正規描述與自動驗證

Formal Description & Automated Verification

王　凡
國立台灣大學 電機工程系

1

產業升級壓力下的一代

- 大前研一（未來分析家）：「台灣✓✓論
  」*的優勢只剩下五年。*
- 杜書伍（聯強國際）
  年優勢
- 區… *台灣和大陸的*
  還有五年的差距，……

台灣未來產業的優勢在哪裡？
各位五年後的競爭力在哪裡？

2

Verification（驗證）?

- 找出系統設計中的所有錯誤。
- 確認系統中已經（接近）沒有錯誤。

非常困難！
複雜系統的決勝關鍵！
各位同學的一條生路！
台灣產業的一條生路！

3

簡介

- 瞭解正規指定與電腦輔助技術的理論與製作

4

方法：

- **建立基本的電腦輔助驗證的知識。**
  - 前面數節課，教授基礎知識。
- **互動教學，聊解最新動態。**
  - 依同學人數，排定論文研讀報告。
- **學期報告，深入探討技術問題。**
  - 研究問題探討、實驗系統製作。
  - 報告中要顯示適當的努力，以解決問題。

5

教學進度規畫：

1、9/16：課程簡介、問卷調查
2、9/23：自動驗證/分析模型的建立
3、9/30：學期計畫討論
4、10/9：自動驗證/分析模型的建立與討論
5、10/16：古典時態邏輯與自動機理論的定義
6、10/23：真時自動機理論與模型建立
7、10/30：混和自動機與各種無限狀態空間系統的模型
8、11/6：期中考
9、11/13：表達能力與計算複雜度

6

教學進度規畫（續）：

10、11/20：符號式自動驗證技術
11、 11/27：複雜度的管理、組合式驗證
12、 12/4：無限狀態系統的自動驗證技術
SOC 自動驗證技術
13、 12/11：學期實習計畫報告
14、 12/18： 學期實習計畫報告
15、 12/25：論文閱讀報告
16、2004/1/1：放假
17、1/8：論文閱讀報告
18、1/15：期末考

7

課程網頁

http://cc.ee.ntu.edu.tw/~farn/courses/FMV/

8

參考資料：

- **Formal Methods for Real-Time Systems**
  edited by C. Heitmeyer, D. Mandrioli, Wiley
- **Chapter 16: Temporal and Modal Logic E. Allen Emerson, in** *Handbook of Theoretical Computer Science***,** edited by J. van Leeuwen, Elsevier.
- **Model Checking,** E. Clarke, O. Grumberg, D. Peled, MIT Press
- 重要論文

9

基本資料調查　921 U4000 *電腦輔助驗證*　王凡　民92年秋季

- 請問您的大名：
- 請問您的系所、年級？
- 您曾修過任何邏輯的課程嗎？請略述之。
- 你曾修過資料結構、演算法嗎？
- 您曾修過計算理論（Theory of Computation）嗎？
- 您爲何想要修本門課？
- 您未來的生涯規畫？
- 您有任何其他事，想要我知道嗎？

10

規格指定、描述 vs 驗證

- **規格指定（specification) :**
  - 使用者對系統的要求。
- **行爲描述（description）：**
  - 使用者對系統行爲的描述
  - 與specification間，並無一定的區隔
- **驗證 （verification) :**
  - 系統設計有達到指定規格嗎？

11

正規指定、描述 vs 自動驗證

- **正規 （formal) 規格指定 :**
  - 以數學的嚴謹語意表達規格指定
- **自動 （automatic) 驗證 :**
  - 以電腦輔助（computer-aided) 技術進行驗證

12

爲什麼要研究正規描述方法 ？

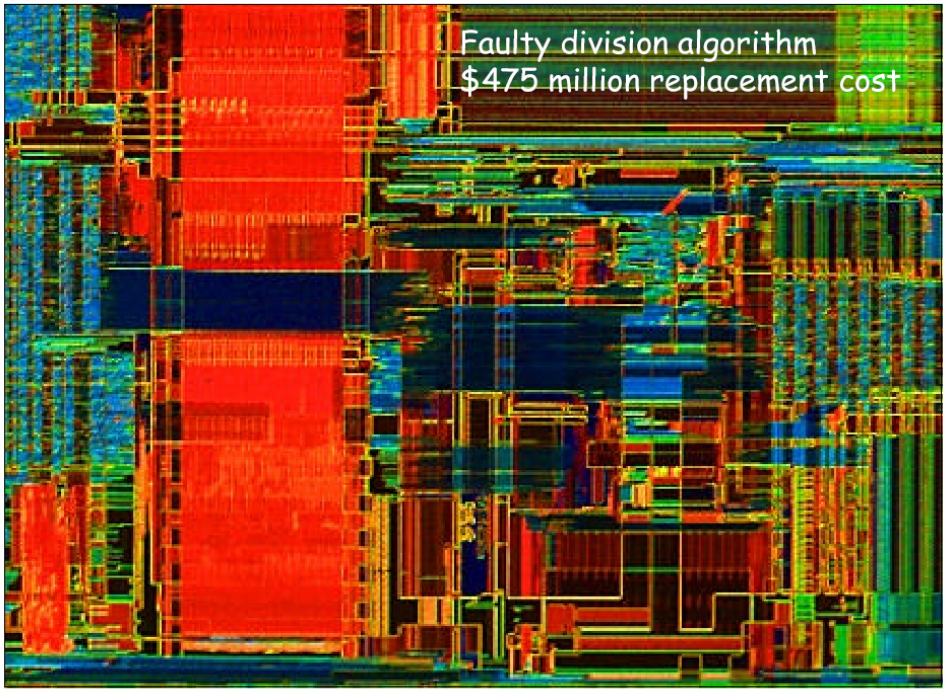
- 經由數學化的嚴謹定義，讓工程師、使用者更深入瞭解要設計的系統
- 減少自然語言的含混，減少溝通、討論、閱讀時的相互誤解
- 對系統的抽象精確描述
- 產生嚴謹的數學模型，以利正規驗證的分析

13

爲什麼要研究自動驗證技術 ？

- 系統設計愈趨複雜
- 解除大型系統設計之負擔
  - 釋放人類的創造力
- 避免設計錯誤晚期修正的龐大成本
- 未來驗證技術的核心

14



Pentium Bug

- **Floating Point divide**
  - 應有十八位數準確性
  - 只有四位數準確性
  - Pentium 60MHz、90MHz
  - 範例：

    5505001 / 294911

    錯誤答案：18.66600093

    正確答案：18.6665197

16

Pentium Bug（續）

- 只對特定數對（**number pair**）產生錯誤
- 重複發生。
- 影響大型科學、統計、工程計算、試算表、模擬、等應用。
- 由**Pentium compile**的軟體，都有可能受影響。

17

Pentium Bug（續）

- 由**Lynchburg College**的**Dr. Thomas R. Nicely**首度發現
  - nicely@acavax.lynchburg.edu
- **Compuserve** 網路，於 **10/30/1994**發佈
- **First in printed media on 11/7/1994**
- 在**1994**年中，**bug**已經排除。但**Intel**在年底才對大客戶提供正確的晶片。要退貨，需個別處理，證明確實發生錯誤

18

Pentium Bug（續）

- 引發了**formal verification**研究的熱潮
- 目前**Intel**維持了一個龐大的**formal method**研究團隊。其規模與組織，爲業務機密。
- 鹹魚翻身！
  - 許多研究經費流入。
  - 適時的理論、實驗突破。

19

THE "BUG" HEARD 'ROUND THE WORLD

Discussion of the Software Problem Which Delayed the First Shuttle Orbital Flight

John R. Garman

Deputy Chief

Spacecraft Software Division

NASA, Johnson Space Center

Houston, Texas

Aug 24, 1981

ACM SIGSOFT Software Engineering Notes,

Vol. 6, Nr. 5, Oct, 1981

20

THE "BUG" HEARD 'ROUND THE WORLD（續）

Discussion of the Software Problem Which Delayed the First Shuttle Orbital Flight

- **4/10/1981,** 太空梭預定第一次發射前二十分鐘，發現第五套備用電腦不能**initialize**
- 當時太空梭上，有四套**General Purpose Computer**（**GPC**）以及一套備用電腦。
- 以**FO/FS**爲容錯設計要求
  - 一套GPC出錯，仍能Vote；兩套錯，仍能安全返航

21

THE "BUG" HEARD 'ROUND THE WORLD（續）

Discussion of the Software Problem Which Delayed the First Shuttle Orbital Flight

- **GPC**與備用電腦上的程式系統，由不同人員發展。
- cyclic processing
- **GPC**上的程式在預定發射前，已經跑了三十個小時，未嘗出錯。

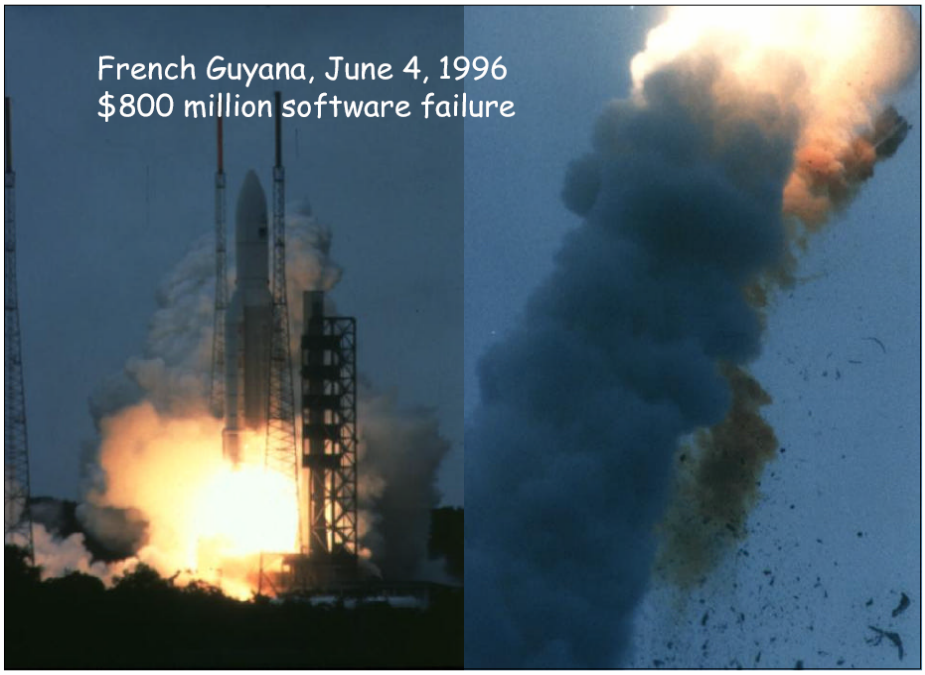
22

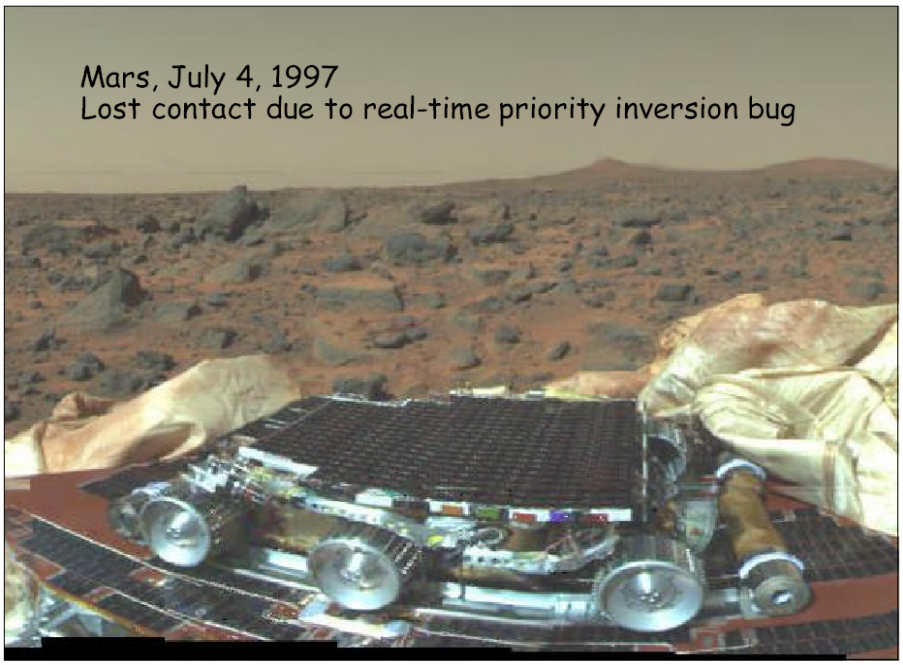
THE "BUG" HEARD 'ROUND THE WORLD（續）

Discussion of the Software Problem Which Delayed the First Shuttle Orbital Flight

- 在錯誤發生後一個小時，**IBM將GPC中的memory dump出來，發現一個Software bug，timing synchrony出了錯誤。**
- **GPC中有些processes已經out-of-phase了**
- **備份電腦抓不到out-of-phase的資料，判斷GPC已經出錯。**

23





Therac-25 Incidents

- Medical linear accelerator by AECL
- Computer-controlled (DEC PDP-11)
- Dual modes of X-ray and electron beams
- Successor to Therac-20 and Therac-6 by AECL, used CGR
- available in late 1982
- 11 Therac-25 were installed

6 accidents with death and serious injuries from 6/1985 to 1/1987

26

Therac-25 incidents (continued, 2)

- **Independently developed by AECL after breaking up with CGR**
- **A fault-tree safety analysis was performed with the assumption that software was correct.**
- **Controlled by legacy software from Therac-20 and Therac-6**
  - Therac-20 and -6 only used computer for convenience
  - Get rid of hardware interlock since software never went wrong with Therac-20 and Therac-6
    - ***In fact, most software errors of Therac-20 and Therac-6 had been masked by hardware interlocks.***

27

Therac-25 incidents (continued, 3)

- **Error message happened so often that technicians thought they were normal.**
- **Most of the errors did not hurt.**
- **The AECL said**
  - "Improper scanning was not possible!"
  - "This incident was never reproted to AECL prior to this date …" (after 10 months of a filed lawsuit)

28

Therac-25 incidents (continued, 4)

- **On May 2, 1986, FDA declared Therac-25 defective and demanded CAP.**
- **AECL remedied something and claimed that Therac-25 was 10,000 times safer.**
- **FDA believed them.**
- **Software errors have been identified in all these six admitted accidents.**
- **Finally, the hardware interlocks were put back in on Feb. 2, 1987.**

29

Therac-25 incidents (continued, 5)

- **Worst accidents series in 35-year history of medical accelerator**
- **References:**
  - N. Leveson, C.S. Turner, *An investigation of theTherac-25 accidents*, IEEE Computer, Vol. 26, Nr. 7, July 1993, pp.18-41

30

GOVERNMENT NEWS GCN July 13, 1998

Software glitches leave Navy Smart Ship dead in the water

Gregory Slabodkin, GCN Staff

"Using Windows NT, which is known to have some failure modes, on a warship is similar to hoping that luck will be in our favor," DiGiorgio said.

Some more bugs (1)

- *Mars climate orbiter smashed into the planet instead of reaching a safe orbit ($165M), 1999*
  - **Failure to convert English measures to metric values**
  - **Software shut the engine off 100ft above the surface.**
- *US Vicennes mistook airbus 320 for a F-14 and shot it down, 1st Gulf War, 1988.*
  - **290 people dead**
  - **Why: Software bug - cryptic and misleading output displayed by the tracking software**

32

Some more bugs (2)

Failure of the London Ambulance Service on 26 and 27 November 1992

- **Load increased**
- **Emergencies accumulated**
- **System made incorrect allocations**
  - **more than one ambulance being sent to the same incident**
  - **the closest vehicle was not chosen for the emergency**
- **At 23:00 on October 28 the LAS eventually instigated a backup procedure, after the death of at least 20 patients**

33

Some more bugs (3)

- British destroyer H.M.S. Sheffield; sunk in the Falkland Islands war
  - **ship's radar warning system software allowed missile to reach its target**
- An Air New Zealand airliner crashed into an Antarctic mountain
- North American Aerospace Defense Command reported that the U.S. was under missile attack;
  - **traced to faulty computer software - generated incorrect signals**
- Manned space capsule Gemini V missed its landing point by 100 miles;
  - **software ignored the motion of the earth around the sun**

["*The development of software for ballistic-missile defense*," by H. Lin, Scientific American, vol. 253, no. 6 (Dec. 1985), p. 48]

34

Some more bugs (4)

- An error in an aircraft design program contributed to several serious air crashes

  [“Software Engineering: Report on a Conference sponsored by the NATO Science Committee, Brussels NATO Scientific Affairs Division,” 1968, p. 121]

- Dallas/Fort Worth air-traffic system began spitting out gibberish in the Fall of 1989 and controllers had to track planes on paper

  ["Ghost in the Machine," Time Magazine, Jan. 29, 1990. p. 58]

35

Some more bugs (5)

- **F-18 fighter plane crashed**
  - due to a missing exception condition

  [ACM SIGSOFT Software Engineering Notes, vol. 6, no. 2]
- **F-14 fighter plane was lost**
  - to uncontrollable spin, traced to tactical software

  [ACM SIGSOFT Software Engineering Notes, vol. 9, no. 5]
- **Chicago cat owners were billed $5 for unlicensed dachshunds.**
  - A database search on "DHC" (for dachshunds) found "domestic house cats" with shots but no license

  [ACM SIGSOFT Software Engineering Notes, vol. 12, no. 3]

36

Some more bugs (6)

- CyberSitter censors "menu */ #define"
  - **because of the string "nu...de“**

    [Internet Risks Forum NewsGroup (RISKS), vol. 19, issue 56]

- London‘s Docklands Light Railway – train stopped in the middle of nowhere due to future station location programmed in software

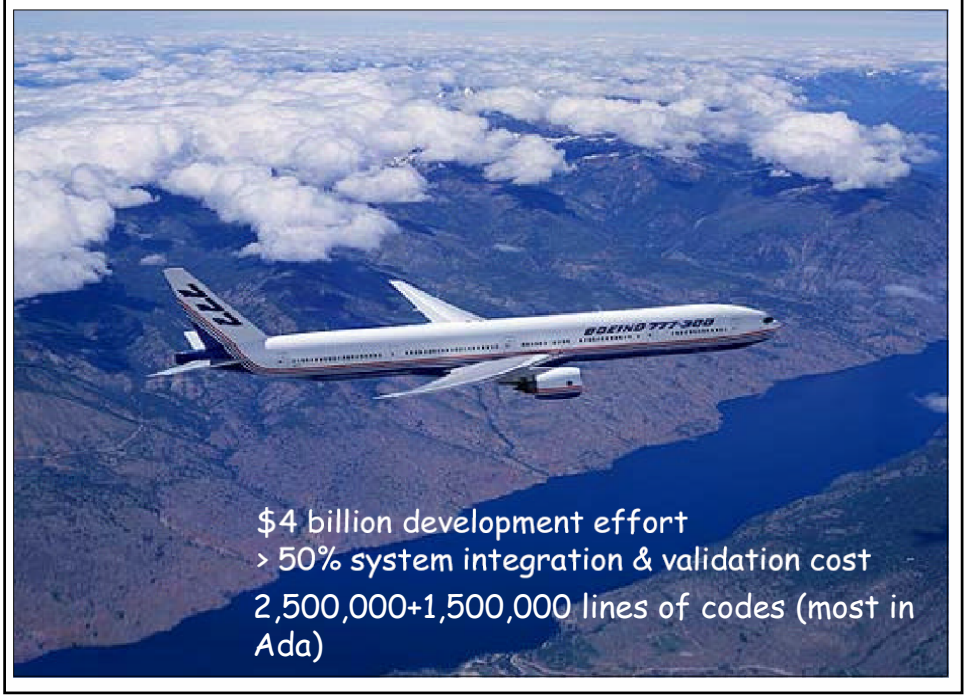
37

Some more bugs (7)

CNN.com

- Russia: Software bug made Soyuz stray.
  - **STAR CITY, Russia(AP) – A computer software error likely sent a Russian spacecraft into a rare ballistic descent that subjected the three men on board to check-crushing gravity loads that made it hard to breathe, space experts said Tuesday.**
- Korean Air crashed in Guam and killed 228 people.
  - **A poorly programmed ground-based altitude warning system**
- Faulty software in anti-lock brakes forced the recall of 39,000 trucks and tractors and 6,000 school buses in 2000.
- Mars Polar lander, $165M, 1999.
  - **Software shut the engines off 100 feet above the surface.**
- **US$59.5 billions loss in economy, 0.6%GDP, April 27, 2003**

38


$4 billion development effort
> 50% system integration & validation cost
2,500,000+1,500,000 lines of codes (most in Ada)

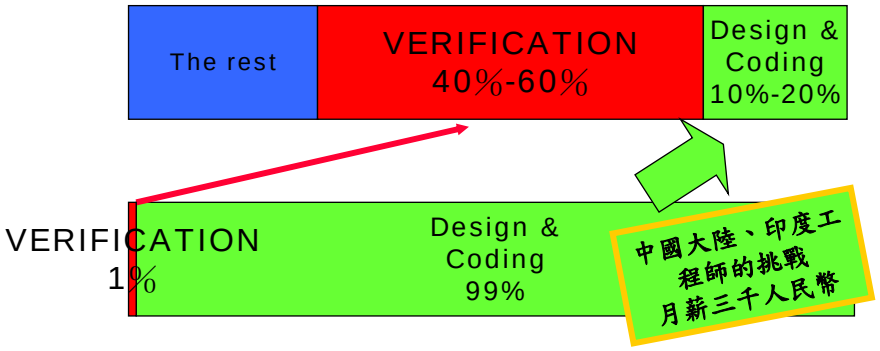

400 horses
100 microprocessors
M NH 2519

複雜系統的軟體錯誤

- 特殊的事件順序下，激醒蟄伏中軟體錯誤。
- 傳統的**test**、**simulation**不能測出所有的事件順序。

41

大型計畫預算分配



台灣人才投資分配

42

自動化驗證 = FV + simulation + testing

歐美工業界維持競爭力的法門

- Intel、Motorola、Ericsso… …
- Cadence、Synop… Graphics、…

國內工業界…

- …好產品，盡快賣到…
- 缺少自動化驗證技術，開發…雜系統。

台灣未來產業的優勢在哪裡？

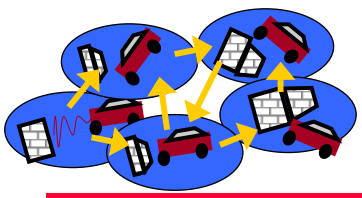
中國大陸、印度工程師的挑戰
月薪三千人民幣

43

Three techniques in verification

- **Testing (原型車撞實體牆)**
  - Expensive
  - Low coverage
  - Late in development cycles

- **Simulation (虛擬車撞虛擬牆)**
  - Economic
  - Low coverage
  - Don't know what you haven't seen.

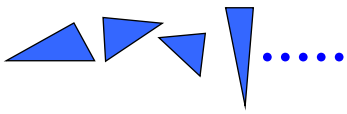


- **Formal Verification(虛擬車證明安全)**
  - Expensive
  - Functional completeness
    - ◆100% coverage
  - Automated!
    - ◆With algorithms and proofs.

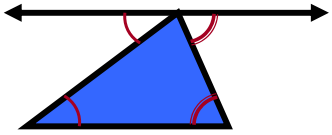
44

三角形內角和是180度？

- **Testing** (把看得到Δ的都測一測)
  - Expensive
  - Low coverage
  - Late in development cycles



- **Simulation** (產生一系列Δ，加以測量)
  - Economic
  - Low coverage
  - Don't know what you haven't seen.



- **Formal Verification**(因為平行線.....)
  - Expensive
  - Functional completeness
    - ◆100% coverage
  - Automated!
    - ◆With algorithms and proofs.

45

複雜系統的軟體錯誤（續）

- 但電腦自動驗證的技術，在面對龐大的複雜度時，還是有不能盡力之時。
- 目前可求助於下列三者的整合：
  - 良好的系統設計規範
  - 模擬、測試，以檢驗明顯的錯誤
  - 正規驗證（自動與手做），以證明系統設計的可靠性。

46

Promise of Formal Verification (FV) ?

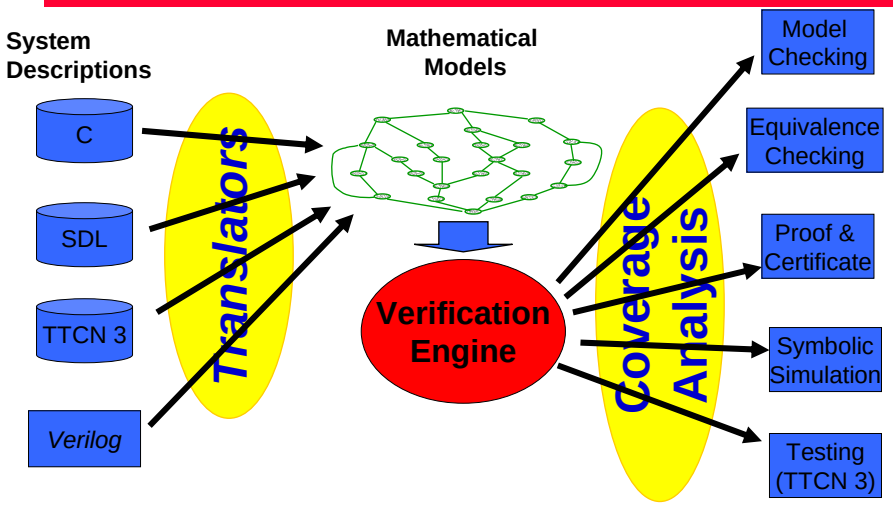
Use mathematics to prove the correctness of system designs!

Advantage:

- **Functional completeness**
  - Mechanical & exhaustive exploration of all cases
- **Automated verification**
  - Cut down verification cost
  - Relieve us of mechanical verification tasks

47

Verification Infrastructure



48

重要勢力分佈（美國）

- 模式檢驗
  - AT&T、CMU、UC-Berkeley、Stanford、North Carolina、Cornell、Intel、Cadence-Berkeley
- 一階邏輯
  - UT-Austin、SRI、MIT、MITRE、XEROX-PARC

49

重要勢力分佈（歐洲）

- * 處理代數 **(process algebra)**
  - + Oxford、Cambridge、Edinburgh、Uppsala
- * 正規方法**(formal methods)**
  - + Oxford、IFAD、IBM UK、CRI、Formal Systems Ltd、Praxis、CWI、VERILOG

50

重要研討會（專屬）

- **CAV (Computer-Aided Verification)**
- **FME (Formal Method Europe)**
- **AMAST (Algebraic Methodo. & Sw Tec.)**
- **TACAS (Tools & Algorithms for CAS)**
- **Hybrid**
- **SAS (Static Analysis Symposium)**
- **FORTE(Formal Description Technique)**

51

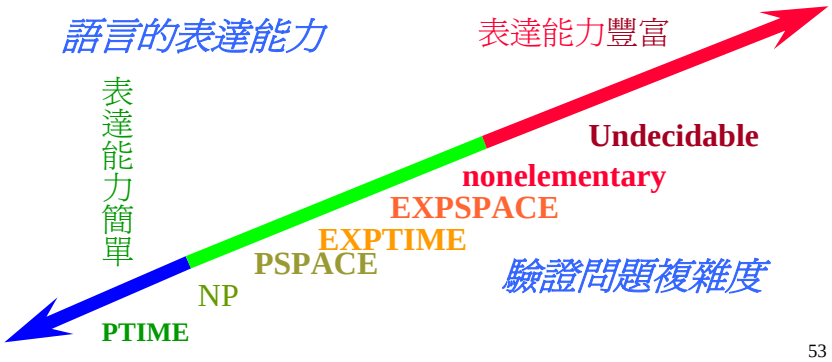
重要研討會（含相關 session）

*** FOCS**
*** LICS**
*** POPL**
*** STACS**
*** RTAS**
*** CONCUR**
*** SAS**
*** FTRTFT**
*** STOC**
*** PODC**
*** MFCS**
*** RTSS**
*** ICALP**
*** FORTE**
*** CADE**
*** RTCSA**

52

今日之理論：

- 指定表達能力 vs 驗證複雜度 的平衡點

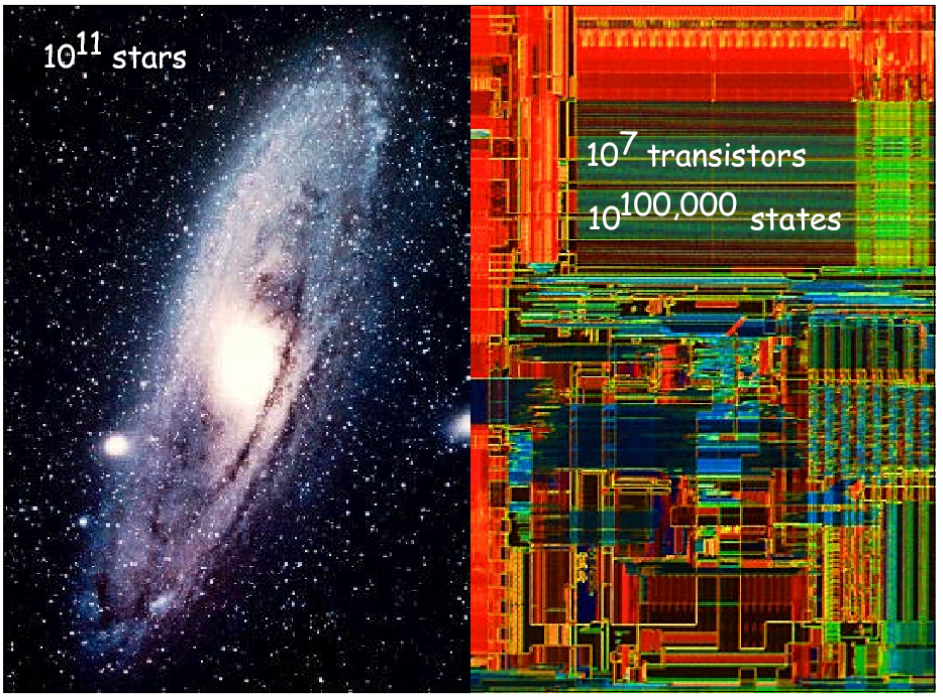


53

驗證複雜度的來源 ：

- 系統狀態的可能 — 天文數字
  - 每一個變數的值
  - 程式執行的位址
  - 各通訊通道的內容
- 無限制的變數值範圍
- 非規則行爲 — 不能用有限自動機描述的行爲
  - 堆疊(stack)、排序(queue)
  - 多項式運算、加減乘除
  - 演繹法 （induction)

54



今日之理論

- **指定表達能力 vs 驗證複雜度的平衡點**
- **非規則系統→證明檢驗 (proof checking)**
  - 由工程師遵循經驗法則進行良好設計
  - 由工程師導引自動驗證工具（十八般武藝）
- **規則行爲系統 ★★★**
  - 所需時間與記憶容量仍是驚人
  - 高效能自動驗證演算法
    - 控制計算時間、記憶容量

56

流派介紹

- 正規行爲描述
- 自動驗證技術

57

流派介紹：*正規行爲描述*

- 時態邏輯（**temporal logics**）
- 自動機理論（**automaton theory**）
- 處理代數（**process algebra**）
- 一階邏輯（**first-order logics**）
- **Petri-Net**
- 正規方法（**formal methods**）
- 圖形語言：**statechart**、**modechart**
- 傳統的正規程式語意 **(formal semantics)**

58

流派介紹：*正規行爲描述*

- **時態邏輯（temporal logics）**
  - 無時鐘的時態邏輯
  - 真時時態邏輯
- **自動機理論（automaton theory）**
  - 無時鐘的discrete 自動機
  - 有時鐘的dense-time 自動機
  - 混合式自動機（hybrid automata）

59

正規行爲描述：時態邏輯

- 由**modal logics** 的一支，發展出來。
  - 討論possible world的動態結構
  - 兩個運算元：□、◇
    - □：所有的possible world，命題皆成立
    - ◇：存在一個possible world，命題成立
  - modal logics 的模型爲Kripke structure(directed graph)
- **在時態邏輯中，**
  - □：自今而後，命題永遠成立
  - ◇：將來有一刻，命題成立

60

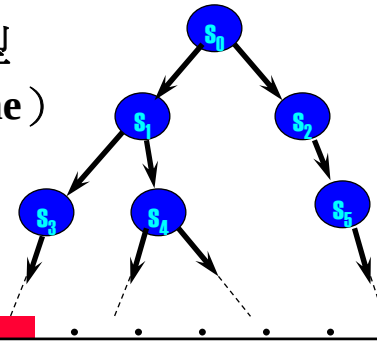
正規行為描述：時態邏輯

- 線性結構（**linear-time**）的模型（**model**）

- 樹狀結構的模型（**branching-time**）

∃ ： 存在一個path

∀ ： 對所有path



61

正規行為描述：時態邏輯

線性結構（**linear-time**）

與**PLTL**（**propositional Linear Temporal Logic**）

– 遲早有一刻，process P 會被執行。

◇**p** ：*liveness property*

★Amir Pnueli, 1996 Turing Award Winner

62

正規行為描述：時態邏輯

線性結構（**linear-time**）

與**PLTL**（**propositional Linear Temporal Logic**）

– Process P 和 Q 永遠不會同時在執行狀態

$\Box \neg (p \wedge q)$ *: safety property*

63

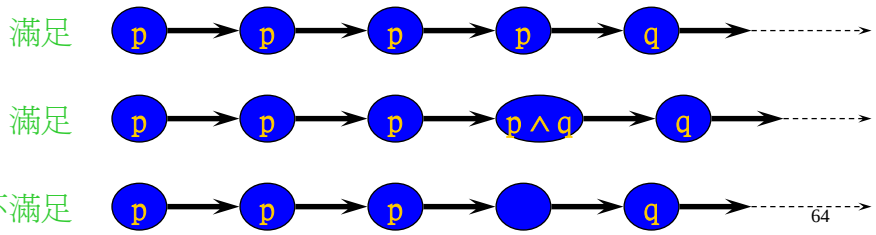
正規行為描述：時態邏輯

線性結構（**linear-time**）

與**PLTL**（**propositional Linear Temporal Logic**）

p一直被滿足，直到q被滿足。(p until q)

$p \mathcal{U} q$



64

正規驗證：線性時態邏輯

- 滿足（**satisfaction**）的關係
  - 一個線性結構滿足一個線性時態邏輯公式
- 一個線性時態邏輯公式定義一個模型集合。
- 一個邏輯公式是unsatisfiable如果他的模型集合是空集合。
- 測驗$P \wedge \neg S$的unsatisfiability
  - P 是系統行爲描述
  - S 是specification
  - PLTL的satisfiability問題是PSPACE-complete

65

正規行爲描述：時態邏輯

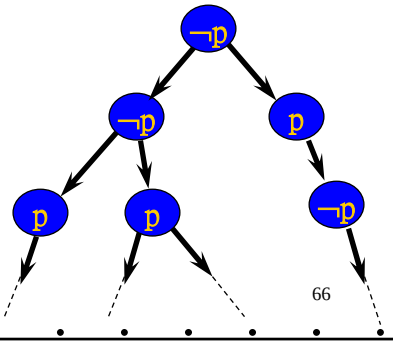
樹狀結構（branching-time）與

CTL（Computation Tree Logic）

- 對所有可能的計算，p 遲早會實現。

$\forall\Diamond p$

Inevitibility



66

正規行為描述：時態邏輯

樹狀結構（branching-time）

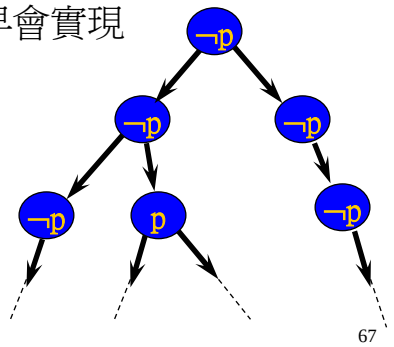
與CTL（Computation Tree Logic）

– 存在一個路徑，p遲早會實現

$\exists \Diamond p$

$\exists \neg p \mathcal{U} p$

Reachability



67

正規驗證：樹狀時態邏輯

- 滿足（**satisfaction**）的關係
  - 一個樹狀結構滿足一個樹狀時態邏輯公式
- 一個樹狀時態邏輯公式定義一個模型集合。
- 一個邏輯公式是unsatisfiable如果他的模型集合是空集合。
- 測驗$P \wedge \neg S$的unsatisfiability
  - **CTL 的satisfiability問題是deterministic EXPTIME-complete**

68

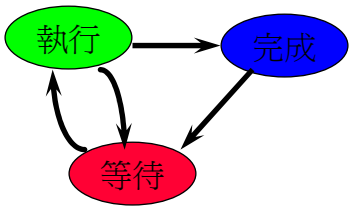
模型檢驗（model-checking）

- 給定一個模型 *M*，和一個時態邏輯公式 *P*，請問 *M* 滿足*P* 嗎？
- 通常*M*以有限自動機表達。
- 在**CTL**的情況下，模型檢驗可在**PTIME**內解決。

69

模型檢驗（model-checking）

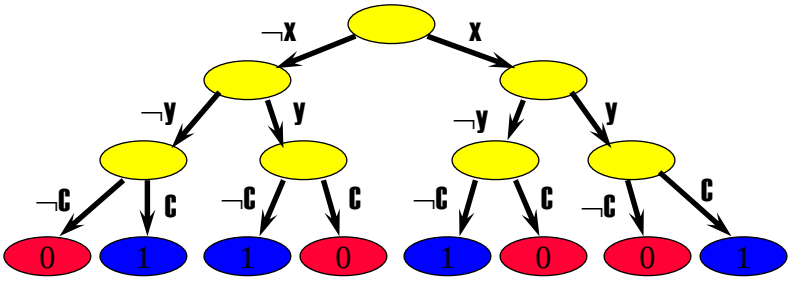
- 狀態空間分析
  - 狀態空間表示法：*有向圖*
    - 節點：系統狀態
    - 箭頭：狀態轉換
- 規則系統行爲



- 有限但極大(天文數字)之狀態空間

70

一個全加器的其邏輯公式

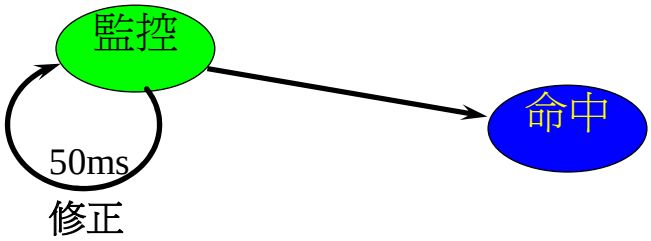


$$S = x \oplus y \oplus C$$

71

規則系統行爲: real-time automata

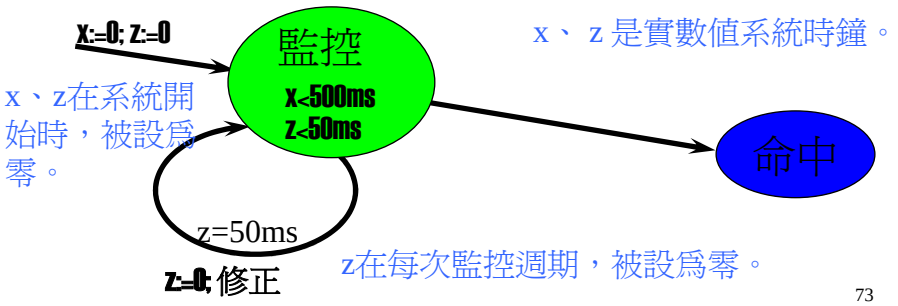
- 即時（**real-time)** 系統行爲
  - 每50ms監測敵蹤並修正飛彈方向，直到命中目標爲止



72

規則系統行爲: timed automata

- 即時（**real-time) 系統行爲**
  - 在500ms的限期內，每50ms監測敵蹤並修正飛彈方向，直到命中目標爲止



x、z在系統開始時，被設爲零。

x、 z 是實數值系統時鐘。

z在每次監控週期，被設爲零。

73

Timed automata模型檢驗

TCTL （Timed CTL）

∀□（監控→ ∀◇$_{<500}$命中）

∀□（監控→ x.∀◇（x＜500 ∧ 命中））

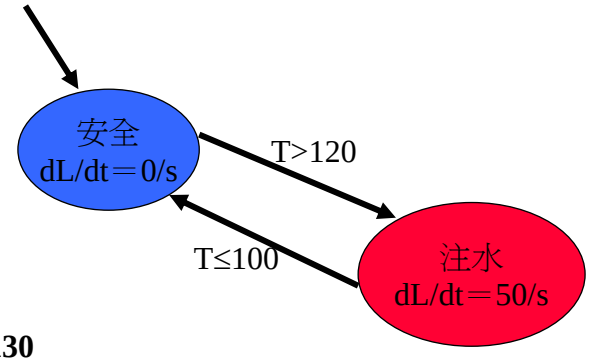
∀□x.（監控→ ∀◇（x＜500 ∧ 命中））

74

混和自動機：Hybrid automata

水量：**L**

溫度：**T**



模型檢驗：

∀□ **T＜130**

75

正規行為描述：*Process Algebra*

- **CSP,** Communicating Sequential Processes
  - C.A.R. Hoare, Turing Award winner
  - *Communicating Sequential Processes*, Prentice-Hall, 1985
- **CCS,** Calculus of Communicating Systems
  - Robin Milner, Turing Award winner
  - *Communication and Cuncurrency*, Prentice-Hall, 1989
  - strong equivalence, observational equivalence, observational congruence

76

正規行爲描述 : Process Algebra

自動販賣機

VM = (*in5p*→ *choc*→*VM* | *in2p*→ *toffee*→*VM*)

– action 集合： α VM = {in5p, choc, in2p, toffee}

- 模型（model）：traces
  - *in5p choc in5p*
  - *in5p choc in2p toffee in2p toffee*
- **process : the behavior pattern of a object**
  - 語法上，是rule的集合
  - 語意上，是traces的集合

77

正規行爲描述 : Process Algebra

traces 的運算

- prefix : *x*→*P* (*x then P*)
  - a guarded command
- recursion : *P* = (*x*→*P*)

  or *P* = μX. (*x*→*X*)（*least fixed-point*）
- choice : P = (P_1 | P_2)

78

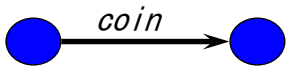
Process Algebra ：traces的運算

- 串接 ：**(catenation)**
  〈coin,choc〉^〈coin,toffee〉=〈coin,choc,coin,toffee〉
- 限制 ：**(restriction)**
  〈coin,choc,coin,toffee〉↾{coin}=〈coin,coin〉
- **head & tail**
  〈coin,choc,coin,toffee〉$_0$ = coin
  〈coin,choc,coin,toffee〉’=〈choc,coin,toffee〉
- **ordering**
  $s \leqslant t = (\exists u.s^{\frown}u = t)$
- 長度（**length**）：#〈coin,choc,coin,toffee〉=4

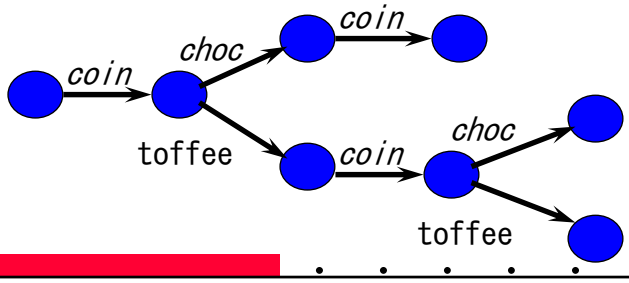
79

正規行爲描述：Process Algebra

- $(coin \rightarrow STOP)_{\alpha VMS}$



- VMCT = (coin → (choc →VMCT | toffee → VMCT))

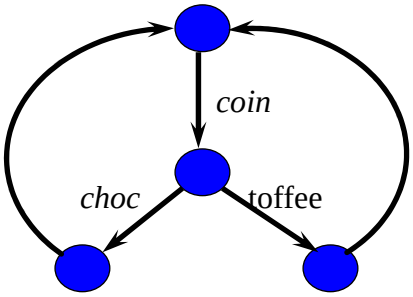


80

正規行爲描述 : Process Algebra

無限的行爲

VMCT = (coin → (choc →VMCT | toffee → VMCT))



81

Process Algebra ：processes的運算

concurrency:P||Q

- 在 α **P** $\cap$ α **Q**中的所有**action,** 必須同步
- ***(c→P)||(c→Q) = (c→(P||Q))***
- ***(c→P)||(d→Q) = STOP*** ； 如果**c≠d**

  deadlock

82

Process Algebra：processes的運算

- **nondeternimism : *P*Π *Q* (*P*、*Q*均等)**
  P* □ *Q (*P*優先)
- **communications:**
  (c!v→ P) || (c?v→ Q)
- **concatenation : *P;Q***
- **hiding : *P/C***
  – 將 P 中所有 C 中的action 都隱藏起來
  – C ：一個action 的集合

83

Process Algebra ：specification

- ***tr* : 當成是一個任意的trace變數**
- **specification, 一個對process 的要求**
- **在任何時候，投入VM 的錢幣數，大於已賣出巧克力數**
  NOLOSS =(#(tr ↾{choc}) ⇔ #(tr ↾{coin}))
- **在掉出巧克力前，不會再吃錢**
  FAIR =((tr ↓ coin) ⇔ (tr ↓ choc) + 1)
  ↓:selection

84

Process Algebra ：satisfaction

- ***P* sat *S***
  - P 是一個process； S是一個trace specification
- **Verification techniques**
  - Proof-checking
    - laws of processes and traces
  - Model-checking（模型檢驗）
    - 有限狀態空間之搜尋

85

正規行為描述 **: *first-order logic***

- **atoms, functions, predicates,** ∨, ∧, ¬, ∃, ∀

  ∀ x(Man(x)→Mortal(x))

  凡人皆會死！
- 驗證問題(*satisfiability problem*)：無解！
  - Resolution Principle - J.A. Robinson
    - proof-checking : mechanical theorem proving
    - 利用電腦輔助，經由驗證人員的導引。

86

正規驗證方法 : first-order logic

兩個**satisfiability problem**有解的子集

- **Presburger Arithmetic,**N,+,-,∨, ∧,¬,∃,∀

  $\exists z \forall y\ (z<y \vee \exists x(x+z<y))$

  – elementary decision procedure
- **1st-order logic with** **p(x)** **and** ≤

  (單引數邏輯函數)

  $\forall y(p(y) \rightarrow \exists x\ (y \leq x \wedge\ q(x))$

  – nonelementary decision procedure

87

正規驗證方法 : first-order logic

Boyer & Moore 的 NqThm

– *A Computational Logic Handbook*
Academic Press, 1988
– quantifier-free, first-order, 與pure-Lisp類似
– 一個非常有名的theorem-proving environment

example : (pp. 190-197 of the book)

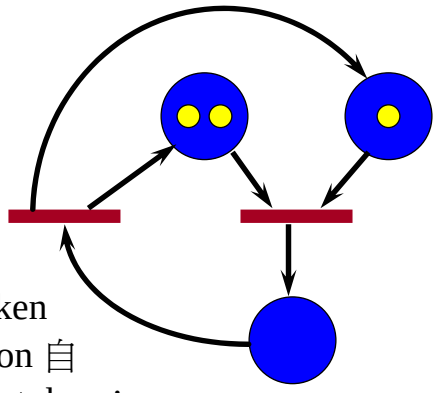
“一個list，反反爲正。”

- 錯誤示範：(PROVE-LEMMA REVERSE-REVERSE (REWRITE)
  (EQUAL (REVERSE (REVERSE X)) X))
- 正確示範：(PROVE-LEMMA REVERSE-REVERSE (REWRITE)
  (IMPLIES (PROPERP X)
  (EQUAL (REVERSE (REVERSE X)) X)))

88

正規行爲描述 : Petri-Net

- 1972 年，C.A. Petri
- places : 有限數個
- tokens :
- transitions
  - enabled : 如果每個來源place中，都有token
  - firing: enabled transition 自來源places中，各取一token；在目的places中，各放一個token。



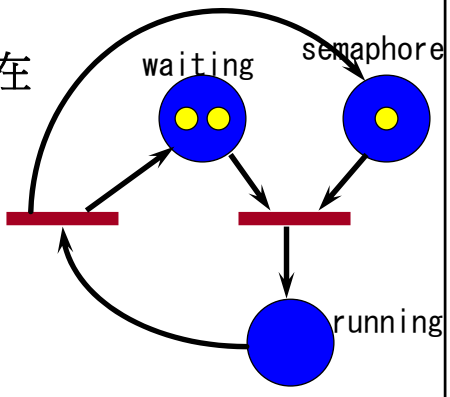
89

正規行爲描述 : Petri-Net

- 實質上同於Karp、Miller 的向量加法系統
- 不能檢驗事件的不存在
- 狀態（state）

marking : places → N

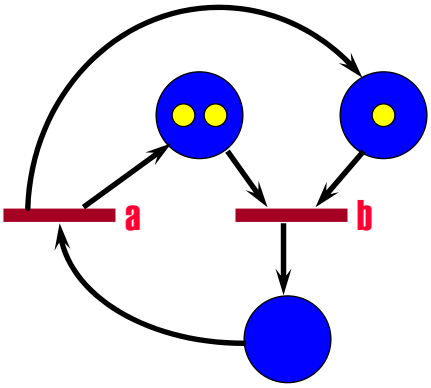
(2,1,0)



90

正規行為描述：Petri-Net

- computation :
  （state sequence）
  由enabled transitions
  串接起來的
  marking sequence
  – *interleaving semantics*



$(2,1,0) \xrightarrow{b} (1,0,1) \xrightarrow{a} (2,1,0) \xrightarrow{b} (1,0,1) \xrightarrow{a} \cdots\cdots$

91

Petri-Net ： 驗證問題

- **經由一個initial marking，可以走到無限個marking**
- **Reachability 問題：**
  給定兩個marking M1、M2,是否有一個computation ，可以從M1走到M2?
- **Reachability 問題有解，但不知複雜度。**
- **Coverability問題**
- **Boundedness 問題**

92

正規行爲描述：正規方法 (Formal Methods)

- 由工程界發展出來
- 最早是*VDM*，*Vienna Development Method*
- **Z notation**
- **RAISE,** Rigorous Approach to Industrial Software Engineering
- **Estelle** ： ESPRIT SEDOS的產品
  - 語意爲Petri-net；ISO的OSI電腦網路結構標準；
- **SDL** ： Specification & Description Language
  - CCITT Z1.00 標準；用作明確的protocol specification

93

Formal Methods：VDM

- 始於**1960**年代，**IBM research, Vienna**
- 由**C.B.Jones**、**D.Bjørner**在**1970-1982**年間，嚴謹的加以定義。
- 比較於其他的學院派研究，廣爲工業界採用，並有實效
- 目前有許多軟體系統，實驗性或商用性的
- **stepwise refinement**

94

Formal Methods：VDM

三個元件

- **direct definition**
- **implicit specification**
  - model-oriented specification
    - set-theoretical notation
      - 邏輯函數、集合、關係、函數、sequences
- **proof obligation**

  direct definition→implicit specification

95

Formal Methods：VDM

三個元件範例

- **direct definitioln**

  sumn : N→N

  sumn(n) if n=0 then 0 else n +sumn(n-1)
- **implicit specification**

  sumn (n:N) r:N

  post r=n(n+1)/2*
- **proof obligation**

  n ∈ N sumn(n) ∈ N ∧ post-sumn(n,sumn(n))

96

VDM proof example

兩個rules

- **rule 1 :** $\dfrac{}{sumn(0)=0}$
- **rule 2 :** $\dfrac{n \in N;\ n \neq 0;\ sumn(n-1)=k}{sumn(n)=n+k}$

97

VDM proof example

From $n \in N$

1. $sumn(0) = 0$	Rule 1
2. $sumn(0) \in N$	1, N
3. $0=0*(0+1)/2$	N
4. $sumn(0) = 0*(0+1)/2$	=-subs(3,1)
5. $post\text{-}sumn(0,sumn(0))$	post-sumn,4
6. $sumn(0) \in N \wedge post\text{-}sumn(0,sumn(0))$	$\wedge$ -I(2,5)
7. from $n \in N$, $sumn(n) \in N$, $post\text{-}sumn(n,sumn(n))$	

98

VDM proof example （續）

7.1	$n+1 \neq 0$	h7,***N***
7.2	$n+1 \in N$	h7,***N***
7.3	$sumn(n) = n*(n+1)/2$	*post-sumn*, ih7
7.4	$sumn(n+1) = n+1+n*(n+1)/2$	Rule 2(7.2,7.1,7.3)
7.5	$sumn(n+1) \in N$	7.4, ***N***
7.6	$sumn(n+1) = (n+1)*(n+2)/2$	7.4, ***N***
7.7	$post\text{-}sumn(n+1)=(n+1)*(n+2)/2$	*post-sumn*, 7.6
	infer $sumn(n+1) \in N \wedge post\text{-}sumn(n+1,sumn(n+1))$ $\wedge$ -*I*(7.5,7.7)	
infer $sumn(n) \in N \wedge post\text{-}sumn(n,sumn(n))$		***N***-*ind*(6,7)

99

正規行為描述：圖形語言

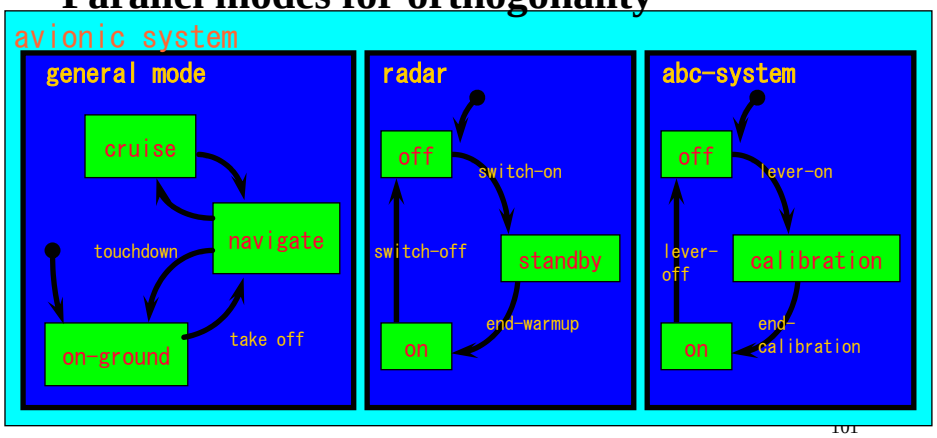
statechart

- **David Harel, 1986**
  - science of computer programming
- 由自動機理論擴展出來
- 可以描述並行計算、離散事件系統
- **nested modules**

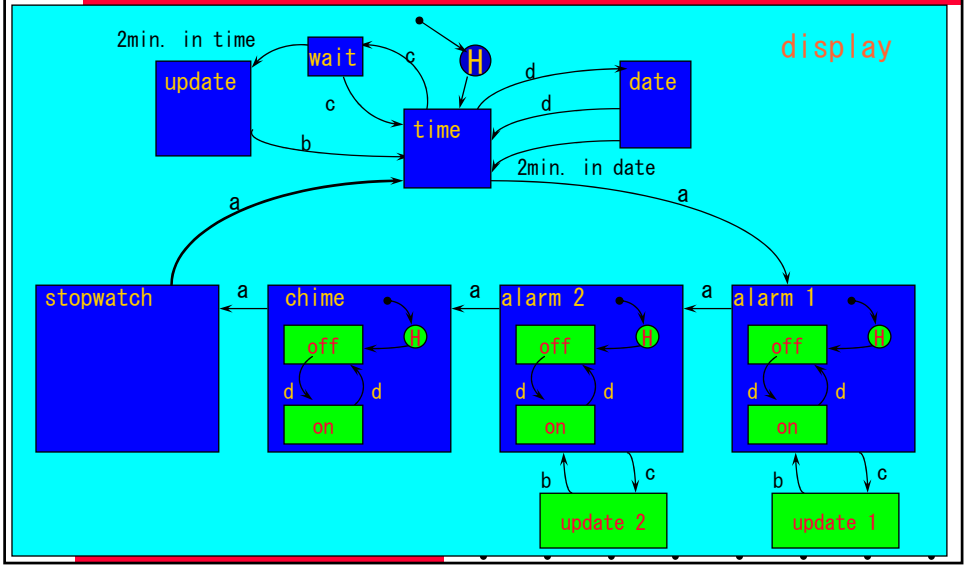
100

正規行爲描述：statechart

Parallel modes for orthogonality



正規行爲描述：statechart



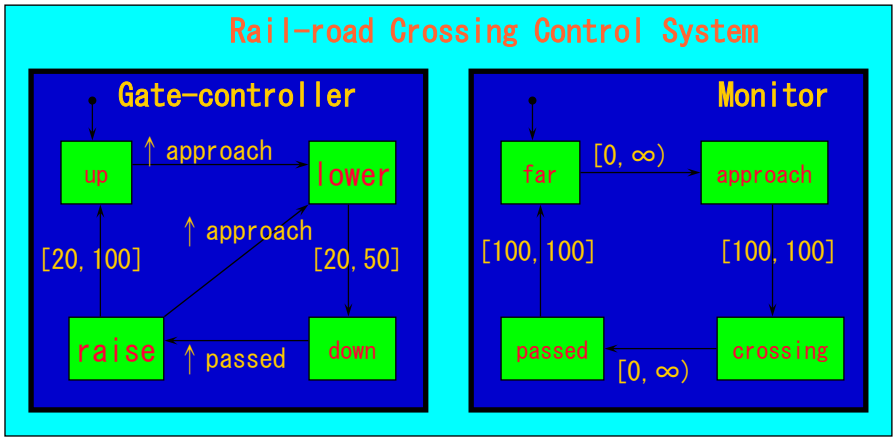
正規行爲描述：圖形語言

modechart

- **F. Jahanian, A.K. Mok, 1986**
  - IEEE Transactions on SE
- **statechart的real-time擴充**
- **semantics 由 RTL 定義。**
  - RTL（real-time logic）也是由Jahanian和Mok提出的。（IEEE TC）

103

正規行爲描述：modechart



104

流派介紹：*傳統的正規語意*

- operational semantics
  - 定義抽象電腦
- denotational semantics
  - 將程式視爲函數 (denotation)
- axiomatic semantics
  - pre-condition與post-conditions
  - 可以和正規驗證技術掛勾

105

正規驗證技術：Axiomatic Semantics

- **sequential programs**
  - E.W.Dijkstra, Turing Award

    A Discipline of Programming, Prentice-Hall, 1976
- **distributed programs**
  - Chandy、Misra的UNITY

    Parallel Program Design - A Foundation

    Addison-Wesley, 1988

106

正規驗證技術：Axiomatic Semantics

- **guarded-command language**

  $x,y := x+y,0 \ \text{if}\ y>0$
- **Fairness assumption，interleaving semantics**
- **program composition**
- **precondition、postcondition**

  $\{y>0\}\ x,y := x+y,0 \ \text{if}\ y>0\ \{y=0\}$
- **proof-checking**
  - 許多laws
  - safety properties、liveness properties

107

正規驗證技術：Axiomatic Semantics

Example : *Given N, M, compute x,y such that*

- $x \times N + y = M$
- $0 \Leftrightarrow y < N$

Program *division*
declare *x,y,z,k* : integer
initially *x,y,z,k = 0, M, N, 1*
assign $z,k := 2z, 2,k \quad \text{if } y \geq 2xz$ ~
$N, 1 \quad \text{if } y < 2xz$
$x,y := x+k, y-z \quad \text{if } y \geq 2xz$
end *{division}*

108

正規驗證技術：Axiomatic Semantics

properties

Given a program F

- *p unless q* iff for all *s* in **F**, $\{p \wedge \neg q\} s \{p \vee q\}$
- **stable** *p* iff *p unless false*
- **invariant** *p* iff **stable** $p \wedge$ (initial condition $\rightarrow p$)
- *p ensures q* iff (*p unless q*) $\wedge \exists s$ in **F**, $\{p \wedge \neg q\} s \{q\}$

109

正規驗證技術：Axiomatic Semantics

properties

Given a program **F**, $p \rightarrow q$ iff

$$\frac{p \text{ ensures } q}{p \rightarrow q} \qquad \frac{p \rightarrow q, \quad q \rightarrow r}{p \rightarrow r}$$

$$\frac{\forall \ w \text{ in } W(p(w)) \rightarrow q)}{(\exists \ w \text{ in } W(p(w))) \rightarrow q}$$

110

正規驗證技術：Axiomatic Semantics

Some theorems :

$$\frac{p \text{ unless } q,\ q \text{ unless } r}{p \vee q \text{ unless } r} \qquad \frac{p \text{ unless } q}{p \vee r \text{ unless } q \vee r}$$

$$\frac{p \rightarrow q,\ r \text{ unless } b}{p \wedge r \quad (q \wedge r) \vee b}$$

111

流派介紹：*自動驗證技術*

- 證明檢驗（**proof-checking**）
  - resolution-based prover
  - Floyd-Hoare Logics : axiomatic prover
  - proof paradigm : VDM
- **automatic verification**
  - 模型檢驗（model-checking）
  - language inclusion
  - bi-simulation

112

實際工業運用上之成就（I）

- 一階邏輯（**first-order logic)**
  - Boyer & Moore 的 NqThm
  - 證明檢驗
  - 1985 年，驗證四位元 FM 8501 CPU 至位元層次。
  - 高階語言編譯程式（compiler)
  - 參考書：*A Computational Logic Handbook*, Academic Press, 1988

113

實際工業運用上之成就（II）

- 正規方法（**formal method)**
  - occam
  - IMS T800 transputer
  - 研發時間估計節省十二個月
  - 1990 年英國女王獎！！！
    - （49 位獲獎）

114

實際工業運用上之成就（III）

- 正規方法（**formal method)**
  - Z notations 的正規描述方法
  - IMS CICS
  - 研發經費估計節省5百餘萬元
  - 1992 年英國女王獎！！！

115

實際工業運用上之成就（IV）

- 模式檢驗 （**model-checking)**
  - Burch, Clarke, McMillan, Dill, Hwang
  - 二元決定圖（BDD）符號處理
  - CMU 的 SMV 系統， 全自動
  - Intel 32位元ALU，8 個暫存器，兩層 pipeline
  - 狀態空間數目達 10^{120}
  - 在SUN 4 上 4 時 20 分。

116

實際工業運用上之成就（V）

- **Symbolic Trajectory Evaluation**
- **Hazelburst, Seger**
- **64 位元乘法器**
- **簡單的、或 Wallace tree**
- **硬體驗證（no memory）**
- **在Sparc 10/51 上 800 秒。**

117

實際工業運用上之成就（VI）

- 一階邏輯與混合式自動機
  - Bosscher, Polak, Vaandrager
  - 證明檢驗
  - Phillip 音響控制網路實體層
  - 考慮信號電壓的變化、晶體頻率的穩定性
  - 不考慮匯流排擠撞 (collison)、信號延誤
  - 證明1/17 頻率誤差可接受。
    - （Phillip 以 5% 爲設計標準）
  - 後經Henzinger 的 HyTech 自動驗證完成。

118

Workout 1: proof of a simple program

Initially x = 0;

- **consumer:**

for (; 1;) if (x > 0) x--;

- **Producer:**

for (; 1;) if (x< 1) x++;

- **Atomicity assumption**
- **Please prove that x is always no greater than 1.**

119

Another workout: Petrification

1. **請將92頁的兩個簡單程式，model成一個Petri-net（見63頁）。**
2. **請解釋下列名詞：**
   1. Safety property
   2. Liveness property
   3. Fairness assumption
   4. Interleaving semantics

120