抽象解釈に基づく段階的プログラム構成法のための環境構築に関する研究

石川俊

北陸先端科学技術大学院大学 情報科学研究科

平成10年 2月 13日

キーワード: 段階的詳細化, 抽象解釈, インクリメンタル・プログラミング, 開発環境.

ソフトウェアの規模が大きくなると、仕様決定の遅れやバグの存在によってプログラム全体を通して実行することが難しい。この原因の1つは、これまでのソフトウェア開発の手法では、まずその仕様を細部まで決定した後でないとプログラムを構築できないためである。

この問題を解決するために段階的詳細化の手法がいくつか提案されている。その一つとして吉岡らによってISDR法(Incremental Software creation methodology based on Data Reification)が提案されている。

ISDR法では、データの集合を抽象的な値の集合と捉え、その抽象値を段階的に具体的な値に変化させ、その変化に従ってソフトウェアを詳細化する。

ISDR法によってソフトウェアを段階的に詳細化する手順をまとめると次のようになる。

- **抽象化段階:** 原始プログラムを作成する
  1. もっとも原始的な入出力データをつくる。
  2. 抽象化したデータに基づいて仕様を作成し、それに対するプログラムを定義し、実行する。
- **詳細化段階:** 完全なプログラムになるまで以下の過程で繰り返し詳細化する。
  1. 抽象的な入出力データを一段階具体化する。
  2. 上の段階で具体化したデータを使ってプログラムを詳細化する。

ISDR 法ではソフトウェアの詳細化途中の段階で作成したプログラムの解釈を形式的に定義することによってプログラムの一部がまだ定義されていない未完成なプログラムでも実行可能であるという特徴を持っている。しかし ISDR 法に基づいたプログラムの実行、開発環境は実現されていない。

本研究では ISDR 法に基づいたプログラムの開発と実行を支援する環境を実装する。そして、この環境を使って実際にプログラムを作成し、ISDR 法によるソフトウェアの段階的構築の有効性ついて考察する。

この環境を実現するために ISDR 法に基づいたプログラム言語 AL の処理系を実装し、AL によるプログラムの作成を支援する環境 Alchemy を実装する。

AL は ML に抽象実行の機能を追加した言語でバージョンとデータドメインの概念を持ち、プログラムを複数の段階に分けて構成することができる。また抽象値とその段階的な具体化にあわせて各段階で関数を定義する。AL の処理系は ISDR 法に基づき、未完成なプログラムであっても実行することができる。抽象解釈に基づく ISDR 法のプログラムの解釈は関数の詳細化関係に基づいて最新の関数を呼び出し、静的に推論した型情報に基づき実引数や返り値を抽象化する。

Alchemy はプログラム作成のためにリビジョンの管理、データドメインの定義と具体化、プログラムの編集といった機能を提供する。また、作成したプログラムの実行や情報を得るために、AL 処理系とのユーザーインターフェース、詳細化の方針を示すツール、テンプレートを示すツール、といった機能を提供する。

ISDR 法では一連の詳細化でプログラムが完成することを仮定している。詳細化途中で仕様が変更された場合には該当するバージョン以降の定義をすべて破棄しなければならない。しかし該当するバージョンから分岐させることで、いままでの定義を残したまま詳細化をやり直すことができると考えられる。これに対応して Alchemy は ISDR 法のバージョンの概念を拡張して木構造となるバージョンの分岐を可能にしている。この木構造で管理される各詳細化の段階をリビジョンと呼ぶ。一般の Revision Control System はチェックインしたプログラムについての記述を完全に固定して保存するものであるため、過去の記述について変更することはできない。しかし Alchemy におけるリビジョンの概念はプログラムを各詳細化段階に分けることを目的とする。そのため各段階の記述を固定することはせずに、任意のリビジョンに対してデータや関数の定義を操作することを可能とした。定義を操作するときは ISDR 法がデータの具体化と関数の詳細化に関して与えているいくつかの制約に沿って行う必要がある。Alchemy では詳細化が済んでいるリビジョンに対する操作が可能であるので、より詳細度の高いリビジョンの記述に対して矛盾が起きないように、より強い制約を加える必要がある。本研究ではリビジョンのデータや関数に対する各種操作に対してどのような制約を与えるか検討し、これを実装する。実際にはユーザーが制約を破るような操作を行おうとした場合に、その操作を無効とすることで制約を満たす。

最後に、AL と Alchemy を使ってある程度の規模を持ったプログラムを実際に作成することで、ISDR 法によるソフトウェアの段階的構成の有効性について考察する。

AL で作成したプログラムは一般的にかなり冗長になる。しかし、データの具体化に対応して詳細化した関数は前段階でも同じ領域のデータについて詳細化されているので、似通った構造になる傾向がある。。Alchemy ではプログラムコード編集中に任意のバージョンのプログラムコードを参照する機能がある。これを利用することで、ユーザーが実際に記述するコード量を削減する。