Homework Series 2

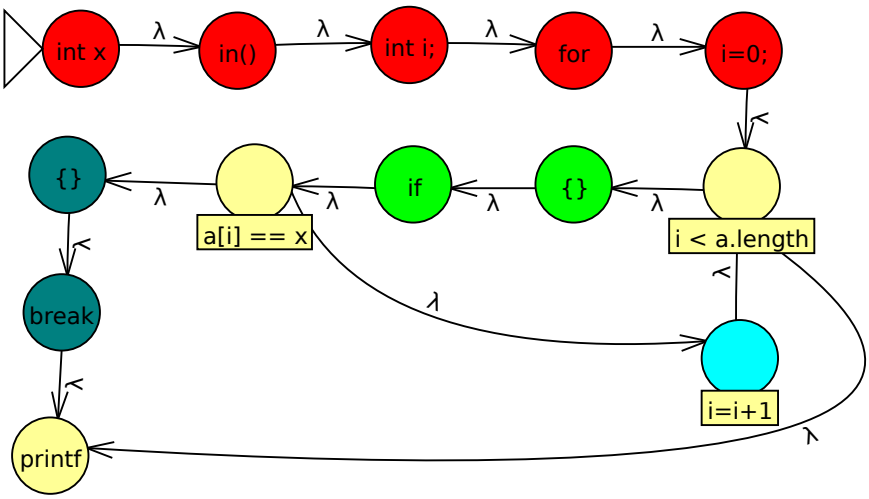
Martijn Verkleij (s1466895)

June 15, 2015

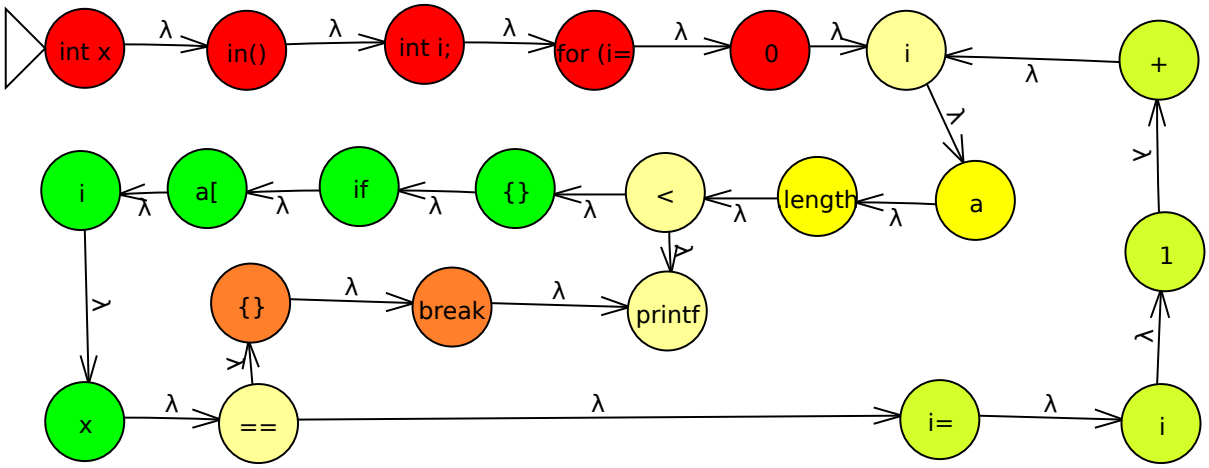
Chapter 1

Control Flow Graphs

1.1 Control Flow Graph



1.2 Control Flow Graph Uitgebreid



1.3 Vergelijking

De twee CFG's hebben op het aantal nodes na dezelfde structuur. het aantal basic blocks is bijvoorbeeld hetzelfde gebleven. Dit betekent dat de twee CFG's dezelfde gereduceerde CFG zullen opleveren. Omdat daarin alleen de basic blocks van een programma voorkomen, en deze grotere CFG's enkel uiteenzettingen van de basic blocks bevatten, zal dit ook gelden voor andere programma's.

Chapter 2

ILOC to CFG

Een incomplete implementatie kan gevonden worden in `s1466895/q2_2/Iloc2CFG.java`. Deze is niet in werkende staat.

Chapter 3

CalcCompiler

Doel van de opgave was het maken van een TreeVisitor die een gegeven expressie omzet naar een ILOC programma dat beschikt over twee registers en een stack. Deze opgave is meegeleverd in de broncode. De tests zijn als bijlage toegevoegd.

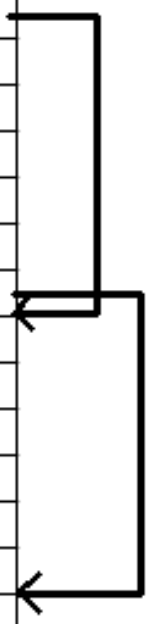
Chapter 4

Activation Records

De opgave vraagt om een representatie van de stack met de activation records. deze volgen hieronder, twee subopgaven per keer.

4.1 1+2

main	Local Data Area	A: [_,_]
	Caller's ARP	0
	return address	0
	return value	-
	RSA	-
	parameters	-
setx	Local Data Area	-
	Caller's ARP	-------->
	return address	29
	return value	-
	RSA	?
	parameters	false
perform	Local Data Area	x: _
	Caller's ARP	----->
	return address	37
	return value	-
	RSA	?
	parameters	A
main	Local Data Area	A: [3,10]
	Caller's ARP	0
	return address	0
	return value	-
	RSA	?
	parameters	-



4.2 3+4

max	Local Data Area	-
	Caller's ARP	--------->
	return address	25
	return value	-
	RSA	?
	parameters	a[1],a[2],x
setx	Local Data Area	-
	Caller's ARP	-------->
	return address	29
	return value	-
	RSA	?
	parameters	false
perform	Local Data Area	x: _
	Caller's ARP	----->
	return address	37
	return value	-
	RSA	?
	parameters	A
main	Local Data Area	A: [3,10]
	Caller's ARP	0
	return address	0
	return value	-
	RSA	?
	parameters	-
perform	Local Data Area	X: 3
	Caller's ARP	----->
	return address	37
	return value	-
	RSA	?
	parameters	A
main	Local Data Area	A: [3,10]
	Caller's ARP	0
	return address	0
	return value	-
	RSA	?
	parameters	-

Chapter 5

Convert

Het bestand `s1466895/q2_5/convert.iloc` bevat een ILOC-implementatie, die in uitvoer gelijk is aan het java-programma. Het maakt gebruik van de stack voor het recursief doorgeven van de variabelen, net als in opgave `6-CC.1`. De ILOC-code kan uitgevoerd worden met `s1466895/q2_5/ConvertTest.java`.

Appendices

Appendix A

Testresultaten opgave 3

A.1 $1 + -3 * 4$

$Processing 1 + -3 * 4$
$Outcome : -11$
$loadI 1 => r_1$
$pushr_1$
$loadI 3 => r_1$
$pushr_1$
$pop => r_1$
$rsubI r_1, 0 => r_2$
$pushr_2$
$loadI 4 => r_1$
$pushr_1$
$pop => r_1$
$pop => r_2$
$multr_1, r_2 => r_2$
$pushr_2$
$pop => r_1$
$pop => r_2$
$addr_1, r_2 => r_2$
$pushr_2$
$pop => r_1$
$out"Outcome : ", r_1$

A.2 $1 + -(3 * 4)$

$Processing 1 + -(3 * 4)$
$Outcome : -11$
$loadI 1 => r_1$
$pushr_1$
$loadI 3 => r_1$
$pushr_1$
$loadI 4 => r_1$
$pushr_1$
$pop => r_1$
$pop => r_2$
$multr_1, r_2 => r_2$
$pushr_2$
$pop => r_1$
$rsubI r_1, 0 => r_2$
$pushr_2$
$pop => r_1$
$pop => r_2$
$addr_1, r_2 => r_2$
$pushr_2$

$pop => r_1$
$out"Outcome:", r_1$

A.3 $(1 + -3) * 4)$

$Processing (1 + -3) * 4$
$Outcome: -8$
$loadI 1 => r_1$
$pushr_1$
$loadI 3 => r_1$
$pushr_1$
$pop => r_1$
$rsubIr_1, 0 => r_2$
$pushr_2$
$pop => r_1$
$pop => r_2$
$addr_1, r_2 => r_2$
$pushr_2$
$loadI 4 => r_1$
$pushr_1$
$pop => r_1$
$pop => r_2$
$multr_1, r_2 => r_2$
$pushr_2$
$pop => r_1$
$out"Outcome:", r_1$

A.4 $- - 56 + (1 + -3) * 4$

$Processing - -56 + (1 + -3) * 4$
$Outcome: 48$
$loadI 56 => r_1$
$pushr_1$
$pop => r_1$
$rsubIr_1, 0 => r_2$
$pushr_2$
$pop => r_1$
$rsubIr_1, 0 => r_2$
$pushr_2$
$loadI 1 => r_1$
$pushr_1$
$loadI 3 => r_1$
$pushr_1$
$pop => r_1$
$rsubIr_1, 0 => r_2$
$pushr_2$
$pop => r_1$
$pop => r_2$
$addr_1, r_2 => r_2$
$pushr_2$
$loadI 4 => r_1$
$pushr_1$
$pop => r_1$
$pop => r_2$
$multr_1, r_2 => r_2$
$pushr_2$
$pop => r_1$
$pop => r_2$
$addr_1, r_2 => r_2$
$pushr_2$

$pop => r_1$
$out"Outcome:", r_1$