OBLIG 1 INF 5110

Andreas Heim (andrheim)

Gjennomføring

Semantisk analyse

Nodene har selv ansvar for å sjekke semantikken og legge til seg selv i symboltabellen. For å håndtere konvertering fra int til float introduserte jeg en funksjon i Type-klassene, Type.canBeCastTo(otherType). Selv om denne i praksis bare gjelder mellom int og float, så føler jeg at koden blir litt ryddigere for hver “instanceof”-expression som jeg slipper å skrive. I alle henseender innenfor kompilatoren vil en int være likeverdig en float, så jeg kunne sannsynligvis ha gjort slik at Int arvet float.

Symboltabell

Symboltabellen er stackbasert og legger en ny HashMap<Name, Type> på stacken hver gang man går inn i et nytt skop. Tabellen tar ikke vare på “gamle” skop.

Siden grunnstrukturen i symboltabellen er en hashmap, så viste det seg problematisk å håndtere metodesignatur. Jeg følte ikke at metodesignatur og returtype hang sammen, så jeg introduserte en ny klasse i syntakstreet, FunctionName. Denne klassen tar imot formelle parametre i tillegg til funksjonsnavnet.

I CallStatement blir det laget en instans av FunctionName på bakgrunn av aktuelle parametere. I FunctionName-klassen har jeg overskrevet equals-metoden til å sammenlikne, navn, parameter-rekkefølge/type og om parameteren er pass-by-value/reference, slik at lookup-funksjonen i HashMap fungerer som ønsket.

Bytekode

Alle standardbibliotek blir lagt til symboltabellen og kodegenerert i Program-klassen (øverste nivå).

Alle Type-klassene har en metode “getByteCodeType()” som returnerer korresponderende type for bytecode-biblioteket. Unntaket er ClassType som trenger oppslag i kodefilen for å finne strukt-id. Dette blir sjekket der hvor variable blir deklarert i syntakstreet, men denne logikken kunne også med fordel vært flyttet inn i ClassType eller en Util-klasse.

I de aritmetiske og logiske uttrykksklassene genererer selve operanden sin egen kode, dette gjør kodegenereringsmetodene små og lettleste.

I AssignStatement-klassen er det lagt inn sjekk for om variabelen er lokal eller global. Dvs. om man finner en lokal variabel brukes denne, hvis ikke leter man etter (og bruker) en global. Denne sjekken kunne blitt enklere hvis symboltabellen hadde blitt tatt vare på etter semantisk analyse.

Kjente feil & Mangler

I AndOp og OrOpExpression er det ikke lagt inn short-circuit evaluering slik som beskrevet i språknotatet.

Mangelfull output fra parser/semantisk analyse om hvor feilen befinner seg.