Capitolo 1

Introduzione

1.1 Analisi statica

Analisi statica (alla SWE): analisi di un programma senza l'esecuzione a *runtime*. Permette di ottenere informazioni sul programma stesso.

L'obiettivo del corso di Verifica del *software* comprende l'utilizzo della correttezza per comprendere l'esecuzione di un programma a *runtime*. Permette di ottenere la piena garanzia di correttezza del programma.
La verifica può venire fatta per qualunque versione del codice sorgente (*byte-code* o altro). Viene svolta con analizzatori di codice o moduli. Quest'ultimi effettuano una verifica progressiva, parallela alla scrittura.
Polyspace: strumento di analisi di codice statico, dimostra l'esistenza di errori a *runtime* critici. Verifica codice C, C++ o Ada.
Un problema esistente è la mancata scalabilità della verifica.
Quando si scrive un analizzatore è bene utilizzare un linguaggio di programmazione robusto (come Ada). Tuttavia la quasi assenza di persone competenti in tale ambito incide sulla difficoltà di risolvere i problemi legati agli analizzatori.

Alcuni esempi di analizzatori:

- Interproc è un analizzatore accademico che inferisce invarianti;
- Jandom è un analizzatore Java, scritto in Scala (ovvero Java funzionale avanzato).

Di recente è l'impiego degli analizzatori per gli algoritmi di *Machine Learning*.

1.2 Motivazioni

Le motivazioni che portano allo studio dell'analisi statica:

1. Fallimenti *software*: caso di Ariane 5.
   Il razzo una volta lanciato in aria si autodistrugge.
   Si era verificato un *software error*, conversione tra virgola mobile a intero, con perdita d'informazione, che ha lanciato un'eccezione non catturata

causando la chiusura di tutti i programmi del razzo. L'ultimo programma eseguito è stato l'autodistruzione.

2. *Meltdown* e *Spectre*: i processori moderni usano l'esecuzione speculativa (tecnica di otimizzazione. L'eleaboratore esegue operazioni necessarie forse solo in un secondo tempo. `https://it.wikipedia.org/wiki/Esecuzione_speculativa`.) per velocizzare il lavoro. Un *team* di ricercatori ha su tale tecnica individuato un *bug* di sicurezza, causato delle *miss prediction* che lasciavano dati sensibili all'inteno delle *cache*, innescando eventuali *time attact*. L'Intel, precedentemente la pubblicazione della ricerca, avvisata dal *team*, ha mitigato il problema in modo *hardware*. Un modo per impedire la nascita di questi *bug* è svolgere già durante lo sviluppo del codice l'analisi statica.

Una buona tecnica, per punti, che permette di prevenire fallimenti del codice è la seguente:

- Scegliere un buon linguaggio;
- Svolgere progettazione;
- Effettuare *code testing*;
- Utilizzare un metodo formale di analisi statica. Questo permette la totale garanzia della correttezza del codice prodotto.