Datalogiens videnskabs teori. Individuel opgave 1

Sebastian Ostenfeldt Jensen GJX 653

4. december 2014

1 Program Analysis

(a) En turingmaskine kan bruges til at lave all tænkelige automatiske beregninger [1]. Et turingkomplet programmeringssprog er et sprog der kan simulere en turingmaskine. Alle beregninger som kan laves på en turingmaskine kan derfor også laves med et turingkomplet programmeringssprog. De fleste imperative sprog inklusiv JAVA er turingkomplette [2]. At analysere et WHILE program for variabler der ikke bliver læst er et problem der kan beskrives som en automatisk beregning. Man kan evaluere programmet fra start til slut og for vær konditional statement, analysere de mulige logiske udfald af logiske statements og derefter analysere de mulige branch programmet kan eksekvere for variabler. Alle variabler der ikke er indeholdt i de mulige branch kan fjernes fra WHILE programmet.

(b) Der findes specialiserede programmeringssprog som ikke er turingkomplette. Det vil sige at de ikke kan simulere en turingmaskine og derfor ikke nødvendigvis kan løse en vær automatisk beregning. F.eks. er HTML et specialiseret sprog som ikke har konditional statements basseret på logiske statements. HTML vil derfor ikke kunne bruges til at løse opgaven i a.

(c) Ud over variabler der ikke bliver læst vil det selvfølgelig også være relevant at fjerne andre program stumper som aldrig vil blive eksekveret samt andre optimeringer der for programmet til at køre hurtigere eller bruge mindre hukommelse. Det. Kan f.eks. være matematiske beregninger eller logiske udtryk der kan blive optimeret. Betragt det logiske udtryk

$$A\&(A|B)$$

Det vil altid vil evaluere til A uanset hvad B måtte være. Uden optimering vil det være sandsynlig at computeren først vil analyserer OR udtrykket og derefter And udtrykket. En compiler vil derfor normalt ændre udtrykket til altid at være A.

2 Malware

(a) Et malware detektor program fungere bl.a. ved at analysere programkode, netværks trafik og hvordan programmer opføre sig. Dette gøres ved

hjælp af matematiske og logiske analyser som kan udføres på en turingmaskine. Men da en turingmaskine kan løse alle tænkelige automatiske beregninger vil det også kunne lade sig gøre at finde en ny metode til at omgå de nyeste malware detektorer. Der findes altså ingen turingmaskine algoritme der kan detekte all malware [3]. Derfor at det ikke muligt at lave en perfekt malware detektor.

(b) Det er klart at virus producenter og antivirus producenter tjener penge på at der eksisterer malware. Virus producenter kan f.eks. samle store data mængder sammen som de kan sælge, eller snyde computere til at besøge bestemte sider med det formål at folk køber ting fra disse sider og Antivirus producenterne ville ikke kunne sælge deres software hvis ikke der fandtes malware. Men ser man bort fra den økonomiske del af problematikken, kan man spørge sig selv ikke man kan stoppe denne krig en gang for alle? Det kan man ikke da det som nævnt i opgaven oven for ikke er muligt at lave en malware detektor der detekter all malware. Og derfor vil kampen mellem antivirus producenter og dem der laver malware fortsætte.

3 Church-Turing Thesis

(a) Der findes mange forskellige programmeringssprog der tjener lige så mange forskelige formål. De kan opdeles i de turingkomplette[2] sprog og de specialiserede svage sprog som ikke er turingkomplette. Et specialiseret svagt sprog er f.eks. html der har til formål at præsentere indhold i en browser og sql der bliver brugt til at hente data fra databaser. Disse svage sprog er ikke Matematisk bedre end turingkomplette sprog. De imperative sprog som JAVA, C, ML, Python, mv. turingkomplette sprog og matematisk set er ingen af dem bedre end de andre da de alle kan løse alle tænkelige automatiske beregninger. Men der findes andre kriterier der gør at nogle sprog er at fortrække frem for et andet. Disse kriterier kan være hvor nemt, det er at udvikle i, hastigheden af programmer skrevet i sproget og størrelsen af programmet skrevet i sproget. F.eks. hvis man skal programmere en mikroprocessor med begrænset ram og hastighed, kan det være en fordel at skrive i sproget C da man her har bedre mulighed for at styre hukommelse allokation. Men hvis man skal lave et stort program hvor man er mange programmører der arbejder på samme

program kan et objekt orienteret sprog som JAVA være at fortrække, da man her nemt kan dele programmet op i forskelige klasser (del komponenter) hvilket gør et stort program mere overskueligt. Samtidig kan vær enkelt programmør arbejde sin del uden at kende til hvordan de andre programmører løser deres del. Python er kendt for at være et af de nemmeste sprog at lære at programmere i samt at programmer ofte kan skrives med færre linjer end f.eks. JAVA og C [4]. Filosofien bag sproget er at det skal være sjovt at programmere i, hvilket også navnet antyder. Derfor er det en fordel at have mange forskelige sprog at vælge imellem alt efter opgavens art og i hvilket miljø den skal løses.

(b) De specialiseret svage sprog kan ikke løse alle tænkelige automatiske beregninger, men der er andre fordele ved de svage sprog som retfærdigøre deres eksistens. De kan være utroligt effektive til bestemte ting. Man kan lave et svagt sprog hvor det er muligt automatisk at bevise korrektheden af et program, samt sætte grænser for tid og pladsforbrug, hvor dette med et turingkomplet sprog kan være umuligt. Et eksempel på et svagt sprog er googles eget udviklede sprog Sawzall som bruges til at lave statistik på store data mængder fordelt på mange maskiner[1]. Et program skrevet i Sawzall på ca. 5 linjer ville nok fylder langt over 200 linjer i JAVA. Et andet eksempel er SQL som bruges til at hente og ændre data i databaser. I SQL er det meget simpelt at specificere hvilken data man vil have til bage på en forespørgsel vha. konditional statements. Uanset om vi har at gøre med turingkomplette sprog eller svage sprog har de enkelte sprog forskellige fordele og ulemper og netop derfor er det ikke et men mange forskellige sprog.

4 referencer

1. . Torben Mogensen. Nogle programmeringssprog er mere lige end andre. 2010

2. . Wikipedia. Turing completeness. http://en.wikipedia.org/wiki/Turing_completeness

3. . Fred Cohen. Computer Viruses Theory and Experiments. Computers and Security, 6 (1) 22–35, Feb. 1987. http://all.net/books/Dissertation.pdf

4. . Wikipedia. Python programming language.

http://en.wikipedia.org/wiki/Python_(programming_language)