連載

5分で分かる!? 有名論文ナナメ読み

但馬将貴（日本電信電話（株）ソフトウェアイノベーションセンタ）

Matej Balog et al. : DeepCoder : Learning to Write Programs

（ICLR2017）

プログラム合成とその課題

背景

ソフトウェア開発においてプログラムの作成は非常に工数のかかる工程であり，省力化が求められている．プログラム作成の工数削減を目指した研究には，モデル駆動開発のように，実現したい仕様を記述したモデル図からプログラムを自動生成する技術があり，最近ではローコード開発ツールとして企業における活用も進んでいる．しかしモデル駆動開発は，モデルの形式やモデルからコードへの変換ルールをあらかじめ厳密に定義した上で仕様に基づいて詳細なモデルを作成する必要があり，ルール設定とモデル作成のコストがかかるという課題が存在する．今回紹介する「プログラム合成」は，あらかじめ用意したプログラム部品を，仕様情報を満たすように自動で組み合わせることでプログラムを自動生成する技術であり，モデル駆動開発のような変換ルール設定やモデル作成が不要という利点がある．プログラム部品には，プログラムを構成する式や関数などがある．仕様情報には，自然言語で書かれた仕様や入出力例などがあるが，マシンが扱いやすい形式である入出力例がよく用いられている．プログラム合成の具体例を図 -1 に示す．ここでは，開発者は「入力として与えた整数列について，0 より大きな数のみ 4 倍して昇順としたものを出力とする」という仕様のプログラムを作成したいと考えており，そのような仕様を満たす入力と出力の組合せを入出力例（A）としてプログラム合成技術へ与える．プログラム合成技術は，用意された関数一覧（B）から，（A）を満たすように関数を自動で組み合わせ，生成プログラム（C）を出力する．また，（C）のように特定の用途向けに記述される言語を DSL（ドメイン特化言語）と呼び，この例では整数列の操作に特化した関数セットの組合せで記述される言語となっている．プログラム合成では，このような入出力例を満たすプログラム（ここでは関数の組合せ）を，次のような探索手法によって求めてきた．

従来の探索手法とその課題

深さ優先探索

深さ優先探索では，初めにプログラム中で最初に使用する関数をランダムに決定する．その後，事前に決めたプログラム長（プログラムを構成する関数の個数）以下となるような関数の組合せをすべて試行する．入出力例を満たすプログラムが得られなかった場合，別の関数を最初に使用する関数に決定し，再び探索を行う．しかしこの手法では，関数の

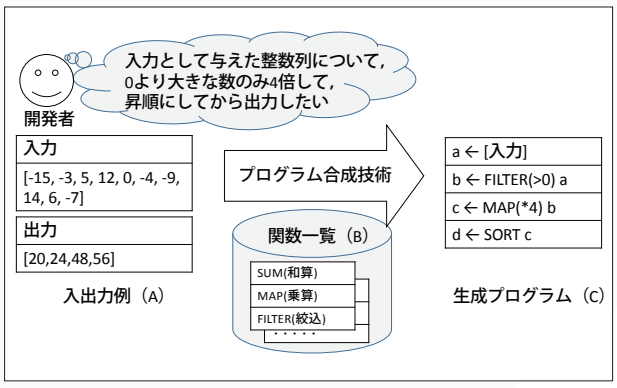

図 -1　入出力例を用いたプログラム合成の具体例

組合せを全探索しなくてはならないため，合成に膨大な時間がかかるという課題がある．

ソートと加算

深さ優先探索の課題を解決するために，ソートと加算という手法がある．ソートと加算では，深さ優先探索において合成を行う関数を一部に限定（これをアクティブセットという）して探索する．探索に失敗するたびに，アクティブセット中の関数群を合成に失敗した数の少ない関数群に更新することで，使用される確率が高いと予想される関数群の中から合成を行うことができる．しかしこの手法は合成に失敗した際の情報を使用しているため，適切なアクティブセットが作れるようになるまで時間がかかるという課題がある．

深層学習によるプログラム合成（提案手法）

特徴

提案手法である DeepCoder は，入出力例からプログラム中に使用される確率の高い関数を，深層学習モデルを使って予測することでプログラム合成の高速化を実現する．DeepCoder は，既存のあらゆる探索手法において合成する関数を選択するフェーズに適用できる．2021 年現在，プログラム合成に深層学習を導入した研究は数多く存在するが，DeepCoder はそのきっかけとなった研究の 1 つであり，機械学習・AI 分野のトップ国際会議 5th International Conference on Learning Representations（ICLR2017）で発表された後，数多く引用されている．

アルゴリズム

学習フェーズ

学習フェーズでは入出力例のパターンと使用する関数の関係性を学習する．図 -2 に示すように，プログラムの複数の入出力例を入力とし，そのプログラム中で使用されている関数の有無（ありの場合は 1，なしの場合は 0）を出力とするように学習する．そのように学習することで，推論時には使用する関数を 0 ～ 1 の範囲で予測することができ，その値を関数の使用確率と捉えることができる．学習データとなる入出力例とプログラムは次の方法で用意する．初めに，DSL の文法規則に基づき，DSL の文法を満たすプログラムを列挙する．次に，冗長な変数や，等価なプログラムを切り捨てる．そして，あらかじめ決められた入出力の範囲内で，具体的な入出力例を 500 個生成する．

合成フェーズ

合成フェーズでは探索手法に DeepCoder を適用する．入出力例を与えると，DeepCoder は関数の使用確率を予測する．深さ優先探索では，プログラム中で最初に使用する関数を決定する際と，関数の組合せを試行する際に，使用確率が高い関数を選ぶ．ソートと加算では，アクティブセットの関数を更新する際に，使用確率が高い関数を選ぶ．このように DeepCoder はあらゆる探索手法において，関数セットの中から関数を選択する手法に適用できる．

評価結果と今後の展望

紹介論文では，評価方法として，従来の探索手法に DeepCoder を組み込み，合成がどれくらい高速化されるかを測定している．評価結果として，深さ

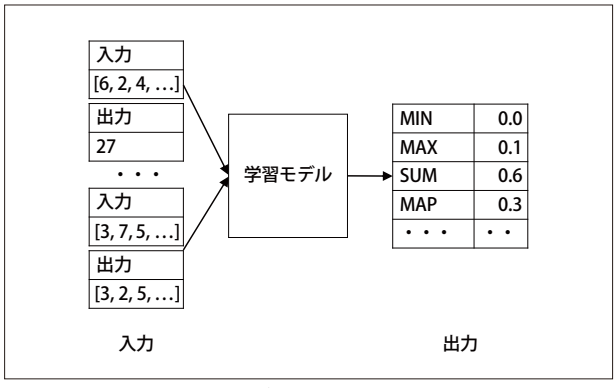

図 -2　DeepCoder の学習モデル

優先探索では約 3 倍～ 15 倍，ソートと加算では約 31 倍～ 62 倍となっている．その他の探索手法にも適用して評価をしており，最大で約 467 倍（いずれもプログラム長：3 の探索）の高速化を実現している．このように，DeepCoder では既存の探索手法の大幅な高速化を実現できたが，使用している関数の種類は 34 個であり，これに含まれない関数を用いた複雑なプログラムは表現できない．複雑なプログラムを生成するためには，より多くの関数が必要になる．関数の数が増加するほど，関数の使用確率の予測が困難になるだけでなく組合せの試行にも時間がかかるため，関数の使用確率および使用順番等も含めた予測が必要と考えられる．しかし，入出力例のみから関数の使用確率および使用順番等を含めて予測することは難しい可能性があるため，今後は入出力例だけでなく自然言語で記述されたドキュメント等の仕様情報も活用することが，効率的なプログラム合成に有効だと考えられる．

DeepCoder は，深層学習を用いることでプログラム合成を大幅に高速化できることを示した．DeepCoder のコンセプトを発展させ，GitHub 等で公開されている世界中の膨大な量のプログラムを学習することで，あらゆる種類のプログラムが自動生成できるようになる可能性がある．人が作ってほしいプログラムのイメージを与えるだけで，プログラムを自動生成してくれる．このような未来はそう遠くはないかもしれない．

（2021 年 3 月 18 日受付）

但馬将貴
masaki.tajima.zx@hco.ntt.co.jp
2017 年信州大学大学院理工学系研究科情報工学専攻修士課程修了．同年東日本電信電話（株）入社．2019 年 NTT ネットワークサービスシステム研究所．2020 年 NTT ソフトウェアイノベーションセンタ．主に自動プログラミング，テスト自動化などのソフトウェア開発技術に関する研究開発に従事．