Praktyczne metody weryfikacji

Sławomir Lasota

Uniwersytet Warszawski

semestr zimowy 06/07

I. Motywacja czyli po co?

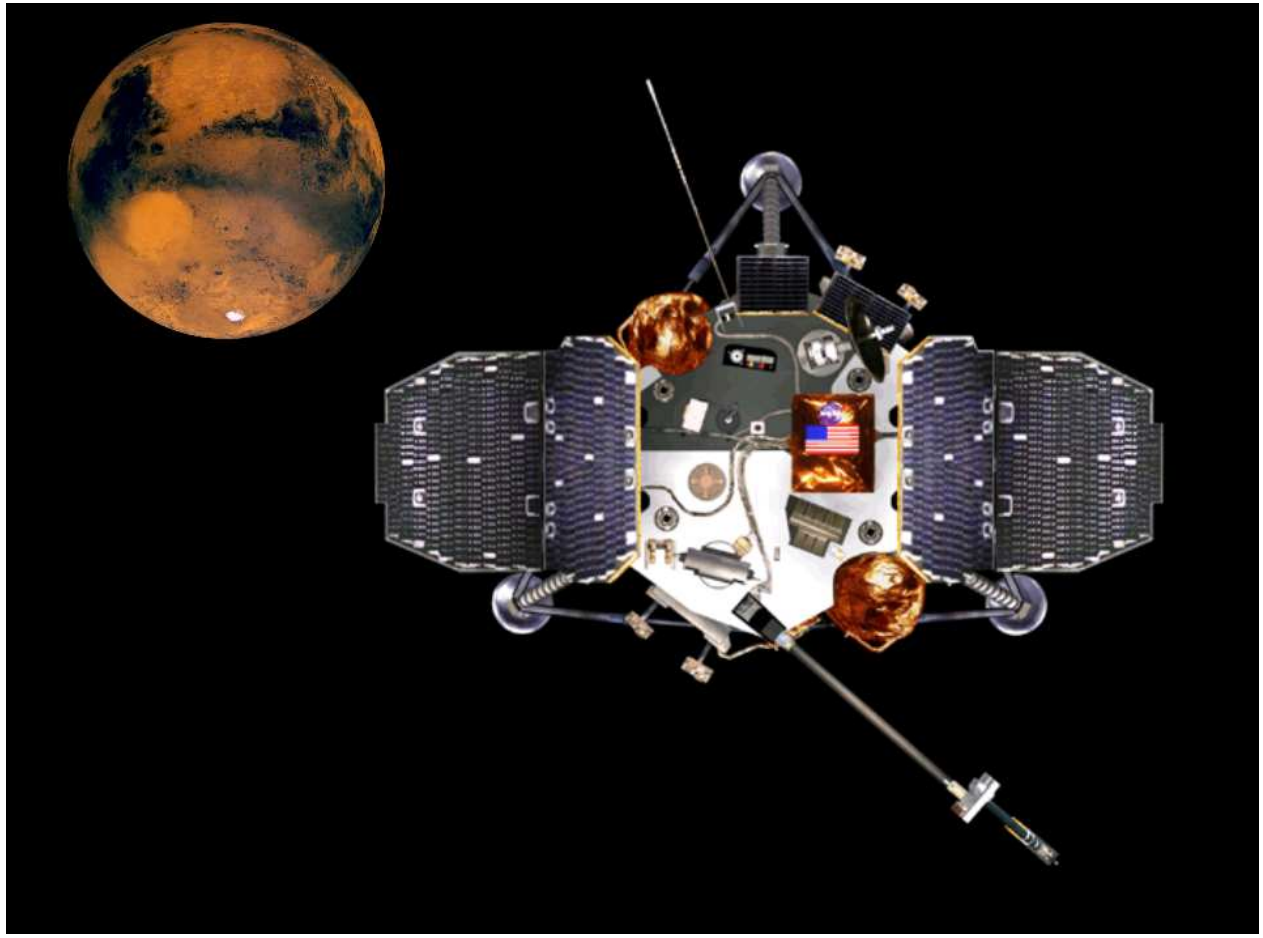
czerwiec 1996

– nieobsłużony wyjątek

– szacunkowy koszt:

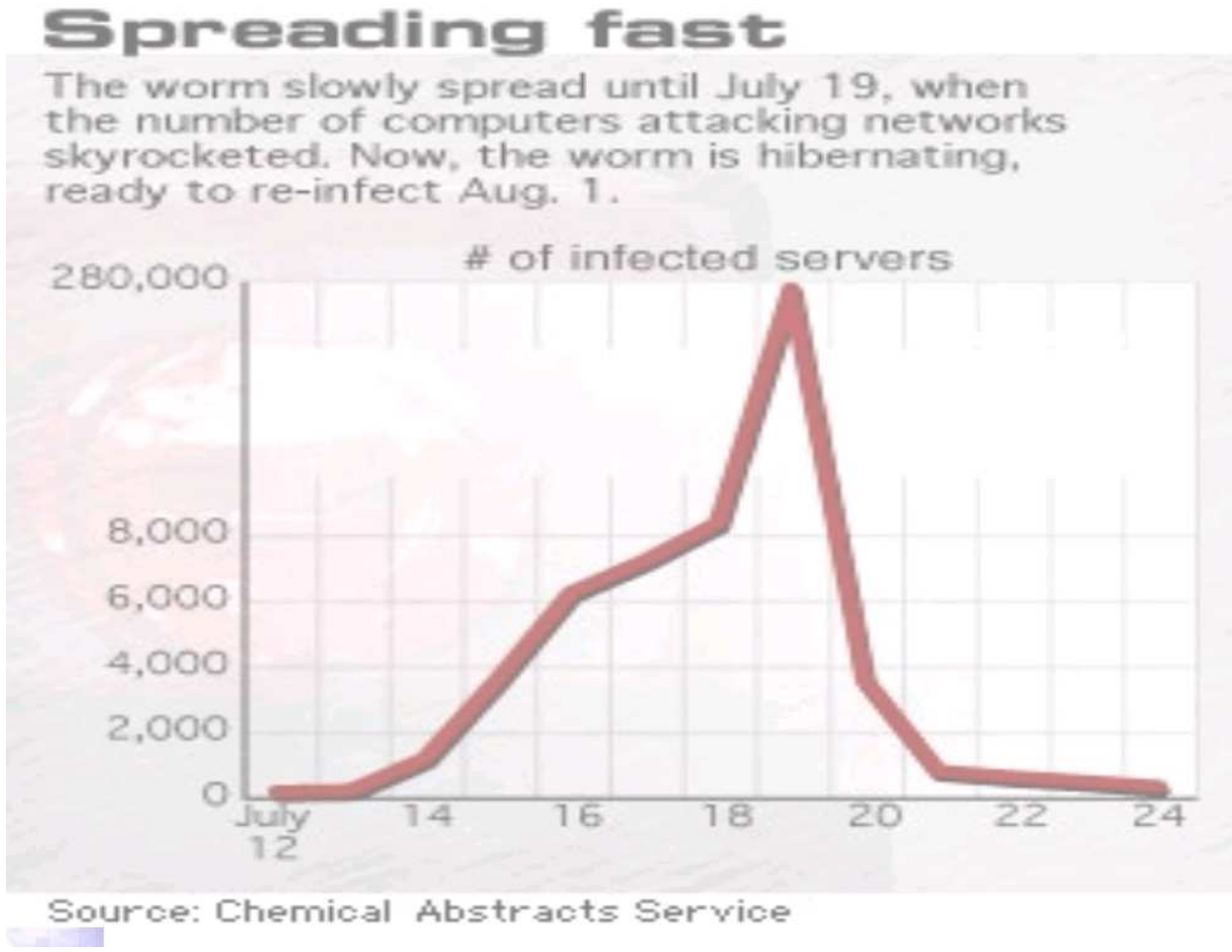
600 mln euro

grudzień 1999



– awaria z powodu niezainicjowanej zmiennej

lipiec 2001



– przepełnienie bufora

– szacunkowy koszt: 2.5 miliarda USD

Błędy

- kosztowne
- nieakceptowalne (ang. *safety-critical*)

Rozwiązanie:

formalna weryfikacja = dowód poprawności

II. Weryfikacja, ale jak?

Weryfikacja modelowa (ang. *model checking*)

– model M – możliwe zachowania

(abstrakcja rzeczywistego systemu)

– specyfi kacja własnośći ϕ – dopuszczalne zachowania

– test

$$M \models \phi$$

Przykładowe własności ϕ:

- bezpieczeństwo: wszystkie stany osiągalne spełniają dany wymóg
- żywotność: zawsze osiągniemy stan, w którym dany wymóg jest spełniony
- sprawiedliwość (ang. *fairness*): zasób zostanie przyznany, o ile będziemy *konsekwentnie* go żądać
- ...

Inne podejścia

- testowanie
- audyt kodu źródłowego
- dowodzenie poprawności (proof checker)
- analiza statyczna
- …

Weryfikacja interakcyjna

- problem sprowadza się do dowodzenia twierdzeń
- narzędzie wspomagające (ang. *proof checker*)
- wymaga pracy eksperta

Weryfikacja modelowa

- analiza dynamiczna vs. statyczna
- przeszukiwanie stanów osiągalnych
- algorytm weryfi kacji, narzędzie
- w pełni automatyczna

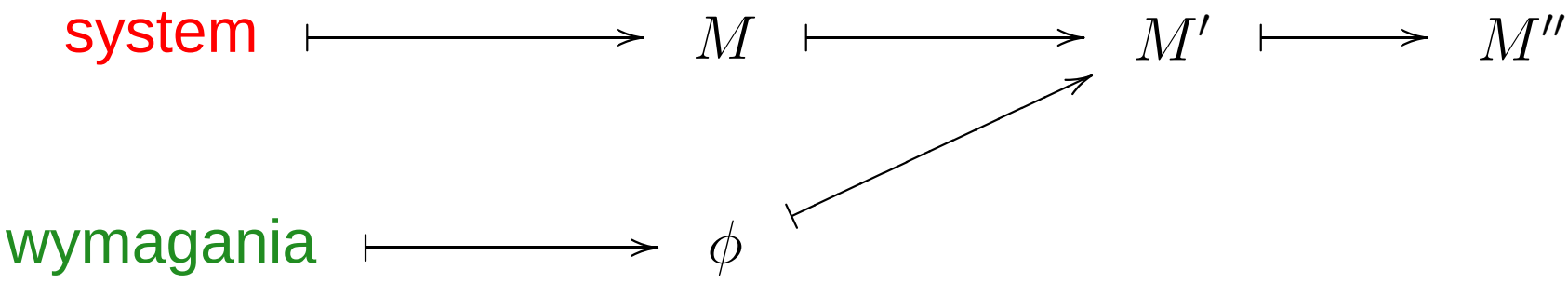
Weryfikacja modelowa – zalety

- w pełni automatyczna:
  - tworzymy model
  - formułujemy wymagania
  - weryfi kujemy
- kontrprzykład, gdy odpowiedź negatywna

Weryfikacja modelowa – ograniczenia

- złożoność obliczeniowa, eksplozja stanów
- nie weryfi kacja, tylko tropienie błędów
  - weryfi kujemy model a nie sam system
  - sprawdzamy tylko wymaganie ϕ, innych nie
  - błędy w narzędziach
- wykonanie abstrakcji wymaga pracy eksperta

Od rzeczywistości do modelu



$M'' \models \phi$?

Motto:

celem formalnej weryfi kacji nie jest tworzenie poprawnego oprogramowania, lecz dostarczenie metodologii, która pozwoliłaby zwiększyć niezawodność (zmiejszyć liczbę błędów).

Dziedziny zastosowań:

- sprzęt (SMV, NuSMV, Murphi)
- protokoły, oprogramowanie systemowe, sterowniki (Spin)
- oprogramowanie (BLAST, PathFinder, CBMC)
- systemy zależne od czasu (Uppaal, Kronos)
- systemy stochastyczne (PRISM)

Zagadnienia teoretyczne

- teoria języków i automatów
- logiki temporalne, rozstrzygalność i złożoność
- algorytmika
- matematyczne modele systemów współbieżnych
- teoria grafów

Zagadnienia praktyczne

- heurystyki dla wydajności
- zastosowanie do rzeczywistych, dużych systemów
- włączenie weryfi kacji do procesu projektowania sprzętu i oprogramowania

III. Jaki model?

Jaki model?

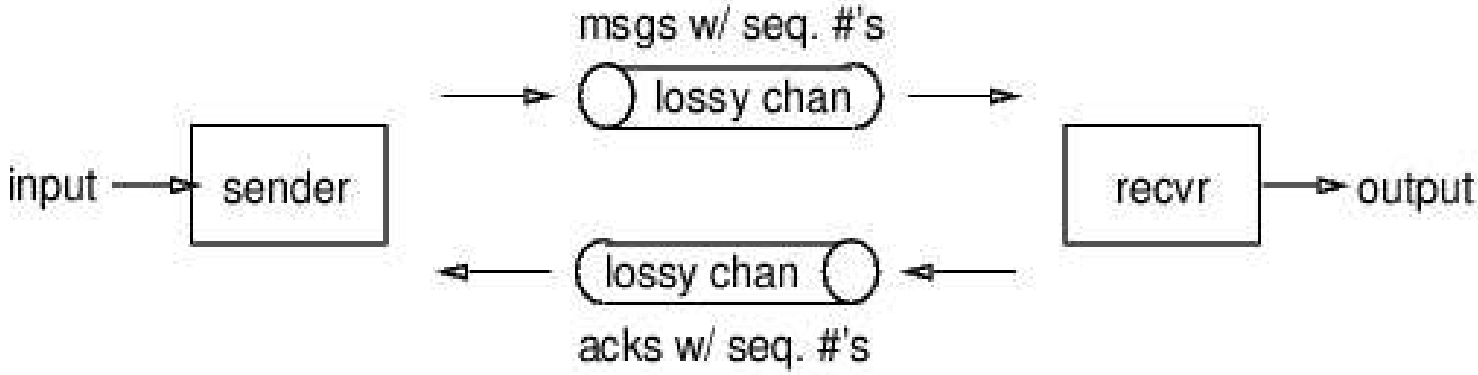
- funkcyjny: dane $\mapsto$ wynik
- reaktywny:
  - interakcja z otoczeniem
  - działanie może się nie kończyć
  - wymagania = własności temporalne
  - przykłady: system operacyjny, bankomat, serwer WWW, ...

Model = sterowanie + interakcja

- brak skomplikowanych danych i

  skomplikowanych obliczeń na danych
- kompozycjonalność
- dopuszczamy niedeterminizm
- ścisłość matematyczna

np. ABP


msgs w/ seq. #'s
lossy chan
input
sender
recvr
output
lossy chan
acks w/ seq. #'s

Własności modeli

- modele wysokiego poziomu (abstrakcyjne),

  niedospecyfi kowane (częściowe) $\rightarrow$ niedeterminizm

- interakcja pomiędzy składowymi $\rightarrow$ współbieżność
- prototypowanie/symulacja $\rightarrow$ model wykonywalny
- weryfi kacja formalna $\rightarrow$ semantyka matematyczna

Typowy model: FSM (stany i przejścia)

+ zmienne (np. liczniki)
+ komunikacja (kanały komunikacyjne)
+ zegary
+ …

Stan = punkt sterowania

+ wartości zmiennych
+ zawartość kanałów komunikacyjnych
+ …

Opis kompozycjonalny

- złożenie równoległe
- przemianowanie (kopie modułu)
- …

Zło˙zenie równoległe

- synchroniczne (hardware)
- asynchroniczne (software lub anynchr. hardware)

Inne modele – ró˙znice

- pojęcie stanu
- atomowy krok obliczeń
- interakcja
- równoważność semantyczna

Inne modele – przykłady

- różne rozszerzenia automatów
- algebra procesów: CSP, CCS, ACP, rachunek Π, ...
- języki synchronniczne: SCCS, Esterel, Lustre
- sieci Petriego
- pamięć dzielona: Promela (Spin)

Eksplozja stanów!

$$M = M_1 || \ldots || M_n$$

$$S \approx S_1 \times \ldots \times S_n$$

Metody walki z eksplozją

- podejście symboliczne
  - symboliczna weryfi kacja modelowa
  - ograniczona weryfi kacja modelowa

    (ang. *bounded model checking*)
- niezależność modułów
  - redukcje częściowo-porządkowe

IV. Historia

Dowodzenie poprawności programów

- [Dijkstra]
- [Floyd]
- [Hoare] podejście strukturalne

– Boyer–Moore prover: automatyczne dowodzenie poprawności prostych programów w LISPie
– [Owicki-Gries] rozszerzenie Hoare’a dla programów współbieżnych
– [Pnueli] logika temporalna dla systemów reaktywnych
– weryfi kacje protokołów, zjawisko eksplozji stanów

Model checking

- [Clarke, Emerson], [Queille, Sifakis]
- model explicite (skończony system przejść, automat)
- specyfi kacja = formuła logiki temporalnej

  (później też automat)
- weryfi kacja automatyczna
- kontrprzykład gdy odpowiedź negatywna
- zweryfi kowano małe układy cyfrowe, znaleziono błędy

Metody walki z eksplozją:

- metody abstrakcji
- symboliczna weryfi kacja modelowa
- redukcje częściowo-porządkowe

- systemy nieskończeniestanowe, np.
  - zależne od czasu
  - parametryczne
- narzędzia (SMV, Murphi, . . .)
- zastosowania przemysłowe
- zastosowanie SAT-solverów:

  ograniczona weryfi kacja modelowa

- narzędzia dla rozszerzonych modeli:
  - zależnych od czasu
  - stochastycznych
  - …
- zastosowania przemysłowe cd
- nowe dziedziny zastosowań, np. bioinformatyka
- weryfi kacja oprogramowania

  (ang. *software model checking*)

Plan

wykład		laboratorium	
LTL, translacja do ω-automatów	(3x)	SPIN	(3x)
CTL, OBDD, symboliczna w.m.	(3x)	NuSMV	(3x)
abstrakcje	(2x)	Blast	(1x)
redukcje częśc.-porz.	(1x)		
ograniczona w.m., redukcja do SAT	(1x)	NuSMV	
automaty czasowe	(2x)	Uppaal	(1x)
weryfi kacja param.	(1x)		
weryfi kacja stochast.	(1x)		