

Sviluppo di Software Sicuro - S³ Condizioni di verifica in pratica

Corso di Laurea Magistrale in
Sicurezza Informatica: Infrastrutture e Applicazioni
Università di Pisa – Polo di La Spezia
C. Montangero
Anno accademico 2009/10

Sommario

- Il nucleo del linguaggio delle asserzioni SPARK
- Examiner per la verifica
 - Generazione delle vc
 - Semplificazione delle vc: Simplifier
- Esempi semplici
- Un esempio dal vivo:
 - prodotto per somme successive
- Esercizio: divisione

Il linguaggio delle annotazioni

- Per il contesto di prova dei vincoli ai sottoprogrammi (pre, post e assert):
 - il linguaggio delle espressioni SPARK, più
 - la decorazione (tilde) per denotare i valori iniziali delle variabili in out
 - l'implicazione logica, denotata da " \rightarrow "
 - l'equivalenza logica, denotata da " \leftrightarrow "
 - le espressioni quantificate (tipizzate)
 - universale, esistenziale su intervalli
 - ...

S3: VC - C.Montangero - Copyright 2010

3

Integer increment

```

1 package P
2 is
3 end P;
4
5 package body P
6 is
7
8  procedure Inc(X : in out Integer)
9    --# derives X from X;
10   --# post X = X~ + 1;
11  is
12  begin
13    X := X + 1;
14  end Inc;
15
16 end P;
```

Condizioni di verifica:

For path(s) from start to finish:

procedure_inc_1.

H1: true .

H2: $x \geq \text{integer_first}$.H3: $x \leq \text{integer_last}$. \rightarrow C1: $x + 1 = x + 1$.

```

+++ Flow analysis of subprogram Inc
performed: no errors found.
```

15

16 end P;

S3: SPARK - C.Montangero - Copyright 2010

4

Swap routine

<pre> 1 package P is end P; ... 5 package body P 6 is 7 procedure Swap(X, Y : in out Integer) 8 --# derives X from Y & Y from X; 10 --# post X = Y~ and Y = X~; 11 is 12 T : Integer; 13 begin 14 T := X; 15 X := Y; 16 Y := T; 17 end Swap; +++ Flow analysis of subprogram Swap performed: no errors found. 18 19 end P;</pre>	<p>Condizioni di verifica:</p> <pre> procedure_swap_1. H1: true . H2: x >= integer__first . H3: x <= integer__last . H4: y >= integer__first . H5: y <= integer__last . -> C1: y = y . C2: x = x .</pre>
--	---

S3: SPARK - C.Montangero - Copyright 2010

5

Swap routine difettosa

<pre> 1 package P is end P; ... 5 package body P 6 is 7 procedure Swap(X, Y : in out Integer) 8 --# derives X from Y & Y from X; 10 --# post X = Y~ and Y = X~; 11 is 12 T : Integer; 13 begin 14 T := X; 15 X := Y; 16 Y := T + 1; -- error, but information flow 17 -- unaffected 18 end Swap; +++ Flow analysis of subprogram Swap performed: no errors found. 18 19 end P;</pre>	<p>Condizioni di verifica:</p> <pre> procedure_swap_1. H1: true . H2: x >= integer__first . H3: x <= integer__last . H4: y >= integer__first . H5: y <= integer__last . -> C1: y = y . C2: x + 1 = x . !!!!!!!</pre> <p>Usando Simplifier:</p> <pre> ... -> C1: false .</pre>
---	---

S3: SPARK - C.Montangero - Copyright 2010

6

Examiner e Simplifier

- L'esempio sopra mostra che
 - Simplifier può identificare un difetto
 - Examiner può dare informazioni per individuarlo
- Il prossimo mostra che
 - Examiner può identificare un difetto
 - Simplifier può dare informazioni aggiuntive sulla sua natura

S3: VC- C.Montangero - Copyright 2010

7

Swap routine con difetto di flusso

```

1 package P
2 is
3 end P;
4
5 package body P
6 is
7 procedure Swap(X, Y : in out Integer)
8 --# derives X from Y &
9 --# Y from X;
10 --# post X = Y~ and Y = X~;
11 is
12 T : Integer;
13 begin
14 T := X;
15   ^1
16 X := Y;
17 Y := X;
18 end Swap;

```

Condizioni di verifica:

For path(s) from start to finish:
procedure_swap_1.

H1: true .
H2: x >= integer__first .
H3: x <= integer__last .
H4: y >= integer__first .
H5: y <= integer__last .

->
C1: y = y .
C2: y = x .

Usando Simplifier:

...
->
C1: y = x

S3: SPARK - C.Montangero - Copyright 2010

8

Swap routine con difetto di flusso

```

1 package P
2 is
3 end P;
4
5 package body P
6 is
7 procedure Swap(X, Y : in out Integer)
8 --# derives X from Y &
9 --# Y from X;
10 --# post X = Y~ and Y = X~;
11 is
12 T : Integer;
13 begin
14 T := X;
15   ^1
16   !!! ( 1) Flow Error : Ineffective statement.
17 X := Y;
18 Y := X;
19 end Swap;

```

Condizioni di verifica:

For path(s) from start to finish:
procedure_swap_1.

H1: true .
H2: x >= integer__first .
H3: x <= integer__last .
H4: y >= integer__first .
H5: y <= integer__last .

->
C1: y = y .
C2: y = x .

Usando Simplifier:

...
->
C1: y = x

S3: SPARK - C.Montangero - Copyright 2010

9

Cicli: radice quadrata

```

7 procedure IntRoot(N : in Natural; Root : out Natural)
9 --# derives Root from N;
10 --# pre N >= 0;
11 --# post (Root * Root <= N) and (N < (Root + 1) * (Root + 1));
13 is
14 R : Natural := 0;
15 S, T : Natural := 1;
16 begin
17 loop
18 --# assert (T = 2 * R + 1) and (S = (R + 1) * (R + 1)) and (R * R <= N);
21 exit when S > N;
22 R := R + 1;
24 T := T + 2;
25 S := S + T; -- S := (R+1)^2 + 2R + 1 = R^2 + 4R + 2 = (R+2)^2 !!!!
26 end loop;
27 Root := R;
28 end IntRoot;

```

S3: SPARK - C.Montangero - Copyright 2010

10

VC: inizializzazione

For path(s) from start to assertion of line 18:

procedure_introot_1.

H1: $n \geq 0$.

H2: $n \geq \text{natural_first}$.

H3: $n \leq \text{natural_last}$.

->

C1: $1 = 2 * 0 + 1$.

$(T = 2 * R + 1)$

C2: $1 = (0 + 1) * (0 + 1)$

$(S = (R + 1) * (R + 1))$

C3: $0 * 0 \leq n$.

$(R * R \leq N);$

S3: VC - C.Montangero - Copyright 2010

11

VC: corpo del ciclo

For path(s) from assertion of line 18 to assertion of line 18:

procedure_introot_2.

H1: $t = 2 * r + 1$.

H2: $s = (r + 1) * (r + 1)$.

H3: $r * r \leq n$.

H4: $\text{not } (s > n)$.

guardia

->

C1: $t + 2 = 2 * (r + 1) + 1$.

$H1[t/t+2, r/r+1]$

C2: $s + (t + 2) = (r + 1 + 1) * (r + 1 + 1)$.

$H2[s/s+t, t/t+2, r/r+1]$

C3: $(r + 1) * (r + 1) \leq n$.

$H3[r/r+1]$

S3: VC - C.Montangero - Copyright 2010

12

VC: finalizzazione

For path(s) from assertion of line 18 to finish:

procedure_introot_3.

H1: $t = 2 * r + 1$.

H2: $s = (r + 1) * (r + 1)$.

H3: $r * r \leq n$.

H4: $s > n$.

guardia

->

C1: $r * r \leq n$.

post[Root/R]

C2: $n < (r + 1) * (r + 1)$.

- Simplifier le verifica tutte 😊

S3: VC - C.Montangero - Copyright 2010

13

Pragmatica

- I file generati da Examiner: .vcg
- Per semplificare:
 - tasto destro sul vcg: SPARK-> Simplify
 - I file generati da Simplifier: .siv
- Tutti in una cartella con il nome del file Ada
 - con gli switch di timet

S3: VC - C.Montangero - Copyright 2010

14

Esercizio

- Divisione per sottrazioni successive:
 - $Ddo = Dre * Q + R$ **and** $R < Dre$
- Condizione di uscita: $R < Dre$