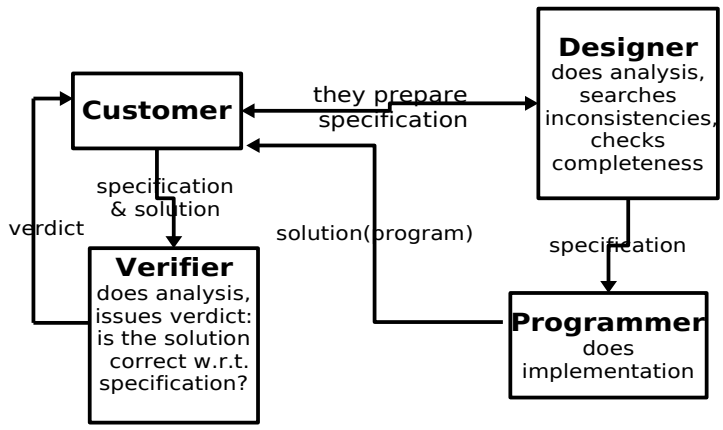
Logika Algorytmiczna

czyli

Rachunek Programów

dla celów specyfikacji i weryfikacji oprogramowania

Niewiele osób dziś myśli, że programiści piszą bezbłędne programy. Jednak w dalszym ciągu większość wierzy w magię, w zaklęcia... Jak bowiem nazwać wiarę w to, że program P nie spłata psikusa (lub gorzej!), wiarę wynikającą z tego, że program ten był testowany? Przypuśćmy, że firma producent uruchomiła ten program 1000 lub 10000 razy na pewnych danych. Nic złego się nie stało. Skąd użytkownik programu czerpie wiedzę, że program P zakończy obliczenia i da dobry wynik za każdym razem, dla jego danych? Nie ma takich gwarancji. A czy mogą być? Owszem, gwarancje mogą być dostarczane razem z oprogramowaniem. Do tego by zrealizować takie zadanie trzeba znać i stosować rachunek programów.



Do czego ten rachunek może służyć?

Do tworzenia *specyfikacji* jako zbioru formuł algorytmicznych. Taka specyfikacja jest bogatsza od Javowego interface właśnie o te formuły.

Do zapisywania argumentów powstających w procesie *weryfikacji* jako dowodów przeprowadzanych na gruncie logiki algorytmicznej = rachunku programów.

Skąd bierze się pewność?

Ponieważ, własności semantyczne programów możemy zapisywać jako formuły algorytmiczne i ponieważ prawdziwe formuły posiadają dowody (uwaga, te dowody różnią się od dowodów na gruncie logiki pierszego rzędu!)

Co znajdę w tym repozytorium?

- Monografie na temat logiki algorytmicznej.
- Zbiór artykułów.
- Zbiór przykładów.
- Artykuły wprowadzające do projektu SpecVer – zastosowanie AL.

Czy istnieją lepsze narzędzia?

W tym samym czasie co logika algorytmiczna lub później, powstały inne logiki pogramów: systemy algebr algorytmicznych (Glushkov 1965), logika Floyda-Hoare'a (1967-69), rachunek najsłabszego warunku wstępnego (Dijkstra 1975), logika dynamiczna V. Pratta (1976).

Porównanie tych formalizmów z logiką algorytmiczną daje następujące wyniki: logika Hoare'a jest fragmentem AL, rachunek Dijkstry zawiera błędy, jest bliski AL, logika dynamiczna ma dwie wersje: w pierwszej przyjmuje jako aksjomaty wszystkie formuły prawdziwe w strukturze liczb naturalnych, w drugiej wzoruje się na aksjomatyzacji AL podanej przez G. Mirkowską, natomiast rachunek Glushkova nie jest kompletny.

Porównując zasięg *zastosowań* otrzymujemy następujące informacje: logika Hoare'a - kilkadziesiąt dowodów niedużych programów, rachunek Dijkstry - nieco mniej udowodnionych twierdeń o programach, logika dynamiczna –

znamy jeden dowód poprawności programu obliczającego x^n . Posługując się AL potrafimy podać dowód kilkunastostronicowego programu.

Ponadto, AL jako jedyna sprawdziła się jako narzędzie specyfikacji struktur danych. Więcej, jedynie AL cieszy się twierdzeniem o aksjomatycznej definicji semantyki języka programowania (programów iteracyjnych).

Ile to jest warte?

Policz sam: większość kosztów oprogramowania bierze się z konieczności jego utrzymania (*ang.* maintenance). Zapewne możesz oszacować obroty sprzętem. Powiedzmy, ze jest to 10 miliardów euro rocznie. A oprogramowanie to obroty wielokrotnie większe – powiedzmy 50 miliardów euro. Z tej kwoty odejmij 40%. To jest tort , z którego możesz odcinać ciastka dla siebie. O ile będziesz umiał dowodzic popraność oprogramowania względem specyfikacji.

W edukacji? Czy nie powinniśmy przedyskutować na nowo curriculum studiów informatycznych?

W tworzeniu nowych (dobrze płatnych) zawodów? Popatrzmy na diagram. Juz powstają firmy specjalizujące się w tworzeniu specyfikacji oprogramowania, które ma powstać. Troche trudniej zlokalizować firmę, która podejmie się weryfikacji produktu programistycznego względem zamówionej specyfikacji. Ale czas pokaże, że firmy zamawiające oprogramowanie zaczną postępować podobnie do inwestorów budujących duże budynki. A więc potrzebni będą: architekci specyfikacji, budowniczowie systemów programistycznych spełniających warunki wymienione w specyfikacji i audytorzy (sprawdzający czy produkt programistyczny jest poprawną implementacją oprogramowania).

Na ile lat dajecie gwarancję poprawności związaną z dowodem : na zawsze.

Co dalej?

Mamy kilka pomysłów na kolejne projekty badawcze wykorzystujące AL:

- SpecVer – ten projekt ma na celu wytworzenie narzędzi wspomagających pracę inżynierów oprogramowania w różnych jej fazach:
  - formułowanie specyfikacji,
    Specyfikacje są załącznikiem do umów cywilno-prawnych o wytworzenie oprogramowania.
    Taki załącznik nie może zawierać wymagań sprzecznych.
    Ponadto, specyfikacja niekompletna naraża stronę zamawiającą oprogramowanie na koszty (niemałe) wynikające z konieczności rozpoczęcia prac od nowa.
  - weryfikowanie zgodności produktu programistycznego z jego specyfikacją,
    Wiadomo, że proces ten nie będzie nigdy zautomatyzowany (tw. O nierozstrzygalności problemu stopu). Proponujemy wytwoerzenie narzędzi wspomagających ludzi w tym procesie. M. in. języka i proof-checkera (tj. programu sprawdzającego czy przedstawiony tekst jest dowodem poprawności programu).
  - Testem tego projektu może być stworzenie biblioteki klas na wzór biblioteki STL, z tą różnicą, że każda klasa miałaby: 1° swoją specyfikację, 2° kod źródłowy i 3° dowód poprawności kodu względem specyfikacji.
- Projekt (na razie bez nazwy) zmierzający do skodyfikowania wiedzy o algorytmice i matematyce dyskretnej w sieci komputerów.
  Podobna idea przyświeca Donaldowi Knuthowi w jego „*The art of programming*”. Od czasu gdy Knuth rozpoczął pisanie swej monografii upłynęło prawie 50 lat i dokonał się ogromny postęp.
  Trudno dziś zgromadzić w jednym dziele całą znaną nam wiedzę o algorytmice.
  Ponadto, w proponowanym projekcie można zawrzeć nowe fakty. Wierzymy bowiem głęboko, że zastosowanie AL pozwoli rozwinąć algorytmiczne teorie wielu dziedzin matematyki dyskretnej jako teorii algorytmicznych, opartych na AL, zamiast na rachunku predykatów, np.
  - algorytmiczną teorię liczb naturalnych (w tej teorii mieszczą się dowód twierdzenia o stopie algorytmu Euklidesa wyprowadzony z aksjomatu (algorytmicznego) być liczbą naturalną
    $$\forall\, x\in\mathbb{N}\{\, y:=0\,;\textbf{\textit{while}}\; x\neq y\;\textbf{\textit{do}}\; y:=y+1\;\textbf{\textit{od}}\,\}\, x=y \quad ,$$
    jak również wniosek z ostatniego twierdzenia Fermata stwierdzający, że pewien program nie kończy obliczeń, itp.
  - algorytmiczną teorię drzew binarnych poszukiwań,
  - algorytmiczną teorię ciał Archimedesowych, a w niej formalne dowody poprawności wielu algorytmów analizy numerycznej,
  - etc.

Dołącz do nas! Przekaż te informacje dalej! Sprzedaj swoją wiedzę o AL!

mailto: g.mirkowska@uksw.edu.pl , a.salwicki@uksw.edu.pl