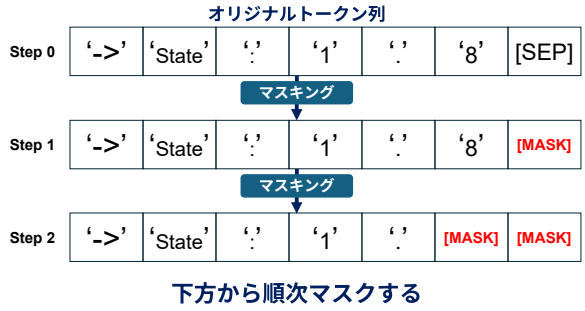

図 2: 反例予測モデルにおける下方マスクスケジューリング

表 1: 実験環境

カテゴリ	詳細
Hardware Information	
GPU	NVIDIA A100-SXM4-40GB (40960 MiB)
CPU	Intel(R) Xeon(R) CPU @ 2.20GHz
RAM	83 GiB
Software Information	
Operating System	Linux-6.1.85+-x86_64-with-glibc2.35
Python Version	3.10.12

り，本研究でも同様の現象が起きていないか検討を行った．データセットに検査結果が True となるデータが多く含まれていることが原因で，検査精度が非常に高い数値をとったと我々は結論付けた．検査結果が False となるような仕様をデータセットに追加するため，データセットの仕様部の時相論理記号の先頭に否定論理演算子「!」をつけ，データセットの拡充を図った．このとき，実行不可能になったファイルを除いた 67 ファイルが対象となり，新たに 525 個のデータをもつモデル検査データセットを構築した．元の仕様と「!」トークンを追加した仕様とでは両者の類似度は高いにも関わらず，その結果が真逆になることから，導入したクロスエントロピー損失では正しく分類学習が行われない可能性が高い．そのため，新しいモデル検査データセットに対する有効な損失関数を新たに導入する必要がある．

5 結論と今後の展望

本論文では，状態爆発問題を回避するため，自然言語処理モデルによる反例予測手法を提案した．反例を予測する機械学習を用いた LTL 検査近似手法は，我々の知る限りではこれまでにない．また，仕様の真偽判定に自然言語処理モデルを適用した点も既存研究とは異なっている．本研究は新規手法の提案に留まり，*bert_bin* モデルと *bert_mask* モデルに対して，新しく構築したデータセットを用いた評価は発表時に示す．併せて，同期型訓練と非同期型訓練に関する比較も示す．

今回は公開されている smv ファイルを利用したが，集めたデータにおける，実行結果の真偽の割合には大きく偏りがあることが課題となった．そのため，今後は設計モデルのデータの自動生成や，ライブラリ化を行いたい．

表 2: *bert_mask* の評価結果

指標	*bert_mask*
Accuracy	1.0000
Precision	1.0000
Recall	1.0000
F1 Score	1.0000
ROC-AUC	1.0000

謝辞

本研究の一部は JSPS 科研費 23K05416 の助成を受けた．

参考文献

[1] E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Trans. Program. Lang. Syst.*, Vol. 8, No. 2, pp. 244–263, April 1986. Place: New York, NY, USA Publisher: Association for Computing Machinery.

[2] Weijun Zhu, Huanmei Wu, and Miaolei Deng. Ltl model checking based on binary classification of machine learning. *IEEE access*, Vol. 7, pp. 135703–135719, 2019.

[3] 張超群, 岸知二. 検証パターンに注目した機械学習に基づくモデル検査手法の評価. ソフトウェア工学の基礎ワークショップ論文集, Vol. 29, pp. 79–84, 2022.

[4] Anian Ruoss, Grégoire Delétang, Sourabh Medapati, Jordi Grau-Moya, Li Kevin Wenliang, Elliot Catt, John Reid, and Tim Genewein. Grandmaster-level chess without search, 2024.

[5] 西崎真也. 電子情報通信学会『知識の森』12 群 2 編 7 章 数理論理学. Accessed: 2024-06-12.

[6] 土屋達弘, 菊野亨. モデル検査入門. 計測と制御, Vol. 48, No. 11, pp. 797–802, 2009.

[7] Alessandro Cimatti, Edmund Clarke, Enrico Giunchiglia, Fausto Giunchiglia, Marco Pistore, Marco Roveri, Roberto Sebastiani, and Armando Tacchella. Nusmv 2: An opensource tool for symbolic model checking. In *Computer Aided Verification: 14th International Conference, CAV 2002 Copenhagen, Denmark, July 27–31, 2002 Proceedings 14*, pp. 359–364. Springer, 2002.

[8] G.J. Holzmann. The model checker spin. *IEEE Transactions on Software Engineering*, Vol. 23, No. 5, pp. 279–295, 1997.

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[10] 近江崇宏, 金田健太郎, 森長誠, 江間見亜利ほか. BERT による自然言語処理入門: Transformers を使った実践プログラミング. 株式会社 オーム社, 2021.

[11] sct-dse public. Mutation testing with hyperproperties - smv models, 2024. Accessed: 2024-06-13.

[12] Milad Moradi and Matthias Samwald. Evaluating the robustness of neural language models to input perturbations. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 1558–1570, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[13] Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T Freeman. Maskgit: Masked generative image transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11315–11325, 2022.