形式手法とプログラミング言語の型

比 嘉 健 太[†1] 河 野 真 治[†1]

任意のプログラムに対して形式手法が適用可能であるべきであると考える．そのために，全てのプログラムへ形式手法が対話的に実行できる処理系を考えたい．その処理系が実行するプログラムに記述するべき情報量とプログラムに対する制約の強度について興味がある．

Formal Methods and Types of Programming Languages

Yasutaka Higa[†1] and Shinji Kono[†1]

I think any program must be check by formal methods. I want to programming language executer that applicative formal methods to any program interactively. I interest constraints of program and amounts of information that enough for formal methods.

1. はじめに

形式手法によるチェックはプログラム実装言語の処理系とユーザ間で対話的に行なわれるべきだと考える．任意のプログラムに対して形式手法が適用可能であり，必要ならばユーザが制約をさらに厳しくする．制約の追加が対話的入力であり，反例の指摘などが対話的出力に相当する．

対話的に実行する理由は，必要とされる仕様の厳密さに柔軟に対応するためである．例えば，使い捨てのプログラムなどに厳密な仕様を策定するのはコストが高すぎる．しかし非常に単純なバグを自動で見つけるために形式手法を利用したい．そのバグのみを見つけるための条件を対話的な出力とし，ユーザが入力をする．他にも，徐々に仕様が複雑になり整理したい，といった場合なども対話的に必要な分だけ対応したい．そのためにも仕様が存在しないような任意のプログラムに対しても形式手法を実行したい．

そのような対話的チェックを実現するには，実装言語が通常の処理系に加えて形式手法をサポートする必要があると考える．

2. 形式手法をサポートした言語

任意のプログラムに対して形式手法を適用可能な処理系は作成できると考える．なぜなら，アプリケーションの仕様に関わらず実行してはいけない処理などが存在するからである．

例えば，0による除算や配列外へのアクセスはどのようなプログラムにおいても実行してはいけない．よって，全てのプログラムが共通に持つべき仕様として定義可能である．処理系が仕様のチェック機構を持つことにより，全てのプログラムに対して形式手法が導入可能となる．仕様はユーザ定義の満たすべき条件として定義する．基本的な満たすべき条件の拡張として記述することにより，仕様が存在するプログラムにも対応することができる．

ここで，プログラム実装言語が自動で仕様をチェックするために必要な情報を考えたい．どれだけの情報をプログラムに記述すれば自動でチェックできるかは私の中ではまだはっきりしていない．仕様をチェックするに足る情報の量と，その情報から仕様をチェックする手法に興味がある．

3. 型と証明支援系言語

現在，私がプログラムに追加する情報として注目しているのが型である．型と論理は対応しており，証明が存在する定理は型によって表現できる．満たすべき性質を証明として型が記述できれば，その型は性質を満たすと言える．仕様を全て論理で記述し，対応する型を記述すると仕様を満たすプログラムが記述できると思われる．しかし，論理を満たす型の記述には非常に大きなコストがかかる．

私は研究において関数型プログラミング言語 Haskell

†1 琉球大学工学部情報工学科
Department of Information Engineering, University of the Ryukyus.

と証明支援系プログラミング言語 Agda を利用している．プログラムの変更を表すことのできるデータ型を用意し，異なるバージョンのプログラム間の関係を形式化する試みである．この研究において，Haskell でデータ型とそのデータ型への処理を 50 行ほど記述した．そのデータ型と処理が満たすべき条件を Agda で記述すると 500 行ほどとなった．行換算すると実装に対して仕様には 10 倍のコストがかかっている．Agda では非常に単純な性質は自動で導くが，ほとんどの性質は自ら記述する．Agda が行なうのは，性質を満たしているかのチェックと満たすべき性質が必要とする性質の指摘である．Agda によって性質を記述すると，性質の詳細や依存関係は明確になるが，性質を直接解くコストは必須である．

個別のプロジェクトにおいて個別のデータ型を定義し，満たすべき性質を記述するのでは非常に大きなコストがかかってしまう．よって，既に証明された型どうしを組み合せることによってプログラムを記述するのが望ましいと考える．そこで問題となるのが，証明された型のみで任意のプログラムを記述可能なのか，という点である．これは，型がプログラムに加えている制約の強さの問題でもある．任意のプログラムに対して形式手法が適用可能な処理系を考えた際に，どの程度の制約が必要なのかも考えなくてはならない．

4. ま と め

私は形式手法は実行可能なプログラムに対して処理系と対話的に行なうべきものだと考えている．仕様が定まっていなくとも満たすべき条件は存在し，それはどのプログラムにも共通と言って良い．さらに，具体的な仕様が決まるのならば拡張として定義していく．

対話的に形式手法を実行するためには実行可能プログラムの処理系が形式手法をサポートする必要がある．そのためにプログラムに加えるべき情報と制約に興味がある．特に，現在は型の情報と制約について調べている．私としては依存型は情報量は十分だが制約が強すぎ，線形型では情報量が不十分なのでは無いかと考えている．