ニューラルネットワークを用いたプログラムの停止性検証のための辞書式ランキング関数推論

朱 光漪　小林 直樹　酒寄 健

ランキング関数は，プログラムの各反復に伴って整礎的な順序の下で狭義単調減少する関数であり，プログラムの停止性検証等に用いられている．Giacobbe ら（2022）は，プログラムの実行トレースを入力として，ニューラルネットワークがランキング関数として振る舞うように学習する手法を提案した．しかし、この手法で推論できる辞書式ランキング関数の表現には制限がある．本研究では，ニューラルネットワークを用いたより柔軟な辞書式ランキング関数を推論する手法を提案する．また，提案手法を実装し，TermCOMP のベンチマークを用いてその有効性を評価した．

1　はじめに

プログラムの停止性検証は，プログラム検証における重要な課題の一つである．その代表的な手法として，ランキング関数を用いる方法がある．ランキング関数は，プログラム状態を入力として，出力がループの各反復に伴って整礎的な順序の下で狭義単調減少する関数である．各ループについてその存在を示すことでプログラムの停止性を証明できる [1].

ランキング関数の推論手法には様々なものがあるが，Giacobbe ら [2] は，プログラムの実行トレースを学習データとして用い，ニューラルネットワークをランキング関数として学習する手法を提案した．学習したニューラルネットワークがランキング関数の条件を満たすかは，形式検証によって確認される．プログラムによっては，出力が複数の成分からなり，辞書式順序に従って減少する辞書式ランキング関数が必要となるが，Giacobbe らの手法が推論できる辞書式ランキング関数の表現には制限がある．特に，単一のループからなるプログラムに対してはスカラー値を出力するランキング関数しか推論できない．

本研究では，Giacobbe らの枠組みを拡張し，一つのループに対して辞書式ランキング関数を推論する手法を提案する．辞書式ランキング関数を学習する手法を提案する．また，提案手法を実装し，TermCOMP [4] のベンチマークを用いてその有効性を評価した．

2　ランキング関数とニューラルネットワークによる推論

本節では，ランキング関数とニューラルネットワークを用いた既存の推論手法を説明した後，本研究で対象とする辞書式ランキング関数を導入する．

2.1　ランキング関数

ランキング関数は，ループヘッドにおけるプログラム状態（変数の値）を受け取り，ループの各反復に伴って整礎的な順序の下で狭義単調減少する値を返す関数である．整礎的な順序は無限降下列が存在しないため，ランキング関数が存在すればループは有限回の反復で停止する．

例えば，図 1 のプログラムを考える．変数は x, y

---

*Neural Network-Based Lexicographic Ranking Function Inference for Program Termination

This is an unrefereed paper. Copyrights belong to the Author(s).

Konami Shu, Naoki Kobayashi, Ken Sakayori, 東京大学大学院情報理工学系研究科, Graduate School of Information Science and Technology, The University of Tokyo.

```
while (x < y) {
  x = x + 2;
  y = y + 1;
}
```

図 1　ランキング関数 $\max(y-x,0)$ を持つプログラム

の 2 つであり，これらの変数の値がランキング関数の入力となる．このプログラムでは，各反復で $y-x$ が 1 ずつ減少し，ループに入る場合 $y-x$ は常に 0 より大きいため，

$$r(x,y) = \max(y-x,0) \tag{1}$$

はこのプログラムのランキング関数である．

本論文では，このように単一の値を出力するランキング関数を，スカラー値ランキング関数と呼ぶ．

2.2　ニューラルネットワークによるランキング関数推論

前節で述べた $r(x,y) = \max(y-x,0)$ は，入力が x と y，重みが -1 と 1，バイアス 0，中間層 1 層・1 ノードのニューラルネットワーク $\mathrm{ReLU}(-x+y)$ として表現できる（図 2）．

Giacobbe ら [2] は，このようなランキング関数として振る舞うニューラルネットワークを，プログラムの実行トレースから学習する手法を提案した．その手順は次のとおりである．

まず，ループヘッドに到達するたびにプログラム状態を記録し，ある反復での状態と次の反復での状態のペアを収集する．例えば，図 1 のプログラムを初期値 $(x,y) = (1,5)$ で実行すると，次の反復での状態は $(x,y) = (3,6)$ となり，ペア $((1,5),(3,6))$ が得られる．同様に実行を続けると，

$$\{((1,5),(3,6)),((3,6),(5,7)),((5,7),(7,8)),\\ ((7,8),(9,9))\}$$

が得られる．これらを集めた集合を学習データとする．

次に，各ペアについて，ニューラルネットワークの出力が狭義単調減少するように，重みとバイアスを学習する．

最後に，各反復でニューラルネットワークの出力が狭義単調減少することを形式検証によって確認する．また，中間層に ReLU 活性化関数を用いるため，

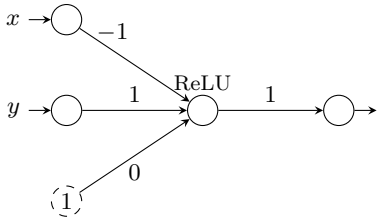

図 2　式 (1) のランキング関数に対応したニューラルネットワーク

ニューラルネットワークの出力は常に非負であり，出力の無限降下列は存在しない．以上の手順を通じて，ランキング関数の条件を満たすニューラルネットワークを得る．

2.3　辞書式ランキング関数

ランキング関数の出力を値の組とし，各反復でその組が辞書式順序に従って減少するものを，辞書式ランキング関数という．

単一のスカラー値では停止性を示しにくいプログラムでも，辞書式ランキング関数であれば停止性を示せる場合がある．例えば，図 3 のプログラムは辞書式ランキング関数が必要となる．各反復で x は 1 増加し，y は x の値だけ増加する．そのため，$x<2$ の間は x が増加し，その後は $x-y$ が減少し，最終的に停止する．この振る舞いから，

$$r(x,y) = (\max(2-x,0),\max(x-y,0)) \tag{2}$$

はこのプログラムの辞書式ランキング関数である．実際に，初期値 $(x,y) = (-1,-3)$ で実行すると，ループヘッドでの実行トレースは

$$(-1,-3) \to (0,-3) \to (1,-2) \to (2,0) \to (3,3)$$

となる．各状態に対する r の値は

$$(3,2) \to (2,3) \to (1,3) \to (0,2) \to (0,0) \tag{3}$$

となり，辞書式順序に従って減少する．

本研究では，このような辞書式ランキング関数を推論する手法を提案する．

3　提案手法

提案手法は，2.2 節で述べた枠組みに基づく．ニューラルネットワーク学習のフェーズで辞書式ランキング関数の各成分を逐次的に学習する．全体の流れを図 4

```
while (x > y) {
  x = x + 1;
  y = y + x;
}
```

図 3　辞書式ランキング関数
$(\max(2-x,0),\max(x-y,0))$ **を持つプログラム**

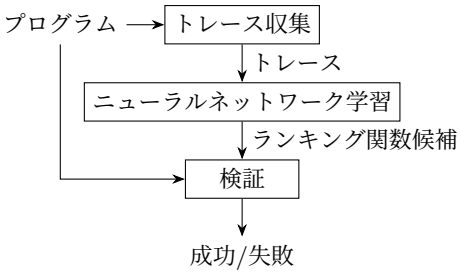

図 4　推論手法の全体の流れ

に示す．推論は以下の 3 ステップから構成される．

1. **トレース収集**：プログラムをランダムな入力で実行し，ある反復と次の反復でのプログラム状態のペアを収集し，学習データとする．
2. **ニューラルネットワーク学習**：学習データを用いて，各反復の状態間で出力値が減少するようにニューラルネットワークを学習する．ニューラルネットワークのパラメータは実数値であり，整数に丸めたものをランキング関数の候補として扱う．
3. **形式手法による検証**：得られたランキング関数の候補が実際にランキング関数の条件を満たすかを形式手法を用いて検証する．具体的には，ループの各反復においてランキング関数候補の出力が狭義単調減少するという性質を safety property としてプログラムのループヘッドに記述し，既存の検証器を用いて検証する．

3.1　ニューラルネットワークのモデル

本研究で用いるニューラルネットワークは，中間層を 1 層持ち，n 次元の入力に対して 1 次元の値を出力する．パラメータを θ とし，このニューラルネットワークが表現する関数を $f(\theta;\cdot):\mathbb{R}^n\to\mathbb{R}$ と表記する．中間層のノード数を h，第 j ノードの重みを $w_j\in\mathbb{R}^n$，バイアスを $b_j\in\mathbb{R}$ とする．中間層では ReLU 活性化関数を用い，出力層の重みは固定して学習しない．このとき，プログラム状態 $o\in\mathbb{R}^n$ に対する出力は

$$f(\theta;o)=\sum_{j=1}^{h}\mathrm{ReLU}(w_j\cdot o+b_j)$$

となる．学習するパラメータは $\theta=(w_1,b_1,\ldots,w_h,b_h)$ である．

中間層で ReLU 活性化関数を用いるため，出力は常に非負である．パラメータを整数に丸めれば出力は非負整数となり，その上の順序は整礎的である．したがって，ループの各反復に伴って出力が狭義単調減少するよう学習できれば，このニューラルネットワークをランキング関数とみなすことができる．

3.2　辞書式ランキング関数の学習

辞書式ランキング関数は複数の関数の組 $r=(r_1,\ldots,r_k)$ であり，各反復でその出力が辞書式順序に従って減少する．各 r_i を第 i 成分と呼ぶ．提案手法では，辞書式ランキング関数の各成分を順に学習する．

辞書式順序で減少する組は次の性質を持つ．第 1 成分は広義単調減少し，第 1 成分が変化しないペアに対して第 2 成分が広義単調減少する．同様に，第 1 成分から第 $i-1$ 成分までが変化しないペアに対して第 i 成分が広義単調減少する．例えば，式 (3) の第 1 成分は $3\to2\to1\to0\to0$ と広義単調減少し，第 1 成分が変化しない最後のペアについて，第 2 成分が $2\to0$ と狭義単調減少する．

この性質から，各成分の学習では，すべての学習データのペアについて広義単調減少し，できるだけ多くのペアで狭義単調減少することを目指す．そのために，以下の損失関数を最小化するようにパラメータを更新する．

$$\arg\min_{\theta}\frac{1}{|D|}\sum_{(o,o')\in D}\max(f(\theta;o')-f(\theta;o),0)+ p\times\max(f(\theta;o')-f(\theta;o)+\delta,0) \tag{4}$$

$\delta>0$，$0<p<1$ とする．$D\subset\mathbb{R}^n\times\mathbb{R}^n$ は，ある反

復でのプログラム状態 o と，その次の反復でのプログラム状態 o' のペアの集合である．損失関数の第 1 項は出力が増加した場合にのみ正の値となるため，広義単調減少を満たすように学習を促す．第 2 項は，出力が δ だけ減少しない場合に正の値となるため，狭義単調減少を満たすように学習を促す．辞書式ランキング関数では，まず広義単調減少を満たすことが重要であるため，第 2 項には重み $p(<1)$ を与えて学習する．

第 i 成分の学習に用いる学習データを D_i とし，D_1 はトレース収集で得られたペアの集合全体とする．D_i を用いて損失関数が収束するまで学習し，得られたニューラルネットワークの係数を整数値に丸めることで，第 i 成分の候補 r_i を得る．次に，r_i の出力が変化しなかったペアのみを抽出し，

$$D_{i+1} = \{(o, o') \in D_i \mid r_i(o) = r_i(o')\}$$

を次の成分の学習データとする．この操作を損失関数が 0 になるまで繰り返し，辞書式ランキング関数 $(r_1, \ldots, r_k)$ を得る．

なお，辞書式ランキング関数を必要としないループでは，第 1 成分の学習のみで損失関数が 0 となるため，本手法は通常のランキング関数も統一的に学習できる．

例 1 図 3 のプログラムを考える．ループヘッドにおける変数 (x, y) の値を記録し，ある反復と次の反復での状態のペアを集めると，

$$\begin{aligned} D_1 = \{&\big((-1,-3),(0,-3)\big),\ \big((0,-3),(1,-2)\big),\\ &\big((1,-2),(2,0)\big),\ \big((2,0),(3,3)\big),\ \ldots\} \end{aligned}$$

のような学習データが得られる．

学習には，図 2 のように中間層 1 層，ノード 1 個のニューラルネットワークを用いる．1 回目の学習で $\max(2-x, 0)$ が得られたとする．この関数は D_1 のすべてのペアで広義単調減少する．値が変化しなかった組のみを残すと $D_2 = \{((2,0),(3,3)), \ldots\}$ となる．2 回目の学習で損失が 0 となり，$\max(x-y, 0)$ が得られた場合，$(\max(2-x, 0),\ \max(x-y, 0))$ が辞書式ランキング関数候補となる．

3.3 検証

学習で得られた辞書式ランキング関数候補 r は，有限個の実行トレースから得られたものであり，さらに係数の丸めを経ているため，すべての実行において辞書式順序で減少するとは限らない．そこで，ループの各反復において r の出力が辞書式順序で減少するという性質を safety property として記述し，既存の検証器を用いて検証する．

例えば，(r_1, r_2) からなる 2 つの成分の辞書式ランキング関数が得られた場合，while 文のループヘッドに，

$$\texttt{assert}(r_1(o') < r_1(o) \ \texttt{||}\ r_1(o') == r_1(o)\ \texttt{\&\&}\ r_2(o') < r_2(o))$$

を追加し，この assertion が任意のプログラム入力に対して成功することを検証する．(o, o') はある反復と次の反復でのプログラム状態のペアである．検証に成功した場合，(r_1, r_2) は正しい辞書式ランキング関数である．失敗した場合は，推論に失敗したものとして扱う．

4 実験

3 節で述べた手順に従い，C プログラムのループについて辞書式ランキング関数を推論するプロトタイプ νRank を実装した．現在の実装では，ループを 1 つだけ含むプログラムを対象とする．νRank は辞書式ランキング関数の各成分について学習を 3 回試行し，学習データに含まれるすべてのペアで値が広義単調減少する結果のうち，狭義単調減少するペアが最も多いものを採用する．いずれの結果も広義単調減少とならない場合は，推論を失敗とする．学習には機械学習ライブラリ PyTorch [13] を，検証には Ultimate Automizer [6]（バージョン 0.3.1）を用いた．

スカラー値ランキング関数のみを推論する設定でも実験した．この設定では，式 (4) の第 1 項を除いた $\max(f(\theta; o') - f(\theta; o) + \delta, 0)$ を損失関数とし，成分数の上限を 1 とする．実装のその他の部分は νRank と共通である．

比較対象には TermCOMP 2023 の C Integer カテゴリで優勝した AProVE [3]（2026 年 2 月 15 日公開のリリース）を用いた．

表 1　実験結果（150 件中の成功数）

手法	成功数（個）	成功率（%）
νRank（辞書式）	109.2	72.8
νRank（スカラー値）	88.2	58.8
AProVE	141	94

4.1　ベンチマークと実験環境

ベンチマークには TermCOMP [4] の C Integer カテゴリに含まれる整数を扱う C プログラムを用いた．停止しないプログラムおよび複数のループを含むプログラムを除外 †1 した 150 件を評価対象とした．

学習には AdamW [11] を用い，学習率は 0.01，中間層はノード数 2 の 1 層とした．損失関数のパラメータ $\delta = 1, p = 0.6$ とし，各学習のエポック数の上限を 3000 とした．

実験は Apple M1，16 GB メモリの Mac（macOS Tahoe 26.5）上で実施し，GPU は使用していない．Ultimate Automizer および AProVE は Docker 上で実行した．νRank についてはトレース収集，学習，検証のそれぞれに 120 秒，AProVE には 360 秒のタイムアウトを設けた．

4.2　実験結果

表 1 に実験結果を示す．νRank の結果は乱数に依存するため，ニューラルネットワークの初期化に用いる乱数シード 3 種とトレース収集に用いる乱数シード 2 種の組合せ，計 6 通りの設定で実験し，その平均値を示す．

νRank は 150 件中平均 109.2 件（72.8%）について正しいランキング関数を推論できた．このうち，辞書式であったものは平均 22.7 件であった．

スカラー値ランキング関数のみを推論する設定では，結果は平均 88.2 件であり，辞書式への拡張によって平均 21.0 件増加した．6 設定を通して 1 回以上成功した件数で比較すると，辞書式のみが推論できたのは 23 件，そのうち 21 件は成分数が 2 以上であり，成分数の増加が推論成功に寄与している．

†1　複数のループを含むプログラムを除外したのは現在の実装上の制限によるものであり，提案手法自体は各ループに対して独立に適用できる．

AProVE は 150 件中 141 件（94%）について停止性を証明できた．整数 C プログラムに対する停止性判定の既存手法は強力であり，提案手法は解決件数で上回るものではない．証明できなかった 9 件の内訳は，タイムアウトが 5 件，メモリ不足による強制終了が 3 件，停止性を判定できなかった例が 1 件であった．

4.3　考察

4.3.1　辞書式ランキング関数が有効に働いた例

νRank は，各反復における変数の変化が線形であり，1 つの成分では順序が定まらないループに対して辞書式ランキング関数を推論できている．PodelskiRybalchenko-TACAS2011-Fig4_true-termination を例に示す．

```
while (x > 0 && y > 0) {
  if (__VERIFIER_nondet_int() != 0) {
    x = x - 1;
    y = __VERIFIER_nondet_int();
  } else {
    y = y - 1;
  }
}
```

第 1 の分岐では x が 1 減少するが y は任意の値に変化する．第 2 の分岐では x は変化せず y が 1 減少する．x と y のいずれか一方だけでは反復ごとの順序が定まらないが，x を優先する辞書順であればすべての反復で減少する．νRank はこのループに対し，$(\mathrm{ReLU}(2x, 0) + 1,\ \mathrm{ReLU}(y, 0))$ を推論し，検証に成功した．同様の構造を持つ他のプログラムでも推論に成功した．

4.3.2　乱数によるばらつき

6 設定すべてで成功した 88 件のうち，辞書式ランキング関数が推論されたものは 8 件（9.1%）である．設定によって成否が変わったプログラム 39 件のうち，辞書式ランキング関数が推論されたものは 14 件（35.9%）を占める．成分数が増えるほど推論は不安定になることが観測できた．

原因は辞書式ランキング関数の探索方法にあると考えられる．第 2 成分以降の学習には，第 1 成分の値が変化しなかったペアのみを用いる．そのため成分

が深くなるほど学習データが減少し，初期値の乱数に対して結果が敏感になり，正確な学習も難しくなる．実際に，第 2 成分の学習データが 50 を下回った場合の成功例は確認できなかった．

4.3.3 推論に失敗した例

失敗の原因は主に 3 つに分類できた．

第一に，変数の更新が非線形である場合である．推論に失敗した ChawdharyCookGulwaniSagivYang-ESOP2008-aaron12_true-termination は次のループを含む．

```
while (x >= y) {
  if (__VERIFIER_nondet_int() != 0) {
    x = x + 1;
    y = y + x;
  } else {
    x = x - z;
    y = y + (z * z);
    z = z - 1;
  }
}
```

分岐によって変化量が異なることに加え，第 2 の分岐では y が z^2 ずつ増加するため，トレースから線形なランキング関数を見つけることが難しくなると考えられる．

第二に，ランキング関数が大きな定数項を必要とする場合である．次のループをもつ Overflow の推論に失敗した．

```
while (i <= 2147483647) { i = i + 1; }
```

変数が 1 つしかないため，ランキング関数は定数から i を引いた形に限られ，定数項は係数の 2×10^9 倍を要する．勾配法で到達できる範囲を超えており，νRank では推論が難しいと考えられる．

第三に，ループへの進入条件が狭くトレースを収集できない場合である．例えば，ガードが未初期化の変数を含んでいたり，ある変数が特定の値と等しいことを含んでいた利する場合，一様乱数から生成した初期値がこれらを満たす確率は小さく，学習データが得られない．

5 関連研究

記号的なアプローチに基づく停止性検証やランキング関数推論は，これまでに多く研究されてきた．テンプレートに基づくランキング関数の合成手法 [14] を基礎として，辞書式ランキング関数を対象とする手法 [10] や，反例を用いて制約を更新していく手法 [5] などが提案されている．停止性検証ツールとしては，本研究の比較対象である AProVE [3]，Ultimate Automizer [6]，MuVal [16] などがあり，いずれも記号的なアプローチに基づいている．

一方，機械学習を用いた手法の研究は比較的少ない．Giacobbe ら [2] の研究がニューラルネットワークを用いた代表的な研究の一つである．シグモイド関数を活性化関数として用いることで，非多項式ランキング関数の合成を扱う手法も提案されている [15]．また，近年では，ローカル大規模言語モデルを用いてランキング関数を直接合成する PROTON [12] のような手法も提案されている．しかし，これらの手法はいずれも，辞書式ランキング関数の推論を直接の対象としていない．

学習したニューラルネットワークの重みとバイアスから整数上の論理式を抽出することで，ニューラルネットワークを用いたプログラム検証や不変条件合成への応用も研究されている [7]．学習したネットワークから記号的な表現を取り出す点は本研究と共通する．

6 まとめと今後の展望

本研究では，辞書式ランキング関数をニューラルネットワークによって推論する手法を提案した．提案手法を νRank として実装し，TermCOMP のベンチマークを対象に評価実験を行った．スカラー値ランキング関数では解決できない，辞書式ランキング関数を必要とするループにも提案手法は適用可能であることを確認した．一方で，変数の更新が非線形なプログラム，ランキング関数が大きな定数項を必要とするプログラム，ループへの進入条件が狭くトレースを収集できないプログラムなどは，推論に失敗する場合があり，提案手法の限界も明らかになった．

今後の展望として，検証で失敗した場合，得られた反例を学習データに追加して再学習させる方法が考えられる．また，ニューラルネットワークを用いた提案手法を，高階プログラムの Liveness 検証 [8] [9] へ

応用することも検討したい.

謝辞

本研究は，JST 経済安全保障重要技術育成プログラム【JPMJKP24U4】の支援を受けたものです.

参 考 文 献

[1] Floyd, R. W.: *Assigning Meanings to Programs*, Springer Netherlands, 1993, pp. 65–81.

[2] Giacobbe, M., Kroening, D., and Parsert, J.: Neural termination analysis, *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE 2022*, Roychoudhury, A., Cadar, C., and Kim, M.(eds.), ACM, 2022, pp. 633–645.

[3] Giesl, J., Aschermann, C., Brockschmidt, M., Emmes, F., Frohn, F., Fuhs, C., Hensel, J., Otto, C., Plücker, M., Schneider-Kamp, P., Ströder, T., Swiderski, S., and Thiemann, R.: Analyzing Program Termination and Complexity Automatically with AProVE, *Journal of Automated Reasoning*, Vol. 58, No. 1(2017), pp. 3–31.

[4] Giesl, J., Rubio, A., Sternagel, C., Waldmann, J., and Yamada, A.: The Termination and Complexity Competition, *Tools and Algorithms for the Construction and Analysis of Systems*, Beyer, D., Huisman, M., Kordon, F., and Steffen, B.(eds.), Cham, Springer International Publishing, 2019, pp. 156–166.

[5] Gonnord, L., Monniaux, D., and Radanne, G.: Synthesis of ranking functions using extremal counterexamples, *Proceedings of the 36th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '15, New York, NY, USA, Association for Computing Machinery, 2015, pp. 608–618.

[6] Heizmann, M., Hoenicke, J., and Podelski, A.: Software Model Checking for People Who Love Automata, *Computer Aided Verification*, Sharygina, N. and Veith, H.(eds.), Berlin, Heidelberg, Springer Berlin Heidelberg, 2013, pp. 36–52.

[7] Kobayashi, N., Sekiyama, T., Sato, I., and Unno, H.: Towards neural-network-guided program synthesis and verification, *Formal Methods Syst. Des.*, Vol. 67, No. 2(2025), pp. 222–254.

[8] Kobayashi, N., Tanahashi, K., Sato, R., and Tsukada, T.: HFL(Z) Validity Checking for Automated Program Verification, *Proc. ACM Program. Lang.*, Vol. 7, No. POPL(2023).

[9] Kuwahara, T., Terauchi, T., Unno, H., and Kobayashi, N.: Automatic Termination Verification for Higher-Order Functional Programs, *Programming Languages and Systems - 23rd European Symposium on Programming, ESOP 2014, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2014*, Shao, Z.(ed.), Lecture Notes in Computer Science, Vol. 8410, Springer, 2014, pp. 392–411.

[10] Leike, J. and Heizmann, M.: Ranking Templates for Linear Loops, *Log. Methods Comput. Sci.*, Vol. 11, No. 1(2015).

[11] Loshchilov, I. and Hutter, F.: Decoupled weight decay regularization, *International Conference on Learning Representations*, 2019.

[12] Mukhopadhyay, D., Metta, R., Karmarkar, H., and Madhukar, K.: PROTON 2.1: Synthesizing Ranking Functions via fine-tuned locally Hosted LLM (Competition Contribution), *Tools and Algorithms for the Construction and Analysis of Systems*, Gurfinkel, A. and Heule, M.(eds.), Cham, Springer Nature Switzerland, 2025, pp. 242–247.

[13] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S.: PyTorch: An Imperative Style, High-Performance Deep Learning Library, *Advances in Neural Information Processing Systems*, Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R.(eds.), Vol. 32, Curran Associates, Inc., 2019.

[14] Podelski, A. and Rybalchenko, A.: A Complete Method for the Synthesis of Linear Ranking Functions, *Verification, Model Checking, and Abstract Interpretation, 5th International Conference, VMCAI 2004*, Steffen, B. and Levi, G.(eds.), Lecture Notes in Computer Science, Vol. 2937, Springer, 2004, pp. 239–251.

[15] Tan, W. and Li, Y.: Synthesis of ranking functions via DNN, *Neural Computing and Applications*, Vol. 33, No. 16(2021), pp. 9939–9959.

[16] Unno, H., Terauchi, T., Gu, Y., and Koskinen, E.: Modular Primal-Dual Fixpoint Logic Solving for Temporal Verification, *Proc. ACM Program. Lang.*, Vol. 7, No. POPL(2023).