EIDI 2 Cheatsheet

github contributors: https://github.com/zepatrik/eidi2_cheatsheet

20. Februar 2018

1 Logik

$$\neg(A \vee B) \equiv \neg A \wedge B$$
$$A \vee (B \wedge A) \equiv A \wedge (B \vee A) \equiv A$$
$$A \implies B \equiv \neg A \vee B$$

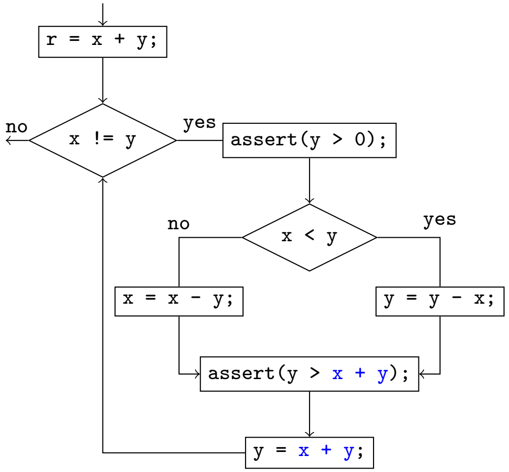
2 Verifikation

2.1 WP

$$\mathsf{WP}[\![\texttt{x = read();}]\!](B) \equiv \forall x.B$$
$$I \Rightarrow \mathsf{WP}[\![\texttt{b}]\!](B_0, B_1) \equiv I \Rightarrow (((\neg b) \Rightarrow B_0) \wedge (b \Rightarrow B_1))$$
$$\begin{aligned}\mathsf{WP}[\![\texttt{b}]\!](B_0, B_1) &\equiv (b \vee B_0) \wedge (\neg b \vee B_1) \\ &\equiv (\neg b \wedge B_0) \vee (b \wedge B_1) \vee (B_0 \wedge B_1) \\ &\equiv (\neg b \wedge B_0) \vee (b \wedge B_1)\end{aligned}$$

2.2 Terminierung

1. vor jedem Schleifendurchlauf `r > 0`
2. `r` wird bei jedem Durchlauf kleiner



3 ocaml

3.1 Funktoren

```
module type A = sig
  type t
  val f : t -> t
end

module B: A = struct
  type t = int
  let f x = x + 1
end

module Ext(X: A) = struct
  include X
  let g x = f (f x)
end

module C = Ext (B)
```

3.2 Threaded tree Beispiel

```
open Thread open Event

let rec min = function
  | Leaf a -> a
  | Node (a,b) ->
    let c = new_channel () in
    let f t = sync (send c (min t)) in
    let _ = create f a in
    let _ = create f b in
    let x = sync (receive c) in
    let y = sync (receive c) in
    if x < y then x else y
```

3.3 Exceptions

```
type t = exn

try (
  raise Failure "this should fail"
) with Failure s -> print_string s

exeption Custom of int
try (
  raise Custom 0
) with _ -> ()
```

3.4 Invarianten Tipps

1. Wir benotigen eine Aussage uber den Wert der Variablen, uber die wir etwas beweisen wollen (x) in der Schleifeninvariante. Die Aussage muss dabei mindestens so prazise ($\neq, \geq, \leq, =$) sein, wie die Aussage, die wir beweisen wollen.
2. Variablen, die an der Berechnung von x beteiligt sind **und** Werte von einer Schleifeniteration in die nachste transportieren ("loop-carried"), mussen in die Schleifeninvariante aufgenommen werden.
3. Die Schleife zu verstehen ist unerlasslich. Eine Tabelle fur einige Schleifendurchlaufe kann helfen die Zusammenhange der Variablen (insbesondere mit dem Schleifenzahler i) aufzudecken. Oft lassen sich mit einer Tabelle, in der man die einzelnen Berechnungsschritte notiert, diese Zusammenhange deutlich leichter erkennen, als mit einer Tabelle, die nur konkrete Werte enthalt.
4. Die Variablen, die zur Berechnung von x im Beweisziel verwendet werden und diejenigen, die dazu in der Invariante verwendet werden, mussen in Beziehung stehen. Wenn diese Beziehung nicht aus der Verzweigungsbedingung folgt, mussen weitere Aussagen zur Invariante hinzugefugt werden.
5. Bei einer Bedingung mit Ungleichung ($<, \leq, >, \geq$) kann der Schleifenzahler (i) von der anderen Seite ($\geq, \leq$) begrenzt werden, sodass am Ende der Schleife Gleichheit folgt. Naturlich darf das nur dann geschehen, wenn diese Begrenzung im Programm auch tatsachlich gilt.
6. Werden Programmeingaben begrenzt, z.B. durch Ziehen von Betragen, vorzeitigen Programmabbruch bei unerwunschten Eingaben oder durch gegebene Zusicherungen, so kann die Information uber die Werte der Eingabevariablen ($n, m, ...$), die innerhalb der Schleife zulassig sind, in der Invariante sinnvoll sein.
7. Existiert kein klassischer Schleifenzahler, kann ein gedanklicher Zahler verwendet werden um den Zusammenhang zwischen den Variablen herzustellen.
8. Sind in der Schleife Verzweigungen enthalten, so hangt x, neben dem Schleifenzahler, meist auch von einer Variablen ab, die in der Bedingung der Verzweigung vorkommt. In der Invariante muss diese Abhangigkeit dann aufgenommen werden. Lasst sich keine "einfache" Beziehung finden, so kann in der Invariante eine Fallunterscheidung verwendet werden.
9. In einem Terminierungsbeweis wird eine Aussage uber r in der Invariante benotigt. Es gilt dabei immer die fur r ermittelte Berechnungsvorschrift.
10. Fur die Terminierung benotigen wir Aussagen uber alle Variablen in der Invarianten, die fur die Berechnung von r benotigt werden und uber die sich keine starken Beziehungen aus der Schleifenbedingung folgern lassen (wobei "stark" hier mindestens $\leq, \geq$ meint).