Program Termination

http://d3s.mff.cuni.cz

Department of Distributed and Dependable Systems

D3S

Pavel Parízek

FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

Introduction

- Task: determine correctly whether a given program will **always finish** (for all inputs)
  - or give the answer ***"unknown"***

- Practical motivation
  - Infinite execution of some loop may result in a non-responsive operating system
  - Common userspace applications get often stuck

Literature

- B. Cook, A. Podelski, and A. Rybalchenko. **Proving Program Termination**. Communications of the ACM, 54(5), 2011.
- B. Cook, A. Podelski, and A. Rybalchenko. **Termination Proofs for Systems Code**. PLDI 2006, ACM.

Example program

```
x := input()
y := input();
while x > 0 and y > 0 do
  if input() = 1 then
    x := x - 1;
    y := y + 1;
  else
    y := y - 1;
  fi
done
```

Taken from B. Cook, A. Podelski, and A. Rybalchenko.
Proving Program Termination. Comm. of the ACM, 54(5), 2011.

Terminology

- Transition relation R
  - $R \subseteq S \times S$, $(s,s') \in R$ iff $s \rightarrow s'$

- Termination argument
- Well-order relation
- Ranking function f
- Well-founded relation
  - $T = \{ (s_1,s_2) \mid f(s_1) > f(s_2) \}$

- Disjunctive termination argument
  - $T = T_1 \cup T_2 \cup \ldots \cup T_N$

Current state of the art

- What can be proved (disproved)
  - Famous complex problems: Ackermann's function
  - Industrial examples: Windows device drivers
    - Prover: **Terminator (T2)**
  - Sequential programs that use arithmetic expressions
- Research challenges
  - More complex programs (dynamic allocation, threads)
  - Processing non-linear arithmetic operators (*,/,%)
- Other applications: checking liveness properties

Terminator (T2)

- Termination prover
  - Developed by Microsoft Research (B. Cook et al.)

- How it works
  - Iterative proving based on abstraction refinement

- Source code freely available
  - http://mmjb.github.io/T2/
- Implementation languages: F#, ML

Dafny

- Simple language and program verifier

- Important features
  - Contracts: precondition, postcondition, invariant
  - Program termination analysis

- Resources
  - https://github.com/dafny-lang/dafny
  - http://dafny.org/dafny/

Dafny: Example 1

```
method Compute(x: int, y: int, z: int) {
  var x1: int := x;
  var y1: int := y;
  while (x1 > 0 && y1 > 0) {
    if (z == 1) {
      x1 := x1 - 1;
      y1 := y1 + 1;
    }
    else {
      y1 := y1 - 1;
    }
  }
}
```

Dafny: Example 2

```
method Compute(n: int) {
   var x: int := 0;
   while (x < n)
   {
      x := x + 1;
   }
}
```