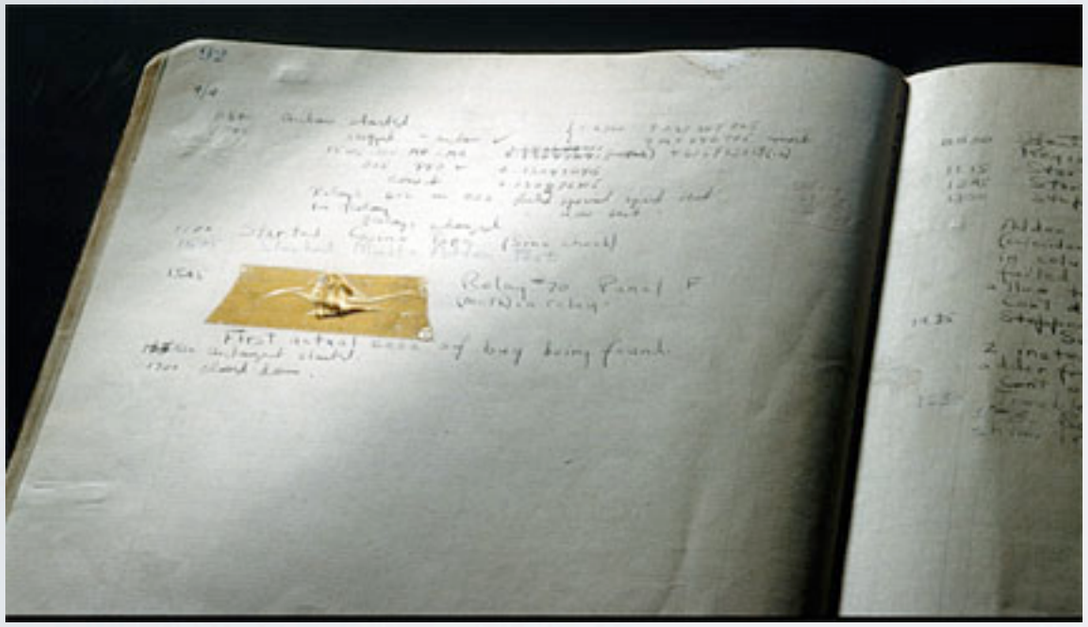
WERYFIKACJA WSPOMAGANA KOMPUTEROWO

Sławomir Lasota
Uniwersytet Warszawski
2009/1020

Teoria i praktyka wspomaganych komputerowo metod analizy formalnej oprogramowania i układów sprzętowych.

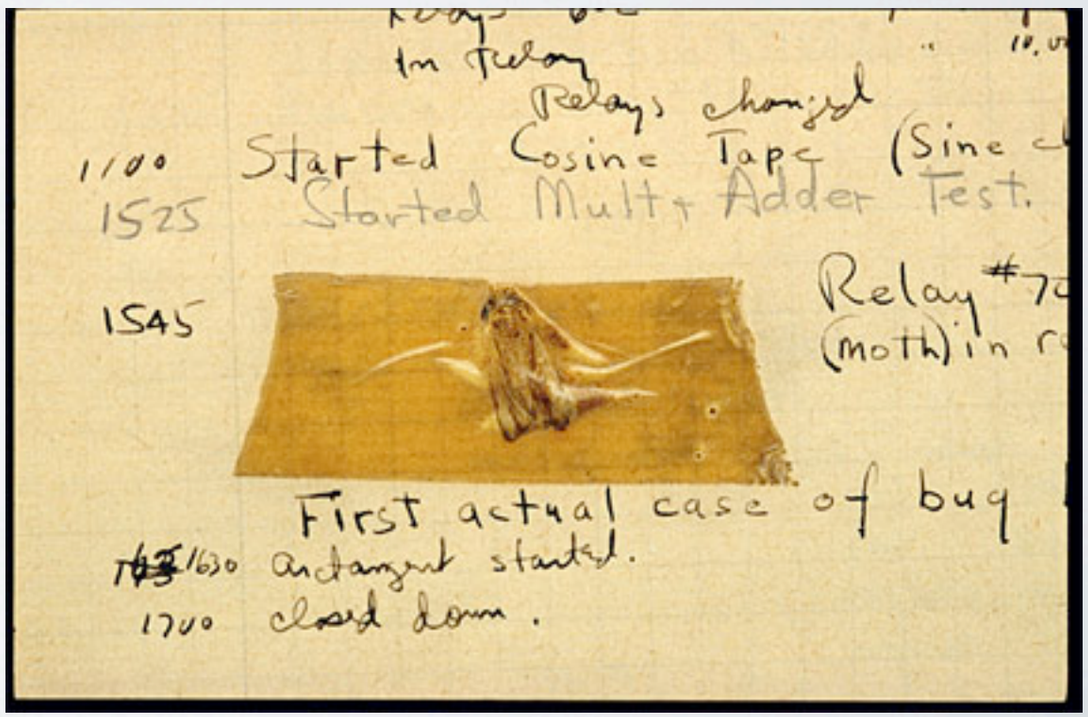
I. Motywacja

PLUSKWY :)

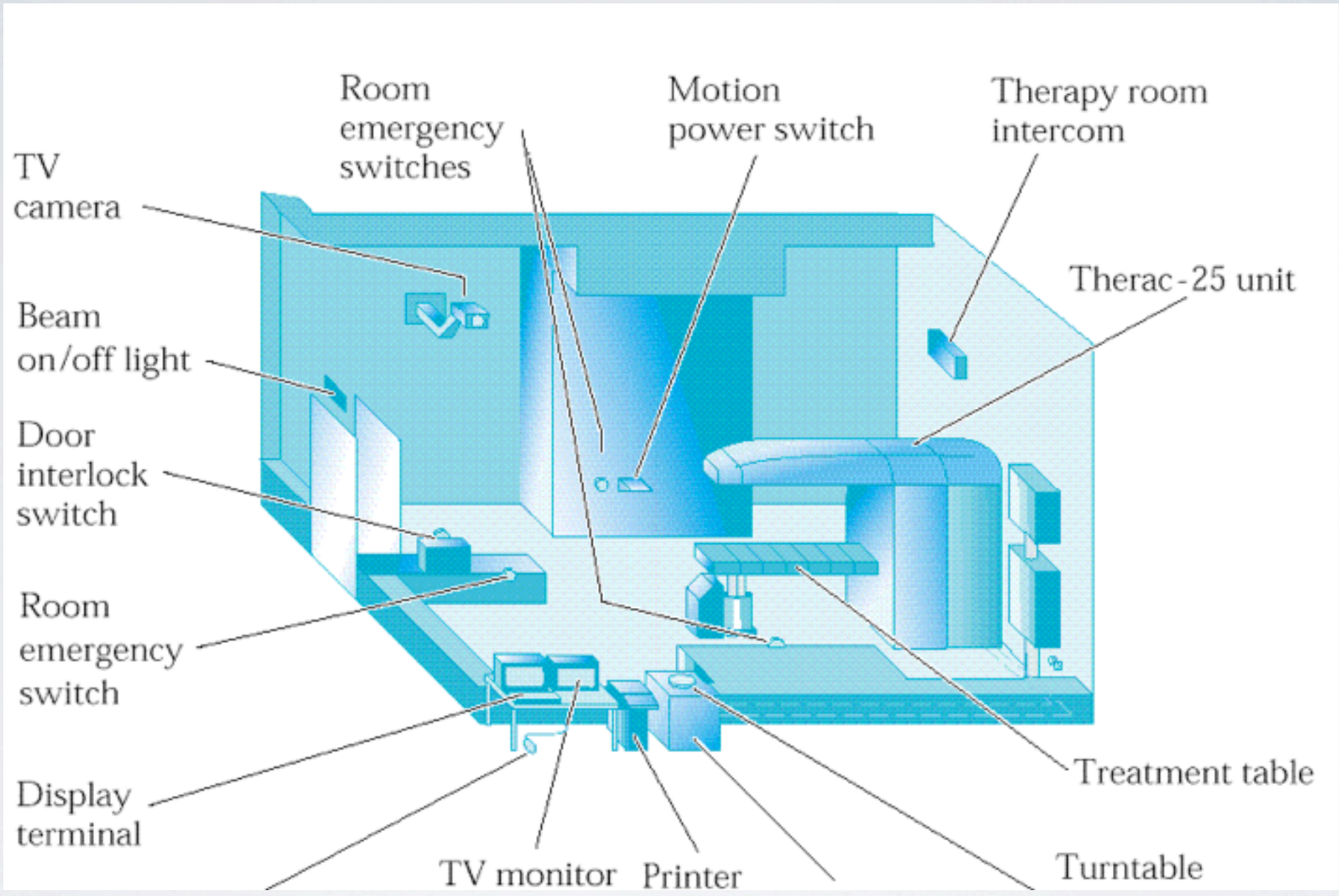


dokumentacja pracy nad komputerem Mark II

1947 Harward



THERAC-25



1985-87

- „wyścig”
- przynajmniej 6 ofiar

PENTIUM

- rzadki błąd w operacji dzielenia zmiennopozycyjnego
- Intel wymienił wszystkie wadliwe procesory

październik 1994

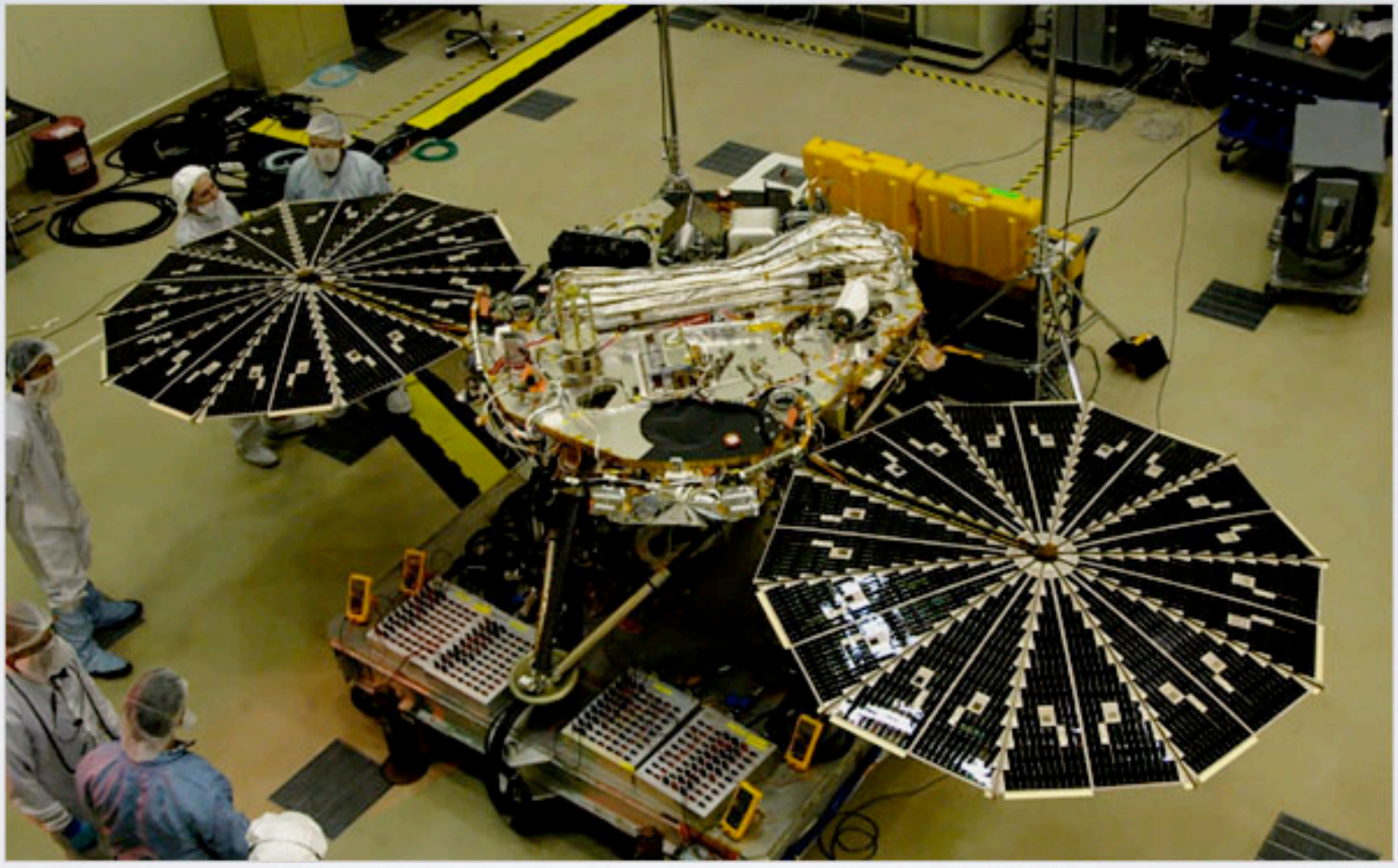
ARIANNE 5

czerwiec 1996

ARIANNE 5

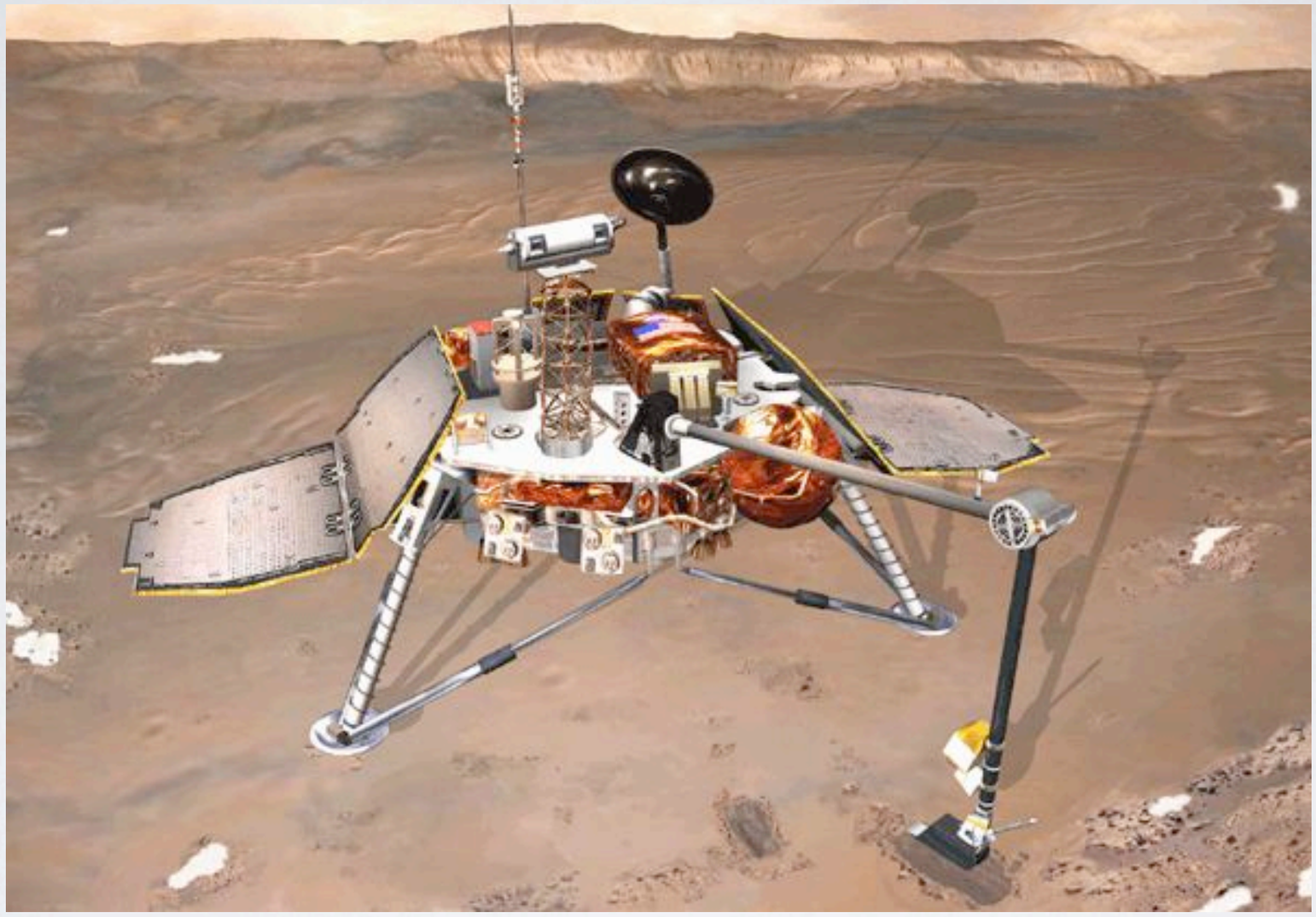
- nieobsłużony wyjątek
- szacunkowy koszt: 600 mln euro

MARS POLAR LANDER



styczeń 1999

MARS POLAR LANDER



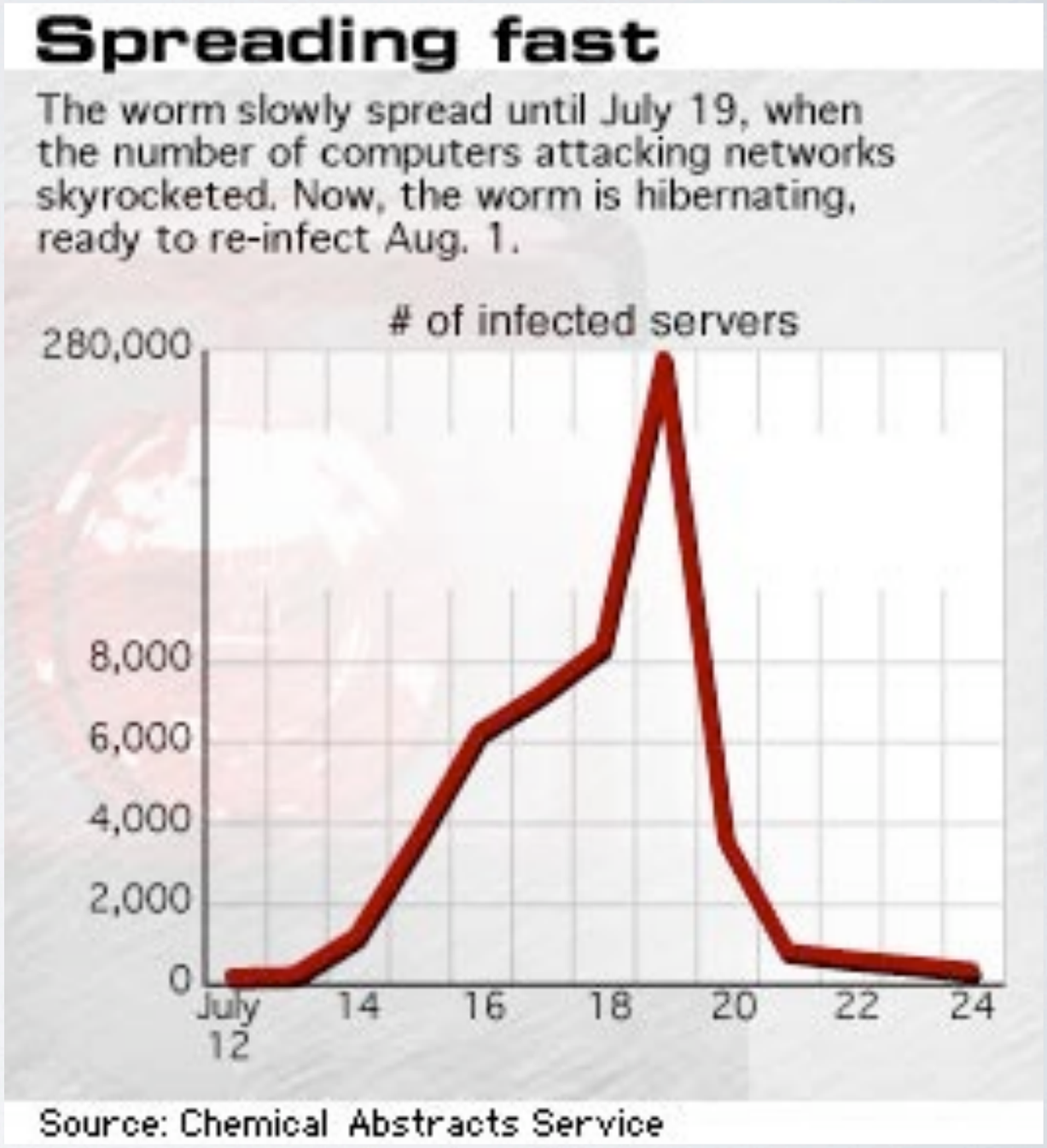
grudzień 1999

awaria z powodu
niezainicjowanej zmiennej

CODE RED

- usterka w Microsoft Internet Information Server (przepełnienie bufora)
- szacunkowy koszt: 2.5 miliarda dolarów

lipiec 2001



OSTATNIO

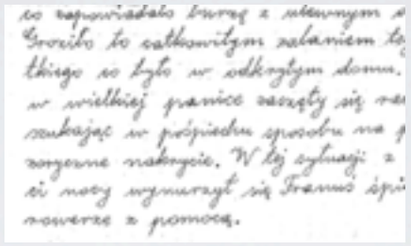
- ...
- styczeń 2010: niemieckie karty płatnicze
- październik 2009: utrata danych z telefonów T-Mobile
- grudzień 2008: awaria odtwarzacza Zune (31 grudnia roku przestępnego)
- październik 2008: awaria samolotu Airbus A330-303, 12 poważnie rannych
- 2007, Excel 2007: 77.1 * 850 = 100.000
- wrzesień 2006: przegrzewające się baterie Sony

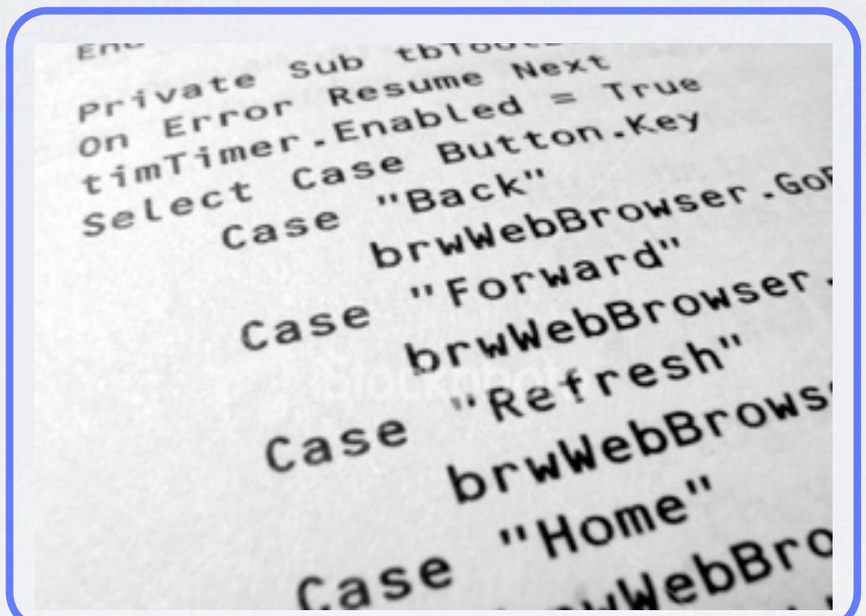
PODSUMOWANIE

- błędy są kosztowne
- ... i często nieakceptowalne (ang. safety critical)
- w zmiejszeniu ich liczby może pomóc weryfikacja formalna
- testowanie wykazuje obecność błędów, a nie ich brak - weryfikacja formalna pozwala wykazać brak błędów

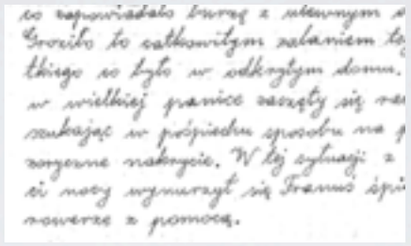
II. Weryfikacja formalna

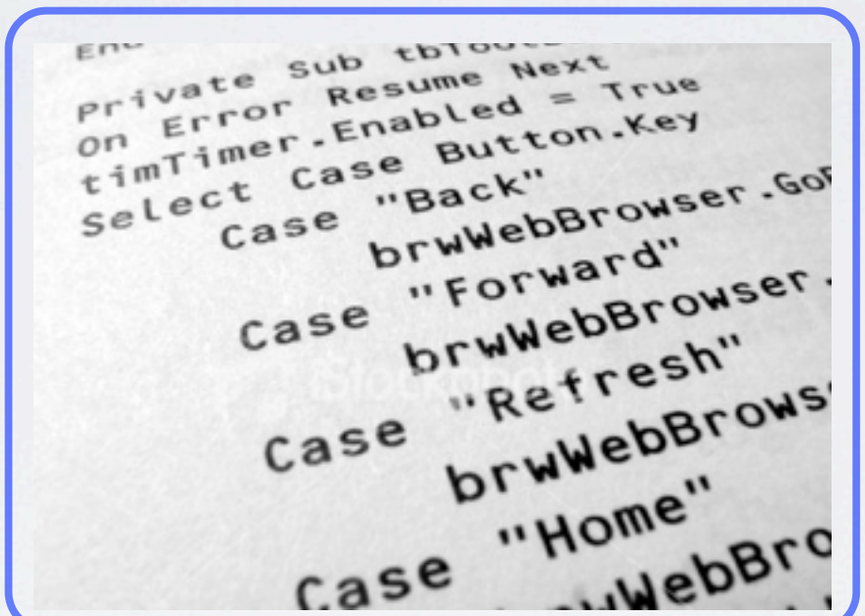
WERYFIKACJA A POSTERIORI





WERYFIKACJA A POSTERIORI



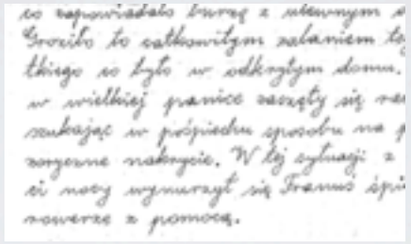


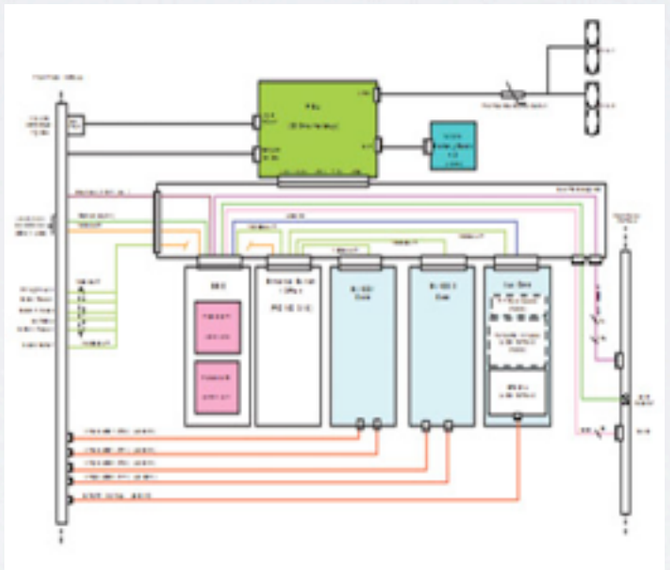
automatycznie!

✔

✘

WERYFIKACJA A POSTERIORI





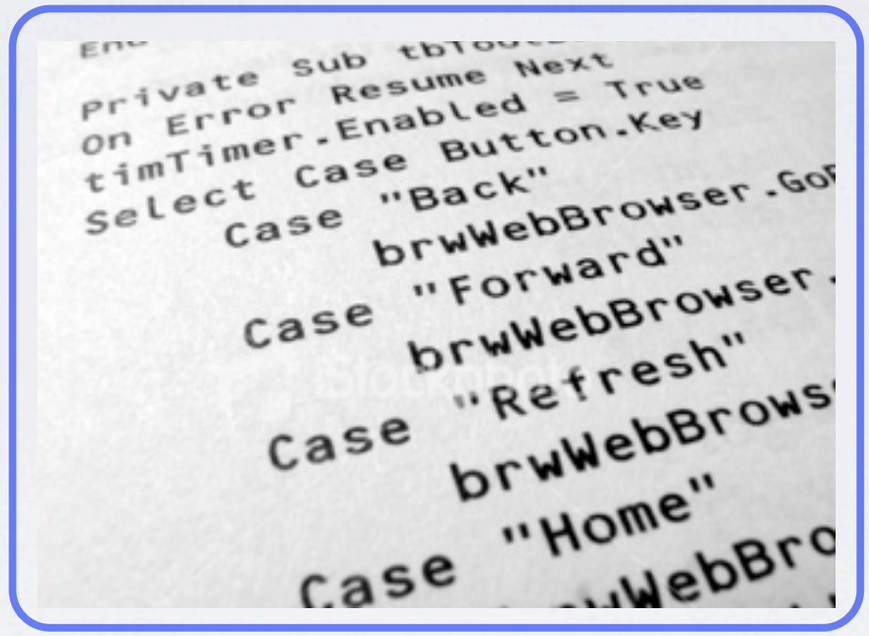
automatycznie!

✔

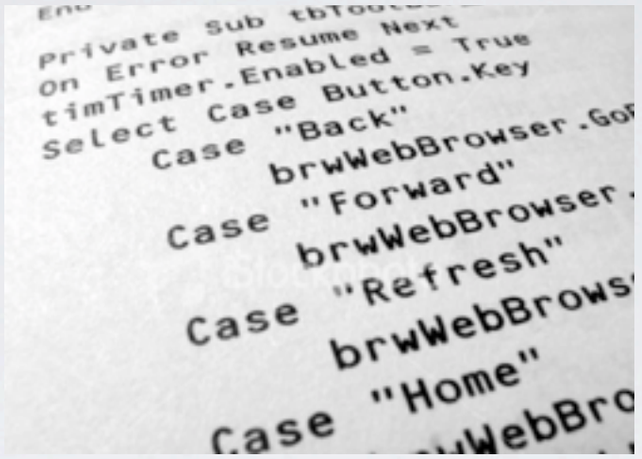
✘

OGRANICZENIE

każde nietrywialne pytanie jest nierozstrzygalne !



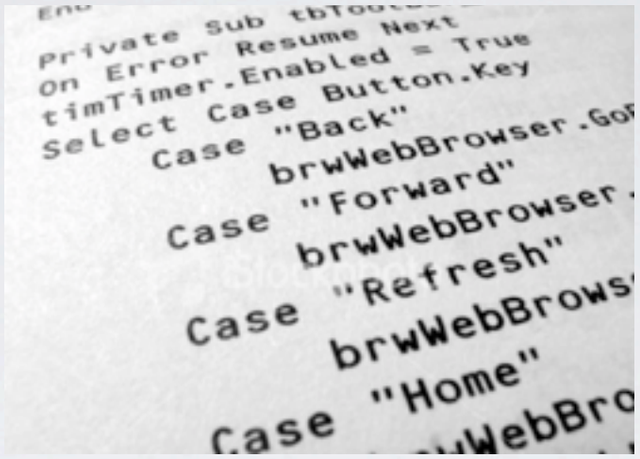
METODA 1: INTERAKCYJNA



✔

✘

METODA 1: INTERAKCYJNA

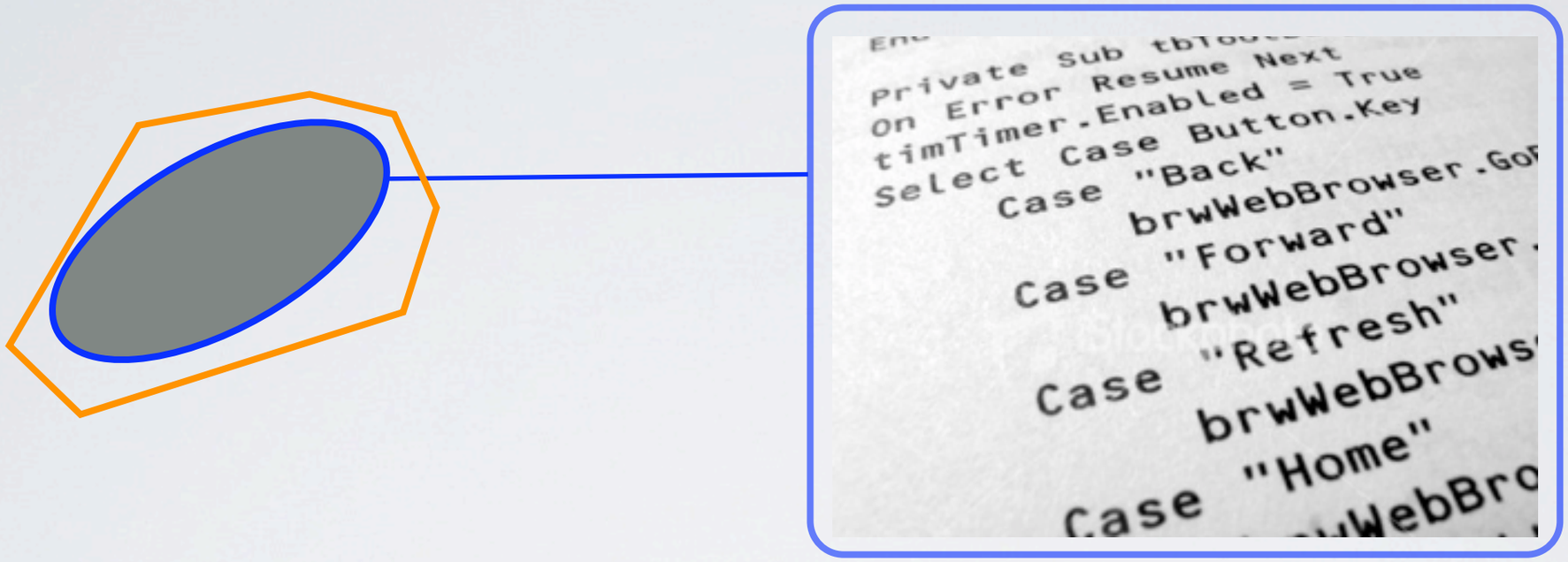


✔

✘

stosowana w dowodzeniu poprawności programów

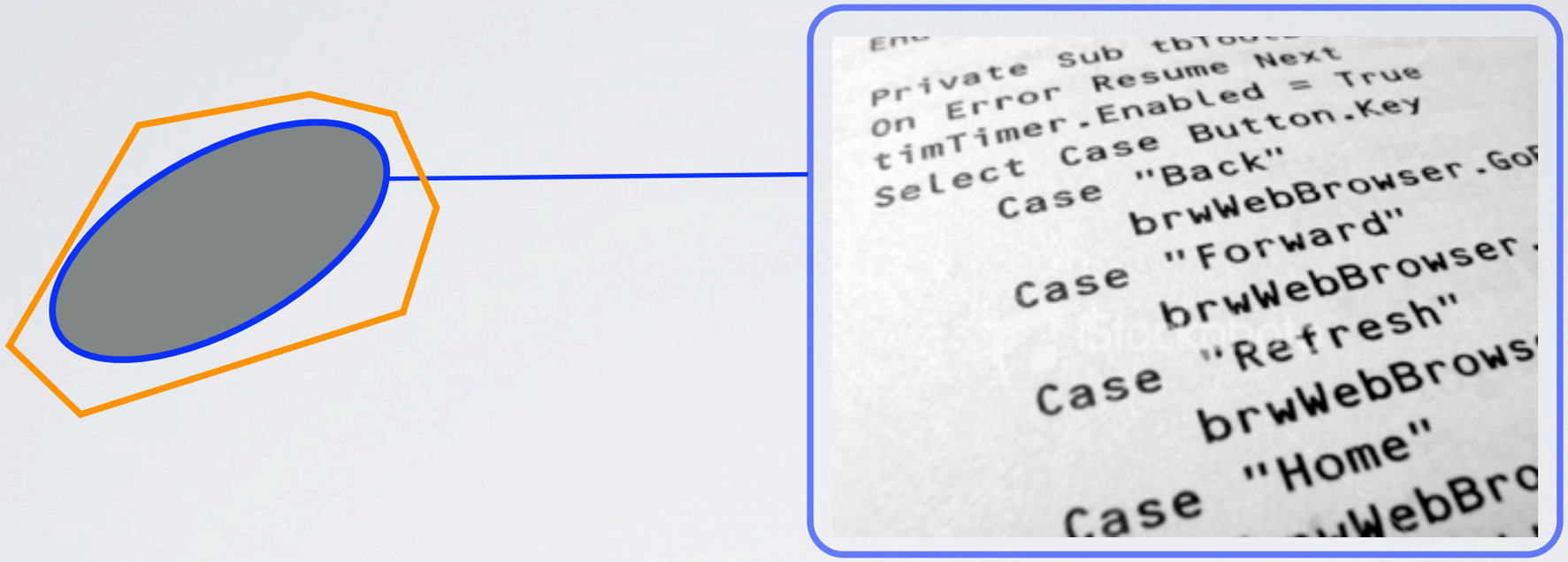
METODA 2: PRZYBLIŻENIE



na pewno ✔

chyba ✘

METODA 2: PRZYBLIŻENIE

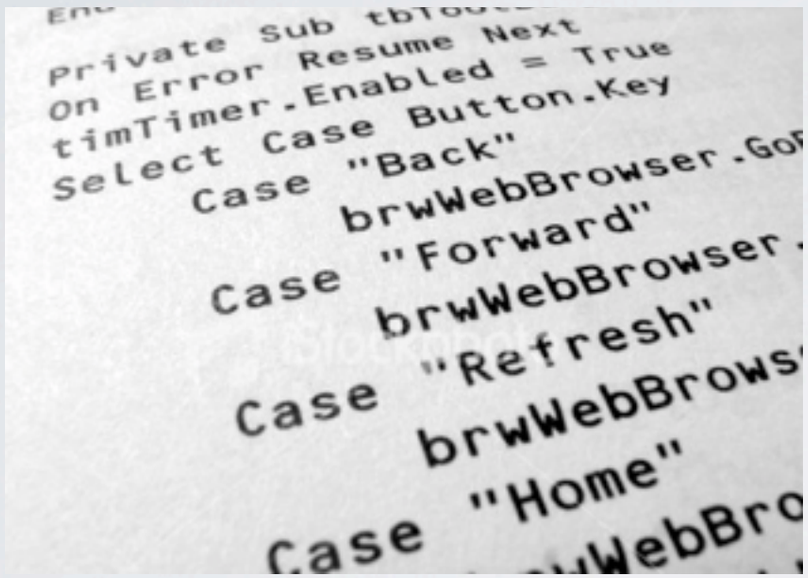


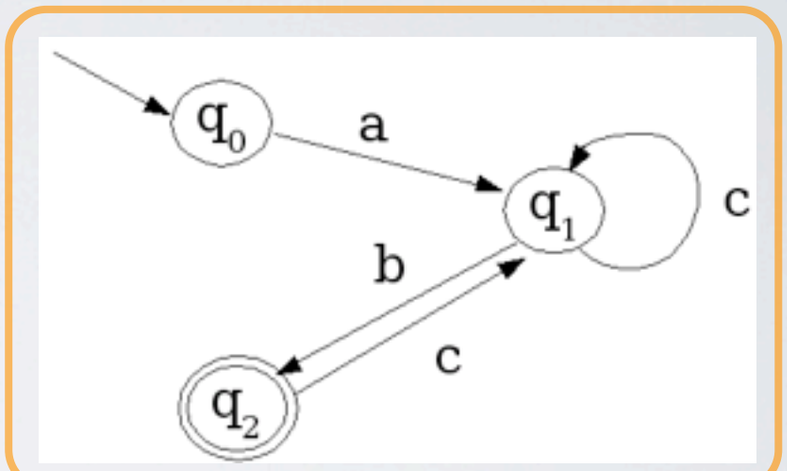
na pewno ✔

chyba ✘

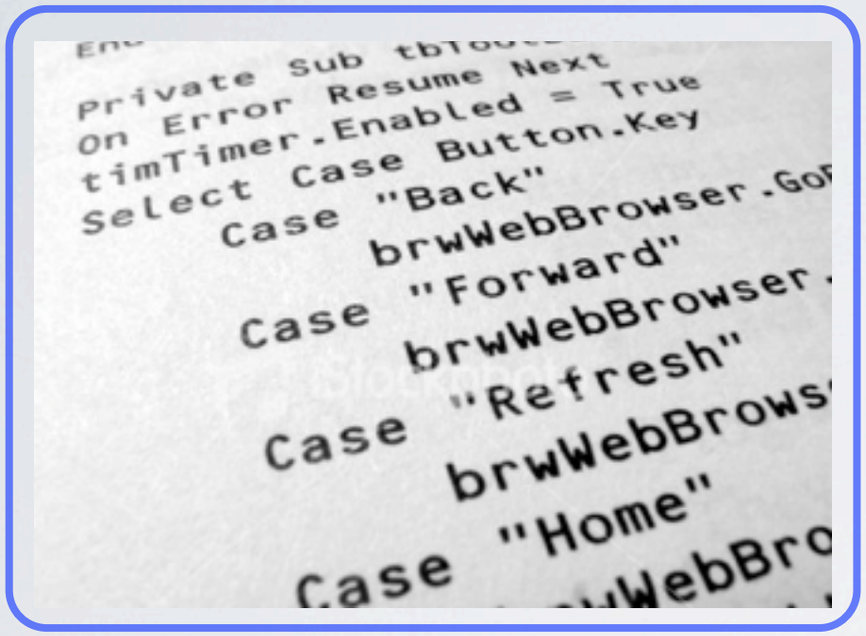
stosowana w statycznej analizie programów

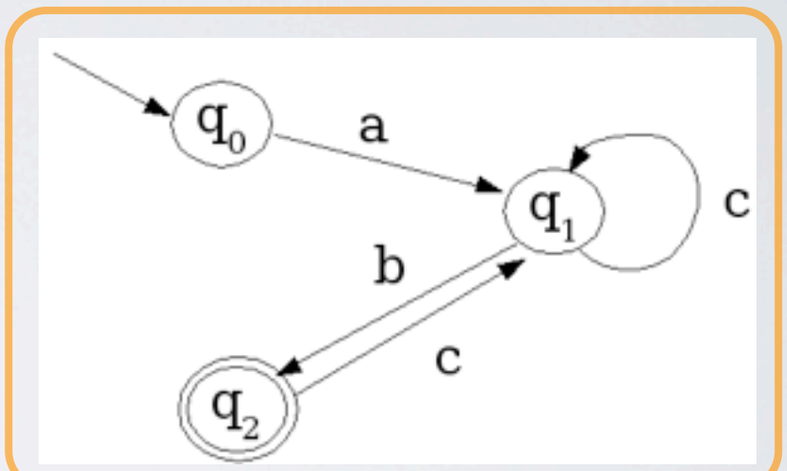
METODA 3: ABSTRAKCJA





METODA 3: ABSTRAKCJA





stosowana w weryfikacji modelowej

abstrakcja = przybliżenie

OGRANICZENIA

- Metoda 1 (interakcyjna): duży nakład pracy
- Metoda 2 (przybliżenie): fałszywe alarmy
- Metoda 3 (abstrakcja): weryfikujemy nie system, tylko model

MOTTO

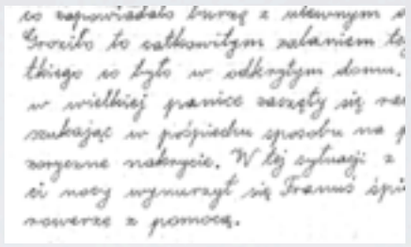
Celem formalnej weryfikacji nie jest tworzenie poprawnych systemów komputerowych ...

MOTTO

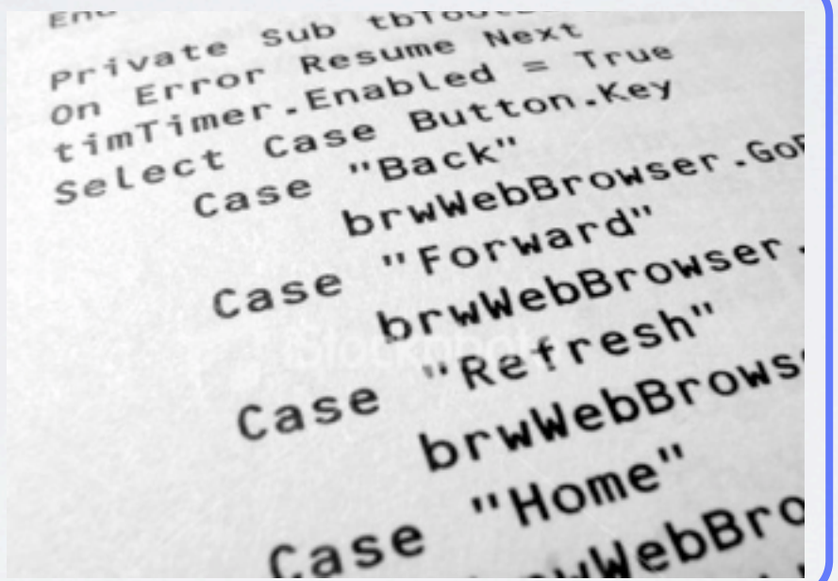
Celem formalnej weryfikacji nie jest tworzenie poprawnych systemów komputerowych ...

lecz dostarczenie metodologii, która pozwoliłaby zwiększyć ich niezawodność (zmniejszyć liczbę błędów).

WERYFIKACJA A WALIDACJA



walidacja



weryfikacja

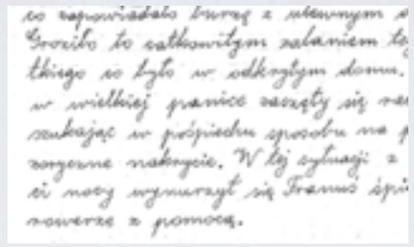
✔

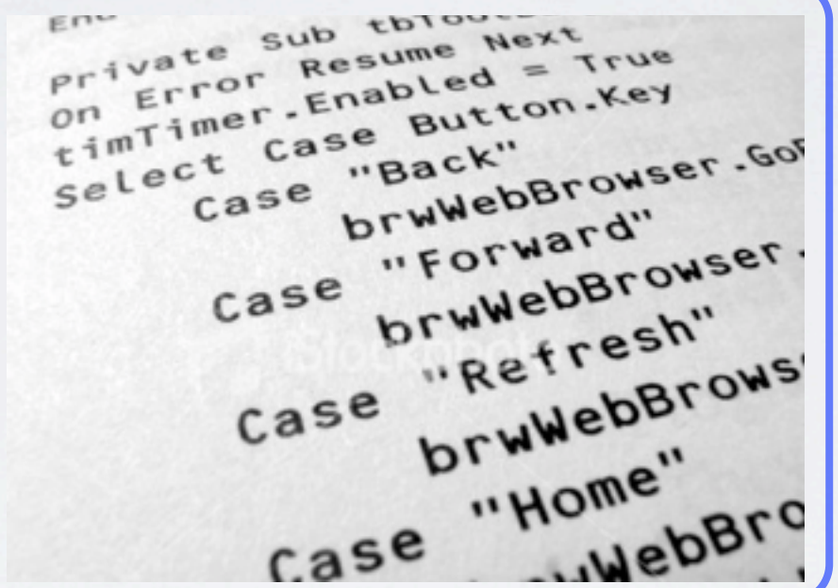
✘

WERYFIKACJA A WALIDACJA

do we build the right thing?

walidacja





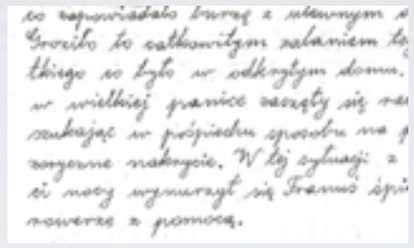
weryfikacja

✔

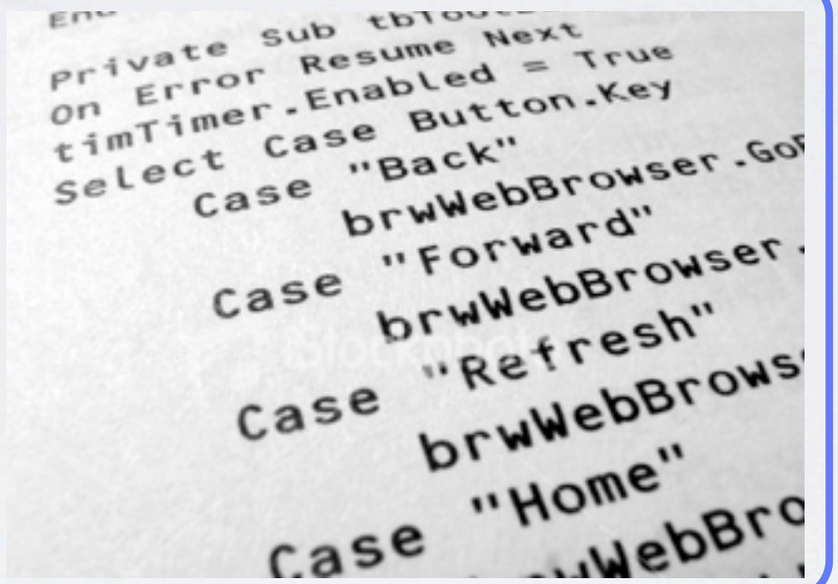
✘

WERYFIKACJA A WALIDACJA

do we build the right thing?



walidacja



weryfikacja

do we build the thing right?

✔

✘

WERYFIKACJA FORMALNA

- dowodzenie poprawności programów
- statyczna analiza programów (ang. static analysis)
- weryfikacja modelowa (ang. model-checking)

WERYFIKACJA FORMALNA

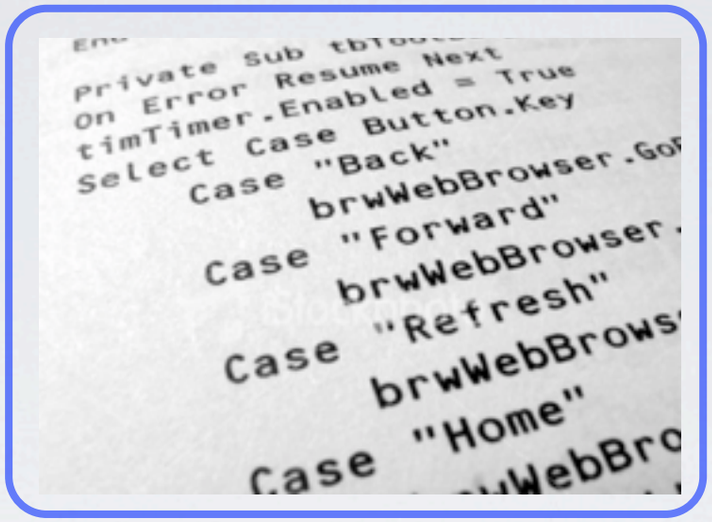
przybliżona

szersze znaczenie

- dowodzenie poprawności programów
- ~~statyczna~~ analiza programów (ang. static analysis)
- weryfikacja modelowa (ang. model-checking)

III. Dowodzenie poprawności programów

DOWODZENIE POPRAWNOŚCI



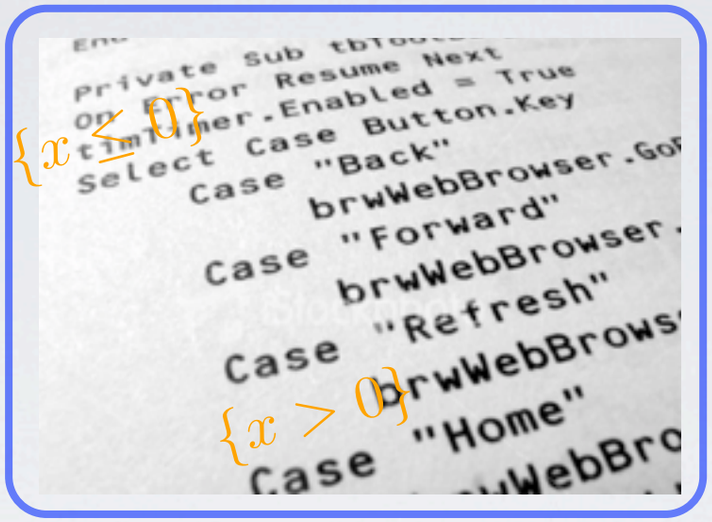
obligacje dowodowe

system wspomagający dowodzenie

dowód

?

DOWODZENIE POPRAWNOŚCI



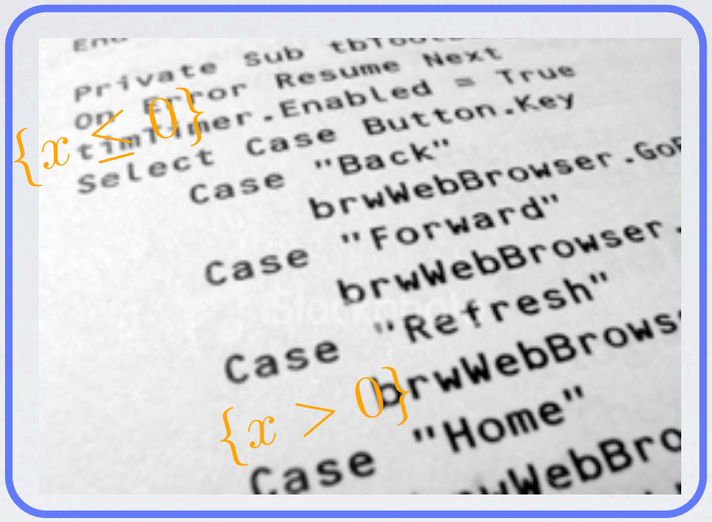
obligacje dowodowe

system wspomagający dowodzenie

dowód

?

DOWODZENIE POPRAWNOŚCI



obligacje dowodowe

automatycznie
lub
interakcyjnie

system wspomagający dowodzenie

dowód

?

DOWODZENIE POPRAWNOŚCI - CECHY CHARAKTERYSTYCZNE

- analizujemy udekorowany program źródłowy
- na ogół tylko częściowo automatycznie
- na ogół konieczny duży nakład pracy specjalisty
- stosowalne do niewielkich programów
- parametryzacja/generalizacja

PIONIERZY

PIONIERZY

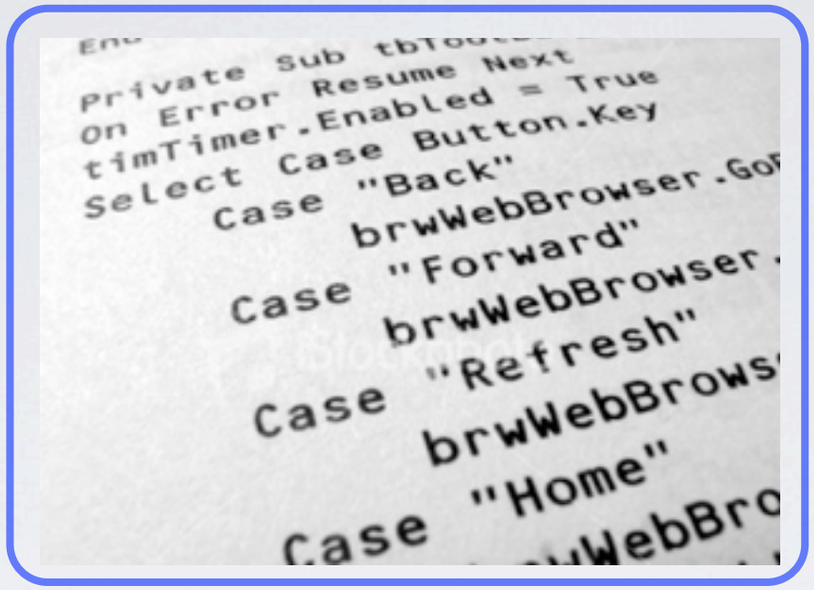
Edsger Dijkstra

Robert Floyd

C.A.R. Hoare

IV. Statyczna analiza programów

ANALIZA STATYCZNA



analizator statyczny

na pewno ✔

chyba ✘

ANALIZA STATYCZNA - CECHY CHARAKTERYSTYCZNE

- analizujemy program źródłowy (diagram przepływu sterowania)
- analiza przybliżona - fałszywe alarmy (ang. false positives)
- na ogól ukierunkowana na specyficzną własność
- w pełni automatyczna
- stosowalna do programów dużego rozmiaru
- gdy odp. negatywna, informacja diagnostyczna

ANALIZA STATYCZNA - ZASTOSOWANIA

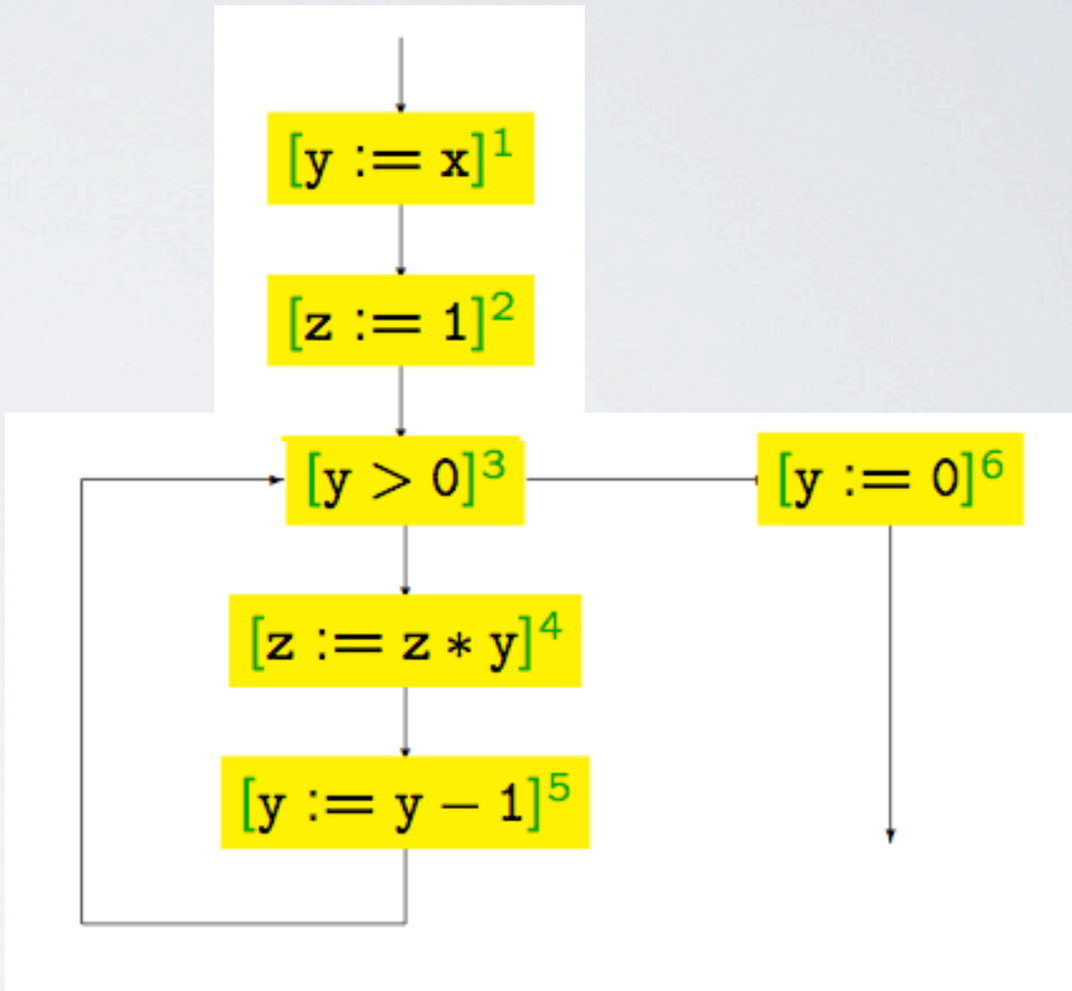
- w optymalizacji kompilacji
- w weryfikacji jakości kodu
- w weryfikacji programów

ANALIZA STATYCZNA - METODY

- analiza przepływu danych
- analiza przepływu sterowania
- analiza typów
- analiza kształtu
- ...
- abstrakcyjna interpretacja

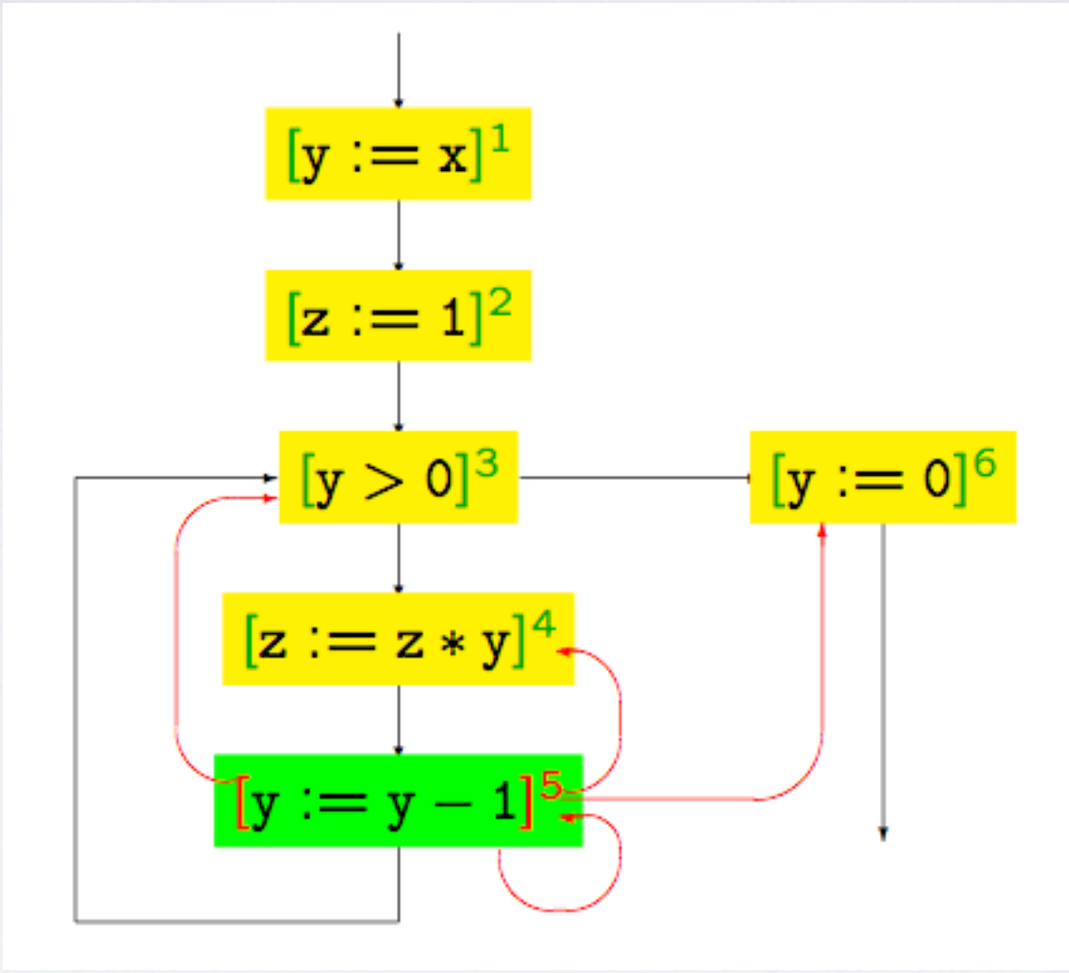
ANALIZA STATYCZNA - PRZYKŁAD

$[y := x]^1;$
$[z := 1]^2;$
while $[y > 0]^3$ **do**
  $[z := z * y]^4;$
  $[y := y - 1]^5$
od;
$[y := 0]^6$



[Nielson, Nielson, Hankin 2005]

„DOCIERAJĄCE'' PRZYPISANIA



[Nielson, Nielson, Hankin 2005]

„DOCIERAJĄCE'' PRZYPISANIA

	$\{(x,?),(y,?),(z,?)\}$
$[y := x]^1;$	$\{(x,?),(y,1),(z,?)\}$
$[z := 1]^2;$	$\{(x,?),(y,1),(y,5),(z,2),(z,4)\}$
while $[y > 0]^3$ **do**	$\{(x,?),(y,1),(y,5),(z,2),(z,4)\}$
$[z := z * y]^4;$	$\{(x,?),(y,1),(y,5),\boxed{(z,2)},(z,4)\}$
$[y := y - 1]^5$	$\{(x,?),\boxed{(y,1)},(y,5),\boxed{(z,2)},(z,4)\}$
od;	$\{(x,?),(y,1),(y,5),(z,2),(z,4)\}$
$[y := 0]^6$	$\{(x,?),(y,6),(z,2),(z,4)\}$

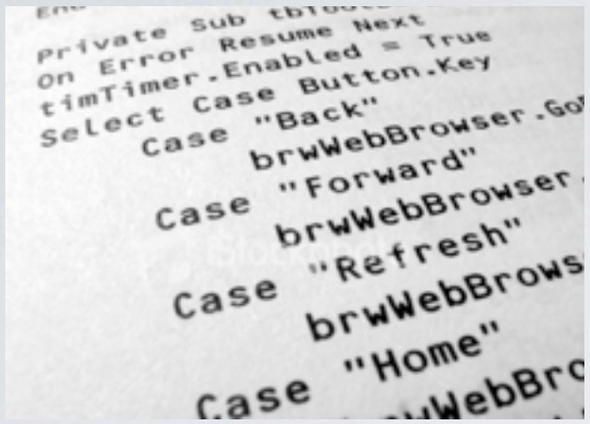
[Nielson, Nielson, Hankin 2005]

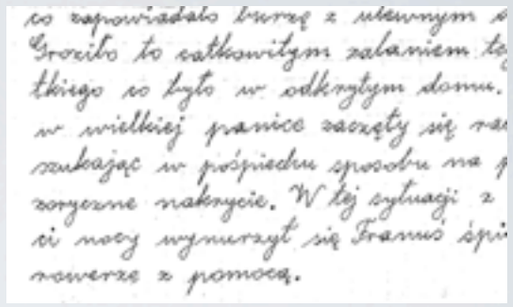
„DOCIERAJĄCE'' PRZYPISANIA

- formalizujemy problem jako układ równań
- najmniejsze rozwiązanie
- algorytm iteracyjny

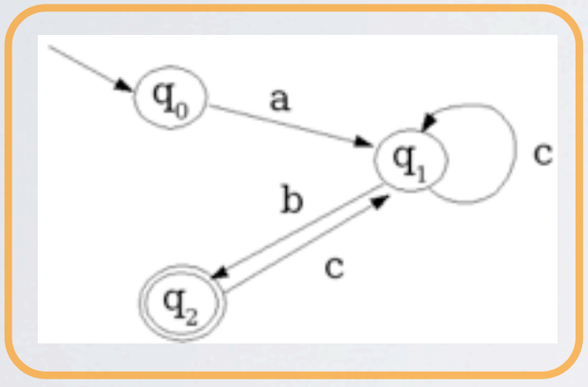
V. Weryfikacja modelowa

WERYFIKACJA MODELOWA





$\exists_X((\forall_x \exists_{y \leq x} y \in X)$



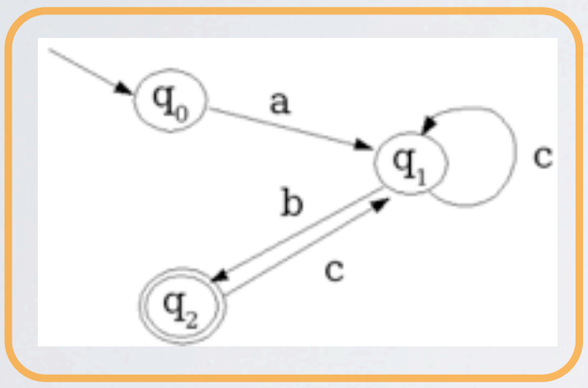
weryfikator modelowy

✔

kontrprzykład

błąd

WERYFIKACJA MODELOWA



$\exists_X((\forall_x \exists_{y \leq x} y \in X)$

weryfikator modelowy

✔

kontrprzykład

↓

błąd

WERYFIKACJA MODELOWA

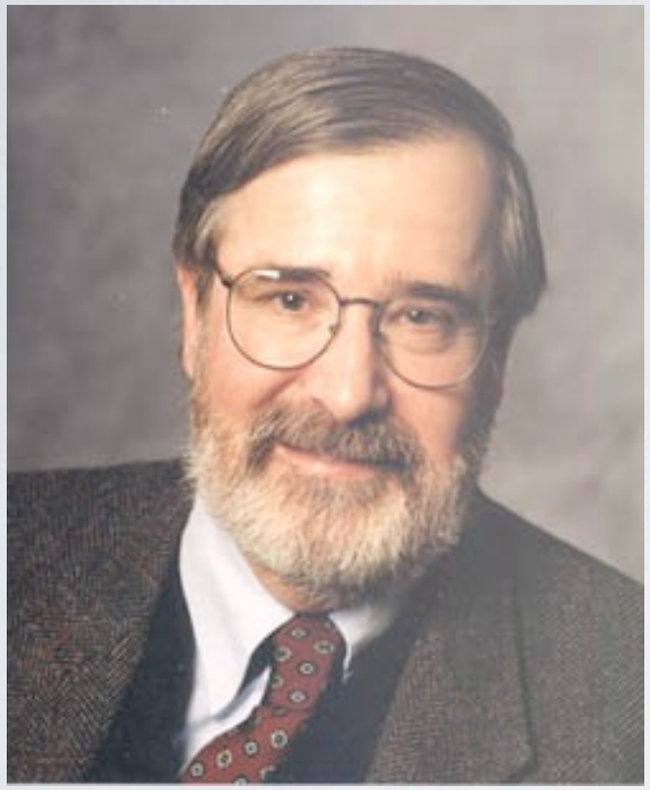
- model M - możliwe zachowania systemu
- specyfikacja własności Φ - dopuszczalne zachowania systemu
- automatycznie sprawdzamy, czy

M spełnia Φ

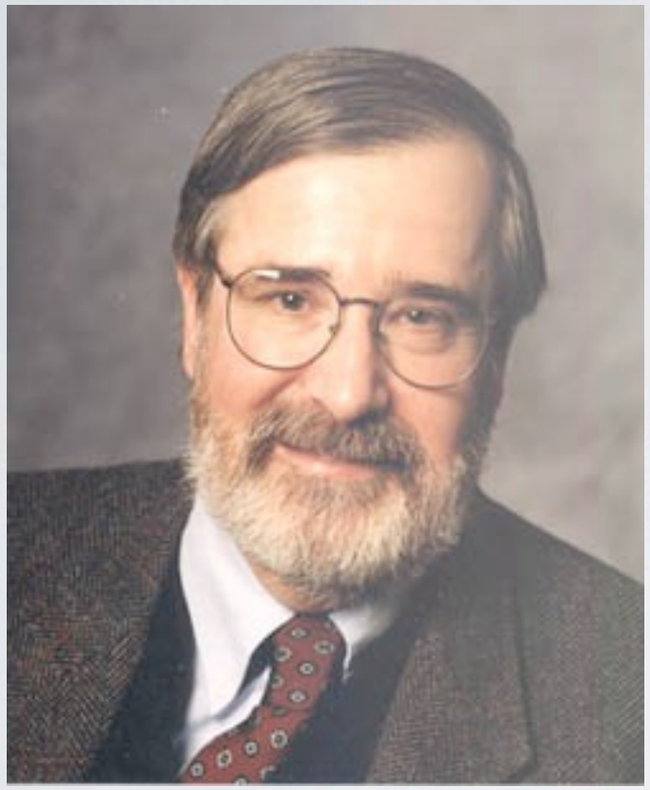
PRZYKŁADOWE WŁASNOŚCI

- bezpieczeństwo: wszystkie stany osiągalne spełniają ϕ
- żywotność: zawsze osiągniemy stan, który spełnia ϕ
- sprawiedliwość: ϕ będzie spełnione nieskończenie wiele razy

NAGRODA TURINGA 2007



NAGRODA TURINGA 2007

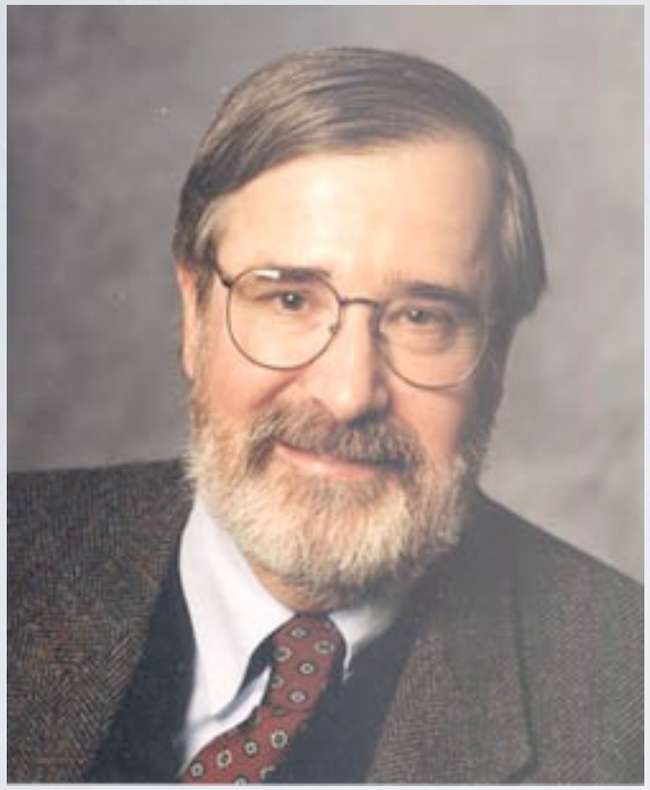


Ed Clarke

Allen Emerson

Joseph Sifakis

NAGRODA TURINGA 2007



Ed Clarke

Allen Emerson

Joseph Sifakis

nagroda Turinga 1978

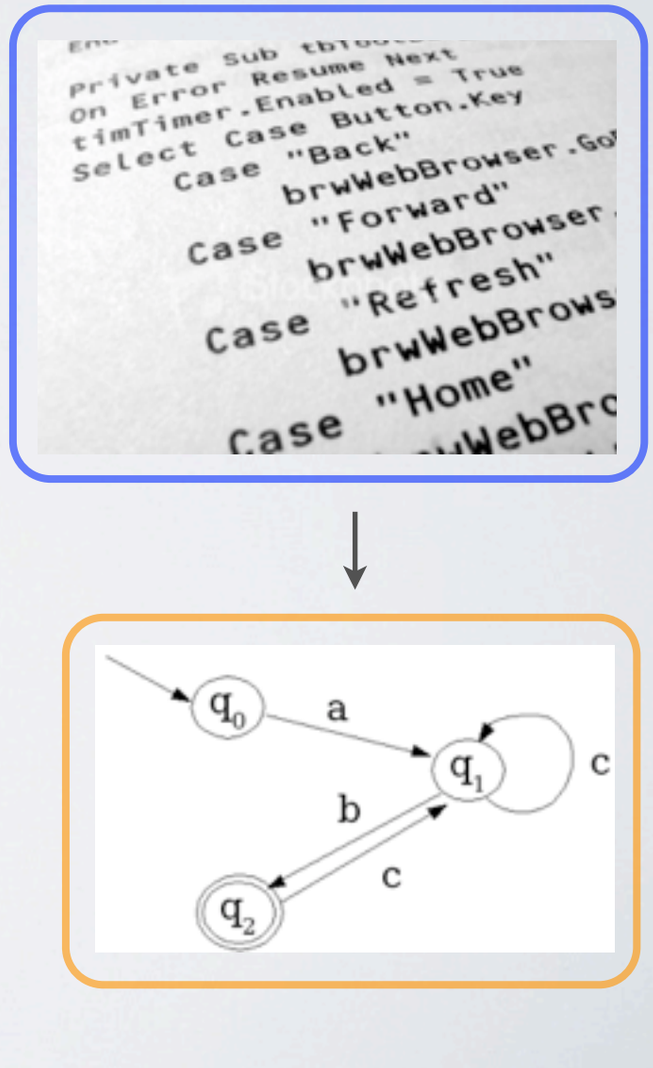
Robert Floyd

WERYFIKACJA MODELOWA - CECHY CHARAKTERYSTYCZNE

- model systemu (graf stanów)
- analiza poprzez przeszukiwanie wszystkich przebiegów (zachowań) modelu
- specyfikacja wymagań = formuła temporalna
- metoda (prawie) w pełni automatyczna
- stosowalna do programów średniego rozmiaru
- gdy odp. negatywna, informacja diagnostyczna - kontrprzykład

OD SYSTEMU DO MODELU

- tylko częściowo automatycznie
- kluczowy jest odpowiedni wybór poziomu abstrakcji
- weryfikujemy nie system, lecz jego model!

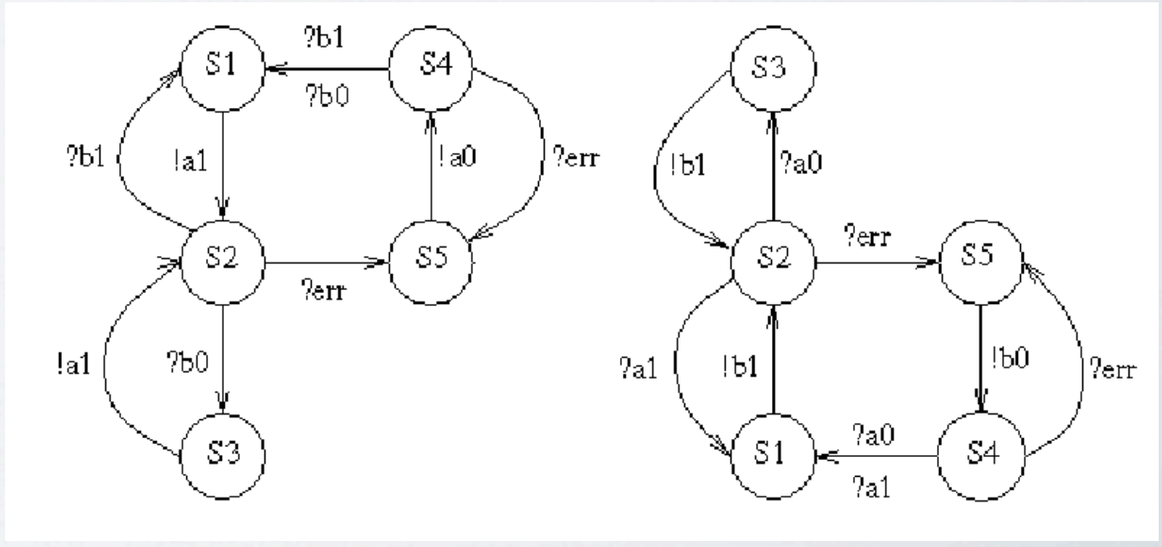


JAKI MODEL?

- funkcyjny (relacyjny): dane/wynik
- reaktywny:
  - interakcja z otoczeniem
  - działanie może się nie kończyć

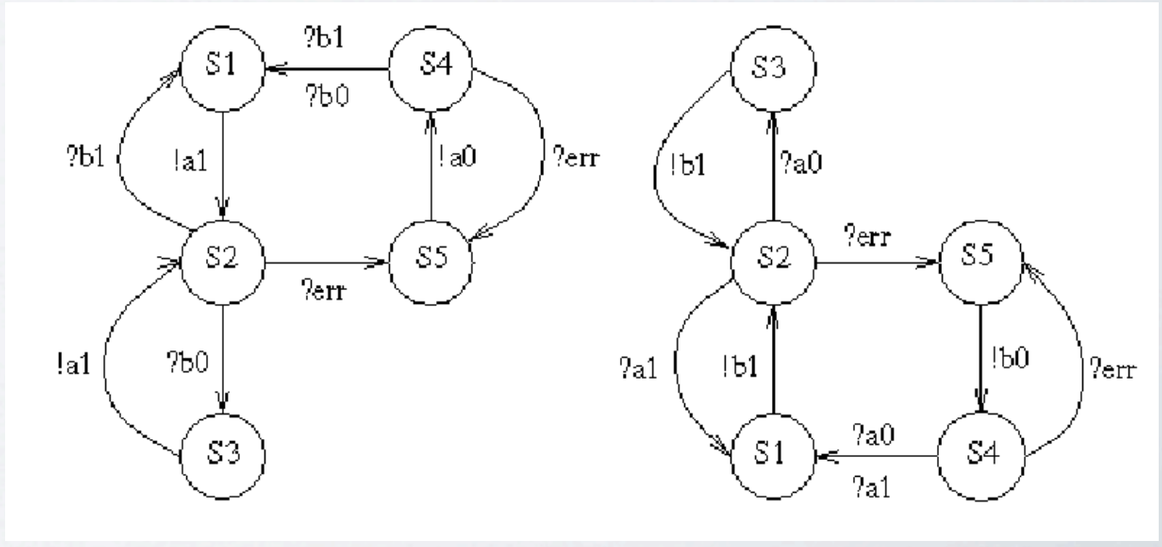
MODEL = STEROWANIE + INTERAKCJA

- brak skomplikowanych danych i obliczeń na nich
- abstrakcyjny (niedeterminizm)
- kompozycjonalny
- współbieżność, interakcja między składowymi (niedeterminizm)



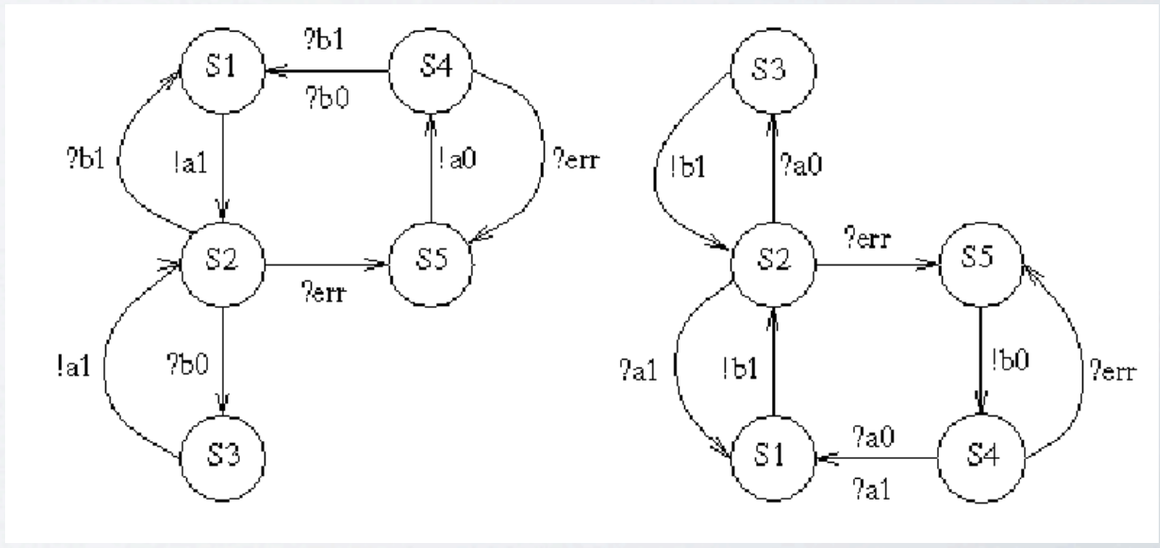
STAN

- stan lokalny = punkt sterowania +
  - wartości zmiennych +
  - zawartość kanałów komunikacyjnych +
  - ...
- stan globalny = stany lokalne + ...

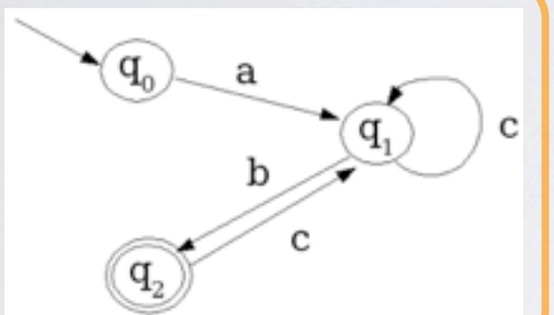


EKSPLOZJA STANÓW

$$G = L_1 \times \ldots \times L_n$$



WERYFIKACJA MODELOWA



$\exists_X((\forall_x \exists_{y \leq x} y \in X)$

weryfikator modelowy

✔

brak zasobów!

kontrprzykład

↓

błąd

NIE TYLKO OPROGRAMOWANIE

- sprzęt (NuSMV)
- protokoły, oprogramowanie systemowe, sterowniki (Spin)
- oprogramowanie (CBMC, Java Pathfinder)
- systemy zależne od czasu (UPPAAL)
- systemy probabilistyczne (PRISM)

VI. Historia

PREHISTORIA

- Goldstine, v. Neumann (1947), Turing (1949) } diagramy, asercje
- Floyd (1967), Hoare (1969), Dijkstra (1976)
- Pratt, Harel (1976-79): logika dynamiczna programów
- Owicki, Gries (1976): logika Hoare'a dla programów współb.
- Kamp (1968): LTL, Pnueli (1977): zastosowanie w weryfikacji
- lata 70': analiza statyczna w optymalizacji kompilacji
- (1979) lint - analiza statyczna programów w C
- (1971) Boyer-Moore theorem prover

PREHISTORIA

- Goldstine, v. Neumann (1947), Turing (1949) } diagramy, asercje
- Floyd (1967), Hoare (1969), Dijkstra (1976)
- Pratt, Harel (1976-79): logika dynamiczna programów
- Owicki, Gries (1976): logika Hoare'a dla programów współb.
- Kamp (1968): LTL, Pnueli (1977): zastosowanie w weryfikacji
- lata 70': analiza statyczna w optymalizacji kompilacji
- (1979) lint - analiza statyczna programów w C
- (1971) Boyer-Moore theorem prover

nagroda Turinga 1996

HISTORIA (LATA 80')

- Clarke, Emerson (1980), Ben-Ari, Manna, Pnueli (1981): CTL*
- Clarke, Emerson (1981), Queille, Sifakis (1982): narodziny weryfikacji modelowej
- EMC: kilkadziesiąt tysięcy stanów
- (pocz. lat 80') weryfikacja kompozycjonalna: „assume-guarantee''
- lata 80': systemy wspomagania dowodzenia i ich zastosowanie w weryfikacji:
    - Boyer-Moore, Isabelle, HOL, PVS, Coq, Mizar, ...

HISTORIA (LATA 90')

- Clarke, McMillan, i inni (1988-1990): symboliczna weryfikacja modelowa oparta na BDD
    - SMV: 10^20 ... 10^50 stanów (układy sprzętowe)
- (1994-95) narzędzia komercyjne:
    - systemy wspom. dowodzenia, weryfikatory modelowe
- (1998-99) ograniczona weryfikacja modelowa oparta na SAT
- lata 90': systemy wspomagania dowodzenia - dalszy rozwój:
    - Boyer-Moore, Isabelle, HOL, PVS, Coq, Mizar, ...

HISTORIA (LATA 00')

- rozwój metod weryfikacji modelowej opartej na SAT
- weryfikacja modelowa oprogramowania (abstrakcje)
- narzędzia (przykładowe, dla języków C i Java):
    - dowodzenie poprawności: ESC/Java2, KeY
    - analiza statyczna: FindBugs, PMD, Splint, Coverity, SLAM
    - weryfikacja modelowa: CBMC, Java Pathfinder, Bandera, Bogor, BLAST, Magic
- systemy zależne od czasu i probabilistyczne
- nowe zastosowania, np. w bioinformatyce

VII. Podsumowanie

GRANICE

- podział nie jest sztywny
- metody stosujące łącznie weryfikacje modelową, analizę statyczną i/lub dowodzenie poprawności
- weryfikacja modelowa programów (ang. software model checking)
- wstępna analiza statyczna przed weryfikacją modelową
- weryfikacja modelowa jako dowodzenie poprawności
- ...

OGRANICZENIA

- weryfikacja na ogół częściowa
- błędy w narzędziach lub w specyfikacji

INNE METODY

- dynamiczna analiza programów
- testowanie / symulacja, miary pokrycia testami, ...
- metryki jakości kodu źrodłowego (zarządzanie jakością kodu)
- audyt kodu źrodłowego
- systematyczna konstrukcja poprawnych programów
- ...

PODSTAWY TEORETYCZNE

- logika, teoria mnogości (twierdzenia o punktach stałych)
- teoria automatów
- modele współbieżności
- algorytmy grafowe

CZEGO NIE BĘDZIE?

- dostowanie ogólnej metodologii do specyfiki zastosowań
- włączenie weryfikacji formalnej do procesu tworzenia systemów komputerowych
- zarządzanie procesem weryfikacji
- zastosowania do rzeczywistych systemów
- heurystyki dla wydajności
- ...

PORÓWNANIE

- dowodzenie porawności programów
- analiza statyczna programów
- weryfikacja modelowa

PORÓWNANIE

parametryzacja

praca eksperta

precyzja

- dowodzenie porawności programów
- analiza statyczna programów
- weryfikacja modelowa

szybkość

fałszywe alarmy

pełna automatyczność

współbieżność

eksplozja stanów

układy sprzętowe