Slimme Opsomming van Programma's

KU LEUVEN

Gemaakt door Kobe Van der Linden, Begeleid door Tom Schrijvers

Introductie

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

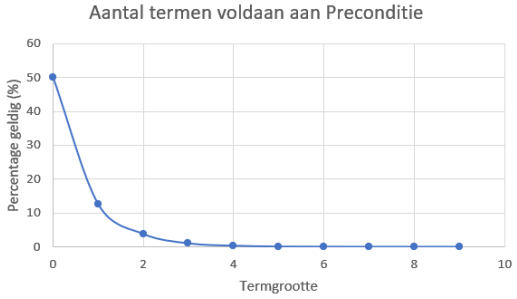
Dit doen we door **Property Based Testing.** Hierbij gaat een **generator** willekeurige termen van het gevraagde type genereren en testen op de property. Zo probeert hij een tegenvoorbeeld te genereren. Vaak komen hier **precondities** in voor, dit zijn eigenschappen van volgende vorm:

Conditie1 ⇒ Conditie2

Preconditie

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 **generator** zodat alle termen aan preconditie voldoen. Onderstaand voorbeeld geeft een generator voor even getallen.

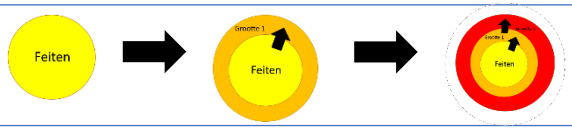
Even(succ(succ(X))) -: Even(X))

Even(Zero).

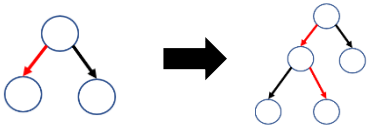
Methode

Hieronder staat een beschrijving van 3 generator algoritmes voor termen te generen die altijd aan de preconditie voldoen.

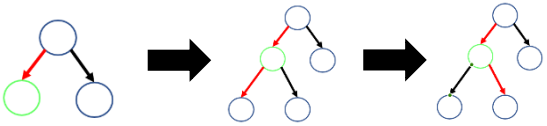
Bottom-up



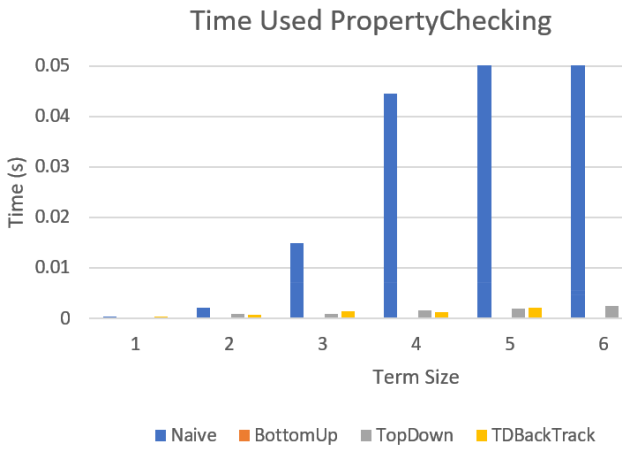
Top-Down



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 back tracken we steeds de eerste node. Hier zie je de resultaten van 1 van de testen, omdat de **fout** al met relatief **kleine termen** kon gevonden worden vindt bottom-up heel snel een tegenvoorbeeld die niet aan de property voldoet.

Conclusie

De nieuwe aanpak om de preconditie op te nemen in de generator werkt dus wel degelijk **sneller** .

Verder zien we **weinig** verschil met Top-Down en Top-Down met backtracking (in een bepaalde node). Een mogelijke oorzaak is dat er maar weinig regels zijn gebruikt in de testen 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