Semantyka i Weryfikacja programów

Praca domowa 3.

Hubert Michalski hm438596

6 lutego 2024

1 Zadanie

Na kolejnej stronie podany jest program zapisany w języku $TINY$ rozszerzonym o operację **div2** dzielenia całkowitego przez 2 oraz, na potrzeby formułowania asercji, o operację podnoszenia liczb całkowitych do całkowitej nieujemnej potęgi. Jak widać z podanej specyfikacji, jest to kolejna wersja liczenia pierwiastka całkowitego liczby całkowitej dodatniej. Udowodnij częściową poprawność tego programu względem podanej specyfikacji, podając niezmienniki obu pętli oraz wstawiając odpowiednie formuły w nawiasy $\{\ldots\}$ tak, aby podane niezmienniki i asercje zapisały przeprowadzony dowód częściowej poprawności programu w logice Hoare'a. Jeśli w dwóch sąsiednich wierszach występują nawiasy $\{\ldots\}$ to pomiędzy wstawionymi tam asercjami powinna zachodzić implikacja. Można też dodać dodatkowe nawiasy i wpisać w nie odpowiednie asercje. Poza niezmiennikami γ_1 i γ_2 , wymagane jest przynajmniej podanie formuł $\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5$ (ale ewentualne błędy w innych formułach też będą wpływały na ostateczną ocenę rozwiązania).

2 Rozwiązanie

```
{ n > 0 }
i := 1;
{ n > 0 ∧ i = 1 }
kw := 4;
{ n > 0 ∧ i = 1 ∧ kw = 4 }
while {γ_1: kw = 4i^2 ∧ i^2 ≤ n ∧ ∃_{k∈ℕ}.i = 2^k }
    kw ≤ n
  do
  ({ kw = 4i^2 ∧ 4i^2 ≤ n ∧ ∃_{k∈ℕ}.2i = 2^k }
   i := 2*i;
   {α_1: kw = i^2 ∧ i^2 ≤ n ∧ ∃_{k∈ℕ}.i = 2^k }
   kw := 4*kw
   { kw = 4i^2 ∧ i^2 ≤ n ∧ ∃_{k∈ℕ}.i = 2^k }
  )
{α_2: kw = (2i)^2 ∧ i^2 ≤ n < kw ∧ ∃_{k∈ℕ}.i = 2^k }
r := i;
{ kw = (r + i)^2 ∧ r = i ∧ r^2 ≤ n < kw ∧ ∃_{k∈ℕ}.i = 2^k }
dri := kw div2;
{ kw = (r + i)^2 ∧ r = i ∧ dri = 2ri ∧ r^2 ≤ n < kw ∧ ∃_{k∈ℕ}.i = 2^k }
ik := dri div2;
{ kw = (r + i)^2 ∧ r = i ∧ dri = 2ri ∧ ik = i^2 ∧ r^2 ≤ n < kw ∧ ∃_{k∈ℕ}.i = 2^k }
while {γ_2: kw = (r + i)^2 ∧ dri = 2ri ∧ ik = i^2 ∧ r^2 ≤ n < kw ∧ ∃_{k∈ℕ}.i = 2^k }
     i > 1
  do
  ({ ik/4 = i^2/4 ∧ dri/2 = 2ri/2 ∧ kw = (r + i)^2 ∧ r^2 ≤ n < kw ∧ ∃_{k∈ℕ}.i = 2^k ∧ i > 1 }
   i := i div2;
   {α_3: ik/4 = i^2 ∧ dri/2 = 2ri ∧ kw = (r + 2i)^2 ∧ r^2 ≤ n < kw ∧ ∃_{k∈ℕ}.i = 2^k }
   ik := (ik div2) div2;
   { ik = i^2 ∧ dri/2 = 2ri ∧ kw = (r + 2i)^2 ∧ r^2 ≤ n < kw ∧ ∃_{k∈ℕ}.i = 2^k }
   dri := dri div2;
   {α_4: ik = i^2 ∧ dri = 2ri ∧ kw = (r + 2i)^2 ∧ r^2 ≤ n < kw ∧ ∃_{k∈ℕ}.i = 2^k }
   if (kw - dri - 3*ik) ≤ n
   then
      { ik = i^2 ∧ dri + 2ik = 2(r+i)i ∧ kw = (r + 2i)^2 ∧ (r+i)^2 ≤ n < kw ∧ ∃_{k∈ℕ}.i = 2^k }
      r := r + i;
      {α_5: ik = i^2 ∧ dri + 2ik = 2ri ∧ kw = (r + i)^2 ∧ r^2 ≤ n < kw ∧ ∃_{k∈ℕ}.i = 2^k }
      dri := dri + 2*ik
      { ik = i^2 ∧ dri = 2ri ∧ kw = (r + i)^2 ∧ r^2 ≤ n < kw ∧ ∃_{k∈ℕ}.i = 2^k }
   else
      { ik = i^2 ∧ dri = 2ri ∧ kw = (r + i)^2 + dri + 3ik ∧ r^2 ≤ n < (r + i)^2 ∧ ∃_{k∈ℕ}.i = 2^k }
      kw := kw - dri - 3*ik;
      { ik = i^2 ∧ dri = 2ri ∧ kw = (r + i)^2 ∧ r^2 ≤ n < kw ∧ ∃_{k∈ℕ}.i = 2^k }
  )
{ r^2 ≤ n < (r + i)^2 ∧ ∃_{k∈ℕ}.i = 2^k ∧ i ≤ 1 }
{ r^2 ≤ n < (r + 1)^2 }
```