

Metode formale. Introducere

5 oct. 2004

- Funcționarea corectă a sistemelor. Importanță și dificultate.
- Ce sunt metodele formale ?
- Domenii și exemple de aplicație
- Modelare și specificare

Verificare formală. Curs 1

Marius Minea

Metode formale. Introducere

3

Erori grave: Therac-25

- aparat medical pentru terapie cu radiație
- 6 accidente cu morți și răni grave (1985-87, SUA, Canada)
- cauza directă: erori în programul de control

Analiză retrospectivă [Leveson 1995]:

- încredere excesivă în software (în analiza produsului)
- **fiabilitate \neq siguranță**
- lipsa măsurilor de siguranță hardware
- lipsa practicilor de **ingineria programării** (proiectare defensivă, specificare, documentație, simplitate, analiză formală, testare)
- **corectarea unei erori nu face sistemul mai sigur !!**

Verificare formală. Curs 1

Marius Minea

Metode formale. Introducere

5

Eroarea din procesorul Pentium

Eroare în unitatea de împărțire cu virgulă mobilă, 1994

- algoritmul de împărțire cu refacere, în baza 4
- determină următoarea cifră din cât dintr-un tabel
- câteva intrări marcate greșit ca "don't care"
- cost: cca. 500 milioane dolari
- **Analiză ulterioară**
- Circuitul putea fi verificat formal
 - demonstrare automată de teoreme [Clarke, German & Zhao]
 - cu structuri speciale de date pentru reprezentarea înmulțirii/împărțirii [Bryant & Chen]
- dar accentul a fost pus pe componente mai complexe (unitatea de execuție, coerența memoriei cache)

Verificare formală. Curs 1

Marius Minea

Obiectivele cursului

- urmărirea *corectitudinii* sistemelor proiectate
- cunoașterea principalelor tipuri și surse de *erori*
- cunoașterea *metodelor formale* ca alternativă la simulare și testare
- obișnuința cu *rigoarea* în descrierea sistemelor
- construirea de *modele* corespunzătoare pentru sistemele proiectate
- exprimarea *neambiguă* a specificațiilor (proprietăților dorite)
- *evaluarea aplicabilității* metodelor formale (manuale sau automate)
- cunoașterea unor *utilitare* de verificare

Verificare formală. Curs 1

Marius Minea

Metode formale. Introducere

4

Erori: racheta Ariane 5

- Autodistrugere după o defecțiune la 40 s de la lansare (1996)
- Cauza: conversia 64-bit float \rightarrow 16-bit int generează o excepție de depășire netratată în programul ADA
- Cost: 500 milioane dolari (racheta), 7 miliarde dolari (proiectul)

Analiză retrospectivă

- principala cauză: **reutilizarea nejudicioasă de software**
- cod preluat de la Ariane 4, fără reanalizare corespunzătoare:
 - execuția nu mai era necesară în timpul erorii
 - analiza absenței depășirii pentru variabilele neprotejate
 - \Rightarrow **necesitatea specificării și respectării unei interfețe**
- proiectarea greșită a **redundanței**: sistemul de referință inerțial și cel de rezervă scoase din funcțiune de aceeași eroare

Verificare formală. Curs 1

Marius Minea

Metode formale. Introducere

6

Erori: probele trimise pe Marte

Mars Pathfinder, 1997

- ajunsă pe Marte, proba spațială se reseta frecvent
- cauza: **inversiune de prioritate** între procese cu resurse comune
- fenomenul și soluția: cunoscute în literatura de specialitate ! [Sha, Rajkumar, Lehoczky. Priority Inheritance Protocols, 1990]

1. procesul A de prioritate mică obține resursa R
 2. A întrerupt de C (prioritate mare)
 3. C așteaptă eliberarea lui R; A revine în execuție
 4. A întrerupt de B (prioritate medie, $A < B < C$)
- \Rightarrow C așteaptă după B, fără a fi direct condiționat de B !

Soluția: ridicarea priorității unui proces care obține o resursă (A) la nivelul celui mai prioritar proces care poate solicita resursa (C)

Verificare formală. Curs 1

Marius Minea

Erori: probele trimise pe Marte

Mars Climate Orbiter, 1998

- dezintegrare la intrarea în atmosferă
- eroarea tehnică: discrepanța între unități de măsură în sistemele anglo-american și metric
- erori multiple de proces: **lipsa unor interfețe formale**

Mars Polar Lander, 1998

- trenul de aterizare e activat prematur la intrarea în atmosferă
- șocul e interpretat ca aterizare, motoarele sunt oprite
- eroarea: **lipsa testării de integrare**

Verificare formală. Curs 1

Marius Minea

Ce sunt metodele formale?

“... limbaje, tehnici și unelte matematice pentru specificarea și verificarea sistemelor” [Clarke & Wing, 1996]

Sau, mai în detaliu: “un set de unelte și notații,
– cu o semantică formală,
– folosite pentru a specifica neambiguu cerințele unui sistem,
– care admite demonstrarea de proprietăți ale acelei specificații
– și demonstrarea corectitudinii unei implementări în raport cu acea specificație”

[Hinchey & Bowen, *Applications of Formal Methods*, 1995]

Verificare formală. Curs 1

Marius Minea

Metode formale: Necesitate și dificultăți

Utilitate îndeosebi pentru:

- **complexitate**: tehnici de abstracție / aproximare
- **concurență**: foarte greu de reprodus și analizat altfel
- **criticalitate**: (avionică, bancar, medical, securitate)

Dinamica erorilor în software [după John Rushby, SRI]

- 20-50 erori/kloc înainte de testare, 2-4 după
- examinarea formală a codului poate reduce erorile înainte de testare de cca 10 ori

Studiu de caz pe 10kloc timp real, distribuit:

- verificare și validare: 52% cost (57% timp)
- din acesta, 27% cost în examinare, 73% în testare
- 21% pt. 4 defecte în testarea finală, din care 1 de proiectare
- eliminarea erorilor la examinări detaliate ale codului:

de 160 de ori mai eficientă decât la testare

Verificare formală. Curs 1

Marius Minea

Cum se pot detecta erorile ?

Testare

- + direct pe produs ⇒ teste concludente
- erorile detectate târziu sunt costisitoare
- diagnosticul necesită observabilitate completă

Simulare

- + efectuată deja în faza de proiectare
- simulatorul poate fi mai lent decât sistemul real

- Testarea sau simularea exhaustivă e adeseori imposibilă

Testarea poate demonstra prezența erorilor, dar nu absența lor.
(E. W. Dijkstra, 1979)

Verificare formală. Curs 1

Marius Minea

Ce pot garanta metodele formale?

- Nu există garanții absolute
- O metodă formală nu poate fi mai bună decât modelul și specificațiile care sunt folosite
 - *modelul și specificațiile* trebuie validate

Dar pot oferi:

- o procedură logic consistentă de raționament
 - o acoperire exhaustivă, adeseori imposibilă altfel
 - mecanizare și automatizare ⇒ performanță și corectitudine
- Pot **complementa** cu succes simularea, testarea, etc.*

Verificare formală. Curs 1

Marius Minea

Cauzele și costul erorilor (cont.)

Erori în programe

[după NASA JPL (sondele Voyager, Galileo)]

- majoritatea: deficiențe în specificarea cerințelor și a interfețelor
- 1 eroare la 3 pagini de cerințe și 21 pagini de cod
- doar 3 din 197 erau erori de programare
- 2/3 din erorile funcționale: omisiuni în specificarea cerințelor
- majoritatea erorilor de interfață: datorate proastei comunicări

Erori în hardware

[după Kurt Keutzer, UC Berkeley]

- > 50% din proiecte au erori după prima fabricare
- erorile pe linie de cod hardware trebuie reduse în ritmul creșterii numărului de tranzistori pe chip
- practic zero erori la nivelele inferioare de implementare
- problemele sunt în proiectarea la nivel înalt
(hazarduri în pipeline, execuție nesecvențială, procesoare superscalare, coerența memoriilor cache, protocoale complexe, etc.)

Verificare formală. Curs 1

Marius Minea

O privire de ansamblu

- Cauzele cele mai frecvente de erori: din concepție, defecte simultane, interacțiuni neprevăzute
 - lipsurile principale: în aplicarea timpurie a metodelor formale
 - costul principal: eliminarea târzie a erorilor
- Potențialul maxim al metodelor formale:
 - în modelarea și verificarea la nivel înalt
 - pentru sisteme complexe, concurente, distribuite, reactive, în timp real, tolerante la eroare, etc.

Verificare formală. Curs 1

Marius Minea

Metodele formale în ciclul de producție

- Analiza cerințelor.
 - identifică contradicții, ambiguități, omisiuni
- Proiectare
 - descompunerea în componente și specificarea interfețelor
 - proiectarea prin rafinare succesivă
- Verificare
- Testare și depanare
 - generarea direcționată de cazuri de test
- Analiză
 - model abstract, mai puțin complex decât sistemul real

Verificare formală. Curs 1

Marius Minea

Aplicații: Proiectare hardware

- Verificarea echivalenței combinatoriale
 - este deja standard în utilitarele CAD
- Verificarea circuitelor secvențiale
 - marile companii au colective speciale (IBM, Intel, Motorola, Fujitsu, Siemens, etc.)
 - folosesc verificatoarele disponibile public sau cele proprii
- protocoalele de coerență cache Gigamax și Futurebus+
- Motorola 68020: modelat în demonstratorul Boyer-Moore, verificarea codul binar produs de compilatoare
- AAMP-5 (procesor pentru avionică): modelat în PVS, verificarea microcodului pentru instrucțiuni
- procesoare cu pipelining / superscalare tip DLX

Verificare formală. Curs 1

Marius Minea

Aplicații: Avionică

Lockheed C130J

- analiza codului ADA cu anotații în limbajul SPARK
- software "corect prin construcție", cost redus
- TCAS-II (Traffic Collision Avoidance System)
 - instalat obligatoriu pe avioanele comerciale din S.U.A.
- alertă și deviere automată în cazul apropierii periculoase
- specificația redactată într-un limbaj formal (RSML)
- s-a verificat completitudinea și consistența [Heimdahl, Leveson '96]
- s-a abandonat încercarea de a specifica în engleză
 - Modele formale pentru sisteme complexe sunt fezabile
 - Pot fi analizate de experții din domeniul de aplicație

Verificare formală. Curs 1

Marius Minea

Alte Aplicații

- Telefonie. Specificarea și analiza interacțiilor între diversele caracteristici ale sistemului telefonic.
- Sisteme electronice de consum. Verificarea manuală, apoi automată a protocolului de control din componentele audio Philips,
- Sisteme de control în electronica auto.
- Protocoale de comunicație.
- Protocoale de securitate. Analiza folosind logici speciale pentru a raționa despre mesaje criptate, intruși, etc.
- Software de sistem. Verificarea driverelor de periferice.

Verificare formală. Curs 1

Marius Minea

Metode formale: Specificare

- Specificarea: necesară în orice metodă formală (poate fi unicul aspect)
 - necesită limbaj cu **sintaxă** și **semantică** definită formal (matematic)
 - Limbajul de specificare definește:
 - un domeniu sintactic (notația)
 - un domeniu semantic (universul de obiecte considerat)
 - o definiție precisă a obiectelor care satisfac o specificație
- [M. Chechik, *Automated Verification*, curs, U. Toronto]

Verificare formală. Curs 1

Marius Minea

Sintaxă și semantică

Sintaxa

- un alfabet de simboluri (ex. propoziții, operatori logici)
- reguli gramaticale pentru formarea unor formule bine definite

Semantica

- Domeniul semantic variază de la limbaj al limbaj:
- secvențe de stări, secvențe de evenimente, structuri de sincronizare (în limbajele de specificare a sistemelor concurente)
 - funcții intrare → ieșire, relații, computații, transformatoare de predicate (în cazul limbajelor de programare)

Proprietăți ale specificațiilor

- neambiguă: un înțeles bine definit (NU: limbaj fără semantică formală, limbaj natural, scheme grafice cu mai multe interpretări)
- consistentă (necontradictorie)
 - există cel puțin un obiect care o satisface
- poate fi incompletă
 - comportament la latitudinea implementării sau nondeterminism

Dacă limbajul are un sistem de *inferență logică* se pot *demonstra* proprietăți pornind de la o specificație.

Specificare: Limbajul Larch

[Guttag, Hornig, Garland, MIT/DEC SRC]

1. abstracție (specificare) independentă de limbaj
2. specificare de interfață pentru module într-un anumit limbaj

```
Table: trait
  includes Integer
  introduces
    new: -> Tab
    add: Tab, Ind, Val -> Tab
    lookup: Tab, Ind -> Val
  asserts \forall i, i1: Ind, v: Val, t: Tab
    \not (i \in new);
    i \in add (t, i1, v) == i = i1 \/\ i \in t
    lookup(add(t, i, v), i1) ==
      if i = i1 then v else lookup(t, i1)
```

Tipuri de specificații

- **declarative** (nu trebuie să reprezinte o funcție calculabilă)
- **executabile** (ex. limbajele de programare)
- **comportamentale** (orientate pe proprietăți) (ex. funcționalitate, reactivitate)
 - descriu comportamentul sistemelor în raport cu proprietățile ce trebuie satisfăcute
- **structurale** (orientate pe modele) (ex. diagrame, conexiuni, ierarhie)
 - construiesc un model al sistemului folosind noțiuni matematice precise (mulțimi, funcții, logica predicatelor)

Uneori: același limbaj pentru specificație și model (sau implementare)
⇒ e posibilă rafinarea pe nivele succesive de abstracție

Specificare: Limbajul Z

- bazat pe logica de ordinul I și teoria mulțimilor
- descriere funcțională, declarativă
- folosit extensiv la proiecte industriale în Marea Britanie

<pre>PhoneDB members : PPerson telephones : Person ↔ Phone</pre>	<pre>FindPhones ≡PhoneDB name? : Person numbers! : PPhone</pre>
<pre>dom phones ⊆ members</pre>	<pre>name? ∈ dom phones numbers = phones({name?})</pre>

– o *schemă* (PhoneDB) (stări + evtl. tranziții), și un *invariant*

– operații care schimbă starea (Δ) sau nu (Ξ)

Limbajul Larch (cont.)

Specificare de interfață pentru limbajul C:

```
mutable type table
uses Table(table for Tab, char for Ind,
           char for Val, int for Int);
constant int maxTabSize;
table table_create(void) {
  ensures result' = new /\ fresh(result);
}
char table_read(table t, char i)
  requires i \in t^;
  ensures result = lookup(t^, i);
}
```

- definește precondiții și postcondiții
- interfața rămâne la nivelul abstract (fără algoritmi)

Specificare: alte limbaje

VDM (Vienna Development Method)

- originează din eforturile grupului IBM Viena în anii '70
- similară și înrudită cu Z

B

- dezvoltată de Jean-Raymond Abrial
- spre deosebire de Z, are și suport automatizat puternic
- precondiții / postcondiții, invarianți, rafinament
- suport pentru generarea automată de cod
- utilizare industrială (metroul din Paris, Alstom, n · 10kloc)

Noțiuni de specificare de interfață încorporate direct în limbaje, de ex. Eiffel (design by contract)

Verificare formală. Curs 1

Marius Minea

Modelare: Communicating Sequential Processes (CSP)

Exemplu [Hoare]: automat pentru ciocolată, cu monede

Alfabetul: $\alpha_V = \{in1p, in2p, small, large, out1p\}$

Comportamentul:

$$V = (in2p \rightarrow (large \rightarrow V | small \rightarrow out1p \rightarrow V) \\ | in1p \rightarrow small \rightarrow V)$$

sau, formal:

$$V = \mu X. (in2p \rightarrow (large \rightarrow X | small \rightarrow out1p \rightarrow X) \\ | in1p \rightarrow small \rightarrow X)$$

(unica soluție a ecuației de mai sus)

CSP: formalism (algebră a proceselor) axat pe acțiuni, cu nondeterminism, compoziție sincronă, etc.

Verificare formală. Curs 1

Marius Minea

Noțiunea de corectitudine

- În general: sistemul **satisface o proprietate** (specificație)
- Comportament **funcțional** corect.
 - sistemul privit ca realizând o funcție intrare \rightarrow ieșire
 - exemplu de formalism: tripletele lui Hoare

$$\{P\} S \{Q\}$$

{ precondiție } program(sistem) { postcondiție }

Exemplu de raționament:

$$\frac{\{P\} S_1 \{Q_1\} \quad Q_1 \Rightarrow Q_2 \quad \{Q_2\} S_2 \{R\}}{\{P\} S_1; S_2 \{R\}}$$

Verificare formală. Curs 1

Marius Minea

Modelarea sistemelor concurente

Două direcții de abordare:

- programare imperativă + adăugiri (semafoare, monitoare, rendezvous, etc.)
- model de calcul concurent, bazat pe interacțiunea proceselor ("interacțiune indivizibilă")

Comunicarea și concurența sunt noțiuni complementare [Milner]

- Communicating Sequential Processes [Hoare]
- Calculus of Communicating Systems [Milner]

Verificare formală. Curs 1

Marius Minea

Modelare: Automate cu stări finite

- Variante:
 - etichete pe stări sau pe tranziții
 - tranziții specificate ca funcții sau relații
 - augmentat sau nu cu variabile (date)
- Structură Kripke:
 - = automat etichetat cu *propoziții atomice* dintr-o mulțime AP :

$$M = (S, S_0, R, L)$$

- S : mulțime finită de stări
- S_0 : mulțimea stărilor inițiale
- $R \subseteq S \times S$: relație de tranziție totală
- $L : S \rightarrow 2^{AP}$: funcție de etichetare a stărilor

Verificare formală. Curs 1

Marius Minea

Noțiunea de corectitudine (cont.)

Comportament **temporal** corect

- pentru sisteme *reactive*: execuție (conceptual) infinită
- comportamentul definit ca reacție la o secvență de intrare
- specificare: de ex. logică temporală
- proprietăți: absența blocajului, reacție în timp limitat, etc.

Exemple:

- orice cerere este urmată de un răspuns în cel mult 5 secunde
- orice proces obține resursa de un număr infinit de ori
- pe orice traiectorie, se ajunge la un moment dat în starea stabilă

Verificare formală. Curs 1

Marius Minea

Procedee de verificare

Două mari categorii:

Explorarea spațiului stărilor (model checking)

- specificarea de regulă în logică temporală
- algoritmi de explorare exhaustivă verifică valoarea de adevăr sau produc o secvență de execuție ca și contraexemplu
- verificarea echivalenței: specificarea e la rândul ei un model

Demonstrarea de teoreme

- reprezentare într-un sistem logic cu axiome și reguli de deducție
- domeniul analizat reprezentat și el printr-un grup de axiome și reguli (o teorie)
- demonstrare mecanizată: ghidată manual sau automată