Verifica del codice con Interpretazione Astratta

Daniele Grasso
grasso@dsi.unifi.it
grasso.dan@gmail.com

Università di Firenze, D.S.I., Firenze, Italy

December 15, 2009

Sommario

Background

Safety Critical System

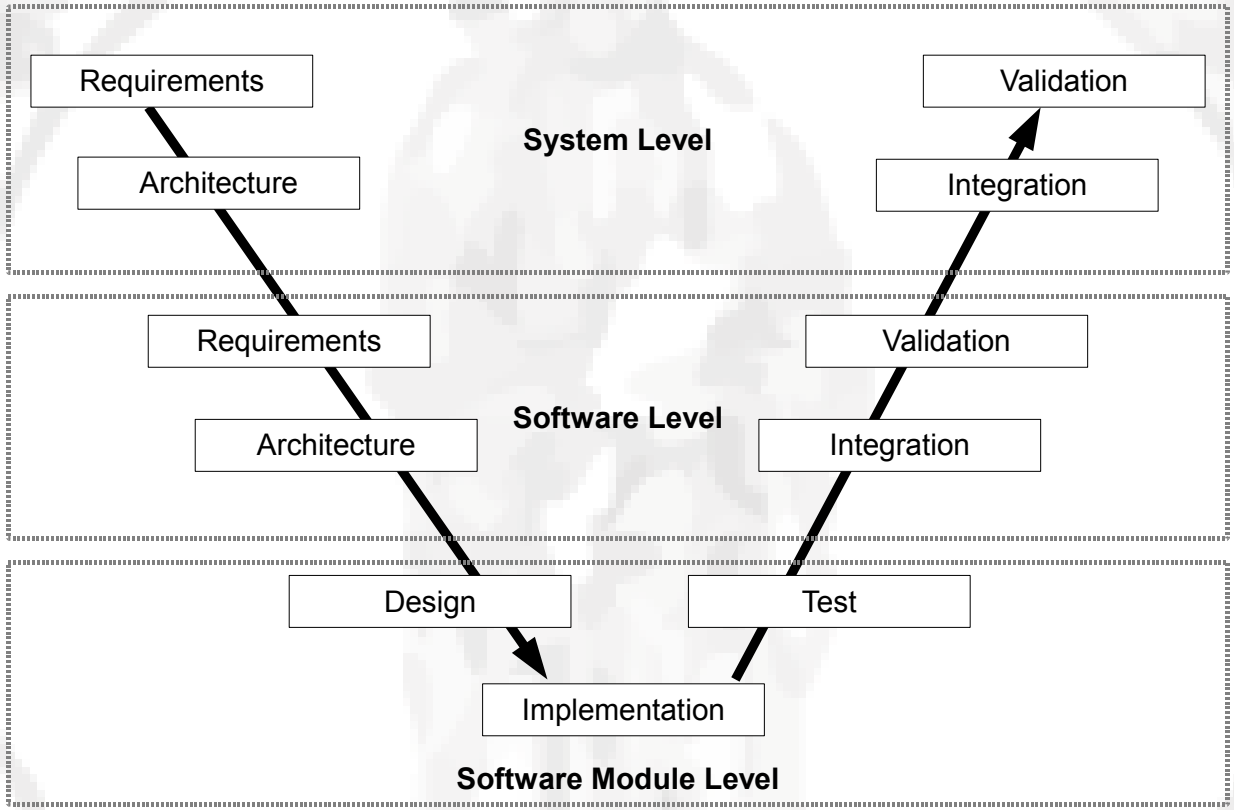
Un sistema è detto safety critical quando un suo errato funzionamento può provocare danni più o meno gravi a cose e persone che ne fanno parte o che lo circondano.

Sviluppo guidato da standard internazionali:

- Functional Safety of electrical/electronic/programmable electronic safety related system (IEC 61508)
- Railway applications - Communication, signalling and processing system - Software for railway control and protection systems (EN 50128)
- Software Considerations in Airborne Systems and Equipment Certification (DO-178)
- ...

EN 50128

Standard per applicazioni ferroviarie.
Modello di sviluppo del software



EN 50128

Tecniche consigliate per la verifica del software

Table A.5 – Verification and Testing (clause 11)

TECHNIQUE/MEASURE		Ref	SWS IL0	SWS IL1	SWS IL2	SWS IL3	SWS IL4
1.	Formal Proof	B.31	-	R	R	HR	HR
2.	Probabilistic Testing	B.47	-	R	R	HR	HR
3.	Static Analysis	D.8	-	HR	HR	HR	HR
4.	Dynamic Analysis and Testing	D.2	-	HR	HR	HR	HR
5.	Metrics	B.42	-	R	R	R	R
6.	Traceability Matrix	B.69	-	R	R	HR	HR
7.	Software Error Effect Analysis	B26	-	R	R	HR	HR

HR Higly Recommended

R Recommended

EN 50128

Tecniche consigliate per la verifica del software

Table A.5 – Verification and Testing (clause 11)

TECHNIQUE/MEASURE	Ref	SWS IL0	SWS IL1	SWS IL2	SWS IL3	SWS IL4
1. **FORMAL PROOF**	B.31	-	R	R	HR	HR
2. Probabilistic Testing	B.47	-	R	R	HR	HR
3. Static Analysis	D.8	-	HR	HR	HR	HR
4. Dynamic Analysis and Testing	D.2	-	HR	HR	HR	HR
5. Metrics	B.42	-	R	R	R	R
6. Traceability Matrix	B.69	-	R	R	HR	HR
7. Software Error Effect Analysis	B26	-	R	R	HR	HR

HR Higly Recommended

R Recommended

EN 50128

Tecniche consigliate per la verifica del software

Table A.5 – Verification and Testing (clause 11)

TECHNIQUE/MEASURE		Ref	SWS IL0	SWS IL1	SWS IL2	SWS IL3	SWS IL4
1.	Formal Proof	B.31	-	R	R	HR	HR
2.	Probabilistic Testing	B.47	-	R	R	HR	HR
3.	**STATIC ANALYSIS**	D.8	-	HR	HR	HR	HR
4.	Dynamic Analysis and Testing	D.2	-	HR	HR	HR	HR
5.	Metrics	B.42	-	R	R	R	R
6.	Traceability Matrix	B.69	-	R	R	HR	HR
7.	Software Error Effect Analysis	B26	-	R	R	HR	HR

HR Higly Recommended

R Recommended

EN 50128

Tecniche consigliate per la verifica del software

Table A.5 – Verification and Testing (clause 11)

TECHNIQUE/MEASURE	Ref	SWS IL0	SWS IL1	SWS IL2	SWS IL3	SWS IL4
1. Formal Proof	B.31	-	R	R	HR	HR
2. Probabilistic Testing	B.47	-	R	R	HR	HR
3. Static Analysis	D.8	-	HR	HR	HR	HR
4. **DYNAMIC ANALYSIS AND TESTING**	D.2	-	HR	HR	HR	HR
5. Metrics	B.42	-	R	R	R	R
6. Traceability Matrix	B.69	-	R	R	HR	HR
7. Software Error Effect Analysis	B26	-	R	R	HR	HR

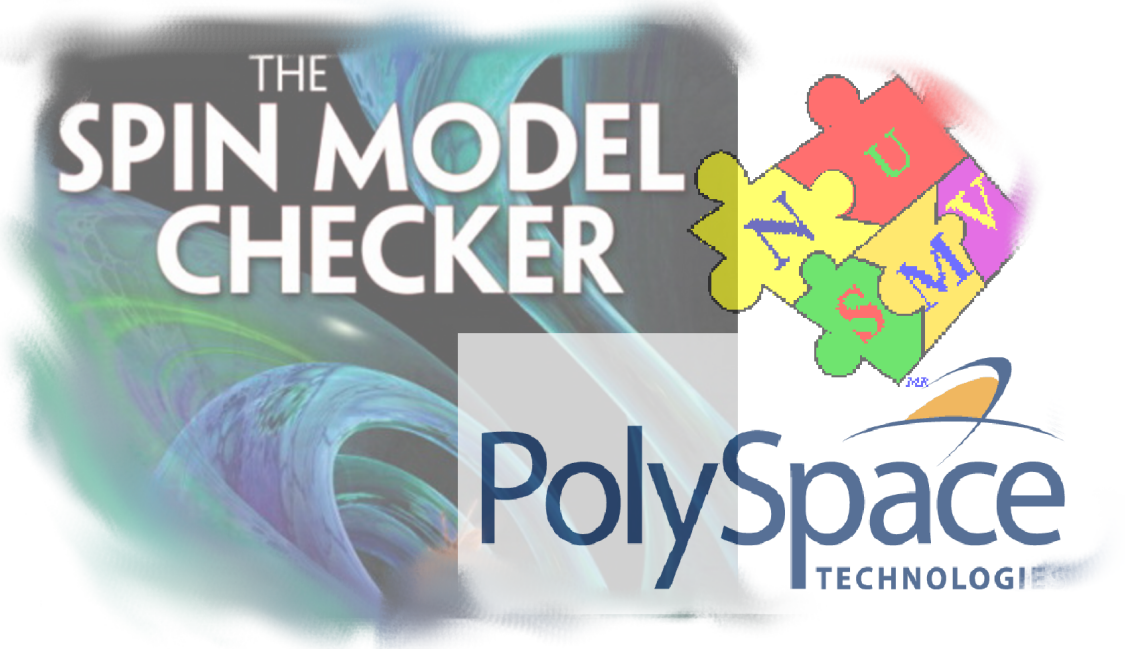
HR Higly Recommended

R Recommended

Metodi Formali

I metodi formali sono tecniche rigorose basate su formalismi matematici per la specifica e la verifica del software. Il model checking è una tecnica di verifica formale.

- Fase di design (NuSMV, Spin,...)
- Fase di implementazione (CBMC, PolySpace)



Model Checking

Osservazione

Se il modello descrive tutti i possibili comportamenti del sistema, non è sufficiente la verifica della sua correttezza?

Analisi Statica

Tecniche di analisi statica segnalate nella EN 50128

TECHNIQUE/MEASURE		Ref	SWS IL0	SWS IL1	SWS IL2	SWS IL3	SWS IL4
1.	Boundary Value Analysis	B.4	-	R	R	HR	HR
2.	Checklists	B.8	-	R	R	R	R
3.	Control Flow Analysis	B.9	-	HR	HR	HR	HR
4.	Data Flow Analysis	B.11	-	HR	HR	HR	HR
5.	Error Guessing	B.21	-	R	R	R	R
6.	Fagan Inspections	B.24	-	R	R	HR	HR
7.	Sneak Circuit Analysis	B.55	-	-	-	R	R
8.	Symbolic Execution	B.63	-	R	R	HR	HR
9.	Walkthroughs/Design Reviews	B.66	HR	HR	HR	HR	HR

Analisi Statica

Tecniche di analisi statica segnalate nella EN 50128

TECHNIQUE/MEASURE	Ref	SWS IL0	SWS IL1	SWS IL2	SWS IL3	SWS IL4
1. Boundary Value Analysis	B.4	-	R	R	HR	HR
2. Checklists	B.8	-	R	R	R	R
3. **CONTROL FLOW ANALYSIS**	B.9	-	HR	HR	HR	HR
4. Data Flow Analysis	B.11	-	HR	HR	HR	HR
5. Error Guessing	B.21	-	R	R	R	R
6. Fagan Inspections	B.24	-	R	R	HR	HR
7. Sneak Circuit Analysis	B.55	-	-	-	R	R
8. Symbolic Execution	B.63	-	R	R	HR	HR
9. Walkthroughs/Design Reviews	B.66	HR	HR	HR	HR	HR

Analisi Statica

Tecniche di analisi statica segnalate nella EN 50128

TECHNIQUE/MEASURE	Ref	SWS ILO	SWS IL1	SWS IL2	SWS IL3	SWS IL4
1. Boundary Value Analysis	B.4	-	R	R	HR	HR
2. Checklists	B.8	-	R	R	R	R
3. Control Flow Analysis	B.9	-	HR	HR	HR	HR
4. **DATA FLOW ANALYSIS**	B.11	-	HR	HR	HR	HR
5. Error Guessing	B.21	-	R	R	R	R
6. Fagan Inspections	B.24	-	R	R	HR	HR
7. Sneak Circuit Analysis	B.55	-	-	-	R	R
8. Symbolic Execution	B.63	-	R	R	HR	HR
9. Walkthroughs/Design Reviews	B.66	HR	HR	HR	HR	HR

Analisi Dinamica

Tecniche di analisi dinamica segnalate nella EN 50128

TECHNIQUE/MEASURE	Ref	SWS IL0	SWS IL1	SWS IL2	SWS IL3	SWS IL4
1. Test Case Execution from Boundary Value Analysis	B.4	-	HR	HR	HR	HR
2. Test Case Execution from Error Guessing	B.21	R	R	R	R	R
3. Test Case Execution from Error Seeding	B.22	-	R	R	R	R
4. Performance Modelling	B.45	-	R	R	HR	HR
5. Equivalence Classes and Input Partition Testing	B.19	-	R	R	HR	HR
6. Structure-Based Testing	B.58	-	R	R	HR	HR

Analisi Dinamica

Tecniche di analisi dinamica segnalate nella EN 50128

TECHNIQUE/MEASURE	Ref	SWS IL0	SWS IL1	SWS IL2	SWS IL3	SWS IL4
TEST CASE EXECUTION FROM BOUNDARY VALUE ANALYSIS	B.4	-	HR	HR	HR	HR
2. Test Case Execution from Error Guessing	B.21	R	R	R	R	R
3. Test Case Execution from Error Seeding	B.22	-	R	R	R	R
4. Performance Modelling	B.45	-	R	R	HR	HR
5. Equivalence Classes and Input Partition Testing	B.19	-	R	R	HR	HR
6. Structure-Based Testing	B.58	-	R	R	HR	HR

Analisi Dinamica

Tecniche di analisi dinamica segnalate nella EN 50128

TECHNIQUE/MEASURE	Ref	SWS IL0	SWS IL1	SWS IL2	SWS IL3	SWS IL4
1. Test Case Execution from Boundary Value Analysis	B.4	-	HR	HR	HR	HR
2. Test Case Execution from Error Guessing	B.21	R	R	R	R	R
3. Test Case Execution from Error Seeding	B.22	-	R	R	R	R
4. Performance Modelling	B.45	-	R	R	HR	HR
5. Equivalence Classes and Input Partition Testing	B.19	-	R	R	HR	HR
STRUCTURED BASE TESTING	B.58	-	R	R	HR	HR

Sfida Attuale

La dimensione dei software è notevolmente aumentata

- Aumento delle performance hardware
- Progettazione di sistemi sempre più complessi
- Uso di generatori automatici di codice
- ...

Sempre più difficile verificare l'assenza di errori nel codice con l'uso delle tradizionali tecniche di analisi: simulazioni, test, code review. Servono tecniche alternative.

Sfida Attuale

La dimensione dei software è notevolmente aumentata

- Aumento delle performance hardware
- Progettazione di sistemi sempre più complessi
- Uso di generatori automatici di codice
- ...

Sempre più difficile verificare l'assenza di errori nel codice con l'uso delle tradizionali tecniche di analisi: simulazioni, test, code review. Servono tecniche alternative.

Abstract Interpretation

- Teoria dell'Approssimazione di insiemi di oggetti ed operazioni
- Analisi della semantica a diversi livelli di astrazione

Semantica di un programma

E' una descrizione matematica formale che definisce tutte le possibili esecuzioni del programma

Semantica di un linguaggio di programmazione

Definisce formalmente la semantica di tutti i possibili programmi scrivibili nel linguaggio di programmazione

Abstract Interpretation

- Teoria dell'Approssimazione di insiemi di oggetti ed operazioni
- Analisi della semantica a diversi livelli di astrazione

Semantica di un programma

E' una descrizione matematica formale che definisce tutte le possibili esecuzioni del programma

Semantica di un linguaggio di programmazione

Definisce formalmente la semantica di tutti i possibili programmi scrivibili nel linguaggio di programmazione

Abstract Interpretation

- Teoria dell'Approssimazione di insiemi di oggetti ed operazioni
- Analisi della semantica a diversi livelli di astrazione

Semantica di un programma

E' una descrizione matematica formale che definisce tutte le possibili esecuzioni del programma

Semantica di un linguaggio di programmazione

Definisce formalmente la semantica di tutti i possibili programmi scrivibili nel linguaggio di programmazione

Abstract Interpretation

- Semantica concreta del programma (descrive tutti i possibili comportamenti)
- Dominio Astratto che modella solo alcune proprietà delle computazioni concrete
- Semantica Astratta che verifica le proprietà del programma nel dominio astratto

Idea

La verifica che la semantica del programma soddisfi la sua specifica può essere effettuata a dei livelli di astrazione in cui vengono eliminati fattori irrilevanti e viene quindi ridotta la complessità del problema

Abstract Interpretation

- Semantica concreta del programma (descrive tutti i possibili comportamenti)
- Dominio Astratto che modella solo alcune proprietà delle computazioni concrete
- Semantica Astratta che verifica le proprietà del programma nel dominio astratto

Idea

La verifica che la semantica del programma soddisfi la sua specifica può essere effettuata a dei livelli di astrazione in cui vengono eliminati fattori irrilevanti e viene quindi ridotta la complessità del problema

Abstract Interpretation

- Semantica concreta del programma (descrive tutti i possibili comportamenti)
- Dominio Astratto che modella solo alcune proprietà delle computazioni concrete
- Semantica Astratta che verifica le proprietà del programma nel dominio astratto

Attenzione

L'astrazione deve essere *sound*, cioè se la semantica astratta soddisfa la specifica astratta allora la semantica concreta soddisfa la specifica concreta, e *completa*, cioè se la semantica concreta soddisfa la specifica concreta questo dovrebbe essere provabile anche in astratto.

Abstract Interpretation

Linguaggio che permette il prodotto tra numeri interi

Semantica Concreta del linguaggio:

η: Exp $\rightarrow$Z

η(n)=n $\quad\quad$ η(e1*e2)= η(e1)*η(e2)

Semantica Astratta che considera solo il segno delle espressioni

ρ: Exp $\rightarrow${-,0,+}

ρ(n)= - se n<0 $\quad$ ρ(n)= 0 se n=0 $\quad$ ρ(n)= + se n>0

ρ(e1*e2)= ρ(e1)$x^a\rho$(e2)

Funzione di concretizzazione Mappa un valore astratto in un insieme di valori concreti:

γ:{-,0,+}$\rightarrow$ P(Z) (insieme delle parti di Z)

Funzione di astrazione Mappa un insieme di valori concreti S nel valore astratto più preciso che rappresenta S

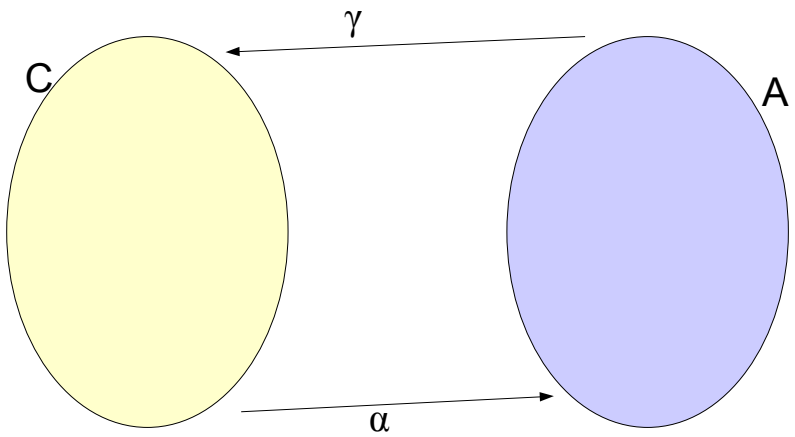
α : P(Z) $\rightarrow$A

Abstract Interpretation

Dominio Concreto C: Reticolo Completo

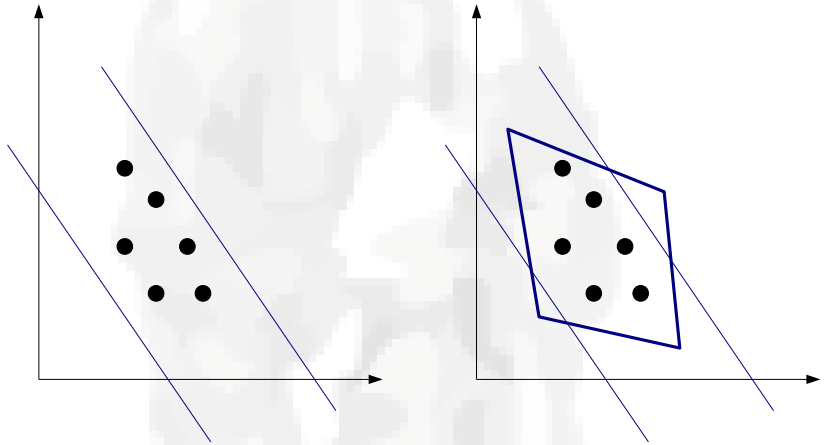
Dominio Astratto A: Reticolo Completo

Funzione di concretizzazione γ e Funzione di astrazione α:
Coppia di funzioni che definiscono una Connessione di Galois



Abstract Interpretation

- Sostituiamo il dominio concreto con quello astratto



- Reinterpretiamo le operazioni del programma nel dominio astratto

PolySpace

Tool distribuito da TheMathWorks che esegue l'analisi statica del codice sfruttando l'interpretazione astratta.

- Effettua la verifica di tutti i comportamenti possibili del codice
- Rileva staticamente la presenza di errori che possono portare a fallimenti a runtime
- Rileva la presenza di codice non raggiungibile

Come fa?

Esegue la verifica di un superset delle possibili computazioni ottenuto dall'utilizzo di approssimazioni superiori applicate agli elementi del dominio

PolySpace

Tool distribuito da TheMathWorks che esegue l'analisi statica del codice sfruttando l'interpretazione astratta.

- Effettua la verifica di tutti i comportamenti possibili del codice
- Rileva staticamente la presenza di errori che possono portare a fallimenti a runtime
- Rileva la presenza di codice non raggiungibile

Come fa?

Esegue la verifica di un superset delle possibili computazioni ottenuto dall'utilizzo di approssimazioni superiori applicate agli elementi del dominio

PolySpace

PolySpace rileva le principali categorie di errori runtime:

- Sforamento di array
- Accessi illegali tramite puntatori
- Lettura di variabili non inizializzate
- Underflow e Overflow su interi e su float
- Divisioni per zero
- .....

Dov'è la fregatura?

Lavora con delle overapproximation, quindi genera falsi positivi

PolySpace

PolySpace rileva le principali categorie di errori runtime:

- Sforamento di array
- Accessi illegali tramite puntatori
- Lettura di variabili non inizializzate
- Underflow e Overflow su interi e su float
- Divisioni per zero
- .....

Dov'è la fregatura?

Lavora con delle overapproximation, quindi genera falsi positivi

PolySpace

PolySpace rileva le principali categorie di errori runtime:

- Sforamento di array
- Accessi illegali tramite puntatori
- Lettura di variabili non inizializzate
- Underflow e Overflow su interi e su float
- Divisioni per zero
- .....

Dov'è la fregatura?

Lavora con delle overapproximation, quindi genera falsi positivi

PolySpace

Ogni istruzione può generare più errori, PolySpace quindi esegue un numero di check pari al numero dei possibili errori generabili dall'istruzione.

- Tool input: codice sorgente
- Tool output: codice sorgente colorato

Ogni possibile errore runtime è colorato sulla base delle informazioni di analisi

- In verde se non genera mai errori
- In rosso se in tutte le computazioni analizzate genera l'errore
- In arancione se almeno in una computazione genera l'errore
- In grigio se l'istruzione corrispondente non è mai stata raggiunta

PolySpace

Ogni istruzione può generare più errori, PolySpace quindi esegue un numero di check pari al numero dei possibili errori generabili dall'istruzione.

- Tool input: codice sorgente
- Tool output: codice sorgente colorato

Ogni possibile errore runtime è colorato sulla base delle informazioni di analisi

- In verde se non genera mai errori
- In rosso se in tutte le computazioni analizzate genera l'errore
- In arancione se almeno in una computazione genera l'errore
- In grigio se l'istruzione corrispondente non è mai stata raggiunta

Overapproximation

Approssimazioni effettuate dal tool:

- Generazione automatica di funzioni non chiamate
- Inizializzazione full-range delle variabili globali e static
- Interleaving totale delle chiamate generate

Osservazioni

Le approssimazioni implicano la verifica di comportamenti del codice non possibili durante la reale esecuzione.

Overapproximation

Approssimazioni effettuate dal tool:

- Generazione automatica di funzioni non chiamate
- Inizializzazione full-range delle variabili globali e static
- Interleaving totale delle chiamate generate

Osservazioni

Le approssimazioni implicano la verifica di comportamenti del codice non possibili durante la reale esecuzione.

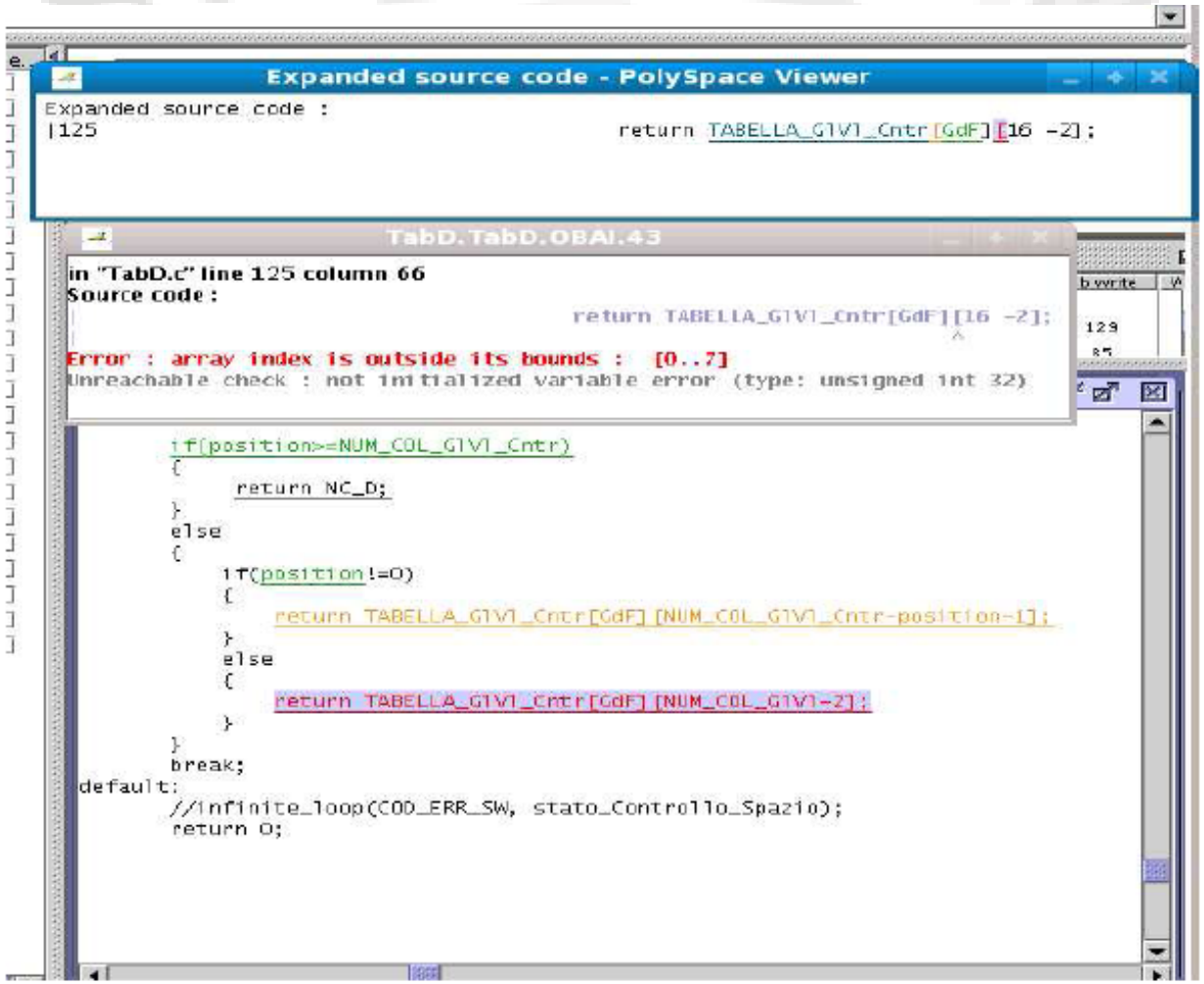
Overapproximation

- Attività principale analisi degli arancioni
- Classificazione degli arancioni sulla base della particolare approssimazione che li può aver generati
- Studio di vincoli da aggiungere al dominio di variazione delle variabili per disambiguare gli arancioni dubbi

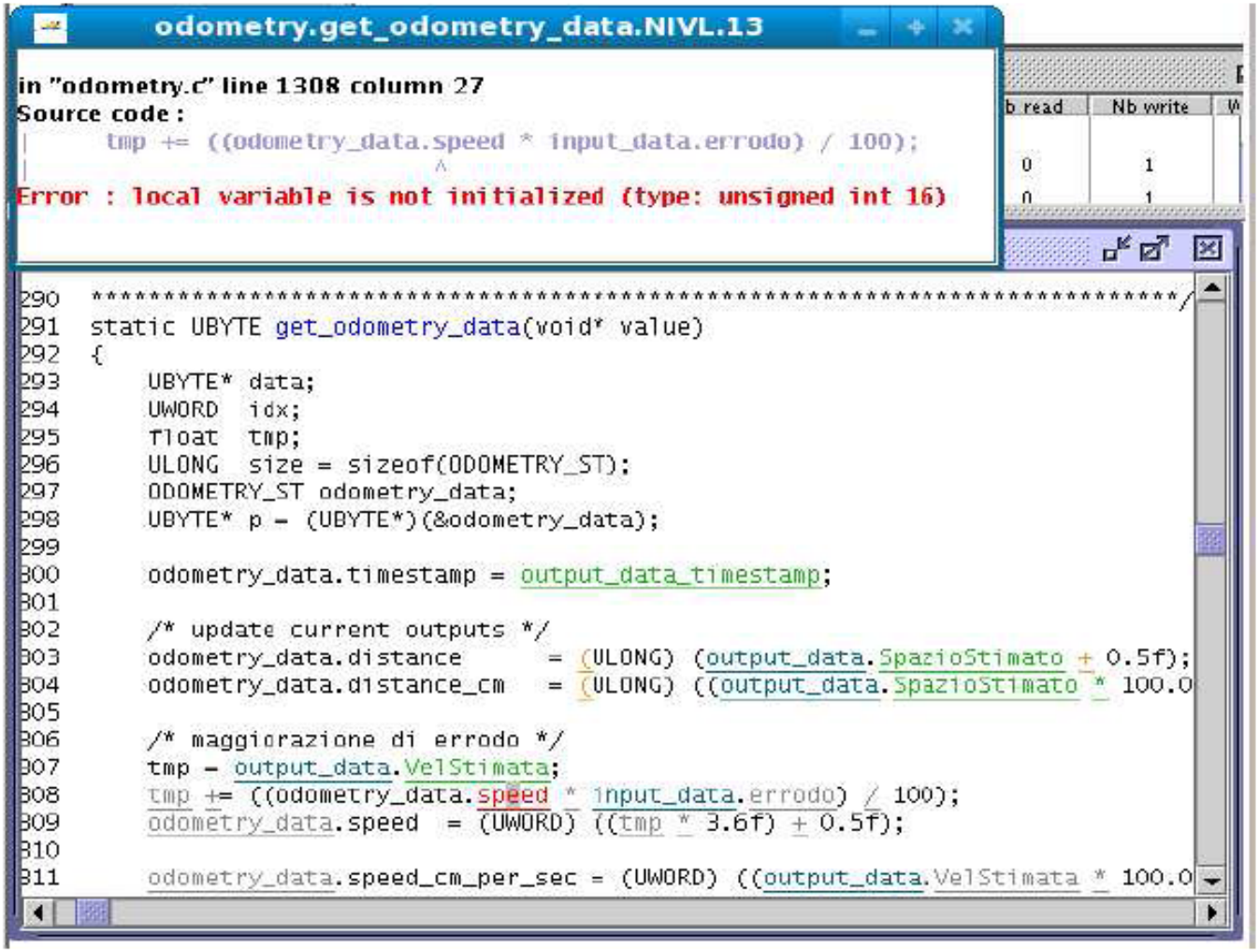
Vincoli:

- Range di variazione delle variabili globali e static
- Specifica del corretto interleaving delle chiamate
- Verifica di integrazione di più moduli contemporaneamente

Esempio1: Risultato Verifica



Esempio2: Risultato Verifica



Analisi Dinamica: CANTATA

CANTATA è un tool, costituito da più moduli eseguibili, che permette di effettuare:

- Testing funzionale (black box)
- Testing strutturale (white box)
  - Statement Coverage
  - Decision Coverage
  - Boolean Coverage
  - Path Coverage
- Analisi Statica del codice

Cosa Faremo:

Testing Strutturale con criterio di copertura All-Paths

Analisi Dinamica: CANTATA

CANTATA è un tool, costituito da più moduli eseguibili, che permette di effettuare:

- Testing funzionale (black box)
- Testing strutturale (white box)
  - Statement Coverage
  - Decision Coverage
  - Boolean Coverage
  - Path Coverage
- Analisi Statica del codice

Cosa Faremo:

Testing Strutturale con criterio di copertura All-Paths

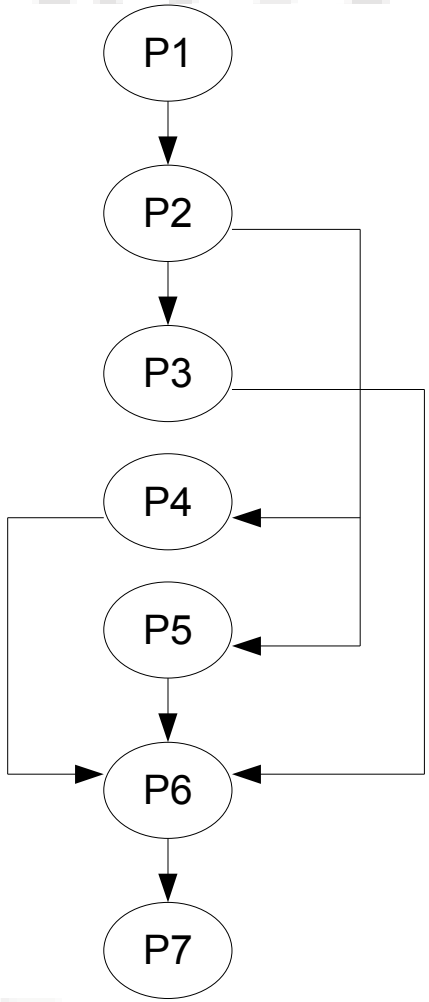
Esempio

Sorgente

```
int myfunct(int,int);
int myfunct(int a, int b){
   int result;
   if(a>b){
       result=a;
     }
   else if(a<b){
       result=b;
     }
   else{
       result=a*b;
     }
   return result;
}
```

Codice Instrumentato: All Paths

```
int myfunct(int,int);
int myfunct(int a, int b){
   int result;
/* P1 */
/* P2 */
   if(a>b){
      result=a;
/* P3 */
     }
   else if(a<b){
      result=b;
/* P4 */
     }
   else{
      result=a*b;
/* P5 */
     }
/* P6 */
/* P7 */
   return result;
}
```



Path e Test Suite

Path n°	Path	Copertura
01	P1>P2>P3>P6>P7	F
02	P1>P2>P4>P6>P7	F
03	P1>P2>P5>P6>P7	F

TEST SUITE 1

Path n°	Input
01	a = 15 , b = 10
02	a = 15 , b = 45
03	a = 50 , b = 50

TEST SUITE 2

Path n°	Input
01	a = 3000 , b = 120
02	a = 3 , b = 450
03	a = 1048576 , b = 1048576

Test: test suite 1

TEST SUITE 1

Path n°	Input
01	a = 15 , b = 10
02	a = 15 , b = 45
03	a = 50 , b = 50

```
Test   Script   Checks   Checks   Checks   Stubs   Paths   Assertions
       Errors   Passed   Failed   Warning  Failed  Failed  Failed
-------------------------------------------------------------------
PTE      0        0        0        0        0       0       0  PASS
001      0        1        0        0        0       0       0  PASS
002      0        1        0        0        0       0       0  PASS
003      0        1        0        0        0       0       0  PASS
-------------------------------------------------------------------
Total    0        3        0        0        0       0       0  PASS
===================================================================
```

Test: test suite 1

TEST SUITE 1

Path n°	Input
01	a = 15 , b = 10
02	a = 15 , b = 45
03	a = 50 , b = 50

Test	Script Errors	Checks Passed	Checks Failed	Checks Warning	Stubs Failed	Paths Failed	Assertions Failed	
PTE	0	0	0	0	0	0	0	PASS
001	0	1	0	0	0	0	0	PASS
002	0	1	0	0	0	0	0	PASS
003	0	1	0	0	0	0	0	PASS
Total	0	3	0	0	0	0	0	PASS

Test: test suite 2

TEST SUITE 2

Path n°	Input
01	a = 3000 , b = 120
02	a = 3 , b = 450
03	a = 1048576 , b = 1048576

```
Check FAILED : R_myfunct >>
Expected  0x0157531B  1099511627776
Item      0x00000000  0
```

```
Test   Script   Checks   Checks   Checks   Stubs   Paths   Assertions
       Errors   Passed   Failed   Warning  Failed  Failed  Failed
-------------------------------------------------------------------
--------
PTE       0        0        0        0        0       0        0    PASS
001       0        1        0        0        0       0        0    PASS
002       0        1        0        0        0       0        0    PASS
003       0        0        1        0        0       0        0 >> FAIL
-------------------------------------------------------------------
Total     0        2        1        0        0       0        0 >> FAIL
===================================================================
```

Test: test suite 2

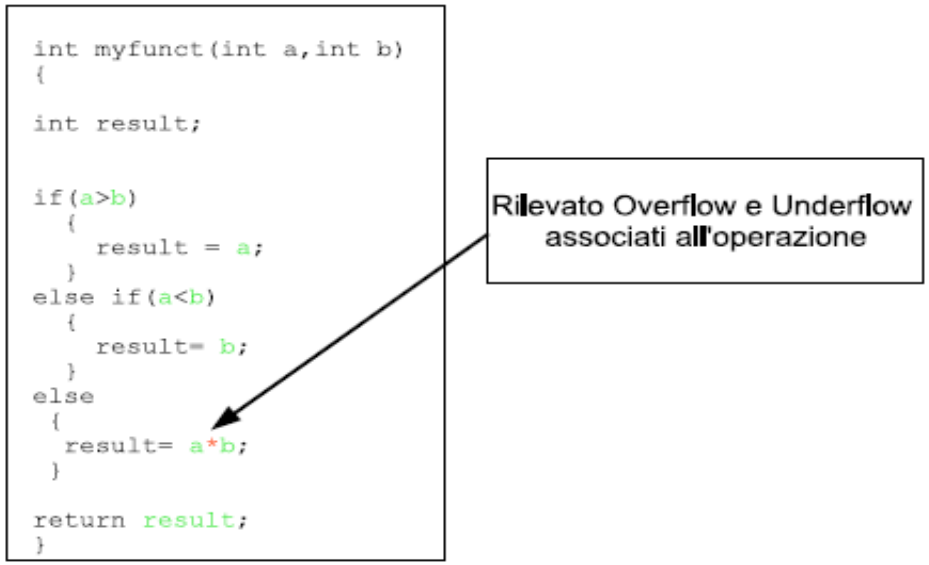
TEST SUITE 2

Path n°	Input
01	a = 3000 , b = 120
02	a = 3 , b = 450
03	a = 1048576 , b = 1048576

```
Check FAILED : R_myfunct >>
Expected  0x0157531B  1099511627776
Item      0x00000000  0

Test   Script   Checks   Checks   Checks   Stubs   Paths   Assertions
       Errors   Passed   Failed   Warning  Failed  Failed  Failed
-------------------------------------------------------------------
--------
PTE       0        0        0        0        0       0        0    PASS
001       0        1        0        0        0       0        0    PASS
002       0        1        0        0        0       0        0    PASS
003       0        0        1        0        0       0        0 >> FAIL
-------------------------------------------------------------------
Total     0        2        1        0        0       0        0 >> FAIL
===================================================================
```

Risultato PolySpace



- PolySpace rileva il possibile problema in corrispondenza del prodotto
- L'arancione indica che in almeno una computazione il tool ha rilevato overflow
- Identifica l'esatto punto in cui può verificarsi il problema

Conclusioni

- Abstract Interpretation soluzione alla crescente complessità software
- Confidenza assoluta sull'assenza di errori runtime nel codice in corrispondenza di input possibili
- Possibilità di verificare correttezza di procedure per determinati range di variazione degli input
- Verifica esaustiva a costi notevolmente inferiori rispetto al testing