形式手法を学び始めて思うことと，形式手法を広めるには

比 嘉 健 太[†1]　河 野 真 治[†1]

1. 現在行なっている研究

プログラムの信頼性の向上は自動で行なわれるべきだと思っています．形式化された範囲で作成されたプログラムは，人による性質の記述無しに高い信頼性を持つのが理想だと考えています．

現在私はプログラムの変更を形式化する研究を行なっています．研究目的はプログラムの変更を形式化することによりプログラムの変更時に完成に近づいているかを判断することです．現在は特定の二点間での信頼性に注目し，2 つのバージョン間でのトレースの相違を指摘しようとしています．

私の研究においてプログラムの変更は Monad として表現します．Monad とは，圏においては Monad 則を満たす自然変換 μ と η と Functor T です．Monad 則を満たすような μ と η と Functor T として，プログラムの変更を表します．

例題の記述にはプログラミング言語 Haskell を使用しています．まず，プログラムの変更を表すようなデータ構造 Delta を定義します．Delta は過去のバージョンの値を持ち，バージョンが上がるたびに値を増やすようにします．プログラムが変更されても過去の値を持ち続けることにより，プログラムの変更を表す試みです．Delta は複数のバージョンの値を持っているため，計算の時にどのバージョンの値と処理の対応を判断しなくてはなりません．バージョンの判断に Monad を使いました．Haskell における Monad はデータ構造とメタな計算との対応付けです．Delta と Delta に対する関数は，互いにどのようなバージョンを持っていても一意になるよう， μ と η を定義しました．一意になることの確認には Monad 則を満たすことにより確認できます．Monad 則を満たす証明のためには証明支援系言語 Agda を用いました．

現在進んでいるのはここまでで，これから二点のトレースの比較をしようと思っています．

2. Agda を学んだ流れとつまづき

形式化や Agda は三年次向けの講義で知りました．もともと形式手法を知らず，バグを自動で検出したかった私には関心の高い講義でした．その講義では Curry-Howard 対応や Agda における証明を行ないました．

型や自然演繹は直感的に理解できました．型の変換は値の変換に応じて型が変わるため，関数によって値を変えることだと思えたからです．自然演繹もルールが数個しかなく，理解するのにそれほど時間はかかりませんでした．しかし，Agda による証明記述はすぐには理解できませんでした．依存型を持つ Agda では，証明は型で記述されます．証明を満たす型を持つ値，というのが直感的に理解できませんでした．

Agda の特徴を取らえられたのは System T や Sytem F を Agda で記述していた時です．四年次になり研究室に配属された私は，プログラムのバグを自動で検出したいと希望しました．そこで，Proofs and Types[?)] を読むことになりました．Proofs and Types ではもう一度 Curry-Howard 対応や自然演繹，それに加えて System T や System F などを学びました．System T における自然数の加法の結合法則を証明を記述している時，等式変形に加法の交換法則を用いました．この時に，規則とそれから導出される証明も規則であり，それらによる等式を変形で証明を記述しているのだと理解しました．理解するまでには Agda で証明を書き続ける必要がありました．

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

また，Agda の agda-mode で実行するたびに必要な証明が指摘されているのを見た時，対話的に実行することのメリットを感じました．この証明を変形するにはどの証明が必要か，などと何度も試せるからです．試していくうちに必要な他の証明が見えてきます．他の証明が必須だと思った時に新な証明を記述していくようにしました．こうすることにより，証明すべき大きな式は見失わずに必要な小さな式に分割することができます．対話的に実行することで問題の変換や詳細化や分割などが行なえるため対話的実行は有用だと思っています．

3. 形式手法を広めるには

形式手法を広めるには対話的な手法を導入するのが良いと思っています．私は Agda を書き続けることで依存型に対する理解を得られましたが，書き続けるには時間が必要にです．理解できている部分だけ実行しつつ，理解できない部分はどうするべきか対話的に聞くことで必要な情報を速く得ます．必要になった部分だけ学習することで，実行したい部分に対して必要十分な理解が得られると思います．このことにより，形式手法を学習する時間は必要な部分だけに納まり，より試しやすいものになると思います．

また，必要な分だけを実行するのは形式手法の学習コスト以外にもメリットがあると思います．必要な分だけを検証することにより，過剰な検証コストを防ぐことができます．検証の範囲と必要な条件が対話的に得られるのであれば，適切なコストで検証を行なうこともできると思います．