

# Testování konečných automatů

Radek Mařík

ČVUT FEL, K13132

October 2, 2014



## Obsah

- 1 Konečný automat - základy
  - Definice
  
- 2 Neformální přístup testování automatů
  - Terminologie
  - Postup
  - Problémy
  
- 3 Formalizace testování automatů
  - Definice
  - Příklad



# Konečné automaty v praxi <sup>[Bei95]</sup>

- výborný model pro testování aplikací řízených pomocí menu,
- **software řízený pomocí menu:** primární ovládání se provádí pomocí výběru z položek menu.
- široké použití v objektově orientovaném návrhu.

## Konečný automat

- abstraktní stroj, jehož počet stavů a vstupních symbolů je konečný a neměnný.
- skládá se ze
  - stavů (vrcholy),
  - přechodů (hrany),
  - vstupů (označení hran) a
  - výstupů (označení hran či uzlů).



# Konečný automat <sup>[HI98]</sup>

- Nechť *Input* je konečná abeceda.
- *Konečný stavový automat* nad *Input* obsahuje následující položky:
  - 1 konečnou množinu  $Q$  prvků nazývanou *stavy*.
  - 2 podmnožinu  $I$  množiny  $Q$  obsahující *počáteční stavy*.
  - 3 podmnožinu  $T$  množiny  $Q$  obsahující *konečné stavy*.
  - 4 konečnou množinu *přechodů*, které pro každý stav a každý symbol vstupní abecedy vrací následující stav.

## Přechodová funkce

$$F : Q \times \text{Input} \rightarrow \mathcal{P}Q$$

- $F(q, \text{input})$  obsahuje možné stavy automatu, do kterých lze přejít ze stavu  $q$  po přijmutí symbolu *input*.
- $\mathcal{P}Q$  označuje množinu všech podmnožin  $Q$  (*potenční množina množiny*  $Q$ ).

# Konečný automat s výstupem <sup>[H198]</sup>

- *Input* konečná abeceda.
- *Konečný automat* nad množinou *Input* obsahuje následující komponenty:
  - 1 Konečná množina  $Q$  prvků nazývaných *stavy*.
  - 2 Podmnožina  $I$  množiny  $Q$  obsahující *počáteční stavy*.
  - 3 Podmnožina  $T$  množiny  $Q$  obsahující *koncové stavy*.
  - 4 Množina *Output* možných výstupů.
  - 5 Konečná množina *přechodů*, které pro každý stav a každý symbol vstupní abecedy vrací množinu možných následujících stavů.

## Výstupní funkce

$$G : Q \times Input \rightarrow Output$$

- pro každý stav a pro každý vstupní symbol určuje výstupní symbol.
- **F** a **G** mohou být parciální funkce.

# Příklady konečných automatů <sup>[H198]</sup>

## Množina *Input*

- akce či příkazy uživatele zadaných na klávesnici,
- kliky či pohyby myše,
- přijmutí signálu ze senzoru.

## Množina stavů $Q$

- hodnoty jistých důležitých proměnných systému,
- mód chování systému,
- druh formuláře, který je viditelný na monitoru,
- zda jsou zařízení aktivní či ne.



# Kódování vstupů <sup>[Bei95]</sup>

## Vstupy

- **Vstupní událost:** rozlišitelná opakovatelná událost jako fixní sekvence aktivy vstupů.
- **Kódování vstupních událostí:** přiřazení jména či čísla.
- **Vstupní symboly:** množina vzájemně různých symbolů použitých pro kódování vstupních událostí.



# Kódování stavů <sup>[Bei95]</sup>

## Stavy

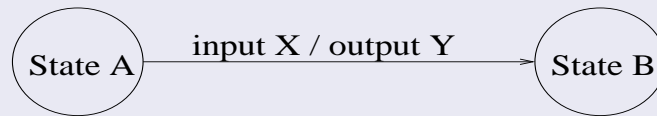
- **Stav:** stavy se zobrazují jako uzly diagramu stavového automatu.
- **Kód stavu:** přiřazení symbolů ke stavům.
- **Okamžitý stav:** stav, ve kterém se právě systém nachází.
- **Počáteční stav:** speciální stav systému, ve kterém se systém nachází před přijmutím jakéhokoli vstupní události.
- **Čítač stavů:** hypotetické nebo aktuální místo paměti držící kód okamžitého stavu.
- **Počet stavů:** počet vzájemně různých kódů stavu.



## Přechody a výstupy <sup>[Bei95]</sup>

### Přechody

- **Přechod:** odezva systému na vstupní událost, při které se může změnit jeho stav.



- **Vlastní přechod:** při přechodu se stav nezmění; hrana vede stavu zpět do tohoto stavu.

### Výstupy

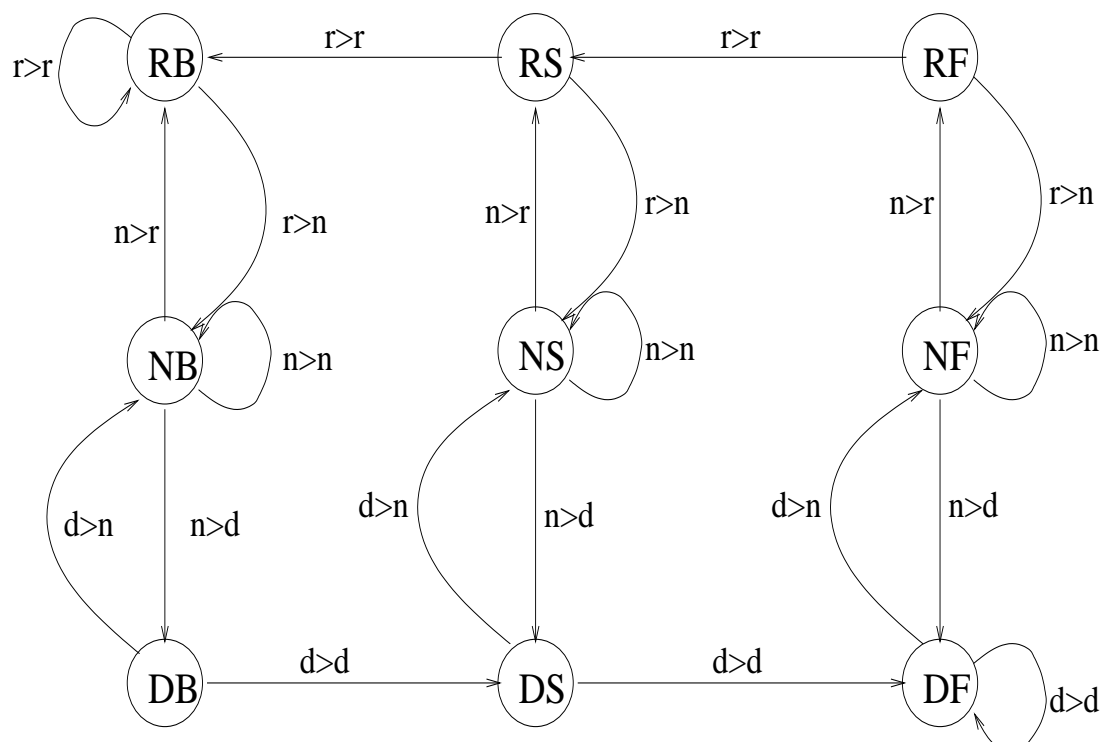
- **Výstupní událost:** systém může produkovat na svém výstupu aktivity při změnách stavu či při přechodech.
- **Kódování výstupu:** symbol výstupní události.
- **Nulový výstup:** hypotetická výstupní událost, při které systém na svém výstupu neprovede žádnou aktivitu.

## Stavový diagram <sup>[Bei95]</sup>

- **Vrcholy:** zobrazují stavy (stav softwarové aplikace).
- **Hrany:** znázorňují přechody (výběr položky v menu).
- **Atributy hran (vstupní kódy):** např. akce myši, Alt+Key, funkční klíče, klávesy pohybu kursoru.
- **Atributy hran (výstupní kódy):** např. zobrazení jiného menu či otevření dalšího okna.

### Model vesmírné lodi *Enterprise*

- tři nastavení impulsního motoru:  
tah vpřed(d), neutrál(n), a zpětný tah(r).
- tři možné stavy pohybu:  
pohyb dopředu(F), zastavena(S), a pohyb vzad(B).
- kombinace vytvoří devět stavů:  
DF, DS, DB, NF, NS, NB, RF, RS, a RB.
- možné vstupy:  $d > d, r > r, n > n, d > n, n > d, n > r, r > n$ .

Stavový prostor Enterprise <sup>[Bei95]</sup>BACKWARD  $\diamond$  STOPPED  $\diamond$  FORWARDVlastnosti stavových diagramů <sup>[Bei95]</sup>

## Vlastnosti

- silně souvislý graf,
- stavové grafy rostou velmi rychle,
- typicky se uvažují všechny možné i nemožné vstupy v daném stavu - implementace systému nemusí být správná.
- pěkná symetrie je velmi řídký jev v praxi.

Přechodové tabulky <sup>[Bei95]</sup>

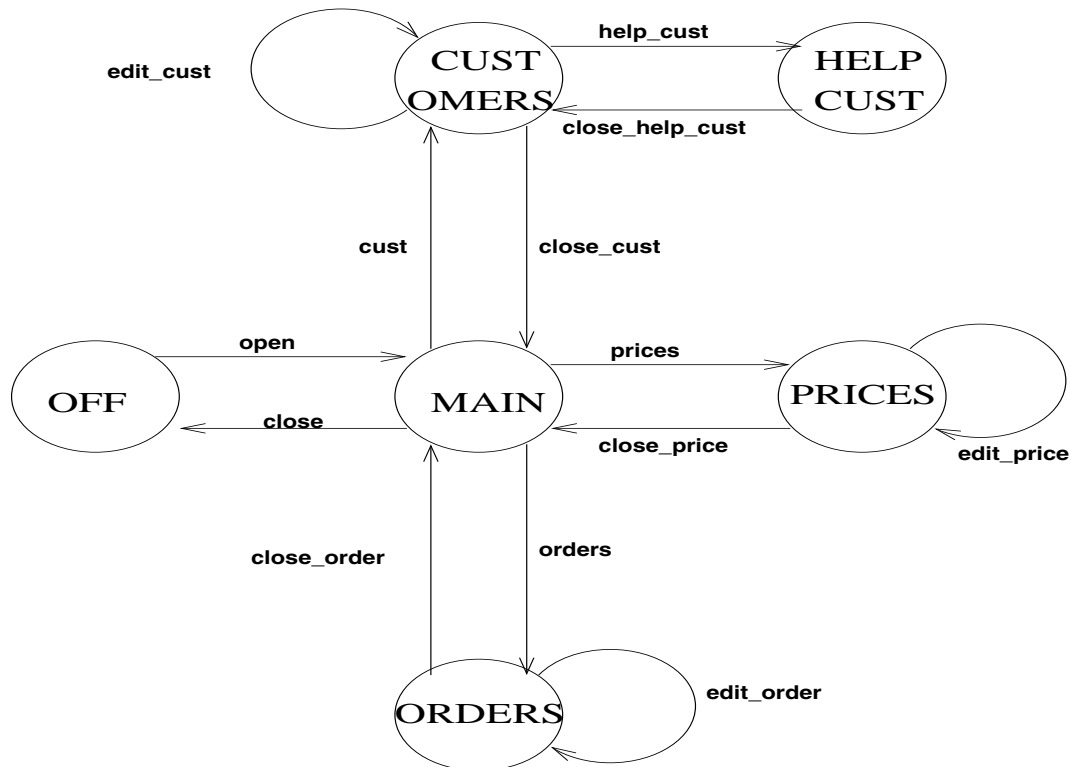
- má pro každý stav jeden řádek a pro každý vstup jeden sloupec,
- ve skutečnosti jsou tabulky dvě s stejným tvarem:
  - tabulka přechodů
  - tabulka výstupů
- hodnotou pole v tabulce přechodů je příští stav,
- hodnotou pole v tabulce výstupů je výstupní kód pro daný přechod.
- **hierarchické (vnořené) automaty** jsou jedinou cestou, jak se vyhnout obrovským tabulkám (např. stavová schémata, angl. statechart, starchart, atd.)

Přechodová tabulka Enterprise <sup>[Bei95]</sup>

## Enterprise

STATE	$r > r$	$r > n$	$n > n$	$n > r$	$n > d$	$d > d$	$d > n$	$r > d$	$d > r$
RB	RB	NB							
RS	RB	NS							
RF	RS	NF							
NB			NB	RB	DB				
NS			NS	RS	DS				
NF			NF	RF	DF				
DB						DS	NB		
DS						DF	NS		
DF						DF	NF		



Příklad - estimátor <sup>[HI98]</sup>Dosažitelnost stavů <sup>[Bei95]</sup>

- **Dosažitelný stav:** stav  $B$  je dosažitelný ze stavu  $A$ , jestliže existuje sekvence vstupů taková, která převede systém ze stavu  $A$  do stavu  $B$ .
- **Nedosažitelný stav:** stav je nedosažitelný, pokud není dosažitelný, zvláště z počátečního stavu. Nedosažitelné stavy znamenají typicky chybu.
- **Silně souvislý:** všechny stavy konečného automatu jsou dosažitelné z počátečního stavu. Většina modelů v praxi je silně souvislá, pokud neobsahují chyby.
- **Isolované stavy:** množina stavů, které nejsou dosažitelné z počátečního stavu. Pokud existují, jedná se o velmi podezřelé, chybové stavy.
- **Reset:** speciální vstupní akce způsobující přechod z jakéhokoliv stavu do počátečního stavu.





## Rozdělení stavů <sup>[Bei95]</sup>

- **Množina počátečního stavu:** Jakmile se provede přechod z této množiny, pak se do této množiny již nelze vrátit (např. boot systému).
- **Pracovní stavy:** po opuštění množiny počátečního stavu, se systém pohybuje v silně souvislé množině stavů, kde se provádí většina testování.
- **Počáteční stav pracovní množiny:** stav pracovní množiny, který je možné považovat za “výchozí stav”.
- **Množina koncových stavů:** dostane-li se systém do této množiny, nelze se zpět vrátit do pracovní množiny, např. ukončovací sekvence programu.
- **Úplně specifikovaný:** je systém, pokud je přechody a výstupní kódy definovány pro jakoukoliv kombinaci vstupního kódu a stavu.
- **Okružní cesta stavu  $A$ :** sekvence přechodů jdoucí ze stavu  $A$  do stavu  $B$  a zpět do  $A$ .



## Ověřování modelu <sup>[Bei95]</sup>

- 1 úplnost a konzistence, tj. kontrola chybějících vstupů, jednoznačnosti, rozpory, atd.
- 2 jednoznačné kódování vstupů,
- 3 minimální automaty,
- 4 modely, které nejsou silně souvislé, jsou typicky chybou modelu nebo chybou v návrhu.



# Obecný návod k testování automatů <sup>[Bei95]</sup>

- ① identifikuj vstupy.
- ② definuj kódy vstupů. Vstupy, které netestujeme se nezahrnují.
- ③ identifikuj stavy.
- ④ definuj kódování stavů.
- ⑤ identifikuj výstupní akce.
- ⑥ definuj kódování výstupních akcí.
- ⑦ specifikuj tabulku přechodů a tabulku výstupů a zkontroluj ji - jeden z nejnamahavějších kroků návrhu,
- ⑧ navrhni testy,
- ⑨ provedě testy,
- ⑩ pro každý vstup ověř jak přechod, tak i výstup.



# Návrh testů <sup>[Bei95]</sup>

- Každý test začíná v počátečním stavu.
- Z počátečního stavu se systém přivede nejkratší cestou k vybranému stavu, provede se zadaný přechod a systém se nejkratší možnou cestou přivede opět do počátečního stavu; vytváříme tzv. okružní cestu.
- Každý test staví na předchozích jednodušších testech.
- Určíme vstupní kód pro každý přechod okružní cesty.
- Určíme výstupní kódy asociované s přechody okružní cesty.
- **Ověříme**
  - kódování vstupů,
  - kódování výstupů,
  - stavy,
  - každý přechod.
- **Je každý koncový stav dosažitelný?**



# Skryté stavy

## • Je systém v počátečním stavu?

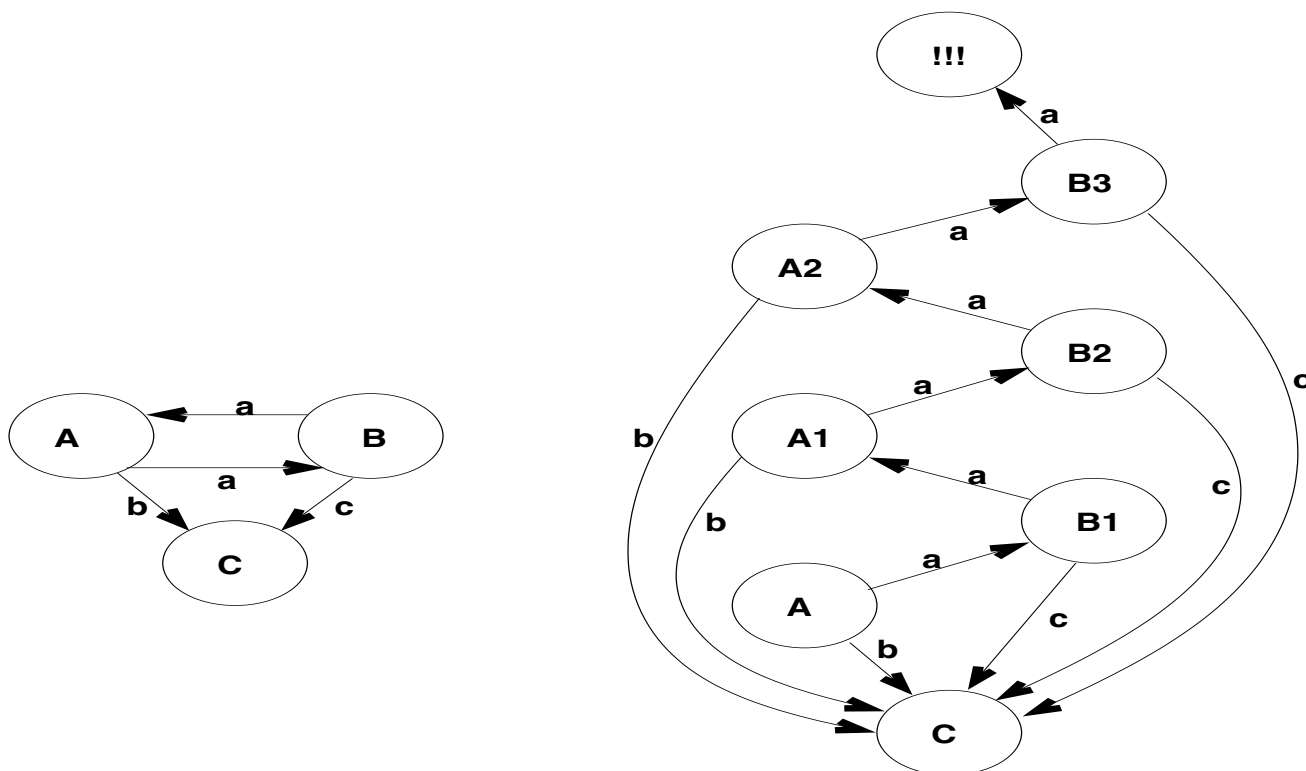
- Test nelze zahájit, pokud systém není potvrzeným způsobem v počátečním stavu.
- Aplikace si uchovávají persistentně své nastavení.
- Jestliže předchozí test selže, v jakém stavu se aplikace nachází?

## • Má implementace skryté stavy?

- Při testování softwaru můžeme předpokládat věci, které nemusí obecně platit.
  - např. že víme, ve kterém stavu se systém nachází.
- Typicky se nejedná o jeden či dva skryté stavy, ale stavový prostor se zdvojnásobuje či jinak násobí.



# Skryté stavy



# Testovatelnost <sup>[Bei95]</sup>

- ① explicitní počítadlo stavů,
- ② resetování do specifického stavu,
- ③ krokování,
- ④ trasování přechodů.
- ⑤ explicitní tabulka vstupního kódování,
- ⑥ explicitní tabulka výstupního kódování,
- ⑦ explicitní tabulka přechodové funkce.

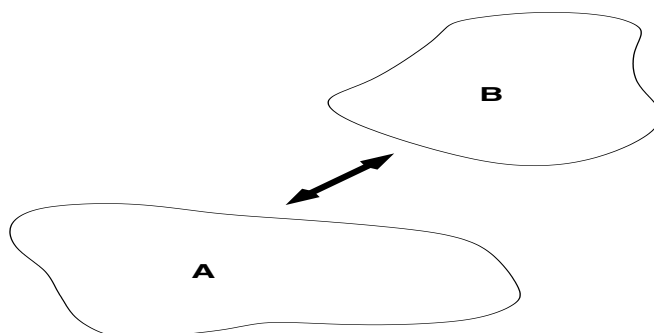
## Omezení:

- velké stavové grafy,
- vnořené modely versus vnořené systémy,
- nedostatečná podpora.



# Testování konečného automatu <sup>[H198]</sup>

- založeno na izomorfismu konečných automatů,
- $\mathcal{A} = (\text{Input}, Q, \mathbf{F}, q_0)$
- $\mathcal{A}' = (\text{Input}, Q', \mathbf{F}', q_0')$
- $g : \mathcal{A} \rightarrow \mathcal{A}'$
- $g : Q \rightarrow Q'$ 
  - ①  $g(q_0) = q_0'$
  - ②  $\forall q \in Q, \text{input} \in \text{Input},$   
 $g(\mathbf{F}(q, \text{input})) = \mathbf{F}'(g(q), \text{input})$



Konstrukce množiny testů <sup>[H198]</sup>

- Nechť  $L$  je množina vstupních sekvencí a  $q, q'$  dva stavy.  $L$  rozliší stav  $q$  od  $q'$ , jestliže existuje sekvence  $k$  v  $L$  taková, že výstup získaný aplikací  $k$  na automat ve stavu  $q$  je různý od výstupu získaný aplikací  $k$  na stav  $q'$ .
- Automat je *minimální*, pokud neobsahuje redundantní stavy.
- Množina vstupních sekvencí  $W$  se nazývá *charakterizační množina*, jestliže může rozlišit jakékoliv dva stavy automatu.
- **Pokrytí stavu** je množina vstupních sekvencí  $L$  taková, že lze nalézt prvek množiny  $L$ , kterým se lze dostat do jakéhokoliv žádaného stavu z počátečního stavu  $q_0$ .
- **Pokrytí přechodů** minimálního automatu je množina vstupních sekvencí  $T$ , která je pokrytím stavů a uzavřená z hlediska pravé kompozice s množinou vstupů  $Input$ .
  - $sequence \in T = L \bullet (Input^1 \cup \{<>\})$

Generování množiny testů <sup>[H198]</sup>

- O kolik je v implementaci více testů než ve specifikaci? ( $k$ )
- $Z = Input^k \bullet W \cup Input^{k-1} \bullet W \cup \dots \cup Input^1 \bullet W \cup W$ 
  - Jestliže  $A$  a  $B$  jsou množiny sekvencí stejné abecedy, pak  $A \bullet B$  značí množinu sekvencí, složených ze sekvencí množiny  $A$  následující sekvencí z  $B$ .
  - $k$  kroků do "neznámého" prostoru následovaných ověřením stavu

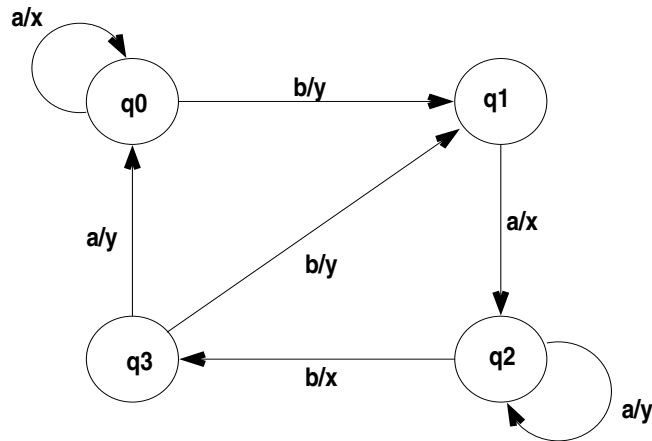
- Konečná **množina testů**:

$$T \bullet Z$$

- Pokrytí přechodů zajišťuje,
  - že všechny stavy a přechody specifikace jsou implementovány,
  - množina  $Z$  zajišťuje, že implementace je ve stejném stavu, který určuje specifikace.
  - Parametr  $k$  jistí, že do jisté úrovně všechny skryté stavy implementace jsou testovány.



# Jednoduchý příklad <sup>[HI98]</sup>



- $Input = \{a, b\}$
- $L = \{ \langle \rangle, b, b::a, b::a::b \}, \langle \rangle \dots$  nulový vstup
- $T = \{ \langle \rangle, a, b, b::a, b::b, b::a::a, b::a::b, b::a::b::a, b::a::b::b \}$
- $W = \{a, b\}$  <sup>[Chy84]</sup>, pp. 31–34  
 $Z = Input \bullet W \cup W$
- $= \{a, b\} \bullet \{a, b\} \cup \{a, b\}$   
 $= \{a, b, a::a, a::b, b::a, b::b\}$



# Testovací množina příkladu <sup>[HI98]</sup>

$$\begin{aligned}
 T \bullet Z &= \\
 &= \{ \langle \rangle, a, b, b::a, b::b, b::a::a, b::a::b, b::a::b::a, b::a::b::b \} \\
 &\quad \bullet \{a, b, a::a, a::b, b::a, b::b\} \\
 &= \{ a, b, a::a, a::b, b::a, b::b, \\
 &\quad a::a, a::b, a::a::a, a::a::b, a::b::a, a::b::b, \\
 &\quad b::a, b::b, b::a::a, b::a::b, b::b::a, b::b::b, \\
 &\quad b::a::a, b::a::b, b::a::a::a, b::a::a::b, b::a::b::a, b::a::b::b, \\
 &\quad b::b::a, b::b::b, b::b::a::a, b::b::a::b, b::b::b::a, b::b::b::b, \\
 &\quad b::a::a::a, b::a::a::b, b::a::a::a::a, b::a::a::a::b, b::a::a::b::a, b::a::a::b::b, \\
 &\quad b::a::b::a, b::a::b::b, b::a::b::a::a, b::a::b::a::b, b::a::b::b::a, b::a::b::b::b, \\
 &\quad b::a::b::a::a, b::a::b::a::b, b::a::b::a::a::a, \\
 &\quad b::a::b::a::a::b, b::a::b::a::b::a, b::a::b::a::b::b, \\
 &\quad b::a::b::b::a, b::a::b::b::b, b::a::b::b::a::a, \\
 &\quad b::a::b::b::a::b, b::a::b::b::b::a, b::a::b::b::b::b \} \\
 &= \dots \text{simplification}
 \end{aligned}$$



# Aplikace <sup>[Bei95]</sup>

- software řízený pomocí menu,
- objektově orientovaný software,
- protokoly,
- řadiče zařízení,
- starší hardware,
- mikropočítače průmyslových a domácích zařízení,
- instalace softwaru,
- software pro archivaci či obnovení.



# Literatura I



Boris Beizer.

*Black-Box Testing, Techniques for Functional Testing of Software and Systems.*  
John Wiley & Sons, Inc., New York, 1995.



Michal Chytil.

*Automaty a gramatiky.*  
SNTL Praha, 1984.



Mike Holcombe and Florentin Ipaté.

*Correct Systems: Building a Business Process Solution.*  
Springer, 1998.

