1.3. Systémový přístup

Se slůvkem **systém** se lze setkat v mnoha různých kontextech a slovních spojeních. Všude signalizuje existenci nějakých vazeb mezi prvky, řádu či pořádku. Nejblíže obecné intuitivní představě je již klasická definice Löfgrena, podle něhož **je systémem popis uspořádání entity (objektu) a opakem systému je chaos**. Systémovým přístupem pak nazýváme úhel pohledu, který se snaží vymezit pro daný účel podstatné rysy, příznaky, vlastnosti či jiné prvky zkoumané entity a uvést je do vzájemných souvislostí. Popis, který tím vznikne, je **systémovým modelem** entity, nebo také **systémem** definovaným na entitě.

Zkoumáním systémových modelů se zabývá věda zvaná **teorie systémů**. Nepřeberné množství jejich variant a typů si účelově člení do základních kategorií vymezených podle nejrůznějších hledisek: např. podle stupně teoretické abstrakce[2], podle míry idealizace[3], podle stupně proměnlivosti popisu[4], podle charakteru oborů hodnot proměnných[5], podle míry neurčitosti popisu[6] apod.

Uvedený přehled základních možností klasifikace systémů zdaleka není vyčerpávající. Uvádíme jej zde proto, abychom objasnili techniku tvorby jemnější účelové kategorizace systémů. Kombinováním základních možností klasifikace lze totiž vytvářet **kategorie systémů zadaných vlastností** a vhodným způsobem tak zužovat obsah pojmu *systém*. Tak např. pojem *uzavřený dynamický systém diskrétní v čase a v prostoru* označuje subkategorii systémů, které nekomunikují s okolím, všechny jejich charakteristiky (včetně času) nabývají izolovaných hodnot, přičemž systém se v čase mění (vyvíjí) přetržitě (tj. po skocích) a to jen na základě působení vnitřních procesů.

Účelová kategorizace systémů (zužování obsahu obecného pojmu *systém*) umožňuje vytvářet různě silné teoretické a technické prostředky „šité na míru" jak teoretickým, tak i praktickým problémům modelování entit. Obecně platí, že čím silnější (obecnější) je prostředek, tím je co do popisu stručnější a snáze se s ním matematicky pracuje. V tom je

---

[2] Nejníže stojí systémy s popisy komunikovatelnými v přirozeném jazyce, vyšší stupeň abstrakce mají popisy jazykem geometrie (grafem), ještě vyšší stupeň abstrakce je vlastní popisu jazykem aplikované matematiky a nejvyšší stupeň abstrakce mají popisy jazykem abstraktní matematiky.

[3] Zde nejvýše (nejvyšší míra idealizace) budou stát izolované (uzavřené) systémy nekomunikující s okolím, níže polouzavřené (jednosměrná komunikace s okolím) a nejníže otevřené systémy (obousměrná komunikace s okolím). Při tomto druhu klasifikace může ovšem také záležet na typu idealizace vlastností (předpoklad linearity apod.).

[4] Parametrem, hrajícím významnou roli při zkoumání systémů je čas, jehož formální reprezentací zpravidla bývá rostoucí celočíselná posloupnost nebo interval reálných čísel. Nebereme-li čas v úvahu nebo nedochází-li v jeho průběhu k postřehnutelným kvalitativním či kvantitativním změnám v popisu systému, mluvíme o *statickém systému*. Systémy, u nichž v průběhu času k postřehnutelným změnám dochází, označujeme přívlastkem *dynamické*.

[5] To, že čas je v popisu systému reprezentován posloupností, resp. kontinuem, vyjadřujeme přívlastkem *diskrétní v čase*, resp. *spojitý v čase*. Podobně můžeme odlišit popisy systémů, v nichž obory hodnot všech ostatních charakteristik popisu jsou spočetnými množinami (*diskrétnost v prostoru*) od popisů, v nichž obor hodnot alespoň jedné charakteristiky je kontinuem (*spojitost v prostoru*). Diskrétním systémům, jejichž obory hodnot (s výjimkou času) jsou konečné, říkáme *konečné* systémy.

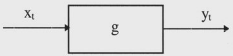
[6] Určuje-li současný popis svou nejbližší změnu jednoznačně, říkáme, že systém je *deterministický* (nulová neurčitost popisu). V opačném případě je *nedeterministický*. Je-li v případě nedeterministického systému dána pravděpodobnostní míra nad oborem možných příštích změn (snižující neurčitost popisu), říkáme, že systém je *stochastický*.

výhoda silnějších prostředků. Naproti tomu slabší prostředky bývají snáze realizovatelné, s jejich realizací bývá spojeno méně problémů a o to v praxi často zejména jde.

2. Fundamentální rozvoj základní představy o automatu

2.1. Abstraktní stroj

Vědní obory bývá zvykem dělit na aplikované (zkoumající přímo realitu) a na teoretické (zkoumající modely reality). Teorie automatů, s jejímiž základy se v tomto kurzu seznámíme, patří k těm druhým. Středem jejího zájmu je podtřída deterministických dynamických systémů - tzv. abstraktní stroje. **Abstraktním strojem** se rozumí jakýkoli deterministický mechanismus, převádějící časově proměnné podněty, které na něj přicházejí z jeho okolí, na reakce, kterými mechanismus na podněty odpovídá a na své okolí působí. Schematicky je abstraktní stroj znázorněn na následujícím obrázku.



Obr. 2.1: Abstraktní stroj

V něm obdélník g znázorňuje transformační mechanismus, x_t je vektor podnětů v okamžiku t, y_t je vektor reakcí na podnět x_t. Slůvko *abstraktní* ve spojení abstraktní stroj znamená, že odhlížíme od všech praktických fyzikálních a prostoročasových aspektů reality. Zejména nás nezajímá frekvence vstupních podnětů (respektive délka časového intervalu mezi jednotlivými podněty), nebereme v úvahu rozměry objektu ani konečnou rychlost šíření signálů apod. Zaznamenáváme jen odpovídající si hodnoty podnětů a reakcí a zkoumáme mechanismus g, který je uvádí do souvislostí.

Mají-li všechny prvky této představy diskrétní povahu, jedná se o model typu **diskrétní abstraktní stroj**. Pak index t vektorů x_t a y_t neudává spojitý čas, ale nabývá jen celočíselných hodnot udávajících *pořadí* podnětu a reakce - měří tzv. *diskrétní čas*. Jsou-li navíc s výjimkou diskrétního času obory všech proměnných veličin tohoto modelu konečné, a i mechanismus g může být zadán konečnou množinou transformačních pravidel, mluvíme o **konečném abstraktním stroji**. Veškerá informace, kterou může mít transformační mechanismus g v okamžiku t k dispozici pro volbu reakce y_t, sestává z hodnoty x_t a z nekonečné posloupnosti dvojic předchozích hodnot . . . ,$(x_{t-2}, y_{t-2}),(x_{t-1}, y_{t-1})$. Z ní si ale v důsledku své konečné podstaty dokáže „zapamatovat" a pro transformaci využít jen její konečnou část. Délku zapamatovatelné sekvence nazýváme **hloubkou paměti** stroje. Je-li hloubka paměti nenulová, mluvíme o **konečném sekvenčním abstraktním stroji**. V opačném případě se jedná o **konečný kombinační abstraktní stroj**, protože reakce y_t je dána jen kombinací hodnot složek vektoru x_t a na předchozích dvojicích podnět-reakce nezávisí.

K tomu, aby byl model užitečným nástrojem zkoumání reality, je nezbytná existence fundamentálních (nikoli jen povrchních a formálních) analogií mezi procesy probíhajícími v modelu a procesy probíhajícími v modelované realitě. Jinak bychom mohli z modelu vyvodit důsledky, ke kterým v realitě nedochází. V následujícím článku této kapitoly se proto podrobně zabýváme fundamentálním rozvojem výše uvedené představy do podoby konečného automatu.

2.2. Konečný automat jako konečný sekvenční abstraktní stroj

V článku 2.1. jsme uvedli, v jakém směru je konečný abstraktní stroj zúžením abstraktního stroje schématicky znázorněného na Obr. 2.1. Obě vnější proměnné, proměnná x a proměnná y se mění po skocích a nabývají konečný počet |x|, resp. |y| izolovaných hodnot. Dvojici (x_t,y_t), kde x_t je hodnota vstupního podnětu v časovém okamžiku t a y_t je hodnota výstupní reakce na podnět x_t, nazveme **aktuálním elementárním chováním** v okamžiku t. Proměnná t je diskrétní čas, a nabývá celočíselných hodnot udávajících *pořadí* podnětu a reakce.

Z podstaty věci vyplývá, že v souladu s přirozeným během času aktuální hodnota proměnné t postupně roste, takže i vzdálená budoucnost se jednou změní na „prchavou" současnost a posléze v minulost. Budeme-li proto v dalším hovořit o *současnosti*, budeme mít na mysli konkrétní okamžik, tj. *aktuální* hodnotu T proměnné t (okamžik pozorování) a jí odpovídající aktuální elementární chování (x_T,y_T). Nižší, resp. vyšší hodnoty proměnné t (vzhledem k aktuální hodnotě T) a jim odpovídající aktuální elementární chování budou složkami aktuální *minulosti* resp. *budoucnosti*. V dalších úvahách budeme konečné a souvislé fragmenty minulosti či budoucnosti definovat začátky a konci příslušných sekvencí aktuálních elementárních chování, které uzavřeme do „špičatých" závorek: Např. $\langle(x_{T-4},y_{T-4}),(x_{T-1},y_{T-1})\rangle$ označuje čtyřčlennou sekvenci $(x_{T-4},y_{T-4}),(x_{T-3},y_{T-3}),(x_{T-2},y_{T-2}),(x_{T-1},y_{T-1})$ konce aktuální historie, a $\langle(x_{T-1},y_{T-1})\rangle$ označuje jednočlennou sekvenci (x_{T-1},y_{T-1}) téhož.

Předpokládejme, že náš stroj začínáme pozorovat v okamžiku t = 0, který také definuje výchozí *současnost*, přičemž stroj funguje již „od nepaměti" a bude po neomezenou dobu fungovat dál. Z obou stran nekonečnou posloupnost ${}_{-\infty}\{(x_t,y_t)\}^{\infty}$ aktuálních elementárních chování minulých i budoucích ve tvaru

$${}_{-\infty}\{(x_t,y_t)\}^{\infty} = \ldots ,(x_{-2},y_{-2}),(x_{-1},y_{-1}),(x_0,y_0),(x_1,y_1),(x_2,y_2), \ldots$$

nazveme **fázovou trajektorií** stroje. Okamžik t = 0 dělí fázovou trajektorií na dvě části: Na její již proběhlou (minulou) část

$${}_{-\infty}\{(x_t,y_t)\}^{-1} = \ldots ,(x_{-2},y_{-2}),(x_{-1},y_{-1}),$$

kterou nazveme **historií** a na očekávanou budoucí část

$${}_{0}\{(x_t,y_t)\}^{\infty} = (x_0,y_0),(x_1,y_1),(x_2,y_2), \ldots ,$$

kterou nazveme **evolucí**.

Z jedné strany je zřejmé, že počet různých historií, kterými náš stroj *mohl* projít, není shora omezen. Stejně tak není shora omezen počet *možných* evolucí. Z druhé strany je zřejmé, že stroj *prošel* jedinou historií a také *projde* jedinou evolucí. Jaká evoluce to bude, to jistě bude záviset hlavně na konkrétní posloupnosti ${}_{0}\{x_t\}^{\infty} = x_0,x_1,x_2, \ldots$ současného a budoucích vstupních podnětů. Dále to bude záviset i na tom, jak dalece historie ovlivňuje vnitřní mechanismus reakcí stroje; nebo, přesněji, jak dlouhou část historie (tj. jak dlouhou sekvenci aktuálních elementárních chování konce aktuální historie) si transformační mechanismus g dokáže uchovat či „zapamatovat".

Abychom snáze zodpověděli otázku - *„Jaká další minimální informace je zapotřebí k tomu, aby k zadané posloupnosti vstupních podnětů ${}_{0}\{x_t\}^{\infty}$ bylo možno jednoznačně predikovat evoluci ${}_{0}\{(x_t,y_t)\}^{\infty}$?"* - provedeme následující posloupnost úvah:

- Nejprve předpokládejme vnitřní mechanismus bez paměti, v němž se do okamžiku t neuchová ani informace o sekvenci $\langle(x_{t-1},y_{t-1})\rangle$. V takovém případě $y_t = g(x_t)$, a proto ${}_0\{(x_t,y_t)\}^\infty = {}_0\{(x_t,g(x_t))\}^\infty$. Vidíme, že je možná jen jediná evoluce, a k její predikci stačí pouhá znalost ${}_0\{x_t\}^\infty$.
- Předpokládejme, že se ve vnitřním mechanismu stroje do každého okamžiku t (a ne déle) uchová informace pouze o sekvenci $\langle(x_{t-1},y_{t-1})\rangle$, a tato informace „stejným způsobem" nezávislým na t (tj. mechanismus se nevyvíjí) ovlivňuje reakci y_t. Pak $y_t = g(x_t,x_{t-1},y_{t-1})$ a

$$ {}_0\{(x_t,y_t)\}^\infty = {}_0\{(x_t,\, g(x_t,x_{t-1},y_{t-1}))\}^\infty . $$

  Vidíme, že všechny historie, které se shodují v (x_{-1},y_{-1}), podnítí touž evoluci. Ovlivní totiž stejně reakci $y_0 = g(x_0,x_{-1},y_{-1})$ a budou se tím pádem shodovat i v (x_0,y_0). To stejně ovlivní reakci $y_1 = g(x_1,x_0,y_0)$ a povede ke shodě i v (x_1,y_1) atd. Z toho vyplývá, že všechny historie shodující se v (x_{-1},y_{-1}) budou mít shodné pokračování, tj. shodnou evoluci.

  Kolik různých evolucí při dané vstupní posloupnosti ${}_0\{x_t\}^\infty$ pak může nastat? Nejvýše tolik, kolik různých jednočlenných sekvencí $\langle(x_{-1},y_{-1})\rangle$ se v historii ${}_{-\infty}\{(x_t,y_t)\}^{-1}$ může vyskytnout. A těch nemůže být více než $|x| \cdot |y|$. Z hlediska evoluce se tak nekonečná množina všech možných historií rozkládá (podle dvojic (x_{-1},y_{-1}) v zapamatovatelné koncové sekvenci) do konečného počtu navzájem disjunktních tříd. K tomu, aby bylo možno jednoznačně predikovat evoluci ${}_0\{(x_t,y_t)\}^\infty$, postačí kromě znalosti ${}_0\{x_t\}^\infty$ znalost toho, do které třídy rozkladu spadá historie stroje.[1]
- Předpokládejme, že se v mechanismu stroje do každého okamžiku t (a ne déle) uchová informace o dvoučlenné sekvenci $\langle(x_{t-2},y_{t-2}),(x_{t-1},y_{t-1})\rangle$, a tato informace „stejným způsobem" nezávislým na t ovlivňuje reakci y_t. Pak

$$ y_t = g(x_t,x_{t-1},y_{t-1},x_{t-2},y_{t-2}) \text{ a } {}_0\{(x_t,y_t)\}^\infty = {}_0\{(x_t,\, g(x_t,x_{t-1},y_{t-1},x_{t-2},y_{t-2}))\}^\infty . $$

  Vidíme, že všechny historie, které se shodují v zapamatovatelné sekvenci $\langle(x_{-2},y_{-2}),(x_{-1},y_{-1})\rangle$, podnítí touž evoluci. Ovlivní totiž stejně reakci $y_0 = g(x_0,x_{-1},y_{-1},x_{-2},y_{-2})$ a budou se tím pádem shodovat i v (x_0,y_0). To stejně ovlivní reakci $y_1 = g(x_1,x_0,y_0,x_{-1},y_{-1})$ a povede ke shodě i v (x_1,y_1) atd. Z toho vyplývá, že všechny tyto historie budou mít shodné pokračování, tj. shodnou evoluci.

  Stejně jako v předchozím případě lze i zde dojít k závěru, že počet možných evolucí stroje (při dané ${}_0\{x_t\}^\infty$) nepřevýší nejvýše možný počet různých dvoučlenných sekvencí konce historie; tento počet je dán číslem $(|x| \cdot |y|)^2$. Z hlediska evoluce se tak nekonečná množina všech možných historií opět rozkládá do konečného počtu navzájem disjunktních tříd. Rozklad je jemnější než v předchozím případě a vznikl dalším rozkladem každé z $|x| \cdot |y|$ tříd předchozího rozkladu.

---

[1] Označme každou z tříd takto definovaného rozkladu nekonečné množiny všech možných historií zapamatovatelnou sekvencí, všem historiím dané třídy společnou. Pro aktuální historii ${}_{-\infty}\{(x_t,y_t)\}^{-1}$ systému pak platí: ${}_{-\infty}\{(x_t,y_t)\}^{-1} \in \langle(x_{-1},y_{-1})\rangle$. Příchod podnětu x_0 pak vyvolá reakci $y_0 = g(x_0,x_{-1},y_{-1})$. Tím vznikne nová aktuální historie ${}_{-\infty}\{(x_t,y_t)\}^0 \in \langle(x_0,y_0)\rangle$ a *současnost* se přesouvá do okamžiku t = 1. Příchod podnětu x_1 pak vyvolá reakci $y_1 = g(x_1,x_0,y_0)$. Vznikne aktuální historie ${}_{-\infty}\{(x_t,y_t)\}^1 \in \langle(x_1,y_1)\rangle$ a *současnost* se přesouvá do okamžiku t = 2. Opakováním těchto kroků postupně generujeme evoluci systému.

Závěry předchozích úvah lze matematickou indukcí zobecnit: Je-li konečný abstraktní stroj s pevným vnitřním mechanismem (tj. mechanismem neměnným v čase) „obdařen" **pamětí s hloubkou n** (n je délka zapamatovatelné sekvence aktuálních elementárních chování), pak k tomu, aby bylo možno jednoznačně predikovat evoluci $_0\{(x_t,y_t)\}^\infty$, postačí kromě znalosti $_0\{x_t\}^\infty$ znalost toho, ve které z nejvýše $(|x| \cdot |y|)^n$ tříd rozkladu nekonečné množiny všech možných historií (indukovaného zapamatovatelnou sekvencí $\langle(x_{-n},y_{-n}),(x_{-1},y_{-1})\rangle$) se stroj právě nachází.[2]

V terminologii teorie systémů je náš konečný abstraktní stroj speciálním deterministickým dynamickým systémem. A systémům s pamětí tohoto druhu se v teorii systémů říká **sekvenční systémy**. V teorii systémů je dále zvykem nazývat informaci, postačující k úplné predikci evoluce systému, **stavem**. Proto jsou i v našem konečném abstraktním stroji jednotlivé třídy výše zmíněného rozkladu nekonečné množiny všech možných historií nazývány stavy a rozklad sám pak **stavovým prostorem**. A jak jsme viděli, počet tříd rozkladu, tj. počet stavů, je konečný. Počet stavů stavového prostoru charakterizuje **rozsah** nebo-li **kapacitu paměti** stroje.[3] Formální popis konečného abstraktního stroje, umožňující úplnou predikci evoluce má po zavedení stavového prostoru tvar:

$$y_t = G(x_t, s_t), \quad s_{t+1} = F(x_t, s_t); \quad x_t \in X, \; y_t \in Y, \; s_t, s_{t+1} \in S.$$

X resp. Y je konečná množina vstupních podnětů resp. výstupních reakcí, neprázdná konečná množina S je stavovým prostorem stroje, indexová proměnná t udává diskrétní čas.

Při postupné konstrukci evoluce se provádí následující kroky: Nechť okamžikem *současnosti* je okamžik t = T. V něm se stroj nachází ve stavu s_T. Příchod podnětu x_T vyvolá reakci $y_T = G(x_T, s_T)$ a následnou změnu stavu s_T na s_{T+1} podle vztahu $s_{T+1} = F(x_T, s_T)$; okamžikem *současnosti* se stává okamžik t = T + 1.[4] Takto fungující konečné sekvenční abstraktní stroje se v teorii automatů nazývají **konečnými automaty**.[5]

---

[2] Předchozí poznámka pod čarou popisuje mechanismus změn s pamětí hloubky n = 1. Nahradíme-li v ní zapamatovatelné jednočlenné sekvence n-člennými zapamatovatelnými sekvencemi, pak po příslušném rozšíření funkce g dostaneme zobecnění této poznámky na mechanismus změn s pamětí hloubky n, kde n je libovolné kladné celé číslo.

[3] Vztah mezi hloubkou a kapacitou paměti je v konkrétních případech silně ovlivněn jak konkrétní závislostí reakce na podnětech, tak stupněm kontraktivity transformačních funkcí. V důsledku toho roste s hloubkou kapacita mnohem pomaleji než exponenciálně (jak nám vycházelo z našich silně zjednodušených úvah).

[4] Z poznámek 1 a 2 vyplývá, že u mechanismu s pamětí hloubky n $s_t = \langle(x_{t-n},y_{t-n}),(x_{t-1},y_{t-1})\rangle$ a $y_t = g(x_t,x_{t-1},y_{t-1},\ldots,x_{t-n},y_{t-n}) = G(x_t, s_t)$. Rovnice $s_{t+1} = F(x_t, s_t)$ popisuje přechod aktuální historie systému z třídy rozkladu $\langle(x_{t-n},y_{t-n}),(x_{t-1},y_{t-1})\rangle$ do třídy rozkladu $\langle(x_{t-n+1},y_{t-n+1}),(x_t,y_t)\rangle$.

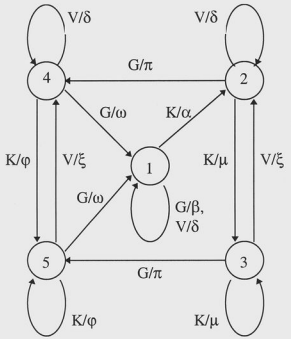
[5] Je-li $|S| = 1$, tj. existuje-li pouze jediný stav, rovnice $s_{t+1} = F(x_t, s_t)$ ztrácí smysl (přechází v tautologii, neboť $s_t = s_{t+1}$ pro všechna t) a $G(x_t, s_t) = g(x_t)$ - viz systém bez paměti. Automaty tohoto typu nereflektují historii a někdy jsou nazývány **kombinačními automaty**.

Cvičení k lekci 2.

Příklad 2.1: Stará teta Evženie jako konečný automat

Rozpoložení - stavy	**Podněty - stimuly**	**Reakce - odezvy**
1 - spí	K - kravál z ulice	α - procitne
2 - vlídná, laskavá	V - masakr na videu	β - chrápe
3 - podrážděná	G - vůně ginu	μ - zlobí se
4 - sentimentální (opilá)		δ - je potichu
5 - rozzuřená (opilá)		π - pije
		ξ - řehtá se
		φ - vzteká se
		ω - kolabuje

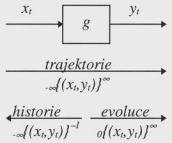
	změna stavu			reakce		
	K	V	G	K	V	G
1	2	1	1	α	δ	β
2	3	2	4	μ	δ	π
3	3	2	5	μ	ξ	π
4	5	4	1	φ	δ	ω
5	5	4	1	φ	ξ	ω



Obr. 2.2

Příklad 2.2: Hloubka paměti konečného automatu

Hledá se pravidlo g přiřazující ke vstupům výstupy. Jedinou informací je historie.



Obr. 2.3

Nemá-li pravidlo (diskrétní systém) paměť, pak $y_t = g(x_t)$. To lze zjistit z historie. Pokud se ukáže, že vztah mezi y a x není funkcí, ale relací, signalizuje to existenci paměti. Kromě vstupu může výstup ovlivňovat jen pořadí předchozích elementárních chování, jiná informace do systému nevstupuje. Hloubkou paměti je délka úseku historie, který ovlivňuje výstup. Hledáme relaci

$$x_{t-n} \times y_{t-n} \times x_{t-n+1} \times y_{t-n+1} \ldots x_{t-1} \times y_{t-1} \times x_t \times y_t$$

s minimálním n, která je funkcí

$$g\colon x_{t-n} \times y_{t-n} \times x_{t-n+1} \times y_{t-n+1} \ldots x_{t-1} \times y_{t-1} \times x_t \to y_t.$$

Toto minimální n je hloubkou paměti.

Příklady:

- Otevírání dveří výtahu (automat bez paměti) - v elementárním chování (x_t,y_t) platí $x_t = y_t$.
- Ovládání motoru výtahu (paměť hloubky 1). U výtahu pro tři podlaží:
  $g(1,3,0) = g(1,3,\uparrow) = g(1,2,\uparrow) = g(1,2,0) = g(1,2,\downarrow) = g(2,3,0) = g(2,3,\uparrow) = \downarrow$,
  $g(1,1,0) = g(1,1,\downarrow) = g(2,2,0) = g(2,2,\downarrow) = g(2,2,\uparrow) = g(3,3,0) = g(3,3,\uparrow) = 0$,
  $g(2,1,0) = g(2,1,\downarrow) = g(3,2,0) = g(3,2,\downarrow) = g(3,2,\uparrow) = g(3,1,0) = g(3,1,\downarrow) = \uparrow$.
- Tlačítkový spínač či vysouvání náplně versatilky (paměť hloubky 1).

Otázka 2.1: V čem je rozdíl mezi hloubkou a kapacitou paměti?

Úloha 2.1: Zjistěte hloubku paměti staré tety Evženie z příkladu 2.1.

3. Algebraická definice konečného automatu

Popis jednoho kroku chování konečného automatu jsme v závěru článku 2.2. uvedli ve formě, v jaké se vyskytuje ve starších učebnicích. Figuruje v něm pět subjektů:

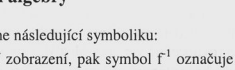
- množina X vstupních podnětů,
- množina Y výstupních reakcí,
- množina S stavů,
- přechodová funkce F,
- výstupní funkce G.

Jde tedy o pětici (S, X, Y, F, G). V moderních učebnicích využívajících aparát abstraktní algebry se tato pětice zapisuje symboly (S, I, O, δ, β). Na tuto symboliku přejdeme i my. Nezáleží na symbolech, ale na jejich pořadí, které určuje jejich význam.

3.1. Elementy abstraktní algebry

Zde i v dalších lekcích užíváme následující symboliku:

- Je-li f: $X \to Y$ bijektivní zobrazení, pak symbol f^{-1} označuje **inverzní zobrazení** k f, tj. zobrazení f^{-1}: $Y \to X$ takové, v němž pro všechna $x \in X$ a $y \in Y$ platí:
  $(f^{-1}(y) = x) \Leftrightarrow (f(x) = y)$.
- Jsou-li f: $X \to Y$ a g: $Y \to Z$ zobrazení, pak symbol $g \circ f$ označuje **kompozici zobrazení**, tj. zobrazení $g \circ f$: $X \to Z$ ve tvaru $[g \circ f](x) = g(f(x))$, vzniklé postupnou aplikací zobrazení f a g.



Obr. 3.1

- Symbolem π_i, resp. $\pi_{i,j}$ označujeme **projekci**, tj. zobrazení π_i: $X_1 \times X_2 \times \ldots \times X_n \to X_i$, resp. $\pi_{i,j}$: $X_1 \times X_2 \times \ldots \times X_n \to X_i \times X_j$ ve tvaru $\pi_i(x_1, x_2, \ldots, x_n) = x_i$, resp. $\pi_{i,j}(x_1, x_2, \ldots, x_n) = (x_i, x_j)$.
- Jsou-li f: $X \to Y$ a g: $X \to Z$ zobrazení, pak symbol g,f označuje **vektorové zobrazení**, tj. zobrazení g,f: $X \to Z \times Y$ ve tvaru $[g,f](x) = (g(x), f(x))$.
- Jsou-li f: $X \to Y$ a g: $U \to V$ zobrazení, pak symbol $f \times g$ označuje **paralelní zobrazení**, tj. zobrazení $f \times g$: $X \times U \to Y \times V$ ve tvaru $[f \times g](x,u) = (f(x), g(u))$. Speciálně symbol f^k označuje paralelní zobrazení f^k: $X^k \to Y^k$ ve tvaru $f^k(x_1, x_2, \ldots, x_k) = (f(x_1), f(x_2), \ldots, f(x_k))$.
- Symbol $\underline{1}$ označuje **identitu** na příslušné množině. Např. zobrazení $\underline{1}$: $X \to X$ ve tvaru $\underline{1}(x) = x$ je identitou na množině X.

3.2. Typy automatů

Ve stejném smyslu jako v předchozí lekci zavedeme proměnnou t, která bude udávat diskrétní čas. Pak můžeme definovat následující typy konečných automatů:

- **Stavový automat** (nebo také „automat bez výstupu") je uspořádaná trojice SA = (S, I, δ), kde S je neprázdná konečná množina stavů (stavový prostor), I je konečná neprázdná

množina hodnot vstupní diskrétní proměnné (vstupní abeceda) a δ: S×I → S je přechodová funkce ve tvaru $\delta(s_t, i_t) = s_{t+1}$.

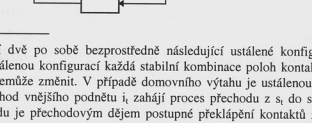
- **Mealyho automat** je uspořádaná pětice ME = (S, I, O, δ, β), kde (S, I, δ) je stavový automat, O je konečná neprázdná množina hodnot výstupní diskrétní proměnné (výstupní abeceda) a β: S×I → O je výstupní funkce ve tvaru $\beta(s_t, i_t) = o_t$ nebo $\beta(\delta(s_t, i_t), i_t) = \beta(s_{t+1}, i_t) = o_t$.

- **Mooreův automat** je speciálním případem Mealyho automatu, který nastává, když existuje tzv. „značkovací funkce" λ: S → O taková, že s její pomocí lze výstupní funkci β vyjádřit ve tvaru $\beta = \lambda \circ \pi_1$, tj. $\beta(s_t, i_t) = \lambda(s_t) = o_t$, nebo ve tvaru $\beta = \lambda \circ \delta$, tj. $\beta(s_t, i_t) = \lambda(\delta(s_t, i_t)) = \lambda(s_{t+1}) = o_t$. V prvním případě jde o Mooreův automat **1. druhu**, ve druhém případě o Mooreův automat **2. druhu**. V obou případech lze Mooreův automat vyjádřit jako pětici MO = (S, I, O, δ, λ), kde S, I, O, δ jsou definovány shodně jako v Mealyho automatu.

Vidíme, že ME je rozšířením SA o výstupní abecedu O a výstupní funkci β. Do funkce β lze zavést časovou závislost uvedenými dvěma způsoby, čímž dostáváme dvě z hlediska funkčních možností naprosto rovnocenné varianty Mealyho automatu. První z nich se shoduje s formálním popisem sekvenčního systému uvedeným v závěru článku 2.2.; přímá souvislost δ s F a β s G je zřejmá. Výstupní funkce λ Mooreova automatu je specifická tím, že je definována pouze na stavovém prostoru S. Oba druhy Mooreova automatu se opět liší pouze zavedením časové závislosti do funkce λ a také jsou funkčně plně rovnocenné.

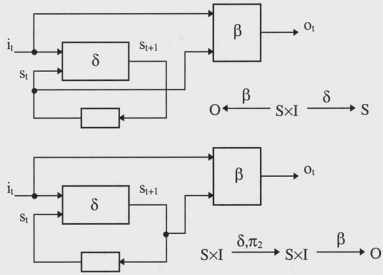
3.3. Funkční schémata a jejich algebraické diagramy

Formální rozdíly mezi jednotlivými typy automatů jsou dobře patrny z jejich funkčních schémat a algebraických diagramů zakreslených v následujících obrázcích. Z algebraických diagramů i z funkčních schémat je dobře patrna časová závislost. Označené bloky funkčních schémat realizují příslušné přechodové či výstupní funkce. Neoznačený blok ve zpětné vazbě bloku δ symbolicky zpožďuje vstupující signál o jednu časovou jednotku (o jeden takt).[1]

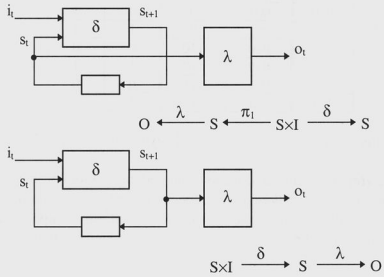


[1] Stavy s_t a s_{t+1} označují dvě po sobě bezprostředně následující ustálené konfigurace systému. V případě reléového obvodu je ustálenou konfigurací každá stabilní kombinace poloh kontaktů, která se samovolně (tj. bez vnějšího podnětu) nemůže změnit. V případě domovního výtahu je ustálenou konfigurací stání výtahu v příslušném podlaží. Příchod vnějšího podnětu i_t zahájí proces přechodu z s_t do s_{t+1}, tzv. přechodový děj. V případě reléového obvodu je přechodovým dějem postupné překlápění kontaktů z jedné stabilní kombinace poloh do druhé, v případě domovního výtahu jde o pohyb klece výtahu z výchozího do cílového podlaží. Náš model zjednodušuje realitu tím, že zohledňuje jen ustálené konfigurace a existenci přechodových dějů nebere v úvahu. V praxi totiž přechodový děj trvá jistou dobu a další vnější podnět, přišlý v době přechodového děje je buď zcela ignorován (některé typy výtahů, u nichž výtah na stisk tlačítka během pohybu nereaguje) nebo vyvolá nejednoznačnou reakci (složitější reléové obvody). Model předpokládá, že časová odlehlost vnějších podnětů $i \in I$ je delší než doba trvání nejdelšího přechodového děje, takže nová ustálená konfigurace se nastolí ještě před příchodem dalšího vnějšího podnětu. Tento předpoklad je ve funkčním schématu znázorněn zpožďovacím blokem ve zpětné vazbě, zajišťujícím i formální korektnost definice přechodové funkce δ.

Obr. 3.2: Funkční schéma stavového automatu a jeho algebraický diagram



Obr. 3.3: Funkční schémata a algebraické diagramy dvou variant Mealyho automatu



Obr. 3.4: Funkční schéma a algebraický diagram Mooreova automatu 1. a 2. druhu

I když je Mealyho automat obecnější než jeho Mooreův protějšek, jsou oba typy funkčně rovnocenné. Ke každému Mealyho automatu lze totiž sestrojit funkčně ekvivalentní Mooreův automat, ovšem s větším počtem stavů. Za jednodušší výstupní funkci tak Mooreův automat „platí" složitější přechodovou funkcí. Důvody, proč se v obou případech zavádí do výstupní funkce časová závislost dvojím způsobem, jsou ryze praktické a souvisí s vlastnostmi modelovaného objektu (viz různé modely výtahu v cvičném příkladu 3.1).

3.4. Formy prezentace konečného automatu

Trojici SA = (S, I, δ) a pětice ME = (S, I, O, δ, β) či MO = (S, I, O, δ, λ), je možno zadat čtyřmi formálně různými, i když z věcného hlediska rovnocennými způsoby: výčtem, tabulkou přechodové a výstupní funkce, orientovaným grafem, stromem. Vysvětlíme si je na následujícím příkladu automatového modelu tříbitového posuvného registru

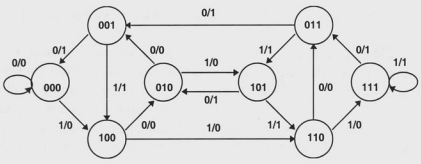
(paměť typu FIFO).

Tříbitový posuvný registr může ve svých třech paměťových buňkách schraňovat $2^3 = 8$ různých binárních čísel, a může se tak nacházet v jednom z osmi stavů. Nachází-li se ve stavu $i_1i_2i_3$, pak vstupní binární signál i_0 změní stav na $i_0i_1i_2$ (posun o jedno místo doprava) a hodnota i_3 se objeví na výstupu. V prvních třech taktech se tak na výstupu postupně objeví počáteční informace uložená v paměti a dále pak vždy o tři takty opožděný vstupní signál. Všechny možnosti změn stavů a výstupů posuvného registru zachycuje následující tabulka, která definuje hodnoty funkcí δ a β Mealyho automatu popisujícího činnost registru.

	δ		β	
S\I	0	1	0	1
000	000	100	0	0
001	000	100	1	1
010	001	101	0	0
011	001	101	1	1
100	010	110	0	0
101	010	110	1	1
110	011	111	0	0
111	011	111	1	1

Obr. 3.5: Tabulka přechodové a výstupní funkce Mealyho automatu (tříbitový posuvný registr)

Z tabulky je zřejmé, že funkce β nezávisí na hodnotě vstupní proměnné, a že $\beta = \lambda \circ \pi_1$, kde $\lambda = \pi_3$, neboť $\lambda(i_1i_2i_3) = \pi_3(i_1i_2i_3) = i_3$. Jde tedy o Mooreův automat 1. druhu, jehož značkovací funkce λ je definována posledním (i předposledním) sloupcem tabulky. Tabulku lze reprezentovat následujícím orientovaným grafem, kterému říkáme přechodový graf automatu.



Obr. 3.6: Přechodový graf tříbitového posuvného registru jako Mealyho automatu

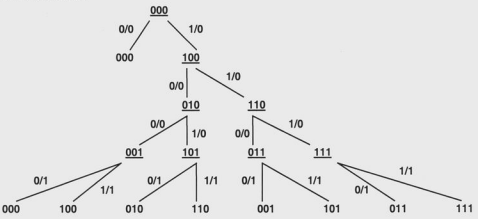
Hrany grafu se u Mealyho automatu označují hodnotami i/o, $i \in I$, $o \in O$, souvisejícími s příslušným přechodem mezi stavy; uzel se označuje stavem, který reprezentuje. Fakt, že náš

automat je Mooreovým automatem 1. druhu, se v Mealyho grafu projevuje tím, že všechny hrany ze společného uzlu vystupující jsou označeny stejnou hodnotou výstupu. To umožňuje „přenést" označení hodnoty výstupu z hran do výchozích uzlů (tj. proti směru šipek). V grafu Mooreova automatu 1. druhu lze tudíž uzly označit hodnotami s/o a hrany pouze hodnotami $i \in I$.

Rozdíl mezi Mealyho a Mooreovým přechodovým grafem je zřejmý z následujícího obrázku, v němž vidíme oba přechodové grafy (vlevo Mealy, vpravo Moore) automatu modelujícího pouze jednobitový posuvný registr.[2]

Obr. 3.7: Přechodové grafy Mealyho a Mooreova automatu pro jednobitový posuvný registr

Přechodový orientovaný graf lze překreslit do tvaru následujícího stromu, jehož konstrukce je zřejmá. Za kořen stromu je třeba volit takový uzel grafu, z něhož jsou všechny ostatní uzly přechodového grafu dosažitelné (v našem případě tuto podmínku splňují všechny uzly). O označování větví a kořenů všech podstromů platí totéž co už bylo řečeno o označování uzlů a hran.[3]



Obr. 3.8: Strom Mealyho automatu (tříbitový posuvný registr)

Prezentace automatu výčtem vychází ze skutečnosti, že vektorová funkce $\delta,\beta \subset S \times I \times S \times O$ a lze ji proto definovat příslušnou množinou čtveřic. Každou ze čtveřic lze považovat za přepis jedné hrany grafu; jejími složkami jsou počáteční stav, aktuální vstup, následný stav, a výstup. Náš automat je tak výčtem definován množinou $\delta,\beta = \{(000,0,000,0), (000,1,100,0), (100,0,010,0), (100,1,110,0), (010,0,001,0), (010,1,101,0), (110,0,011,0), (110,1,111,0), (001,0,000,1), (001,1,100,1), (101,0,010,1), (101,1,110,1), (011,0,001,1), (011,1,101,1), (111,0,011,1), (111,1,111,1)\}$.

[2] Stejně by bylo možno uzly a hrany označit i v grafu Mooreova automatu 2. druhu. Rozdíl by byl pouze v tom, že graf Mooreova automatu 2. druhu by vznikl z příslušného grafu Mealyho automatu „přenesením" hodnot výstupu z hran do uzlů ve směru šipek (u Mooreova automatu 2. druhu jsou v grafu Mealyho automatu stejnou hodnotou výstupu označeny všechny hrany, které do společného uzlu vstupují).

[3] I zde lze v případě Mooreova automatu použít zmíněný Mooreův způsob označování uzlů a hran.

Všimněme si, že každý ze čtyř uvedených popisů plně definuje automat, tedy pětici (S, I, O, δ, β), protože jak množiny S, I, O, tak funkce δ a β lze z popisu sestrojit. V případě $SA = (S, I, \delta)$ by v uvedených formách popisu odpadla tabulka výstupní funkce, v orientovaném grafu a stromu by u hran (Mealy) či uzlů (Moore) nebyl uveden výstup, a při popisu výčtem by čtveřice množiny $\delta,\beta \subset S\times I\times S\times O$ zdegenerovaly na trojice množiny $\delta \subset S\times I\times S$.

Cvičení k lekci 3.

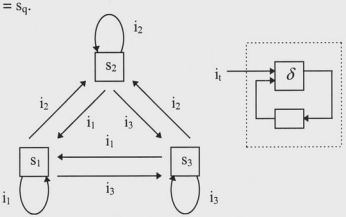
Příklad 3.1: Výtah v třípodlažním domě vykonává tři funkce: Ovládá chod motoru, přemisťuje kabinu mezi podlažími, otevírá a zavírá dveře v cílových podlažích. Kvůli zjednodušení dalšího popisu zanedbáme existenci venkovních přivolávacích tlačítek v podlažích a budeme předpokládat, že výtah je ovládán jen tlačítky v kabině.

a) Přemisťování kabiny mezi podlažími

$S = \{s_1, s_2, s_3\}$ ..... množina stavů; jejími prvky jsou symboly podlaží indexované čísly podlaží,

$I = \{i_1, i_2, i_3\}$ ..... množina signálů vyvolaných stiskem příslušného tlačítka.

Přemisťování kabiny výtahu lze popsat přechody na množině stavů, znázorněnými následujícím grafem. Jedná se o graf a funkční schéma stavového automatu (automatu bez výstupu) SA = (S, I, δ); $\delta(s_p, i_q) = s_q$.



Obr. 3.9

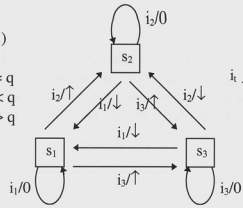
Totéž můžeme popsat i kombinačním automatem $S = \{s\}$, $I = \{i_1, i_2, i_3\}$, $O = \{s_1, s_2, s_3\}$ s výstupní funkcí $g(i_q) = s_q$.[4]

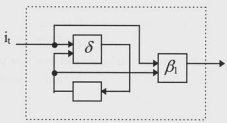
b) Model ovládání chodu motoru

Vyjdeme ze stavového automatu modelujícího přemisťování kabiny mezi podlažími a rozšíříme jej o množinu výstupů O_1 a a výstupní funkci β_1 takto:

[4] Obecně ke každému kombinačnímu automatu s výstupní funkcí g: $I \rightarrow O$, existuje stejně fungující stavový automat (S, I, δ) s množinou stavů $S = O$ a přechodovou funkcí δ: $S \times I \rightarrow S$ ve tvaru $\delta(s_p, i_q) = g(i_q)$. Opačně to ale neplatí. Náhrada kombinačního automatu stejně fungujícím stavovým automatem má jisté přednosti, a proto ji v dalším budeme často používat.

$A_1 = (S, I, O_1, \delta, \beta_1)$
$O_1 = \{\uparrow, \downarrow, 0\}$
$\beta_1(s_p, i_q) = 0$ pro $p = q$
$\beta_1(s_p, i_q) = \uparrow$ pro $p < q$
$\beta_1(s_p, i_q) = \downarrow$ pro $p > q$





Obr. 3.10

Tomu odpovídá následující tabulka přechodové a výstupní funkce:

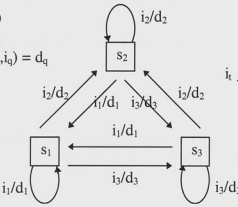
	δ			β_1		
S / I	i_1	i_2	i_3	i_1	i_2	i_3
s_1	s_1	s_2	s_3	0	↑	↑
s_2	s_1	s_2	s_3	↓	0	↑
s_3	s_1	s_2	s_3	↓	↓	0

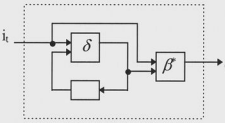
Obr. 3.11

c) Model otevírání dveří

Tento model dostaneme jiným rozšířením téhož stavového automatu:

$A_2 = (S, I, O_2, \delta, \beta_2)$
$O_2 = \{d_1, d_2, d_3\}$
$\beta_2(s_p, i_q) = \beta'(\delta(s_p, i_q), i_q) = d_q$





Obr. 3.12

Jemu odpovídá následující tabulka přechodové a výstupní funkce:[5]

[5] Rozdíl oproti předchozímu případu (automat A_1) je v tom, že automat A_2 nejprve změní stav (kabina přijede do cílového podlaží), a až pak dá výstup (otevře dveře). V důsledku kontraktivity přechodové funkce se během přechodu ztrácí informace o výchozím stavu s_t (dochází k disipaci), a v okamžiku, kdy je třeba otevírat dveře, je k dispozici jen cílový stav s_{t+1}. Proto je β_2 definována pomocí β', do níž místo výchozího stavu vstupuje

	δ			β_2		
S/I	i_1	i_2	i_3	i_1	i_2	i_3
s_1	s_1	s_2	s_3	d_1	d_2	d_3
s_2	s_1	s_2	s_3	d_1	d_2	d_3
s_3	s_1	s_2	s_3	d_1	d_2	d_3

Obr. 3.13

Otázka 3.1: Jakého typu jsou automaty A_1 a A_2 v příkladu 3.1?

Úloha 3.1: Nechť f: $X \to Y$ je bijekce, g: $Y \to Z$, h: $U \to V$, i: $U \to W$ jsou libovolná zobrazení. Pokud je to nutné, doplňte výrazy 1) až 6) závorkami tak, aby každý výraz korektně definoval nějaké zobrazení. Uveďte definiční obory a obory hodnot těchto zobrazení.

1) $f^{-1} \circ f$
2) $f \circ f^{-1}$
3) $g \circ f \times h , i$
4) $g \times f \times h , i$
5) $g \times f \times h \times i$
6) $g , f^{-1} \times h , i$

Úloha 3.2: Nechť f: $X \to Y$, g: $Y \to Z$, h: $U \to V$, k: $V \to W$. Dokažte, že platí

$$(g \times k) \circ (f \times h) = (g \circ f) \times (k \circ h)$$

---

cílový stav. V tomto konkrétním případě cílový stav informuje i o vstupním podnětu, takže je postačujícím vstupem do funkce β'' (zvláštní vstup pro podnět i_i je zde nadbytečný).

4. Elementy systémové koncepce v teorii automatů

4.1. Stručná historie vývoje teorie automatů

V roce 1936 definoval britský matematik A. M. Turing stroj, který se dnes nazývá Turingův stroj. Turingův stroj je z dnešního pohledu automatem s nekonečnou (vnější) pamětí; sehrál významnou roli při budování teorie algoritmů a studenti PF JCU se s ním blíže seznámí v rámci předmětu Teoretické základy informatiky v části pojednávající o teorii výpočetního procesu.

V roce 1943 americký fyziolog McCulloch a americký matematik Pitts vytvořili idealizovaný model neuronové sítě, který je v jistém smyslu také předobrazem automatu.[1] S tímto přístupem se studenti PF JCU seznámí opět v předmětu Teoretické základy informatiky v části pojednávající o neuropočítačích.

První práce, ve kterých byl konečný automat definován v podstatě tak, jak jsme jej definovali v předchozích lekcích, vyšly po roce 1950. Touto problematikou se zabývali zejména dva američtí matematikové, Moore a Mealy, po nichž jsou základní typy konečných automatů pojmenovány. Od této chvíle se teorie automatů rozvíjela někdy i velmi bouřlivým tempem. V teoretické oblasti se pak více či méně odděleně rozvíjely dvě koncepce teorie automatů: **lingvistická koncepce** a **systémová koncepce**. V lingvistické koncepci je automat nástrojem na rozpoznávání, reprezentaci či klasifikaci formálních jazyků - s touto koncepcí se studenti PF JCU setkají v kurzu *teorie formálních jazyků*.[2] V tomto kurzu teorie automatů jsme se zatím zabývali a i v dalších jeho lekcích se budeme zabývat výhradně koncepcí systémovou.[3]

V oblasti aplikací se systémová koncepce teorie automatů uplatňuje zejména v technice ve spojitosti s modelováním diskrétních dynamických systémů (logické obvody v systémech řízení a regulace, číslicové počítače, telekomunikační systémy apod.), ale svůj význam si podržela i v oblasti biologie (celulární automaty, otázky samoreprodukce, modelování růstu jednoduchých organismů apod.). A v novější době nabývají na důležitosti i aplikace systémového přístupu teorie automatů v různých oblastech společenskovědních oborů (např. výzkum stability systémů).

V rámci systémové koncepce je zkoumána jak struktura automatu (tj. jeho vnitřní mechanismus popsaný přechodovou a výstupní funkcí, případně z jakých komponent již realizovaný automat sestává nebo z jakých komponent by žádaný automat bylo možno složit),

---

[1] Kanonická síť automatu definovaná v článku 7.3 je speciální neuronovou sítí s neurony provádějícími operaci logického součinu.

[2] V lingvistickém přístupu je automat (zvaný akceptor) výsledkem formalizace syntaktické analýzy jazyka. V případě regulárního jazyka (jazyk typu 3 v klasifikaci jazyků podle Chomského) jde o konečný deterministický či nedeterministický mechanismus (tzv. konečný deterministický či nedeterministický akceptor), v případě bezkontextového jazyka (typ 2) jde o zásobníkový automat, v případě kontextových jazyků (typ 1 a 0) jde o Turingovy stroje. Zásobníkové automaty a Turingovy stroje jsou mechanismy s vnější pamětí (zásobník, nekonečná páska) neomezenou co do kapacity.

[3] V systémovém přístupu je konečný automat odvozen z představy abstraktního stroje (viz lekce 2.) - jde o konečný deterministický mechanismus. Do systémové koncepce však spadá i stochastický automat, který je odvozen z představy o chování diskrétního náhodného procesu, a proto deterministickým mechanismem býti nemůže. Stochastickými automaty se zde zabývat nebudeme.

tak jeho chování (závislost výstupů na vstupech). V dalších článcích této lekce se nejprve seznámíme se základními pojmy strukturálního přístupu a pak stručně pojednáme o problematice kompozice, dekompozice, analýzy a syntézy automatu vycházející ze vztahu mezi strukturou a chováním. O chování automatu podrobněji pojednáme v lekci 5.

4.2. Pojem struktury automatu

U dynamických systémů, mezi něž patří i automaty, je pojem **struktura** definován jednak z hlediska morfologického, jednak z pohledu časového. V souvislosti s tím mluvíme o struktuře:

- **morfologické** (blokové schéma propojení podsystémů) a
- **dynamické** (popis mechanismu změn, idea abstraktního stroje - viz formy prezentace trojice či pětice - čl. 3.4.).

Morfologickou strukturu konečného automatu je možno vnímat ve dvou základních rovinách - v rovině teoretické a v rovině realizační - a v každé z nich na více rozlišovacích úrovních. Na nejnižší rozlišovací úrovni je automat prezentován jako jeden celek (integrální systém). V rovině teoretické je morfologická struktura integrálního systému popsána příslušným funkčním schématem znázorňujícím vazby mezi přechodovou a výstupní funkcí (viz článek 3.3.). V rovině realizační je morfologická struktura integrálního systému popsána jen jedním blokem, tzv. černou schránkou, do níž nevidíme a pozorujeme jen její vnější projevy (chování).

Při přechodu na vyšší rozlišovací úrovně automat ztrácí svou integritu a postupně se rozpadá na vzájemně propojené komponenty (integrální systémy nižšího řádu). V teoretické rovině jsou komponenty také automaty (obvykle jednodušší). Říkáme, že výsledný automat je **kompozicí** (složeninou) více automatů. Opakem kompozice je **dekompozice** (rozklad) automatu na jednotlivé subautomaty. V rovině realizační jsou komponenty automatu speciální funkční bloky, např. logické moduly či obvody.[4]

Změna v morfologické struktuře vede v obecném případě i ke změnám v dynamické struktuře (tj. v pětici (S, I, O, δ, β)) složeniny. Základní problém kompozice - ze známých dynamických struktur komponent zadané morfologické struktury určit dynamickou strukturu celku. Problém dekompozice je opačný - k dané dynamické struktuře celku nalézt vhodnou morfologickou strukturu a určit dynamické struktury jejích komponent. Vzhledem k tomu, že operace inverzní jsou obecně obtížnější než operace přímé, je i problém dekompozice obtížnější než problém kompozice a na rozdíl od kompozice není obecně jednoznačný (může mít více než jedno řešení).

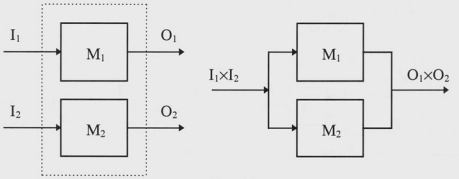
4.3. Kompozice a dekompozice konečného automatu

Dány konečné automaty $M_1 = (S_1, I_1, O_1, \delta_1, \beta_1)$, $M_2 = (S_2, I_2, O_2, \delta_2, \beta_2)$ a $M_3 = (S_3, I_3, O_3, \delta_3, \beta_3)$. Pak:

- **Paralelní kompozicí** automatů M_1 a M_2 je automat
  $M_p = M_1 \parallel M_2 = (S_1 \times S_2, I_1 \times I_2, O_1 \times O_2, \delta_p, \beta_p)$, kde $\parallel$ je symbol operace paralelní kompozice,

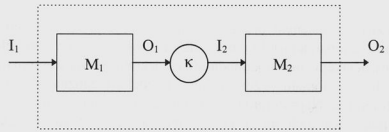
[4] S realizační rovinou morfologické struktury se blíže seznámíme v lekci 6. V této lekci se pohybujeme výhradně v rovině teoretické.

$\delta_p((s_1,s_2),(i_1,i_2)) = (\delta_1(s_1,i_1), \delta_2(s_2,i_2))$ a $\beta_p((s_1,s_2),(i_1,i_2)) = (\beta_1(s_1,i_1), \beta_2(s_2,i_2))$.
Morfologickou strukturu paralelní kompozice znázorňují následující varianty blokových schémat (viz Obr. 4.1):



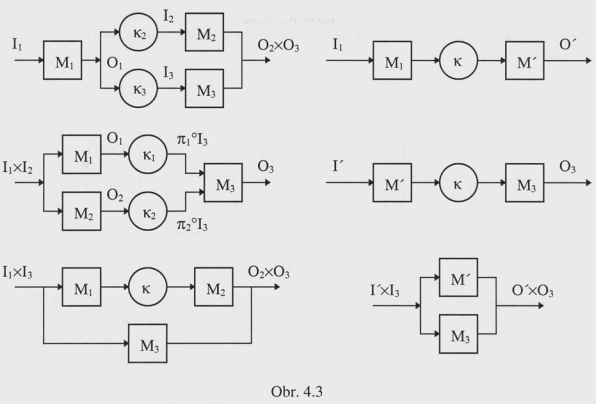
Obr. 4.1

- **Sériovou kompozicí** automatů M_1 a M_2 je automat $M_s = M_1 \Theta_\kappa M_2 = (S_1 \times S_2, I_1, O_2, \delta_s, \beta_s)$, kde Θ_κ je symbol operace sériové kompozice s přizpůsobovací funkcí κ: $O_1 \rightarrow I_2$ (anglicky *interface*), $\delta_s((s_1,s_2),i_1) = (\delta_1(s_1,i_1), \delta_2(s_2,\kappa(\beta_1(s_1,i_1))))$ a $\beta_s((s_1,s_2),i_1) = \beta_2(s_2,\kappa(\beta_1(s_1,i_1)))$.
Morfologickou strukturu sériové kompozice znázorňuje následující blokové schéma (viz Obr. 4.2):



Obr. 4.2

Oba výše uvedené základní typy kompozice lze indukcí rozšířit na libovolný počet členů. Kromě toho je lze i různě kombinovat a vytvářet tak vícestupňové smíšené (sérioparalelní) kompozice. Základní morfologické struktury sérioparalelních kompozic uvádí následující obrázek:



Obr. 4.3

Dekompozice je opačným procesem kompozice, a proto také může být *paralelní*, *sériová* či *smíšená* (sérioparalelní). Praktický smysl má pouze netriviální dekompozice,[5] jejíž komponenty jsou v nějakém smyslu jednodušší než dekomponovaný automat. Jak jsme viděli, kompozice několika automatů vždy dává výsledek jednoznačně určený příslušnou morfologickou strukturou. Naproti tomu netriviální dekompozici automat vůbec mít nemusí, a pokud nějakou má, může jich mít i více. Navíc se často stává, že zpětnou kompozicí dekomponovaného automatu nedostaneme původní automat, ale složeninu s vyšším počtem stavů, která chování původního automatu pouze simuluje.[6]

Najít všechny možné netriviální dekompozice automatu je úkolem abstraktně pojaté **analýzy dekomponovatelnosti**, využívající k tomuto účelu jen čistě algebraické prostředky.[7]

---

[5] Triviální sériovou dekompozici tvoří původní automat v sérii s kombinačním automatem, který na výstupu „vrací" vstupní podněty. Triviální paralelní dekompozice sestává ze dvou původních automatů nastavených do stejných stavů a na něž přicházejí stejné vstupní podněty. Je zřejmé, že triviální dekompozice existuje vždy.

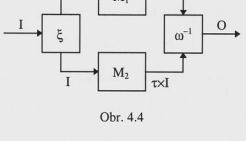
[6] Nevyhnutelně to nastane v případě, kdy počet stavů dekomponovaného automatu je prvočíslo. Počet stavů zpětné složeniny prvočíslem býti nemůže (je roven součinu počtů stavů všech komponent), a proto musí být jiný (vyšší). Co rozumíme simulací se dozvíme v lekci 5.

[7] Postačující podmínkou dekomponovatelnosti konečného automatu $M = (S, I, O, \delta, \beta)$ je existence netriviálního SV rozkladu $\sigma = \{\langle s\rangle_\sigma : s \in S\}$ stavového prostoru S. Triviálním rozkladem je rozklad sdružující všechny prvky do jediné třídy (nejhrubší rozklad) nebo rozklad sestávající jen z jednoprvkových tříd (nejjemnější rozklad). σ je SV rozklad $\Leftrightarrow \forall a,b \in S: [(\langle a\rangle_\sigma = \langle b\rangle_\sigma) \Rightarrow (\forall i \in I: \langle\delta(a,i)\rangle_\sigma = \langle\delta(b,i)\rangle_\sigma)]$.

Splnění postačující podmínky umožňuje *sériovou* dekompozici na komponenty M_1 a M_2, v níž „image machine" $M_1 = (\sigma, I, \sigma\times I, \delta_1, \beta_1)$, kde $\delta_1(\langle s\rangle_\sigma, i) = \langle\delta(s,i)\rangle_\sigma$ a β_1 je identitou na $\sigma\times I$, tj. $\beta_1(\langle s\rangle_\sigma, i) = (\langle s\rangle_\sigma, i)$. Interface κ je také identitou na $\sigma\times I$ a „tail machine" $M_2 = (\tau, \sigma\times I, O, \delta_2, \beta_2)$, kde $\tau = \{\langle s\rangle_\tau : s \in S\}$je libovolný netriviální rozklad ortogonální k SV rozkladu σ, tj. takový, že $\forall s \in S: \langle s\rangle_\sigma \cap \langle s\rangle_\tau = \{s\}$. Rozklad τ vždy existuje a můžeme jej vytvořit např. tak, že z každé třídy SV rozkladu σ vybereme po jednom prvku a z takto vybraných prvků vytvoříme jednu třídu rozkladu τ. Poté z neprázdných zbytků tříd rozkladu σ vybereme opět po jednom prvku a z nich vytvoříme další třídu rozkladu τ. Postup opakujeme až do vyčerpání všech tříd rozkladu σ.

Rozklad τ bude mít tolik tříd, kolik má největší třída SV rozkladu σ prvků. Nechť $X_0 \in \tau$ a $o \in O$ jsou libovolné prvky. Pak pro δ_2 a β_2 platí:$[\langle x\rangle_\tau \cap \langle y\rangle_\sigma = \emptyset] \Rightarrow [\delta_2(\langle x\rangle_\tau,\langle y\rangle_\sigma,i)) = X_0$ a $\beta_2(\langle x\rangle_\tau,\langle y\rangle_\sigma,i)) = o]$, $[\langle x\rangle_\tau \cap \langle y\rangle_\sigma = \{s\}] \Rightarrow [\delta_2(\langle x\rangle_\tau,\langle y\rangle_\sigma,i)) = \langle\delta(s,i)\rangle_\tau$ a $\beta_2(\langle x\rangle_\tau,\langle y\rangle_\sigma,i)) = \beta(s,i)]$.

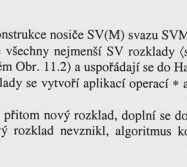
Existence *paralelní* dekompozice navíc vyžaduje od rozkladu τ i SV vlastnost. V ní obě komponenty $M_1 = (\sigma, I, \sigma\times I, \delta_1, \beta_1)$ a $M_2 = (\tau, I, \tau\times I, \delta_2, \beta_2)$ budou typu „image machine" a přichází na ně stejné vstupní podněty. To zajišťuje vstupní kódovací funkce ξ: $I \to I\times I$ ve tvaru $\xi(i) = (i,i)$. Naproti tomu výstupy z obou paralelních komponent je třeba parciální dekódovací funkcí ω^{-1}: $(\sigma\times I)\times(\tau\times I) \to O$ - (ve tvaru $\omega^{-1}(\langle\langle y\rangle_\sigma,k\rangle,\langle\langle x\rangle_\tau,j\rangle) = \beta(s,i)$ při $\langle x\rangle_\tau \cap \langle y\rangle_\sigma = \{s\}$a k=j=i; v opačném případě, tj. k≠j nebo $\langle x\rangle_\tau \cap \langle y\rangle_\sigma = \emptyset$, $\omega^{-1}(\langle\langle y\rangle_\sigma,k\rangle,\langle\langle x\rangle_\tau,j\rangle)$ = libovolný prvek množiny O) - převádět do adekvátních prvků množiny O. Proto musí být vstupy a výstupy komponent připojeny k vstupnímu kodéru ξ a výstupnímu dekodéru ω^{-1} tak jak je to znázorněno v následujícím obrázku:



Obr. 4.4

Aby jak v sériové, tak v paralelní dekompozici složenina fungovala stejně jako integrální automat M nastavený do počátečního stavu $s \in S$, je třeba komponenty M_1 a M_2 nastavit do počátečních stavů $\langle s\rangle_\sigma \in \sigma$ a $\langle s\rangle_\tau \in \tau$. Popsaným způsobem je možno jak v sériové tak paralelní dekompozici dále rozkládat i komponenty M_1 a M_2 a tím postupně získávat různé jemnější a smíšené rozklady automatu M.

Z uvedeného je zřejmé, že analýza dekomponovatelnosti musí vzít v potaz všechny existující SV rozklady stavového prostoru S automatu M. Za tím účelem nejprve konstruuje tzv. *svaz SV rozkladů*, tj. algebraickou strukturu SVM = (SV(M), ≤, *, +), kde SV(M) je systém všech SV rozkladů množiny S automatu M, ≤ je relace částečného uspořádání na SV(M) definovaná takto: $\forall\sigma,\tau \in SV(M)$: $\sigma \le \tau \Leftrightarrow (\forall s \in S$: $\langle s\rangle_\sigma \subseteq \langle s\rangle_\tau)$; * a + jsou binární, idempotentní ($\sigma*\sigma = \sigma$, $\sigma+\sigma = \sigma$), komutativní ($\sigma*\tau = \tau*\sigma$, $\sigma+\tau = \tau+\sigma$) a asociativní ($\sigma*(\tau*\rho) = (\sigma*\tau)*\rho$, $\sigma+(\tau+\rho) = (\sigma+\tau)+\rho$) operace na SV(M). Operace * je definována předpisem $\sigma*\tau = \{\langle s\rangle_\sigma \cap \langle s\rangle_\tau$: $s \in S\}$, operace + je definována algoritmem popsaným tímto vývojovým diagramem:



A: $\sigma\cup\tau = \{\langle s\rangle_\sigma$: $s \in S\}\cup\{\langle s\rangle_\tau$: $s \in S\}$
B: $\exists X,Y \in (\sigma\cup\tau)$: $X \cap Y \neq \emptyset$?
C: $\sigma\cup\tau \approx [(\sigma\cup\tau) - \{X,Y\}]\cup(X\cup Y)$
D: $\sigma+\tau = \sigma\cup\tau$

Obr. 4.5

Algoritmus konstrukce nosiče SV(M) svazu SVM sestává ze tří následujících kroků:

- Vytvoří se všechny nejmenší SV rozklady ⟨s,s´⟩, obsahující dvojici s,s´∈S v jedné třídě rozkladu (viz čl. 11.2 a v něm Obr. 11.2) a uspořádají se do Hasseovského diagramu (viz čl. 11.1).
- Další rozklady se vytvoří aplikací operací * a + na všechny dvojice nesrovnatelných prvků v Hasseovském diagramu.
- Vznikne-li přitom nový rozklad, doplní se do Hasseovského diagramu a opakuje se předchozí krok. Pokud žádný nový rozklad nevznikl, algoritmus končí a výsledný diagram je Hasseovským diagramem svazu SVM.

Tento přístup své závěry vyvozuje pouze z existence a vlastností specifických rozkladů (tzv. SV rozkladů - viz poznámka pod čarou 7) stavového prostoru dekomponovaného automatu a věcnou podstatu problému v potaz nebere. V technické praxi se spíše setkáváme s intuitivním přístupem k dekompozici systémů, založeném naopak právě na znalosti věcné stránky, který nehledá všechny existující dekompozice, ale jen ty z jistých hledisek „dostatečně rozumné". Dekompozicemi, k nimž se dospělo na základě znalosti věcné stránky jsou sériové dekompozice zvažované v následujícím příkladu, na němž si ukážeme, co je obecně hlavním přínosem dekompozice problému na dílčí podproblémy. Souvislost věcného přístupu s čistě algebraickým přístupem objasníme v cvičném příkladu 4.6 a na řešení úlohy 4.5.

Příklad sériové dekompozice: V článku 3.4. je uveden automatový model tříbitového posuvného registru koncipovaného jako integrální systém $\rightarrow\boxed{i_1\,|\,i_2\,|\,i_3}\rightarrow$ s počtem $2^3 = 8$ vnitřních stavů. Obecně platí, že integrální n-bitový posuvný registr by měl 2^n vnitřních stavů. Jednobitový posuvný registr $\rightarrow\boxed{i}\rightarrow$ by tak měl jen $2^1 = 2$ stavy (viz čl. 3.4.), dvoubitový $\rightarrow\boxed{i_1\,|\,i_2}\rightarrow$ $2^2 = 4$ stavy. Tříbitový posuvný registr lze sériově dekomponovat buď do tří jednobitových registrů $\rightarrow\boxed{i}\rightarrow\boxed{i}\rightarrow\boxed{i}\rightarrow$, nebo do jedné ze dvou možných kombinací jednobitového a dvoubitového posuvného registru $\rightarrow\boxed{i_1\,|\,i_2}\rightarrow\boxed{i}\rightarrow$ či $\rightarrow\boxed{i}\rightarrow\boxed{i_1\,|\,i_2}\rightarrow$. Realizace dekomponovaných verzí je jednodušší než realizace verze integrální: V prvním případě má komponenta jen dva vnitřní stavy, ve druhém nejvýše čtyři; tedy v obou případech méně než původních osm. Za toto zjednodušení se v praxi platí prodloužením reakční doby (reakční doba složeniny je součtem reakčních dob komponent) nebo-li snížením rychlosti funkce posuvného registru. Třísložková varianta tak bude zhruba třikrát (dvousložková dvakrát) pomalejší než varianta integrální.

Význam dekompozice vynikne zejména v rozměrnějších případech. Např. integrální třicetidvoubitový posuvný registr by musel mít 2^{32}, tj. více než $4 \cdot 10^9$ vnitřních stavů, a byl by prakticky nerealizovatelný, zatímco v jeho dekomponované variantě do 32 jednobitových (32 krát pomalejší) či 16 dvoubitových komponent (16 krát pomalejší) je realizováno jen $2 \cdot 32 = 4 \cdot 16 = 64$ vnitřních stavů.[8]

Oproti sériové dekompozici má paralelní dekompozice (pokud existuje) výhodu v tom, že také zjednodušuje řešení (i když ne vždy tak výrazně jako v uvedeném příkladu) a přitom nezpomaluje funkci složeniny.[9] Vhodného kompromisu mezi rychlostí reakce a složitostí realizace proto v praxi obvykle dosahujeme smíšenými (sériovo-paralelními) dekompozicemi.

Z tvaru Hasseovského diagramu svazu SVM a z Hasseovských diagramů jeho podsvazů lze pak snadno odvodit jak morfologické struktury možných dekompozic automatu M tak i dynamické struktury příslušných komponent (viz řešení úlohy 4.5 a příklad 4.7).

[8] Za případ sériové dekompozice řešení problému lze považovat i klasický algoritmus, který je rozkladem postupu řešení na jednoduché na sebe navazující dílčí kroky. Bez možnosti této dekompozice bychom ve většině případů k výsledku sotva došli (mohli bychom jej jen intuitivně odhadovat).

[9] Přednosti sériové a paralelní dekompozice vyniknou zejména při srovnávání možností člověka a číslicového počítače. Počítač ke zpracování dat využívá převážně sekvenční (sériový) přístup, zatímco funkce lidského mozku je založena na masivním paralelismu neuronové tkáně. I když elektrochemické procesy probíhající v elementech neuronové tkáně jsou řádově stomilionkrát pomalejší než analogické fyzikální procesy v přechodových vrstvách mezi polovodiči, počítač dominuje jen v elementárních výpočetních a logických operacích. Naproti tomu ve vyhodnocování složitých situací (např. v rozpoznávání lidských tváří) je roční dítě

4.4. Struktura versus chování

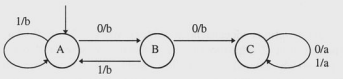
V článku 2.2. jsme dvojici (podnět, reakce) v okamžiku t nazvali **aktuálním elementárním chováním** konečného automatu. Posloupnost aktuálních elementárních chování, kterou automat prochází, jsme nazvali *fázovou trajektorií* okamžikem aktuální přítomnosti rozdělenou na *historii* a *evoluci*.

Elementárním chováním nazveme množinu všech možných *aktuálních elementárních chování*. Elementární chování pak obecníme na *chování* tak, že aktuální hodnoty podnětu (x_t) a reakce (y_t) v dvojici aktuálního elementárního chování nahradíme libovolně dlouhými konečnými sekvencemi[10] neboli **řetězci** $x = x_t x_{t+1} x_{t+2} \ldots x_{t+n}$ a $y = y_t y_{t+1} y_{t+2} \ldots y_{t+n}$ bezprostředně následujících hodnot vstupní a výstupní proměnné. Množinu všech možných takto vzniklých dvojic nazveme **chováním** konečného automatu. Elementární chování tak bude zvláštním případem chování pro jednočlenné řetězce vstupních podnětů.

Definice chování jakožto množiny všech možných dvojic vstupních a výstupních řetězců je *algebraickou definicí* chování. Tato definice je upřesněna v lekci 5, v níž se chováním automatu zabýváme podrobněji. Každou množinu lze alternativně definovat i výčtem vlastností jejích prvků[11] - v našem případě výčtem vztahů mezi vstupním a výstupním řetězcem. Tím od algebraické definice přecházíme k verbálnímu popisu chování. Zde i v navazujících cvičeních budeme chování automatu popisovat verbálně.

Chování automatu jednoznačně vyplývá z jeho struktury. Zjištění chování ze zadané struktury je problémem **analýzy** automatu. Při analýze automatu nejprve zohledníme vliv jeho morfologické struktury na prvky pětice (S, I, O, δ, β) celku. Pak dosazením dynamických struktur komponent do pětice odvodíme dynamickou strukturu celku. Z ní pak už snadno vyčteme verbální popis chování.

Příklad analýzy automatu – úkolem je z dané dynamické struktury vyčíst chování.



Obr. 4.6

Řešení: Automat vydává symbol *b* tak dlouho, dokud není na vstupu dvakrát za sebou symbol 0. Od toho okamžiku vydává pouze symbol *a*

---

rychlejší a úspěšnější než výkonné superpočítače. Tato výkonnost mozku je přičítána právě masivnímu paralelismu jeho dekomponované struktury i funkce a odtud přichází i motivace pro výzkum a vývoj paralelních výpočetních systémů (celulární výpočetní systémy, neuropočítače apod.) využívajících paralelních výpočetních procedur.

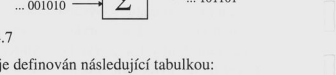
[10] Přípustná je i sekvence nulové délky (prázdná sekvence ε).

[11] Pozor! Neplést výčet vlastností prvků s výčtem prvků.

Opačný problém, tj. hledání struktury, která by vedla k danému chování, je problémem **syntézy** automatu. Syntéza je úlohou víceznačnou. V oblasti morfologie její víceznačnost souvisí s víceznačností možné dekompozice, ale jak uvidíme zejména v dalších lekcích, víceznačná je i v oblasti dynamiky. Syntézou morfologické struktury jsme se v podstatě zabývali již v předchozím článku, a proto se v dalším zaměříme výhradně na strukturu dynamickou (tj. budeme se zabývat jen integrálními automaty). Víceznačnost syntézy, tedy fakt, že dané chování lze zajistit více různými (dokonce i vzájemně si nepodobnými) strukturami, je pro realizaci přínosem.[12] Umožňuje klást na výběr struktury další požadavky (např. požadavek homogenity či symetrie struktury, minimálního počtu vnitřních stavů apod.) a tak systém „ušít na míru" potřebám praxe.

Příklad syntézy automatu - konstrukce dynamické struktury realizující zadané chování.

Zadání: Sestrojit paralelní sčítačku dvojic binárních čísel, do níž operandy vstupují současně, po jednotlivých řádech odzadu. Sčítačka postupně tiskne výsledek také po jednotlivých řádech odzadu (viz Obr. 4.7):



Obr. 4.7

Algoritmus sčítání, stejný pro všechny řády, je definován následující tabulkou:

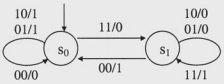
přenos „1" z předchozího řádu	
NE	ANO
0 + 0 = 0	0 + 0 = 1
0 + 1 = 1 + 0 = 1	0 + 1 = 1 + 0 = 0 (vzniká přenos „1")
1 + 1 = 0 (vzniká přenos „1")	1 + 1 = 1 (vzniká přenos „1")

Obr. 4.8

[12] V oblasti poznávání světa je přínos víceznačnosti syntézy sporný. Např. fyzika zjišťuje chování přírody pozorováním a experimentováním. Z odpozorovaných jevů pak „syntetizuje" teorii jakožto ucelený a logicky zdůvodněný systém „přírodních zákonů" a principů, který má pozorované chování vysvětlit. Zde je nejednoznačnost syntézy spíše na závadu a brání nám v nalezení té správné představy. Kdybychom ji měli, fungovala by jak v mikrosvětě tak v makrosvětě a žádný nový objev by nás nepřekvapil, protože by jej vysvětlovala a tudíž předpokládala. Místo toho máme jinou teorii pro mikrosvět (kvantovou mechaniku) a jinou pro makrosvět (relativistickou mechaniku), a více než dvacet základních fyzikálních konstant, které umíme změřit, ale ne vysvětlit. Ze čtyř základních přírodních sil - *gravitace* (drží pohromadě vesmír), *silná interakce* (drží pohromadě částice v atomových jádrech), *elektromagnetická interakce* (udržuje elektrony na drahách kolem atomového jádra, díky ní drží pohromadě pevné látky i atomy v molekulách a fungují elektrické stroje a přístroje), *slabá interakce* (způsobuje radioaktivitu typu beta rozpad a pomáhá hvězdám svítit) - se zatím podařilo sjednotit (ve smyslu vysvětlit jejich vzájemnou souvislost) jen poslední dvě. Na cestě k sjednocení všech čtyř v „unitární teorii" je zatím největší překážkou nejslabší z nich - gravitace, která je řádově 10^{-48} krát slabší než elektromagnetická interakce (stejný nepoměr je mezi poloměry protonu a dnes známého vesmíru). Zohlednit tak extrémně rozdílné hodnoty v jediné rovnici matematika neumí („nekonečně malé" zanedbává). Proto nové objevy nám původní představy mění (viz např. zobecnění klasické mechaniky na relativistickou). Jediné co víme jistě je to, že žádná ze současných představ, byť sebeužitečnější, není tou pravou. Ale věříme, že rozvoj poznání nás k ní stále přibližuje.

Řešení: Automat ME = (S, I, O, δ, β), S = {s_0, s_1}, s_0 - bez přenosu, s_1 - s přenosem, I = {00, 01, 10, 11}, O = {0, 1}, tabulka funkcí δ a β vznikne transformací algoritmu sčítání do formy automatu:

	δ				β			
S/I	00	01	10	11	00	01	10	11
s_0	s_0	s_0	s_0	s_1	0	1	1	0
s_1	s_0	s_1	s_1	s_1	1	0	0	1

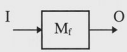


Obr. 4.9

Cvičení k lekci 4.

Příklad 4.1: Automaty $A_1 = (S, I, O_1, \delta, \beta_1)$ - model ovládání chodu motoru, - a automat $A_2 = (S, I, O_2, \delta, \beta_2)$ - model otevírání dveří domovního výtahu - v příkladu 3.1 se v každé hodnotě diskrétního času nachází ve stejném stavu (je jím stání kabiny výtahu v příslušném podlaží) a přichází na ně i shodné vstupní podněty (stisk tlačítka pro jízdu do příslušného patra). V důsledku této vzájemné závislosti bude výsledný model řízení výtahu (automat A) paralelní kompozicí ve tvaru $A = A_1||A_2 = (S, I, O_1 \times O_2, \delta, (\beta_1, \beta_2))$.

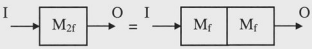
Příklad 4.2: Sériovou kompozicí (bez účasti přizpůsobovacího členu) dvou jednobitových posuvných registrů, tj. automatů $M_f = (S, I, O, \delta_f, \beta_f)$



	δ_f		β_f	
S\I	0	1	0	1
0	0	1	0	0
1	0	1	1	1

Obr. 4.10

dostaneme dvoubitový posuvný registr $M_{2f} = M_f \Theta M_f = (S \times S, I, O, \delta_{2f}, \beta_{2f})$



	δ_{2f}		β_{2f}	
S\I	0	1	0	1
00	00	10	0	0
01	00	10	1	1
10	01	11	0	0
11	01	11	1	1

Obr. 4.11

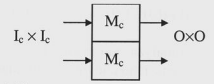
v němž pro vstup $0 \in I$ ve stavu $10 \in S \times S$ dostaneme
$\delta_{2f}(10, 0) = (\delta_f(1, 0), \delta_f(0, \beta_f(1, 0))) = (0, \delta_f(0, 1)) = 01$,
$\beta_{2f}(10, 0) = \beta_f(0, \beta_f(1, 0)) = \beta_f(0, 1) = 0$.

Příklad 4.3: Rozšířením množiny I automatu M_f z příkladu 4.2 o „nedestruktivní čtení" R dostaneme jednobitovou paměťovou buňku $M_c = (S, I_c, O, \delta_c, \beta_c)$.

	δ_c			β_c		
S\Ic	0	1	R	0	1	R
0	0	1	0	0	0	0
1	0	1	1	1	1	1

Obr. 4.12

Paralelní kompozicí dvou buněk M_c dostaneme dvoubitovou paměťovou buňku $M_{2c} = M_c||M_c = (S \times S, I_c \times I_c, O \times O, \delta_{2c}, \beta_{2c})$ - viz Obr. 4.13:

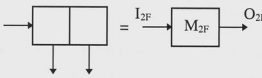


Obr. 4.13

definovanou tabulkou

	δ_{2c}									β_{2c}								
	00	01	0R	10	11	1R	R0	R1	RR	00	01	0R	10	11	1R	R0	R1	RR
00	00	01	00	10	11	10	00	01	00	00	00	00	00	00	00	00	00	00
01	00	01	01	10	11	11	00	01	01	01	01	01	01	01	01	01	01	01
10	00	01	00	10	11	10	10	11	10	10	10	10	10	10	10	10	10	10
11	00	01	01	10	11	11	10	11	11	11	11	11	11	11	11	11	11	11

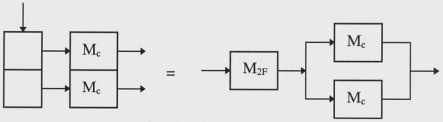
Příklad 4.4: Dvoubitový posuvný registr M_{2f} z příkladu 4.2 pozměníme následovně: Výstup z obou buněk bude paralelní, vstupní množinu rozšíříme o prvky L (loading) a R (nedestruktivní čtení), a výstupní množinu rozšíříme o prvek R tak, aby vznikl výsledný složený automat $M_{2F} = (S_{2F}, I_{2F}, O_{2F}, \delta_{2F}, \beta_{2F})$:



	δ_{2F}				β_{2F}			
S\I	0	1	L	R	0	1	L	R
00	00	10	00	00	RR	RR	00	RR
01	00	10	01	01	RR	RR	01	RR
10	01	11	10	10	RR	RR	10	RR
11	01	11	11	11	RR	RR	11	RR

Obr. 4.14

Sériovou kompozicí $M_{2F} \Theta M_{2c}$ (M_{2c} viz příklad 4.3) dostaneme sérioparalelní kompozici $M_{2C} = M_{2F} \Theta M_{2c} = M_{2F} \Theta (M_c || M_c) = (S_{2F} \times S \times S, I_{2F}, O \times O, \delta_{2C}, \beta_{2C})$,



Obr. 4.15

v níž

$\delta_{2C}((s_1,(s_2,s_3)), i) = (\delta_{2F}(s_1, i),(\delta_{2c}((s_2,s_3), \beta_{2F}(s_1, i))))$ a

$\beta_{2C}((s_1,(s_2,s_3)), i) = \beta_{2c}((s_2,s_3), \beta_{2F}(s_1, i))$.

Složený automat M_{2C} má $4 \cdot 2 \cdot 2 = 16$ vnitřních stavů a např. pro vstup L ve stavu (01,(1,0)) dostáváme:

$\delta_{2C}((01,(1,0)), L) = (\delta_{2F}(01, L),(\delta_{2c}((1,0), \beta_{2F}(01, L)))) = (01,(\delta_{2c}((1,0), 01))) =$ (01,(0,1)) a

$$\beta_{2C}((01,(1,0)), L) = \beta_{2c}((1,0), \beta_{2F}(01, L)) = \beta_{2c}((1,0), 01) = (1,0).$$

Příklad 4.5 (analýza automatu): Cílem analýzy složeného automatu M_{2C} z příkladu 4.4 je zjistit jeho chování a verbálně je popsat. Výsledek pak lze shrnout takto:

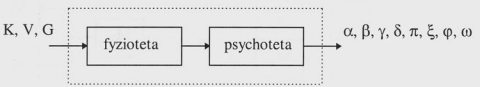
Morfologicky je M_{2C} sériovou kompozicí dvou komponent: modifikovaného posuvného registru M_{2F} s paralelním výstupem (vstupní komponenta), a dvoubitové paměti M_{2c} (výstupní komponenta). V každém taktu své činnosti (nezávisle na hodnotě vstupního podnětu) M_{2C} na výstup paralelně vydá aktuální obsah paměti M_{2c}. Ke změnám obsahu paměti M_{2c} přitom dochází jen při vstupním podnětu L, a to tak, že se do paměti M_{2c} paralelně přenese obsah posuvného registru M_{2F}, postupně naplňovaného jen vstupními hodnotami 0 a 1. Vstupní podnět R nemění obsah registru ani paměti.

Závěr: Automat M_{2C} je dvoubitovou binární pamětí se sériovým zápisem a paralelním čtením. K nedestruktivnímu čtení dochází při podnětu R, k zápisu dat postupně nahromaděných ve vstupním registru dochází při podnětu L.

Příklad 4.6 (sériová dekompozice *staré tety Evženie* z cvičného příkladu 2.1): Každý z pěti stavů (rozpoložení) integrální tety Evženie, označených čísly 1 až 5, má svůj fyziologický a psychický aspekt. Z fyziologického pohledu může být utlumená (spí), opilá nebo střízlivá. Psychicky na tom může být dobře (má dobrou náladu) nebo špatně (má špatnou náladu). Předpokládáme-li, že ve spánku je na tom psychicky dobře, můžeme její jednotlivá rozpoložení vyjádřit příslušnou kombinací obou aspektů podle následující tabulky:

rozpoložení (stav)	aspekt fyziologický	aspekt psychický (nálada)
1 - spí	A - utlumená	D - dobrá
2 - vlídná	B - střízlivá	D - dobrá
3 - podrážděná	B - střízlivá	E - špatná
4 - sentimentální	C - opilá	D - dobrá
5 - rozzuřená	C - opilá	E - špatná

Integrální tetu lze pak sériově dekomponovat do dvou z hlediska počtu stavů menších subtet (fyziotety a psychotety) např. takto:



Obr. 4.16

kde

fyzioteta:

	δ_1			β_1		
	K	V	G	K	V	G
A	B	A	A	A_K	A_V	A_G
B	B	B	C	B_K	B_V	B_G
C	C	C	A	C_K	C_V	C_G

psychoteta:				δ_2									β_2					
	A_K	A_V	A_G	B_K	B_V	B_G	C_K	C_V	C_G	A_K	A_V	A_G	B_K	B_V	B_G	C_K	C_V	C_G
D	D	D	D	E	D	D	E	D	D	α	δ	β	γ	δ	π	φ	δ	ω
E				E	D	E	E	D	D				γ	ξ	π	φ	ξ	ω

Prázdná políčka v posledním řádku tabulky přechodové a výstupní funkce psychotety odpovídají výchozí kombinaci (A,E) stavů fyziotety a psychotety, která neodpovídá žádnému stavu integrální tety (viz úvodní tabulka rozkladu stavů), a proto (pokud ji subtetám nevnutíme vnějším zásahem) nemůže nastat. Tato políčka lze vyplnit jakkoli. V praxi tato situace nastává poměrně často a obvykle tato políčka vyplňujeme tak, abychom co nejlépe vyhověli dodatečným konstrukčním, ekonomickým či jiným požadavkům.[13]

Úloha 4.1: Pomocí rovnic pro δ_{2C} a β_{2C} uvedených v příkladu 4.4 vypočtěte všechny prvky tabulky přechodové a výstupní funkce složeného automatu M_{2C}.

Úloha 4.2: Určete dynamické struktury (přechodové a výstupní funkce) integrálních automatů definovaných třemi základními morfologickými strukturami sérioparalelních kompozic uvedenými v článku 4.3. na Obr. 4.3.

Úloha 4.3: (syntéza automatů) - Sestrojte dynamické struktury automatů se vstupní abecedou {0, 1} a výstupní abecedou {0, 1} tak, aby vyhovovaly popsanému chování:

a) Výstupní symbol 1 se objeví právě tehdy, jestliže se na vstup poprvé dostane symbol 0 následující za skupinou dvou nebo více symbolů 1. Ve všech ostatních případech vydá 0.

b) Automat se vstupy 0, 1 a výstupy 0, 1 vydá 1 právě tehdy, jestliže na vstupu je 1, která je od počátku v pořadí sudým symbolem 1. Jinak vydá 0.

c) Odpovědí na libovolné první dva symboly je 0. Dále se každý n-tý výstupní symbol rovná (n-2)-mu vstupnímu symbolu.

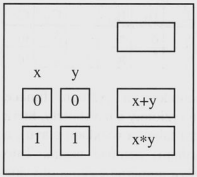
d) Výstup 1 je vydán, kdykoli je na vstupu člen skupiny symbolů 1, která je předcházena skupinou 00. Ve všech ostatních případech vydá 0.

e) Symbol 1 je vydán tehdy, jestliže je na vstupu (3k+1)-ní symbol pro nějaké přirozené číslo k. Jinak je vstupní a výstupní symbol stejný.

f) Automat se třemi vnitřními stavy odpoví na čtrnáctičlennou vstupní posloupnost 00001000100010 výstupní posloupností 01010000101001. Najděte jej.

[13] Z pohledu čistě algebraického přístupu ve smyslu poznámky pod čarou 7 v článku 4.3. fyziologický aspekt definuje SV rozklad $\sigma = \{A, B, C\}$ stavového prostoru integrální tety, s třídami $A = \{1\}$, $B = \{2, 3\}$, $C = \{4, 5\}$. Psychický aspekt definuje rozklad $\tau = \{D, E\}$ (není to SV rozklad) s třídami $D = \{1, 2, 4\}$ a $E = \{3, 5\}$. Rozklady σ a τ jsou vzájemně ortogonální, fyzioteta je „image machine", psychoteta je „tail machine" (viz jejich definice v poznámce pod čarou 7).

Úloha 4.4: (syntéza automatu - řešená úloha) - zadání:



Na obrázku vlevo je znázorněn ovládací panel Booleovského kalkulátoru, který k zadané dvojici hodnot Booleovských proměnných x a y vypočte jejich logický součet nebo logický součin. Hodnoty proměnných se zadají postupným stiskem tlačítek ve sloupcích pod proměnnými a přijetí každé hodnoty kalkulátor oznámí pípnutím. Stisk tlačítka požadované operace zobrazí její výsledek na displeji umístěném v pravém horním rohu. Vytvořte Mealyho model kalkulátoru.

Obr. 4.17

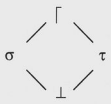
Řešení: I = {A, B, C, D, E, F}, O = {#, 0, 1}, kde symbol # značí pípnutí a symboly A až F značí podněty vyvolané stiskem příslušného tlačítka (význam je uveden v závorce za symbolem): A(x=0), B(y=0), C(x=1), D(y=1), E(x*y), F(x+y). U podnětů E a F výstup závisí i na dvojicích hodnot xy, proto je výhodné vzít tyto dvojice za hodnoty stavové proměnné: S = {a, b, c, d}, kde a = 00, b = 01, c = 10, d = 11. Hledaný automat ME = (S, I, O, δ, β) je pak popsán následující tabulkou vektorové funkce (δ,β): S×I → S×O.

	A	B	C	D	E	F
a	a,#	a,#	c,#	b,#	a,0	a,0
b	b,#	a,#	d,#	b,#	b,0	b,1
c	a,#	c,#	c,#	d,#	c,0	c,1
d	b,#	c,#	d,#	d,#	d,1	d,1

Úloha 4.5 (analýza dekomponovatelnosti automatu - řešená úloha).

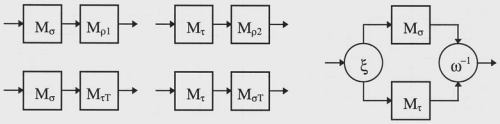
Zadání: Vytvořte všechny rozklady stavového prostoru Booleovského kalkulátoru z úlohy 4.4; podle návodu v poznámce pod čarou 7 proveďte analýzu dekomponovatelnosti kalkulátoru, nakreslete schémata možných dekompozic a definujte jejich komponenty.

Řešení: Z patnácti existujících rozkladů $\perp = \{\{a\},\{b\},\{c\},\{d\}\}$, $\top = \{\{a,b,c,d\}\}$, $\sigma = \{\{a,b\},\{c,d\}\} = \{U,V\}$, $\tau = \{\{a,c\},\{b,d\}\} = \{W,X\}$, $\rho = \{\{a,d\},\{b,c\}\} = \{Y,Z\}$, $\pi = \{\{a\},\{b,c,d\}\}$, $\nu = \{\{b\},\{a,c,d\}\}$, $\mu = \{\{c\},\{a,b,d\}\}$, $\eta = \{\{d\},\{a,b,c\}\}$, $\alpha = \{\{a\},\{b\},\{c,d\}\}$, $\gamma = \{\{a\},\{c\},\{b,d\}\}$, $\varepsilon = \{\{a\},\{d\},\{b,c\}\}$, $\zeta = \{\{b\},\{c\},\{a,d\}\}$, $\varphi = \{\{b\},\{d\},\{a,c\}\}$, $\psi = \{\{c\},\{d\},\{a,b\}\}$, jsou jen první čtyři SV rozklady. SV(M) = $\{\perp, \top, \sigma, \tau\}$ a Hasseovský diagram svazu SVM má tento tvar:



Obr. 4.18

Vzhledem k tomu, že $\sigma*\tau = \sigma*\rho = \tau*\rho = \perp$, jsou (kromě jiných) možné i tyto morfologické struktury dekompozic,



Obr. 4.19

a dynamické struktury jejich komponent jsou definovány následujícími tabulkami vektorových funkcí:

$\mathbf{M}_\sigma$:	A	B	C	D	E	F
U	U,(U,A)	U,(U,B)	V,(U,C)	U,(U,D)	U,(U,E)	U,(U,F)
V	U,(V,A)	V,(V,B)	V,(V,C)	V,(V,D)	V,(V,E)	V,(V,F)

$\mathbf{M}_\tau$:	A	B	C	D	E	F
W	W,(W,A)	W,(W,B)	W,(W,C)	X,(W,D)	W,(W,E)	W,(W,F)
X	X,(X,A)	W,(X,B)	X,(X,C)	X,(X,D)	X,(X,E)	X,(X,F)

$\mathbf{M}_{\tau T}$:	U,A	U,B	U,C	U,D	U,E	U,F	V,A	V,B	V,C	V,D	V,E	V,F
W	W,#	W,#	W,#	X,#	W,0	W,0	W,#	W,#	W,#	X,#	W,0	W,1
X	X,#	W,#	X,#	X,#	X,0	X,1	X,#	W,#	X,#	X,#	X,1	X,1

$\mathbf{M}_{\rho 1}$:	U,A	U,B	U,C	U,D	U,E	U,F	V,A	V,B	V,C	V,D	V,E	V,F
Y	Y,#	Y,#	Z,#	Z,#	Y,0	Y,0	Z,#	Z,#	Y,#	Y,#	Y,1	Y,1
Z	Z,#	Y,#	Y,#	Z,#	Z,0	Z,1	Y,#	Z,#	Z,#	Y,#	Z,0	Z,1

$\mathbf{M}_{\sigma T}$:	W,A	W,B	W,C	W,D	W,E	W,F	X,A	X,B	X,C	X,D	X,E	X,F
U	U,#	U,#	V,#	U,#	U,0	U,0	U,#	U,#	V,#	U,#	U,0	U,1
V	U,#	V,#	V,#	V,#	V,0	V,1	U,#	V,#	V,#	V,#	V,1	V,1

$\mathbf{M}_{\rho 2}$:	W,A	W,B	W,C	W,D	W,E	W,F	X,A	X,B	X,C	X,D	X,E	X,F
Y	Y,#	Y,#	Z,#	Z,#	Y,0	Y,0	Z,#	Z,#	Y,#	Y,#	Y,1	Y,1
Z	Y,#	Z,#	Z,#	Y,#	Z,0	Z,1	Z,#	Y,#	Y,#	Z,#	Z,0	Z,1

Parciální dekódovací funkce ω^{-1}: $(\sigma\times I)\times(\tau\times I) \rightarrow O$ v paralelní dekompozici je definována následující tabulkou, v níž prázdná políčka lze vyplnit libovolně:

ω^{-1}:	U,A	U,B	U,C	U,D	U,E	U,F	V,A	V,B	V,C	V,D	V,E	V,F
W,A	#						#					
W,B		#						#				
W,C			#						#			
W,D				#						#		
W,E					0						0	
W,F						0						1
X,A	#						#					
X,B		#						#				
X,C			#						#			
X,D				#						#		
X,E					0						1	
X,F						1						1

Úloha 4.6: Které další dekompozice Booleovského kalkulátoru, neuvedené v řešení úlohy 4.5 jsou možné? Najděte je a definujte dynamické struktury jejich komponent.

Příklad 4.7: Následující obrázek (Obr. 4.20) ilustruje postup získávání dekompozice automatu v případě složitých Hasseovských diagramů svazu SVM. Silně vyznačený integrální automat M definovaný na množině S vnitřních stavů má Hasseovský diagram svého svazu SVM nakreslen nad sebou. Mezi nejvýše položeným triviálním (nejhrubším) SV rozkladem $\lceil S$ a nejníže položeným triviálním (nejjemnějším) SV rozkladem $\perp S$ leží pět různých netriviálních SV rozkladů A, B, C, D, E množiny S.

Při konstrukci nejjemnějších dekompozic je třeba začínat zdola, tj. od rozkladu E. Ten umožní sériovou dekompozici typu $M = M_E,M_1$ zakreslenou napravo od M. Automat M_E (image machine) je definován na stavovém prostoru E (jeho stavy jsou třídy rozkladu E množiny S). Přechodem od M k M_E jsme přešli od množiny S k množině $E = \{\langle s\rangle_E: s\in S\}$, v níž třídy $\langle s\rangle_E$ jsou nejmenšími a dále nedělitelnými prvky (novými elementárními stavy). Hasseovský diagram svazu SVM_E automatu M_E odvodíme z Hasseovského diagramu svazu SVM tím, že v něm vypustíme rozklady, z nichž nevede „klesající cesta" do E; v takto vzniklém podgrafu jsou $\perp E$ a $\lceil E$ nové triviální rozklady, a index E u zbylých rozkladů vyjadřuje fakt, že se jedná o SV rozklady množiny E. V našem případě dostaneme druhý diagram zleva.

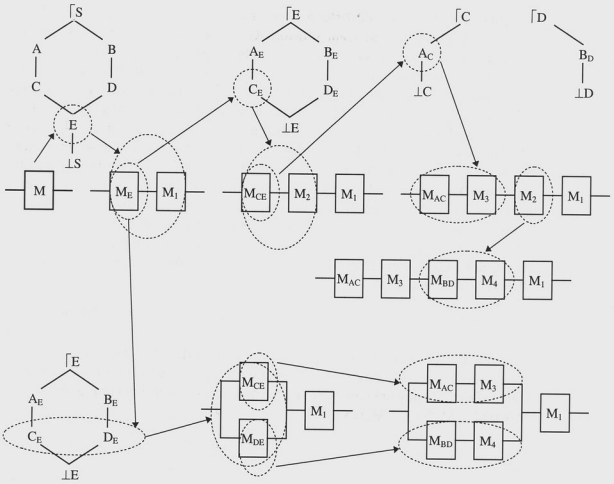
Netriviální SV rozklady v nově vzniklém diagramu lze využít k další dekompozici automatu M_E. Tvar diagramu říká, že neporovnatelné rozklady C_E a D_E jsou vzájemně ortogonální (existence přímých spojů s $\perp E$). To umožňuje jak sériovou tak paralelní dekompozici automatu M_E.

V sériové variantě dekompozice automatu M_E (horní část obrázku) jsme v našem případě ze dvou možností (C_E či D_E) zvolili SV rozklad C_E k dekompozici M_E =

M_{CE},M_2, čímž jsme dostali třístupňovou sériovou dekompozici $M = M_{CE},M_2,M_1$. M_{CE} je definován na stavovém prostoru $C = \{\langle x\rangle_C: x \in E\}$, což je SV rozklad rozkladu E. Z Hasseovského diagramu jeho svazu SVM_{CE} (třetí graf zleva) odvozeného z diagramu svazu SVM dle výše uvedených zásad je zřejmá existence netriviálního SV rozkladu množiny C - je jím SV rozklad A_C. To umožňuje dekomponovat M_{CE} na M_{AC},M_3 a dospět k dekompozici $M = M_{AC},M_3,M_2,M_1$. Z uvedené posloupnosti Hasseovských diagramů vyplývá, že automaty M_{AC}, M_3 a M_1 dále dekomponovat nelze (neexistuje netriviální SV rozklad jejich stavového prostoru). U automatu M_2 jsme měli možnost zvolit za jeho stavový prostor množinu $D = \{\langle x\rangle_D: x \in E\}$. Pokud jsme volili jinak, ani M_2 dekomponovatelný není. V opačném případě (D je stavovým prostorem automatu M_2) M_2 dekomponovatelný je. Vyplývá to z existence netriviálního SV rozkladu B_D množiny D (viz poslední diagram vpravo). Dekompozicí M_2 pak dostaneme již dále nedekomponovatelný výsledek, kterým je pětistupňová sériová dekompozice $M = M_{AC},M_3,M_{BD},M_4,M_1$.

V paralelní variantě dekompozice automatu M_E (spodní část obrázku) byla pro dekompozici automatu M_E využita existence ortogonálních SV rozkladů C_E a D_E rozkladu E. Tak vznikla dekompozice $M = (M_{CE}||M_{DE}),M_1$. Další dekompozice automatů M_{CE} a M_{DE} proběhla analogicky s výše popsanou sériovou variantou. Dále nedekomponovatelným výsledkem je pak sérioparalelní dekompozice $M = ((M_{AC},M_3)||(M_{BD},M_4)),M_1$.[14]

[14] Automaty M_{BD} a M_4 mají v sériové i sérioparalelní variantě shodné stavové prostory, ale definovány jsou různě (v sériové variantě jsou komponenty „tail machine" M_2, v sérioparalelní variantě jsou komponenty „image machine" M_{DE}).



Obr. 4.20

5. Chování konečného automatu

5.1. Úvod

V článku 4.4. jsme v rámci algebraického přístupu (po zohlednění symboliky zavedené v lekci 3.) nazvali elementárním chováním konečného automatu množinu $\{(i, \beta(s, i)): i \in I, s \in S\} \subseteq I \times O$, respektive $\{(i, \lambda(s)): i \in I, s \in S\} \subseteq I \times O$ pro Mealyho, respektive Mooreův automat. Vzhledem k tomu, že $I \times O$ je konečná množina, je možno elementární chování definovat výčtem. Výběr relevantních dvojic zde provádí vnitřní mechanismy, znázorněné funkčními schématy v článku 3.3.

Zobecnit elementární chování na chování znamená zjišťovat reakce systému nejen na prvky konečné množiny I, ale na všechny řetězce, které lze z jejích prvků vytvořit, a těch je nekonečně mnoho. Chování konečného automatu je tak popsáno nekonečnou množinou dvojic řetězců shora neohraničené délky. Tuto množinu nelze definovat výčtem, ale lze ji definovat rekurzivně. Co to konkrétně znamená se dozvíme hned v následujícím článku.

5.2. Rekurzivní definice chování konečného automatu

Znát chování konečného automatu znamená umět ke každé konečné posloupnosti (řetězci) vstupních podnětů přiřadit relevantní řetězec výstupních reakcí. Za tím účelem je třeba nejprve definovat řetězec a pak zobecnit popis vnitřního mechanismu konečného automatu tak, aby byl aplikovatelný na vstupní řetězce. Nástroji k tomuto účelu příhodnými jsou **rekurzivní definice** množiny, jíž lze definovat prvky nekonečných množin, a **rekurzivní rozšíření** operace, jímž lze definovat operace nad nimi.

Rekurzivní definice obecně sestává ze tří částí: *báze* (v ní se výčtem definuje konečný počet prvků), *rekurze* (pravidla jak z daných prvků vytvořit nové), a *ohraničení* (prvky nezařazené do báze ani nevzniklé rekurzí do množiny nepatří). V našem případě jsou relevantní následující definice:

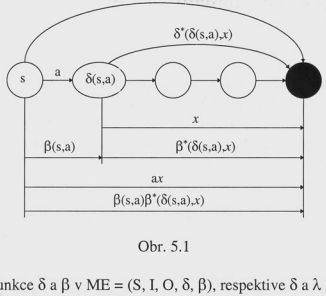
- **Rekurzívní definice řetězce**:
  1) **Báze**: Každý prvek konečné množiny (abecedy) je řetězec.
  2) **Rekurze**: Jsou-li x, y řetězce, pak jejich spojení (zřetězení), označené xy je řetězec. Je-li x řetězec, pak každé ε, pro něž platí $\varepsilon x = x\varepsilon = x$ je řetězec a nazveme ho **prázdný řetězec**.
  3) **Ohraničení**: Jinak už nic není řetězec.

  Množinu všech řetězců nad abecedou I, resp. O, označíme I^*, resp. O^*.[1] Délku řetězce x označíme $|x|$ a budeme jí rozumět počet písmen abecedy obsažených v x. Platí:
  $|xy| = |x| + |y|$, $|\varepsilon| = 0$.

- Princip **rekurzivního rozšíření přechodové a výstupní funkce** konečného automatu je znázorněn na Obr. 5.1:

---

[1] Alternativní algebraická definice množiny řetězců nad abecedou I: $I^* = \bigcup_{n=0}^{\infty} I^n$, kde I^n označuje n-násobný kartézský součin množiny I ($I^n = I \times I \times I \times \ldots \times I$) a $I^0 = \emptyset$ (prázdná množina).



Obr. 5.1

V souladu s ním funkce δ a β v ME = (S, I, O, δ, β), respektive δ a λ v MO = (S, I, O, δ, λ) zobecníme na δ^*: S×I* → S a β^*: S×I* → O* takto:

$$\delta^*(s, \varepsilon) = s, \quad \delta^*(s, ax) = \delta^*(\delta(s, a), x);$$

$$\beta^*(s, \varepsilon) = \varepsilon, \quad \beta^*(s, ax) = \beta(s, a)\beta^*(\delta(s, a), x), \quad \text{kde } a \in I, s \in S, x \in I^*.$$

V případě MO 1. druhu $\beta(s, a) = \lambda(s)$, u MO 2. druhu $\beta(s, a) = \lambda(\delta(s, a))$.

Chováním konečného automatu pak rozumíme množinu $\{(y, \beta^*(s, y)): s \in S, y \in I^*\}$.

5.3. Simulace a ekvivalence chování

Nechť s a s´ jsou různé stavy téhož automatu M nebo stavy ve dvou různých automatech M_1 a M_2 v nichž $I_1 = I_2 = I$ a $O_1 = O_2 = O$. Stavy s, s´ nazveme **ekvivalentními stavy**, právě když platí $\beta^*(s, x) = \beta^*(s´, x)$ nebo $\beta_1^*(s, x) = \beta_2^*(s´, x)$ pro všechna $x \in I^*$.

Automaty M_1 a M_2 nazveme **ekvivalentními automaty**, jestliže ke každému stavu v M_1 existuje ekvivalentní stav v M_2 a naopak. Je zřejmé, že ekvivalentní automaty produkují shodné chování, tj. $\{(y, \beta_1^*(s, y)): s \in S_1, y \in I^*\} = \{(x, \beta_2^*(s, x)): s \in S_2, x \in I^*\}$, a proto (v případě jejich počátečního nastavení do ekvivalentních stavů) budou vzájemně funkčně *zaměnitelné*.

V případě, kdy ke každému stavu v M_1 existuje ekvivalentní stav v M_2 a naopak tomu tak není (tj. v M_2 existuje stav, nemající ekvivalentní stav v M_1), je

$$\{(y, \beta_1^*(s, y)): s \in S_1, y \in I^*\} \subset \{(x, \beta_2^*(s, x)): s \in S_2, x \in I^*\}.$$

Pak automat M_2 dokáže ve funkci plně nahradit automat M_1, ale naopak tomu tak není (automat M_1 „umí méně" než M_2). Říkáme, že M_2 (na podmnožině svých stavů, ekvivalentních se stavy v S_1) *simuluje* chování automatu M_1.

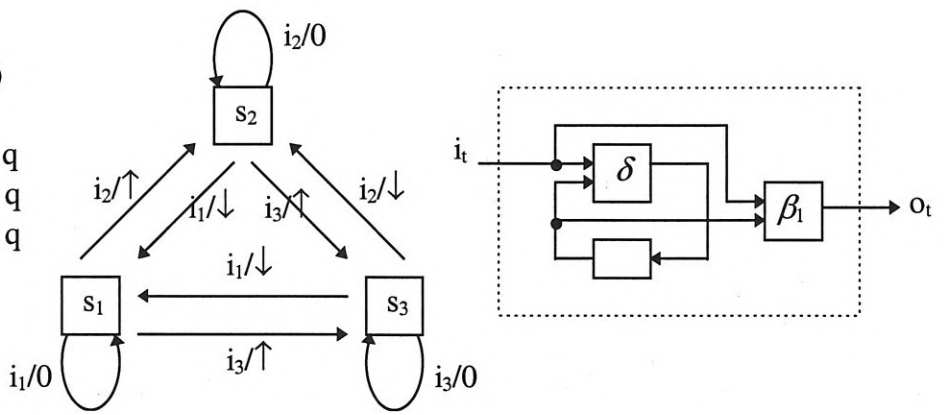
5.4. Konstrukce ekvivalentního MO k danému ME.

V lekci 3. jsme poznali, že Mooreův automat je speciálním případem Mealyho automatu. Od Mooreova modelu přejdeme k Mealyho modelu jen tím, že přeneseme hodnotu výstupu z uzlu na hrany. U MO 1. druhu přenášíme ve směru šipek (např. v grafu posuvného registru - viz čl. 3.4.), u MO 2. druhu proti směru šipek (např. v grafu modelu otevírání dveří

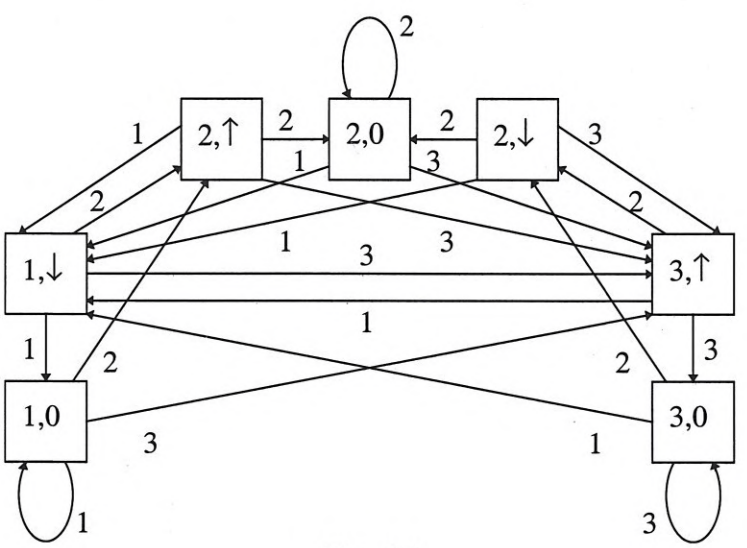
výtahu - viz příklad 3.1c). Toto jsou jen formální změny, v pětici definující automat tím došlo jen ke změně označení výstupní funkce (z λ na β). Že nově vzniklý automat je s původním ekvivalentní je na první pohled zřejmé.

Jak využít zmíněnou techniku přenosu výstupních hodnot, tentokrát naopak z hran do uzlu, ke konstrukci ekvivalentního MO k danému ME? Vysvětlíme si to na příkladu transformace ME automatu pro ovládání chodu motoru výtahu (viz Obr. 3. v příkladu 3.1b - výchozí graf je zopakován na následujícím obrázku) do ekvivalentního MO automatu 2. druhu. Jelikož v tomto případě jde o transformaci ME → MO2, půjde o přenos z hran do uzlu cílového (tj. ve směru šipek).

$A_1 = (S, I, O_1, \delta, \beta_1)$
$O_1 = \{\uparrow,\downarrow,0\}$
$\beta_1(s_p,i_q) = 0$ pro $p = q$
$\beta_1(s_p,i_q) = \uparrow$ pro $p < q$
$\beta_1(s_p,i_q) = \downarrow$ pro $p > q$



Z grafu tohoto ME je zřejmé, že posun po hranách stávajícího grafu provést nelze. Do stavu s_1 vstupují hrany označené výstupními hodnotami 0 a ↓, do stavu s_2 vstupují hrany označené výstupními hodnotami 0,↓,↑, do stavu s_3 vstupují hrany označené výstupními hodnotami 0 a ↑. Proto rozštěpíme každý ze stavů s_1 a s_3 na dva substavy, stav s_2 na tři substavy, a každý substav označíme jak číslem podlaží ze kterého vznikl, tak symbolem toho, zda do něj kabina výtahu přijela zdola (↑), shora(↓), či v něm zůstala stát (0). Z každého substavu pak vychází tři orientované hrany, označené vstupním podnětem (kvůli větší přehlednosti označíme podnět jen indexem stisknutého tlačítka, které uvedlo kabinu do pohybu) a vedoucí do substavu označeného číslem cílového podlaží spolu s adekvátním směrem pohybu kabiny. Dostaneme tak následující graf Mooreova automatu 2. druhu (MO2):



Obr. 5.2

Každý stav tohoto MO2 je ekvivalentní s tím stavem v původním ME, jehož štěpením vznikl, a naopak, každý stav ME je ekvivalentní se všemi substavy MO2 vzniklými jeho rozštěpením. Proto jsou ME a MO2 ekvivalentními automaty.

Zobecníme-li uvedený postup a popíšeme-li jeho kroky aparátem abstraktní algebry, můžeme výsledek formulovat následovně:

K danému Mealyho automatu ME = (S, I, O, δ, β) vždy existuje ekvivalentní Mooreův automat 2. typu MO2 = (S´, I, O, δ´, λ), v němž
S´ = S×O,
$\delta' = (\delta,\beta)^{\circ}\pi_{13}$: S×O×I → S×O, δ´((s, o), i) = (δ(s, i), β(s, i)), s ∈ S, i ∈ I, o ∈ O;
$\lambda = \pi_2$: S×O → O, λ(s, o) = o;
a v němž projekce π_1: S×O → S přiřazuje stavům z S´ ekvivalentní stavy v S.

5.5. Klidový a autonomní režim automatu

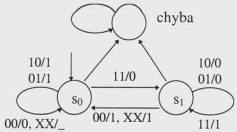
Vraťme se k příkladu paralelní sčítačky dvojic binárních čísel z článku 4.4. (viz Obr. 4.9). Tato sčítačka je nedokonalá tím, že nepozná, kde operandy začínají a kde končí, a proto donekonečna tiskne výsledky. Neví, kdy má se sčítáním začít a kdy má přestat sčítat. Chceme-li to napravit, potřebujeme kromě dvou *aktivních* vstupních signálů 0 a 1 ještě nějaký *klidový* vstupní signál X, který by sčítačce řekl, aby nereagovala.[2] V realizacích automatu bývá nepřítomnost podnětu vždy signalizována přítomností nějakého klidového signálu X.[3] Zcela analogicky je třeba i množinu výstupních reakcí automatu doplnit o příslušnou klidovou výstupní reakci, kterou automat reaguje na permanentní přítomnost klidového vstupního signálu X. V případě naší sčítačky ji označíme symbolem „_“. Při ní se papír neposouvá a nic se netiskne.

V dalším budeme předpokládat, že u sčítanců nestejné délky je kratší sčítanec zepředu doplněn nulami na délku delšího sčítance. Z pohledu naší sčítačky pak součet 38 + 7 = 45 by měl v jejím binárním provedení vypadat následovně:
X X X 1 0 0 1 1 0 X X X
X X X 0 0 0 1 1 1 X X X
_ _ _ 1 0 1 1 0 1 _ _ _ výstup ME
Na obou vstupech sčítačky se musí klidový signál X vyskytovat současně. Pokud tomu tak není, jde o poruchu v datech. Při jejím výskytu by automat měl přejít do zvláštního absorpčního stavu a ohlásit chybu v datech.

Po zohlednění všech výše uvedených požadavků původní graf přechodové a výstupní funkce automatu paralelní sčítačky dostane následující podobu (do chybového stavu automat přejde při vstupech X0, 0X, X1, 1X):

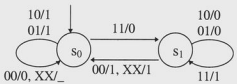
[2] V praxi to lze realizovat třemi hodnotami vstupního signálu (1 - kladný potenciál, 0 - záporný potenciál, X - nulový potenciál) a vstupním obvodem typu „polarizované relé“, který na nulový potenciál nereaguje. Zachovat binární signály na všech vstupních portech umožňuje jiné řešení, využívající zvláštní binární řídící vstup, kterým se sčítačka zapíná a vypíná. Klidovým signálem je pak každá kombinace s hodnotou 0 na řídícím vstupu (viz Příklad 5.4 ve cvičení k této lekci).

[3] V případě výtahu je klidovým signálem X nestlačení žádného z ovládacích tlačítek.



Obr. 5.3

V dalším budeme předpokládat, že k chybám v datech nedochází, a proto vezmeme v úvahu jen tento subautomat:

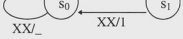


Obr. 5.4

Sčítačka realizující tento subautomat dokáže sečíst dvě libovolně velká binární čísla.

Nyní si položme otázku: *„Je možno podobným způsobem sestrojit paralelní násobičku, která by dokázala libovolně velká binární čísla vynásobit?“* Abychom mohli odpovědět, musíme prozkoumat, co se děje v konečném automatu, vystaveném delší dobu působení klidového signálu. Paralelní násobička totiž musí v konečné fázi výpočtu v takovém režimu pracovat. Má-li např. vynásobit číslo 2^N sebou samým, bude výsledkem číslo 2^{2N} ($2^N \cdot 2^N = 2^{2N}$). V binárním kódu bude aktivní vstupní sekvencí na obou vstupech N nul zakončených jedničkou, na výstupu pak dostaneme sekvenci 2N nul zakončených jedničkou. Posledních N cifer výsledku tak automat na výstup vydá při klidovém signálu XX na svém vstupu.

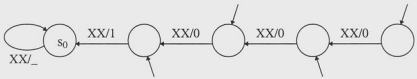
Takovému režimu práce říkáme **autonomní režim** a automat se v něm chová *autonomně*, tj. tak, že na klidový vstupní signál odpoví na výstupu signálem aktivním.[4] Autonomní režim konečného automatu lze znázornit podgrafem grafu přechodové a výstupní funkce, v němž jsou zobrazeny jen přechody a reakce na klidový podnět. V případě výše uvedené binární sčítačky na bázi Mealyho automatu by šlo jen o jeden přechod mezi dvěma stavy, tedy o tento podgraf:



Obr. 5.5

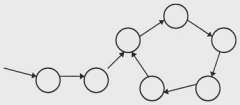
[4] Odpoví-li automat na vstupní klidový signál výstupním signálem klidovým, říkáme, že pracuje v **klidovém režimu**. Na rozdíl od **autonomního režimu** klidový režim automatu není zajímavý, a proto bývá v popisu automatu zcela ignorován (nemůže-li automat pracovat v autonomním režimu, pak klidový podnět a klidovou reakci v množinách I a O automatu zpravidla nezohledňujeme).

V případě námi zvažované paralelní násobičky by automat ve finální fázi násobení čísla 2^N sebou samým musel projít segment podgrafu svého autonomního režimu sestávající z řetězce N + 1 vnitřních stavů znázorněného v následujícím obrázku:



Obr. 5.6

Z podstaty věci vyplývá, že pokud automat, pracující v autonomním režimu, na své pouti řetězcem stavů vstoupí do stavu, kterým již prošel, zacyklí se - viz Obr. 5.7 - a z nekonečného opakování cyklu jej pak může vysvobodit jen příchod aktivního podnětu.



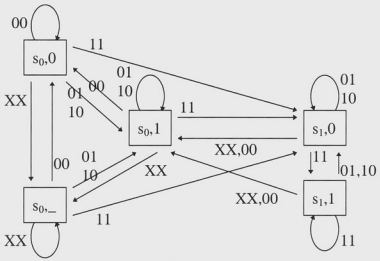
Obr. 5.7

Z toho ale bezprostředně plyne, že paralelní násobička, která by dokázala vynásobit libovolně velká binární čísla, nemůže být konečným automatem. Kdyby jím byla, měla by konečný počet stavů. A pokud by počet jejích stavů byl menší než číslo N (a takových čísel N je nekonečně mnoho), pak při násobení čísla 2^N sebou samým by ve zmíněné finální fázi výpočtu musela projít řetězcem N + 1 svých stavů. V takovém řetězci by se nutně některé stavy musely opakovat, čímž by vznikl cyklus, ve kterém opakovaně na výstup vydává jen nuly, a to není správný výsledek.

Došli jsme tak k závěru, který lze vyjádřit takto: I když konečný automat libovolně velká čísla sečíst umí, vynásobit libovolně velká čísla neumí.

Cvičení k lekci 5.

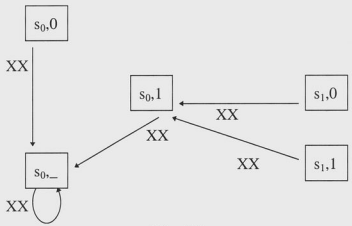
Příklad 5.1: Grafem přechodové a výstupní funkce Mooreova automatu 2. druhu (MO2), který je ekvivalentním automatem k Mealyho subautomatu paralelní sčítačky na Obr. 5.4 je tento graf:



Obr. 5.8

Příklad 5.2: Grafem přechodové a výstupní funkce automatu MO1 (Mooreův automat 1. druhu) ekvivalentního k Mealyho subautomatu paralelní sčítačky na Obr. 5.4 je stejný graf jako na Obr. 5.8 s tím, že zápis výstupu bude v MO1 oproti MO2 o takt opožděn (MO1 připisuje výsledek operace k následujícímu stavu).

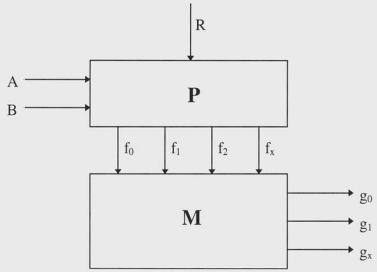
Příklad 5.3: Autonomní režim automatů MO2 a MO1 z předchozích dvou příkladů je znázorněn na Obr. 5.9:



Obr. 5.9

Příklad 5.4 (varianta paralelní sčítačky dvojic binárních čísel s binárními vstupními a výstupními signály): Vstupní obvody paralelní sčítačky dle Obr. 5.4 rozlišují na každém ze dvou vstupních portů tři úrovně signálu s hodnotami 0, 1 a X. Výstupní obvody pak na jediný výstupní port dodávají tři úrovně signálu (hodnoty 0, 1 a _). V technické praxi se snáze pracuje s dvouhodnotovými (binárními) signály, k jejichž dalšímu zpracování lze s výhodou využít aparát dvouhodnotové logiky -

viz lekce 6. Blokové schéma jedné z možných variant naší paralelní sčítačky pracující výhradně s binárními signály je uvedeno na Obr. 5.10:



RAB	000	001	010	011	100	101	110	111
$f_0f_1f_2f_x$	0001	0001	0001	0001	1000	0100	0100	0010

Obr. 5.10

V něm A a B jsou binární vstupy operandů a R je binární řídící vstup. Při R = 0 je každá kombinace hodnot na vstupech A a B pokládána za klidový vstup. Převodník P transformuje vstupní trojice binárních hodnot RAB do čtveřic vstupních binárních hodnot $f_0f_1f_2f_x$ automatu M dle uvedené převodní tabulky. V každém kroku funkce automatu je „živý" právě jeden jeho vstup (vektory (f_0,f_1,f_2,f_x) jsou jednotkovými vektory). Pořadí „živého" vstupu nese informaci o tom, jedná-li se o vstup klidový (f_x), a v opačném případě nese informaci i o počtu aktuálně sčítaných jedniček v operandech (počet aktuálně sčítaných jedniček je dán indexem „živého" vstupu).

Konečný automat M pak podle své tabulky přechodové a výstupní funkce transformuje vstupní jednotkové vektory (f_0,f_1,f_2,f_x) na výstupní jednotkové vektory (g_0,g_1,g_x). Index aktuálně „živého" výstupního portu informuje o aktuálně prováděné činnosti: g_0 - tisk 0 na posouvající se papír, g_1 - tisk 1 na posouvající se papír, g_x - papír se neposouvá a nic se netiskne (klidový stav). Na Obr. 5.11 vidíme tabulku přechodové a výstupní funkce (pro vektorovou funkci $[\delta,\beta]$) Mealyho automatu M:

S \ I	1000	0100	0010	0001
s_0	$(s_0,100)$	$(s_0,010)$	$(s_1,100)$	$(s_0,001)$
s_1	$(s_0,010)$	$(s_1,100)$	$(s_1,010)$	$(s_0,010)$

Obr. 5.11

Na Obr. 5.12 je tabulka přechodové a výstupní funkce k němu ekvivalentního MO2:

S \ I	1000	0100	0010	0001	λ
$(s_0,100)$	$(s_0,100)$	$(s_0,010)$	$(s_1,100)$	$(s_0,001)$	100
$(s_0,010)$	$(s_0,100)$	$(s_0,010)$	$(s_1,100)$	$(s_0,001)$	010
$(s_0,001)$	$(s_0,100)$	$(s_0,010)$	$(s_1,100)$	$(s_0,001)$	001
$(s_1,100)$	$(s_0,010)$	$(s_1,100)$	$(s_1,010)$	$(s_0,010)$	100
$(s_1,010)$	$(s_0,010)$	$(s_1,100)$	$(s_1,010)$	$(s_0,010)$	010

Obr. 5.12

Úloha 5.1: Řešení úlohy 4.4 definuje automat Booleovského kalkulátoru, řešení úlohy 4.5 definuje jeho pět možných dekompozic. U každé varianty dekompozice zjistěte všechny dvojice ekvivalentních stavů v integrální a dekomponované verzi. O jaký druh funkční podobnosti (zaměnitelnost či simulaci) se zde jedná?

Úloha 5.2: Totéž co v úloze 5.1 zjistěte u „staré tety Evženie“ (viz příklad 2.1) a její dekompozice (viz příklad 4.6). O jaký druh funkční podobnosti se jedná tam?

Úloha 5.3: K Mealyho automatu (ME) „staré tety Evženie“ (viz příklad 2.1) sestrojte tabulku přechodové a výstupní funkce ekvivalentního MO2.

Úloha 5.4: K Mealyho automatu (ME) Booleovského kalkulátoru (viz řešení úlohy 4.4) sestrojte tabulku přechodové a výstupní funkce ekvivalentního MO2.

Otázka 5.1: Dokáže číslicový počítač von Neumannovy koncepce vynásobit dvě libovolně velká čísla? Vysvětlete proč ano, či proč ne.

6. Systémy typu logická síť a jejich realizace

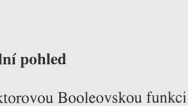
V této lekci se zaměříme na speciální kategorii kombinačních automatů snadno fyzicky realizovatelných logickým obvodem, která je nejčastěji užívanou bází pro implementaci a hardwarovou realizaci[1] konečných automatů.

6.1. Funkční a strukturální pohled

Z funkčního pohledu

je logická síť systémem s n, (resp. k) binárními vstupy (resp. výstupy), v němž vztah mezi vstupy a výstupy je popsán vektorovou Booleovskou funkcí, tj. zobrazením $f = (f_1,f_2, ... f_k): \{0,1\}^n \rightarrow \{0,1\}^k$. Funkce f je obvykle zadána tabulkou.

Příklad pro n = 3 a k = 2:



x	0	0	0	0	1	1	1	1
y	0	0	1	1	0	0	1	1
z	0	1	0	1	0	1	0	1
f_1	0	1	0	0	1	1	0	0
f_2	1	1	0	1	0	1	1	0

Obr. 6.1

Strukturální pohled

Vektorovou Booleovskou funkci $f = (f_1,f_2, ... f_k): \{0,1\}^n \rightarrow \{0,1\}^k$ lze rozložit do jejích složek, tj. do dílčích Booleovských funkcí $f_i: \{0,1\}^n \rightarrow \{0,1\}$, i = 1 až k. Každou z nich lze pak vyjádřit ekvivalentním výrazem Booleovy algebry, v němž jsou symboly proměnných vázány výhradně operacemi logického součtu, logického součinu a negace.[2] Přímo z tabulky funkce lze získat ekvivalentní Booleovský výraz buď v úplné disjunktivní normální formě (ÚDNF - součet součinů)[3] nebo v úplné konjunktivní normální formě (ÚKNF - součin součtů)[4]. Systém Booleovských výrazů všech složek popisuje strukturu logické sítě algebraicky. V našem příkladu (Obr. 6.1) je jednou z možností tato dvojice výrazů:

---

[1] Co rozumíme implementací a realizací automatu upřesníme v následující lekci.

[2] Negace je unární operací na množině {0,1} a značíme ji pruhem nad negovanou proměnnou či výrazem. Logický součet (disjunkce) a logický součin (konjunkce) jsou binárními operacemi na množině {0,1} a použijeme pro ně běžných symbolů aritmetických operací (+ a ·). Definiční tabulky těchto tří základních operací lze nalézt na Obr. 6.4. Výraz Booleovy algebry je pak výsledkem syntézy příslušné Booleovské funkce z těchto tří základních operací.

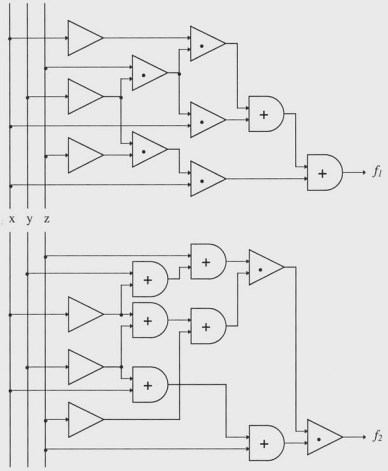
[3] Konstrukce ÚDNF pro f_i: Z tabulky vybereme všechny sloupce, v nichž $f_i = 1$. Ke každému takovému sloupci vytvoříme konjunkci (log. součin) všech vstupních proměnných, v níž proměnná, mající ve sloupci hodnotu 0, bude negována (jde o tzv. elementární konjunkce). ÚDNF funkce f_i pak bude disjunkcí (log. součtem) všech těchto elementárních konjunkcí. Přínos ÚDNF Booleovské funkce je v tom, že dává dokonalý přehled o kombinacích hodnot vstupních proměnných, které funkce připouští.

[4] Konstrukce ÚKNF pro f_i: Z tabulky vybereme všechny sloupce, v nichž $f_i = 0$. Ke každému takovému sloupci vytvoříme disjunkci (log. součet) všech vstupních proměnných, v níž proměnná, mající ve sloupci hodnotu 1, bude negována (jde o tzv. elementární disjunkce). ÚKNF funkce f_i pak bude konjunkcí (log. součinem) všech těchto elementárních disjunkcí. Vztah ÚKNF a ÚDNF: Vytvořením ÚDNF pro všechny sloupce tabulky, v

$$f_1 = \overline{x} \cdot \overline{y} \cdot z + x \cdot \overline{y} \cdot \overline{z} + x \cdot \overline{y} \cdot z, \quad \ldots\ldots\ldots\ldots\ldots \text{ (ÚDNF } f_1\text{)}$$

$$f_2 = (x + \overline{y} + z) \cdot (\overline{x} + y + z) \cdot (\overline{x} + \overline{y} + \overline{z}). \quad \ldots\ldots\ldots \text{ (ÚKNF } f_2\text{)}$$

Každý Booleovský výraz v ÚDNF či ÚKNF lze graficky znázornit sítí vzájemně propojených logických modulů tří typů (invertorů, log. sčítaček a log. násobiček), provádějících příslušné Booleovské operace nad nejvýše dvěma vstupními proměnnými. Schéma zapojení modulů všech složek vektorové Booleovské funkce graficky znázorňuje strukturu logické sítě. V našem případě je jednou z možností tato struktura (Obr. 6.2):[5]



Obr. 6.2

Popsaný přechod od tabulky ke schématu zapojení modulů je procesem syntézy struktury logické sítě z popisu její funkce. A jelikož je syntéza úlohou víceznačnou (tj. má více možných řešení), lze na ni klást různé optimalizační požadavky.

---

nichž funkce $f_i = 0$, dostaneme ÚDNF negace funkce f_i. Negováním této ÚDNF dostaneme po aplikaci De Morganových pravidel pro negování výrazů (negací součtu je součin negací a negací součinu je součet negací) ÚKNF funkce f_i.

[5] Neoznačené moduly jsou invertory; sčítačky resp. násobičky jsou označeny symboly + resp. •.

6.2. Optimalizace struktury logické sítě

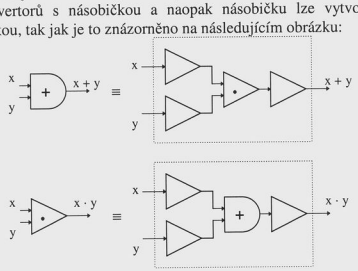
Jedním z optimalizačních požadavků kladených na strukturu logické sítě je požadavek homogenizace struktury (tj. požadavek minimalizace počtu *typů* modulů), druhým je minimalizace celkového počtu zapojených modulů. K homogenizaci struktury nás nutí převážně důvody výrobní, k minimalizaci počtu modulů zase důvody spíše ekonomické.

Homogenizace struktury

Každý soubor různých typů modulů, provádějících Booleovské operace nad nejvýše dvěma vstupními proměnnými a postačujících k syntéze struktury libovolné logické sítě, nazveme *úplným souborem modulů* (ÚSM). A každý ÚSM, který by po vypuštění některého typu modulu přestal být ÚSM, nazveme *minimálním úplným souborem modulů* (MÚSM). Z tvaru ÚDNF či ÚKNF Booleovské funkce vyplývá, že soubor tří typů modulů zmíněných v článku 6.1. [invertor, sčítačka a násobička] je ÚSM. Tento soubor ale není MÚSM. Vyplývá to z De Morganových zákonů Booleovy algebry ($\overline{x+y} = \overline{x} \cdot \overline{y}$, $\overline{x \cdot y} = \overline{x} + \overline{y}$), jejichž platnost dokazuje shoda 5. se 7., respektive 9. s 10. sloupcem následující tabulky hodnot Booleovských výrazů:[6]

1.	2.	3.	4.	5.	6.	7.	8.	9.	10.
x	y	$\overline{x}$	$\overline{y}$	$\overline{x} \cdot \overline{y}$	$x+y$	$\overline{x+y}$	$x \cdot y$	$\overline{x \cdot y}$	$\overline{x} + \overline{y}$
0	0	1	1	1	0	1	0	1	1
0	1	1	0	0	1	0	0	1	1
1	0	0	1	0	1	0	0	1	1
1	1	0	0	0	1	0	1	0	0

Negací výrazů na obou stranách De Morganových rovnic totiž dostáváme $x+y = \overline{\overline{x} \cdot \overline{y}}$, respektive $x \cdot y = \overline{\overline{x} + \overline{y}}$. Dle toho je možno disjunkci složit z jedné konjunkce a tří negací, respektive konjunkci složit z jedné disjunkce a tří negací. Proto i sčítačku lze vytvořit propojením tří invertorů s násobičkou a naopak násobičku lze vytvořit propojením tří invertorů se sčítačkou, tak jak je to znázorněno na následujícím obrázku:



Obr. 6.3

[6] Jedná se o důkaz ekvivalence Booleovských výrazů provedený tabulkovou metodou.

Soubory [invertor, sčítačka] či [invertor, násobička] už MÚSM jsou.

Kromě tří dosud zvažovaných logických modulů existují i další univerzální „stavební kameny", z nichž lze libovolné logické sítě stavět. Při požadavku „nejvýše dvou binárních vstupů" je kompletní bází pro jejich konstrukci úplný seznam všech šestnácti existujících Booleovských funkcí dvou proměnných, uvedený spolu s názvy jednotlivých funkcí a jejich ÚDNF na Obr. 6.4.

x	0	0	1	1	Úplná disjunktní normální forma funkce	Název funkce
y	0	1	0	1		
f_0	0	0	0	0	$f_0 = 0$	Nulová funkce
f_1	0	0	0	1	$f_1 = x \cdot y$	Logický součin
f_2	0	0	1	0	$f_2 = x \cdot \overline{y}$	Inhibice (x)
f_3	0	0	1	1	$f_3 = x \cdot \overline{y} + x \cdot y = x$	Opakování (x)
f_4	0	1	0	0	$f_4 = \overline{x} \cdot y$	Inhibice (y)
f_5	0	1	0	1	$f_5 = \overline{x} \cdot y + x \cdot y = y$	Opakování (y)
f_6	0	1	1	0	$f_6 = \overline{x} \cdot y + x \cdot \overline{y}$	Neekvivalence
f_7	0	1	1	1	$f_7 = \overline{x} \cdot y + x \cdot \overline{y} + x \cdot y = x + y$	Logický součet
f_8	1	0	0	0	$f_8 = \overline{x} \cdot \overline{y}$	Peirceova funkce
f_9	1	0	0	1	$f_9 = \overline{x} \cdot \overline{y} + x \cdot y$	Ekvivalence
f_{10}	1	0	1	0	$f_{10} = \overline{x} \cdot \overline{y} + x \cdot \overline{y} = \overline{y}$	Negace (y)
f_{11}	1	0	1	1	$f_{11} = \overline{x} \cdot \overline{y} + x \cdot \overline{y} + x \cdot y$	Implikace (y → x)
f_{12}	1	1	0	0	$f_{12} = \overline{x} \cdot \overline{y} + \overline{x} \cdot y = \overline{x}$	Negace (x)
f_{13}	1	1	0	1	$f_{13} = \overline{x} \cdot \overline{y} + \overline{x} \cdot y + x \cdot y$	Implikace (x → y)
f_{14}	1	1	1	0	$f_{14} = \overline{x} \cdot \overline{y} + \overline{x} \cdot y + x \cdot \overline{y}$	Shefferova funkce
f_{15}	1	1	1	1	$f_{15} = 1$	Jednotková funkce

Obr. 6.4

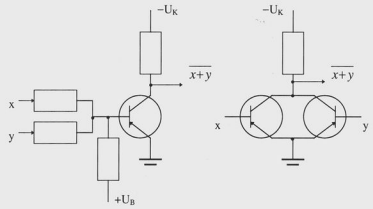
Funkce f_0 a f_{15} definují nulární operace (konstanty) na množině {0,1}, funkce f_3, f_5, f_{10} a f_{12} definují unární operace na množině {0,1}, ostatní funkce definují binární operace na množině {0,1}. Operaci f_1 již provádí naše násobička, operaci f_7 naše sčítačka, operace f_{10} i f_{12} provádí náš invertor s jedním vstupem. Stejně tak může jeden logický modul s jedním vstupem provádět jak operaci f_3, tak operaci f_5. Z celkem 14 různých typů logických modulů, provádějících uvedených 16 operací lze pak sestavit dvacet dvoučlenných MÚSM a dokonce dva jednočlenné.

Jednočlennými MÚSM jsou logické moduly provádějící Peirceovu funkci $f_8 = \overline{x} \cdot \overline{y} = \overline{x + y}$ nebo Shefferovu funkci $f_{14} = \overline{x \cdot y} = \overline{x} + \overline{y}$. Dokládá to Obr. 6.5, v němž symbol f značí Peirceův (f_8) či Shefferův (f_{14}) operátor realizovaný jedním integrálním logickým modulem (viz levé schéma). Po propojení jeho vstupů (prostřední schéma) se v obou případech modul chová jako invertor. Sériová kompozice dvou stejných modulů (pravé schéma) v případě Peirceova operátoru dává logickou sčítačku, v případě Shefferova operátoru logickou

násobičku. Z modulu tak lze sestavit dvojici, o níž už víme, že je MÚSM; tudíž sám je MÚSM.

Obr. 6.5

Významnou z hlediska aplikací je zejména Peirceova funkce, protože ji lze realizovat jednoduchými tranzistorovými obvody (viz Obr. 6.6):



Obr. 6.6

Kdybychom v logické síti našeho úvodního příkladu (Obr. 6.2) nejprve nahradili násobičky kompozicí invertorů se sčítačkou (Obr. 6.3), mohli bychom pak v nově vzniklé logické síti nahradit všechny invertory a sčítačky adekvátně propojenými Peirceovými moduly (Obr. 6.5). Tím bychom získali stejně fungující logickou síť s **homogenní strukturou**, v níž by jediným použitým typem logického modulu byl Peirceovův modul.

Minimalizace počtu modulů

Každý logický modul ve schématu struktury logické sítě provádí jednu Booleovskou operaci. Úpravou Booleovského výrazu, vedoucí ke snížení počtu potřebných operací (a neměnící původní vektorovou Booleovskou funkci) se proto sníží i počet modulů zapojených v síti.

V algebře Booleovských výrazů (Booleova algebra) se výrazy upravují podobně jako v běžné školské (Peanově) aritmetice: Vytýkáním společných částí sčítaných součinů před závorku či naopak odstraňováním závorek (roznásobováním součinu součtů). Těmito kroky získáváme dílčí výrazy, ke kterým existuje jednodušší ekvivalent, a tímto ekvivalentem je nahrazujeme.[7] Postupně lze takto dospět k ekvivalentnímu a z hlediska počtu potřebných logických modulů mnohem úspornějšímu Booleovskému výrazu.

[7] Máme na mysli zejména tyto ekvivalenty: $x + x = x \cdot x = x + 0 = x \cdot 1 = x$, $\overline{x} + x = x + 1 = 1$, $\overline{x} \cdot x = x \cdot 0 = 0$, $x + \overline{x} \cdot y = x + y$.

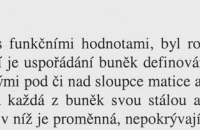
Tak např. struktura funkce $f_1 = \overline{x} \cdot \overline{y} \cdot z + x \cdot \overline{y} \cdot \overline{z} + x \cdot \overline{y} \cdot z$ vyjádřené v ÚDNF vyžaduje celkem 10 modulů (3 invertory, 5 násobiček, 2 sčítačky – viz Obr. 6.2). Výše zmíněnými algebraickými úpravami lze výraz postupně převést do tvaru:

$f_1 = \overline{x} \cdot \overline{y} \cdot z + x \cdot \overline{y} \cdot \overline{z} + x \cdot \overline{y} \cdot z = \overline{x} \cdot \overline{y} \cdot z + x \cdot \overline{y} \cdot (\overline{z} + z) = \overline{x} \cdot \overline{y} \cdot z + x \cdot \overline{y} = \overline{y} \cdot (\overline{x} \cdot z + x) = \overline{y} \cdot (x + z)$;

v tomto novém tvaru ji lze realizovat jen třemi moduly (1 invertor, 1 násobička, 1 sčítačka).

V technické praxi je oblíbeným nástrojem sloužícím k minimalizaci Booleovského výrazu tzv. **Karnaughova mapa**. Karnaughova mapa vznikla modifikací tabulky Booleovské funkce. Jak - to si opět vysvětlíme na příkladu funkce f_1. Na Obr. 6.7 vlevo vidíme její tabulku, vpravo jí odpovídající Karnaughovu mapu:

x	0	0	0	0	1	1	1	1
y	0	0	1	1	0	0	1	1
z	0	1	0	1	0	1	0	1
f_1	0	1	0	0	1	1	0	0



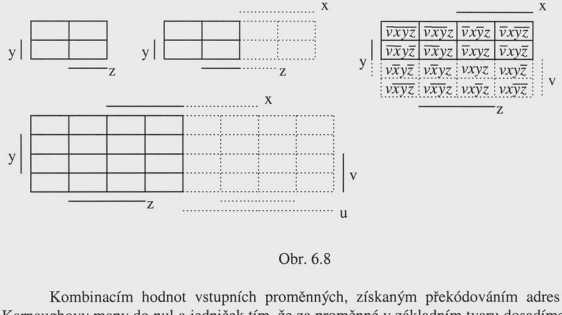
Obr. 6.7

Poslední řádek tabulky, sestávající z buněk s funkčními hodnotami, byl rozebrán a přerovnán do tvaru dvouřádkové matice vpravo. V ní je uspořádání buněk definováno pruhy pojmenovanými po vstupních proměnných, zakreslenými pod či nad sloupce matice a vlevo či napravo od jejích řádků. V takovém uspořádání má každá z buněk svou stálou adresu ve formě elementární konjunkce vstupních proměnných, v níž je proměnná, nepokrývající buňku svým pruhem, negována. Toto schéma nazýváme Karnaughovou mapou tří proměnných.

Na dalším obrázku vidíme, jak se přidáním další vstupní proměnné Karnaughova mapa zdvojuje (tečkovaná část);[8] navíc do buněk Karnaughovy mapy pro čtyři vstupní proměnné v, x, y, z byly vepsány jejich adresy. Z nich je zřejmé, že každá adresa se od čtyř sousedních liší jen v jediné proměnné.[9] A to platí ve všech takto odvozených Karnaughových mapách. Na této vlastnosti Karnaughových map je založena jednoduchá metoda minimalizace Booleovských výrazů popsaná níže.

[8] Karnaughova mapa dvou proměnných je čtvercová. Čtvercovou mapu lze zdvojit v libovolném ze čtyř směrů. Není-li mapa čtvercová, musí se zdvojením doplnit na čtverec. Pruh nové proměnné kryje právě všechny buňky nově vzniklé části. Pruhy stávajících proměnných se symetricky rozšíří i na nově vzniklou část.

[9] První a poslední řádek spolu v uvedeném smyslu sousedí. Lze si to představit tak, že rovinnou mapu srolujeme podle vodorovné osy do válcové plochy, na níž se první a poslední řádek dotknou. Přesně totéž lze říci o levém a pravém sloupci při rolování podle svislé osy.



Obr. 6.8

Kombinacím hodnot vstupních proměnných, získaným překódováním adres buněk Karnaughovy mapy do nul a jedniček tím, že za proměnné v základním tvaru dosadíme 1 a za negované proměnné 0, jsou v původní tabulce dané Booleovské funkce přiřazeny funkční hodnoty (1 nebo 0). Přenesením těchto funkčních hodnot do buněk Karnaughovy mapy získáme *Karnaughovu mapu dané Booleovské funkce*. V případě funkce f_1 (viz Obr. 6.7) je jí mapa uvedená vpravo.

Ve shodě s poznámkou pod čarou 3 ÚDNF f_1 dostaneme logickým součtem adres buněk Karnaughovy mapy s hodnotou 1. Jak už bylo řečeno, tuto verzi f_1 lze realizovat 11 funkčními moduly. Ve shodě s poznámkou pod čarou 4 ÚKNF f_1 dostaneme logickým součinem negovaných adres buněk Karnaughovy mapy s hodnotou 0.[10] K realizaci této verze je třeba 17 funkčních modulů (3 invertory, 10 sčítaček, 4 násobičky).

Avšak vzhledem k tomu, že v Karnaughově mapě se adresy sousedních buněk liší jen v jedné proměnné, je možno při odvozování výrazů pracovat místo s jednotlivými buňkami s bloky buněk[11], přičemž bloky se smějí i překrývat (tj. mohou mít neprázdný průnik). Tento postup vede k výrazům vyznačujícím se malým počtem operací. Zdrojem úspory operací je jednak to, že čím větší jsou bloky, tím méně jich je k popisu funkce zapotřebí, a jednak i to, že čím větší je blok, tím méně proměnných má jeho adresa.[12] Z uvedeného je zřejmé, že cesta k minimálnímu výrazu vede přes největší bloky.

Tak např. v Karnaughově mapě funkce f_1 (Obr. 6.7) vidíme dva dvoučlenné bloky jedniček: V posledním sloupci (adresa bloku je $x \cdot \bar{y}$) a v rozích spodního řádku (adresa bloku je $z \cdot \bar{y}$); proto $f_1 = x \cdot \bar{y} + z \cdot \bar{y}$. K realizaci této verze stačí jen 4 moduly (1 invertor, 1 sčítačka, 2 násobičky). Avšak vidíme tam i dva bloky nul: Čtyřčlenný v prostředních dvou sloupcích (adresa bloku je y) a dvoučlenný v horním řádku (adresa bloku je $\bar{x} \cdot \bar{z}$); proto $f_1 =$

[10] $f_1 = (x+y+z)\cdot(x+\bar{y}+z)\cdot(\bar{x}+\bar{y}+z)\cdot(x+\bar{y}+\bar{z})\cdot(\bar{x}+\bar{y}+\bar{z})$.

[11] Blokem buněk se rozumí každý na válcových plochách z poznámky 9 souvislý obdélník či čtverec sestávající z buněk obsazených stejnou funkční hodnotou, který v případě, že je pruhem nějaké proměnné pokryt částečně, je jím pokryt právě z poloviny. Blokem buněk tak např. může být i čtveřice rohových buněk Karnaughovy mapy, je-li jim přiřazena stejná funkční hodnota.

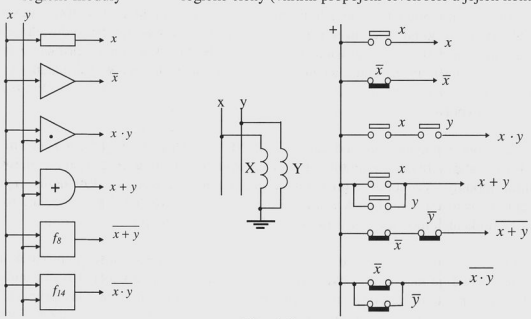
[12] V adrese bloku se nevyskytují proměnné, jejichž pruhy je blok kryt z poloviny.

$\overline{y}\cdot(x+z)$. Tato verze je ještě o jeden modul úspornější než verze předchozí. Je to tím, že v ní byl využit větší blok.

6.3. Logické obvody

Integrální systém fyzicky realizující logický modul nazveme (v souladu s běžnými zvyklostmi) **logickým členem**. A síť vzájemně propojených logických členů pak nazveme **logickým obvodem**. V dalším se zaměříme zejména na logické členy, které jsou realizací nějakého ÚSM (konkrétně opakovacího členu,[13] invertoru, násobičky a sčítačky) nebo MÚSM (modul Peirceovy či Shefferovy funkce). Logickému modulu, resp. logické síti (rovina teoretická), tak odpovídá stejně fungující logický člen, resp. logický obvod (rovina realizační).

Příklad realizace: Logické členy reléových spínacích obvodů



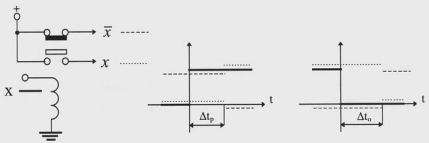
Obr. 6.9

Reakční doby logických obvodů

Na rozdíl od logických modulů, které jsou idealizací skutečnosti zejména v tom smyslu, že ignorují časoprostorové souvislosti, v logickém členu nastává při změně hodnot vstupních proměnných přechodový fyzikální děj, který je zakončen novým stabilním ustáleným stavem (např. přechod kotvy relé z jedné polohy do druhé, otevírání, respektive zavírání tranzistorů apod.). V důsledku tohoto přechodového děje se reakce výstupu logického členu na změnu vstupních signálů zpožďuje (oproti logickému modulu) o dobu Δt_r zvanou **reakční doba**. Reakční doba je dána charakterem vstupní změny a v obecném případě je pro různé vstupní změny i různě dlouhá.

[13] Opakovacími členy se v kratších větvích asynchronních obvodů zpožďuje signál (viz odstavec **Eliminace hazardů** níže).

Příklad reakčních dob: Reakční doby reléových kontaktů[14]



Obr. 6.10

Z uvedeného je zřejmé, že všechny logické členy na Obr. 6.9 mají (v závislosti na vstupní změně) reakční dobu Δt_o nebo Δt_p.

Charakteristickým rysem každého přechodového děje je jistá „setrvačnost" původního stabilního stavu. Navenek se tato setrvačnost projevuje tím, že logický člen obvykle nereaguje na vstupní impulsy s dobou trvání kratší než je délka jeho reakční doby.[15] Existence reakčních dob logických členů tak shora omezuje přípustnou frekvenci změn vstupních signálů.

Zpomalení reakce obvodu a omezení přípustné frekvence vstupních změn nejsou jedinými negativními důsledky existence reakčních dob. Dalším je možnost vzniku impulsů nesprávných hodnot na výstupech logických členů či logických obvodů. Na úrovni logických členů je konkrétní příčinou výskytu nesprávných hodnot nerovnost $\Delta t_o \neq \Delta t_p$. U posledních čtyř členů na Obr. 6.9 vzniká impuls nesprávného výstupu buď při vstupní změně $01 \rightarrow 10$ nebo $10 \rightarrow 01$, má šířku rovnou hodnotě $|\Delta t_o - \Delta t_p|$ a je za příslušnou vstupní změnou opožděn o dobu $\Delta t = \min(\Delta t_o, \Delta t_p)$. Není-li hlavní příčinou různosti reakčních dob Δt_o a Δt_p technická závada např. typu „lepení kontaktů" (kdy $\Delta t_o >> \Delta t_p$), je impuls velmi úzký ($\Delta t_o - \Delta t_p \approx 0$), a proto i neškodný (následné členy jej ignorují).[16] Podobné impulsy vznikající v důsledku různě dlouhých větví logických členů na úrovni logického obvodu (viz níže) již tak neškodné nejsou a vynucují si koncepční technická řešení.

Logický obvod vznikne sériovým, paralelním či sérioparalelním propojením logických členů. Lze v něm vysledovat různě dlouhé větve do série zapojených členů (cesty), kterými se signál vstupní změny souběžně šíří. Každá z těchto větví začíná některým vstupním členem a končí některým členem výstupním. Doba, kterou šířící se signál vstupní změny potřebuje k projití dané větve, je rovna součtu reakčních dob logických členů větev tvořících. Tuto dobu nazveme *reakční dobou příslušné větve*. Má-li logický obvod více různě dlouhých větví, pak nechť Δt_{rmin} (resp. Δt_{rmax}) je nejkratší (resp. nejdelší) zjištěná reakční doba větve. Je zřejmé, že pak Δt_{rmin} (resp. Δt_{rmax}) je také nejkratší (resp. nejdelší) možnou reakční dobou příslušného logického obvodu.

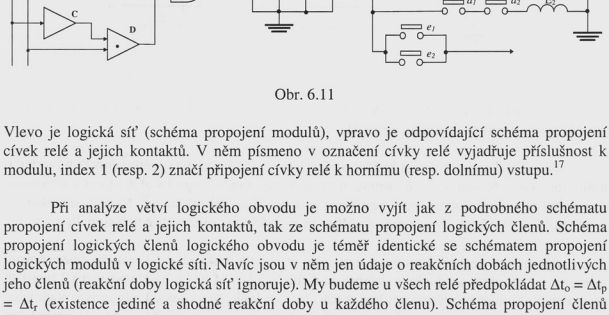
---

[14] Δt_p označuje dobu přítahu kotvy relé, Δt_o dobu odpadu kotvy relé.

[15] Ke změně stabilního stavu je zapotřebí vynaložit jistou energii. Předpokládá se, že energie, kterou systém k tomuto účelu vynakládá, je úměrná šířce vstupního impulsu stimulujícího příslušnou změnu. Je-li impuls kratší než reakční doba, skončí přísun této energie příliš brzy a stabilizační síly udrží systém v původním stavu.

[16] Z uvedených důvodů se v praxi obvykle snažíme vhodným napružením odtrhovacích pružin kotvy relé dosáhnout stavu $\Delta t_0 = \Delta t_p = \Delta t_r$.

Jednoduchý (spíše ilustrační než praktický) příklad vidíme v následujícím obrázku:



Obr. 6.11

Vlevo je logická síť (schéma propojení modulů), vpravo je odpovídající schéma propojení cívek relé a jejich kontaktů. V něm písmeno v označení cívky relé vyjadřuje příslušnost k modulu, index 1 (resp. 2) značí připojení cívky relé k hornímu (resp. dolnímu) vstupu.[17]

Při analýze větví logického obvodu je možno vyjít jak z podrobného schématu propojení cívek relé a jejich kontaktů, tak ze schématu propojení logických členů. Schéma propojení logických členů logického obvodu je téměř identické se schématem propojení logických modulů v logické síti. Navíc jsou v něm jen údaje o reakčních dobách jednotlivých jeho členů (reakční doby logická síť ignoruje). My budeme u všech relé předpokládat $\Delta t_o = \Delta t_p = \Delta t_r$ (existence jediné a shodné reakční doby u každého členu). Schéma propojení členů obvodu bývá přehlednější než schéma propojení cívek a kontaktů, proto z něj vyjdeme. Na první pohled v něm vidíme 4 různé větve: Nejkratší z nich vede přes členy D-E, ostatní 3 jsou stejně dlouhé; jedna vede přes C-D-E, zbylé 2 přes A-B-E. Okamžitě vidíme $\Delta t_{rmin} = 2 \cdot \Delta t_r$, $\Delta t_{rmax} = 3 \cdot \Delta t_r$. Zjištěná trojice hodnot Δt_r (reakční doba členu), Δt_{rmin} a Δt_{rmax} (nejkratší a nejdelší reakční doba obvodu) jsou významnými parametry určujícími možnosti praktického využití daného obvodu.

Výskyt nechtěné reakce logického obvodu (hazard)

V důsledku různě dlouhých reakčních dob jednotlivých větví, jimiž se vstupní změna souběžně obvodem šíří, dorazí některé složky nových podnětů vyvolaných změnou na vstupy výstupních členů obvodu dříve, jiné později. V mezidobí od příchodu první změny podnětu do poslední jsou tak výstupní členy vystaveny sérii přechodových kombinací vstupních podnětů, na níž reagují sérií přechodových výstupních reakcí. Při nepříznivé konstelaci[18] se tak mohou na výstupech obvodu přechodně vyskytnout impulsy nesprávných hodnot. Na rozdíl od situace na úrovni logických členů k tomu dojde i při $\Delta t_o = \Delta t_p = \Delta t_r$ uvnitř časového intervalu $\langle \Delta t_{rmin}, \Delta t_{rmax} \rangle$ po příslušné vstupní změně, a pro šířku Δt jednoho impulsu platí: $\Delta t_r \leq \Delta t \leq (\Delta t_{rmax} - \Delta t_{rmin})$. Tyto impulsy již neškodné nejsou, protože následné logické členy na ně reagují. Zvlášť nebezpečnými jsou v sekvenčních obvodech (viz pozn. pod čarou 19), v nichž se zpětnými vazbami udržují ve hře i delší dobu a spolu s dalšími dříve či později vzniklými impulsy nesprávného výstupu dokáží nepředvídatelným způsobem ovlivnit chování obvodu, případně

[17] Obvod realizuje logickou funkci $\overline{x + y} + \overline{x} \cdot y = \overline{x} \cdot \overline{y} + \overline{x} \cdot y = \overline{x} \cdot (\overline{y} + y) = \overline{x}$.

[18] Konstelací se zde rozumí souběh více faktorů, z nichž k nejvýznamnějším patří morfologická struktura logického obvodu, dynamické struktury logických členů do obvodu zapojených a charakter vstupní změny.

jej rozkmitat. Konstelace, vedoucí k nežádoucímu chování obvodu bývá v odborném žargonu označována jako **hazard**.

V našem jednoduchém logickém obvodu (Obr. 6.11) hazard vzniká jen při vstupní změně 01 → 00, která by výstup obvodu měnit neměla. Přesto jej na dobu Δt_r změní; ukážeme proč: Vstupní kombinace xy = 01 dodává na (horní,dolní) vstupy členu E podnět (0,1), vstupní kombinace xy = 00 dodává tamtéž podnět (1,0). Na oba tyto podněty člen E reaguje hodnotou 1, proto by měl vstupní změnu 01 → 00 ignorovat. V daném obvodu tomu ale tak není. Druhá složka nového podnětu (1,0), vyvolaného vstupní změnou 01 → 00 se šíří kratší větví, a proto dorazí na dolní vstup členu E o Δt_r dříve než první složka na horní vstup. Přechodně (po dobu Δt_r) se tak na vstupech členu E objeví podnět (0,0), na nějž člen E zareaguje výstupní hodnotou 0.

Eliminace hazardů

Identifikaci a následné eliminaci hazardů v logických obvodech se v technické praxi věnuje značná pozornost. Metody eliminace hazardů vychází z poznatku, že k nesprávnému výstupu dochází uvnitř časového intervalu $\langle \Delta t_{rmin}, \Delta t_{rmax} \rangle$ po příslušné vstupní změně, a že doby trvání přechodných podnětů nepřesáhnou délku tohoto intervalu. Hazardy lze pak eliminovat v podstatě třemi způsoby:

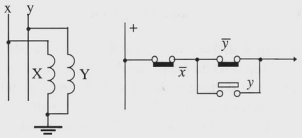
1. Volbou dlouhých reakčních dob koncových logických členů logického obvodu (zaváděním tzv. tlumících členů s reakční dobou $\Delta t_r > (\Delta t_{rmax} - \Delta t_{rmin})$).
2. Symetrizací obvodů, tj. vytvářením obvodů se stejně dlouhými větvemi. Symetrizace může být:
   - umělá - zařazování zpožďovacích členů (opakovačů) do kratších větví obvodu;
   - přirozená - konstrukce symetrických obvodů.
3. Vnější synchronizací kombinačních obvodů.[19]

ad 1. Je-li doba trvání přechodné vstupní kombinace kratší než reakční doba členu, logický člen na ni nezareaguje. V obvodu na Obr. 6.11 by stačilo prodloužit reakční dobu Δt_o relé E_2 snížením tlaku odtrhovací pružiny na kotvu relé. Zvýšením hmotnosti kotvy lze prodloužit jak Δt_o tak i Δt_p.

ad 2. Umělou symetrizaci bychom provedli zařazením opakovacího členu mezi proměnnou y a dolní vstup logického modulu **D**. Kvůli přirozené symetrizaci je třeba opustit koncepci zavedených modulů (negátoru, logického součtu a součinu), v nichž je u každého relé využit jen jeden kontakt, kontakty různých relé jsou pevně propojeny a není k nim přístup. A místo toho sestavovat logické obvody libovolným propojováním i více spínacích a rozpínacích kontaktů jednoho relé.[20] V rámci tohoto přístupu by stejnou funkci plnil tento symetrický obvod:

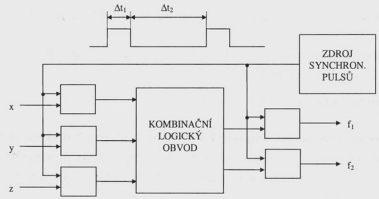
[19] Realizace dopředných logických sítí (tj. sítí bez zpětných vazeb, v nichž výstupy modulů jsou připojeny jen ke vstupům následujících modulů) bývají nazývány **kombinačními** logickými obvody. Kombinační logické obvody jsou systémy bez paměti a jejich výstup je jednoznačně určen poslední kombinací hodnot na vstupech obvodu. Realizace logických sítí se zpětnými vazbami (výstup modulu je připojen i na vstupy téhož modulu či na vstupy některých předchozích modulů) bývají nazývány **sekvenčními** logickými obvody. Sekvenční logické obvody jsou systémy s pamětí. Informace zakódovaná do kombinace vstupních hodnot se v nich díky zpětným vazbám udržuje delší dobu, výstup obvodu pak závisí na sekvenci vstupních podnětů a může být i nestabilní.

[20] Tento univerzálnější a ekonomičtější přístup ke konstrukci reléových obvodů budeme nadále užívat.

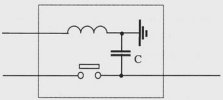


Obr. 6.12

ad 3. Při tomto způsobu se kombinační logický obvod doplní o obvody vnější synchronizace např. tak, jak je to uvedeno v následujícím schématu vnější synchronizace obvodu se třemi binárními vstupy a dvěma binárními výstupy (Obr. 6.13): V něm každá vnější binární proměnná daného logického obvodu je vedena na dolní vstup svého vzorkovacího hradla otevíraného synchronizačními pulsy přiváděnými z generátoru pulsů na horní vstup.[21] Výstupy vstupních hradel jsou vedeny na příslušné vstupy logického obvodu, na výstupech výstupních hradel získáváme jeho výsledné reakce. Pro šířku otevíracích pulsů Δt_1 platí $\Delta t_h < \Delta t_1 < \Delta t_{rmin}$, pro vzdálenost Δt_2 otevíracích pulsů platí $\Delta t_2 > \Delta t_{rmax}$, kde Δt_h je reakční doba hradla, Δt_{rmin} je nejkratší reakční doba logického obvodu, a Δt_{rmax} je nejdelší reakční doba téhož.



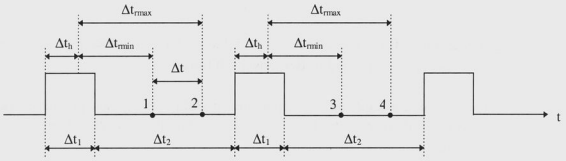
[21] Princip funkce vzorkovacího hradla si vysvětlíme na následujícím silně zjednodušeném schématu jeho možného zapojení (neberoucím v potaz některé energetické aspekty jeho funkce):



Přiváděné synchronizační pulsy prochází cívkou vzorkovacího relé (horní vstup), jehož spínacím kontaktem je hodnota vstupní proměnné, přiváděné na dolní vstup, vzorkována. Kondenzátor C je jednoduchým paměťovým prvkem: Přes sepnutý kontakt vzorkovacího relé se nabíjí (či vybíjí) na aktuální hodnotu vzorkované vstupní proměnné. Tuto hodnotu pak udržuje na výstupu hradla i po rozepnutí vzorkovacího kontaktu (tj. až do okamžiku vzorkování dalším synchronizačním pulsem). Reakční doba hradla Δt_h je součtem reakční doby vzorkovacího relé a doby potřebné k nabití (či vybití) kondenzátoru.

Obr. 6.13

K eliminaci hazardů zde dochází následovně (viz Obr. 6.14): Za dobu Δt_h po příchodu prvního zakresleného otevíracího pulsu se na jednotlivých vstupech logického obvodu objeví hodnoty aktuálního podnětu, který se po dobu $\Delta t_R = \Delta t_1 + \Delta t_2$ (tj. až do okamžiku zpožděného o dobu Δt_h za příchodem dalšího pulsu) nebude měnit. Přechodové reakce na tento podnět se na výstupech logického obvodu (nebo-li na vstupech výstupních hradel) objeví v době mezi okamžiky 1 a 2. V té době jsou hradla zavřená, takže na jejich výstupy neproniknou. Nejpozději v okamžiku 2 se ustálí správná výstupní reakce logického obvodu na aktuální podnět a bude k dispozici na dolních vstupech výstupních hradel nejméně do okamžiku 3. Proto ji druhý zakreslený otevírací puls propustí na výstupy výstupních hradel, kde se udrží také po dobu $\Delta t_R = \Delta t_1 + \Delta t_2$. Současně během druhého otevíracího pulsu vstupní hradla „přečtou" nový aktuální podnět, který vyvolá přechodové reakce mezi okamžiky 3 a 4, a správná reakce na něj se ustálí nejpozději v okamžiku 4. Vše se znovu opakuje.



Obr. 6.14

Synchronní a asynchronní obvody

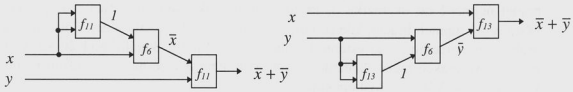
Z uvedeného je zřejmé, že popsaná vnější synchronizace vnucuje všem takto synchronizovaným logickým obvodům umělou a jednotnou reakční dobu $\Delta t_R = \Delta t_1 + \Delta t_2$. Nesprávné reakce vyvolané hazardy se zde nemohou projevit, protože se vyskytují pouze v době uzavření hradel a na jejich výstup neproniknou. Logickým obvodům s vnější synchronizací se říká **synchronní obvody**. Naproti tomu logickým obvodům, u nichž se hazardy eliminují jen symetrizací struktury či tlumícími členy, a jejichž reakční doby jsou tak dány jen přirozenými fyzikálními přechodovými ději jejich členů a nejsou ovlivňovány z vnějšku, se říká **asynchronní obvody**.

Pro správnou funkci synchronních (resp. asynchronních) obvodů je nutné, aby frekvence změn vstupních podnětů, udaná počtem změn za sekundu, nepřesáhla hodnotu $1/\Delta t_R$ (resp. $1/\Delta t_r$), kde reakční dobu udáváme v sekundách. Asynchronní obvody tak mohou být mnohem rychlejší než obvody synchronní. Naproti tomu synchronní obvody obecně fungují spolehlivěji než obvody asynchronní. S ohledem na tyto vlastnosti se asynchronní obvody uplatňují především tam, kde se vyžaduje vysoká rychlost zpracování požadavků (např. v řídících systémech digitálních spojovacích systémů); synchronní obvody se uplatňují zejména tam, kde se klade důraz na spolehlivost funkce (např. v číslicových počítačích).

Cvičení k lekci 6.

Příklad 6.1: Má se dokázat, že neekvivalence $f_6 = \overline{x} \cdot y + x \cdot \overline{y}$ spolu s implikací $f_{11} = \overline{x} \cdot \overline{y} + x \cdot \overline{y} + x \cdot y = x + \overline{y}$ nebo $f_{13} = \overline{x} \cdot \overline{y} + \overline{x} \cdot y + x \cdot y = \overline{x} + y$ tvoří úplný soubor Booleovských funkcí.

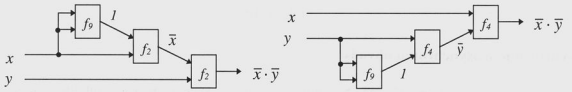
Lze to dokázat např. tak, že zapojením modulů realizujících příslušné operátory vytvoříme Shefferovu funkci $f_{14} = \overline{x} \cdot \overline{y} + \overline{x} \cdot y + x \cdot \overline{y} = \overline{x} + \overline{y}$ (viz následující schémata zapojení modulů):



Obr. 6.15

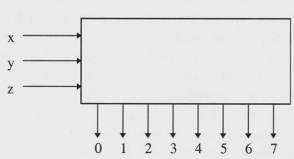
Příklad 6.2: Má se dokázat, že ekvivalence $f_9 = \overline{x} \cdot \overline{y} + x \cdot y$ spolu s inhibicí $f_2 = x \cdot \overline{y}$ nebo $f_4 = \overline{x} \cdot y$ tvoří úplný soubor Booleovských funkcí.

Lze to dokázat např. tak, že zapojením modulů realizujících příslušné operátory vytvoříme Peirceovu funkci $f_8 = \overline{x} \cdot \overline{y}$ (viz následující schémata zapojení modulů):



Obr. 6.16

Příklad 6.3: Má se navrhnout převodník pro vzájemně jednoznačnou transformaci všech 2^3 možných hodnot vektoru (x,y,z) vstupních binárních proměnných do hodnot signálů na 2^3 výstupních portech označených na Obr. 6.17 čísly 0 až 7. Přitom se požaduje, aby převodník na vstupní kombinaci binárně reprezentující číslo N na výstupu odpověděl přítomností signálu právě na portu číslo N.



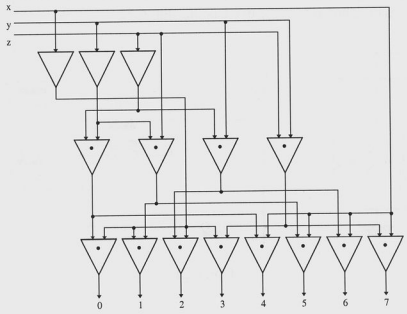
Obr. 6.17

Řešení:
Jedná se o převod vstupních čísel z *binární* soustavy do soustavy *unární*, v níž je číslo dáno počtem „mrtvých" portů nacházejících se nalevo od jediného „živého" výstupního portu. Každý výstupní port se může nacházet ve dvou stavech („živý", „mrtvý"), a lze jej pokládat za binární proměnnou. Naší unární reprezentaci čísla N lze tak ztotožnit s řetězcem, začínajícím inverzí binárního kódu čísla 2^N doplněnou zprava na potřebný rozměr nulami.[22] Docházíme tak k následující transformační tabulce převodníku, mající nenulové hodnoty pouze na hlavní diagonále:

x	y	z	0	1	2	3	4	5	6	7
0	0	0	1							
0	0	1		1						
0	1	0			1					
0	1	1				1				
1	0	0					1			
1	0	1						1		
1	1	0							1	
1	1	1								1

Obr. 6.18

Tabulka definuje Booleovské funkce realizované jednotlivými porty a z ní lze odvodit abstraktní model převodníku ve formě následující logické sítě:



Obr. 6.19

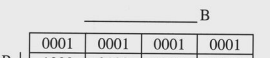
[22] Inverzí řetězce cifer se zde rozumí uvedení cifer v opačném pořadí. Např. pro N = 4 je 10000 binární kód čísla 2^N. Jeho inverzí je 00001 a po doplnění nulami zprava na osm cifer dostáváme 00001000 (unární reprezentace čísla N = 4).

Zásadní rozdíl mezi vstupními a výstupními porty sítě spočívá v tom, že vstupní porty zde představují vzájemně *nezávislé* binární proměnné, zatímco výstupní porty představují množinu čísel, z nichž se jedno vybírá (unární kód čísla). Alternativně lze na výstupní porty nahlížet jako na vzájemně *závislé* binární proměnné, jejichž vzájemný vztah je dán pravidly transformace vstupů na výstupy.

Úloha 6.1: Navrhněte jednoduchý převodník z uvedené osmiportové unární soustavy na tři řády binární soustavy (přechod z „osmidrátu na třídrát").

Úloha 6.2: Horní část logické sítě na Obr. 6.2 je jedním z možných grafických znázornění struktury Booleovské funkce f_1. Znázorněte strukturu téže funkce nějakou jinou logickou sítí.

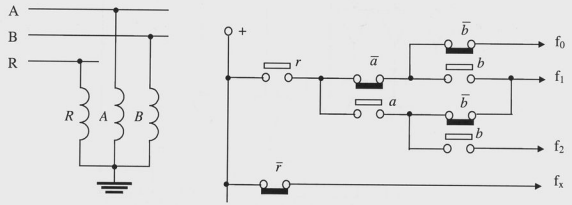
Příklad 6.4: Transformační funkci převodníku P zadanou tabulkou na Obr. 5.10 (Příklad 5.4) je možno vyjádřit Karnaughovou mapou na Obr. 6.20 a z ní pak odvodit příslušné Booleovské výrazy pro f_0, f_1, f_2 a f_x.



$$f_0 = R \cdot \overline{A} \cdot \overline{B}$$
$$f_1 = R \cdot (\overline{A} \cdot B + A \cdot \overline{B})$$
$$f_2 = R \cdot A \cdot B$$
$$f_x = \overline{R}$$

Obr. 6.20

Z nich vyplývá, že převodník je možno realizovat reléovým kontaktovým obvodem, jehož schéma je uvedeno na Obr. 6.21.



Obr. 6.21

Příklad 6.5:

Prvky kontaktových obvodů

Doposud jsme předpokládali, že do logických obvodů vstupují z okolí binární signály, které pak obvody dále zpracovávají. Konkrétně v kontaktových obvodech měly tyto signály formu elektrických potenciálů, přiváděných na cívky relé vstupních členů. Tak tomu obvykle

bývá zejména v případech, kdy jsou vstupy daného obvodu připojeny k výstupům jiných logických obvodů. V praxi bývají logické obvody ovládány i nejrůznějšími **ovládacími prvky**, mezi něž patří spínače, přepínače a tlačítka.

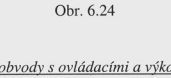
Tlačítka jsou klasickými monostabilními prvky, s jedním stabilním stavem (nestisknuto), ve druhém - nestabilním (stisknuto) je třeba držet je vnější silou. Na Obr. 6.22 je schématická značka tlačítka spínacího, rozpínacího a kombinovaného:

Obr. 6.22

Naproti tomu dvoupolohové *spínače* a *přepínače* jsou bistabilními prvky a mohou být otočné či překlápěcí. Schématické značky (postupně zleva doprava) překlápěcího spínače, dvoupolohového překlápěcího přepínače, překlápěcího křížového přepínače a vícepolohového otočného přepínače jsou na Obr. 6.23:

Obr. 6.23

Ovládací prvky přivádí napětí k cívkám **výkonných prvků** či zapojují do obvodu slaboproudé spotřebiče, jakými jsou např. žárovky. K výkonným prvkům patří nám už známá slaboproudá *relé* a navíc i silnoproudá relé, tzv. *stykače*, které svými silnoproudými kontakty připojují ke zdrojům elektrické energie silnoproudé spotřebiče, jako např. elektromotory, topná tělesa apod. Schéma stykače vidíme na Obr. 6.24. Základem stykače je silný elektromagnet, jehož cívku vidíme uprostřed. Kotva elektromagnetu spíná jak masivní silnoproudé kontakty (na obr. vlevo), tak slaboproudé kontakty (na obr. vpravo). Silnoproudý (výkonový) a slaboproudý (řídící) obvod stykače jsou od sebe odděleny.



Obr. 6.24

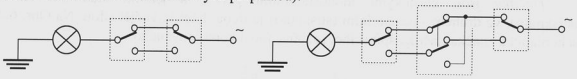
Příklad 6.6:

Jednoduché obvody s ovládacími a výkonnými prvky

Na Obr. 6.25 jsou uvedena základní schémata zapojení monostabilního (vlevo) a bistabilního (vpravo) klopného obvodu pro ovládání elektrického spotřebiče (žárovky).

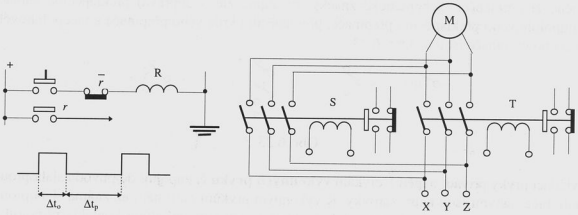
Obr. 6.25

Na Obr. 6.26 vidíme bistabilní klopné obvody ovládané z více míst (v praxi často užívaná zapojení schodišťových vypínačů, v nichž ke zvyšování počtu ovládacích míst dochází vkládáním dalších křížových přepínačů):



Obr. 6.26

Na Obr. 6.27 vlevo je schéma zapojení reléového astabilního *multivibrátoru*, neboli generátoru obdélníkových pulsů. V praxi je toto zapojení použito mimo jiné i u elektrického zvonku.



Obr. 6.27

Na Obr. 6.27 vpravo vidíme zapojení silových obvodů dvojice stykačů pro reverzaci chodu třífázového asynchronního indukčního elektromotoru. Zapojení řídících kontaktů stykačů nesmí připustit současné sepnutí obou stykačů (zkrat fází X a Y) - viz Příklad 7.3.

7. Implementace a realizace automatu

Víme, že pětice (S, I, O, δ, β), v níž S, I, O jsou neprázdné konečné množiny a funkce δ: S×I → S a β: S×I → O jsou spolu svázány některým z funkčních schémat uvedených v článku 3.3. (morfologická struktura v rovině teoretické), je algebraickým popisem konečného sekvenčního abstraktního stroje zvaného konečný automat. Jde o mechanismus, který (z čistě vnějšího pohledu) časovým posloupnostem hodnot z množiny I přiřazuje časové posloupnosti hodnot z množiny O. Realizovat tento automat znamená vytvořit stroj určitého typu, který by právě toto dělal. Takovým strojem může např. být:

a) program pro číslicový počítač či vhodně naprogramovaný mikroprocesor (zde mluvíme o simulaci automatu),

b) VLSI[1] obvod realizující mikročip na bázi kanonické sítě automatu,

c) synchronní či asynchronní logický obvod sestavený ze spínacích prvků dané fyzikální podstaty (pneumatických či hydraulických prvků, elektronických spínacích prvků, elektromagnetických relé a jejich kontaktů apod.).

Proces zrodu každého takového stroje má dvě základní fáze. V první fázi, nazvané **implementace automatu**, navrhneme model stroje. Druhou fází je **fyzická realizace** tohoto modelu:

ad a) Implementačním modelem softwarové či mikroprocesorové simulace automatu je vývojový diagram algoritmů přechodové a výstupní funkce, popsaných příslušnými tabulkami. Fyzickou realizací zde rozumíme přepis vývojového diagramu do formy akceptovatelné počítačem či mikroprocesorem, tj. do příslušného programovacího jazyka či do sekvence mikroinstrukcí mikroprocesoru.

ad b) V případě realizace automatu VLSI obvodem je implementačním modelem kanonická síť automatu, fyzickou realizací pak příslušný mikročip vytvořený VLSI obvodem.

ad c) V případě realizace logickým obvodem je výsledkem implementace automatu logická síť, popsaná systémem Booleovských funkcí nebo schématem zapojení logických modulů. Fyzickou realizací na bázi kontaktových obvodů je pak odpovídající vodivé propojení cívek relé, jejich kontaktů, a zajištění potřebného časování signálů.

V následujících dvou článcích této lekce pojednáme o implementaci i realizaci obecně a výklad doprovodíme příkladem implementace automatu logickou sítí a její realizace kontaktovým obvodem. Ve třetím článku představíme kanonickou síť automatu, k softwarové a mikroprocesorové simulaci se vrátíme až v článku čtvrtém. V posledním pátém článku budeme řešit úlohu opačnou: Jak z daného schématu zapojení logické sítě či kontaktového obvodu zjistit, co daný stroj „dělá", a jak výsledek prezentovat ve formě konečného automatu.

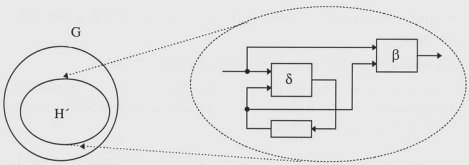
7.1. Implementace automatu

Filosofie implementace vychází z poznatku o jednoznačnosti analýzy, který říká, že systémy s podobnou dynamickou strukturou (tj. s podobným mechanismem změn) se musí i podobně chovat. Požadavek nejvyšší dosažitelné podobnosti chování stroje a jeho vzoru - implementovaného automatu - si tak vynucuje nejvyšší dosažitelnou podobnost dynamické struktury stroje s dynamickou strukturou vzoru. A té lze dosáhnout postupným věrným

[1] VLSI (Very Large Scale Integration) obvod je obvodem s velmi vysokou koncentrací polovodivých přechodů.

přenosem dynamické struktury (mechanismu) implementovaného automatu do mechanismu stroje. Parametry dynamické struktury (např. počet stavů) závisí i na volbě funkčního schématu (morfologická struktura v rovině teoretické). Proto věrný přenos mechanismu obvykle začíná přenosem funkčního schématu implementovaného automatu do morfologické struktury stroje.

Po provedení tohoto kroku bude částí neboli podstrukturou morfologické struktury G stroje morfologická struktura H´, izomorfní[2] s funkčním schématem implementovaného automatu (viz Obr. 7.1):



Obr. 7.1

H´ vymezuje v G implementací ovlivnitelnou oblast a současně zajišťuje i proveditelnost věrného přenosu tím, že pro něj vytváří vhodný rámec. Věrně přenést mechanismus automatu do mechanismu stroje pak v praxi znamená správně naplnit tento rámec. K tomu je třeba provést řadu spolu souvisejících kroků. Uvedeme si je v jejich přirozeném sledu na konkrétním příkladu implementace automatu ME logickou sítí.

Příklad implementace

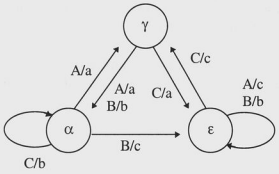
Zadání:

Logickou sítí se třemi binárními vstupy x_1, x_2, x_3 a třemi binárními výstupy y_1, y_2, y_3 s hodnotami signálů 0 a 1

Obr. 7.2

implementovat automat ME = (S, I, O, δ, β), S = {α, ε, γ}, I = {A, B, C}, O = {a, b, c}, v němž funkce δ a β jsou popsány následujícím grafem či tabulkou

[2] Izomorfní znamená identický až na přejmenování prvků. Izomorfizmus je vysoký stupeň strukturální podobnosti, následující hned za rovností neboli identitou (izo = stejný, morphé = tvar).



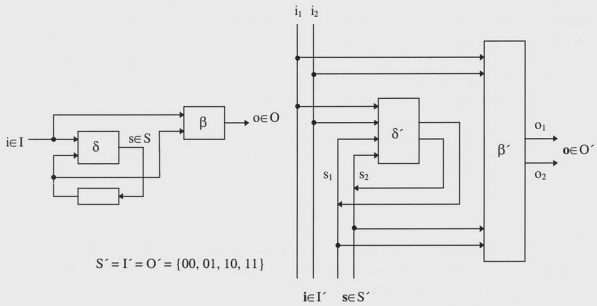
	A	B	C
α	(γ,a)	(ε,c)	(α,b)
ε	(ε,c)	(ε,b)	(γ,c)
γ	(α,a)	(α,b)	(ε,a)

Obr. 7.3

tak, aby A = (1,0,0) = a, B = (0,1,0) = b, a C = (0,0,1) = c.

Řešení:

V souladu s výše uvedeným bude prvním krokem věrného přenosu mechanismu automatu ME vytvoření rámcové morfologické struktury H´ logické sítě. Dostaneme ji prostým přenesením funkčního schématu automatu ME (na Obr. 7.4 vlevo) do prostředí logické sítě (na Obr. 7.4 vpravo):



Obr. 7.4

Vidíme, že struktura H´ je funkčním schématem jistého automatu ME´ = (S´, I´, O´, δ´, β´), a že toto funkční schéma je izomorfní s funkčním schématem implementovaného automatu ME. Počet paralelních cest každé sběrnice přenášející příslušná data automatu ME´ je dán potřebou prosté zobrazitelnosti množin S, I, O do množin S´, I´, O´ vektorů hodnot binárních signálů přenášených paralelními cestami příslušné sběrnice. Je totiž třeba po nich přenášet stejné množství dat jako po cestách funkčního schématu automatu ME. V případě tříprvkových množin stačí dvourozměrné binární vektory. Časová zpoždění ve zpětných vazbách zohledníme až ve fázi fyzické realizace. Při požadavku maximální operační rychlosti

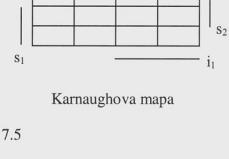
asynchronního obvodu (jímž budeme v našem případě síť realizovat) využijeme jen přirozených reakčních dob a žádná dodatečná zpoždění nezavedeme.

Výše uvedené schéma logické sítě definuje vstupy vektorové funkce [δ´,β´], tj. vstupy do její tabulky. Protože se jedná o tabulku logické funkce, je výhodné vyjádřit ji ve formě Karnaughovy mapy.

[δ´,β´]

S´\I´	00	01	10	11
00				
01				
10				
11				

≈



Tabulka přechodové a výstupní funkce

Karnaughova mapa

Obr. 7.5

Věrný přenos mechanismu automatu ME do logické sítě pak zajistíme správným vyplněním polí této Karnaughovy mapy, čímž vytvoříme adekvátní přechodovou funkci δ´ a výstupní funkci β´ automatu ME´. K tomu je třeba provést následující tři kroky:

1. Zvolit trojici injektivních neboli prostých zobrazení φ: S → S´, ξ: I → I´, ω: O → O´, (ve tvaru φ(s) = s_1s_2 = **s**, ξ(i) = i_1i_2 = **i**, ω(o) = o_1o_2 = **o**) zvaných kódovací funkce a jimi přiřadit prvkům množin automatu ME (vzorům) vektory hodnot binárních proměnných sítě (obrazy). V našem případě tato zobrazení volíme takto:

s	α	ε	γ
φ(s)	00	01	10

i	A	B	C
ξ(i)	01	10	11

o	a	b	c
ω(o)	00	01	11

Obr. 7.6

2. Uvést v rovině obrazů (tj. v Karnaughově mapě vektorové funkce [δ´,β´]) vstupní a výstupní hodnoty do vztahů fundamentálně analogických ke vztahům, definovaným v rovině vzorů přechodovou funkcí δ a výstupní funkcí β (viz Obr. 7.3). Výsledek tohoto kroku vidíme na Obr. 7.7.[3]

---

[3] Tabulka přechodové a výstupní funkce automatu ME přiřazuje každé kombinaci současného stavu $s \in S$ a vstupního podnětu $i \in I$ (vstupy do tabulky) bezprostředně následující stav $\delta(s,i) \in S$ a reakci $\beta(s,i) \in O$ (výstup z tabulky), čímž k danému počátečnímu stavu $s_0 \in S$ a posloupnosti $i_0,i_1,i_2,...$ vstupních podnětů jednoznačně definuje posloupnost $o_0,o_1,o_2,...$ výstupních reakcí. Hodnoty $(\delta(s,i),\beta(s,i))$ jsou v tabulce zapsány v poli o souřadnicích (s,i). Tento vztah mezi podněty na straně jedné a reakcemi na straně druhé je (vzhledem k danému funkčnímu schématu automatu ME) rámcově popsán levým či pravým algebraickým diagramem

$$S \times I \xrightarrow{\delta,\beta} S \times O \qquad O \xleftarrow{\beta} S \times I \xrightarrow{\delta} S$$

Fundamentálně analogický vztah

$$S' \times I' \xrightarrow{\delta',\beta'} S' \times O' \qquad O' \xleftarrow{\beta'} S' \times I' \xrightarrow{\delta'} S'$$

3. Pokud po provedení kroku 2 zůstala v Karnaughově mapě volná pole (tak jak je tomu v i v našem případě), vyplníme je tak, abychom z mapy získali co nejjednodušší Booleovské výrazy. Detailní postup je uveden na Obr. 7.9 a Obr. 7.10: Parciální Karnaughova mapa z Obr. 7.7 je na Obr. 7.9 rozložena do dvou map (vlevo je mapa pro δ': $S' \times I' \rightarrow S'$, vpravo je mapa pro β': $S' \times I' \rightarrow O'$). Každá z nich je na Obr. 7.10 dále rozložena do složek $s_1(t+1)$ a $s_2(t+1)$ vektoru **s** (vlevo), respektive do složek o_1 a o_2 vektoru **o** (vpravo). Parciální Karnaughovy mapy složek jsou na úplné Karnaughovy mapy doplněny tak, aby v nich vznikly velké a souvislé bloky jedniček (v souladu se zvolenou technikou zjednodušování Booleovských výrazů). Takto vzniklé Booleovské výrazy složek funkcí δ' a β' jsou v Obr. 7.10 připsány pod příslušné Karnaughovy mapy.

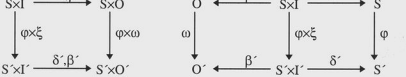
---

v automatu ME′ získáme tak, že:

- Souřadnicím (s,i) pole (v němž je vepsána hodnota $(\delta(s,i),\beta(s,i))$ - hodnota vzoru) tabulky přechodové a výstupní funkce $[\delta,\beta]$ automatu ME přiřadíme paralelním zobrazením $\varphi\times\xi$: $S\times I \rightarrow S'\times I'$ souřadnice $(\varphi(s),\xi(i))$ souvztažného pole v Karnaughově mapě vektorové funkce $[\delta',\beta']$.
- Do tohoto pole Karnaughovy mapy vepíšeme hodnotu $(\varphi(\delta(s,i)),\omega(\beta(s,i)))$, která je výsledkem paralelního zobrazení $\varphi\times\omega$: $S\times O \rightarrow S'\times O'$ hodnoty souvztažného pole (hodnoty vzoru) v tabulce $[\delta,\beta]$.

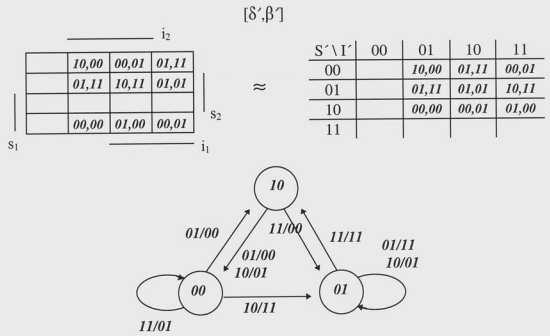
Souvztažnými poli jsou např. pole o souřadnicích (γ,B) v tabulce $[\delta,\beta]$ na Obr. 7.3 a pole o souřadnicích (10,10) v Karnaughově mapě $[\delta',\beta']$ na Obr. 7.7, neboť $\varphi(\gamma) = \mathbf{s} = s_1s_2 = 10 = i_1i_2 = \mathbf{i} = \xi(B)$. Proto jak v Karnaughově mapě, tak i z ní odvozené tabulce přechodové a výstupní funkce automatu ME′ nacházíme v poli o souřadnicích (10,10) hodnotu (00,01), neboť $00 = s_1s_2 = \mathbf{s} = \varphi(\alpha)$, $01 = o_1o_2 = \mathbf{o} = \omega(b)$, kde (α,b) je hodnota souvztažného pole (hodnota vzoru) v tabulce $[\delta,\beta]$.

Popsaný věrný přenos mechanismu lze graficky znázornit těmito algebraickými diagramy relevantních zobrazení:

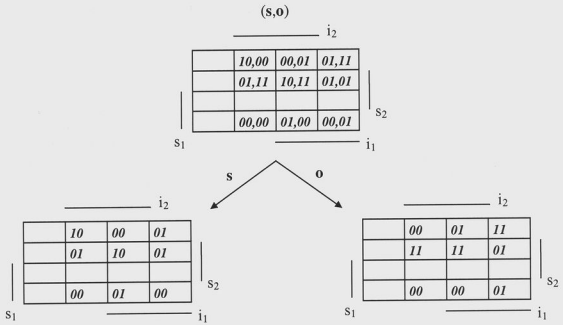


Obr.7.8

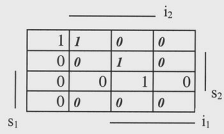
V levém diagramu zobrazení $\varphi\times\xi$ váže souvztažná pole tabulek přechodových a výstupních funkcí a zobrazení $\varphi\times\omega$ transformuje hodnotu vzoru na hodnotu obrazu. Pravý diagram je rozkladem levého diagramu podle složek vektorových funkcí, vytvářejících dvě dílčí „okna". Z popsaného postupu věrného přenosu mechanismu plyne, že od souřadnice $(s,i) \in S\times I$ lze k hodnotě obrazu $(\varphi(\delta(s,i)),\omega(\beta(s,i))) \in S'\times O'$ dojít dvěma různými cestami (viz levý diagram): dolní cestou ve směru kompozice zobrazení $[\delta',\beta']\circ(\varphi\times\xi)$, a horní cestou ve směru kompozice zobrazení $(\varphi\times\omega)\circ[\delta,\beta]$, proto platí rovnost $[\delta',\beta']\circ(\varphi\times\xi) = (\varphi\times\omega)\circ[\delta,\beta]$, matematicky deklarující nezávislost výsledku na cestě, kterou byl dosažen. V terminologii abstraktní algebry se jedná o tzv. **podmínku komutace algebraického diagramu**, která se rozepsáním do složek vektorových funkcí rozpadne na dvojici podmínek $\delta'\circ(\varphi\times\xi) = \varphi\circ\delta$ a $\beta'\circ(\varphi\times\xi) = \omega\circ\beta$. První je podmínkou komutace pravého okna, druhá je podmínkou komutace levého okna pravého algebraického diagramu.


[δ´,β´]
i2
i1
s1
s2
10,00 00,01 01,11
01,11 10,11 01,01
00,00 01,00 00,01
≈
S´\I´ 00 01 10 11
00 10,00 01,11 00,01
01 01,11 01,01 10,11
10 00,00 00,01 01,00
11
10
00
01
01/00
11/00
11/11
01/11
10/01
01/00
10/01
10/11
11/01

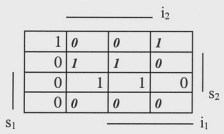
Obr. 7.7


(s,o)
i2
i1
s1
s2
10,00 00,01 01,11
01,11 10,11 01,01
00,00 01,00 00,01
s
o
10 00 01
01 10 01
00 01 00
00 01 11
11 11 01
00 00 01

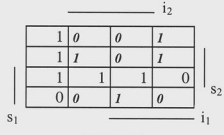
Obr. 7.9



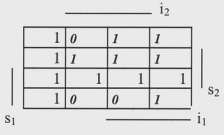
$s_1(t+1) = \overline{i_1}\ \overline{s_1}\ \overline{s_2} + i_1\, i_2\, s_2$



$o_1 = \overline{i_2}\ \overline{s_1}\ \overline{s_2} + i_2\, s_2$



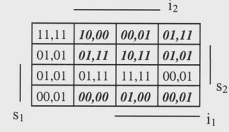
$s_2(t+1) = \overline{i_2}\ \overline{s_1} + \overline{i_1}\ s_2 + i_1\, i_2\, s_1$



$o_2 = \overline{i_2} + s_2 + i_1\ \overline{s_1}$

Obr. 7.10

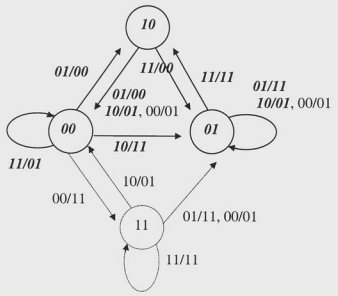
Sloučením doplněných dílčích map podle Obr. 7.10 vznikne Karnaughova mapa úplné vektorové funkce $[\delta', \beta']$ a jí odpovídající automat ME´ (viz Obr. 7.11).



$[\delta', \beta']$

$\approx$

S´\I´	00	01	10	11
00	11,11	***10,00***	***01,11***	***00,01***
01	01,01	***01,11***	***01,01***	***10,11***
10	00,01	***00,00***	***00,01***	***01,00***
11	01,01	01,11	00,01	11,11



Obr. 7.11

Vlevo je Karnaughova mapa, vpravo jí odpovídající tabulka přechodové a výstupní funkce, a pod tím dynamická struktura automatu ME´ = (S´, I´, O´, δ´, β´) vytvořeného logickou sítí. K věrnému přenosu mechanismu implementovaného automatu ME došlo v kroku 2; výsledky tohoto kroku jsou v mapě, tabulce i grafu vyznačeny silně a proloženě. V grafu vytvořeného ME´ je patrný podgraf izomorfní s grafem implementovaného ME, svědčící o shodě v mechanismech obou automatů.

Vytvořením automatu ME´ (obrazu) zohledňujícího mechanismus automatu ME (vzoru) fáze implementace ještě nekončí. Výsledkem implementace má být model stroje, který „dělá" přesně to, co automat ME. A to, jak vidno, automat ME´ nedělá. Automat ME funguje v prostředí s obecnými vnějšími proměnnými I a O, automat ME´ v prostředí s binárními vektorovými vnějšími proměnnými I´ a O´. V terminologii matematické lingvistiky má mechanismus automatu ME´ jak složitější abecedu (více vstupních podnětů a výstupních reakcí), tak i složitější gramatiku (více stavů a přechodů, tj. více pravidel). Aby automat ME´ „dělal" to, co automat ME, je třeba jej přimět reagovat na podněty z množiny I, udržet jej v silně vyznačené podmnožině stavů jeho přechodového grafu, a převést jeho reakce z množiny O´ na adekvátní reakce z množiny O.

K tomu je nutné a stačí připojit vstup automatu ME´ na výstup vstupního převodníku s kódovací funkcí ξ: I → I´, kterou jsme při věrném přenosu struktury již použili, a na výstup automatu ME´ připojit vstup výstupního převodníku s transformační funkcí ω^{-1}: O´ → O, přičemž souvislost transformační funkce ω^{-1} s již použitou kódovací funkcí ω je zřejmá z porovnání jejich tabulek:

ω: O → O´

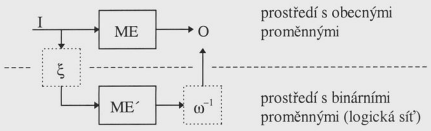
o	a	b	c
ω(o)	00	01	11

ω^{-1}: O´ → O

o	00	01	10	11
ω^{-1}(**o**)	a	b	-	c

Obr. 7.12

Transformační funkce ω^{-1} vznikne libovolným dodefinováním parciální funkce inverzní ke kódovací funkci ω na funkci úplnou. Kompozice funkcí $\omega^{-1} \circ \omega$ je identitou na množině O. Situace pak vypadá takto:



Obr. 7.13

a model stroje dostává následující obecnou podobu:

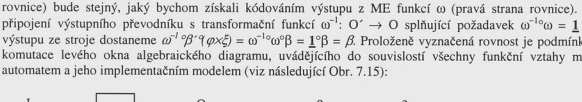
Obr. 7.14

K tomu, aby se uvedený model stroje choval vskutku tak, jako automat ME ve stavu $s \in S$, stačí nastavit automat ME´ do stavu $\varphi(s) \in S'$.[4] Je tomu tak proto, že náš model stroje je automatem schopným simulovat (viz článek 5.3.) chování implementovaného automatu ME a stavy s a $\varphi(s)$ jsou ekvivalentními stavy.[5]

---

[4] Připojení vstupního převodníku s kódovací funkcí ξ spolu s nastavením automatu ME´ do počátečního stavu $\varphi(s)$ zajistí stálou existenci prvního úseku spodních cest (tj. zobrazení $\varphi \times \xi$) v pravém algebraickém diagramu relevantních zobrazení (viz předchozí poznámka pod čarou). Složku ξ průběžně vytváří převodník, složku φ pak průběžně generuje společná část mechanismu obou automatů. Plyne to z podmínky komutace pravého okna ve tvaru $\delta'^{\circ}(\varphi \times \xi) = \varphi^{\circ}\delta$, která říká, že vazba okamžitých stavů zobrazením φ se přenáší na následníky (přesněji: každé dva stavy, jeden v ME, druhý v ME´, pokud budou vázány zobrazením φ: S → S´, budou mít na každý vstup i ∈ I své následníky také vázány zobrazením φ: S → S´).

Podmínka komutace levého okna ve tvaru $\beta'^{\circ}(\varphi \times \xi) = \omega^{0}\beta$ k tomu dodává, že přitom výstup z ME´ (levá strana rovnice) bude stejný, jaký bychom získali kódováním výstupu z ME funkcí ω (pravá strana rovnice). Po připojení výstupního převodníku s transformační funkcí ω^{-1}: O´ → O splňující požadavek $\omega^{-1} \circ \omega = \underline{1}$ na výstupu ze stroje dostaneme $\omega^{-1} \circ \beta' \circ (\varphi \times \xi) = \omega^{-1} \circ \omega^{0} \circ \beta = \underline{1} \circ \beta = \beta$. Proloženě vyznačená rovnost je podmínkou komutace levého okna algebraického diagramu, uváděj ícího do souvislostí všechny funkční vztahy mezi automatem a jeho implementačním modelem (viz následující Obr. 7.15):



Obr. 7.15

[5] Na vstupní řetězec $\boldsymbol{x} \in I^*$ zareaguje automat ME ve stavu $s \in S$ výstupním řetězcem $\beta^*(s, \boldsymbol{x})$, a náš implementační model ve stavu $\varphi(s) \in S'$ zareaguje na stejný vstupní řetězec výstupním řetězcem $\omega^{-1*}(\beta'^*(\varphi(s), \xi^*(\boldsymbol{x})))$. Dokážeme-li platnost rovnosti $\omega^{-1*}(\beta'^*(\varphi(s), \xi^*(\boldsymbol{x}))) = \beta^*(s, \boldsymbol{x})$ pro $\forall \boldsymbol{x} \in I^*$, dokážeme, že stavy s a φ(s) jsou ekvivalentní stavy. Důkaz provedeme indukcí na délku řetězce a vyjdeme při něm z následujících předpokladů:

1. $f: A \to B$, $f^*: A^* \to B^*$, $(\varepsilon \in A^*, |\varepsilon| = 0,) \Rightarrow (f^*(\varepsilon) = \varepsilon)$, $(a_1 a_2 a_3 \ldots a_n = \boldsymbol{x} \in A^*) \Rightarrow (f^*(\boldsymbol{x}) = f(a_1)f(a_2)f(a_3)\ldots f(a_n))$, $\forall (a \in A, \boldsymbol{x} \in A^*)$: $f^*(a\boldsymbol{x}) = f(a)f^*(\boldsymbol{x})$ - *definice* f^* *jako rozšíření* f *na řetězce prvků definičního oboru* A;
2. $\delta^*(s, \varepsilon) = s$, $\delta^*(s, i\boldsymbol{x}) = \delta^*(\delta(s, i), \boldsymbol{x})$; $\beta^*(s, \varepsilon) = \varepsilon$, $\beta^*(s, i\boldsymbol{x}) = \beta(s, i)\beta^*(\delta(s, i), \boldsymbol{x})$, kde $i \in I$, $s \in S$, $\boldsymbol{x} \in I^*$ - *rozšíření přechodové a výstupní funkce na řetězce*;
3. $\delta'(\varphi(s), \xi(i)) = \varphi(\delta(s, i))$, $\omega^{-1}(\beta'(\varphi(s), \xi(i)) = \beta(s, i)$ - *podmínky komutace algebraického diagramu.*

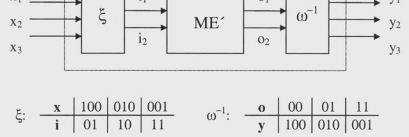
Důkaz (horní index u rovnítka odkazuje na použitý předpoklad pro substituci):

- pro $|\boldsymbol{x}| = 0$, tedy pro $\boldsymbol{x} = \varepsilon$, $\underline{\omega^{-1*}(\beta'^*(\varphi(s), \xi^*(\varepsilon)))} =^1 \omega^{-1*}(\beta'^*(\varphi(s), \varepsilon)) =^2 \omega^{-1*}(\varepsilon) =^1 \varepsilon =^2 \underline{\beta^*(s, \varepsilon)}$;
- pro $|\boldsymbol{x}| = 1$, tedy pro $\boldsymbol{x} = i \in I$, $\underline{\omega^{-1*}(\beta'^*(\varphi(s), \xi^*(i)))} =^1 \omega^{-1*}(\beta'^*(\varphi(s), \xi(i))\varepsilon)) =^2$ $\omega^{-1*}(\beta'(\varphi(s), \xi(i))\beta'^*(\delta(\varphi(s), \xi(i)), \varepsilon)) =^2 \omega^{-1*}(\beta'(\varphi(s), \xi(i))) =^1 \omega^{-1}(\beta'(\varphi(s), \xi(i))) =^3 \beta(s, i) =^2 \underline{\beta^*(s, i)}$.
- Induktivní krok: z předpokladu platnosti

$$\omega^{-1*}(\beta'^*(\varphi(s), \xi^*(\boldsymbol{x}))) = \beta^*(s, \boldsymbol{x}) \text{ pro } |\boldsymbol{x}| = n \qquad (*)$$

dokážeme platnost pro $|i\boldsymbol{x}| = n+1$.

Posledními kroky implementace je konstrukce převodníku z I do I´ (kodéru) a převodníku z O´ do O (dekodéru) v prostředí logické sítě. Z pohledu logické sítě se současná situace v námi zvažovaném případě jeví tak, jak ji popisuje Obr. 7.16:



ξ:

x	100	010	001
i	01	10	11

ω^{-1}:

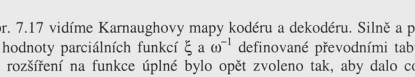
o	00	01	11
y	100	010	001

Obr. 7.16

Na něm je již částečně rozkryta struktura bloku označeného v Obr. 7.2 otazníkem. V převodních tabulkách $\mathbf{x} = (x_1,x_2,x_3)$, $\mathbf{i} = (i_1,i_2)$, $\mathbf{o} = (o_1,o_2)$, $\mathbf{y} = (y_1,y_2,y_3)$, a příslušné transformační vztahy jsou důsledkem požadavku (viz zadání příkladu) A = (1,0,0) = a, B = (0,1,0) = b, a C = (0,0,1) = c a dalších dílčích transformačních vztahů z Obr. 7.6 a Obr. 7.12. Podrobná struktura ME´ je uvedena na Obr. 7.4, její funkce je z pohledu teorie automatů zřejmá z Obr. 7.11 a v prostředí sekvenčních logických sítí je popsána soustavou Booleovských výrazů (viz Obr. 7.10)

$$s_1(t+1) = \overline{i_1}\;\overline{s_1}\;\overline{s_2} + i_1\,i_2\,s_2\;,\quad s_2(t+1) = \overline{i_2}\;\overline{s_1} + \overline{i_1}\;s_2 + i_1\,i_2\,s_1\,,$$
$$o_1 = \overline{i_2}\;\overline{s_1}\;\overline{s_2} + i_2\,s_2\;,\quad o_2 = \overline{i_2} + s_2 + i_1\,\overline{s_1}\,.$$

Na Obr. 7.17 vidíme Karnaughovy mapy kodéru a dekodéru. Silně a proloženě jsou v nich zapsány hodnoty parciálních funkcí ξ a ω^{-1} definované převodními tabulkami na Obr. 7.16, a jejich rozšíření na funkce úplné bylo opět zvoleno tak, aby dalo co nejjednodušší Booleovské výrazy.



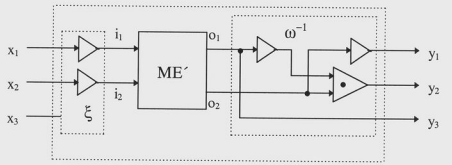
$i_1 = \overline{x_1}\,,\ i_2 = \overline{x_2}$ $\qquad$ $y_1 = \overline{o_2}\,,\ y_2 = \overline{o_1}\;o_2\,,\ y_3 = o_1$

Obr. 7.17

Je zřejmé, že při takto zvoleném rozšíření funkcí ξ a ω^{-1} kodér pouze neguje své vstupy x_1 a x_2, zatímco vstup x_3 zcela ignoruje; dekodér na výstupu y_3 vydává svůj vstup o_1, na y_1 vydává negaci svého vstupu o_2 a na y_2 vydává konjunkci negací ostatních výstupů.

$\underline{\omega^{-1*}(\beta^{*}(\varphi(s),\ \xi^{*}(\mathbf{ix})))} =^{1} \omega^{-1*}(\beta^{*}(\varphi(s),\ \xi(i)\xi^{*}(\boldsymbol{x}))) =^{2} \omega^{-1*}(\beta^{*}(\varphi(s),\ \xi(i))\beta^{*}(\delta^{*}(\varphi(s),\ \xi(i)),\ \xi^{*}(\boldsymbol{x}))) =^{1}$
$\omega^{-1}(\beta^{*}(\varphi(s),\ \xi(i)))\ \omega^{-1*}(\beta^{*}(\delta^{*}(\varphi(s),\ \xi(i)),\ \xi^{*}(\boldsymbol{x}))) =^{3} \beta(s,\ i)\omega^{-1*}(\beta^{*}(\delta^{*}(\varphi(s),\ \xi(i)),\ \xi^{*}(\boldsymbol{x}))) =^{5}$
$\beta(s,\ i)\omega^{-1*}(\beta^{*}(\varphi(\delta(s,\ i)),\ \xi^{*}(\boldsymbol{x}))) =^{(*)} \beta(s,\ i)\beta^{*}(\delta(s,\ i),\ \boldsymbol{x}) =^{2} \underline{\beta^{*}(s,\ \mathbf{ix})}$. Tím je důkaz dokončen.

Po zohlednění těchto faktů bude konečným výsledkem implementace našeho automatu ME logická síť znázorněná na následujícím obrázku:



Obr. 7.18

7.2. Realizace logické sítě automatu kontaktovým obvodem

V souladu s tím, co bylo řečeno v úvodu této lekce, je cílem následující fáze konstrukce stroje „fungujícího stejně" jako automat ME z Obr. 7.3, návrh schématu logického obvodu, realizujícího logickou síť z Obr. 7.18. Tato síť byla v předchozím článku popsána tímto systémem devíti Booleovských výrazů:

1. $s_1(t+1) = \overline{i_1}\,\overline{s_1}\,\overline{s_2} + i_1\, i_2\, s_2$,
2. $s_2(t+1) = \overline{i_2}\,\overline{s_1} + \overline{i_1}\, s_2 + i_1\, i_2\, s_1$,
3. $o_1 = \overline{i_2}\,\overline{s_1}\,\overline{s_2} + i_2\, s_2$,
4. $o_2 = \overline{i_2} + s_2 + i_1\,\overline{s_1}$,
5. $i_1 = \overline{x_1}$,
6. $i_2 = \overline{x_2}$,
7. $y_1 = \overline{o_2}$,
8. $y_2 = \overline{o_1}\, o_2$,
9. $y_3 = o_1$.

Užijeme-li k negaci příslušných výrazů De Morganových zákonů Booleovy algebry (viz článek 6.2), pak substitucí výrazů 5. a 6. do výrazů 1. až 4. a výrazů 3. a 4. do výrazů 7. až 9. vyřadíme z popisu sítě vnitřní proměnné i_1, i_2, o_1, o_2 a popis struktury logické sítě se tím transformuje do následující soustavy pěti Booleovských výrazů:

$s_1(t+1) = x_1\,\overline{s_1}\,\overline{s_2} + \overline{x_1}\,\overline{x_2}\, s_2$,

$s_2(t+1) = x_2\,\overline{s_1} + x_1\, s_2 + \overline{x_1}\,\overline{x_2}\, s_1$,

$y_1 = \overline{x_2}\,(x_1 + s_1)\,\overline{s_2}$,

$y_2 = (x_2 + s_2)(\overline{x_2} + s_1 + s_2)(x_2 + \overline{x_1}\,\overline{s_1} + s_2)$,

$y_3 = \overline{x_2}\, s_2 + x_2\,\overline{s_1}\,\overline{s_2}$.

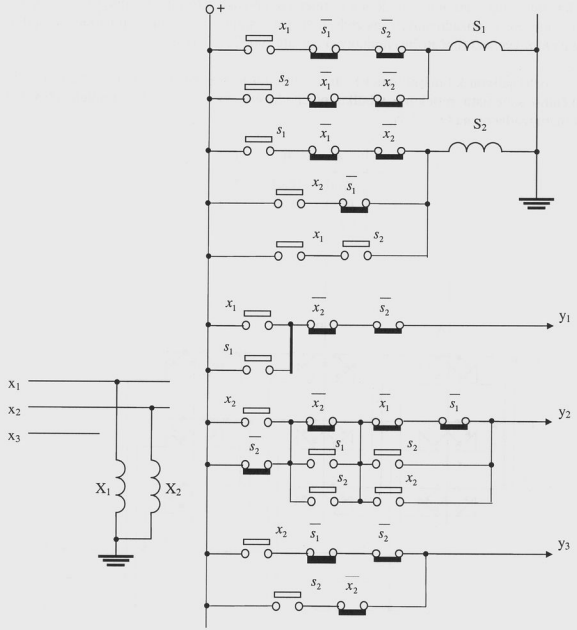
Fyzická realizace stroje bude záviset na tom, z jakých logických členů chceme obvod sestavit. Pokud bychom jej sestavovali z tří typů logických členů provádějících unární operaci negace, binární operace konjunkce a disjunkce (viz druhý až čtvrtý modul shora na Obr. 6.9),

sestavili bychom (podobně jako na Obr. 6.2) k dané soustavě Booleovských výrazů jí odpovídající logickou síť, a stejným způsobem, jakým by v ní byly propojeny vstupy a výstupy jednotlivých modulů, bychom v logickém obvodu propojili vstupy a výstupy jim odpovídajících logických členů.

Pokud bychom požadovali homogenní logický obvod (na bázi členů realizujících buď Peirceovu nebo Shefferovu funkci), nahradili bychom nejprve ve výše zmíněné logické síti buď moduly konjunkce nebo moduly disjunkce příslušným schématem podle Obr. 6.3. Tím bychom dostali síť jen se dvěma typy modulů a jejich substitucí schémata podle Obr. 6.5 bychom získali výslednou homogenní logickou síť. Podle schématu zapojení modulů v této síti bychom pak propojili vstupy a výstupy jednotlivých stejnorodých členů obvodu.

Ve shodě s poznámkou pod čarou 19 článku 6.3 sestavíme náš obvod ze samostatných relé s větším počtem spínacích i rozpínacích kontaktů. Konkrétně použijeme čtyři relé X_1, X_2, S_1 a S_2 s následujícími počty kontaktů:
X_1 - 3 spínací a 3 rozpínací,
X_2 - 5 spínacích a 4 rozpínací,
S_1 - 3 spínací a 4 rozpínací,
S_2 - 5 spínacích a 4 rozpínací.

Čtyři cívky těchto relé a všech 31 jejich kontaktů pak propojíme (tak jak přikazuje výše uvedená pětice Booleovských výrazů) do obvodu dle schématu na Obr. 7.19. Kontaktový obvod bude pracovat v asynchronním režimu, doby přítahu i odpadu všech čtyř relé budou nastaveny na stejnou hodnotu Δt a vnější podněty budou na vstup přicházet s frekvencí $1/\Delta t$ (čímž pro okamžiky diskrétního času dostáváme $t_{n+1} = t_n + \Delta t$). Že se jedná o sekvenční obvod (obvod s pamětí) je zřejmé z toho, že cívky relé S_1 a S_2 jsou spínány i svými kontakty (realizace zpětných vazeb v šíření signálu).



Obr. 7.19

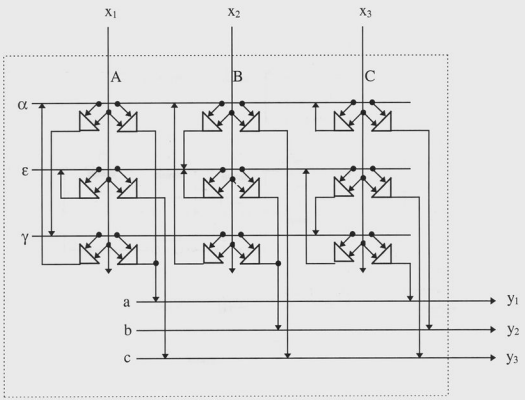
7.3. Kanonická síť automatu

Logická síť na Obr. 7.4, reflektující funkční schéma implementovaného automatu, je sítí *binární*. Logické moduly δ' a β' v ní zapojené provádí nad příslušnými binárními proměnnými Booleovské operace (definované algebraickými výrazy pod Karnaughovými mapami na Obr. 7.10), jejichž výsledky jsou (v důsledku věrného přenosu struktury automatu popsaného v článku 7.1.) ve shodě s výsledky obecnějších operací, definovaných v implementovaném automatu funkcemi δ a β. Každý spoj v této binární logické síti představuje jednu binární proměnnou, nabývající hodnot z množiny {0,1}, a proto lze tuto síť „ušít na míru" jen těm automatům, v nichž počty stavů, vstupních podnětů i výstupních podnětů jsou

mocninami čísla 2. V ostatních případech musíme počítat s tím, že chování implementovaného automatu bude jen segmentem chování této binární logické sítě (viz Obr. 7.11), kterou v požadovaných mezích drží jen vstupní a výstupní převodníky spolu s nastavením výchozí (počáteční) hodnoty stavu sítě (viz Obr. 7.18).

Alternativou k binární logické síti na Obr. 7.4 je logická síť na Obr. 7.20, která místo funkčního schématu reflektuje tabulku přechodové a výstupní funkce implementovaného automatu (zadanou na Obr. 7.3):

	A	B	C
α	(γ,a)	(ε,c)	(α,b)
ε	(ε,c)	(ε,b)	(γ,c)
γ	(α,a)	(α,b)	(ε,a)



Obr. 7.20

Každý řádek tabulky automatu přísluší jedné hodnotě (α, ε či γ) stavové proměnné S. V logické síti mu odpovídá vnitřní stejně označený vodorovný spoj. Každý sloupec tabulky přísluší jedné hodnotě (A, B či C) vstupní proměnné I. V logické síti mu odpovídá svislý spoj, vycházející ze stejně označeného vstupního portu. Jednotlivým hodnotám (a, b či c) výstupní proměnné O jsou v logické síti přiřazeny shodně pojmenované výstupní porty. Každé vnitřní pole tabulky je průsečíkem jednoho řádku a jednoho sloupce. V síti tomuto poli odpovídá příslušné překřížení vodorovného a svislého spoje. Situaci implementovaného automatu, popsanou dvojicí (aktuální stav, aktuální podnět) zohledňujeme v tabulce výběrem příslušného řádku a sloupce, v síti pak výběrem příslušné dvojice na sebe kolmých spojů.

Vybraným spojům přiřadíme hodnotu výběru 1 a nevybraným spojům přiřadíme hodnotu výběru 0.

Aktuální výstup a následující stav (následníka) pro zvažovanou situaci nalezneme v poli tabulky ležícím v průsečíku vybraného řádku a sloupce. Tento fakt je v síti zohledněn dvojicemi logických násobiček, operujících nad hodnotami výběru křížících se spojů. Výstup z levé násobičky je vždy veden k vodorovnému spoji, korespondujícímu s následníkem. Výstup z pravé násobičky je veden k výstupnímu portu, označenému aktuálním výstupem. Uvedeným zapojením všech výstupů logických násobiček došlo k věrnému přenosu struktury implementovaného automatu do struktury sítě.

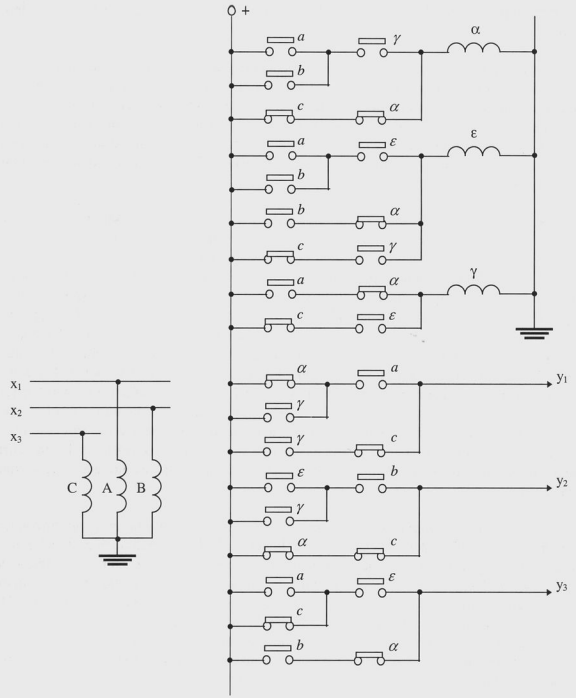
Na rozdíl od sítě z Obr. 7.4 se v případě sítě z Obr. 7.20 nejedná o logickou síť binární, ale o logickou síť *unární*,[1] pro níž se vžil název **kanonická síť automatu**. Kanonická síť funguje následovně: Jednotlivé vstupní porty (svislé spoje) označené A, B a C, z nichž v každém okamžiku právě jeden „pálí" (má hodnotu výběru 1), jsou hodnotami unární vstupní proměnné I = {A, B, C}.[2] Stejně tak vnitřní vodorovné spoje označené α, ε a γ (resp. výstupní porty označené písmeny a, b, c) jsou hodnotami unární stavové proměnné S = {α, ε, γ} (resp. výstupní proměnné O = {a, b, c}), a proto v každém okamžiku také „pálí" právě jeden z nich.[3] Na danou situaci zareaguje výstupem 1 ta dvojice logických násobiček, která má oba vstupy připojeny k „horkým" (tj. vybraným) spojům. Pravá z nich vybere výstupní reakci a levá následující stav. Je-li např. síť v aktuálním stavu α („pálí" horní vodorovný spoj) a je-li aktuálním vstupem podnět A („pálí" levý svislý spoj), zareaguje na tuto situaci dvojice násobiček umístěná v levém horním rohu sítě: pravá z nich vybere reakci „a" (přenese režim „pálí" na horní výstupní port), levá vybere následující stav „γ" (přenese režim „pálí" na vnitřní spodní vodorovný spoj). Je zřejmé, že kanonickou síť lze implementovanému automatu přesně „ušít na míru".

Kanonická síť je *implementačním schématem* automatu. Při jeho *realizaci* provádíme výběr spoje připojením elektrického napětí a logické násobičky realizujeme stejně jako v binárních logických obvodech. Schéma realizace kanonické sítě z Obr. 7.20 reléovým kontaktovým obvodem je na Obr. 7.21. Popisuje stejnou situaci (aktuální stav α, aktuální vstup C) jako schéma na Obr. 7.19. V obou schématech je zapojen stejný počet kontaktů (31), kanonická síť vyžaduje 2 relé navíc. Homogenita obvodu (jen spínací prvky), stejně dlouhé větve (neexistence hazardů) a jednoduchost implementace předurčuje kanonickou síť k realizaci technologií VLSI. I zde je pro správnou funkci obvodu zapotřebí, aby se frekvence příchodu vstupních dat rovnala spínací frekvenci použitých relé.

[1] Základní rozdíl mezi binární a unární logickou sítí je v tom, že každý port a spoj v binární síti představuje binární **proměnnou** s hodnotami 0 a 1, zatímco v unární síti představuje jen konkrétní **hodnotu** nějaké proměnné (nikoli nutně binární); přiřazení symbolů 1 či 0 informuje, zda byla či nebyla tato hodnota vybrána (tj. zda je či není aktuální hodnotou). Proměnná je v unární síti reprezentována svazkem spojů či portů (množinou všech svých možných hodnot).

[2] Pokud vstupní proměnná I přichází v jiném než unárním kódu, je třeba ji do unárního kódu nejprve převést. Převodem hodnot z binárního do unárního kódu jsme se zabývali v Příkladu 6.3 cvičení k lekci 6.

[3] Pokud bychom si přáli, aby výstupní proměnná O odcházela v jiném než unárním kódu, museli bychom ji do tohoto kódu výstupním převodníkem převést.



Obr. 7.21

7.4. Jiné možnosti implementace a realizace automatu

V předchozích dvou článcích jsme v závěrečné fázi procesu realizace automatu logickou síť popsanou výsledným systémem Booleovských výrazů či kanonickou sítí znázornili schématem zapojení cívek a kontaktů elektromagnetických relé. S reléovými systémy tohoto typu se však v moderní automatizační, řídící, telekomunikační a výpočetní technice prakticky již nesetkáme. Reléové obvody sehrály významnou roli jen v samých počátcích rozvoje těchto oborů, a v průběhu technologické revoluce byly postupně nahrazovány obvody elektronkovými, tranzistorovými, obvody na bázi VLSI a obvody optoelektronickými. Tím ale vývoj ještě zdaleka neskončil. Současný jeho trend směřující k extrémní miniaturizaci, k dalšímu snížení reakčních dob logických členů a k redukci energetických ztrát spojených s překlápěním fyzikálních systémů je usměrňován zejména pokrokem v rozvoji tzv. nanotechnologií.[4]

I když přechod na modernější technologii je přechodem do kvalitativně odlišného prostředí kladoucího vyšší nároky zejména na technické vybavení výrobce obvodů, principy implementace se jím nijak zásadně nemění. Pouze komponenty logické sítě (logické moduly) tam mohou být komplexnější. Vzhledem k tomu, že předmětem zkoumání teorie automatů jsou obecné principy konstrukce a chování automatů, nezávislé na jejich konkrétním provedení, zaměření praktické části výkladu na reléové obvody, nevyžadující žádné speciální fyzikální ani technické znalosti, je didakticky nejvhodnější. A proto u něj i v dalším setrváme.

Naproti tomu k podstatnému zjednodušení implementační i realizační fáze dochází v případě **simulace automatu číslicovým počítačem**. Počítač je mnohem sofistikovanější prostředek než elektromagnetické relé. Je to stroj na modelování (interpretaci) algoritmů. Proto se zde přenesení struktury automatu do struktury počítače realizuje softwarově (tj. vhodným programem). Implementačním modelem je vývojový diagram příslušného algoritmu, v němž jsou prvky konečných množin automatu reprezentovány hodnotami proměnných typu INTEGER. Přechodovou a výstupní funkci reprezentují proměnné typu POLE. Je zadána počáteční hodnota stavové proměnné, aktuální hodnoty vstupní proměnné zadává vstupní zařízení. V závislosti na okamžitých hodnotách vstupní a stavové proměnné se vyhledá hodnota přechodové funkce (nový aktuální stav), a výstupní funkce (aktuální výstup). Pak se aktuálním stavem přepíše zadávaný počáteční stav.

Na mikroprocesor určený k **simulaci automatu mikroprocesorem** lze pohlížet jako na jisté torzo číslicového počítače, obsahující jen obvody k simulaci automatu nezbytné. Základem je paměť typu EPROM. Do paměti jsou ve vhodném kódu zapsány tabulky přechodové a výstupní funkce simulovaného automatu. Algoritmus čtení hodnot paměti popsaný v předchozím případě zde bývá realizován hardwarově. Přečtený výstup bývá předán řadiči procesoru, který podle něho sestavuje a řídí svůj systém ovládaných výstupních modulů. Provést změnu či inovaci automatu je v tomto případě jednoduché, stačí jen přepsat paměť EPROM.

---

[4] nano - 10^{-9}; nanotechnologie je obor zabývající se jevy v objektech o rozměrech 10^{-9} m, tj. obsahujících jen několik vrstev atomů. Uznávaným „otcem nanotechnologie" je laureát Nobelovy ceny za fyziku Richard P. Feynman. Její základy formuloval ve slavné přednášce proslovené 29.12.1959 na zasedání Americké fyzikální společnosti, v níž předběhl dobu o celá desetiletí (viz její překlad „Tam dole je spousta místa" ve [4]). Principiálními možnostmi redukce rozměrů, redukce reakčních dob a energetických ztrát v obvodech počítačů se Feynman zabýval v přednášce proslovené krátce nato v Japonsku (viz její překlad „Počítače v budoucnosti" tamtéž).

7.5. Analýza schémat

V článcích 7.1 až 7.3 jsme k obrazu *abstraktního* stroje, jehož dynamická struktura byla popsána zadaným konečným automatem, konstruovali *reálný* stroj. Výsledkem tohoto úsilí bylo schéma zapojení logických členů nebo schéma zapojení cívek a kontaktů elektromagnetických relé. V tomto článku budeme řešit úlohu opačnou: Budeme mít za úkol ze schématu zapojení zjistit, co daný stroj „dělá", a výsledek zjištění prezentovat ve formě konečného automatu.

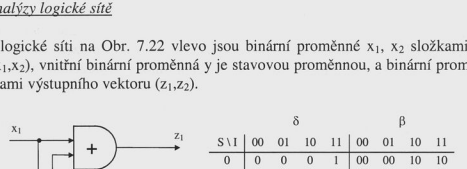
Obecný postup řešení této úlohy je možno rozložit do následujících tří kroků:

- Ze schématu zapojení odstraníme synchronizační obvody nebo opakovače (vyrovnávací zpožďovací členy) a zrestaurujeme torzo schématu, se kterým budeme dále pracovat.
- Ve schématu vyznačíme vstupní, výstupní a vnitřní binární proměnné a z jeho struktury odvodíme soustavu Booleovských výrazů popisujících závislost vnitřních a výstupních proměnných na vstupních a vnitřních proměnných. Sestavíme vektory vstupních, vnitřních a výstupních binárních proměnných a s ohledem na závislost jejich složek určíme množiny I, S a O konečného automatu.
- Z vypočtených hodnot Booleovských výrazů pro relevantní kombinace prvků množin S a I sestavíme tabulku přechodové a výstupní funkce automatu.

Poslední dva kroky postupu jsou ilustrovány jak příkladem analýzy jednoduché logické sítě, tak příkladem analýzy jednoduchého kontaktového obvodu.

Příklad analýzy logické sítě

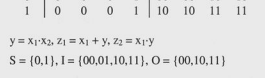
V logické síti na Obr. 7.22 vlevo jsou binární proměnné x_1, x_2 složkami vstupního vektoru (x_1,x_2), vnitřní binární proměnná y je stavovou proměnnou, a binární proměnné z_1, z_2 jsou složkami výstupního vektoru (z_1,z_2).



	δ				β			
S \ I	00	01	10	11	00	01	10	11
0	0	0	0	1	00	00	10	10
1	0	0	0	1	10	10	11	11

$y = x_1 \cdot x_2$, $z_1 = x_1 + y$, $z_2 = x_1 \cdot y$

$S = \{0,1\}$, $I = \{00,01,10,11\}$, $O = \{00,10,11\}$



Obr. 7.22

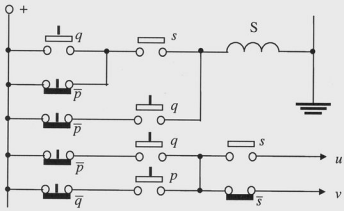
Závislost vnitřních a výstupních proměnných na vstupních a vnitřních proměnných je popsána trojicí Booleovských výrazů $y = x_1 \cdot x_2$, $z_1 = x_1 + y$, $z_2 = x_1 \cdot y$. Z nich vyplývá, že je-li $z_1 = 0$, je i $z_2 = 0$, a tak kombinace $(z_1,z_2) = (0,1)$ nemůže nastat. Proto také $01 \notin O$. Pro přechodovou a výstupní funkci platí:

$$\delta(y,(x_1,x_2)) = x_1 \cdot x_2, \text{ a } \beta(y,(x_1,x_2)) = (z_1,z_2) = (x_1+y,\ x_1 \cdot y).$$

Např. $\delta(0,(1,1)) = 1 \cdot 1 = 1$ a $\beta(0,(1,1)) = (1+0,\ 1 \cdot 0) = (1,0)$; takto byly vypočteny hodnoty v polích tabulky na Obr. 7.22 vpravo nahoře, která je předlohou pro sestrojení přechodového grafu umístěného vpravo dole.

Příklad analýzy kontaktového obvodu

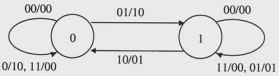
V kontaktovém obvodu na Obr. 7.23 jsou vstupními binárními podněty stavy tlačítek P a Q, jejichž spínací a rozpínací kontakty jsou do obvodu zapojeny. Vnitřní binární proměnnou je stav relé S (s hodnotami sepnuto, nesepnuto), výstupními binárními proměnnými jsou proměnné u a v.



$$s \cong \overline{p} \cdot q + (\overline{p} + q) \cdot s$$
$$u = (\overline{p} \cdot q + p \cdot \overline{q}) \cdot s$$
$$v = (\overline{p} \cdot q + p \cdot \overline{q}) \cdot \overline{s}$$

S = {0,1}, I = {00,01,10,11}, O = {00,10,01}

	δ				β			
S \ I	00	01	10	11	00	01	10	11
0	0	1	0	0	00	10	10	00
1	1	1	0	1	00	01	01	00



Obr. 7.23

Přiřadíme-li podnětu „stisknuté tlačítko" a stavu „sepnuté relé" hodnotu 1 („nestisknuté tlačítko" a „nesepnuté relé" - hodnoty 0), je vstupním vektorem dvojice (p,q), stavovým vektorem proměnná s a výstupním vektorem dvojice (u,v). Ze schématu odvodíme Booleovské výrazy uvedené na Obr. 7.23 vpravo.[5] Z nich (a stejně tak ze schématu) je okamžitě zřejmé, že $11 \notin O$. Pro přechodovou a výstupní funkci platí:

$$\delta(s,(p,q)) = \overline{p} \cdot q + (\overline{p} + q) \cdot s\,,$$

$$\beta(s,(p,q)) = (u,v) = ((\overline{p} \cdot q + p \cdot \overline{q}) \cdot s\,,\ (\overline{p} \cdot q + p \cdot \overline{q}) \cdot \overline{s}\,).$$

Např. $\delta(0,(1,1)) = 0 \cdot 1 + (0+1) \cdot 0 = 0$ a $\beta(0,(1,1)) = ((0 \cdot 1 + 1 \cdot 0) \cdot 0,\ (0 \cdot 1 + 1 \cdot 0) \cdot 1) = (0,0)$.

Z tabulky i grafu výsledného automatu vidíme, že analyzovaný kontaktový obvod je reléovou realizací posuvného registru, který reaguje jen na neshodu podnětů na vstupních tlačítkách a shodu podnětů ignoruje (dává na výstupu klidový signál 00). Neshodu podnětů

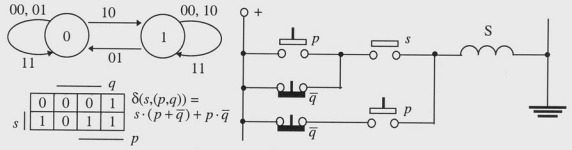
[5] Symbol $\cong$ říká, že následující hodnotou proměnné na levé straně bude aktuální hodnota výrazu na straně pravé.

propouští na výstup zpožděnou o takt (tj. do výskytu další neshody) a typ neshody uchovává ve svých stavech.

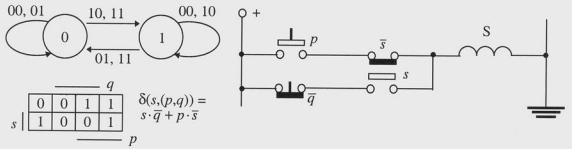
Cvičení k lekci 7.

Příklad 7.1 (bistabilní klopné obvody ovládané dvěma tlačítky):

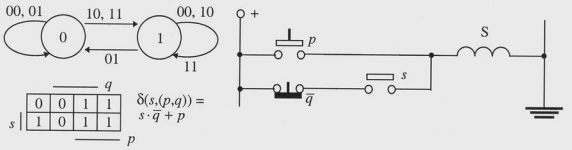
Na Obr. 6.25 vpravo a na Obr. 6.26 byly uvedeny příklady zapojení bistabilních klopných obvodů ovládaných bistabilními ovládacími prvky (spínači a přepínači). V sérii následujících čtyř obrázků vidíme příklady zapojení bistabilních klopných obvodů ovládaných dvěma monostabilními ovládacími prvky (zapínacím tlačítkem P a vypínacím tlačítkem Q). V nich si relé „pamatuje" poslední stisk tlačítek.[11] Jednotlivé varianty zapojení se liší jen reakcí na současný stisk obou tlačítek: V prvním případě obvod svůj stav nemění, ve druhém mění (a jsou-li tlačítka současně tisknuta delší dobu, funguje jako astabilní multivibrátor), ve třetím je preferován stav 1, ve čtvrtém stav 0.



Obr. 7.24

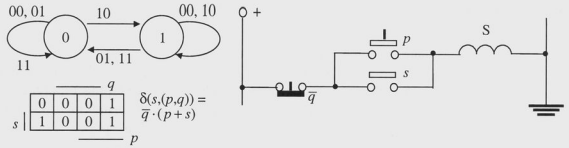


Obr. 7.25



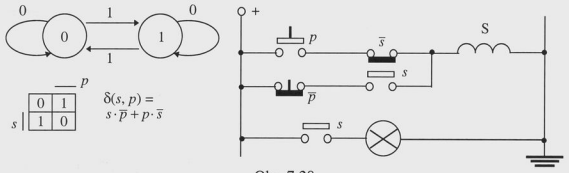
Obr. 7.26

[11] V odborném žargonu se takto zapojeným relé říká „paměťová relé".



Obr. 7.27

Příklad 7.2 (bistabilní klopný obvod ovládaný jedním tlačítkem):
Zaměníme-li v Obr. 7.25 vypínací kontakt tlačítka Q vypínacím kontaktem tlačítka P a použijeme-li relé S s dostatečně dlouhými reakčními dobami, získáme bistabilní klopný obvod ovládaný jen jedním tlačítkem - viz Obr. 7.28, na němž je obvod využit k zapínání a vypínání žárovky. Ke správné funkci obvodu je zapotřebí uvolnit stisk tlačítka bezprostředně poté, co uslyšíme charakteristické „cvaknutí" kotvy relé.



Obr. 7.28

Příklad 7.3 (návrh schématu zapojení řídícího obvodu dvojice stykačů pro reverzaci chodu třífázového asynchronního indukčního elektromotoru).

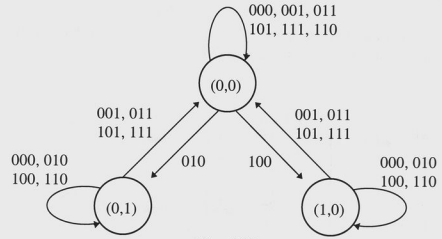
Zadání:

Pro zapojení silových obvodů stykačů dle Obr. 6.27 navrhnout adekvátní zapojení řídícího obvodu, v němž jsou stykače S a T ovládány třemi tlačítky A, B a C tak, že:

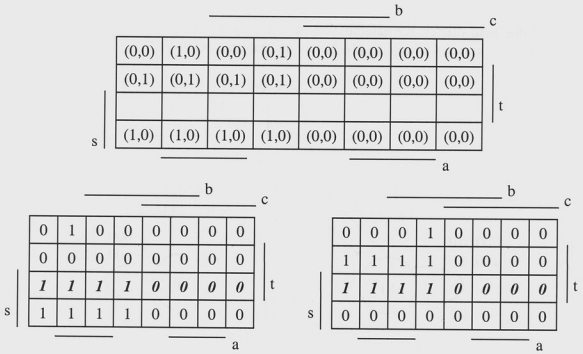
- tlačítko C je vypínací a po jeho stisknutí stykače motor odpojí od sítě (vypnou), a to bez ohledu na to, je-li tlačítko C stisknuto samostatně či v kombinaci s tlačítky A, B;
- tlačítka A a B jsou zapínací, každé pro jiný směr chodu motoru; na zapínací tlačítka stykače reagují jen jsou-li oba ve vypnutém stavu (a tedy i motor je od sítě odpojen - viz Obr. 6.27), přičemž současný stisk obou zapínacích tlačítek stykače ignorují.

Řešení:

V souladu s výše uvedenými požadavky je možno systém ovládání stykačů formulovat jako stavový automat s přechodovým grafem podle Obr. 7.29.[12] V něm dvojice čísel označující stavy tohoto automatu udávají aktuální kombinace stavů dvojice stykačů (S,T), a trojice čísel v označení hran udávají aktuální kombinace vstupních podnětů z tlačítek ABC: sepnutému stykači či stisknutému tlačítku je přiřazena hodnota 1.



Obr. 7.29



Obr. 7.30

[12] Aktuálním výstupem systému je vždy jen jeden ze tří žádaných stavů motoru: Motor může být buď od sítě odpojen, nebo může být k síti připojen tak aby běžel jedním či druhým směrem. Výstupní funkce je definována na stavech našeho automatu (jde o Mooreův automat 2. druhu), a je zajištěna uvedeným zapojením silových kontaktů stykačů do obvodu el. napájení motoru - viz Obr.6.27. Proto není třeba se jí zde zabývat.

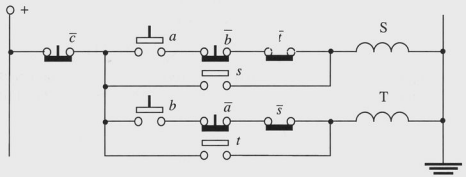
V horní části Obr. 7.30 je uveden přepis přechodového grafu z Obr. 7.29 do Karnaughovy mapy. Z ní jsou odvozeny níže položené dílčí Karnaughovy mapy, levá pro stykač S, pravá pro stykač T. Tlustě a proloženě je v nich vyznačena volba přechodů ze stavu (1,1), do něhož se v normálním provozním režimu nelze dostat. Tato volba podstatně zjednodušuje následující logické výrazy odvozené z map. Pro stykač S:

$$s^* = \bar{c} \cdot (s + \bar{t} \cdot a \cdot \bar{b}),$$

pro stykač T:

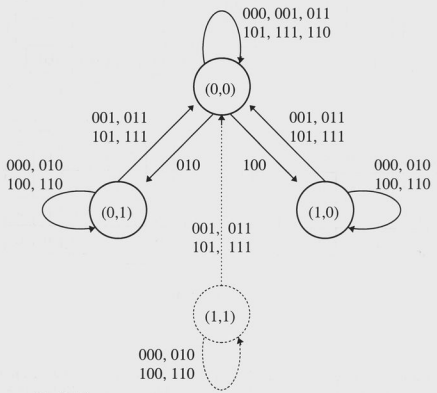
$$t^* = \bar{c} \cdot (t + \bar{s} \cdot \bar{a} \cdot b),$$

kde s^* a t^* jsou následné stavy stykačů S a T. Realizací těchto výrazů vzniklo schéma zapojení kontaktů a cívek stykačů na Obr. 7.31.



Obr. 7.31

Z doplnění dílčích Karnaughových map je zřejmé, že kontaktový obvod z Obr. 7.31 realizuje stavový automat s přechodovým grafem na Obr. 7.32

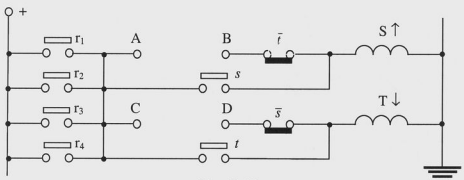


Obr.7.32

Úloha 7.1: Porovnáním schématu na Obr. 7.31 se schématem na Obr. 7.27 zjistíme, že obvod na Obr. 7.31 sestává ze dvou paměťových relé doplněných o vzájemně se blokující rozpínací kontakty zařazené v hlavní větvi (tzv. paměťová relé se vzájemnou blokací). Obvod si „pamatuje", které z tlačítek A či B bylo v klidovém stavu stisknuto jako první a tento paměťový záznam je vymazán až stiskem vypínacího tlačítka C. Využijte tento poznatek k návrhu obvodu pro ovládání N relé N zapínacími tlačítky (jedno pro každé relé) a jedním (společným) vypínacím tlačítkem tak, aby manipulací s tlačítky nikdy nemohlo dojít k současnému sepnutí dvou a více relé.

Úloha 7.2 (návrh ovládání motoru výtahu):

Uvažujte výtah pro čtyři podlaží. V každém podlaží je jedno přivolávací tlačítko (označení T_1^* až T_4^*) a koncový spínač (K_1 až K_4) rozpínaný kabinou v okamžiku jejího příjezdu (a zůstává rozepnut po celou dobu její přítomnosti v podlaží). V kabině jsou umístěna čtyři cílová tlačítka (označení T_1 až T_4) a zátěžový spínač (Z), který po vstupu do kabiny vyřazuje ze hry venkovní tlačítka. Využijte schéma pro reverzaci chodu motoru (Obr. 7.31), ve kterém provedete následující změny (viz Obr. 7.33):



Obr. 7.33

Stykač S je určen pro jízdu nahoru, stykač T pro jízdu dolů. Vypínací tlačítko je zaměněno spínacími kontakty čtyř paměťových relé ze vzájemnou blokací spínaných cílovými tlačítky v kabině, případně přivolávacími tlačítky v jednotlivých podlažích. Vypnutí sepnutého paměťového relé (a tím i zastavení výtahu) provede příslušný koncový spínač cílového podlaží po dojezdu kabiny.

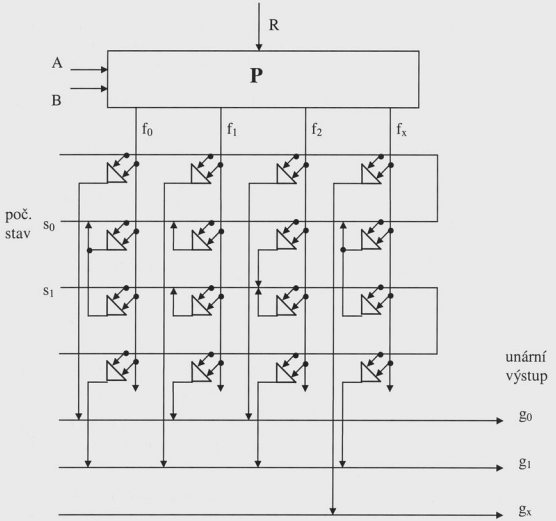
Úkoly:

- Nakreslete kontaktové schéma obvodu ovládání paměťových relé.
- Navrhněte kontaktové obvody, které je třeba připojit ke svorkám A-B a C-D, aby výtah správně fungoval, a zapojte je do obvodu.
- Posuďte své řešení z hlediska možnosti výskytu hazardů. Pokud taková možnost hrozí, změňte schéma tak, aby systém fungoval spolehlivě.

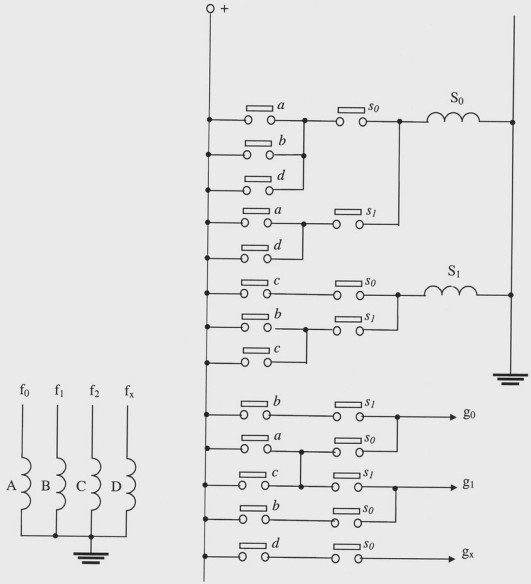
Příklad 7.4 (*kanonická síť automatu M paralelní sčítačky binárních čísel z Obr. 5.10 a její realizace*): Varianta paralelní sčítačky představená v Příkladu 5.4 je na Obr. 5.10 znázorněna sériovou kombinací převodníku P (funkce převodníku je zadána tabulkou pod schématem v Obr. 5.10) a automatu M. Tabulka přechodové a výstupní funkce Mealyho verze (ME) automatu M je uvedena na Obr. 5.11,

tabulka přechodové a výstupní funkce Mooreovy verze 2. typu (MO2) automatu M je uvedena na Obr. 5.12. Realizací převodníku P jsme se už zabývali v Příkladu 6.4 a jejím výsledkem je reléový kontaktový obvod na Obr. 6.21. Při realizaci ME vyjdeme z následující modifikace tabulky z Obr. 5.11, do níž jsme dosadili 1000 = f_0, 0100 = f_1, 0010 = f_2, 0001 = f_x, 100 = g_0, 010 = g_1, 001 = g_x. Z ní odvodíme kanonickou síť automatu ME (Obr. 7.34) a relevantní reléový kontaktový obvod (Obr. 7.35).

S \ I	f_0	f_1	f_2	f_x
s_0	(s_0,g_0)	(s_0,g_1)	(s_1,g_0)	(s_0,g_x)
s_1	(s_0,g_1)	(s_1,g_0)	(s_1,g_1)	(s_0,g_1)



Obr. 7.34

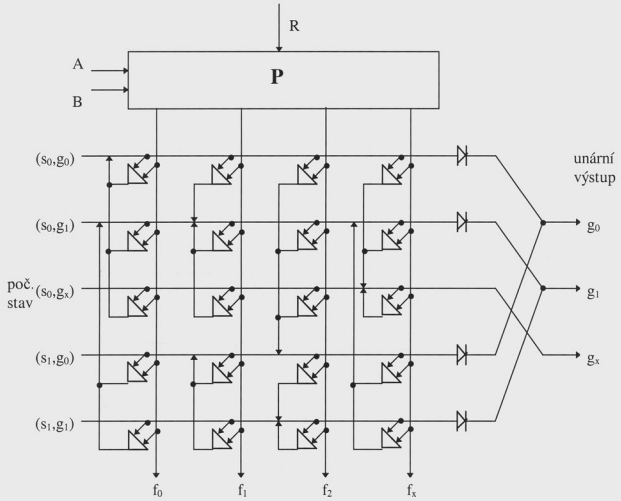


Obr. 7.35

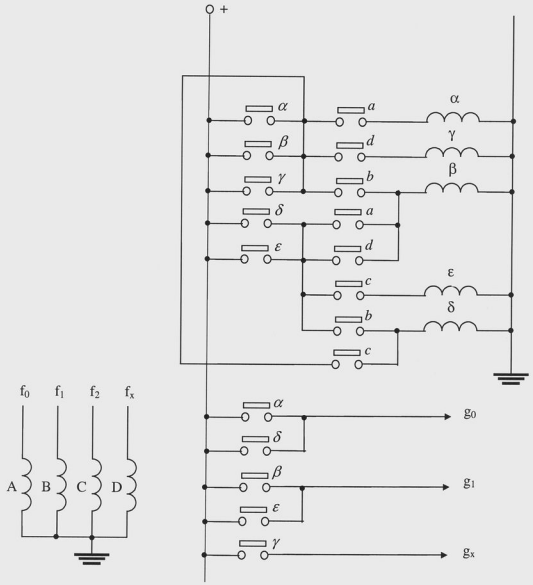
Příklad 7.5 (*totéž co v Příkladu 7.4, ale pro verzi MO2*): Při realizaci MO2 vyjdeme z následující modifikace tabulky z Obr. 5.12, do níž jsme dosadili $1000 = f_0$, $0100 = f_1$, $0010 = f_2$, $0001 = f_x$, $100 = g_0$, $010 = g_1$, $001 = g_x$. Z ní odvodíme kanonickou síť automatu MO2 (Obr. 7.36) a relevantní reléový kontaktový obvod (Obr. 7.37).

U MO2 je aktuální hodnota výstupu určena výhradně stavem, do něhož automat přechází. Při unárně kódovaném výstupu a shodných úrovních vnitřních i výstupních signálů lze funkci λ v kanonické síti realizovat jednosměrným připojením vodorovných spojů (stavů) k výstupním portům (viz Obr. 7.36). Jednosměrnost připojení je ve schématu znázorněna diodami, zabraňujícími zpětnému šíření signálů. Při reléové realizaci (viz Obr. 7.37) jsou vnější obvody od vnitřních galvanicky odděleny, proto žádné diody nejsou zapotřebí.

S \ I	f_0	f_1	f_2	f_x	λ
(s_0,g_0)	(s_0,g_0)	(s_0,g_1)	(s_1,g_0)	(s_0,g_x)	g_0
(s_0,g_1)	(s_0,g_0)	(s_0,g_1)	(s_1,g_0)	(s_0,g_x)	g_1
(s_0,g_x)	(s_0,g_0)	(s_0,g_1)	(s_1,g_0)	(s_0,g_x)	g_x
(s_1,g_0)	(s_0,g_1)	(s_1,g_0)	(s_1,g_1)	(s_0,g_1)	g_0
(s_1,g_1)	(s_0,g_1)	(s_1,g_0)	(s_1,g_1)	(s_0,g_1)	g_1

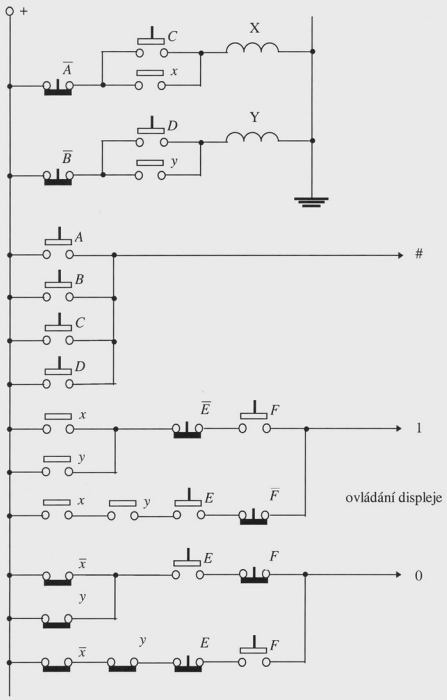


Obr. 7.36



Obr. 7.37

Příklad 7.6 (*reléová realizace Booleovského kalkulátoru z úlohy 4.4*): V řešení úlohy 4.4 jsme tlačítka na ovládacím panelu (Obr. 4.17) označili písmeny A až F. Jednou z možných reléových realizací uvedeného řešení je následující kontaktový obvod:



Obr. 7.38

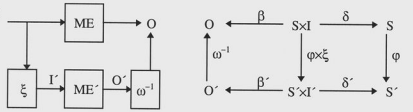
Je zřejmé, že u dvojice výstupů k displeji může být napětí nejvýše na jednom z nich. Napětí na horním výstupu rozsvítí na displeji jedničku, napětí na dolním výstupu rozsvítí na displeji nulu.

Úloha 7.3: Ke kontaktovému schématu na Obr. 7.38 sestrojte Karnaughovu mapu komponenty výstupní funkce pro ovládání displeje.

8. Strukturální podobnost automatů

8.1. Úvod

V článku 7.1. předchozí lekce jsme uvedli konkrétní příklad dvou různých zpracování vstupního podnětu se stejným výsledkem (viz levé schéma na Obr. 8.1):



Obr. 8.1

Podnět z množiny I může být:

- buď předán na vstup automatu ME (nacházejícího se v aktuálním vnitřním stavu $s \in S$) a přetvořen v reakci z množiny O vnitřním mechanismem automatu ME, nebo
- přes vstupní kodér s kódovací funkcí ξ veden na vstup automatu ME´ (nacházejícího se v aktuálním vnitřním stavu $\varphi(s) \in S'$), zpracován vnitřním mechanismem automatu ME´, a následně prohnán výstupním dekodérem s dekódovací funkcí ω^{-1};

v obou případech bude výsledek týž.

Je tomu tak proto, že algebraický diagram relevantních zobrazení (na Obr. 8.1 uprostřed), podrobně zdůvodněný v poznámkách pod čarou 3 a 4 v lekci 7., komutuje (tj. splňuje dvojici podmínek na Obr. 8.1 vpravo). Věcně je tato komutace důsledkem dvojice spolu souvisejících faktorů:

- existence fundamentální analogie ve vnitřních mechanismech automatů ME a ME´, a
- znalosti dvojic podobně fungujících prvků v jejich dynamických strukturách.

V našem případě je zmíněná fundamentální analogie důsledkem věrného přenosu struktury, a dvojice podobně fungujících prvků jsou přitom vázány prostými zobrazeními (injekcemi) φ (definuje dvojice ekvivalentních stavů), ξ (ukazuje na analogicky působící podněty) a ω (uvádí do souvislostí odpovídající reakce).[1]

Zatímco injektivnost zobrazení ω je nezbytně nutná (bez ní by podmínka $\omega^{-1}\circ\omega = \underline{1}$ nebyla splnitelná a výstupní dekodér zajišťující komutaci levého okna v diagramu na Obr. 8.1 by nebylo možno sestrojit), injektivnost zobrazení φ a ξ nezbytně nutná není. Budou-li φ a ξ kontraktivní, pak všechny situace $(s,i) \in S\times I$ automatu ME, spadající do stejné třídy rozkladu indukovaného jádrem zobrazení $\varphi\times\xi$,[2] budou pro automat ME´ jedna od druhé nerozlišitelné a jeho mechanismus na ně bude reagovat shodně. Komutace algebraického diagramu nám pak říká, že na tyto situace shodně reaguje i mechanismus automatu ME (jinak by totiž diagram

[1] Naproti tomu transformační funkce ω^{-1} injektivní nebyla a v našem příkladu ani injektivní býti nemohla (viz Obr. 7.12). Podmínka $\omega^{-1}\circ\omega = \underline{1}$ vyžaduje jen její surjektivitu (tj. býti „na" - nebo-li pokrýt celý obor hodnot) a tento požadavek je spolu s ostatními požadavky na funkci ω^{-1} podmínkou $\omega^{-1}\circ\beta\circ(\varphi\times\xi) = \beta$ náležitě zohledněn.

[2] Jedná se o ekvivalenci $\ker(\varphi\times\xi)$ na $S\times I$, v níž pro každé dva prvky (s_1,i_1) a (s_2,i_2) ze stejné třídy rozkladu platí: $\varphi(s_1) = \varphi(s_2)$ a $\xi(i_1) = \xi(i_2)$.

komutovat nemohl), a proto nižší rozlišovací schopnost automatu ME´ existenci fundamentální analogie mezi mechanismy v cestě nestojí.[3] Fundamentální analogie existovat bude, i když nebude tak očividná jako po věrném přenosu struktury, protože „stejně fungující podstruktura" v automatu ME´ již nebude izomorfní se strukturou automatu ME.

Takto obecněji pojatou fundamentální analogii mezi vnitřními mechanismy automatů nazýváme **strukturální podobností** automatů.[4] Věrný přenos struktury, který provádíme při implementaci, je pro ni podmínkou postačující, nikoli však nutnou. Existence strukturální podobnosti není vázána na existenci podstruktury H´ v automatu ME´ izomorfní se strukturou automatu ME (znázorněné na Obr. 7.1). Nutná a postačující zároveň je jen existence trojice $(\varphi,\xi,\omega^{-1})$ libovolných zobrazení mezi definičními množinami obou automatů, zajišťující komutaci výše uvedeného algebraického diagramu. Věcným důsledkem toho je fakt, že sériovaraleln í kompozice automatů a převodníků (na Obr. 8.1 vlevo) dává na obou výstupech shodné reakce.

8.2. Strukturální podobnost jako podobnost ke vzoru[5]

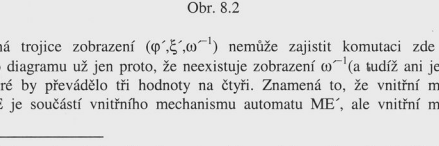
V našem konkrétním případě automatů ME a ME´ (viz příklad v textu lekce 7) tvoří jednu z možných trojic $(\varphi,\xi,\omega^{-1})$, zajišťujících komutaci algebraického diagramu, zobrazení $\varphi: S \to S'$, $\xi: I \to I'$, $\omega^{-1}: O' \to O$, definovaná např. takto:

s	α	β	γ
φ(s)	00	01	10

i	A	B	C
ξ(i)	01	10	11

o	00	01	10	11
ω^{-1}(**o**)	a	b	b	c

Kdybychom ve schématu v Obr. 8.1 zaměnili vzor za obraz a naopak, dostali bychom Obr. 8.2:



Obr. 8.2

V něm žádná trojice zobrazení $(\varphi',\xi',\omega'^{-1})$ nemůže zajistit komutaci zde uvedeného algebraického diagramu už jen proto, že neexistuje zobrazení ω'^{-1}(a tudíž ani jej realizující dekodér), které by převádělo tři hodnoty na čtyři. Znamená to, že vnitřní mechanismus automatu ME je součástí vnitřního mechanismu automatu ME´, ale vnitřní mechanismus

[3] V důkazu ekvivalentnosti stavů vázaných zobrazením φ (viz poznámka pod čarou 5 v lekci 7) žádné speciální vlastnosti zobrazení φ, ξ, ω^{-1} předpokládány nebyly.

[4] Mluvíme-li o strukturální podobnosti, máme na mysli strukturu dynamickou, nikoli morfologickou. Jen z podobnosti morfologických struktur nelze usuzovat na podobnost v dynamických strukturách celků.

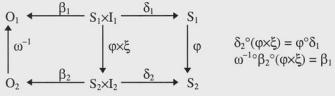
[5] Logika zná dva typy podobnosti: Symetrickou, v níž je-li A podobno B, pak také B je podobno A; a tzv. podobnost ke vzoru, která symetrická není. Řeknu-li např. o nějakém malém chlapci, že hraje na housle jak Paganini, nechci tím říci, že Paganini hrál podobně jako tento chlapec. Řeknu-li o nějakém mraku, že se podobá velrybě, nechci tím říci, že velryba se podobá mraku. Paganini i velryba jsou v tomto případě vzory a jsou svým způsobem obsažnější: Co zahraje chlapec, jistě by zahrál i Paganini (a pravděpodobně i lépe), opačně to platit nemusí.

automatu ME´ není součástí vnitřního mechanismu automatu ME. Strukturální podobnost tak není typem symetrické podobnosti, ale typem nesymetrické podobnosti, tzv. **podobnosti ke vzoru**, kde vzorem podobnosti je automat ME´.[6]

Nicméně i zde zjišťovat podobnost znamená zjišťovat stupeň shody. A podle stupně shody měříme i stupeň podobnosti. V matematice nejvyššímu stupni podobnosti říkáme **identita** neboli **rovnost**. Dva automaty $ME_1 = (S_1, I_1, O_1, \delta_1, \beta_1)$ a $ME_2 = (S_2, I_2, O_2, \delta_2, \beta_2)$ jsou identické, tj. $ME_1 = ME_2$, právě když $S_1 = S_2$, $I_1 = I_2$, $O_1 = O_2$, $\delta_1 = \delta_2$, $\beta_1 = \beta_2$. Že v tomto případě se jedná o symetrickou strukturální podobnost, a že identické automaty jsou také ekvivalentními automaty (z hlediska chování), jistě nepřekvapí. Přesněji jsou různé stupně strukturální podobnosti vymezeny v následujícím článku.

8.3. Vstup-výstupní homomorfismus

Nechť M_1 a M_2 jsou automaty. V případě Mealyho automatů $M_1 = (S_1, I_1, O_1, \delta_1, \beta_1)$ a $M_2 = (S_2, I_2, O_2, \delta_2, \beta_2)$. Pak trojice $\phi = (\varphi,\xi,\omega^{-1})$, kde $\varphi: S_1 \to S_2$, $\xi: I_1 \to I_2$, $\omega^{-1}: O_2 \to O_1$, jsou zobrazení splňující podmínky $\delta_2\circ(\varphi\times\xi) = \varphi\circ\delta_1$ a $\omega^{-1}\circ\beta_2\circ(\varphi\times\xi) = \beta_1$ komutace obou oken algebraického diagramu



Obr. 8.3

se nazývá **vstup-výstupním homomorfismem** (zkratka VVH) z M_1 do M_2. Existenci VVH ϕ z M_1 do M_2 v obecném případě zapisujeme výrazem $\phi: M_1 \to M_2$.

Speciální VVH:

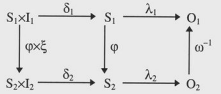
- Je-li ve VVH $\phi = (\varphi,\xi,\omega^{-1})$ zobrazení $\varphi: S_1 \to S_2$ injektivní, říkáme, že M_2 **realizuje** M_1 a píšeme $\phi: M_1 \leq M_2$.
- Jsou-li ve VVH $\phi = (\varphi,\xi,\omega^{-1})$ všechna zobrazení bijektivní, jde o **izomorfismus** automatů $\phi: M_1 \approx M_2$. V případě identit ($\phi = (\underline{1},\underline{1},\underline{1})$) jde o **rovnost** automatů $M_1 = M_2$.
- Speciální VVH $\phi = (\varphi,\underline{1},\underline{1})$, kde $\varphi: S_1 \to S_2$ je surjektivní a ξ, ω^{-1} jsou identity, nazýváme redukčním VVH neboli **redukcí**.

Speciální VVH popisují různé stupně strukturální podobnosti. Nejvyšším stupněm podobnosti je rovnost automatů. Těsně pod ní je izomorfismus, neboli identita s přejmenováním prvků. V obou případech jde o symetrickou podobnost, neboť z existence $\phi = (\varphi,\xi,\omega^{-1})$: $M_1 \to M_2$ bezprostředně plyne $\phi' = (\varphi',\xi',\omega'^{-1})$: $M_2 \to M_1$, kde φ', ξ', a ω'^{-1} jsou zobrazení inverzní k φ, ξ, a ω^{-1}. V ostatních případech jde o nesymetrickou podobnost ke vzoru. Při $\phi: M_1 \leq M_2$ jsme blízko tomu, co jsme při implementaci nazývali věrným přenosem struktury. Naproti tomu existence redukce s kontraktivní funkcí φ signalizuje možnost snížení počtu stavů automatu M_1.

[6] Implementace a teorie podobnosti posuzují funkci systému z opačných úhlů. Proto pro implementaci je vzorem ME, pro podobnost je vzorem ME´.

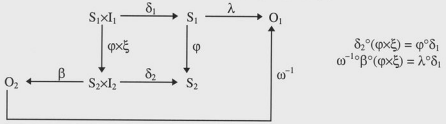
VVH $\phi = (\varphi,\xi,\omega^{-1})$: $M_1 \to M_2$ může existovat i mezi různými modely automatů. Tomu pak odpovídají různé algebraické diagramy a komutační podmínky, např.:

a) $M_1 = MO2_1 = (S_1, I_1, O_1, \delta_1, \lambda_1)$, $M_2 = MO2_2 = (S_2, I_2, O_2, \delta_2, \lambda_2)$



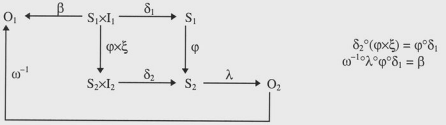
Obr. 8.4

b) $M_1 = MO2 = (S_1, I_1, O_1, \delta_1, \lambda)$, $M_2 = ME = (S_2, I_2, O_2, \delta_2, \beta)$



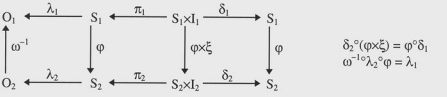
Obr. 8.5

c) $M_1 = ME = (S_1, I_1, O_1, \delta_1, \beta)$, $M_2 = MO2 = (S_2, I_2, O_2, \delta_2, \lambda)$



Obr. 8.6

d) $M_1 = MO1_1 = (S_1, I_1, O_1, \delta_1, \lambda_1)$, $M_2 = MO1_2 = (S_2, I_2, O_2, \delta_2, \lambda_2)$



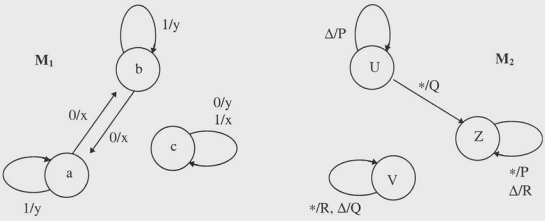
Obr. 8.7

V posledním případě prostřední okno diagramu komutuje z definice, neboť $\pi_1{}^\circ(\varphi\times\xi) = \varphi^\circ\pi_1$, a proto se zde podmínky komutace shodují s podmínkami v případě a).

Dokázat existenci strukturální podobnosti předpokládá nalézt vhodnou trojici zobrazení φ, ξ, a ω^{-1} a přesvědčit se, že splňuje obě komutační podmínky pro všechny kombinace vstupů (viz příklad 8.1 cvičení).

Cvičení k lekci 8.

Příklad 8.1: Dány automaty M_1 a M_2 (Obr. 8.8),



Obr. 8.8

a trojice zobrazení $\phi = (\varphi,\xi,\omega^{-1})$, kde φ: $S_1 \to S_2$, ξ: $I_1 \to I_2$, ω^{-1}: $O_2 \to O_1$ ve tvaru

s_1	a	b	c
$\varphi(s_1)$	V	V	Z

i_1	0	1
$\xi(i_1)$	Δ	*

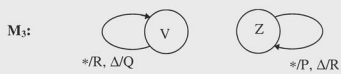
o_2	P	Q	R
$\omega^{-1}(o_2)$	x	x	y

Obr. 8.9

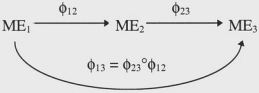
Má se dokázat, že ϕ: $M_1 \to M_2$ je homomorfismus. Vzhledem k tomu, že v obou případech se jedná o Mealyho automaty, k důkazu stačí ověřit platnost podmínek $\delta_2\circ(\varphi\times\xi) = \varphi\circ\delta_1$ a $\omega^{-1}\circ\beta_2\circ(\varphi\times\xi) = \beta_1$ (viz Obr. 8.3). Výsledek úplné kontroly obou podmínek (pro všechny kombinace vstupních hodnot z množiny $S_1\times I_1$) uvádí následující tabulka:

	$\delta_2\circ(\varphi\times\xi)$	$\varphi\circ\delta_1$	$\omega^{-1}\circ\beta_2\circ(\varphi\times\xi)$	β_1
(a,0)	V	V	x	x
(a,1)	V	V	y	y
(b,0)	V	V	x	x
(b,1)	V	V	y	y
(c,0)	Z	Z	y	y
(c,1)	Z	Z	x	x

Z výsledku kontroly je zřejmé, že homomorfismem je i ϕ: $M_1 \to M_3$, kde M_3 je komponenta automatu M_2, definovaná na stavech V a Z (viz následující graf):



Příklad 8.2 (důkaz, že kompozicí VVH je také VVH):
Dány Mealyho automaty $ME_i = (S_i, I_i, O_i, \delta_i, \beta_i)$, i = 1,2,3 a VVH
$\phi_{12} = (\varphi_1, \xi_1, \omega_1^{-1})$: $ME_1 \to ME_2$ a $\phi_{23} = (\varphi_2, \xi_2, \omega_2^{-1})$: $ME_2 \to ME_3$. Má se dokázat, že kompozice $\phi_{23}\circ\phi_{12} = \phi_{13} = (\varphi_2\circ\varphi_1, \xi_2\circ\xi_1, \omega_2^{-1}\circ\omega_1^{-1})$: $ME_1 \to ME_3$ (viz Obr. 8.10) je také VVH.



Obr. 8.10

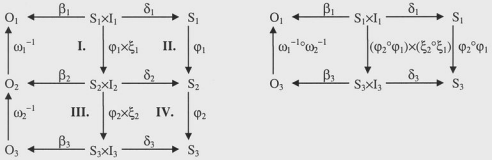
Nejprve dokážeme platnost vztahu

$$(\varphi_2\times\xi_2)\circ(\varphi_1\times\xi_1) = (\varphi_2\circ\varphi_1)\times(\xi_2\circ\xi_1); \qquad (A)$$

Z definice symbolů operací (viz článek 3.1.) totiž postupně plyne:

$$[(\varphi_2\times\xi_2)\circ(\varphi_1\times\xi_1)](s,i) = [(\varphi_2\times\xi_2)](\varphi_1(s),\xi_1(i)) = (\varphi_2(\varphi_1(s)), \xi_2(\xi_1(i))) =$$
$$= ([\varphi_2\circ\varphi_1](s),[\xi_2\circ\xi_1](i)) = [(\varphi_2\circ\varphi_1)\times(\xi_2\circ\xi_1)](s,i).$$

Existence dílčích VVH ϕ_{12} a ϕ_{23} signalizuje komutaci vyznačených oken v levém diagramu na Obr. 8.11:



Obr. 8.11

Proto platí:

I. $\omega_1^{-1}\circ\beta_2\circ(\varphi_1\times\xi_1) = \beta_1$,

II. $\delta_2\circ(\varphi_1\times\xi_1) = \varphi_1\circ\delta_1$,

III. $\omega_2^{-1}\circ\beta_3\circ(\varphi_2\times\xi_2) = \beta_2$,

IV. $\delta_3\circ(\varphi_2\times\xi_2) = \varphi_2\circ\delta_2$.

Je třeba dokázat, že pak komutuje i pravý diagram na Obr. 8.11, tj. že platí:

$$\beta_1 = \omega_1^{-1}\circ\omega_2^{-1}\circ\beta_3\circ((\varphi_2\circ\varphi_1)\times(\xi_2\circ\xi_1)),$$
$$\varphi_2\circ\varphi_1\circ\delta_1 = \delta_3\circ((\varphi_2\circ\varphi_1)\times(\xi_2\circ\xi_1)).$$

Důkaz (horní index u rovnítka odkazuje na použitý předpoklad pro substituci):

- $\beta_1 =^{I.} \omega_1^{-1} \circ \beta_2 \circ (\varphi_1 \times \xi_1) =^{III.} \omega_1^{-1} \circ \omega_2^{-1} \circ \beta_3 \circ (\varphi_2 \times \xi_2) \circ (\varphi_1 \times \xi_1) =^{(A)}$
  $\omega_1^{-1} \circ \omega_2^{-1} \circ \beta_3 \circ ((\varphi_2 \circ \varphi_1) \times (\xi_2 \circ \xi_1))$,
- $\varphi_2 \circ \varphi_1 \circ \delta_1 =^{II.} \varphi_2 \circ \delta_2 \circ (\varphi_1 \times \xi_1) =^{IV.} \delta_3 \circ (\varphi_2 \times \xi_2) \circ (\varphi_1 \times \xi_1) =^{(A)} \delta_3 \circ ((\varphi_2 \circ \varphi_1) \times (\xi_2 \circ \xi_1))$.

Úloha 8.1: Dokažte, že trojice $(\varphi, \xi, \omega^{-1})$ z příkladu implementace v lekci 7 je VVH bez ohledu na to, jakou hodnotu $\omega^{-1}(10)$ zvolíme.

9. Funkční podobnost automatů

Na rozdíl od strukturální podobnosti automatů, která je podobností v jejich vnitřních mechanismech, funkční podobnost automatů je podobností v jejich vnějších projevech, tedy podobností v chování. Zde se ptáme, do jaké míry mohou srovnávané automaty vykonávat tutéž funkci, tj. do jaké míry mohou být funkčně zaměnitelné v systémech, do kterých jsou zapojeny pouze svými vstupy a výstupy. Lze tento automat nahradit tímto, či je dokonce možno bez problémů kterýkoli z nich zaměnit druhým? To jsou otázky, které si klademe.[1]

9.1. Zaměnitelnost a simulovatelnost

V článku 5.2. jsme chování konečného automatu M definovali jako množinu dvojic $\{(y, \beta^*(s, y)): s \in S, y \in I^*\}$. V článku 5.3. jsme pak nesymetrickou funkční podobnost dvou **kompatibilních automatů**, (tj. automatů definovaných na shodných prostorech podnětů a shodných prostorech reakcí) nazvali **simulací**, a v případě simulace automatu M_1 automatem M_2 jsme tento typ funkční podobnosti[2] definovali vztahem

$$\{(y, \beta_1^*(s, y)): s \in S_1, y \in I^*\} \subset \{(x, \beta_2^*(s, x)): s \in S_2, x \in I^*\}.$$

Symetrickou funkční podobnost automatů M_1 a M_2 jsme nazvali **zaměnitelností** a definovali ji vztahem

$$\{(y, \beta_1^*(s, y)): s \in S_1, y \in I^*\} = \{(x, \beta_2^*(s, x)): s \in S_2, x \in I^*\}.$$

Z článku 5.3. je též zřejmé, že zaměnitelnými mohou být jen ekvivalentní automaty.

Předpokladem simulovatelnosti automatu M_1 se stavovým prostorem S_1 kompatibilním automatem M_2 se stavovým prostorem S_2 je splnění podmínky:

$$\forall s_1 \in S_1 \exists s_2 \in S_2 \forall x \in I^*: \beta_1^*(s_1, x) = \beta_2^*(s_2, x).$$

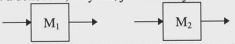
Tato podmínka uvádí do zvláštního vztahu (relace) $s_1 \mathbf{R} s_2$ stavy automatu M_1 se stavy automatu M_2 ($\mathbf{R} \subseteq S_1 \times S_2$). Definiční obor relace **R** - dom(**R**) = S_1, obor hodnot relace **R** - range(**R**) $\subseteq S_2$.[3] V důsledku symetrie, reflexivity a tranzitivnosti dvojic ekvivalentních stavů je relace **R** ekvivalencí mezi množinami S_1 a range(**R**) - jedná se o ekvivalenci příslušných stavových podprostorů. Můžeme psát:

$$\{(y, \beta_1^*(s, y)): s \in S_1, y \in I^*\} = \{(x, \beta_2^*(s, x)): s \in \text{range}(\mathbf{R}), x \in I^*\}.$$

Odtud je okamžitě zřejmé, že simulovatelnost automatu M_1 vyžaduje existenci subautomatu M automatu M_2 ekvivalentního s automatem M_1. Množinou stavů subautomatu M je množina S = range(**R**) $\subseteq S_2$. K zaměnitelnosti obou automatů je navíc třeba aby range(**R**) = S_2.

---

[1] Existence strukturální podobnosti automatů M_1 a M_2 je vázána na existenci VVH $\phi = (\varphi, \xi, \omega^{-1})$. V případě $\phi = (\varphi, \xi, \omega^{-1})$: $M_1 \to M_2$ je automat M_1 (na Obr. 9.1 vlevo) nahraditelný automatem M_2 se vstupním kodérem ξ a výstupním dekodérem ω^{-1} (na Obr. 9.1 vpravo). Naproti tomu u funkční podobnosti automatů se ptáme na možnost jejich záměny bez kodérů a dekodérů, tedy tak, jak naznačuje tento obrázek:



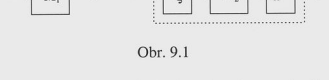
Nutným (ale ne postačujícím) předpokladem funkční podobnosti je proto **kompatibilita** obou automatů ve smyslu shody jak množin vstupních podnětů ($I_1 = I_2 = I$) tak i množin výstupních reakcí ($O_1 = O_2 = O$) obou automatů.

[2] Podobně jako u strukturální podobnosti i zde se jedná o podobnost ke vzoru (vzorem podobnosti je chování automatu M_2).

[3] $\mathbf{R} = \{(s_1, s_2): s_1 \in S_1, s_2 \in S_2, s_1 \text{ a } s_2 \text{ jsou ekvivalentními stavy}\}$.

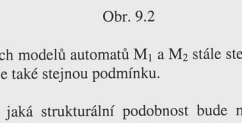
9.2. Vztah mezi strukturální a funkční podobností

Existence VVH $\phi = (\varphi,\xi,\omega^{-1})$: $M_1 \to M_2$ zajišťuje simulovatelnost chování automatu M_1 kompatibilním systémem čárkovaně vyznačeným v Obr. 9.1:



Obr. 9.1

K tomu je nutné a stačí bezprostředně před začátkem simulace uvést do synchronismu vnitřní mechanismy automatů M_1 a M_2 tím, že automat M_2 nastavíme do stavu $\varphi(s)$ ekvivalentního s aktuálním stavem s automatu M_1. Udržení synchronismu obou mechanismů je pak při správné funkci vstupního kodéru[4] zajištěno komutací tohoto okna algebraického diagramu,



Obr. 9.2

které má ve všech variantách modelů automatů M_1 a M_2 stále stejný tvar (viz Obr. 8.3 až Obr. 8.7), a proto v nich generuje také stejnou podmínku.

Ptáme-li se na to, jaká strukturální podobnost bude mít za následek i podobnost funkční, pak dodatečný požadavek kompatibility srovnávaných automatů a nepotřebnosti obou zakreslených převodníků ponechává ze všech možných typů strukturální podobnosti ve hře pouze VVH $\phi = (\varphi,\underline{1},\underline{1})$: $M_1 \to M_2$. V ní funkce φ hraje roli relace $\mathbf{R}$ a platí: $\mathrm{dom}(\varphi) = \mathrm{dom}(\mathbf{R}) = S_1$ a $s_1 \mathbf{R} s_2 \Leftrightarrow \varphi(s_1) = s_2$, neboli $\mathbf{R} = \{(s,\varphi(s)): s \in S_1\}$. Jedná-li se o redukční VVH (v němž φ: $S_1 \to S_2$ je surjektivní, neboli $\mathrm{range}(\varphi) = S_2$), pak příslušnou funkční podobností je zaměnitelnost automatů.[5] Není-li φ surjektivní, pak automat M_2 automat M_1 pouze simuluje.

Opačná otázka, zda z existence funkční podobnosti automatů lze usuzovat i na jejich strukturální podobnost, je zodpovězena negativně. Důvodem je nejednoznačnost syntézy systémů (možnost existence více struktur schopných vykazovat dané chování) zmíněná již v článku 4.4.[6]

---

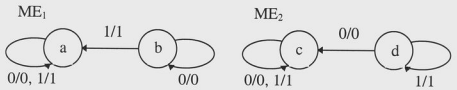
[4] Správnou funkcí zde rozumíme požadavek, aby vstupní převodník ξ: $I_1 \to I_2$ také transformoval všechny podněty nepatřící do množiny I_1 na podněty nepatřící do množiny I_2. V opačném případě by na nějaký podnět (nepatřící do I_1) automat M_1 nereagoval, ale čárkovaný systém s automatem M_2 ano. Historie obou automatů by pak měla nestejnou délku, čímž by mohlo dojít k výpadku jejich vnitřních mechanismů ze synchronismu a následnému odlišnému chování. K tomu nemůže dojít u kompatibilních automatů ($I_1 = I_2$) s převodníkem $\xi = \underline{1}$. Z hlediska realizace nejlepším takovým převodníkem je žádný převodník (tj. stejný podnět přichází na oba vstupy).

[5] Pokud redukce není izomorfismem (φ není bijektivní), pak z nesymetrické strukturální podobnosti zde plyne symetrická funkční podobnost.

[6] Kladná odpověď na předchozí otázku byla naopak umožněna jednoznačností analýzy systému (chování ze struktury jednoznačně vyplývá).

Cvičení k lekci 9.

Příklad 9.1: Dány kompatibilní automaty $ME_1 = (S_1, I, O, \delta_1, \beta_1)$ a $ME_2 = (S_2, I, O, \delta_2, \beta_2)$:



Obr. 9.3

Oba automaty v každém stavu na výstup vrací vstupní podnět, což je činí vzájemně zastupitelnými. Přesto mezi nimi žádný redukční VVH neexistuje. Žádná z následujících čtyř trojic zobrazení

$\phi_i = (\varphi_i, \underline{1}, \underline{1})$: $ME_1 \to ME_2$, $\phi'_i = (\varphi'_i, \underline{1}, \underline{1})$: $ME_2 \to ME_1$, i = 1,2, kde

φ_1:	a	b
	c	d

φ_2:	a	b
	d	c

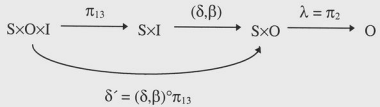
φ'_1:	c	d
	a	b

φ'_2:	c	d
	b	a

totiž VVH není, neboť

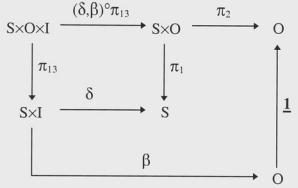
$$\varphi_1(\delta_1(b,0)) \neq \delta_2(\varphi_1(b),0), \varphi_2(\delta_1(a,0)) \neq \delta_2(\varphi_2(a),0),$$
$$\varphi'_1(\delta_2(d,0)) \neq \delta_1(\varphi'_1(d),0), \varphi'_2(\delta_2(c,1)) \neq \delta_1(\varphi'_2(c),1).$$

Příklad 9.2: V článku 5.4. byl uveden obecný postup konstrukce ekvivalentního Mooreova automatu 2. druhu (MO2) k danému Mealyho automatu (ME). Ukázali jsme tam, že automaty ME = (S, I, O, δ, β) a MO2 = (S×O, I, O, $(\delta,\beta)\circ\pi_{13}$, π_2), kde zobrazení π_{13}: S×O×I → S×I a π_2: S×O → O jsou projekce (viz Obr. 9.4), jsou ekvivalentní a tudíž zaměnitelné.



Obr. 9.4

Zde ukážeme, že $\phi = (\pi_1, \underline{1}, \underline{1})$: MO2 → ME, kde MO2 a ME jsou zmíněné automaty a zobrazení π_1: S×O → S je projekce, je redukční VVH. K tomu stačí dokázat komutaci algebraického diagramu uvedeného na Obr. 9.5



Obr. 9.5

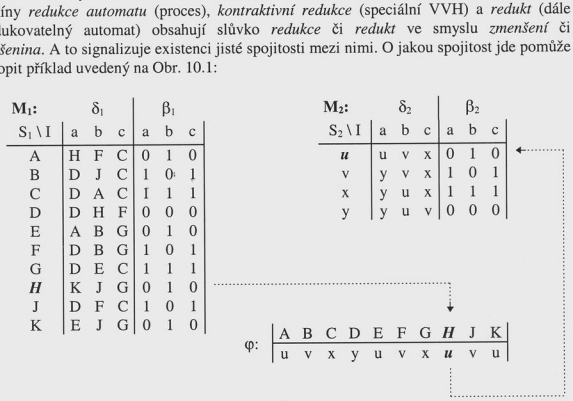
Komutace malého okna diagramu vyžaduje aby $\delta^{\circ}\pi_{13} = \pi_1^{\circ}(\delta,\beta)^{\circ}\pi_{13}$. Komutace velkého okna diagramu vyžaduje aby $\beta^{\circ}\pi_{13} = \pi_2^{\circ}(\delta,\beta)^{\circ}\pi_{13}$. Z definice příslušných projekcí a vektorové funkce bezprostředně plyne $\delta = \pi_1^{\circ}(\delta,\beta)$ a $\beta = \pi_2^{\circ}(\delta,\beta)$. Tím je komutace diagramu zajištěna.

10. Minimalizace automatu

V předchozí lekci jsme prozkoumali vztah mezi strukturální a funkční podobností kompatibilních automatů a došli jsme k závěru, že existence redukčního VVH $\phi = (\varphi, \underline{1}, \underline{1})$: $M_1 \rightarrow M_2$ má za následek funkční ekvivalenci nebo-li zaměnitelnost automatů M_1 a M_2. Zvlášť zajímavý je případ, kdy redukční VVH není izomorfismem, tj. když funkce φ je kontraktivní (tzv. **kontraktivní redukce**). Pak automat M_2 má méně stavů než automat M_1, a proto může být ekonomičtější či jinak výhodnější realizovat M_2 místo M_1. Vzniká tak otázka: Jaký nejmenší počet stavů vůbec může mít automat M_2, strukturálně podobný a funkčně zaměnitelný s automatem M_1? Touto otázkou se ptáme na (z hlediska počtu stavů) minimální automat nebo-li na tzv. **redukt**.

10.1. Vlastnosti kontraktivní redukce

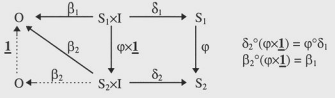
V souladu s výše uvedeným nazveme reduktem každý automat, k němuž neexistuje automat strukturálně podobný, funkčně zaměnitelný a s menším počtem stavů. Proces, kterým z automatu, který reduktem není, dostaneme jeho redukt, nazýváme **redukcí automatu**. Termíny *redukce automatu* (proces), *kontraktivní redukce* (speciální VVH) a *redukt* (dále neredukovatelný automat) obsahují slůvko *redukce* či *redukt* ve smyslu *zmenšení* či *zmenšenina*. A to signalizuje existenci jisté spojitosti mezi nimi. O jakou spojitost jde pomůže pochopit příklad uvedený na Obr. 10.1:



Obr. 10.1

V něm M_1 a M_2 jsou kompatibilní Mealyho automaty zadané tabulkami svých přechodových a výstupních funkcí. V důsledku kompatibility na vstupu mají obě tabulky stejný počet sloupců (tj. v obou tabulkách má řádkový vektor týž rozměr). Kontraktivní a surjektivní funkce φ: $S_1 \rightarrow S_2$ zadaná třetí tabulkou je složkou redukčního VVH $\phi = (\varphi, \underline{1}, \underline{1})$: $M_1 \rightarrow M_2$ definovaného následujícím algebraickým diagramem:[1]

[1] Vyznačená úprava algebraického diagramu je důsledkem platnosti vztahu $\underline{1} \circ \beta_2 = \beta_2$.



Obr. 10.2

Dvojice podmínek $\delta_2\circ(\varphi\times\underline{1}) = \varphi\circ\delta_1$ a $\beta_2\circ(\varphi\times\underline{1}) = \beta_1$ definuje pravidla, podle nichž se řádkové vektory tabulky automatu M_1 na Obr. 10.1 transformují na řádkové vektory tabulky automatu M_2.[2] Transformace se provádí tak, že všechny stavy nacházející se v levé části výchozího vektoru (tj. aktuální stav a jeho následníci - tzv. *subvektor stavů*) se překódují funkcí φ na stavy cílového vektoru, a hodnoty výstupních reakcí (pravá část řádkového vektoru - tzv. *subvektor reakcí*) se ponechají beze změny. Např. osmý řádek první tabulky (pro aktuální stav ***H*** automatu M_1) je takto transformován do prvního řádku tabulky druhé (pro aktuální stav ***u*** automatu M_2). V Obr. 10.1 je toto znázorněno čárkovanými šipkami a jejich průchod tabulkou funkce φ znázorňuje přejmenovávání aktuálního stavu automatu M_1.

Snadno lze nahlédnout, že do prvního řádku druhé tabulky se takto transformuje nejen osmý, ale i první, pátý a desátý řádek tabulky první - tj. všechny řádky, jejichž aktuálním stavům je funkcí φ přiřazena funkční hodnota *u*. A zcela analogicky se podle týchž pravidel do druhého, resp. třetího, či čtvrtého řádku druhé tabulky transformují ty řádky první tabulky, jejichž aktuálním stavům je funkcí φ přiřazena funkční hodnota *v*, resp. *x*, či *y*.

V žargonu moderní algebry to lze vyjádřit tak, že funkce $\varphi: S_1 \rightarrow S_2$, definuje na svém definičním oboru binární relaci $Ker(\varphi) \subseteq S_1\times S_1$, zvanou **jádro redukčního VVH**, takto:

$$\forall s,s' \in S_1: ((s,s') \in Ker(\varphi)) \Leftrightarrow (\varphi(s) = \varphi(s')).^3$$

Tato relace je reflexivní (φ(s) = φ(s)), symetrická (φ(s)=φ(s´) ⇒ φ(s´)=φ(s)) a tranzitivní ((φ(s)=φ(s´) & φ(s´)=φ(s´´)) ⇒ φ(s)=φ(s´´)). Jedná se tudíž o ekvivalenci, a ta vytváří na množině S_1 rozklad

$$S_1/Ker(\varphi) = \{\varphi^*(a): a\in S_2\}, \text{ kde } \varphi^*(a) = \{s\in S_1: \varphi(s) = a\}.$$

V našem příkladu na Obr. 10.1 $S_1/Ker(\varphi) = \{\{A,E,H,K\}, \{B,F,J\}, \{C,G\}, \{D\}\}$, neboť $\varphi^*(u) = \{A,E,H,K\}$, $\varphi^*(v) = \{B,F,J\}$, $\varphi^*(x) = \{C,G\}$, $\varphi^*(y) = \{D\}$. Na Obr. 10.3 jsme množinu S_1 v tabulce funkce φ uspořádali podle hodnot φ(s), s∈S_1 (spodní tabulka), čímž se tento rozklad stal očividným.

[2] Jedná se o modifikaci poněkud obecnějších pravidel podrobně popsaných už v poznámce pod čarou 3 v článku 7.1. Tabulka automatu M_1 tam definovala vektorovou funkci [δ,β], tabulka automatu M_2 se tam nacházela ve formě Karnaughovy mapy.

[3] Jádro redukčního VVH uvádí do korespondence prvky z definičního oboru funkce φ mající stejnou funkční hodnotu.

M_1:	δ_1			β_1		
$S_1 \setminus I$	a	b	c	a	b	c
A	H	F	C	0	1	0
E	A	B	G	0	1	0
H	K	J	G	0	1	0
K	E	J	G	0	1	0
B	D	J	C	1	0	1
F	D	B	G	1	0	1
J	D	F	C	1	0	1
C	D	A	C	1	1	1
G	D	E	C	1	1	1
D	D	H	F	0	0	0

M_1:	δ_1			β_1		
$S_1 \setminus I$	a	b	c	a	b	c
A	$\varphi^*(u)$	$\varphi^*(v)$	$\varphi^*(x)$	0	1	0
E	$\varphi^*(u)$	$\varphi^*(v)$	$\varphi^*(x)$	0	1	0
H	$\varphi^*(u)$	$\varphi^*(v)$	$\varphi^*(x)$	0	1	0
K	$\varphi^*(u)$	$\varphi^*(v)$	$\varphi^*(x)$	0	1	0
B	$\varphi^*(y)$	$\varphi^*(v)$	$\varphi^*(x)$	1	0	1
F	$\varphi^*(y)$	$\varphi^*(v)$	$\varphi^*(x)$	1	0	1
J	$\varphi^*(y)$	$\varphi^*(v)$	$\varphi^*(x)$	1	0	1
C	$\varphi^*(y)$	$\varphi^*(u)$	$\varphi^*(x)$	1	1	1
G	$\varphi^*(y)$	$\varphi^*(u)$	$\varphi^*(x)$	1	1	1
D	$\varphi^*(y)$	$\varphi^*(u)$	$\varphi^*(v)$	0	0	0

φ:	A	E	H	K	B	F	J	C	G	D
	u	u	u	u	v	v	v	x	x	y

Obr. 10.3

Stejným uspořádáním množiny S_1 i v tabulce přechodové a výstupní funkce automatu M_1 jsme dostali i levou tabulku na Obr. 10.3. Z ní je okamžitě zřejmé, že ve všech stavech z téže třídy rozkladu $S_1/Ker(\varphi)$ automat M_1 reaguje shodně.[4] Této vlastnosti rozkladu říkáme **konzistence** rozkladu. Konzistence rozkladu $S_1/Ker(\varphi)$ je důsledkem komutace levého okna diagramu na Obr. 10.2, tj. důsledkem platnosti vztahu $\beta_2 \circ (\varphi \times \underline{1}) = \beta_1$.

Pravá tabulka na Obr. 10.3 vznikla záměnou následných stavů levé tabulky za označení tříd rozkladu $S_1/Ker(\varphi)$, v nichž se následné stavy nachází. Z ní je zřejmé, že všechny stavy z jedné třídy rozkladu $S_1/Ker(\varphi)$ na daný podnět přechází opět do nějaké jedné třídy rozkladu $S_1/Ker(\varphi)$. Tuto vlastnost rozkladu nazýváme **substituční vlastností**, zkráceně též **SV vlastností** rozkladu. SV vlastnost rozkladu $S_1/Ker(\varphi)$ je důsledkem komutace pravého okna diagramu na Obr. 10.2, tj. důsledkem platnosti vztahu $\delta_2 \circ (\varphi \times \underline{1}) = \varphi \circ \delta_1$.

Závěry vyvozené z analýzy uvedeného příkladu nás opravňují vyslovit tezi, že *rozklad, indukovaný jádrem redukčního VVH je SV a konzistentní.* Podaří-li se tuto tezi obecně dokázat, stane se teorémem. Za tím účelem je třeba nejprve převést výše uvedené verbální charakteristiky SV vlastnosti a konzistence rozkladu do podoby formálních definic:

- substituční vlastnost rozkladu $S_1/Ker(\varphi)$ můžeme definovat formulí

$$\forall a,b \in S_1: (\varphi(a) = \varphi(b)) \Rightarrow \forall i \in I: (\varphi(\delta_1(a,i)) = \varphi(\delta_1(b,i)));$$

- konzistenci rozkladu $S_1/Ker(\varphi)$ můžeme definovat formulí

$$\forall a,b \in S_1: (\varphi(a) = \varphi(b)) \Rightarrow \forall i \in I: (\beta_1(a,i) = \beta_1(b,i)).$$

[4] Stavy ze třídy {A,E,H,K} mají subvektor reakcí (0,1,0), stavy ze třídy {B,F,J} mají subvektor reakcí (1,0,1), a stavy ze třídy {C,G} mají subvektor reakcí (1,1,1).

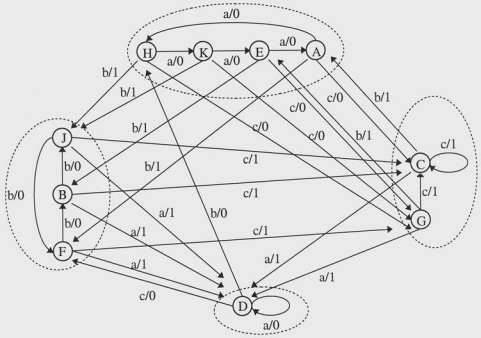
Při důkazu tvrzení „*rozklad* $S_1/Ker(\varphi)$ *je SV a konzistentní*“ vyjdeme z podmínek komutace algebraického diagramu na Obr. 10.2:

1. $\delta_2(\varphi(a),i) = \varphi(\delta_1(a,i))$,
2. $\beta_2(\varphi(a),i) = \beta_1(a,i)$.

Důkaz (horní index u symbolu implikace odkazuje na užitou podmínku komutace:

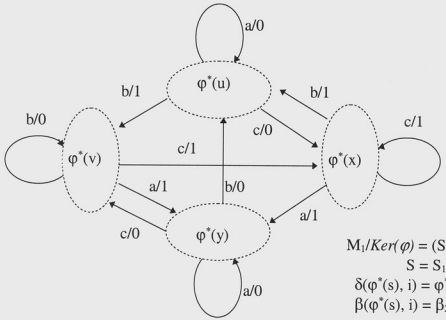
- $(\varphi(a) = \varphi(b)) \Rightarrow (\delta_2(\varphi(a),i) = \delta_2(\varphi(b),i)) \Rightarrow^1 (\varphi(\delta_1(a,i)) = \varphi(\delta_1(b,i)))$ ..... SV vlastnost,
- $(\varphi(a) = \varphi(b)) \Rightarrow (\beta_2(\varphi(a),i) = \beta_2(\varphi(b),i)) \Rightarrow^2 (\beta_1(a,i) = \beta_1(b,i))$ ..... konzistence.

Provedením důkazu jsme se ujistili, že to, co vidíme v pravé tabulce na Obr. 10.3 není dílem náhody, ale důsledkem robustní zákonitosti. Grafické vyjádření této zákonitosti vidíme na Obr. 10.4:



Obr. 10.4

V něm třídy rozkladu $S_1/Ker(\varphi)$ vytváří jakési „makrostavy“ znázorněné čárkovanými ovály, mezi nimiž automat M_1 při své funkci přechází. Složky „makrostavů“ (konkrétní stavy z množiny S_1) zde nejsou podstatné, chovají se stejně, a proto lze od nich odhlédnout. V důsledku toho je automat M_1 z Obr. 10.1 ekvivalentní s automatem $M_1/Ker(\varphi) = (S, I, O, \delta, \beta)$, v němž $S = S_1/Ker(\varphi)$ a jehož grafická i tabulková podoba přechodové a výstupní funkce je uvedena na Obr. 10.5. Je také zřejmé, že automat $M_1/Ker(\varphi)$ z Obr. 10.5 je izomorfní s automatem M_2 v Obr. 10.1. Jedná se o izomorfizmus $\phi = (\varphi', \underline{1}, \underline{1})$: $M_1/Ker(\varphi) \rightarrow M_2$ v němž funkce φ': $S_1/Ker(\varphi) \rightarrow S_2$ je ve tvaru $\varphi'(\varphi^*(s)) = s$, $s \in S_2$.



$M_1/Ker(\varphi) = (S, I, O, \delta, \beta)$,
$S = S_1/Ker(\varphi)$,
$\delta(\varphi^*(s), i) = \varphi^*(\delta_2(s, i))$,
$\beta(\varphi^*(s), i) = \beta_2(s, i), \quad s \in S_2$.

	δ			β		
S \ I	a	b	c	a	b	c
φ*(u)	φ*(u)	φ*(v)	φ*(x)	0	1	0
φ*(v)	φ*(y)	φ*(v)	φ*(x)	1	0	1
φ*(x)	φ*(y)	φ*(u)	φ*(x)	1	1	1
φ*(y)	φ*(y)	φ*(u)	φ*(v)	0	0	0

Obr. 10.5

10.2. Podílový automat

V článku 10.1 jsme dokázali, že s existencí redukčního VVH $\phi = (\varphi, \underline{1}, \underline{1})$: $M_1 \rightarrow M_2$ je úzce spjata i existence SV a konzistentního rozkladu stavového prostoru automatu M_1. Zobrazení φ: $S_1 \rightarrow S_2$, které je složkou redukčního VVH, umožňuje tento rozklad (který jsme označili jako $S_1/Ker(\varphi)$) snadno a rychle nalézt. Viděli jsme, že informace obsažená v tomto SV a konzistentním rozkladu (zviditelněná tabulkami na Obr. 10.3 či grafem na Obr. 10.4) postačuje k sestrojení automatu $M_1/Ker(\varphi)$, jehož stavy jsou jednotlivé třídy rozkladu $S_1/Ker(\varphi)$. Automat $M_1/Ker(\varphi)$ je kompatibilní a izomorfní s automatem M_2, a proto je funkčně zaměnitelný i s automatem M_1. Postup jeho konstrukce je v článku 10.1 podrobně popsán. Výsledek ilustračního příkladu v grafické i tabulkové formě vidíme na Obr. 10.5.

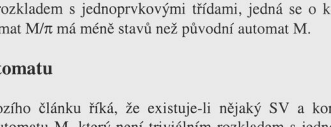
Pozorný čtenář si jistě všiml, že dojít od automatu M_1 z Obr. 10.1 k automatu na Obr. 10.5 jsme mohli i bez znalosti automatu M_2 a redukčního VVH $\phi = (\varphi, \underline{1}, \underline{1})$: $M_1 \rightarrow M_2$. Tato znalost posloužila jen k nalezení SV a konzistentního rozkladu $S_1/Ker(\varphi)$, pak už k ničemu potřebná nebyla. SV a konzistentní rozklady stavových prostorů automatů je však možno

hledat i jinak.[5] Pokud nějaký takový SV a konzistentní rozklad získáme, pak na třídách tohoto rozkladu můžeme postupem uvedeným v článku 10.1 sestrojit automat s menším počtem stavů, zaměnitelný s původním automatem. Tento automat nazýváme **podílovým automatem** původního automatu.

Obecně lze podílový automat sestrojit na každém SV a konzistentním rozkladu stavového prostoru libovolného automatu M = (S, I, O, δ, β). Je-li takovým rozkladem rozklad $\pi = \{\langle s\rangle_\pi: s\in S\}$, v němž symbol $\langle s\rangle_\pi$ označuje třídu rozkladu π obsahující stav s, je SV vlastnost rozkladu π definována podmínkou $\forall a,b\in S: ((\langle a\rangle_\pi = \langle b\rangle_\pi) \Rightarrow \forall i\in I: (\langle\delta(a, i)\rangle_\pi = \langle\delta(b, i)\rangle_\pi)$ a konzistence rozkladu π podmínkou $\forall a,b\in S: ((\langle a\rangle_\pi = \langle b\rangle_\pi) \Rightarrow \forall i\in I: (\beta(a, i) = \beta(b, i)))$. K němu příslušným **podílovým automatem** je pak automat $M/\pi = (\pi, I, O, \delta_\pi, \beta_\pi)$, v němž pro δ_π: $\pi\times I \to \pi$ platí $\delta_\pi(\langle s\rangle_\pi, i) = \langle\delta(s, i)\rangle_\pi$ a pro β_π: $\pi\times I \to O$ platí $\beta_\pi(\langle s\rangle_\pi, i) = \beta(s, i)$. Pro funkci φ: $S \to \pi$ definovanou předpisem $\varphi(s) = \langle s\rangle_\pi$ v tom případě platí:

- $\delta_\pi\circ(\varphi\times\underline{1}) \approx \delta_\pi(\varphi(s), i) = \delta_\pi(\langle s\rangle_\pi, i) = \langle\delta(s, i)\rangle_\pi = \varphi(\delta(s, i)) \approx \varphi\circ\delta$;
- $\beta_\pi\circ(\varphi\times\underline{1}) \approx \beta_\pi(\varphi(s), i) = \beta_\pi(\langle s\rangle_\pi, i) = \beta(s, i) \approx \beta$.

Proto trojice $\phi = (\varphi, \underline{1}, \underline{1})$: $M \to M/\pi$ s takto definovanou funkcí φ je redukčním VVH (viz diagram na Obr. 10.6 spolu s podmínkami komutace jeho oken) a automat M je s automatem M/π funkčně zaměnitelný.



Obr. 10.6

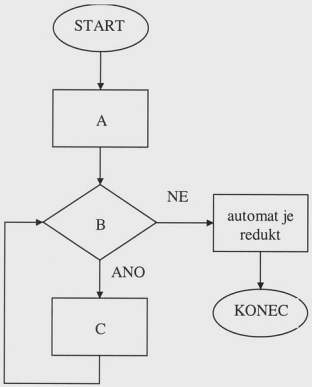
Není-li π triviálním rozkladem s jednoprvkovými třídami, jedná se o kontraktivní redukční VVH a podílový automat M/π má méně stavů než původní automat M.

10.3. Redukce automatu

Závěr předchozího článku říká, že existuje-li nějaký SV a konzistentní rozklad π stavového prostoru automatu M, který není triviálním rozkladem s jednoprvkovými třídami, není automat M reduktem, neboť jeho podílový automat M/π je automatu M strukturálně podobný, je s ním funkčně zaměnitelný a má méně stavů než automat M. Sestrojení automatu M/π pak může být prvním krokem redukce automatu M. Nemá-li stavový prostor automatu M/π jiný SV a konzistentní rozklad kromě triviálního rozkladu s jednoprvkovými třídami, je tento krok i krokem posledním a automat M/π je redukt. V opačném případě, tj. existuje-li SV a konzistentní rozklad σ stavového prostoru automatu M/π, který není triviálním rozkladem s jednoprvkovými třídami, sestrojíme na něm podílový automat $(M/\pi)/\sigma$ a opět zkoumáme, je-li už reduktem či ne.

[5] Například tak, že se zaměříme jen na netriviální konzistentní rozklady (do stejné třídy zde lze zařadit jen stavy shodující se v subvektoru reakcí), u nichž pak zjišťujeme existenci SV vlastnosti. Je zřejmé, že stavový prostor automatu M_2 z Obr. 10.1 nemá netriviální konzistentní rozklad (tím méně netriviální SV a konzistentní rozklad), zatímco stavový prostor automatu M_1 má 149 netriviálních konzistentních rozkladů, z nichž jen jediný (již zmíněný rozklad $S_1/Ker(\varphi)$) má i SV vlastnost. V lekci 11 se seznámíme s podstatně efektivnějšími postupy získávání SV a konzistentních rozkladů i výsledných reduktů.

Uvedený postup je znázorněn vývojovým diagramem na Obr. 10.7:



Legenda:

A: Zadat redukovaný automat.

B: Existuje netriviální SV a konzistentní rozklad stavového prostoru zkoumaného automatu?

C: Sestrojte na tomto SV a konzistentním rozkladu podílový automat.

Obr. 10.7

Vzhledem k tomu, že počet stavů zadaného automatu je konečný, a při každém průchodu procesu redukce blokem B se počet stavů aktuálního automatu sníží nejméně o jeden, musí redukce zadaného automatu po konečném počtu jejích kroků skončit a výsledný automat je hledaným reduktem.

Cvičení k lekci 10.

Příklad 10.1: Na Obr. 10.8 je tabulka přechodové a výstupní funkce automatu M:

M:	δ		β	
S \ I	a	b	a	b
A	B	C	0	1
B	A	C	0	1
C	D	E	1	0
D	C	B	0	1
E	A	F	1	0
F	A	E	1	0

Obr. 10.8

Automat M není reduktem, protože rozklady $\pi = \{\{A\},\{B\},\{C\},\{D\},\{E,F\}\}$ a $\sigma = \{\{A,B\},\{C\},\{D\},\{E\},\{F\}\}$ jsou SV a konzistentní. Podílové automaty M/π a M/σ, definované tabulkami na Obr. 10.9,

M/π:	δ_π		β_π	
π \ I	a	b	a	b
{A}	{B}	{C}	0	1
{B}	{A}	{C}	0	1
{C}	{D}	{E,F}	1	0
{D}	{C}	{B}	0	1
{E,F}	{A}	{E,F}	1	0

M/σ:	δ_σ		β_σ	
σ \ I	a	b	a	b
{A,B}	{A,B}	{C}	0	1
{C}	{D}	{E}	1	0
{D}	{C}	{A,B}	0	1
{E}	{A,B}	{F}	1	0
{F}	{A,B}	{E}	1	0

Obr. 10.9

také nejsou redukty, protože rozklady jejich stavových prostorů
$\rho = \{\{\{A\},\{B\}\},\{\{C\},\{D\}\},\{\{E,F\}\}\}$ a
$\tau = \{\{\{A,B\}\},\{\{C\}\},\{\{D\}\},\{\{E\},\{F\}\}\}$ jsou SV a konzistentní. Oba podílové automaty $(M/\pi)/\rho$ a $(M/\sigma)/\tau$ na nich sestrojené (viz Obr. 10.10) jsou izomorfní a už redukty jsou.

(M/π)/ρ:	$\delta_{\pi\rho}$		$\beta_{\pi\rho}$	
ρ \ I	a	b	a	b
{{A},{B}}	{{A},{B}}	{{C}}	0	1
{{C}}	{{D}}	{{E,F}}	1	0
{{D}}	{{C}}	{{A},{B}}	0	1
{{E,F}}	{{A},{B}}	{{E,F}}	1	0

(M/σ)/τ:	$\delta_{\sigma\tau}$		$\beta_{\sigma\tau}$	
τ \ I	a	b	a	b
{{A,B}}	{{A,B}}	{{C}}	0	1
{{C}}	{{D}}	{{E},{F}}	1	0
{{D}}	{{C}}	{{A,B}}	0	1
{{E},{F}}	{{A,B}}	{{E},{F}}	1	0

Obr. 10.10

11. Algoritmy redukce automatu

V Příkladu 10.1 jsme v první fázi redukce automatu M na Obr. 10.8 hledali SV a konzistentní rozklady jeho množiny stavů S = {A, B, C, D, E, F}.[1] Nalezli jsme dva: $\pi = \{\langle s\rangle_\pi: s\in S\} = \{\{A\}, \{B\}, \{C\}, \{D\}, \{E,F\}\}$ a $\sigma = \{\langle s\rangle_\sigma: s\in S\} = \{\{A,B\}, \{C\}, \{D\}, \{E\}, \{F\}\}$,[2] a sestrojili na nich podílové automaty M/π a M/σ. V další fázi redukce jsme hledali SV a konzistentní rozklady množin π a σ. A zde jsme našli ke každé jen jeden: $\rho = \{\langle x\rangle_\rho: x\in\pi\} = \{\{\{A\},\{B\}\}, \{\{C\}\}, \{\{D\}\}, \{\{E,F\}\}\}$, a $\tau = \{\langle x\rangle_\tau: x\in\sigma\} = \{\{\{A,B\}\}, \{\{C\}\}, \{\{D\}\}, \{\{E\},\{F\}\}\}$. Je zřejmé, že ρ ≠ τ, protože $\{\{A\},\{B\}\} \neq \{\{A,B\}\}$ a $\{\{E,F\}\} \neq \{\{E\},\{F\}\}$.

Avšak odhlédneme-li od toho, jak byly rozklady ρ a τ získány, a nebudeme-li jejich třídy definovat prvky naposled rozkládané množiny, ale prvky původní množiny S, bude $\rho = \{\langle s\rangle_\rho: s\in S\} = \{\{A,B\}, \{C\}, \{D\}, \{E,F\}\} = \{\langle s\rangle_\tau: s\in S\} = \tau$. Nesbližujeme mezi nimi žádných rozdílů a ztotožní se s rozkladem η (viz pozn. pod čarou 2). Stejně tak po odhlédnutí od obsahu jednotlivých tříd výsledných rozkladů neshledáme žádný rozdíl mezi redukty (M/π)/ρ, (M/σ)/τ a M/η. Toto sjednocení pohledu umožní porovnat a částečně uspořádat různé rozklady téže výchozí množiny, a to bez ohledu na způsob a postup, jakým byly získány.

11.1. Částečné uspořádání rozkladů a jeho Hasseovský diagram

Porovnáním tříd rozkladů $\pi = \{\langle s\rangle_\pi: s\in S\}$ a $\eta = \{\langle s\rangle_\eta: s\in S\}$ zjistíme, že pro ∀s∈S platí: $\langle s\rangle_\pi \subseteq \langle s\rangle_\eta$. Vyjádřeno slovy: každá třída rozkladu π je podmnožinou nějaké třídy rozkladu η. Uvedený vztah inkluze mezi třídami generuje vztah π ≤ η mezi rozklady, který interpretujeme jako **srovnatelnost** rozkladů π a η (jde o srovnatelné rozklady). Směr symbolu srovnatelnosti navíc říká, že rozklad π není „hrubší" (tj. nemá méně tříd) než rozklad η.[3] Je-li alespoň jedna inkluze mezi třídami ostrá, tj. $\exists s\in S: \langle s\rangle_\pi \subset \langle s\rangle_\eta$ (jako tomu je v případě $\{A\} \subset \{A,B\}$), má rozklad η méně tříd než rozklad π („η je hrubší než π" nebo také „π je jemnější než η"), což zapisujeme π < η. Relací < je možno různé rozklady téže množiny S uspořádat. Avšak vzhledem k možné existenci nesrovnatelných rozkladů (nesrovnatelnými rozklady jsou π a σ, protože nesplňují podmínku $\forall s\in S: \langle s\rangle_\pi \subseteq \langle s\rangle_\sigma$ ani podmínku $\forall s\in S: \langle s\rangle_\sigma \subseteq \langle s\rangle_\pi$), nemusí jít o uspořádání úplné, ale jen o uspořádání částečné.

Strukturu částečného uspořádání různých rozkladů zadané množiny S graficky znázorňuje tzv. Hasseovský diagram. Setkali jsme se s ním v jeho klasické podobě již na Obr. 4.18 a na Obr. 4.20. Rozklady jsou v něm uspořádány ve vrstvách; každou vrstvu tvoří vzájemně nesrovnatelné rozklady, spojnice (úsečky) vedené mezi vrstvami spojují sousední srovnatelné rozklady (tj. jen takové, mezi nimiž žádný jiný srovnatelný rozklad neleží). Přitom jemnější rozklad vždy leží v nižší vrstvě.

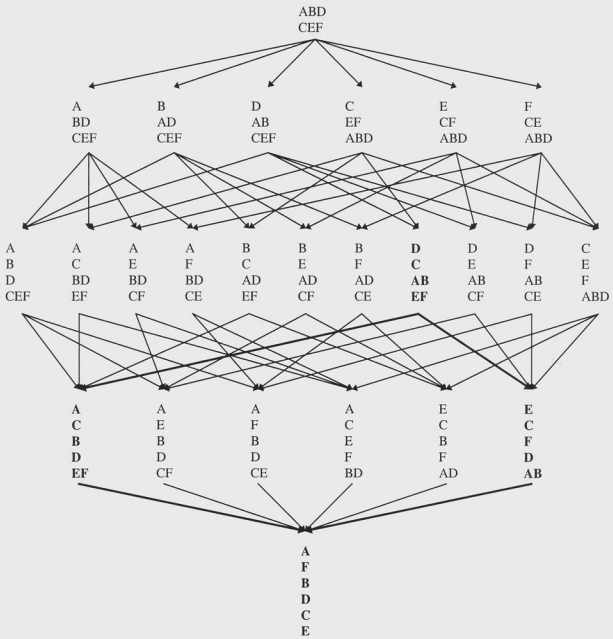
---

[1] Rozkládáním množiny se rozumí seskupování všech jejích prvků do vzájemně disjunktních podmnožin (tříd). Výsledkem je rozklad - nová množina, jejímiž prvky jsou všechny takto vytvořené třídy (vzájemně disjunktní podmnožiny prvků rozkládané množiny).

[2] Existuje ještě třetí SV a konzistentní rozklad $\eta = \{\langle s\rangle_\eta: s\in S\} = \{\{A,B\}, \{C\}, \{D\}, \{E,F\}\}$ množiny S, který jsme v řešení Příkladu 10.1 nezvažovali.

[3] Stejně „hrubé" by oba rozklady byly jen v případě, kdyby pro ∀s∈S platilo $\langle s\rangle_\pi = \langle s\rangle_\eta$. Pak by ale π = η. Je zřejmé, že takto zavedená binární relace ≤ je reflexivní (a ≤ a), antisymetrická (a ≤ b, b ≤ a ⇒ a = b) a tranzitivní (a ≤ b, b ≤ c ⇒ a ≤ c).

Za účelem zvýšení srozumitelnosti dalšího výkladu nahradíme v klasické verzi Hasseovského diagramu neorientované spojnice (úsečky) orientovanými spojnicemi (šipkami), vedoucími od hrubšího k jemnějšímu rozkladu. Příklad této modifikace Hasseovského diagramu vidíme na Obr. 11.1: Jedná se o znázornění struktury částečného uspořádání všech konzistentních rozkladů stavového prostoru S automatu M definovaného tabulkou na Obr. 10.8.



Obr. 11.1

Dvojice srovnatelných rozkladů je v diagramu propojena orientovanou cestou (posloupností sestávající ze shodně orientovaných spojnic a případných mezilehlých rozkladů). Naopak nesrovnatelné rozklady poznáme podle toho, že mezi nimi žádná orientovaná cesta nevede.

Hasseovský diagram všech konzistentních rozkladů lze nejsnáze sestrojit systematickým postupem „shora dolů“. Postup využívá toho, že jakýmkoli dalším rozkladem třídy konzistentního rozkladu vznikne opět konzistentní rozklad. Výchozím rozkladem je nejvýše položený rozklad sdružující do stejné třídy všechny stavy se shodným subvektorem reakcí (viz poznámka pod čarou 4 v lekci 10, vztahující se k automatu na Obr. 10.3). Rozkladem jedné z víceprvkových tříd výchozího rozkladu do dvou podtříd dostaneme s ním sousedící jemnější rozklad ve vrstvě bezprostředně nižší, a mezi oba rozklady zakreslíme orientovanou spojnici.

Opakováním téhož pro každou víceprvkovou třídu výchozího rozkladu, a každou možnost jejího dělení na dvě podtřídy, získáme kompletní obsazení bezprostředně nižší vrstvy. V takto vytvořené vrstvě považujeme každý rozklad za nový výchozí rozklad. Aplikací výše uvedeného postupu na všechny nové výchozí rozklady vytvoříme další vrstvu ještě jemnějších rozkladů. Postupným generováním dalších vrstev dle téhož „receptu“ dospějeme k nejnižší vrstvě s jediným triviálním rozkladem s jednoprvkovými třídami. Hasseovský diagram všech konzistentních rozkladů je hotov.

11.2. Svaz SV a konzistentních rozkladů stavového prostoru automatu

V závěru poznámky pod čarou 7 jsme v lekci 4 zavedli nad systémem rozkladů stavového prostoru automatu dvojici binárních operací $*$ (součin rozkladů) a $+$ (součet rozkladů). Operaci $*$ jsme definovali předpisem $\sigma*\tau = \{\langle s\rangle_\sigma \cap \langle s\rangle_\tau : s \in S\}$, operaci $+$ jsme definovali vývojovým diagramem jejího algoritmu uvedeným na Obr. 4.5. Obě tyto operace zachovávají konzistenci (jsou-li konzistentní oba operandy, je konzistentní i výsledek),[4] a proto každá množina **všech** konzistentních rozkladů je vůči těmto operacím uzavřená (výsledky operací nad jejími libovolnými dvěma prvky v ní leží také).

Z Hasseovského diagramu množiny rozkladů uzavřené vůči těmto operacím je možno jejich výsledky nad libovolnými operandy snadno a rychle zjistit: Jsou-li operandy srovnatelnými rozklady, je výsledkem operace $*$ jemnější z nich, a výsledkem operace $+$ je hrubší z nich. V případě nesrovnatelných operandů bude výsledek operace $*$ jemnějším rozkladem obou operandů; bude jím rozklad, v němž se některá z orientovaných cest, vycházejících z jednoho operandu, potká s některou z orientovaných cest, vycházejících z druhého operandu. Naproti tomu výsledek operace $+$ bude hrubším srovnatelným rozkladem k oběma operandům; bude jím rozklad, z něhož vychází dvojice orientovaných cest, z níž jedna cesta vede k jednomu a druhá cesta vede ke druhému operandu.

V moderní algebře se částečně uspořádaná množina prvků daných vlastností, uzavřená vzhledem k dvojici binárních, idempotentních, komutativních a asociativních operací (blíže viz poznámka pod čarou 7 v lekci 4), souvisejících s relací uspořádání tak jak bylo výše popsáno (viz souvislost výsledků operací s cestami v Hasseovském diagramu), nazývá **svazem**. V námi diskutovaném příkladu svazu na Obr. 11.1 se jedná o **svaz konzistentních rozkladů** stavového prostoru S automatu M z Obr. 10.8, formálně definovaný čtveřicí $KM = (K(M), \leq, *, +)$, v níž $K(M)$ je množina všech konzistentních rozkladů indukovaná automatem

[4] Mají-li třídy $\langle x\rangle_\sigma$ a $\langle y\rangle_\tau$: $x,y \in S$ dvou různých konzistentních rozkladů σ a τ neprázdný průnik, leží prvky průniku jak ve třídě $\langle x\rangle_\sigma$, tak ve třídě $\langle y\rangle_\tau$. Elementární chování automatu ve stavech z množin $\langle x\rangle_\sigma$ a $\langle y\rangle_\tau$ se proto bude shodovat s jeho elementárním chováním ve stavech ležících v průniku. Operace $*$ vybere z obou tříd společný průnik, operace $+$ obě třídy sjednotí. V obou případech tak nově vytvořené třídy konzistenci nenaruší.

M (nosič svazu). Vzhledem k tomu, že všechny prvky této čtveřice jsou zjistitelné z Hasseovského diagramu na Obr. 11.1, lze tento diagram považovat za grafickou reprezentaci svazu KM.

Výsledky operací $*$ a $+$ na konkrétním automatu nezávisí, avšak nosič příslušného svazu na něm závislý je. Např. ke zjišťování konzistence rozkladu je třeba vědět, ve kterých stavech daný automat vykazuje shodu v elementárním chování. Tato informace je beze zbytku obsažena v jediném nejhrubším konzistentním rozkladu. Proto je možno z tohoto jediného rozkladu výše popsaným postupem „shora dolů" zrekonstruovat i ostatní rozklady nosiče svazu konzistentních rozkladů. Obecně je možno k rekonstrukci nosiče svazu rozkladů daných vlastností z jeho nějakého „plně informativního torza" využít i dvojici operací $*$ a $+$, pokud obě tyto operace dané vlastnosti zachovávají. Přitom je třeba respektovat fakt, že se jedná o binární operace (ke každému výsledku je třeba dvou operandů), a že chybějící rozklad je možno získat jen jejich aplikací na nesrovnatelné operandy.

Obecně vhodné „plně informativní torzo" k tomuto účelu užívané sestává ze všech nejmenších rozkladů $\langle s,s'\rangle$ daných vlastností, obsahujících dvojici $s,s' \in S$ v jedné třídě rozkladu.[5] Postup rekonstrukce příslušného nosiče svazu lze pak shrnout do následujících tří kroků:

- Uspořádáme všechny rozklady $\langle s,s'\rangle$ do Hasseovského diagramu.
- V Hasseovském diagramu vyhledáme všechny dvojice nesrovnatelných prvků a aplikujeme na ně operace $*$ a $+$.
- Vzniknou-li tím nové rozklady, doplníme je do Hasseovského diagramu a opakujeme předchozí krok. Nevznikne-li žádný nový rozklad, je množina rozkladů uspořádaných v Hasseovském diagramu uzavřená vzhledem k operacím $*$ a $+$, je tedy hledaným nosičem svazu a Hasseovský diagram reprezentuje příslušný svaz.

Dalším významným svazem rozkladů, využívaným zejména při analýze dekomponovatelnosti automatu (viz poznámka pod čarou 7 v lekci 4, Úloha 4.5 a Příklad 4.7) je **svaz SV rozkladů** stavového prostoru automatu, definovaný čtveřicí SVM = (SV(M), $\leq$, $*$, $+$). Při jeho konstrukci vycházíme z nejjemnějších SV rozkladů $\langle s,s'\rangle$ vytvořených nad všemi dvojicemi $s,s' \in S$ algoritmem zadaným vývojovým diagramem na Obr. 11.2. Pro automat M z Obr. 10.8 dostáváme:

$\langle A,B\rangle = \{\{A,B\},\{C\},\{D\},\{E\},\{F\}\}$,
$\langle A,C\rangle = \langle A,E\rangle = \langle A,F\rangle = \langle B,D\rangle = \langle C,D\rangle = \langle C,E\rangle = \langle D,E\rangle = \langle D,F\rangle = \{\{A,B,C,D,E,F\}\}$,
$\langle A,D\rangle = \langle B,C\rangle = \langle B,E\rangle = \langle B,F\rangle = \langle C,F\rangle = \{\{A,D\},\{B,C,E,F\}\}$,
$\langle E,F\rangle = \{\{E,F\},\{A\},\{B\},\{C\},\{D\}\}$.

Uspořádáním těchto čtyř rozkladů do výchozího Hasseovského diagramu a následnou aplikací operací $*$ a $+$, které SV vlastnost rozkladů zachovávají,[6] vygenerujeme Hasseovský diagram

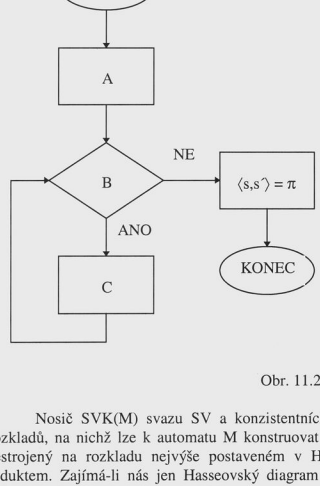
[5] Sestavíme jej takto:

- Z množiny S stavů automatu se vyberou všechny dvojice stavů, kterým vlastnosti rozkladu zjevně nezakazují výskyt ve stejné třídě.
- Pro každou takovou dvojici $s,s' \in S$ se vytvoří rozklad $\langle s,s'\rangle$, který (pokud existuje) je nejjemnějším rozkladem daných vlastností sdružujícím stavy s a s´ do jedné třídy.
- „Plně informativní torzo" sestává ze všech rozkladů $\langle s,s'\rangle$, které v předchozím kroku vznikly.

Je zřejmé, že v případě automatu M z Obr. 10.8 všechny nejjemnější **konzistentní** rozklady $\langle s,s'\rangle$ jsou umístěny v předposlední vrstvě diagramu na Obr. 11.1 (tj. bezprostředně nad nejjemnějším triviálním rozkladem).

[6] Zachování SV vlastnosti rozkladu operacemi $*$ a $+$ je třeba dokázat. Pro součin SV rozkladů π a σ platí:
$\pi*\sigma = \{\langle s\rangle_{\pi*\sigma}: s \in S\}$, kde $\langle s\rangle_{\pi*\sigma} = \langle s\rangle_\pi \cap \langle s\rangle_\sigma$;

svazu SV rozkladů na Obr. 11.3. V něm silně vyznačené rozklady se vyskytují i na Obr. 11.1 (i tam jsou silně vyznačeny), takže jsou nejen SV, ale i konzistentní a silně vyznačená podstruktura (jak v Obr. 11.3, tak v Obr. 11.1) znázorňuje **svaz SV a konzistentních rozkladů** SVKM = (SVK(M), ≤, ∗, +) automatu M z Obr. 10.8.



Legenda: A: Zadán rozklad π množiny S obsahující kromě samých jednoprvkových tříd i jednu dvouprvkovou třídu {s,s´}.

B: $\exists(a,b\in S,\ i\in I{:}): \langle a\rangle_\pi = \langle b\rangle_\pi$, $\langle\delta(a,i)\rangle_\pi \neq \langle\delta(b,i)\rangle_\pi$? (Existuje v některé ze tříd rozkladu π nějaká dvojice různých stavů (a,b), která na nějaký podnět i nepřechází do téže třídy rozkladu π?)

C: $\pi \approx (\langle\delta(a,i)\rangle_\pi \cup \langle\delta(b,i)\rangle_\pi) \cup (\pi - \{\langle\delta(a,i)\rangle_\pi, \langle\delta(b,i)\rangle_\pi\})$ (Sjednocení zjištěné dvojice tříd $\langle\delta(a,i)\rangle_\pi \neq \langle\delta(b,i)\rangle_\pi$ narušujících podmínku SV rozkladu tím, že jsou v π disjunktní.)

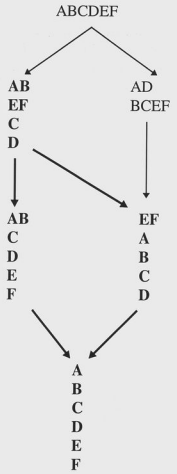
Obr. 11.2

Nosič SVK(M) svazu SV a konzistentních rozkladů SVKM sestává právě z těch rozkladů, na nichž lze k automatu M konstruovat podílové automaty. A podílový automat, sestrojený na rozkladu nejvýše postaveném v Hasseovském diagramu svazu SVKM, je reduktem. Zajímá-li nás jen Hasseovský diagram SVKM, můžeme jej vygenerovat přímo např. tím, že z rozkladů ⟨s,s´⟩ získaných algoritmem na Obr. 11.2 použijeme jen ty, které jsou konzistentní. V případě automatu M z Obr. 10.8 existují takové rozklady dva: {{A,B},{C},{D},{E},{F}} a {{E,F},{A},{B},{C},{D}}. Jsou nesrovnatelné a výsledky jejich součtu a součinu je doplní na nosič SVK(M) svazu SVKM.

$\langle a\rangle_{\pi+\sigma} = \langle b\rangle_{\pi+\sigma} \Rightarrow \exists s\in S$: a,b$\in\langle s\rangle_{\pi+\sigma} \Rightarrow$ a,b$\in(\langle s\rangle_\pi \cap\langle s\rangle_\sigma) \Rightarrow$ a,b$\in\langle s\rangle_\pi$, a,b$\in\langle s\rangle_\sigma \Rightarrow \forall i\in$ I: $(\delta(a,i),\delta(b,i)\in\langle\delta(s,i)\rangle_\pi$, $\delta(a,i),\delta(b,i)\in\langle\delta(s,i)\rangle_\sigma) \Rightarrow \forall i\in$ I: $\delta(a,i),\delta(b,i)\in(\langle\delta(s,i)\rangle_\pi \cap\langle\delta(s,i)\rangle_\sigma) \Rightarrow \forall i\in$ I: $\delta(a,i),\delta(b,i)\in\langle\delta(s,i)\rangle_{\pi+\sigma} \Rightarrow \forall i\in$ I: $\langle\delta(a,i)\rangle_{\pi+\sigma} = \langle\delta(b,i)\rangle_{\pi+\sigma}$.

Součet SV rozkladů π a σ sjednocuje třídy rozkladů π a σ s neprázdným průnikem a sestavuje z nich rozklad π+σ. Z algoritmu výpočtu plyne: $\langle s\rangle_\pi \subseteq \langle s\rangle_{\pi+\sigma}$ a $\langle s\rangle_\sigma \subseteq \langle s\rangle_{\pi+\sigma}$. Proto také $(\langle s\rangle_\pi \cup\langle s\rangle_\sigma) \subseteq \langle s\rangle_{\pi+\sigma}$.

$\langle a\rangle_{\pi+\sigma} = \langle b\rangle_{\pi+\sigma} \Rightarrow \exists s\in S$: a,b$\in(\langle s\rangle_\pi \cup\langle s\rangle_\sigma) \Rightarrow \forall i\in$ I: $\delta(a,i),\delta(b,i)\in(\langle\delta(s,i)\rangle_\pi \cup\langle\delta(s,i)\rangle_\sigma) \Rightarrow \forall i\in$ I: $\delta(a,i),\delta(b,i)\in\langle\delta(s,i)\rangle_{\pi+\sigma} \Rightarrow \forall i\in$ I: $\langle\delta(a,i)\rangle_{\pi+\sigma} = \langle\delta(b,i)\rangle_{\pi+\sigma}$.



Obr. 11.3

11.3. Redukce automatu shora a zdola

Základním problémem redukce automatu postupem znázorněným na Obr. 10.7 je potřeba opakovaného hledání SV a konzistentních rozkladů stavového prostoru podílového automatu, získaného jako výsledek předchozího kroku. V předchozích článcích této lekce jsme uvedli tři postupy, umožňující najít všechny SV a konzistentní rozklady stavového prostoru automatu bezprostředně z jeho zadání, aniž bychom postupně vytvářeli příslušné podílové automaty. Přitom jsme viděli, že množinou stavů reduktu je:

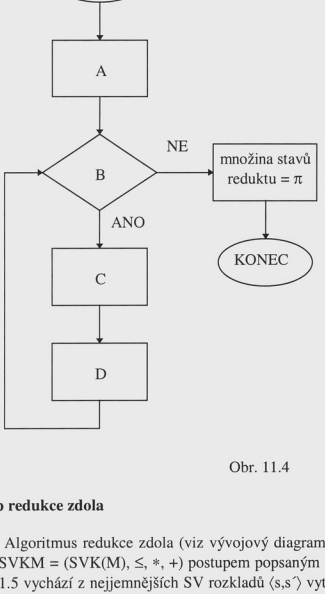
- nejvýše položený (nejhrubší) rozklad v Hasseovském diagramu svazu SVKM,
- nejvýše položený (nejhrubší) SV rozklad v Hasseovském diagramu svazu KM,
- nejvýše položený (nejhrubší) konzistentní rozklad v Hasseovském diagramu svazu SVM.

V tomto článku uvedeme dva různé postupy, kterými se lze k nejhrubšímu SV a konzistentnímu rozkladu stavového prostoru redukovaného automatu nejsnáze dostat.

Postup redukce shora

Algoritmus redukce shora (viz vývojový diagram na Obr. 11.4) se opírá o konstrukci svazu KM = (K(M), ≤, *, +) postupem „shora dolů", popsaným v článku 11.1. Prvním zkoumaným rozkladem je zde (stejně jako tam) nejhrubší rozklad v nosiči K(M), sdružující do stejné třídy všechny stavy redukovaného automatu se shodným subvektorem reakcí (krok A).

Navíc (stejně jako pak i v každém následujícím zkoumaném rozkladu) v něm hledáme třídu, jejíž prvky podmínku SV rozkladu nesplňují (krok B). Pokud ve zkoumaném rozkladu (rozklad π) žádná taková třída není, jsme v cíli.[7] Pokud nějaká taková třída existuje, rozložíme ji na podtřídy (rozklad σ jedné třídy rozkladu π) tak, aby v každé podtřídě byly právě všechny stavy shodující se v subvektoru tříd svých následníků rozkladu π (krok C). V posledním kroku (krok D) nahradíme rozkládanou třídu jejím rozkladem, čímž získáme nový a jemnější zkoumaný rozklad π. Postup se vrací ke kroku B.[8]



Legenda: A: $\pi = \{\langle s\rangle_\pi : s \in S\}$,
$\langle a\rangle_\pi = \langle b\rangle_\pi \Leftrightarrow$
$\forall i \in I: \beta(a,i) = \beta(b,i)$
(Sestrojit nejhrubší konzistentní rozklad π množiny S.)

B: $\exists (a,b \in S, i \in I:): \langle a\rangle_\pi = \langle b\rangle_\pi$,
$\langle \delta(a,i)\rangle_\pi \neq \langle \delta(b,i)\rangle_\pi$?
(Existuje v některé ze tříd rozkladu π nějaká dvojice různých stavů (a,b), která na nějaký podnět i nepřechází do téže třídy rozkladu π?)

C: $\sigma = \{\langle s\rangle_\sigma : s \in \langle a\rangle_\pi\}$,
$\langle x\rangle_\sigma = \langle y\rangle_\sigma \Leftrightarrow$
$\forall i \in I: \langle \delta(x,i)\rangle_\pi = \langle \delta(y,i)\rangle_\pi$
(Konstrukce rozkladu σ zjištěné třídy $\langle a\rangle_\pi$ podle subvektoru tříd následníků jejích prvků v rozkladu π.)

D: $\pi \approx (\pi - \langle a\rangle_\pi) \cup \sigma$
(Zjemnění rozkladu π záměnou třídy $\langle a\rangle_\pi$ rozkladem σ.)

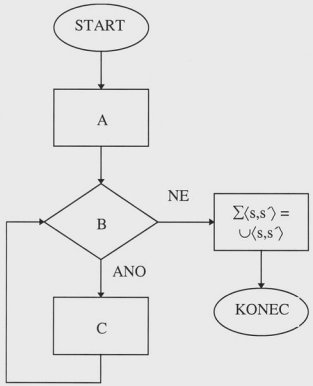
Obr. 11.4

Postup redukce zdola

Algoritmus redukce zdola (viz vývojový diagram na Obr. 11.5) se opírá o konstrukci svazu SVKM = (SVK(M), ≤, *, +) postupem popsaným v závěru článku 11.2. Algoritmus na Obr. 11.5 vychází z nejjemnějších SV rozkladů $\langle s,s'\rangle$ vytvořených nad všemi **konzistentními** dvojicemi stavů $s,s' \in S$ algoritmem zadaným vývojovým diagramem na Obr. 11.2. Vzhledem k tomu, že v kroku C algoritmu na Obr. 11.2 dochází ke slučování disjunktních tříd rozkladu,

[7] U automatu M_1 na Obr. 10.1 (viz také Obr. 10.3) byl SV už první zkoumaný rozklad.
[8] V případě automatu M na Obr. 10.8 dostaneme následující posloupnost zkoumaných rozkladů:
{{A,B,D}, {C,E,F}} → {{A,B}, {D}, {C,E,F}} → {{A,B}, {D}, {C}, {E,F}}.

může výsledný nejjemnější SV rozklad $\langle s,s' \rangle$ postrádat konzistenci, i když výchozí rozklad s jedinou dvouprvkovou třídou $\{s,s'\}$ konzistentní byl. Proto v kroku A algoritmu na Obr. 11.5 nejprve z rozkladů $\langle s,s' \rangle$ (vygenerovaných algoritmem na Obr. 11.2 pro všechny konzistentní dvojice (s,s')) vybereme jen ty, které jsou **konzistentní** a všechny tyto rozklady sjednotíme. Tím vytvoříme systém množin $\cup\langle s,s' \rangle$, který je pokrytím množiny stavů S,[9] nikoli však jejím rozkladem.



Legenda:

A: Zadán systém množin $\cup\langle s,s' \rangle$, vzniklý sjednocením všech konzistentních rozkladů $\langle s,s' \rangle$ množiny S.

B: $\exists X,Y \in \cup\langle s,s' \rangle$: $X \cap Y \neq \emptyset$? (Je pravda, že $\cup\langle s,s' \rangle$ není rozkladem množiny S?)

C: $\cup\langle s,s' \rangle \approx (\cup\langle s,s' \rangle - \{X,Y\}) \cup (X \cup Y)$ (Náhrada dvojice množin s neprázdným průnikem jejich sjednocením.)

Obr. 11.5

V kroku B hledáme v systému $\cup\langle s,s' \rangle$ dvojici množin s neprázdným průnikem. Pokud žádnou nenajdeme, je $\cup\langle s,s' \rangle$ hledaným rozkladem, což vyjádříme jeho přejmenováním na $\sum\langle s,s' \rangle$ a algoritmus končí. V opačném případě obě množiny s neprázdným průnikem v systému $\cup\langle s,s' \rangle$ nahradíme jejich sjednocením (krok C) a vracíme se nazpět ke kroku B.

Je zřejmé, že každý průchod algoritmu krokem C snižuje počet množin v pokrytí $\cup\langle s,s' \rangle$ o jednu, takže po konečném počtu průchodů krokem C zůstanou v pokrytí $\cup\langle s,s' \rangle$ jen disjunktní třídy. Z porovnání s algoritmem na Obr. 4.5 (v poznámce pod čarou 7 lekce 4) je také zřejmé, že algoritmus na Obr. 11.5 je zobecněním binární operace + (součet rozkladů) zachovávající jak konzistenci tak i SV vlastnost rozkladů (viz poznámky pod čarou 4 a 6 v této lekci) na větší počet operandů. Vzhledem k tomu, že shodný výsledek (jako algoritmem na Obr. 11.5) získáme i postupným přičítáním SV a konzistentních rozkladů $\langle s,s' \rangle$ algoritmem na Obr. 4.5, je rozklad $\sum\langle s,s' \rangle$ také SV a konzistentní.[10]

[9] Prvky systému $\cup\langle s,s' \rangle$ jsou podmnožiny množiny S. Jejich sjednocením bychom dostali právě množinu S.

[10] V závěru článku 11.2 jsme viděli, že v případě automatu M z Obr. 10.8 existují dva konzistentní rozklady $\langle s,s' \rangle$: {{A,B},{C},{D},{E},{F}} a {{E,F},{A},{B},{C},{D}}. Proto v kroku A algoritmu zadané výchozí

Zbývá tedy už jen odpovědět na otázku: „Jak víme, že rozklad $\sum\langle s,s'\rangle$ je nejvýše položený SV rozklad v Hasseovském diagramu konzistentních rozkladů?“ Je tomu tak proto, že pro každý SV a konzistentní rozklad π platí: $\pi \leq \sum\langle s,s'\rangle$. Toto tvrzení dokážeme sporem. Za tím účelem budeme předpokládat, že tvrzení „každý SV a konzistentní rozklad $\pi \leq \sum\langle s,s'\rangle$“ neplatí, tj. že buď „existuje SV a konzistentní rozklad $\pi > \sum\langle s,s'\rangle$“, nebo že „existuje SV a konzistentní rozklad π nesrovnatelný s rozkladem $\sum\langle s,s'\rangle$“:

- „existuje SV a konzistentní rozklad $\pi > \sum\langle s,s'\rangle$“ $\Rightarrow \exists a,b \in S$: $(\langle \mathbf{a}\rangle_\Sigma \neq \langle \mathbf{b}\rangle_\Sigma, \langle a\rangle_\pi = \langle b\rangle_\pi)$;
  $\langle a\rangle_\pi = \langle b\rangle_\pi \Rightarrow \exists \langle a,b\rangle$: $\langle a,b\rangle \leq \sum\langle s,s'\rangle$;
  $\langle a,b\rangle \leq \sum\langle s,s'\rangle \Rightarrow \langle \mathbf{a}\rangle_\Sigma = \langle \mathbf{b}\rangle_\Sigma$.
- „existuje SV a konzistentní rozklad π nesrovnatelný se $\sum\langle s,s'\rangle$“ $\Rightarrow$
  $\exists a,b,c \in S$: $(\langle a\rangle_\pi = \langle b\rangle_\pi, \langle b\rangle_\Sigma = \langle c\rangle_\Sigma, \langle a\rangle_\pi \neq \langle c\rangle_\pi, \langle \mathbf{a}\rangle_\Sigma \neq \langle \mathbf{c}\rangle_\Sigma)$;
  $(\langle a\rangle_\pi = \langle b\rangle_\pi, \langle b\rangle_\Sigma = \langle c\rangle_\Sigma) \Rightarrow \exists \langle a,b\rangle, \langle b,c\rangle$: $(\langle a,b\rangle \leq \sum\langle s,s'\rangle, \langle b,c\rangle \leq \sum\langle s,s'\rangle)$;
  $(\langle a,b\rangle \leq \sum\langle s,s'\rangle, \langle b,c\rangle \leq \sum\langle s,s'\rangle) \Rightarrow (\langle a\rangle_\Sigma = \langle b\rangle_\Sigma, \langle b\rangle_\Sigma = \langle c\rangle_\Sigma) \Rightarrow \langle \mathbf{a}\rangle_\Sigma = \langle \mathbf{c}\rangle_\Sigma$.

V obou případech je spor v důsledcích zvýrazněn tučně.

---

pokrytí $\cup\langle s,s'\rangle$ = {{A,B},{A},{B},{C},{D},{E},{F},{E,F}}. Po čtyřech průchodech krokem C algoritmu dostaneme $\sum\langle s,s'\rangle$ = {{A,B},{C},{D},{E,F}}.

Cvičení k lekci 11.

Příklad 11.1: Sestrojte Hasseovský diagram svazu SV rozkladů stavového prostoru automatu, jehož přechodová funkce je zadána tabulkou na Obr. 11.6.

S/I	a	b	c	d	e
1	2	1	5	8	3
2	1	2	6	7	4
3	4	3	6	6	1
4	3	4	5	5	2
5	5	6	3	4	7
6	6	5	4	3	8
7	7	8	4	2	5
8	8	7	3	1	6

Obr. 11.6

Řešení:

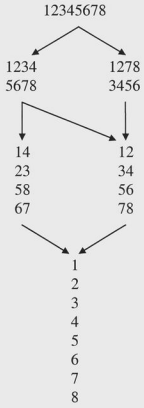
$\langle 1,2\rangle = \langle 3,4\rangle = \langle 5,6\rangle = \langle 7,8\rangle = \{\{1,2\}, \{3,4\}, \{5,6\}, \{7,8\}\}$

$\langle 1,4\rangle = \langle 2,3\rangle = \langle 5,8\rangle = \langle 6,7\rangle = \{\{1,4\}, \{2,3\}, \{5,8\}, \{6,7\}\}$

$\langle 1,3\rangle = \langle 2,4\rangle = \langle 5,7\rangle = \langle 6,8\rangle = \{\{1,2,3,4\}, \{5,6,7,8\}\}$

$\langle 1,7\rangle = \langle 1,8\rangle = \langle 2,7\rangle = \langle 2,8\rangle = \langle 3,5\rangle = \langle 3,6\rangle = \langle 4,5\rangle = \langle 4,6\rangle = \{\{1,2,7,8\}, \{3,4,5,6\}\}$

$\langle 1,5\rangle = \langle 1,6\rangle = \langle 2,5\rangle = \langle 2,6\rangle = \langle 3,7\rangle = \langle 3,8\rangle = \langle 4,7\rangle = \langle 4,8\rangle = \{\{1,2,3,4,5,6,7,8\}\}$



Obr. 11.7