Slim Programma's Opsommen

KU LEUVEN

Gemaakt door Kobe Van der Linden, Begeleid door Tom Schrijvers

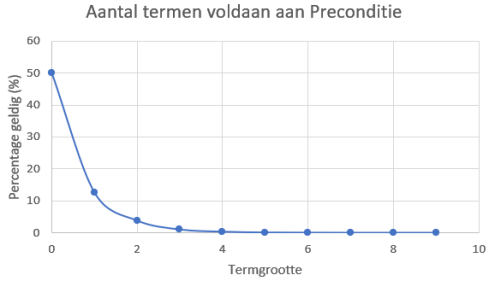
Introductie

Voor het ontwikkelen van programmeertalen moeten **eigenschappen** bewezen worden.

- Manier: **Property Based Testing**
- Input: **willekeurig programma's**

Motivatie

Opsommen van willekeurige programma's bestaat al maar is **inefficiënt** omdat veel programma's niet aan de **preconditie** voldoen.



Aanpak

We schrijven de regels van de preconditie in een **declaratieve vorm**. Deze regels nemen we op in onze constructor.

Bv.

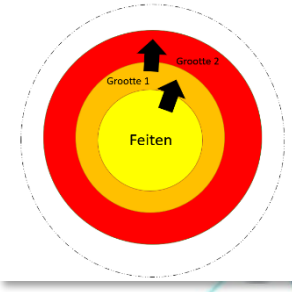
```
Even(succ(succ(X))) -: Even(X))

Even(Zero).
```

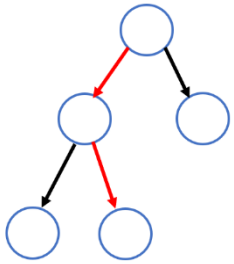
Methode

We onderscheiden 3 methodes:

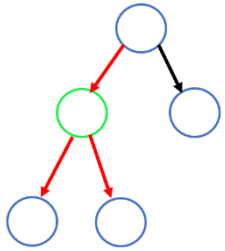
Bottom-up



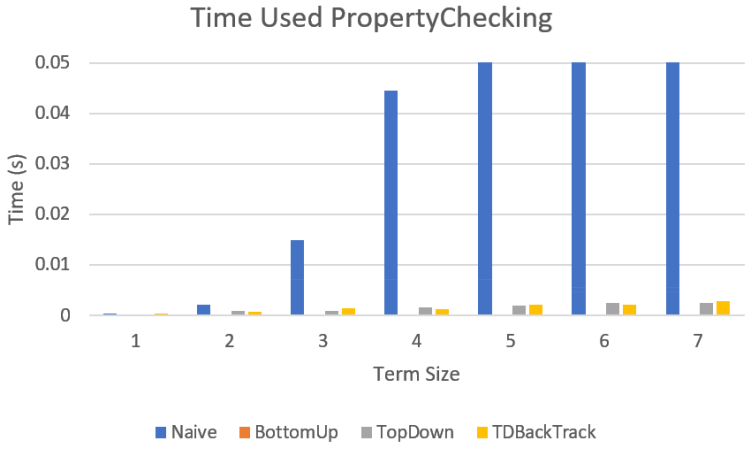
Top-down



Top-down met BackTrack



Resultaten



We hebben de 3 methodes samen met de naïeve manier getest op 3 verschillende fouten. Hierbij bekeken we hoelang het duurde om een tegenvoorbeeld te genereren. Bij TDBackTrack backtracken we de eerste node.

Conclusie

We zien dat alle 3 de gebruikte methodes **sneller** werkte dan het **naïeve** geval.

Verder zien we **weinig** verschil met Top-Down en Top-Down met backtracking (in een bepaalde node). De oorzaak ligt waarschijnlijk dat er maar weinig regels zijn en dus de kans om het juiste pad te kiezen groot is.

Bottom-up kon maar maximaal termen van grootte **2** genereren. Hierna werd het aantal mogelijke programma's te groot .

Referenties

S. CERI, G. GOTTLOB, L. TANCA, What You Always Wanted to Know About Datalog (And Never Dared to Ask)

G. Coremans, Een codegenerator voor het opsommen van programma's

K. Claessen, J. Hughess, QuickCheck: A Lightweight Tool for Random Testing of Haskell Programs