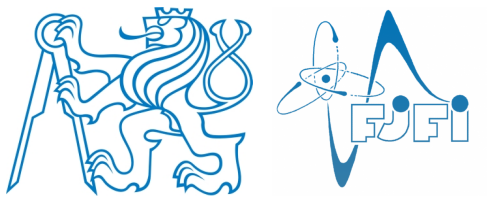
Testování konečných automatů

Radek Mařík

ČVUT FEL
Katedra telekomunikační techniky, K13132

25. října 2017



Obsah

Konečné automaty v praxi [Bei95]

- 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 [HI98]

- 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 množinu Q obsahující *konečné stavy*.
  4. konečnou množinu *přechodů*, které pro každý stav a každý symbol vstupní abecedy vrací následující stav.

Přechodová funkce

$$\mathbf{F} : Q \times Input \to \mathcal{P}Q$$

- $\mathbf{F}(q, input)$ 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 [HI98]

- *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

$$\mathbf{G} : Q \times Input \rightarrow Output$$

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

Příklady konečných automatů [HI98]

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 [Bei95]

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 [Bei95]

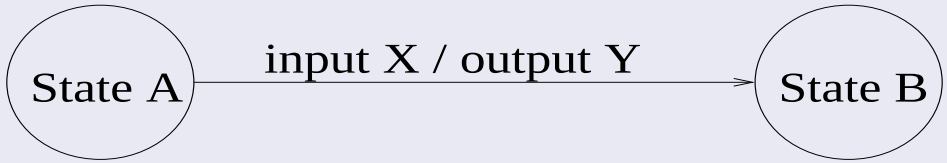
Stavy

- **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 držicí kód okamžitého stavu.
- **Počet stavů:** počet vzájemně různých kódů stavu.

Přechody a výstupy [Bei95]

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 údalost, při které systém na svém výstupu neprovede žádnou akvititu.

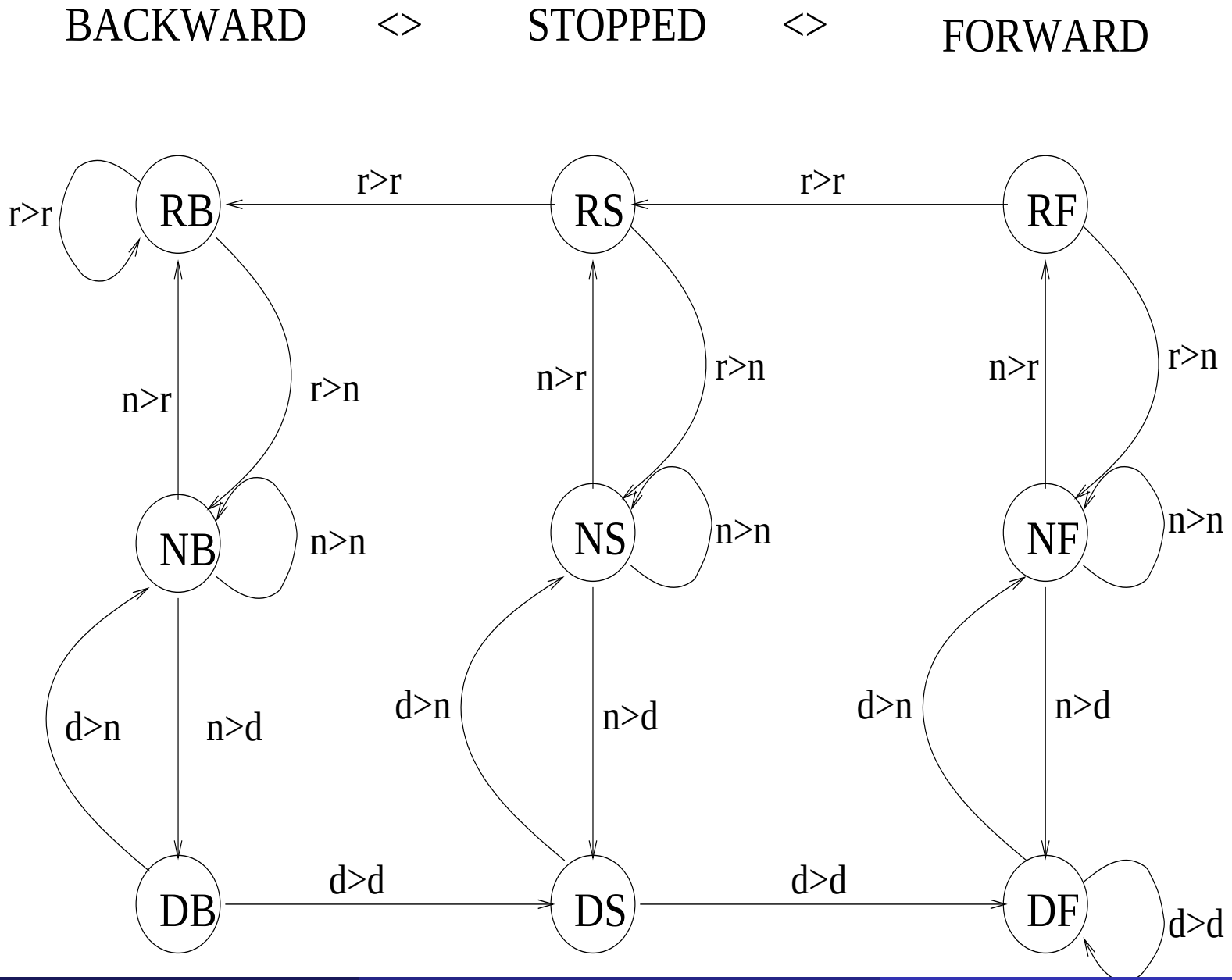
Stavový diagram [Bei95]

- **Vrcholy:** zobrazují stavy (stav softwarové aplikace).
- **Hrany:** znázorňují přechody (výběr položky v menu).
- **Atributy hran (vstupní kódy):** např. akce myší, 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 [Bei95]



Vlastnosti stavových diagramů [Bei95]

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 [Bei95]

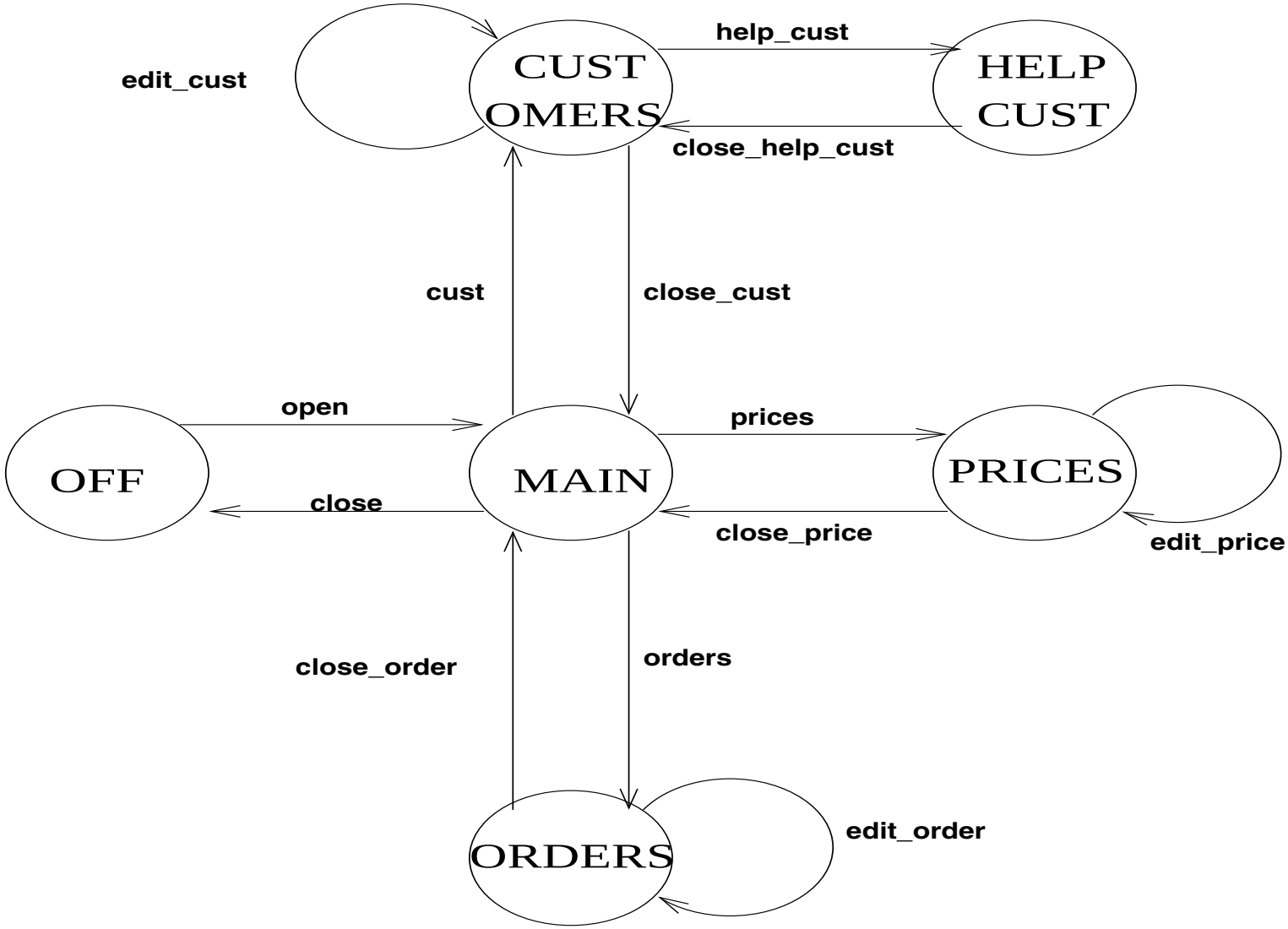
- má pro každý stav jeden řádek a pro každý vstup jeden sloupec,
- ve skutečnosti jsou tabulky dvě s stejným tvarem:
  - tabulka přechodů
  - tabulka výstupů
- hodnotou pole v tabulce přechodů je příští stav,
- hodnotou pole v tabulce výstupů je výstupní kód pro daný přechod.
- **hierarchické (vnořené)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

Přechodová tabulka Enterprise [Bei95]

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 [HI98]



Dosažitelnost stavů [Bei95]

- **Dosažitelný stav:** stav B je dosažitelný ze stavu A, jestliže existuje sekvence vstupů taková, která převede systém ze stav 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 [Bei95]

- **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 [Bei95]

1. úplnost a konzistence, tj. kontrola chybějících vstupů, nejednoznačnosti, 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 [Bei95]

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. proveď testy,
10. pro každý vstup ověř jak přechod, tak i výstup.

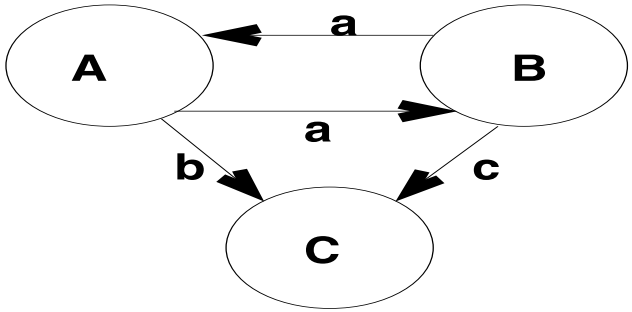
Návrh testů [Bei95]

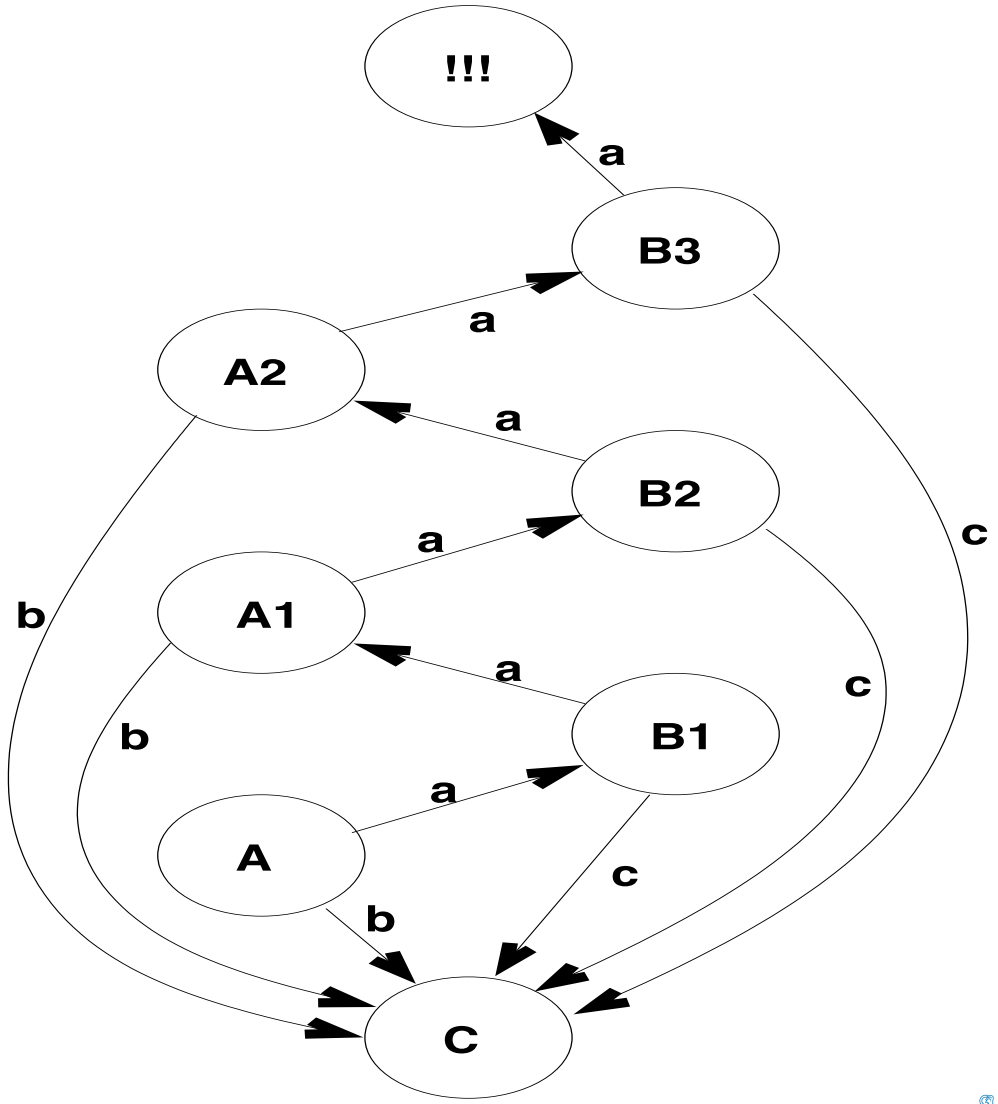
- Každý test začíná v počátečním stavu.
- Z počátečního stavu se systém přivede nejkratší cestou k vybranému stavu, provede se zadaný přechod a systém se nejkratší možnou cestou přivede opět do počátečního staru; vytváříme tzv. okružní cestu.
- Každý test staví na předchozích jednodušších testech.
- 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 [Bei95]

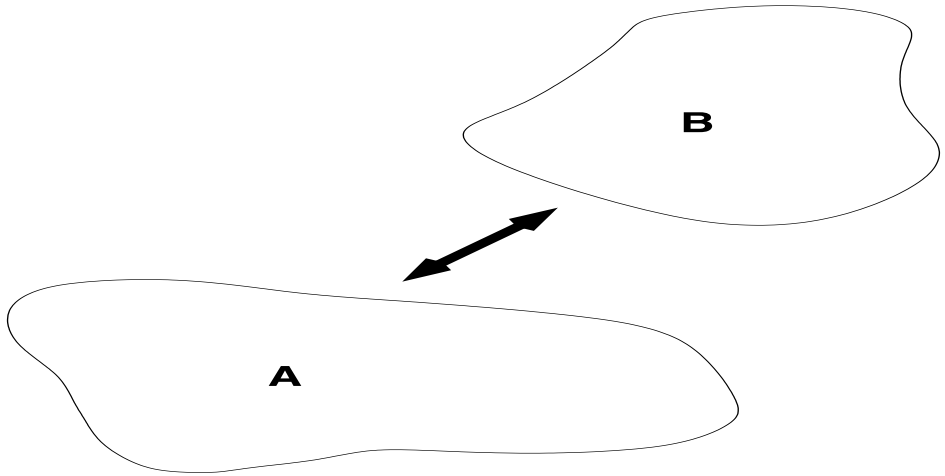
1. explicitní počítadlo stavů,
2. resetování do specifického stavu,
3. krokování,
4. trasování přechodů.
5. explicitní tabulka vstupního kódování,
6. explicitní tabulka výstupního kódování,
7. explicitní tabulka přechodové funkce.

Omezení:

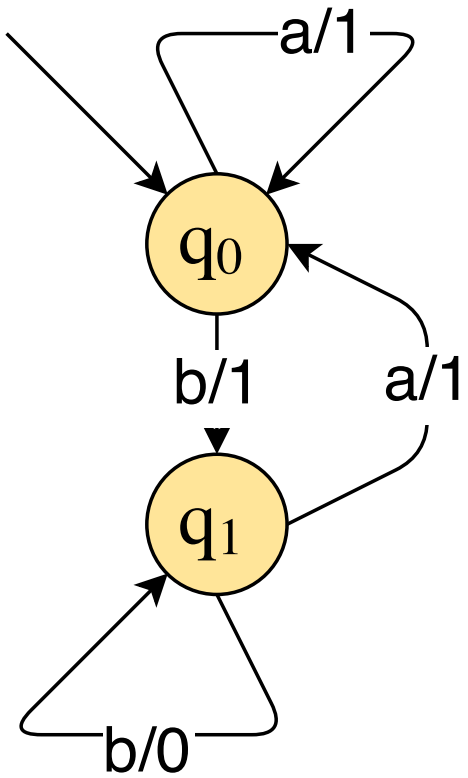
- velké stavové grafy,
- vnořené modely versus vnořené systémy,
- nedostatečná podpora.

Testování konečného automatu [HI98]

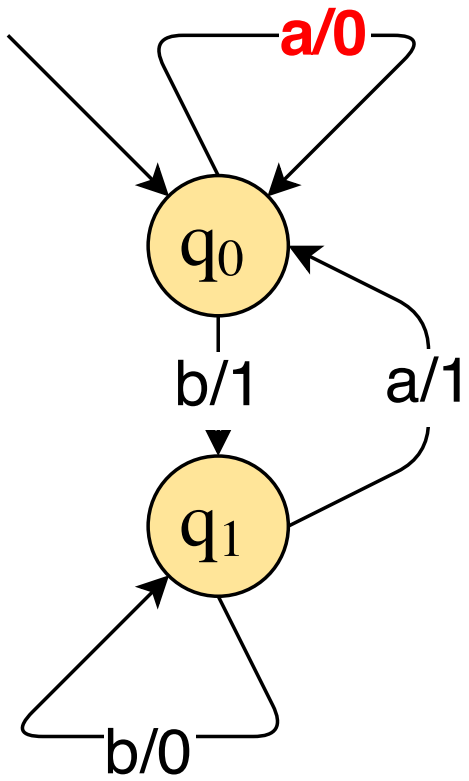
- založeno na izomorfismu konečných automatů,
- $\mathcal{A} = (Input, Q, \mathbf{F}, q_0)$
- $\mathcal{A}' = (Input, Q', \mathbf{F}', {q_0}')$
- $g : \mathcal{A} \to \mathcal{A}'$
- $g : Q \to Q'$
  1. $g(q_0) = {q_0}'$
  2. $\forall q \in Q, input \in Input,$
     $g(\mathbf{F}(q, input)) = \mathbf{F}'(g(q),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



Model chyb I [Mat13]

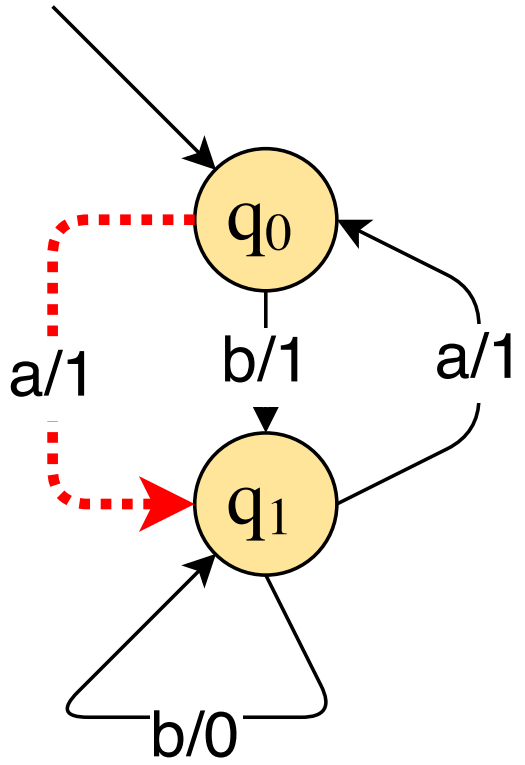


Návrh bez chyby



Operační chyba

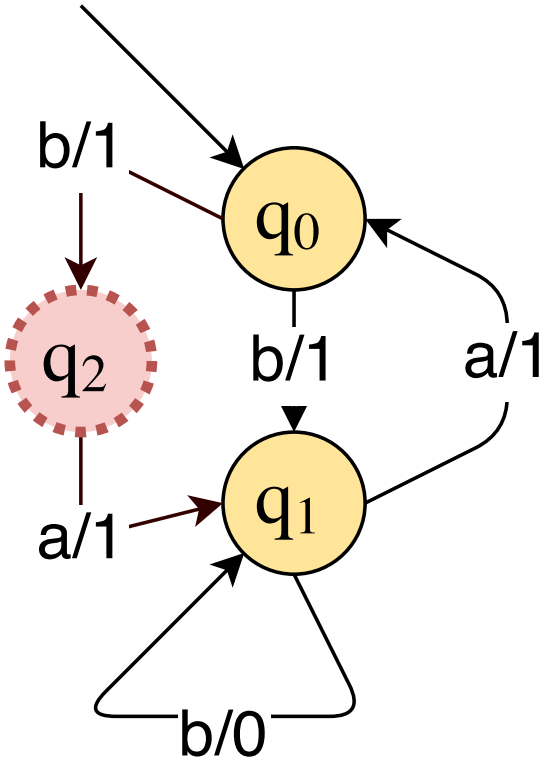
- chybný výstup
- chybějící výstup



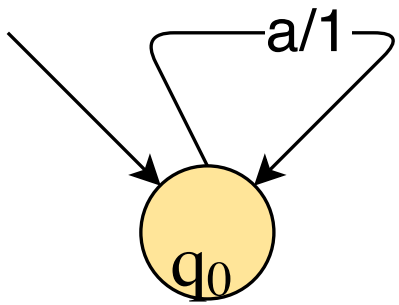
Chyba přechodu

- do jiného stavu
- do nového stavu

Model chyb II [Mat13]



Chyba extra stavu



Chyba chybějícího stavu

Konstrukce množiny testů [HI98, Cho78]

Chowova W metoda

- 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 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*.
  - $sequence \in T = L \bullet (Input^1 \cup \{<>\})$

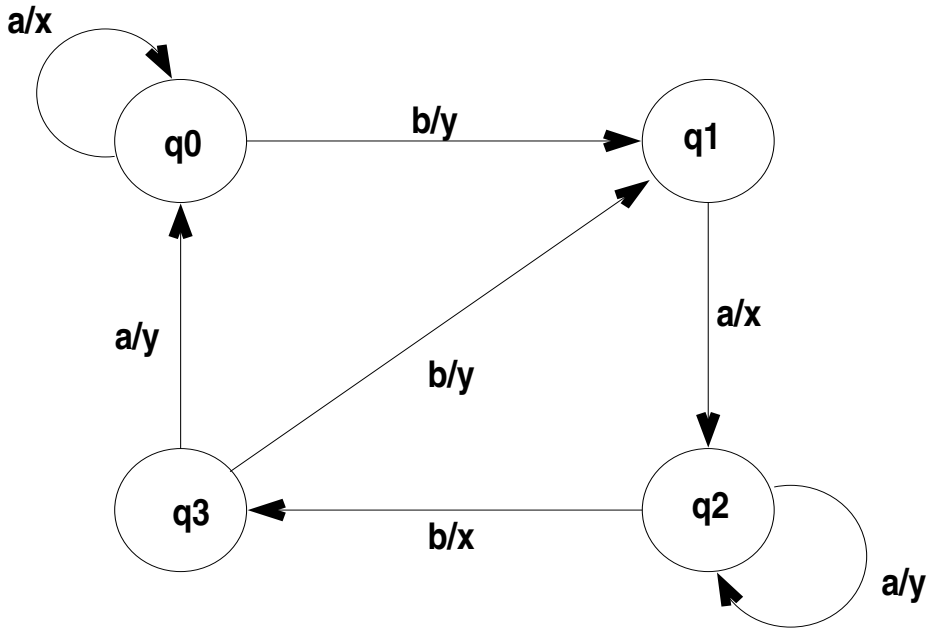
Generování množiny testů [HI98, Cho78]

- O kolik je v implementaci více testů než ve specifikaci? (k)
- $Z = Input^k \bullet W \cup Input^{k-1} \bullet W \cup \cdots \cup Input^1 \bullet W \cup W$
  - 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 zajisťuje,
  - 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 [HI98]



- $Input = \{a, b\}$
- $L = \{<>, b, b::a, b::a::b\}$, $<>$ . . . nulový vstup
- $T = \{<>, a, b, b::a, b::b, b::a::a, b::a::b, b::a::b::a, b::a::b::b\}$
- $W = \{a, b\}$ [Chy84], 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 [HI98]

$$
\begin{aligned}
T \bullet Z = \\
= \quad & \{<>, a, b, b::a, b::b, b::a::a, b::a::b, b::a::b::a, b::a::b::b\} \\
& \bullet \{a, b, a::a, a::b, b::a, b::b\} \\
= \quad & \{a, b, a::a, a::b, b::a, b::b, \\
& a::a, a::b, a::a::a, a::a::b, a::b::a, a::b::b, \\
& b::a, b::b, b::a::a, b::a::b, b::b::a, b::b::b, \\
& b::a::a, b::a::b, b::a::a::a, b::a::a::b, b::a::b::a, b::a::b::b, \\
& b::b::a, b::b::b, b::b::a::a, b::b::a::b, b::b::b::a, b::b::b::b, \\
& b::a::a::a, b::a::a::b, b::a::a::a::a, b::a::a::a::b, b::a::a::b::a, b::a::a::b::b, \\
& b::a::b::a, b::a::b::b, b::a::b::a::a, b::a::b::a::b, b::a::b::b::a, b::a::b::b::b, \\
& b::a::b::a::a, b::a::b::a::b, b::a::b::a::a::a, \\
& b::a::b::a::a::b, b::a::b::a::b::a, b::a::b::a::b::b, \\
& b::a::b::b::a, b::a::b::b::b, b::a::b::b::a::a, \\
& b::a::b::b::a::b, b::a::b::b::b::a, b::a::b::b::b::b\} \\
= \quad & \ldots \text{simplification}
\end{aligned}
$$

Aplikace [Bei95]

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

Mealyho automat [Mea55, Mat13]

Definition 1 (Mealyho automat s konečným počtem stavů je)

- 6-tice $M(X, Y, Q, q_0, \delta, \lambda)$:
- X je konečná množina vstupních symbolů (vstupní abeceda),
- Y je konečná množina výstupních symbolů (výstupní abeceda),
- Q je konečná množina stavů,
- $q_0 \in Q$ je počáteční stav,
- $D \subseteq Q \times X$ je specifikační doména,
- $\delta : D \to Q$ je přechodová funkce,
- $\lambda : D \to Y$ je výstupní funkce.

- Jestliže $D = Q \times X$, potom M je **úplný** Mealyho automat [SP10].
- Řetězec $\alpha = x_1 \dots x_k, \alpha \in I^*$ je **definovaná vstupní sekvence** pro stav $q \in Q$, jestliže existují $q_1, \dots, q_{k+1}$, kde $q_1 = q$ takové, že $(q_i, x_i) \in D$ a $\delta(q_i, x_i) = q_{i+1}$ pro všechna $1 \leq i \leq k$.

Minimalita automatu [SP10, Mat13]

Dán Mealyho automat $M(X, Y, Q, q_0, \delta, \lambda)$ s konečným počtem stavu.

- Rozšíření přechodové a výstupní funkce aplikovanou na vstupní symbol x na definované vstupní sekvence α, zahrnující prázdnou sekvenci ϵ:
  - pro $q \in Q$, $\delta(q, \epsilon) = q$ a $\lambda(q, \epsilon) = \epsilon$
  - $\delta(q, \alpha x) = \delta(\delta(q, \alpha), x)$
  - $\lambda(q, \alpha x) = \lambda(\delta(q, \alpha), x)$
- $\Omega(q)$ je množina všech definovaných vstupních sekvencí pro stav $q \in Q$.
- Dva stavy $q, q' \in Q$ jsou **rozlišitelné**, jestliže existuje $\gamma \in \Omega(q) \cap \Omega(q')$ takové, že $\lambda(q, \gamma) \neq \lambda(q', \gamma)$. Pak říkáme, že γ **rozlišuje** stavy q a q'.
- Dva stavy $q_1, q_2 \in Q; q_1 \neq q_2$ jsou **stavově ekvivalentní**, jestliže po aplikaci jakékoliv vstupní sekvence vedou do stejných nebo ekvivalentních stavů.
- M je **minimální**, jestliže žádné jeho dva stavy nejsou ekvivalentní [Ner58, Gil60].

C-ekvivalence stavů [SP10, Mat13]

Dán Mealyho automat $M(X, Y, Q, q_0, \delta, \lambda)$ s konečným počtem stavu.

- Nechť je dána množina $C \subseteq \Omega(q) \cap \Omega(q')$.
- Stavy $q_1, q_2 \in Q$ jsou C-**ekvivalentní**, jestliže $\lambda(q, \gamma) = \lambda(q', \gamma)$ pro všechny $\gamma \in C$.

Dva automaty $M_1(X, Y, Q_1, q_0^1, \delta_1, \lambda_1)$ a $M_2(X, Y, Q_2, q_0^2, \delta_2, \lambda_2)$ jsou **ekvivalentní**, jestliže

1. pro každý stav $q \in M_1$ existuje $q' \in M_2$ takový, že q a q' jsou ekvivalentní a
2. pro každý stav $q \in M_2$ existuje $q' \in M_1$ takový, že q a q' jsou ekvivalentní.

k-**ekvivalence**

- Nechť $M_1(X, Y, Q_1, q_0^1, \delta_1, \lambda_1)$ a $M_2(X, Y, Q_2, q_0^2, \delta_2, \lambda_2)$ jsou dva automaty.
- Stavy $q_i \in Q_1$ a $q_j \in Q_2$ se považují za k-**ekvivalentní**, jestliže po aplikování jakékoliv vstupní sekvence délky k jsou produkovány identické výstupní sekvence.

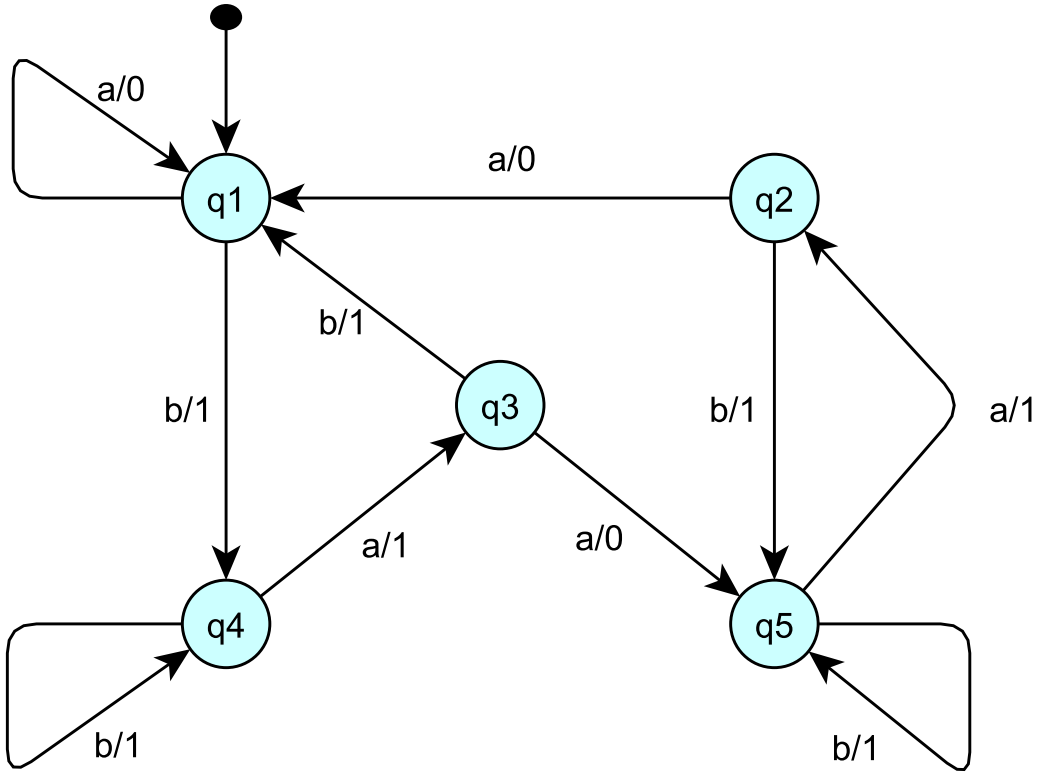
Charakterizační množina W [SP10, Mat13]

Nechť Mealyho automat $M(X, Y, Q, q_0, \delta, O)$ s konečným počtem stavu je minimální a úplný.

- W je konečná množina vstupních sekvencí, která rozliší jakýkoliv pár stavů $q_i, q_j \in Q$.
- Každá vstupní sekvence $\gamma \in W$ má konečnou délku.
- Pro každé dva stavy $q_i, q_j \in Q$ množina W obsahuje (alespoň jednu) vstupní sekvenci γ takovou, že

$$\lambda(q_i, \gamma) \neq \lambda(q_j, \gamma)$$

Příklad charakterizační množiny [HI98]



- $Input = \{a, b\}$
- $W = \{baaa, aa, aaa\}$
- $\lambda(q_1, baaa) = 1101$
- $\lambda(q_2, baaa) = 1100$
- $\lambda(q_1, baaa) \neq \lambda(q_2, baaa) \implies baaa$ rozlišuje stavy q_1 a q_2

k-ekvivalentní rozklad stavů Q [Mat13]

- k-ekvivalentní rozklad stavů Q označovaný jako P_k je soubor n konečných množin $\Sigma_{k,1}, \Sigma_{k,2}, \ldots, \Sigma_{k,n}$ takových

$$\cup_{i=1}^{n} \Sigma_{k,i} = Q$$

- Stavy v $\Sigma_{k,i}$ jsou k-ekvivalentní.
- Jestliže $q_{\ell_1} \in \Sigma_{k,j}$ a $q_{\ell_2} \in \Sigma_{k,j}$ pro $i \neq j$, potom q_{ℓ_1} a q_{ℓ_2} jsou k-rozlišitelné.

Konstrukce W množiny [Mat13]

Postup

1. Vytvoření sekvenci k-ekvivalentních rozklad stavů Q označenou jako $P_1, P_2, \ldots, P_m, m > 0$
2. Prohledání k-ekvivalentních rozkladů v opačném pořadí se současnou kontrukcí rozlišujících sekvencí pro každou dvojici stavů.

- Je garantována konvergence postupu.
- Po skončení postupu každá třída $\Sigma_{K,j}$ konečného rozkladu P_K definuje třídu ekvivalentních stavů (typicky 1).

Neformálně:

- nejprve se zjistí, co lze rozlišit v jednom kroku
- po té ve dvou krocích
- atd.

Konstrukce W množiny [Mat13]

Tabulární reprezentace M.

0-ekvivalenční rozklad $P_0 = \{\Sigma_1 = \{q_1, q_2, q_3, q_4, q_5\}\}$

Současný stav	**Výstup**		**Následující stav**	
	a	b	a	b
q_1	0	1	q_1	q_4
q_2	0	1	q_1	q_5
q_3	0	1	q_5	q_1
q_4	1	1	q_3	q_4
q_5	1	1	q_2	q_5

Konstrukce 1-ekvivalenční rozklad P_1 [Mat13]

1-ekvivalenční rozklad $P_1 = \{\Sigma_1 = \{q_1, q_2, q_3\}, \Sigma_2 = \{q_4, q_5\}\}$.

Σ	**Současný stav**	**Výstup**		**Následující stav**	
		a	b	a	b
1	q_1	0	1	q_1	q_4
	q_2	0	1	q_1	q_5
	q_3	0	1	q_5	q_1
2	q_4	1	1	q_3	q_4
	q_5	1	1	q_2	q_5

Konstrukce 2-ekvivalenční rozklad: přepis P_1 [Mat13]

Přepis P_1, stav q_i je nahrazen $q_{i,j}$, přičemž $q_i \in \Sigma_j$.

Σ	**Současný stav**	**Následující stav**	
		a	b
1	q_1	$q_{1,1}$	$q_{4,2}$
	q_2	$q_{1,1}$	$q_{5,2}$
	q_3	$q_{5,2}$	$q_{1,1}$
2	q_4	$q_{3,1}$	$q_{4,2}$
	q_5	$q_{2,1}$	$q_{5,2}$

Konstrukce 2-ekvivalenční rozklad: konstrukce P_2 [Mat13]

Konstrukce P_2. Rozdělení $\Sigma_{1,j}$ podle skupin příštích stavů.

Σ	**Současný stav**	**Následující stav**	
		a	b
1	q_1	$q_{1,1}$	$q_{4,3}$
	q_2	$q_{1,1}$	$q_{5,3}$
2	q_3	$q_{5,3}$	$q_{1,1}$
3	q_4	$q_{3,2}$	$q_{4,3}$
	q_5	$q_{2,1}$	$q_{5,3}$

Konstrukce 3-ekvivalenční rozklad: konstrukce P_3 [Mat13]

Konstrukce P_3. Rozdělení $\Sigma_{2,j}$ podle skupin příštích stavů.

Σ	**Současný stav**	**Následující stav**	
		a	b
1	q_1	$q_{1,1}$	$q_{4,3}$
	q_2	$q_{1,1}$	$q_{5,4}$
2	q_3	$q_{5,4}$	$q_{1,1}$
3	q_4	$q_{3,2}$	$q_{4,3}$
4	q_5	$q_{2,1}$	$q_{5,4}$

Konstrukce 4-ekvivalenční rozklad: konstrukce P_4 [Mat13]

Konstrukce P_4. Rozdělení $\Sigma_{3,j}$ podle skupin příštích stavů.

Σ	**Současný stav**	**Následující stav**	
		a	b
1	q_1	$q_{1,1}$	$q_{4,4}$
2	q_2	$q_{1,1}$	$q_{5,5}$
3	q_3	$q_{5,5}$	$q_{1,1}$
4	q_4	$q_{3,3}$	$q_{4,4}$
5	q_5	$q_{2,2}$	$q_{5,5}$

Nalezení rozlišujících sekvencí: příklad [Mat13]

1. Nalezněme rozlišující sekvenci stavů q_1 a q_2.
2. Inicializace rozlišující sekvence: $z = \epsilon$.
3. Najdi tabulky P_i a P_{i+1} takové, že (q_1, q_2) jsou ve stejné skupině v P_i a v různých skupinách v P_{i+1}:
   - dostaneme P_3 a P_4.
4. Nalezni vstupní symbol rozlišující q_1 a q_2 v tabulce P_3
   - Rozlišujícím symbolem je b.
   - Prodluž rozlišující sekvenci: $z := z.b = \epsilon.b = b$.
5. Nalezni příští stavy stavů q_1 a q_2 po aplikaci symbolu b
   - dostaneme q_4 a q_5.
6. Najdi tabulky P_i a P_{i+1} takové, že (q_4, q_5) jsou ve stejné skupině v P_i a v různých skupinách v P_{i+1}:
   - dostaneme P_2 a P_3.
7. $(q_4, q_5) \rightarrow P_2, P_3 \rightarrow a \rightarrow z = ba$
8. $(q_3, q_2) \rightarrow P_1, P_2 \rightarrow a \rightarrow z = baa$
9. $(q_1, q_5) \rightarrow P_0, P_1 \rightarrow a \rightarrow z = baaa$
10. Opakuj pro každý pár (q_i, q_j): $W = \{a, aa, aaa, baaa\}$

Literatura I

Boris Beizer.

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

T.S. Chow.

Testing software design modeled by finite-state machines.
IEEE Transactions on Software Engineering, SE-4(3):178–187, May 1978.

Michal Chytil.

Automaty a gramatiky.
SNTL Praha, 1984.

A. Gill.

Characterizing experiments for finite-memory binary automata.
IRE Transactions on Electronic Computers, EC-9(4):469–471, Dec 1960.

Mike Holcombe and Florentin Ipate.

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

Aditya P. Mathur.

Foundations of software testing 2e, slides, 2013.

George H. Mealy.

A method for synthesizing sequential circuits.
Bell System Technical Journal, The, 34(5):1045–1079, Sept 1955.

Literatura II

A. Nerode.

Linear automaton transformations.
Proc. Amer. Math. Soc., 9:541–544, 1958.

A. Simao and A. Petrenko.

Checking completeness of tests for finite state machines.
IEEE Transactions on Computers, 59(8):1023–1032, Aug 2010.