UNIVERSITY OF COPENHAGEN
FACULTY OF SCIENCE

Project outside course scope

Jacob Herbst(mwr148), Matilde Broløs (jtw868)

IFC

An imperative language with static verification

Advisor: Ken Friis Larsen

1. April 2022

Abstract

1 Introduction

2 Background

2.1 Language

IFC is a small imperative programming language with a build in assertion language enabeling the possibility for statically verifying programs by the use of external provers. In this section we will present the syntax and semantics of the imperative language and the assertion language.

⟨*statement*⟩ ::= ⟨*statement*⟩ ';' ⟨*statement*⟩
| ⟨*ghostid*⟩ ':=' ⟨*aexpr*⟩
| ⟨*id*⟩ ':=' ⟨*aexpr*⟩
| 'if' ⟨*bexpr*⟩ '{' ⟨*statement*⟩ '}'
| 'if' ⟨*bexpr*⟩ '{' ⟨*statement*⟩ '}' 'else' '{' ⟨*statement*⟩ '}'
| 'while' ⟨*bexpr*⟩ ⟨*invariant*⟩ ⟨*variant*⟩ '{' ⟨*statement*⟩ '}'
| '{' ⟨*assertion*⟩ '}'
| 'skip'
| 'violate'

⟨*invariant*⟩ ::= '?{' ⟨*assertion*⟩ '}'
| ⟨*invariant*⟩ ';' ⟨*invariant*⟩

⟨*variant*⟩ ::= '!{' ⟨*aexpr*⟩ '}'

⟨*assertion*⟩ ::= 'forall' ⟨*id*⟩ ⟨*assertion*⟩
| 'exists' ⟨*id*⟩ ⟨*assertion*⟩
| '∼' ⟨*assertion*⟩
| ⟨*assertion*⟩ ⟨*assertionop*⟩ ⟨*assertion*⟩
| ⟨*bexpr*⟩

⟨*assertionop*⟩ ::= '/\'
| '\/'
| '=>'

⟨*bexpr*⟩ ::= true
| false
| '!' ⟨*bexpr*⟩
| ⟨*bexpr*⟩ ⟨*bop*⟩ ⟨*bexpr*⟩
| ⟨*bexpr*⟩ ⟨*rop*⟩ ⟨*bexpr*⟩

⟨*bop*⟩ ::= '&&'
| '||'

⟨*rop*⟩ ::= '<'
| '<='
| '='
| '/='
| '>'
| '>='

⟨*aexpr*⟩ ::= ⟨*id*⟩
| ⟨*ghostid*⟩
| ⟨*integer*⟩
| '-'⟨*axpr*⟩
| ⟨*aexpr*⟩ ⟨*aop*⟩ ⟨*aexpr*⟩

⟨*aop*⟩ ::= '+'
| '−'
| '∗'
| '/'
| '%'

⟨*ghostid*⟩ ::= ghost⟨*string*⟩

⟨*id*⟩ ::= ⟨*string*⟩

Vi vil gerne give en kort præsentation. og hvorfor det er interessant. eksempel? mult? skriv noget om det.

2.2 Hoare Logic

- Beskriv hvad hoare logik er, og hvorfor det er brugbart?

2.3 Verification Condition Generation

- snak om Weakest precondition, i en figur? - Beskrive de vigtige af dem? - gennemgå det i eksemplet - Sound og Complete?

2.4 SAT-solvers

- IDK noget klogt.

3 Implementation

- Why have we taken the approach we have. Multistage generation? - Following is interesting: - quantifers - ghosts - while (invarianter og varianten) - Ideally static and dynamic execution should give equivalent results. - Testing