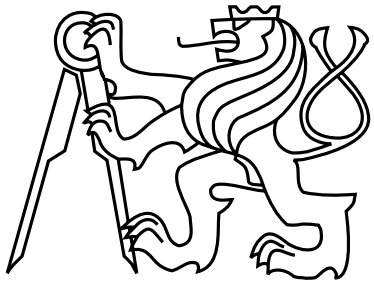
České vysoké učení technické v Praze
Fakulta elektrotechnická



Bakalářská práce

Dekompozice sekvenčních obvodů

Václav Jaša

Vedoucí práce: Doc. Ing. Hana Kubátová, CSc

Studijní program: Elektrotechnika a informatika strukturovaný bakalářský

Obor: Informatika a výpočetní technika

leden 2007

Poděkování

Chtěl bych poděkovat své vedoucí práce Doc. Ing. Haně Kubátové, CSc. za její odborné vedení, cenné připomínky a pomoc při vypracovávání.

Prohlášení

Prohlašuji, že jsem svou bakalářskou práci vypracoval samostatně a použil jsem pouze podklady uvedené v přiloženém seznamu.
Nemám závažný důvod proti užití tohoto školního díla ve smyslu §60 Zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Praze dne 27.12.2007 .....

Abstract

This thesis summarizes the knowledge about the decomposition of sequential circuits. The decomposition allows a better optimalization of the implementation of digital circuits, particulary of FPGS chips, quite popular in nowadays.Thanks to the decomposition we are able to improve i.e. the critical path of the circuits, better place distribution or reduction of a number of intup and output signals. In this thesis there is introduced a short overview of methods and principles used for the decomposition together with theory important for it. The main idea of the thesis is a concept of general decomposition of incompletely specified machines with a multi-state realization. A method for finding of such decomposition is introduced at last together with its implementation, which realization is a part of this thesis.

Abstrakt

Tato práce je shrnutím poznatků týkajících se dekompozice sekvenčních obvodů. Dekompozice umožňuje lepší optimalizaci implementace digitálních obvodů, především u v současnosti hojně užívaných FPGA chipů. Díky dekompozici je možno dosahovat např. kratších kritických cest, lepšího rozmístění na ploše chipu či omezení počtu vstupů a výstupů. V této práci je uveden stručný přehled jednotlivých druhů a metod dekompozice spolu s teorií a pojmy důležitými pro dekompozici. Hlavním bodem je potom představení obecné dekompozice sekvenčního automatu s vícestavovou realizací. Je zde uveden princip tohoto druhu dekompozice a také příklady pro větší názornost. V závěru je představena metoda hledání takovéto dekompozice a popsána její implementace, jejíž realizace je také součástí práce.

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

Většina dnešních digitálních obvodů je implementována pomocí Glushkovova modelu procesu zpracování informací. Ten se skládá ze dvou částí: operační části, kde se provádí všechny datové operace a řídící částí (řadiče), která tyto datové operace řídí. Právě řadič potom podle [4] zabírá až 80% prostoru celého obvodu. Je proto důležité při tvorbě systému věnovat pozornost právě řadiči, jehož vhodnou implmentací se mohou výrazně zlepšovat vlastnosti celého obvodu.

Dekompozicí se myslí rozklad komplexního systému na menší a jednodušší dílčí části společně splňující funkčnost celku a jako taková je běžně a často užívanou metodikou v mnoha oborech lidské činnosti. Dekompozice sekvenčního obvodu je potom jednou ze základních a nejúčinějších metod zjednodušení a zlešení implementace rozsáhlých systémů, obzvláště při dnešnímu trendu stále se zvyšující komplexnosti číslicových obvodů, kdy implementace často naráží na konstrukční omezení cílových technologií. Právě pomocí rozličných metod dekompozice se tyto složité automaty rozloží na části dohromady splňující původní funkčnost, jejichž implementace splňuje technologická omezení daných chipů.

Možností dekompozice sekvenčních obvodů se začal v 50. Letech 20. století zabývat C. E. Shannon a na jeho práci poté navázaly Roth a Karpov, Ashenhurst, Povarov nebo Curtis [1], kteří postupně vytvořili teoretické základy používané dodnes. Významným počinem, ze kterého vychází i tato práce, bylo potom vytvoření algebraického modelu konečného automatu Hartmanisem a Stearnsem, kteří představil teorii rozkladů analyzující strukturu automatu. Ta byla Jóźwiakem upravena v [6, 7], kde představil algebraický model pro přenos informace mezi jednotlivými rozklady.

S tím, jak roste obliba FPGA obvodů roste také prostor pro využití dekomozice. Jelikož FPGA obvody jsou sice schopny snadno realizovat jakoukoliv logickou funkci, ale pouze pokud tato funkce splňuje technologické požadavky na počet vstupů a výstupu. Právě dekompozice funkce nebo konečného automatu a s ní spojené kódování vnitřních stavů je ideální cesta, jak naplno využívat předností těchto obvodů, ačkoliv bohužel často bývá opomíjena výrobci různých implementačních nástrojů.

Tato bakalářská práce obsahuje základními koncepty dekompozice sekvenčního obvodu, jejich rozdíly, přednosti i nevýhodami. Zároveň také předkládá podrobný popis a implementaci úplné obecné dekompozice neúplně zadaného FSM.

2 Teoretická část

V této části budou vysvětleny pojmy důležité pro pochopení postupů popsaných v pozdějších částech práce. Veškeré teoretické poznatky jsou čerpány z [2, 3, 6, 7, 8, 9, 11, 10, 12]. Postupně budou vysvětleny pojmy z oblastí:

- Konečné automaty
- Teorie rozkládů
- Informační aparát

2.1 Konečný automat

Definice 2.1.1 (Úplně zadaný FSM) *úplně zadaný automat M je algebraický systém definovaný jako* $M = \{I, S, O, \delta, \lambda\}$*, kde:*

I - *konečná množina vstupních symbolů*
S - *konečná množina stavů*
O - *konečná množina výstupních symbolů*
δ - *přechodová funkce*
λ - *výstupní funkce*

Definice 2.1.2 (Úplně zadaný FSM typu Moore) *úplně zadaný automat M typu Moore je automat, jehož přechodová a výstupní funkce je definována jako:*

$$\begin{aligned} \delta &: S \times I \to S \\ \lambda &: S \to O \end{aligned}$$

Definice 2.1.3 (Úplně zadaný FSM typu Mealy) *úplně zadaný automat M typu Mealy je automat, jehož přechodová a výstupní funkci je definována jako:*

$$\begin{aligned} \delta &: S \times I \to S \\ \lambda &: S \times I \to O \end{aligned}$$

Automaty typu Moore a Mealy jsou nejpoužívanějšími typy automatů. Jediný rozdíl mezi nimi je ve způsobu výpočtu výstupní funkce - výstup automatu typu Moore je závislý jen na vnitřním stavu ve kterém se nachází, zatímco výstup automatu typu Mealy je závislý jak na aktuálním vnitřním stavu, tak na aktuálním vstupním symbolu. Pro každý automat typu Moore potom existuje ekvivalentní automat typu Mealy a obráceně. Automaty jsou zadávány buď pomocí přechodovému diagramu nebo tabulky přechodů (viz Příklad 2.1.1)

Příklad 2.1.1 *Příklad automatu typu Moore - Tab. 2.1 a automatu typu Mealy - Tab. 2.2 zadaných pomocí tabulkya splňující stejnou funkci. Sloupec označený ps označuje aktuální stav, sloupce 0 a 1 označují následující stav pokud je vstupní symbol 0/1 a sloupec out u automatu Moore, popř.* out_0 *a* out_1 *u automatu Mealy označují výstup, resp. výstup pokud je vstupním symbolem 0/1*

ps	0	1	out
a	a	b	0
b	d	c	0
c	e	c	0
d	a	e	0
e	a	b	1

Tabulka 2.1: FSM typu Moore

ps	0	1	out_0	out_1
a	a	b	0	0
b	d	c	0	0
c	a	c	1	0
d	a	a	0	1

Tabulka 2.2: FSM typu Mealy

Z důvodu možnosti převodu obou typů automatu na druhý typ budu v dalším textu považovat všechny automaty za typ Mealy a při zadávání použiji následující formát tabulek viz Tab. 2.3, používaný ve formátu KISS 2. Sloupec *in* značí vstupní symbol, *ps* představuje aktuální stav, *ns* následující stav pokud je na vstupu daná proměnná a *out* představuje výstupní symbol za výše stanovených podmínek.

in	ps	ns	out
0	a	a	0
1	a	b	0
0	b	d	0
1	b	c	0
0	c	a	1
1	c	c	0
0	d	a	0
1	d	a	1

Tabulka 2.3: Příklad tabulky přechodů

Definice 2.1.4 (Neúplně zadaný FSM) *neúplně zadaný automat M je algebraický systém definován stejně jako úplně zadaný automat jen s tím rozdílem, že:*

$$\lambda \quad : \quad S \times I \rightarrow 2^O$$

Definice 2.1.5 (Nedeterministický FSM) *Nedeterministický automat M je algebraický systém definován stejně jako úplně zadaný automat jen s tím rozdílem, že:*

$$\delta \quad : \quad S \times I \rightarrow 2^S$$

Definice poukazují na fakt, že rozdíl mezi úplně a neúplně zadaným automatem, resp. nedeterministickým automatem je ve výběru následujícího stavu, resp. výstupního symbolu. Zatímco u úplně zadaného automatu je výběr stavu a výstupu jednoznačně určený, u neúplně zadaného nebo nedeterministického automatu může být vybrána množina následujících stavů nebo výstupních symbolů. V následujícím textu bude jako neůplný automat považován automat neúplný i nedeterministický a obráceně.

Příklad 2.1.2 *V Tab. 2.4 je příklad neúplně zadaného automatu. Nedeterminismus se zde projevuje ve stavu* a *a* b*, neúplnost automatu zase ve stavech* a *a* c*. V těchto není předem určeno, zda automat přejde do stavu* b *nebo* c*, resp.* a *nebo* c*. Stejně tak výstupní symboly stavů* a *a* c *nejsou jednoznačné.*

in	ps	ns	out
0	a	b,c	00,11
1	a	a	01
0	b	a,c	11
1	b	b	01
0	c	a	10,01
1	c	b	10,01,11

Tabulka 2.4: Neúplně zadaný FSM

2.2 Teorie rozkladů

Rozklady, potažmo pokrytí (nebude zde probíráno) a systémy množin, jsou základním prostředkem k popisu vztahů uvnitř FSM a pro pochopení principu dekompozice konečného automatu je nezbytné jim dostatečně porozumět.

Definice 2.2.1 (Rozklad) *Rozklad π_S na množině stavů S je definováno jako:*

$$\pi_S = \left\{ B_i \mid B_i \subseteq S \ \wedge \ \bigcup_1 B_i = S \ \wedge \ i \neq j \Rightarrow B_i \cap B_j = 0 \right\}$$

Rozklad na množině stavů S je soubor podmnožin (bloků) $B_1, B_2, \ldots, B_n$ obsahující jednotlivé stavy tak, že každý stav je vždy jen v jedné podmnožině (dále jen bloku) a zároveň sloučením všech bloků musí vzniknout celá množina S.

Příklad 2.2.1 $\pi_S = \{\overline{12}, \overline{3}, \overline{45}\}$ *je rozkladem definovaným nad množinou stavů* $S = \{1, 2, 3, 4, 5\}$, $\pi_S = \{\overline{12}, \overline{23}, \overline{45}\}$ *rozkaldem není.*

Definice 2.2.2 (Systém množin) *Systém množin ϕ_S na množině stavů S je definováno jako:*

$$\phi_S = \left\{ B_i \mid B_i \subseteq S \ \wedge \ \bigcup_1 B_i = S \ \wedge \ i \neq j \Rightarrow B_i \not\subseteq B_j \right\}$$

Systém množin množině stavů S je soubor bloků $B_1, B_2, \ldots, B_n$ obsahující jednotlivé stavy tak, že každý stav je minimálně v jednom bloku, sloučením všech bloků musí vzniknout S a zároveň žádný blok nesmí být podmnožinou jiného bloku.

Příklad 2.2.2 *$\phi_S = \{\overline{12}, \overline{23}, \overline{45}\}$ je systémem množin na množině stavů $S = \{1, 2, 3, 4, 5\}$, zatímco $\phi_S = \{\overline{1}, \overline{12}, \overline{23}, \overline{45}\}$ není.*

Definice 2.2.3 (Součin rozkladů) *Součin rozkladů π_S^1 a π_S^2 je definován jako:*

$$\pi_S = \left\{ B_i \mid \exists B^1 \in \pi_S^1 \exists B^1 \in \pi_S^2 \ : \ B_i = B^1 \cap B^2 \quad \wedge \quad i \neq j \Rightarrow B_i \cap B_j = 0 \right\}$$

Bloky rozkladu π_S vzniklého součinem dvou rozkladů (platí i pro systémy množin) vzniknou průnikem všech možných kombinací dvou bloků takových, že jeden blok náleží do rozkladu π_S^1 a druhý do rozkladu π_S^2

Příklad 2.2.3 $\{\overline{12}, \overline{345}\} \cdot \{\overline{15}, \overline{2}, \overline{34}\} = \{\overline{1}, \overline{2}, \overline{34}, \overline{5}\}$

Definice 2.2.4 (Relace $\leq$) *Relace $\pi_S^1 \leq \pi_S^2$ je platná tehdy, pokud:*

$$\forall B^1 \in \pi^1 \exists B^2 \in \pi^2 : B^1 \subseteq B^2$$

Rozklad (platí i o systémech množin) π^1 je menší nebo roven než π^2 pouze a jen tehdy, pokud každý z jeho bloků je roven nebo podmnožinou bloku z rozkladu π^2.

Příklad 2.2.4 *pro $\{\overline{13}, \overline{2}, \overline{4}, \overline{5}\}$ a $\{\overline{123}, \overline{45}\}$ platí relace $\leq$, ale pro rozklady $\{\overline{13}, \overline{4}, \overline{25}\}$ a $\{\overline{123}, \overline{45}\}$ již neplatí, protože blok $\overline{25}$ není podmnožinou žádného bloku z druhého rozkladu.*

Definice 2.2.5 (Systémový součet) *Systém množin ϕ_S, vzniklý součtem dvou systémů množin nad množinou symbolů* S *vznikne spojením obou systémů množin a vypuštěním takových bloků, které již jsou podmnožinou jiného bloku.*

Příklad 2.2.5 $\{\overline{12}, \overline{23}, \overline{4}, \overline{5}\} + \{\overline{1}, \overline{2}, \overline{3}, \overline{45}\} = \{\overline{12}, \overline{23}, \overline{45}\}$ *je systémový součet a* $\{\overline{12}, \overline{3}, \overline{45}\} + \{\overline{12}, \overline{34}, \overline{5}\}$
$= \{\overline{12}, \overline{345}\}$ *je rozkladový součet*

Definice 2.2.6 (Nulový rozklad) *Nulový rozklad π_S je takový rozklad, ve kterém je každý symbol z množiny S ve svém vlastním bloku.*

Příklad 2.2.6 *Nulový rozklad -* $\pi_S(0) = \{\overline{1}, \overline{2}, \overline{3}, \overline{4}, \overline{5}\}$

Definice 2.2.7 (Jednotkový rozklad) *Jednotkový rozklad π_S je takový rozklad, ve kterém jsou všechny symboly z množiny S v jediném bloku.*

Příklad 2.2.7 *Jednotkový rozklad -* $\pi_S(1) = \{\overline{12345}\}$

Definice 2.2.8 (Indukovaný rozklad) *Indukovaný rozklad $\pi_{A\times B}^A$ je takový rozklad nad prvky množiny kartézského součinu množin A a B, který obsahuje všechny prvky tohoto součinu obsahující elementy jednoho bloku rozkladu A také v jednom bloku tohoto indukovaného rozkladu.*

Příklad 2.2.8 *A = {a,b,c,d}, B = {0,1} a* $\pi_A = \{\overline{a,b}, \overline{c,d}\}$. *Rozklad indukovaný na průniku těchto množin množinou* A: $\pi^A_{A\times B} = \{\overline{(a,0),(a,1),(b,0),(b,1)}, \overline{(c,0),(c,1),(d,0),(d,1)}\}$

Definice 2.2.9 (Systém množin C(ϕ)) *Systém množin C(ϕ) lze definovat tak, že prvky* $q_1, q_2, \ldots, q_n$ *náleží do stejného bloku právě tehdy, pokud ve* ϕ *existují bloky takové, že každá dvojice prvků z bloku C(ϕ) se vyskytuje v alespoň jednom bloku společně.*

Systém množin C(ϕ) se dá také definovat jako *nejmenší systém množin větší nebo rovný oběma sčítancům.*

Příklad 2.2.9 *Pro systém množin* $\phi = \{\overline{12}, \overline{23}, \overline{13}, \overline{24}, \overline{34}\}$ *je* $C(\phi) = \{\overline{123}, \overline{234}\}$

Definice 2.2.10 (Správný systém množin) *Systém množin* ϕ *je správným, pokud platí* $C(\phi) = \phi$.

Definice 2.2.11 (Rozkladový pár) *Dvojice rozkladů P(π^1_S, π^2_S) definovaných nad množinou stavů konečného automatu M je rozkladovým párem právě tehdy, pokud:*

$$\forall D \in \pi^1_S \exists B \in \pi^2_S : \overline{\delta}(D) \subseteq B$$

Z definice plyne, že dvojice rozkladů je rozkladovým párem P(π^1_S, π^2_S) jen tehdy, pokud je pro každý blok π^1_S možné spočítat množinu následujících stavů tak, že všechny tyto stavy jsou v jediném bloku rozkladu π^2_S. Totéž potom platí i o systémovém páru P(ϕ_1, ϕ_2).

Příklad 2.2.10 *Pro FSM specifikovaný v Tab. 2.5 je S = {a,b,c,d}. Máme rozklady* $\pi^1_S = \{ac,b,d\}$ *a* $\pi^2_S = \{ad,bd\}$. *Můžeme snadno ověřit, že každá množina následujících stavů* π^1_S *je podmnožinou bloku nebo blokem rozkladu* π^2_S, *proto platí P(π^1_S, π^2_S).*

Definice 2.2.12 (Substituční vlastnost) *O rozkladu říkáme (platí i pro systémy množin), že má substituční vlastnost SV(π) tehdy, pokud u něj platí: P(π^1_S, π^1_S).*

Příklad 2.2.11 *Pro FSM specifikovaný v Tab. 2.5 je S = {a,b,c,d}. Máme rozklad* $\pi = \{ab,cd\}$. *Snadno můžeme ověřit, že každá množina následujících stavů tohoto rozkladu je podmnožinou bloku nebo celým blokem téhož rozkladu. Proto platí SV(π), nebo-li rozklad* π *má substituční vlastnost.*

Definice 2.2.13 (Systém množin m(ϕ)) *Funkce m(ϕ) vypočítá systém množin následujích symbolů takový, že pokud jsou symboly v jednom bloku* ϕ, *budou v jednom bloku i v nově vytvořeném systému množin m(ϕ). Pokud už v m(ϕ) existuje takový blok, který obsahuje podmnožinu symbolů nového bloku, popř. nový blok je podmnožina bloku jiného, je menší blok nahrazen větším. Pokud existují symboly, které nejsou v žádném bloku, budou všechny tyto symboly v samostatných blocích.*

in	ps	ns	out
0	a	c	1
1	a	a	0
0	b	d	0
1	b	b	1
0	c	b	1
1	c	d	0
0	d	b	1
1	d	c	0

Tabulka 2.5: FSM k příkladu rozkladového páru a substituční vlastnosti

Definice 2.2.14 (Nalezení všech SV systémů množin) *Nechť $\phi_{i,j}$ je takový systém množin, že symboly* i *a* j *jsou v jednom bloku a všechny ostatní symboly v samostatných blocích. Pro výpočet SV systémů můžeme použí následující postup:*

$\tau^1_{i,j} = \phi_{i,j} + m(\phi_{i,j})$

$\tau^k_{i,j} = \tau^{k-1}_{i,j} + m(\tau^{k-1}_{i,j})$

...

Tento postup se opakuje, dokud se dva výsledky po sobě neopakují. Poté se ještě na výsledný systém množin aplikuje funkce C(ϕ). Takový systém množin má potom SV vlastnost. Celý postup se opakuje pro všechny kombinace i *a* j.

2.3 Informační aparát

Informační aparát je nástroj popsaný Józwiakem. Vychází z teorie rozkladů zavedené Hartmanisem, který sice formálně popsal konečný automat jako algebraickou strukturu, popsatelnou pomocí rozkladů, neuvědomoval si však souvislosti mezi jednotlivými rozklady a přenosy mezi nimi.

Tento aparát představuje tzv. základní informaci, pomocí které lze popsat děje uvnitř i mimo jednotlivé automatů. Systém popisuje kde a jak informace vzniká a kde je požadována, popisuje podobnost mezi jednotlivými infomačními toky (rozklady) a nabízí možnost měřit a kvantifikovat tuto informaci. Právě z této teorie vychází koncept použitý v této bakalářské práci.

Definice 2.3.1 (Základní informace) *Základní informace reprezentuje schopnost rozpoznat určitý symbol s_i od nějakého jiného symbolu s_j ($s_i, s_j \in S \ \wedge \ s_i \neq s_j$)*

Příklad 2.3.1 *Mějme automat zadaný Tab. 2.6. Sloupec S označuje tzv. symbol - konkrétní řádek automatu. Budeme-li znát proměnnou x_1, můžeme rozeznat mezi symboly (0,2,4,5,6), pokud bude mít hodnotu 0 nebo symboly (1,3,4,5,6), pokud bude mít hodnotu 1. Můžeme tedy říci, že automat se nachází na řádku 0 nebo 1, 0 nebo 3, 1 nebo 2 a 2 nebo 3 (zapisujeme následovně: $0|1, 0|3, 1|2, 2|3$), resp. při znalosti x_1 můžeme rozlišit mezi těmito symboly. Pokud toto přeneseme do teorie rozkladů a systémů množin, můžeme napsat: $\phi_{x1} = \{\overline{02456}, \overline{13456}\}$.*

Definice 2.3.2 (Informační sada) *Informační sada je množina základních informací, kterou můžeme získat z příslušného rozkladu nebo systému množin.*

S	x_1x_2	ps	ns	out
0	0-	a	b	0
1	1-	a	c	1
2	00	b	a	0
3	1-	b	c	1
4	-1	b	c	1
5	-0	c	b	0
6	-1	c	c	1

Tabulka 2.6: FSM k příkladu informace a informační sady

Příklad 2.3.2 *Mějme opět automat zadaný Tab. 2.6 a rozklad* $\phi_{x1} = \{\overline{02456}, \overline{13456}\}$. *Z tohoto systému množin lze získat informační sadu* $IS_{x1} = \{0|1, 0|3, 1|2, 2|3\}$.

Pomocí analýzy infomace a informačních sad je možné zjistit, jak se jaké vstupní hodnoty podílejí na přenosu informace ze vstupu na výstup, popř. jaká informace je vlastně potřeba a jakou je možno vypustit. Analýzou FSM použitého v příkladech 2.3.1 a 2.3.2 snadno můžeme zjistit tyto systémy množin a informační sady:

$\phi_{x1} = \{\overline{02456}, \overline{13456}\}, IS_{x1} = \{0|1, 0|3, 1|2, 2|3\}$
$\phi_{x2} = \{\overline{01235}, \overline{01346}\}, IS_{x2} = \{2|3, 2|4, 2|6, 4|5, 5|6\}$
$\phi_{ps} = \{\overline{01}, \overline{234}, \overline{56}\},\ \phi_{ns} = \{\overline{05}, \overline{1346}, \overline{2}\}, \phi_{out} = \{\overline{025}, \overline{1346}\}$
$IS_{ps} = \{0|2, 0|3, 0|4, 0|5, 0|6, 1|2, 1|3, 1|4, 1|5, 1|6, 2|5, 2|6, 3|5, 3|6, 4|5, 4|6\}$
$IS_{ns} = \{0|1, 0|2, 0|3, 0|4, 0|6, 1|2, 1|5, 2|3, 2|4, 2|5, 2|6, 3|5, 4|5, 5|6\}$
$IS_{out} = \{0|1, 0|3, 0|4, 0|6, 1|2, 1|5, 2|3, 2|4, 2|6, 3|5, 4|5, 5|6\}$

Tyto informace můžeme dále rozdělit na vstupní a výstupní, tedy tu informaci, která do systému vstupuje pomocí vstupních proměnných a tu informaci, kterou je třeba k výpočtu výstupních proměnných. Vstupní tedy budou ϕ_{x1}, ϕ_{x2} a ϕ_{ps} a výstupní ϕ_{ns} a ϕ_{out}. Můžeme si všimnout, že některé informace vstupující do systému nejsou potřeba k výpočtu výstupních informací. Minimalizace nebo dekompozice systému vede k odstranění právě těxhto přebytečných informací a kontrola zachování potřebné informace k výpočtu výstupních proměnných slouží zase jako ověření správnosti minimalizace či dekompozice.

Informační aparát je užitečný hlavně v případě, že chceme danou informaci kvantifikovat či porovnávat s jinou informací. Toho lze dosáhnout právě díky rozdělení informaci do atomických, jednoduše porovnatelných dvojic. V [6, 7] je rozlišeno mezi těmito charakteristikami míry informace:

- *společná informace* $CI(\phi_1, \phi_2) = IS(\phi_1) \cap IS(\phi_2)$
- *celková informace* $TI(\phi_1, \phi_2) = IS(\phi_1) \cup IS(\phi_2)$

- *chybějící informace* $MI(\phi_1, \phi_2) = IS(\phi_1) \setminus IS(\phi_2)$
- *přidaná informace* $EI(\phi_1, \phi_2) = IS(\phi_2) \setminus IS(\phi_1)$
- *rozdílná informace* $DI(\phi_1, \phi_2) = MI(\phi_1, \phi_2) \setminus EI(\phi_1, \phi_2)$

Na základě těchto charakteristik jsme pak schopni spočítat tyto vlastnosti:

- *schodnost informace* $ISIM(\phi_1, \phi_2) = |CI(\phi_1, \phi_2)|$
- *rozdílnost informace* $IDIS(\phi_1, \phi_2) = |DI(\phi_1, \phi_2)|$
- *ztrátovost informace* $IDEC(\phi_1, \phi_2) = |MI(\phi_1, \phi_2)|$
- *zisk informace* $IINC(\phi_1, \phi_2) = |EI(\phi_1, \phi_2)|$
- *celkové množství informace* $TIQ(\phi_1, \phi_2) = |TI(\phi_1, \phi_2)|$

Pokud bychom chtěli zkoumat rozklady pomocí relací $\cdot, +, \leq$, zjistíme, že relace $\cdot$ zvyšuje množství informace rozkladů, zatímco $+$ toto množství snižuje. Relace $\leq$ potom udává, že menší rozklad obsahuje více informace než větší rozklad a zároveň že informace většího rozkadu je podmnožinou rozkladu menšího.

Jóźwiak pomocí IRMA sice v [8] definuje další nástroje sloužící k popisu procesů uvnitř systému, jako je třeba násobnost informace či pravděpodobnost ztráty informace apod., tyto jsou však užitečné především pro dekompozici logické funkce, která se od dekompozice konečného automatu liší (ačkoliv logickou funkci lze také považovat za jednostavový FSM) a pro porozumění principu dekompozice FSM není jejich znalost a pochopení potřeba.

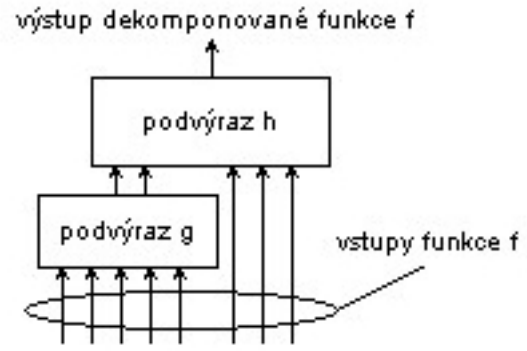
3 Dekompozice sekvenčního automatu

Dekompozice sekvenčního automatu spočívá v jeho rozdělení na několik menších a jednodušších částí, které dohromady splňují funkčnost dekomponovaného automatu. Existuje několik druhů dekompozice, každá se svým vlastním specifickým uplatněním. V této práci se budu věnovat především úplné obecné symbolické dekompozici sekvenčního obvodu (vysvětlení těchto pojmů viz níže), ale pro ucelení představy o dekompozici FSM budou představeny i další druhy dekompozice.

3.1 Dekompozice logické funkce

Ačkoliv logické funkce nejsou implementovány pomocí sekvenčních obvodů, je vhodné je pro přehled uvést. Dekompozice logické funkce znamená rozklad logické funkce na menší podvýrazy. Účelem této dekompozice je obvykle snížení počtu vstupních a výstupních signálů jednotlivých podvýrazů, čímže se dosahuje optimálnější implementace především u programovatelných obvodů (LUT u FPGA obvykle obsahuje 4 nebo 5 vstupů a 2 výstupy).

Dekompozice se provádí výběrem určitého počtu vstupních proměnných, které přenášejí určitou informaci potřebnou k výpočtu výstupní informace. Z těchto proměnných se poté vytvoří podvýraz tak, aby velikost výstupu odpovídala požadavkům na implementaci a zároveň aby se přenesla pokud možno všechna důležitá informace ze vstupních proměnných. Ty proměnné, které už mají veškerou potřebnou informaci obsaženou v tomto podvýrazu již nejsou potřeba a tudíž v dalších podvýrazech se již nepoužívají (viz Obr. 3.1). Podmínkou platné dekompozice logické funkce je, že počet výstupů podvýrazu musí být menší než počet vstupních proměnných, které se díky tomuto podvýrazu dále nemusí používat (toto je logické, jelikož každý výstup podvýrazu tvoří nový vstup celkové funkce a při nedodržení podmínky by se celkový počet vstupů tedy zvětšil).



Obrázek 3.1: Princip dekompozice logické funkce

3.2 Dekompozice sekvenčního obvodu

Dekompozice sekvenčního obvodu je hlavním tématem této práce. V následujícím textu bude uveden jak přehled různých druhů a metod dekompozice, tak detailně vysvětlena metoda úplné obecné dekompozice se zachováním vícestavové realizace a vysvětleny pojmy s touto metodou související.

Definice 3.2.1 (Podautomat) *Jeden funkční celek vystupující z procesu dekompozice, spolu s ostatními podautomaty vzešlé z této dekompozice splňuje funkci automatu do dekompozice vstupujícího.*

Definice 3.2.2 (Původní/originální automat) *Původním, nebo také originálním automatem, se myslí automat vstupující do procesu dekompozice. Podautomaty z této dekompozice vystupující musí společně splňovat funkci právě tohoto automatu.*

Definice 3.2.3 (Dekomponovaný automat) *Automat vzniklý vhodným propojením podautomatů. Má stejnou funkčnost i stejný počet vstupů a výstupů jako původní automat, jen jinou vnitřní strukturu.*

Dekompozice sekvenčních obvodů (FSM) je ve své podstatě mnohem komplexnější než dekompozice logické funkce. Zatímco u dekompozice funkce máme jako vstupující informaci jen informaci ze vstupních proměnných, u sekvenčního obvodu máme navíc informaci o stavu. Tím se nejen komplikuje výpočet výstupní fuknce, ale zároveň je třeba spočítat i následující stav, jinak dekomponovaný automat nebude splňovat funkci původního automatu.

3.2.1 Stavová a úplná dekompozice

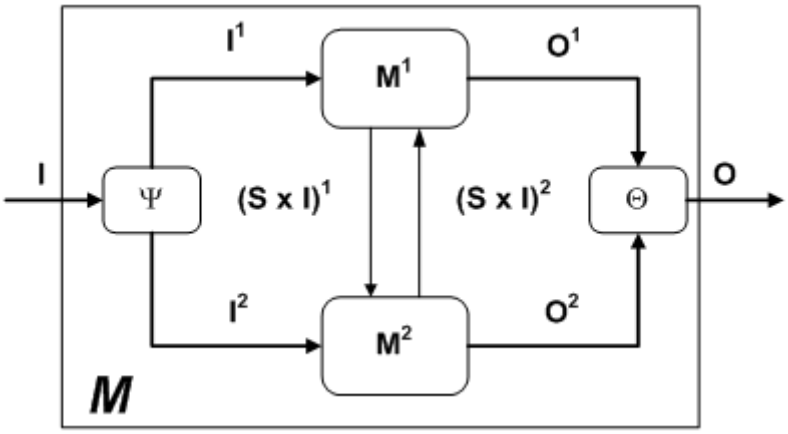
S ohledem na rozsah dekompozice rozeznáváme dekompozici buď stavovou nebo úplnou

Stavová dekompozice sekvenčního obvodu

Stavová dekompozice FSM byla prvním druhem dekompozice konečného automatu představena Hartmanisem v roce 1960. Její princip spočívá ve vytvoření několika podautomatů, které mají vstupy a výstupy stejné jako původní automat, ale mají menší počet vnitřních stavů. Toto se však pro PLA ukázalo jako nedostatečné, jelikož počet vstupních i výstupních proměnných zůstával neměnný a byla omezena možnost přizpůsobení automatu technologii.

Úplná dekompozice sekvenčního obvodu

Úplná dekompozice FSM spočívá v dekompozici nejen množiny stavů, ale také množiny vstupních a výstupních symbolů. Tím se dosahuje redukce počtu vstupních a výstupních proměnných jednotlivých podautomatů. Princip obecné úplné dekompozice je zobrazen na Obr. 3.2.



Obrázek 3.2: Úplná obecná dekompozice

3.2.2 Postupná a současná dekompozice

Podle průběhu spolupráce jednotlivých podautomatů rozeznáváme druhy dekompozicí:

Postupná dekompozice

Postupná dekompozcie se provádí rozkladem původního FSM na podautomaty spolupracující tak, že každý podautomat vykonává svou činnost v jiný okamžik než ostatní podautomaty. Proto také název postupná dekompozice, jelikož podautomaty pracují postupně. Používá se především při návrhu struktury mikroprogramových řadičů [10].

Současná dekompozice

Současná dekompozice funguje na principu okamžité spolupráce jednotlivých podautomatů. Aktuální výstup vzniká sloučením výstupů všech podautomatů (na rozdíl od postupné dekompozice, kde výstup tvoří vždy jen jeden podautomat)

3.2.3 Symbolická a bitová dekompozice

Symbolická dekompozice

Symbolická dekompozice se provádí bez znalosti kódování vstupních a výstupních symbolů, Musíme tedy pracovat s celým symbolem jako celkem a s celou množinou těchto symbolů.

Bitová dekompozice

Provádí se pokud známe kódování vstupních a výstupních symbolů. Metoda spočívá v dekompozici těchto symbolů ne v celku, ale po jednotlivých řádech bitů, čímž se dosahuje vyšší efektivity a flexibility, jelikož se nemusí pracovat s celou množinou symbolů, ale jen s částí.

3.2.4 Dekompozice se společnou a oddělenou výstupní a přechodovou funkcí

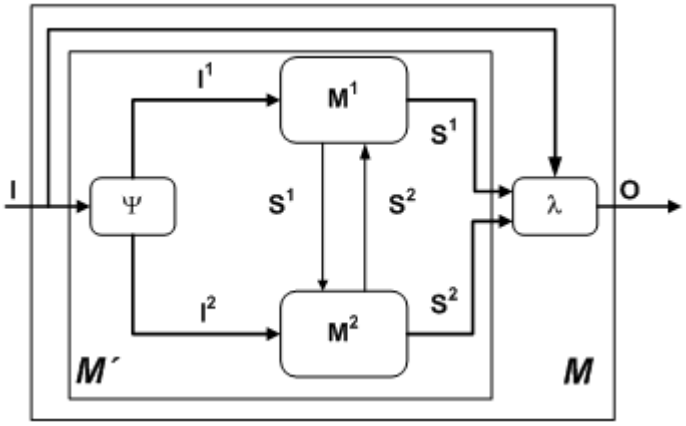
V obou případech se jedná o současnou úplnou dekompozici, oba druhy dekompozice jsou si podobné - mohou se používat k podobným účelům, vytvářet stejné hierarchie podautomatů

apod.

Dekompozice s oddělenou výstupní a přechodovou funkcí

Tato dekompozice se provádí pouze na množině vstupních symbolů a stavů. Výsledkem spolupráce podautomatů je pouze následující stav, nikoliv výstup (viz 3.3). Tento výstup se potom vypočítá ve výstupním bloku θ (viz Obr.) za pomoci informace o aktuálním stavu (dodané podautomaty) a informace o vstupu (dodané přímo ze vstupu dekomponovaného automatu).

Výhodou této dekompozice je snažší hledání vhodné dekompozice, jelikož se nedekomponují výstupní symboly. Nevýhodou je ale složitá výstupní funkce, která u složitějších automatů musí být sama dekomponována (viz Dekompozice logické funkce), což může být samo o sobě poměrně složité a zdlouhavé.



Obrázek 3.3: Obecná dekompozice s oddělenou realizací přechodové a výstupní funkce

Dekompozice se společnou výstupní a přechodovou funkcí

Jedná se o obecnější a univerzálnější typ dekompozice. Spočívá v dekompozici nejen množiny vstupů a stavů, ale také výstupů. Tomuto druhu dekompozice se věnuje tato práce a proto v následujícím textu bude dekompozice automaticky považována právě za dekompozici se společnou výstupní a přechodovou funkcí, ačkoliv jak bylo řečeno výše většina principů se dá buď beze změny nebo s drobnými úpravami použít i při dekompozici s oddělenou výstupní a přechodovou funkcí.

3.2.5 Struktura dekomponovaného automatu

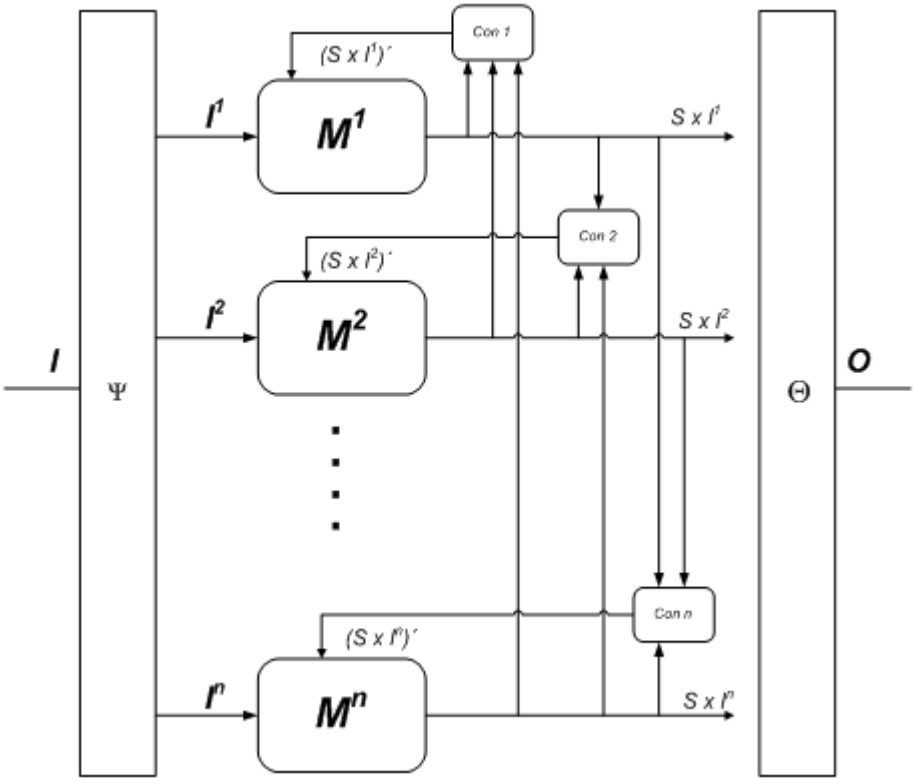
Ačkoliv struktura dekomponovaného automatu se může v mnohém lišit, základní prvky jsou většinou podobné (viz 3.4). Obecná struktura dekomponovaného automatu je zobrazena na Obr. a obsahuje tyto části:

Vstupní blok Ψ: Vstupní blok je reprezentován logickou funkcí, která na základě primárního vstupního symbolu vypočítá vstupní symboly pro jednotlivé podautomaty.

Podautomaty $M_1, \ldots, M_n$: Sekvenční obvody realizující část činnosti původního automatu.

Propojovací členy $con_1, \ldots, con_n$**:** Logické funkce vybírající z celkové informace dostupné od ostatních podautomatů informaci potřebnou právě pro daný podautomat.

Výstupní blok Θ**:** Výstupní blok je reprezentován logickou funkcí, která na základě výstupů z jednotlivých podautomatů vypočítává primární výstupní symbol.



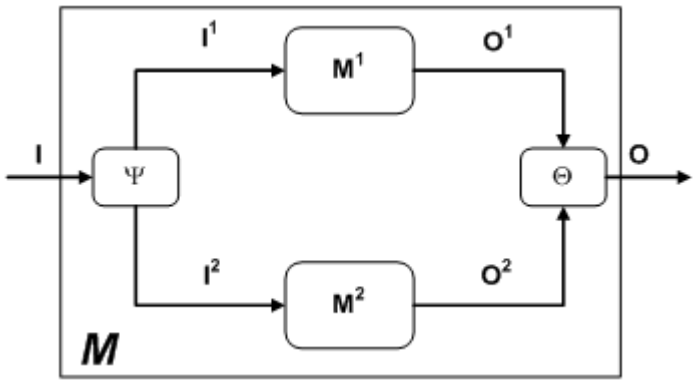
Obrázek 3.4: Struktura úplné obecné dekompozice s n podautomaty

3.2.6 Uspořádání podautomatů

Ačkoliv struktura propojení a spolupráce podautomatů může být velice různorodá, z důvodu dostatečné komplexnosti a také univerzálnosti se používají především tři druhy zapojení - paralelní, seriové a obecné. Tato propojení se mezi sebou liší v množství komunikace mezi jednotlivými podautomaty.

Paralelní dekompozice

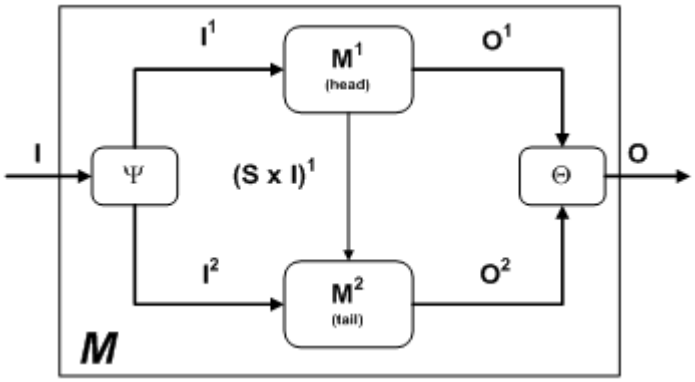
Jedná se o nezávisle pracující podautomaty, které mezi sebou nijak nekomunikují. Výsledný výstup se vypočítá až na základě výstupních symbolů jednotlivých podautomatů ve výstupním bloku. Každý podautomat tedy musí umět vypočítat svůj výstup a následný stav bez znalosti aktuálního stavu jiných podautomatů a informace o vstupním symbolu a stavu jsou jediné dostupné (viz Obr. 3.5).



Obrázek 3.5: Paralelní dekompozice

Sériová dekompozice

U tohoto způsobu propojení jeden podautomat pracuje samostatně bez informace o druhém podautomatu (označuje se head), a druhý automat (tail) používá informaci produkovanou tímto podautomatem k výpočtu výstupu a následujícího stavu. V případě sériového propojení více podautomatů jsou podautomaty uspořádány do takovýchto dvojic jdoucích za sebou (viz Obr. 3.6).



Obrázek 3.6: Sériová dekompozice

Obecná dekompozice

Každý podautomat tohoto propojení využívá informaci dostupnou z ostatních podautomatů k výpočtu výstupu a následujícího stavu. O vhodnou distribuci informace se starají propojovací členy, které vybírají jen tu informaci, kterou podautomat potřebuje k výpočtům (viz Obr. 3.2).

Porovnání uspořádání

Ačkoliv se může zdát použití paralelní dekompozice nejvýhodnější, nemusí tomu tak být vždy. Jednak ne u každého podautomatu může existovat paralelní dekompozice (podle [2] jsme jen u 30% podautomatů schopni najít paralelní dekompozici) a jednak při této dekompozici nemusí docházet k potřebné redukci vnitřních stavů, což snižuje kvalitu dekompozice ([6, 7]).

Obecná dekompozice má tu výhodu, že může používat i informaci dostupnou od ostatních podautomatů a tyto potom nemusí sami o sobě produkovat tolik informace, čímž se dá dosáhnout kvalitnější dekompozice. Na druhou stranu zatěžuje obvod jednak samotnými propojeními mezi podautomaty a jednak propojovacími členy, které mohou sami vyžadovat dekompozici, aby splnili technologické podmínky implementace.

Seriová dekompozice je potom někde mezi těmito dvěmi uspořádáními. Ne každý automat může mít platnou sériovou dekompozici a ne vždy musí být optimální co do zjednodušení struktury podatuomatu. Na druhou stranu nevyžaduje tolik komunikace mezi podautomaty a je tudíž méně náročná na implementaci.

3.3 Dekompozice úplně zadaného konečného automatu

Ačkoliv je tato práce zaměřena na dekompozici neúplně zadaného automatu, pro úplnost a snazší porozumnění problematice zde bude nejprve vysvětlen a předvedena dekompozice úplně zadaného automatu. Tato dekompozice se dá považovat také za podmnožinu dekompozice neúplně zadaného podautomatu, jen ne tolik obecnou.

Teorie použitá v této části práce vychází z [5]. Nejprve zde uvedu některé pojmy potřebné k pochopení principu dekompozice a poté pro názornost předvedu dekompozici na příkladu. Teoretická část popsaná v této části je potom základem pro dekompozici neúplně zadaného automatu, tudíž některé pojmy vysvětlené v této části budou beze změny používány nadále, některé jsou pro neúplně zadané automaty nedostačující a budou tedy nahrazeny sice podobnými, ale více obecnými pojmy.

3.3.1 Teorie

Účelem dekompozice je zredukovat počet stavů a vstupních a výstupních symbolů FSM tak, aby bylo možno automat lépe implmentovat na požadovanou platformu. K tomu se využívají rozklady u úplně zadaných FSM popř. systémy množin u neúplně zadaných FSM. Před vysvětlením principu dekompozice je třeba zavést následující rozklady:

Definice 3.3.1 (Rozklad nad množinou vstupních symbolů π_I) *Rozklad π_I je rozkladem (systémem množin) nad množinou vstupních symbolů, pokud sloučením všech bloků tohoto rozkladu vznikne množina všech vstupních symbolů.*

Definice 3.3.2 (Rozklad nad množinou stavů π_S) *Rozklad π_S je rozkladem (systémem množin) nad množinou stavů, pokud sloučením všech bloků tohoto rozkladu vznikne množina všech stavů.*

Definice 3.3.3 (Rozklad nad množinou výstupních symbolů $\pi_{S\times I}$) *Rozklad $\pi_{S\times I}$ je rozkladem (systémem množin) nad množinou výstupních symbolů, pokud sloučením všech bloků tohoto rozkladu vznikne množina všech prvků kartézského součinu množin vstupních symbolů a stavů.*

Definice 3.3.4 (Rozklad nad množinou primárních výstupních symbolů π_O) *Rozklad π_O je rozkladem (systémem množin) nad množinou primárních výstupních symbolů, pokud sloučením všech bloků tohoto rozkladu vznikne množina primárních výstupních symbolů, tedy výstupních symbolů původního automatu.*

Rozklad $\pi_{I\times S}$ byl zaveden jako rozklad, na který je přímo možné namapovat kombinace vstupních symbolů a aktuálních stavů. Tyto kombinace nám potom dávají veškerou informaci, kterou potřebujeme jak k určení výstupního symbolu, tak následujícího stavu. Ve starších pracích je uváděn jen rozklad π_O, ale zavedení $\pi_{I\times S}$ umožnilo využít efektivněji celkovou vstupní informaci a tím zlepšit výslednou dekompozici.

Definice 3.3.5 (Bloková přechodová funkce $\overline{\delta}$)

$$\overline{\delta} : 2^{S\times I} \rightarrow 2^{S}$$

Funce $\overline{\delta}(D)$ vypočítá ze všech prvků bloku D rozkladu $\pi_{S\times I}$ následující stav a vybere blok z rozkladu π_S takový, že obsahuje všechny tyto následující stavy.

Definice 3.3.6 (Bloková výstupní funkce $\overline{\lambda}$)

$$\overline{\lambda} : 2^{S\times I} \rightarrow 2^{O}$$

Funce $\overline{\lambda}(D)$ vypočítá ze všech prvků bloku D rozkladu $\pi_{S\times I}$ výstupní symbol a vybere blok z rozkladu π_O takový, že obsahuje všechny tyto symboly.

Příklad 3.3.1 *Mějme automat zadaný tabulkou 3.1, dvoublokový rozklad nad množinou výstupních prvků $\pi_{S\times I} = \{(a0, b0), (a1, b1, c0, c1)\}$. $\overline{\lambda}(D_1) = \{\lambda(a, 0), \lambda(b, 0), \} = \{00, 11\}$ a $\overline{\delta}(D_1) = \{\delta(a, 0), \delta(b, 0), \} = \{c\}$, $\overline{\lambda}(D_2) = \{01, 10\}$ a $\overline{\delta}(D_2) = \{a, b\}$*

in	ps	ns	out
0	a	c	00
1	a	a	01
0	b	c	11
1	b	b	01
0	c	a	10
1	c	b	10

Tabulka 3.1: Úplně zadaný FSM

Definice 3.3.7 (Rozkladový pár S - S) *Pár rozkladů π_S^1 a π_S^2 je párem typu S - S tehdy, pokud oba rozklady jsou rozklady nad množinou stavů a pokud platí:*

$$\forall D \in \pi_S^1 \ \exists B \in \pi_S^2 \ : \ \overline{\delta}(D) \subseteq B$$

Definice 3.3.8 (Rozkladový pár I - S) *Pár rozkladů je párem typu I - S tehdy, pokud jeden rozklad je rozkladem nad množinou vstupních symbolů a druhý nad množinou stavů a pokud platí:*

$$\forall A \in \pi_I \ \forall B \in \pi_S \ \exists B' \in \pi_S \ : \ \overline{\delta}_A(B) \subseteq B'$$

Definice 3.3.9 (Rozkladový pár $S \times I$ - S) *Rozkladový pár je párem typu $S \times I$ - S tehdy, pokud jeden rozklad je rozkladem nad množinou výstupních symbolů a druhý nad množinou stavů a pokud platí:*

$$\forall D \in \pi_{S\times I}\ \exists B \in \pi_S\ :\ \overline{\delta}(D) \subseteq B$$

Definice 3.3.10 (Rozkladový pár $S \times I$ - O) *Rozkladový pár je párem typu $S\times I$ - O tehdy, pokud jeden rozklad je rozkladem nad množinou výstupních symbolů a druhý nad množinou primárních výstupů a pokud platí:*

$$\forall D \in \pi_{S\times I}\ \exists C \in \pi_O\ :\ \overline{\lambda}(D) \subseteq C$$

Příklad 3.3.2 *Mějme automat zadaný tabulkou 3.1 a rozklad $\pi_{S\times I}$ z příkladu 3.3.1. Máme-li rozklady $\pi_S = (\overline{a,b};\overline{c})$ a $\pi_O = (\overline{00,11};\overline{01,10})$ tak po aplikaci funcí $\overline{\delta}$ a $\overline{\lambda}$ vidíme, že tyto dvojice jsou rozkladové páry $S \times I$ - S, resp. $S \times I$ - O.*

Definice 3.3.11 (Rozkladová trojice $\pi_I, \pi_S, \pi_{S\times I}$) *Trojice rozkladů $(\pi_I, \pi_S, \pi_{S\times I})$ je rozkladovou trojicí automatu M jen tehdy, pokud $(\pi_{S\times I}, \pi_S)$ tvoří $S \times I$ - S pár a $\pi_S \cdot \pi_I \leq \pi_{S\times I}$, popř. $\pi_S \cdot \pi_I \geq \pi_{S\times I}$ (jedno musí být menší nebo rovno druhému).*

Rozkladová trojice v podstatě reprezentuje vytvořený podautomat. Bloky π_I reprezentují vstupní symboly, bloky π_S jednotlivé stavy a $\pi_{S\times I}$ výstupní symboly tohoto podautomatu. Samotný proces dekompozice potom spočívá právě ve hledání nejvhodnějších rozkladových trojic, kdy se spojují jednotlivé bloky původního automatu tak, aby jejich počet byl co nejnižší a dekomponovaný automat stále splňoval požadavky na funkčnost původního automatu.

Definice 3.3.12 (Podmínky platnosti obecné dekompozice) *Mějme konečný automat M a množinu rozkladových trojic nad tímto automatem, nechť $\pi^{ij}_{S\times I}$ je rozklad takový, že $\pi^{i}_{S\times I} \leq \pi^{ij}_{S\times I}$ a ${\pi'_{S\times I}}^{j} = \prod_{i=1\ldots n} \pi^{ij}_{S\times I}$. Nechť $\pi_{S\times I} = \prod_{i=1\ldots n} \pi^{i}_{S\times I}$. Nechť ${\pi^I_{S\times I}}^{i}$ a ${\pi^S_{S\times I}}^{i}$ jsou rozklady indukované na množině $S \times I$ pomocí rozkladů π^i_S a π^i_I. Nechť $\pi^I_{S\times I} = \prod_{i=1\ldots n} {\pi^I_{S\times I}}^{i}$ a $\pi^S_{S\times I} = \prod_{i=1\ldots n} {\pi^S_{S\times I}}^{i}$. Aby tato dekompozice byla platná, musí platit:*

1. $({\pi^S_{S\times I}}^{i} \cdot {\pi^I_{S\times I}}^{i} \cdot {\pi'_{S\times I}}^{i}, \pi^i_S)$ *je rozkladový pár $S \times I$ - S*
2. ${\pi^S_{S\times I}}^{i} \cdot {\pi^I_{S\times I}}^{i} \cdot {\pi'_{S\times I}}^{i} \leq \pi^i_{S\times I}$
3. $\pi^S_{S\times I} \cdot \pi^I_{S\times I} \leq {\pi'_{S\times I}}^{i}$
4. $(\pi_{S\times I}, \pi_O(0))$ *je rozkladový pár $S \times I$ - O*
5. $\prod_i \pi^i_S = \pi_S(0)$

Pokud množina rozkladových trojic splňuje tyto podmínky, lze být použita pro tvorbu dekomponovaného automatu. Rozklad $\pi^{ij}_{S\times I}$ reprezentuje informaci produkovanou podautomatem i a importovanou podautomatem j, ${\pi'_{S\times I}}^{j}$ je celková informace importovaná podautomatem j produkovaná ostatními podautomaty, $\pi_{S\times I}$ je součinem výstupních rozkladů všech podautomatů, stejně jako $\pi^I_{S\times I}$ a $\pi^S_{S\times I}$ jsou součinem rozkladů na množině $S \times I$ indukovaných pomocí množiny S nebo I všech podautomatů realizujících dekompozici.

První pravidlo znamená, že na základě všech vstupních informací musí být možné jednoznačně vypočítat blok rozkladu π_S, druhé že výstupní rozklad musí být větší (obsahovat méně

informace) než maximální možná dostupná vstupní informace (nebylo by možné dosáhnout informace, jež do systému nevstupuje), třetí že imporované informace musí být méně než získané na základě vnitřní struktury podautomatu. Čtvrté potom klade požadavek na jednoznačnost primárního výstupního symbolu, tedy že z průniku množin informací o výstupech z jednotlivých podautomatů musí být možné jednoznačně vypočítat primární výstupním symbol. Páte pravidlo zaručuje, že bude v jakémkoliv okamžiku jednoznačně rozlišitelný aktuální stav celého dekomponovaného automatu a není nutné k dosažení funkčnosti původního automatu, ačkoliv některé aplikace mohou vyžadovat zachování tohoto chování.

Definice 3.3.13 (Podmínky platnosti paralelní dekompozice) *Množina rozkladových trojic tvoří platnou paralelní dekompozici pokud platí:*

1. $({\pi^S_{S\times I}}^i \cdot {\pi^I_{S\times I}}^i, \pi^i_S)$ *je rozkladový pár* $S \times I$ - S
2. ${\pi^S_{S\times I}}^i \cdot {\pi^I_{S\times I}}^i \leq \pi^i_{S\times I}$
3. $(\pi_{S\times I}, \pi_O(0))$ *je rozkladový pár* $S \times I$ - O
4. $\prod_i \pi^i_S = \pi_S(0)$

Definice 3.3.14 (Podmínky platnosti sériové dekompozice) *Množina rozkladových trojic tvoří platnou sériovou dekompozici pokud platí:*

1. *u prvního podautomatu platí:* $({\pi^S_{S\times I}}^1 \cdot {\pi^I_{S\times I}}^1, \pi^1_S)$ *je rozkladový pár* $S \times I$ - S*, u ostatních potom:* $({\pi^S_{S\times I}}^i \cdot {\pi^I_{S\times I}}^i \cdot \pi^{i(i-1)}_{S\times I}, \pi^i_S)$ *je rozkladový pár* $S \times I$ - S
2. *u prvního podautomatu platí:* ${\pi^S_{S\times I}}^1 \cdot {\pi^I_{S\times I}}^1 \leq \pi^1_{S\times I}$*, u ostatních potom:* ${\pi^S_{S\times I}}^i \cdot {\pi^I_{S\times I}}^i \cdot \pi^{i(i-1)}_{S\times I} \leq \pi^i_{S\times I}$
3. *u všech podautomatů kromě prvního platí:* $\pi^S_{S\times I} \cdot \pi^I_{S\times I} \leq \pi^{i(i-1)}_{S\times I}$
4. $(\pi_{S\times I}, \pi_O(0))$ *je rozkladový pár* $S \times I$ - O
5. $\prod_i \pi^i_S = \pi_S(0)$

3.3.2 Příklad dekompozice úplně zadaného automatu

Následující příklad je ukázkou, jak pomocí dvojice rozkladových trojic vytvořit dekomponovaný automat o dvou podautomatech. Původní automat je zadán tabulkou 3.2, sloupec S označuje symbol nebo-li řádek automatu, *in* vstupní symbol, *ps* vnitřní stav, *ns* následující stav a *out* výstupní symbol.

Mějme dvě rozkladové trojice reprezentující podautomaty M_1 a M_2 vypsané níže. Rozklady jsou zde uvedeny jednak jako symboly dané množiny (stavy, vstupní symboly), jednak rozepsány jako množiny řádků a jednak jako symbolické označení jednotlivých bloků (s, i, y, ...), které je dále použito v tabulkách.

S	in	ps	ns	out
1	00	a	a	00
2	01	a	b	00
3	10	a	c	11
4	11	a	d	01
5	00	b	b	10
6	01	b	b	10
7	10	b	d	01
8	11	b	c	11
9	00	c	c	01
10	01	c	b	01
11	10	c	c	00
12	11	c	d	00
13	00	d	d	10
14	01	d	b	11
15	10	d	d	11
16	11	d	c	00

Tabulka 3.2: Úplně zadaný FSM - příklad dekompozice

podautomat M_1:

$$\pi_S^1 = \{\overline{a,b}, \overline{c,d}\} = \{\overline{1,2,3,4,5,6,7,8}, \overline{9,10,11,12,13,14,15,16}\} = \{s_1, s_2\}$$

$$\pi_I^1 = \{\overline{00}, \overline{01}, \overline{10,11}\} = \{\overline{1,5,9,13}, \overline{2,6,10,16}, \overline{3,4,7,8,11,12,15,16}\} = \{i_1, i_2, i_3\}$$

$$\pi_{S\times I}^1 = \{\overline{1,2,5,6}, \overline{3,4,7,8,14,16}, \overline{9,13}, \overline{11,12,15,16}\} = \{y_1, y_2, y_3, y_4\}$$

podautomat M_2:

$$\pi_S^2 = \{\overline{a,c}, \overline{b,d}\} = \{\overline{1,2,3,4,9,10,11,12}, \overline{5,6,7,8,13,14,15,16}\} = \{t_1, t_2\}$$

$$\pi_I^2 = \{\overline{00,10}, \overline{01}, \overline{11}\} = \{\overline{1,3,5,7,9,11,13,15}, \overline{2,6,10,14}, \overline{4,8,12,16}\} = \{j_1, j_2, j_3\}$$

$$\pi_{S\times I}^2 = \{\overline{1,3,9,11}, \overline{2,4,10,12}, \overline{5,6,8,14,16}, \overline{7,13,15}\} = \{z_1, z_2, z_3, z_4\}$$

Nejprve je potřeba ověřit, zda obě tyto trojice splňují podmínky dekompozice. Snadno přitom zjistíme, že trojice podautomatu M_2 nesplňuje podmínku 1. Aby tato podmínka byla splněna, je potřeba importovat navíc informace $5|7, 5|13$ a $5|15$. Tato informace je dostupná z výstupu podautomatu M_1, proto ji přivedeme na vstup M_2. Nemusíme navíc importovat celou informaci, ale propojovací člen zajistí redukci informace na ${\pi'_{S\times I}}^2 = \{\overline{1,2,5,6}, \overline{3,4,7,8,9,10,11,12,13,14,15,16}\} = \{k_1, k_2\}$. Import této informace potom vede ke splnění všech podmínek pro oba podautomaty.

X_1	X_2	π_I^1	π_I^2
0	0	i_1	j_1
0	1	i_2	j_2
1	0	i_3	j_1
1	1	i_3	j_3

Tabulka 3.3: logická funkce vstupního bloku

Vstupní blok Ψ

V tomto bloku se uskuteční transformace primárního vstupního symbolu na vstupní symboly jednotlivých podautomatů. Toho lze jednoduše dosáhnout logickou funkcí:

Implementace vstupní funkce vyžaduje zakódování symbolických jmen do nějakého binárního tvaru. Je proto nutno počítat s tím, že pro realizaci funkce bude zapotřebí minimálně $\lceil \log_2 |B| \rceil$ výstupních proměnných ($|B|$ označuje počet symbolů).

Stejně jako vstupní funkce je implementováná i výstupní funkce, popř. propojovací člen, i zde je tedy vhodné vyhnout se složitým výrazům s mnoha vstupními resp. výstupními symboly, jejichž implementace by vyžadovala také dekompozici, která sama o sobě bývá poměrně náročná.

Podautomaty M_1 a M_2

Při konstrukci jednotlivých podautomatů využijeme skutečnost, že lze pro oba rozklady nad množinou stavů jednoznačně vypočítat blok stavů následujících. Můžeme proto vytvořit tabulku přechodů tak, že vstupní symboly budou tvořeny bloky vstupního rozkladu, stavy určeny bloky rozkladu nad stavy, následující stavy budou potom určeny na základě kombinace aktuálního bloku a vstupního symbolu, stejně jako výstupní symbol. Např. pokud bude aktuální stav $s_1 = \overline{a, b}$ a aktuální vstup $i_3 = \overline{10, 11}$, můžeme jednoznačně určit následující blok a výstupní symbol z kombinace symbolů $(\overline{3, 4, 7, 8})$, což je blok $s_2 = \overline{c, d}$ a symbol $y_2 = \overline{3, 4, 7, 8, 14, 16}$.

in	ps	ns	out
i_1	s_1	s_1	y_1
i_2	s_1	s_1	y_1
i_3	s_1	s_2	y_2
i_1	s_2	s_2	y_3
i_2	s_2	s_1	y_2
i_3	s_2	s_2	y_4

Tabulka 3.4: tabulka přechodů podautomatu M_1

Tab. 3.4 a 3.5 představují kompletní přechodové tabulky pro oba podautomaty. Stavy a vstupní a výstupní symboly jsou zde označeny symbolickými hodnotami uvedenými výše u popisu rozkladů. Oproti původnímu automatu je zde viditelné snížení počtu stavů. U prvního podautomatu je také redukce počtu vstupních symbolů, u druhého tomu tak není z důvodu dalšího vstupu importované informace. Počet výsupních symbolů je stejný jako u původního automatu, takže zde nedochází k žádné redukci, ale ani k žádnému nárustu. Ve třetím řádku

označuje symbol „-“ skutečnost, že všechny vstupní bloky mají v takovéto kombinaci nulový průnik, tudíž tato situace nemůže nastat.

in	imp in	ps	ns	out
j_1	k_1	t_1	t_1	z_1
j_2	k_1	t_1	t_2	z_2
j_3	k_1	t_1	-	-
j_1	k_1	t_2	t_2	z_3
j_2	k_1	t_2	t_2	z_4
j_3	k_1	t_2	t_1	z_3
j_1	k_2	t_1	t_1	z_1
j_2	k_2	t_1	t_2	z_2
j_3	k_2	t_1	t_2	z_2
j_1	k_2	t_2	t_2	z_4
j_2	k_2	t_2	t_2	z_4
j_3	k_2	t_2	t_1	z_3

Tabulka 3.5: tabulka přechodů podautomatu M_2

Propojovací člen con_2

Propojovací člen se vytvoří stejně jako vstupní blok logickou funkcí (viz Tab. 3.6). Jejím úkolem je jen spojit nepotřebnou informaci do jednoho bloku a tím zminimalizovat komplexnost propojení podautomatů.

Y_1	imp in
x_1	k_1
x_2	k_2
x_3	k_2
x_4	k_2

Tabulka 3.6: logická funkce propojovacího členu con_2

Výstupní blok Θ

Stejně jako vstupní blok a propojovací člen je blok tvořen logickou funkcí vstupních proměnných tvořenými výstupy z podautomatů a výstupnímy symboly tvořeny symboly primárního výstupu (Tab. 3.7).

3.4 Dekompozice neúplně zadaného automatu

V předchozí části byla představena dekompozice úplně zadaného automatu, pracující s rozklady a determinismem vyplývajícím z jednoznačnosti zadání automatu. V případě neúplně zadaných automatů však toto není dostatečné a je třeba definovat nástroje pro práci s nejednoznačnostmi takovýchto automatů.

M_1	M_2	out
y_1	z_1	00
y_1	z_2	01
y_1	z_4	11
y_2	z_1	11
y_2	z_2	00
y_2	z_3	00
y_2	z_4	10
y_3	z_1	01
y_3	z_4	10
y_4	z_1	00
y_4	z_2	00
y_4	z_3	00
y_4	z_4	11

Tabulka 3.7: logická funkce výstupního bloku

Nejvýraznější změnou je potom zavedení systémů množin místo rozkladů, jelikož rozklady nejsou schopny pokrýt *don't-care* stavy, jelikož tyto symboly by byly poté obsaženy ve více blocích, což odporuje definici rozkladu. Dále je ještě třeba nahradit funkce vypočítávající následující blok funkcemi vypočítávající množinu následujících bloků a představit teorii umožňující tuto dekompozici.

3.4.1 Teorie

Definice uvedené v této části jsou zobecněním definic z části předchozí. Dají se tedy všechny aplikovat i na automat definovaný úplně. Níže uvedená metoda je nejobecnější metodou dekompozice a je aplikovatelná na všechny FSM [9].

Definice 3.4.1 (Operátor $\overline{\bigcup}$)

$$\overline{\bigcup}([D_i]_{i=1\ldots n}) = \{B|\exists C^1 \subseteq D_1, C^1 \neq 0 \ldots \exists C^n \subseteq D_n, C^n \neq 0 \ : \ B = \bigcup_{i=1\ldots n} C^n\}$$

Operátor $\overline{\bigcup}$ vybere z vektoru podmnožin symbolů S všechny možné neprázdné kombinace.

Příklad 3.4.1

$$\overline{\bigcup} = \left(\left[\begin{array}{c}\{1,2,3\}\\ \{1,4\}\end{array}\right]\right) = \{\{1\},\{1,2\},\{1,3\},\{1,4\},\{2,4\},\{3,4\},$$
$$\{1,2,3\},\{1,2,4\},\{2,3,4\},\{1,2,3,4\}\}$$

Definice 3.4.2 (Operátor $\overline{\bigcap}$)

$$\overline{\bigcap}([D_i]_{i=1\ldots n}) = \{B|\exists C^1 \in \ldots \exists C^n \in D_n \ : \ B = \bigcap_{i=1\ldots n} C^n\}$$

Operátor $\overline{\bigcap}$ přijímá vektor množin podmnožin symbolů S a vybere všechny možné průniky těchto podmnožin.

Příklad 3.4.2

$$\overline{\bigcap} = \left(\begin{bmatrix} \{1\}\{1,2\}\{2,3\}\{4\} \\ \{1,2,3\}\{4,5\} \\ \{1,2,4,5\} \end{bmatrix} \right) = \{\{1\}\{1,2\}\{2\}\{4\}\}$$

Pomocí těchto dvou nástrojů můžeme zavést upravený pojem *systémová přechodová funkce*, která nahrazuje *blokovou přechodovou funkci* a pojem S × I - S systémový pár:

Definice 3.4.3 (Systémová přechodová funkce Λ^{ϕ_S})

$$\Delta^{\phi_S}(D_i) = \{B' \in \phi_S | \exists B'' \in \overline{\delta}(D) \ : \ B'' \subseteq B'\}$$

Tato funkce vybere ze systému množin nad množinou stavů všechny možné množiny následujících stavů.

in	ps	ns	out
0	a	b,c	00,11
1	a	a	01
0	b	a,c	11
1	b	b	01
0	c	a	10,01
1	c	b	10,01,11

Tabulka 3.8: Neúplně zadaný FSM

Příklad 3.4.3 *Mějme FSM zadaný tabulkou 3.8 a systém množin definovaný nad množinou výstupních symbolů* $\phi_{S\times I} = \{\overline{(a,0),(b,0)}; \overline{(a,1),(b,1),(c,0),(c,1)}\} = \{D_1, D_2\}$ *potom platí:*

$$\Delta(D_1) = \overline{\bigcup} \begin{bmatrix} \delta(a,0) \\ \delta(b,0) \end{bmatrix} = \overline{\bigcup} \begin{bmatrix} (b,c) \\ (a,c) \end{bmatrix} = \{\{a,b\};\{a,c\};\{b,c\};\{c\};\{a,b,c\}\}$$

$$\Delta(D_2) = \overline{\bigcup} \begin{bmatrix} \delta(a,1) \\ \delta(b,1) \\ \delta(c,0) \\ \delta(c,1) \end{bmatrix} = \overline{\bigcup} \begin{bmatrix} (a) \\ (b) \\ (a) \\ (b) \end{bmatrix} = \{\{a,b\}\}$$

Definice 3.4.4 (Systémový pár S × I - S)

$$\forall D \in \phi_{S\times I} \ : \ \Delta^{\phi_S}(D) \neq 0$$

Definice vyjadřuje fakt, že aby byla dvojice S × I - S systémovým párem, stačí pouze, aby mezi množinami následujících bloků byl alespoň jeden podmnožinou bloku či blokem ϕ_S.

Příklad 3.4.4 *Mějme FSM zadaný tabulkou 3.8 a systém množiny $\phi_{S\times I}$ z příkladu 3.4.3 a $\phi_S = \{\overline{a,b,c};\overline{b,c}\}$. Můžeme vidět, že v případě obou bloků $\phi_{S\times I}$ jsou mezi množinamy následujících stavů podmnožiny bloků systému množin ϕ_S a tudíž tyto dva systémy množin tvoří systémový rozklad.*

Výše zmíněné funkce jsou však stále nedostatečné pro jednoznačný výpočet následujícího stavu dekomponovaného automatu. Jelikož každý podautomat by mohl zvolit libovolnou podmnožinu následujících stavů a průnik těchto následujících stavů všech podautomatů by mohl být prázdný, je třeba zajistit, aby každý podautomat přešel do stavu reprezentujícího podmnožinu stavů, do které by přešel i původní automat. To vede k pojmu synchronizovaná množina systémových páru S × I - S.

Definice 3.4.5 (Synchronizovaná množina S × I - S páru) $([\phi^i_{S\times I}]_i, [\phi^i_S]_i)$ *je Synchronizovaný systémový pár jen tehdy, pokud:*

1. *Pokud je systémovým párem podle definice 3.4.4*
2. *Pokud průnik aktuálních bloků celé množiny $\phi^i_{S\times I}$ vede do průniku všech následujících bloků ϕ^i_S každého páru z množiny.*

Podobně potom můžeme definovat i funkce a pojmy pro výpočet výstupních symbolů dekomponovaného automatu. I zde platí, že tyto definice budou platit i pro úplně zadaný automat.

Definice 3.4.6 (Bloková výstupní funkce)

$$\overline{\lambda} : 2^{S\times I} \to 2^O \ a\ \overline{\lambda}(D) = \overline{\bigcup}_{d\in D}\lambda(d)$$

Bloková výstupní funkce pro neúplný automat se liší od výstupní funkce pro úplný automat tím, že nevypočítá jeden výstup ale množinu výsutpů.

Definice 3.4.7 (Systémová výstupní funkce)

$$\Lambda^{\phi_O} : 2^{S\times I} \to 2^O \ a\ \Lambda(D) = \{C \in \phi_O \mid \exists C' \in \overline{\lambda}(D) : C' \subseteq C\}$$

Systémová výstupní funkce vrací blok systému množin nad primárními výstupními symboly, který obsahuje výstupní symboly získané pomocí blokové výstupní funkce.

Příklad 3.4.5 *Mějme automat a systém množin zadané v příkladu 3.4.3, potom můžeme zjistit:*

$$\Lambda(D_1) = \overline{\bigcup}\begin{bmatrix}\lambda(a,0)\\ \lambda(b,0)\end{bmatrix} = \overline{\bigcup}\begin{bmatrix}(00,11)\\ (11)\end{bmatrix} = \{\{00,11\};\{11\}\}$$

$$\Lambda(D_2) = \overline{\bigcup}\begin{bmatrix}\lambda(a,1)\\ \lambda(b,1)\\ \lambda(c,0)\\ \lambda(c,1)\end{bmatrix} = \overline{\bigcup}\begin{bmatrix}(01)\\ (01)\\ (10,01)\\ (10,01,11)\end{bmatrix} = \{\{01\};\{10,01\};\{01,11\};\{10,01,11\}\}$$

Definice 3.4.8 (Systémový pár S × I - O)

$$\forall D \in \phi_{S\times I} \ : \ \Lambda^{\phi_0}(D) \neq 0$$

Definice vyjadřuje fakt, že aby byla dvojice S × I - O systémový pár stačí pouze, aby mezi množinami výstupních symbolů byl alespoň jeden podmnožinou bloku či celým blokem ϕ_O.

Příklad 3.4.6 *Mějme automat a systém množin zadané v příkladu 3.4.3 a* $\phi_O = \{\overline{01}; \overline{11}; \overline{00, 10}\}$, *můžeme vidět, že pro* $\Lambda^{\phi_O}(D_1) = \{\overline{11}\} \neq 0$ *a pro* $\Lambda^{\phi_O}(D_2) = \{\overline{01}\} \neq 0$ *a tudíž* $\phi_{S\times I}$ *a* ϕ_O *tvoří systémový pár S × I - O.*

Výše zmíněné pojmy pak vedou k zavedení tzv. *specializace systému množin.* Zohledňuje fakt, že díky existenci don't care stavů ve výstupních symbolech je možné zredukovat množinu výstupních symbolů zrušením těch symbolů, které mohou být nahrazeny právě díky možnosti vždy vybrat tento a nebo nějaký jiný symbol.

Definice 3.4.9 (Specializace systému množin) *Systém množin* $\phi_{S'}$ *je specializací systému množin* ϕ_S *tehdy, pokud S' je podmnožinou S a pokud:*

$$\forall A \in \phi_S \exists B \in \phi_{S'} : B \subseteq A \ \wedge \ \forall B \in \phi_{S'} \exists A \in \phi_S : A \supseteq B$$

Jak již bylo zmíněno výše, použití systémů množin má ten následek, že jeden symbol může být ve více blocích. Pokud by dekomponovaný automat měl přejít do stavu, který je ve více blocích, nebylo by možné vybrat jednoznačně blok, do kterého by se měl daný podautomat přesunout, proto je třeba zavést tzv. *vícestavovou realizaci automatu M.*

Definice 3.4.10 (Vícestavová realizace konečného automatu) *Neúplně zadaný automat* M_l *je vícestavovou realizací automatu M jen tehdy, pokud:*

1. *Množiny vstupních a výstupních symbolů obou automatů jsou shodné*
2. *Existuje funkce* $L : S \rightarrow S_l$ *a inverzní funkce* L' *realizující vazby mezi stavy M a* M_l
3. δ_l *a* λ_l *realizují vícestavové chování* M_l

Příklad 3.4.7 *V tabulce 3.9 je zadán FSM tvořící vícestavovou realizace FSM z Tab. 3.8. Stavy* a *a* c *byly rozděleny na stavy* a1, a2 *a* c1, c2 *(tyto stavy nazýváme indexované). Je zde také dobře vidět, že následující stavy a výstupní symboly jsou vždy podmnožinou původních stavů.*

Vícestavová realizace FSM nám umožňuje realizovat automat reprezentovaný systémem množin a s minimálně jedním stavem ve více blocích jelikož umožňuje rozlišení mezi těmito bloky. Koncept vícestavové realizace je možný zavést i pro úplně zadané automaty, jelikož i při jejich dekompozici lze nalézt místo rozkladů systémy množin a využít jich pro zlepšení dekompozice. Vícestavová realizace nám dáva větší svobodu ve výběru následujícího stavu a toto je pak možné zohlednit při hledání rozkladové trojice.

Pro využití vícestavové realizace je třeba rozšířit množiny stavů S na S_l a výstupních symbolů z S × I na $S_l \times I$ a dále je potřeba definovat funkce pro přechod mezi oběma automaty:

in	ps	ns	out
0	a1	b,c1,c2	00
1	a1	a2	01
0	a2	b,c2	00,11
1	a2	a1	01
0	b	a1,c	11
1	b	b	01
0	c1	a1,a2	10,01
1	c1	b	10,01,11
0	c2	a2	10
1	c2	b	01,11

Tabulka 3.9: Vícestavová realizace neúplně zadaného FSM

Definice 3.4.11 (Indexovaná systémová přechodová funkce)

$$\Delta^{\phi_{S_l}} : 2^{\phi_{S_l \times I}} \rightarrow 2^{\phi_{S_l}} \ a\ \Delta^{\phi_{S_l}}(D) = \{B'_l \in \phi_{S_l} \mid \exists B' \in \overline{\delta}(L'(D)) : B' \subseteq L'(B'_l)$$

Definice 3.4.12 (Indexovaná systémová výstupní funkce)

$$\Lambda^{\phi_O} : 2^{\phi_{S_l \times I}} \rightarrow 2^{\phi_O} \ a\ \Delta^{\phi_0}(D) = \{C | C \in \phi_O \wedge \exists C' \in \overline{\lambda}(L'(D)) : C' \subseteq C\}$$

Definice 3.4.13 (Systémový pár $S_l\times$ I - S_l)

$$\forall D \in \phi_{S_l \times I} \ : \ \Delta^{\phi_{S_l}}(D) \neq 0$$

Definice 3.4.14 (Systémový pár $S_l\times$ I - O)

$$\forall D \in \phi_{S_l \times I} \ : \ \Lambda^{\phi_0}(D) \neq 0$$

Jelikož se v systému nachází dvě množiny stavů S a S_l, je třeba ještě zavést pojem systémového páru S - S_l, zohledňující fakt, že množina S_l může obsahovat jen kopie stavů obsažených v S.

Definice 3.4.15 (Systémový pár S - S_l)

$$\forall D \in \phi_{S_l} \exists B \in \phi_S \ L'(D) \subseteq B$$

Pomocí výše zmíněných definic si můžeme upravit podmínky platné dekompozice úplně zadaného automatu také pro neúplně zadaný podatutomat:

Definice 3.4.16 (Podmínky platnosti obecné dekompozice) *Mějme konečný automat M a rozkladovou trojici tohoto automatu, nechť $\phi^{ij}_{S_l \times I}$ je rozklad takový, že $\phi^{i}_{S_l \times I} \leq \phi^{ij}_{S_l \times I}$ a ${\phi'_{S_l \times I}}^{j} = \prod_{i=1\ldots n} \phi^{ij}_{S_l \times I}$. Nechť ${\phi^{S}_{S_l \times I}}^{i}$ a ${\phi^{I}_{S_l \times I}}^{i}$ jsou systémové páry indukované na množině $S_l \times I$ pomocí rozkladů ϕ^i_S a ϕ^i_I. Nechť $\phi^I_{S_l \times I} = \prod_{i=1\ldots n} {\phi^{I}_{S_l \times I}}^{i}$ a $\phi^S_{S_l \times I} = \prod_{i=1\ldots n} {\phi^{S}_{S_l \times I}}^{i}$. Nechť ${\phi^{in}_{S_l \times I}}^{i} = {\phi^{I}_{S_l \times I}}^{i} \cdot {\phi^{S_l}_{S_l \times I}}^{i} \cdot {\phi'_{S_l \times I}}^{i}$.*

Pokud bude pro každý podautomat existovat mapovací funkce $L^i : \phi^i_S \rightarrow \phi^i_{S_l}$*, množina systémových trojic bude tovřit platnou dekompozici pokud bude splňovat následující podmínky:*

1. $\left([\phi^{in}_{S_l \times I}{}^i]_i, [\phi^i_{S_l}]_i\right)$ *je synchronizovaný systémový pár* $S_l \times I$ - S_l
2. $\phi^{in}_{S_l \times I}{}^i \leq \phi_{S_l \times I}{}^i$
3. $\phi^{S_l}_{S_l \times I} \cdot \phi^I_{S_l \times I} \leq \phi'_{S_l \times I}{}^i$
4. ϕ_O *má specializaci* $\phi_{O'}$ *a* $(\phi_{S_l \times I}, \phi_{O'})$*) je rozkladový pár* $S_l \times I$ - *O'*
5. $(\prod_i \phi^i_{S_l} = \pi_S(0))$ *je* S_l - *S systémový pár*

Vysvětlení výše definovaných pojmů je analogické úplně zadanému automatu, stejně jako úpravy pro paralelní či sériovou dekompozici. Stejně tak 5. podmínka musí být splněna pouze má-li zůstat zachována schopnost rozeznat aktuání stav. Důkaz platnosti tvrzení viz [9].

3.4.2 Příklad dekompozice neúplně zadaného automatu

in	ps	ns	out
00	a	a	00
01	a	a	11,01
10	a	a	11,01
11	a	c	11
00	b	a,b	11
01	b	a	01
10	b	a	01
11	b	c	11
00	c	b	00
01	c	a	11,01
10	c	a	11,01
11	c	c	11
00	d	d	00
01	d	a	11,01
10	d	a	11,01
11	d	c	11
00	e	d,e	11
01	e	a	01
10	e	a	01
11	e	c	11

Tabulka 3.10: Neúplně zadaný FSM - příklad dekompozice

Mějme automat M zadaný tabulkou 3.10 a množinu dvou systémovách trojic se systémy množin nad množinou stavů $\phi^1_S = \{\overline{a,b}; \overline{a,c}; \overline{d,e}\}$ a $\phi^2_S = \{\overline{a,c,d}; \overline{a,b,e}\}$. Je tedy třeba indexovat stav *a*. Potom můžeme namapovat jednotlivé řádky dle tabulky 3.11. Na tento mapovací prostor je možné namapovat i ostatní systémy množin (stejně bylo mapováno i u úplně zadaného automatu), tím se vytvoří indukované systémy množin $\phi^S_{S_l \times I}{}^i$ a $\phi^I_{S_l \times I}{}^i$.

	00	01	10	11
1a	0	1	2	3
1a	4	5	6	7
b	8	9	10	11
c	12	13	14	15
d	16	17	18	19
e	20	21	22	23

Tabulka 3.11: Mapování řádků na symboly

podautomat M_1:

$\phi_I^1 = \{\overline{00}, \overline{01, 10, 11}\} = \{i_1, i_2\}$

$\phi_{S_l}^1 = \{\overline{a1, b}, \overline{a2, c}, \overline{d, e}\} = \{s_1, s_2, s_3\}$

$\phi_{S_l \times I}^1 = \{\overline{0, 8, 16, 20}, \overline{4, 12}, \overline{1, 2, 3, 5, 6, 7, 9, 10, 11, 13, 14, 15, 17, 18, 19, 21, 22, 23}\} = \{y_1, y_2, y_3\}$

podautomat M_2:

$\phi_I^2 = \{\overline{00, 11}, \overline{01, 10}\} = \{j_1, j_2\}$

$\phi_{S_l}^2 = \{\overline{a1, c, d}, \overline{a2, b, e}\} = \{t_1, t_2\}$

$\phi_{S_l \times I}^2 = \{\overline{0, 1, 2, 3, 4, 7, 11, 12, 13, 14, 15, 16, 17, 18, 19, 23},$
$\overline{1, 2, 5, 6, 8, 9, 10, 13, 14, 17, 18, 20, 21, 22}\} = \{z_1, z_2\}$

Konstrukce tohoto dekomponovaného automatu bude složitější o konstrukci indexovací funkce, ve které se však může zredukovat množina vstupních i výstupních symbolů. Jinak se bude postupovat stejně jako v případě úplně zadaného automatu.

Nejprve je opět potřeba zkontrolovat platnost dekompozice. Aby tato množina systémových trojic splnila podmínky platné dekompozice, musí se importovat informace z podautomatu M_1 do M_2, čímž se zvětší sice počet vstupů do M_2, ale na druhou stranu není třeba vytvářet žádný propojovací člen, protože stačí výstup z M_1 přivést na vstup M_2 (méně informace nestačí pro splnění podmínky).

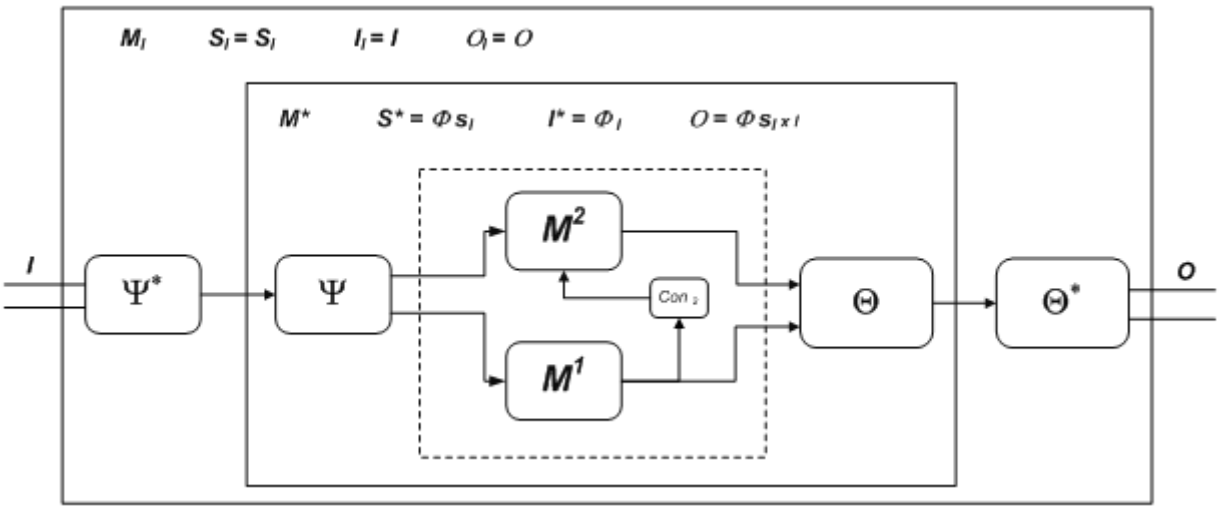
Konstrukce indexovaného automatu M_l

Množina vstupních symbolů může být v tomto automatu redukována na symboly potřebné podautomaty. Tyto lze získat součinem všech systémů množin ϕ_I^i, což je v našem případě následující:

$$\phi_I = \phi_I^1 \cdot \phi_I^2 = \{\overline{00}; \overline{11}; \overline{01, 10}\}$$

Stejně tak je možné upravit množinu stavů a výstupních symbolů. V našem případě množina stavů zůstává beze změn (součin je nulový systém množin) a množina výstupů po vynásobení obou systémů je rovna:

$$\begin{aligned} \phi_{S_l \times I} = \{&\overline{0, 16}(r_1); \overline{8, 20}(r_2); \overline{4, 12}(r_3); \\ &\overline{1, 2, 3, 7, 11, 13, 14, 15, 17, 18, 19, 23}(r_4); \\ &\overline{1, 2, 5, 6, 9, 10, 13, 14, 17, 18, 21, 22}(r_5)\} \end{aligned}$$



Obrázek 3.7: Struktura dekomponovaného FSM s vícestavovou realizací

Pomocí těchto systémů množin jsme potom schopni vytvořit přechodovou tabulku automatu M_I. Pokud při tvorbě tabulky narazíme na stav, který je při indexování rozdělen na dva, je potřeba v množině následujících stavů pro tento řádek uvést oba, resp. všechny tyto indexované stavy jako možné.

Pokud máme již vytvořenou tabulky, můžeme vytvořit přechody mezi původním automatem a podautomaty (struktura dekomponovaného automatu s vícestavovou realizací viz Obr. 3.7). Tyto přechody se opět dají realizovat pomocí logické funkce stejně jako u úplně zadaného automatu.

Konstrukce ostatních částí dekomponovaného automatu

Konstrukce zbývajících částí automatu se provádí prakticky stejně jako u úplně zadaného automatu jen s tím rozdílem, že při vyhodnocování následujících stavů a výstupních symbolů je třeba použít systémovou přechodovou a výstupní funkci definovanou v 3.4.3, resp. v 3.4.7.

4 Implementace dekompozice

V této kapitole bude představena metoda pro nalezení pokud možno co nejefektivnější dekompozice za pomocí nástrojů představených v předchozích částech práce. Nalezení nejideálnější dekompozice je problém obzvláště u rozsáhlých systémů takřka neřešitelný v nějakém rozumném čase. Abychom mohli říci, že nalezená dekompozice je pro daný automat opravdu nejlepší možná, museli bychom vědět, že žádná jiná dekompozice nedosahuje lepších výsledků. Toho lze (prozatím) dosáhnout jen vyzkoušením všech možných kombinací systémových trojic, kterých však i pro malé automaty existuje veliké množství.

Kvalitu dekompozice určuje především rozsáhlost nalezených podautomatů a kvantita jejich vzájemné komunikace, ačkoliv na různé obvody mohou být kladeny rozličné požadavky. Obecně však určujícími parametry, kterými se řídí i zde uváděná metoda, bývá počet bloků v jednotlivých sysémech množin, který určuje počet bitů potřebných pro realizaci těchto systémů množin. Přestože by se mohlo zdát, že nemá cenu dekompozice zachovávající v konečném důsledku např. stejný počet výstupních nebo vstupních bitů, ale vedoucí jen k omezení počtu symbolů (např. máme-li 3bity, které tvoří 8 vstupních symbolů, zredukované na 5 symbolů, pro jejichž zakódování jsou stále potřeba 3 bity), je třeba mít na paměti možnost další dekompozice tohoto podautomatu, která by při menším množství symbolů mohla vést k lepším výsledům.

Z důvodu náročnosti hledání vhodných dekompozic byly vytvořeny heuristiky hledající téměř ideální dekompozice bez nutnosti procházení všech kombinací. Z těchto potom vychází i metoda nalezení obecné dvoustupňové (dekomponovaný automat se skládá ze dvou podautomatů) dekmpozice použitá v této práci. Metoda spočívá v nalezení všech SV systémů množin nad množinou stavů, což je jedna z heuristik používaných pro paralelní a sériové dekompozice. Následně se na základě těchto systémů množin vytvoří ϕ_S^i, ze kterých se poté vytvoří ϕ_I^i. Tyto se také co nejvhodněji sloučí a pomocí obou systémů se vytvoří výstupní systém množin. Ten se upravuje podle informace potřebné k splnění podmínek dekompozice. Pokud by tato dekompozice nevedla k úspěšnému cíli, je potřeba sloučit jiné bloky, popř. vybrat jiné systémy množin.

Dalším faktorem výrazně ovlivňujícím rychlost a kvalitu dekompozice je také kvalita (optimalizace) kódu programu dekompozici provádějícího. Jelikož se jedná o problém prohledávání určitých prostorů a následné práce s nimi, je vhodné použít při realizaci aplikace teorii grafů a nástroje s ní spojené.

4.1 Nalezení SV systémů množin

Pro nalezení všech systémů množin s SV je použita metoda definována v 2.2.14. Příklad hledání takovýchto systémů je popsán v př. 4.1.1. Vycházíme-li z předpokladu, že pokud máme několik různých SV systémů množin, kdy každý může bez pomoci ostatních systémů určit svůj vlastní následující stav, je možné je přímo použít pro paralelní dekompozici popř. pro sériovou dekompozici, pokud bychom měli jeden systém s SV a druhý vytvořili tak, aby oba splňovaly podmínky dekompozice. Pokud máme tedy dva systémy množin s SV a spojíme je dohromady (relace +), výsledný systém už sice nebude mít SV, ale bude potřebovat relativně málo informace od ostatních systémů, aby mohl určit následující stav.

Příklad 4.1.1 *Mějme tabulku 4.1, reprezentující přechody z jednotlivých stavů při různých vstupních symbolech automatu z Tab. 3.1. Potom můžeme vypočítat:*

$$\begin{aligned}
\tau^1_{1,2} &= \{\overline{a,b};\overline{c}\} + \{\overline{a,b};\overline{c}\} = \{\overline{a,b};\overline{c}\} \\
\tau^2_{1,2} &= \{\overline{a,b};\overline{c}\} + \{\overline{a,b};\overline{c}\} = \{\overline{a,b};\overline{c}\} \Rightarrow \textit{SV systém množin} \\
\tau^1_{1,3} &= \{\overline{a,c};\overline{b}\} + \{\overline{a,c};\overline{a,b}\} = \{\overline{a,b};\overline{a,c}\} \\
\tau^2_{1,3} &= \{\overline{a,c};\overline{a,b}\} + \{\overline{a,b};\overline{a,c}\} = \{\overline{a,b};\overline{a,c}\} \Rightarrow \textit{SV systém množin} \\
\tau^1_{2,3} &= \{\overline{b,c};\overline{a}\} + \{\overline{a,c};\overline{b}\} = \{\overline{a,c};\overline{b,c}\} \\
\tau^2_{2,3} &= \{\overline{a,c};\overline{b,c}\} + \{\overline{a,c};\overline{a,b}\} = \{\overline{a,c};\overline{b,c};\overline{b,c}\} \\
\tau^2_{2,3} &= \{\overline{a,c};\overline{b,c}\} + \{\overline{a,c};\overline{a,b}\} = \{\overline{a,c};\overline{b,c};\overline{b,c}\} \\
C(\tau^2_{2,3}) &= \{\overline{a,b,c}\}
\end{aligned}$$

stav	in 0	in 1
a	c	a
b	c	b
c	a	b

Tabulka 4.1: Tabulka pro hledání SV systémů množin

4.2 Hledání vhodných systémů množin

Nalezené SV sytémy množin nad množinou stavů slouží jako základ pro nalezení vhodné dvojice systémových trojic.

Hledání ϕ^1_S a ϕ^2_S

Jako systémy množin nad množinou stavů mohou sloužit již dva libovolně SV systémy bez toho, aby potřebovaly jakékoliv propojení, jelikož sami sobě poskytují dostatek informace k výpočtu následujícího stavu. To ale nemusí být vždy výhodné a lze využít faktu, že informace nutná k výpočtu následujícího stavu podautomatu se dá získat i od druhého podautomatu. Zároveň je však třeba snížit komunikaci mezi podautomaty na miminum.

Metoda zde použitá sčítá a následně aplikuje funkci $C(\phi)$ na všechny možné kombinace nalezených SV systémů množin do té doby, dokud součinem dvou systémů množin nevznikne nový blok, ale dokud se bloky slučují. Dále je třeba si pamatovat nejlepšího dosaženého výsledku a pokud je nově vytvořený systém množin horší, není třeba se jím dále zabývat. Kvalita výsledku je dána velikostí největšího bloku systému a počtem bloků v tomto systémů. Porovnává se součet těchto dvou hodnot a v případě rovnosti také velikost vyšších z těchto hodnot. Pokud je hodnota těchto veličin u nově nalezeného systému množin stejná jako u nejlepšího dosud nalezeného, je třeba s tímto systémem také zacházet jako s možným kandidátem pro dekompozici.

Po nalezení všech nejlepších systémů se vybere takový systém množin, který obsahuje nejvíce relevantní informace. Poté se, pokud je třeba, provede indexace a vytvoří se druhý systém množin (doplňující) tak, aby byla splněna podmínka č. 5 z podmínek platné dekompozice. V tomto novém systému nesmí být v jednom bloku žádné dva stavy takové, které jsou u prvního systému množin také v jednom bloku.

Po skončení této procedury se pomocí níže popsaných metod vytvoří ϕ_I^i a $\phi_{S\times I}^i$. Pokud jsou tyto dvě trojice úspěšné, je vhodné zkusit spojit další bloky těchto rozkladů a otestovat, zda je možné vytvořit platnou dekompozici i z nich. Při slučování je třeba si uvědomit, že je třeba upravit i doplňující systém a i ten kontrolovat, aby se zbytečně nerozrostl. Pokud už nelze žádné bloky spojit, je nalezena finální platná dekompozice.

Hledání ϕ_I^1 a ϕ_I^2

V této fázi již máme ϕ_S. Vstupní systém množin se potom hledá testováním, zda systémy množin ϕ_I^i, kde jsou vždy spojeny dva bloky z $\phi_I(0)$, tvoří se systémy množin z minulého kroku systémové páry I - S. Takto se otestují všechny dvojice vstupních symbolů a pokud u některé dvojice je test úspěšný, mohou se tyto symboly spojit. Pokud je test úspěšný u dvou dvojic symbolů, kde jeden ze symbolů je společný oběma dvojicím, mohou se sloučit jen pokud je test úspešný i pro druhé vstupní symboly.

Takto by se však mohlo postupovat, pokud by se pracovalo jen s SV systémy množin a neimportovala se žádná další informace. Po otestování všech dvojic a případném poslučování všech možných systémů množin, vyberou se opět dva nejslibnější a s těmi se vytváří $\phi_{S\times I}^i$. Pokud je dekompozice platná, opět se zkouší sloučit další bloky a opět ověřovat platnost dekompozice.

Hledání $\phi_{S\times I}^1$ a $\phi_{S\times I}^2$

Jak vyplývá z označení, systém množin $\phi_{S\times I}$ vznikne součinem systémů množin nad stavy a vstupními symboly, popř. ještě importované informace.

Po vytvoření tzv. základního systému množin $\phi_{S\times I}$ je třeba zkontrolovat, jestli splňuje 1. podmínku platné dekompozice. Pokud ne, je třeba zkontrolovat, zda je možné získat potřebnou informaci z druhého podautomatu (musí být možné ji získat jen ze součinu ϕ_S^i a ϕ_I^i). Pokud to možné není, dekompozice nemůže být platná a musí se vybrat jiné systémy množin.

Pokud však je možné informaci získat, popř. pokud není žádná třeba, je třeba otestovat zda platí podmínka č. 4. Pokud neplatí opět nemůže existovay platná dekompozice této dvojice podautomatů. Pokud ano, dekompozice bude platná, a je možné se stejně jako u předchozích případů pokusit se spojit další bloky $\phi_{S\times I}^i$ tak, aby stále byla splněna podmínka č. 1.

Hledání optimálního množství importované informace

Pokud je potřeba importovat nějakou informaci, je vhodné importovat jen minimální množství této informace. Proto se sloučí ty bloky výstupního systému množin, které nenesou relevantní informaci. Maximální možná importovaná informace je pak dána součinem vstupních a stavových systémů množin obou podautomatů. Pokud tedy není požadovaná informace v tomto součinu, je třeba zvolit jiné systémy množin.

Příklad hledání systémové trojice

Pro příklad procesu hledání vhodných trojic zde použiji dekompozici uvedenou v sekci 3.4.2. Nejprve si vytvoříme tabulku přechodů (viz př. 4.1.1 a viz Tab. 4.2).

Pomocí metody nalezení všech SV systémů množin získáme:

$$\begin{aligned}
\phi_{1,2} &= \{\overline{a,b};\overline{d,e};\overline{c}\} \\
\phi_{1,3} &= \{\overline{a,b};\overline{a,c};\overline{d,e}\} \\
\phi_{1,4} &= \{\overline{a,b};\overline{a,d};\overline{d,e};\overline{c}\} \\
\phi_{1,5} &= \{\overline{a,d,e};\overline{a,b};\overline{c}\} \\
\phi_{2,3} &= \{\overline{a,b};\overline{b,c};\overline{d,e}\} \\
\phi_{2,4} &= \{\overline{a,b,d};\overline{d,e};\overline{c}\} \\
\phi_{2,5} &= \{\overline{a,b,d,e};\overline{c}\} \\
\phi_{3,4} &= \{\overline{a,b,d,};\overline{c,d};\overline{d,e}\} \\
\phi_{3,5} &= \{\overline{a,b,d,e};\overline{c,e}\} \\
\phi_{4,5} &= \{\overline{a,b};\overline{d,e};\overline{c}\}
\end{aligned}$$

Kdybychom vybrali systém množin $\phi_{2,5}$, který má jen dva bloky. Je na první pohled vidět, že pro splnění 5. podmínky je třeba rozlišit mezi stavy *a,b,d,e*, z čehož plyne, že systém rozkladů bude mít 4 bloky (např (a,c),(b),(d),(e)). Pro 4 bloky je zpotřebí 2 bitů, kdežto pro zakódování $\phi_{2,5}$ jen jeden, proto se pokusíme takto vyšetřit i ostatní systémy množin.

stav	00	01	10	11
a	a	a	a	c
b	a,b	a	a	c
c	b	a	a	c
d	d	a	a	c
e	d,e	a	a	c

Tabulka 4.2: Tabulka přechodů pro FSM zadaný tab. 3.10

Jako vhodný pár byly tedy nakonec vybrány $\phi_S^1 = \{\overline{a1,b};\overline{a2,c};\overline{d,e}\}$ a $\phi_S^2 = \{\overline{a1,c,d};\overline{a2,b,e}\}$. Podobným způsobem se potom hledají ϕ_I. Opět použijeme tabulku 4.2 (první řádek lze považovat za oba indexované stavy a indexy i a j označují pořadí vstupního symbolu dle hodnoty), tentokrát však procházíme dvojice vstupů a porovnáváme množiny stavů podle ϕ_S. Tímto způsobem dostaneme pomocí ϕ_S^1:

$$\begin{aligned}
\phi_{1,2} &= \{\overline{a,d,e};\overline{a,b}\} \\
\phi_{1,3} &= \{\overline{a,d,e};\overline{a,b}\} \\
\phi_{1,4} &= \{\overline{a,b,c};\overline{d,c,e}\} \\
\phi_{2,3} &= \{\overline{a,b,c,d,e}\} \\
\phi_{2,4} &= \{\overline{a,c}\} \\
\phi_{3,4} &= \{\overline{a,c}\}
\end{aligned}$$

Z tohoto můžeme poznat, že bez obavy ze ztráty potřebné informace můžeme spojit bloky vstupních symbolů 2,3 a 4, resp. 01,10 a 11, jelikož kombinace těchto symbolů vždy vedou do nějakého bloku ϕ_S, kdežto kombinace s prvním symbolem 00 nikoliv. Lze také zkusit spojit bloky, které nelze bezpečně spojit, potom se ale musí informace potřebná k rozlišení stavů neležících v jednom bloku získat z druhého podautomatu.

Zbývající systémy množin $\phi^1_{S\times I}$ a $\phi^2_{S\times I}$ se získají součinem v předchozích krocích vytvořených systémů množin. Jako kontrolu je třeba vyzkoušet, zda tyto systémy množin tvoří S $\times$ I - S rozkladové páry (platí 1. podmínka). V tomto případě zjistíme, že systém množin $\phi^2_{S\times I}$ potřebuje dodatečnou informaci pro splnění podmínky. Tuto informaci snadno zjistíme nalezením bloků, které nevyhovují podmínce a jejich následným rozkladem a určením informace, která tímto rozkladem vznikla. Tato informace je dostupná z $\phi^1_{S\times I}$, lze ji proto importovat a dekompozice je platná.

Pokud platí 1. podmínka, je třeba otestovat, zda platí 4. podmínka, tj. že se dá jednoznačně spočítat výstupní symbol. Pokud je i toto splněno, lze zkoušet spojovat různé kombinace bloků pro snížení počtu výstupních symbolů a tím pádem zlepšení dekompozice. je však nutné kontrolovat, zda výstupní systém množin stále obsahuje informaci potřebnou pro druhý podautomat.

4.3 Implementace metody

Jak již bylo řečeno výše, úspěšnost celé metody je závislá na optimalizaci a kvalitě zdrojového kódu aplikace. Na jednu stranu to klade vysoké nároky na autora kódu, na druhou stranu to umožňuje značně ovlivňovat rychlost a kvalitu dekompozice bez úpravy metody jen s pomocí optimalizace kódu.

Program realizující uvedenou metodu je napsán v jazyce C++. Při spouštění programu je třeba zadat cestu ke zdrojovému souboru ve formátu KISS 2 (popsán např. v [3]). Program se skládá z těchto modulů:

Main

Modul *main* obsahuje základní části programu, kontroluje správnost parametrů a volá jednotlivé obsluhující metody.

Bit

Třída *Bit* popsaná v modulu *bit* reprezentuje pole bitů. Ačkoliv standardní knihovna C++ obsahuje dva nástroje pro práci s bity, pro potřebu této práce jsou tyto metody nedostatečné, jelikož pracují pouze s hodnotami *0* nebo *1* a není zde označení pro *don't care* stav. Samotná třída Bit potom obsahuje také nástroje pro porovnávání bitů a hlavně úpravu dvou vektorů bitů tak, aby pokud obsahují don't care stavy a umožňují takovou úpravu byly shodné.

FSM

FSM je modul obsahující třídu, která reprezentuje konečný automat. Mimo této také obsahuje třídu *Symbol*, který reprezentuje řádek a obsahuje čtyři položky reprezentující vstupní a výstupní symbol objektem typu *Bit*, vnitřní stav a následující stav datovým typem *int*.

Modul obsahuje funkce pro práci s automatem jako přidání řádku, zjištění velikosti automatu apod. Hlavně se ale stará o soudržnost všech údajů o automatu dostupných (zapouzdřuje tato data).

Lexan a Synan

Modul *lexan* obsahuje lexikální analyzátor, který čte znaky vstupního souboru. *Synan* obsahuje syntaktický analyzátor, který tyto znaky zpracovává a přidává řádky do objektu typu *FSM*.

Information

Modul *information* obsahuje třídu Inf reprezentující základní informaci. Třída tedy obsahuje dva celočíselné stavy a nástroje pro práci s ní. V dalších modulech jsou potom objekty typu Inf udržovány v kontejneru *std::set*, který umožňuje rychlé vyhledávání a je tedy vhodný pro uchování velkého množství informace, často potřebné v průběhu procesu dekompozice.

Block

Block obsahuje předpis pro vytvoření objetku Block. Obsahuje pole celočíselných hodnot představující jednotlivé řádky automatu, které jsou v bloku uloženy a identifikátor reprezentující buď vstupní a výstupní symboly pomocí objektu typu *Bit* nebo stavy pomocí objektu *State*, definovaný v modulu *Block* a reprezentující číslo stavu, popř. jeho index.

Part

Modul *part* obsahuje třídu reprezentující systém množin buď nad stavy nebo nad bity. Bloky jsou zde uloženy v kontejneru *std::vector*. Třída obsahuje také nástroje potřebné pro snadnou práci se systémy množin, jako nalezení bloku, velikost (počet bloků) systému apod.

Decomposition

Modul *decomposition* obsahuje veškeré nástroje realizující dekompozici metodou popsanou výše. V tomto modulu se hledají systémové trojice, testuje se platnost dekompozice, hledají SV systémy a pod.

Output

V tomto modulu se vezmou výsledné systémy množin a sestaví se z nich podautomaty a logické funkce potřebné pro dekompozici. Také obsahuje metodu vypisující tyto části do souborů typu KISS 2.

Výstupem programu a tedy celého procesu dekompozice jsou především dva soubory ve formátu KISS 2 *aut1* a *aut2*, představující oba podautomaty. Dále jsou součástí výstupu soubory s funkcemi představujícími vstupní a výstupní blok *in_bl*, resp. *out_bl*. Pokud jsou podautomaty propojené, jsou součástí výstupu také funkce realizující toto propojení *con1* a *con2*.

5 Závěr

V této práci byly shrnuty základní znalosti o dekompozici sekvenčního obvodu. Byly zde popsány, popř. zhodnoceny obecně používané metody. Navíc byla vybrána jedna metoda (úplná obecná dekompozice sekvenčního obvodu se společnou realizací přechodové a výstupní funkce), která byla rozebrána podrobně a navíc byla popsána možnost její implementace.

Z výše uvedeného vyplývá, že vybraný způsob dekompozice je nejuniverzálnější metodou vůbec a lze ji aplikovat na kterémkoliv sekvenčním obvodě, logické funkci nebo relaci. To však ještě neznamená, že by měla vždy nejoptimálnější výsledky. Velmi zde záleží na platformě na které se obvod implementuje, na metodě hledání systémů množin a dalších aspektech.

Největším problémem všech typů dekompozice nadále zůstává složitost hledání optimálních řešení. Ačkoliv různé metody, mezi něž patří i zde uvedená, mohou dosahovat uspokojivých výsledků, s vývojem digitálních obvodů a zvyšováním jejich složitosti i tyto mohou časem přestat vyhovovat a je třeba nadále hledat optimálnější řešení (dekompozice už dnes mohou trvat i několik dnů či týdnů). Ideální metoda by měla najít nejlepší dekompozici pomocí jednoznačně určeného postupu a bez nutnosti omezovat oblast hledání kvůli časové náročnosti. Možnost kvantifikovat množství informace v této prácí popsané by mohlo být základem k takovéto metodě. Je však třeba vyřešit otázku výběru spojovaných bloků bez nutnosti dalších ověřování a testování, popř. vracení se k jiným spojením.

Nejnáročnější částí práce byla tvorba metody pro hledání vhodné systémové trojice. Pro dosažení lepších výsledků aplikace by potom bylo třeba především optimalizovat kód popř. implmentovat vhodnější algoritmy především pro výpočty a porovnávání systémů množin, kdy je třeba často porovnávat velké množství.

Aplikace dekompozice sama o sobě však nemá příliš velkého významu bez vhodného zakódování stavů. To je také oblast s největšími možnostmi využití dekompozice a tam také směřuje výzkum této problematiky (viz [11]). Právě tímto směrem by se měla do budoucna ubírat i tato práce, kde by se mohla uplatnit metoda zde uvedena, popř. by mohla být upravena pro požadavky specifické pro danou problematiku.

A Literatura

[1] Allen H. Curtis: A generalized tree circuits. *Journal of The Association for Computing Machinery*, 8:484–496, 1961.

[2] A. D. Friedman and P. R. Menon: *Teorie a návrh logických obvodů*. SNTL, Praha, 1983.

[3] T. Hrdý: Vliv dekompozice sekvenčního obvodu na jeho implementaci. *Diplomová práce*, 2001.

[4] Lech Jóźwiak: The full decomposition of sequential machines with the output behaviour realization. *EUT Report 88-E-199*, March 1988.

[5] Lech Jóźwiak: General decomposition and its use in digital circuits synthesis. *VLSI Design*, 3:225–248, 1995.

[6] Lech Jóźwiak: Information relationship and measures - an analysis apparatus for efficient information system synthesis. *23rd EUROMICRO Conference '97 New Frontiers of Information Technology*, pages 13–23, 1997.

[7] Lech Jóźwiak: Information relationship measures in application in logic design. *Proc. 29th IEEE International Symposium on Multiple Valued Logic*, pages 228–235, 1999.

[8] Lech Jóźwiak and Chojnacki A: Effective and efficient fpga synthesis through functional decomposition based on information relationship measures. *Proc. 29th IEEE International Symposium on Multiple Valued Logic*, pages 228–235, 1999.

[9] Lech Jóźwiak and A. S. Ślusarczyk: General decomposition of incompletely specified sequential machines with multi-state behavior realization. *Journal of System Architecture on Reconfigurable Computing*, 2004.

[10] V. Pop: Dekompozice konečného automatu. *Bakalářská práce*, Srpen 1998.

[11] A. S. Ślusarczyk: Decomposition and encoding of finite state machine for fpga implementation. *Disertační práce*, 2004.

[12] J. Studenovský: Kódování vnitřních stavů synchronního sekvenčního obvodu. *Diplomová práce*, 1999.

B Seznam použitých zkratek

FSM Final-State Machine - konečný stavový automat

FPGA Field-Programmable Gate Array - pole programovatelných hradel

IRMA Information Relationship Measures Apparatus - aparát pro urční míry informace

IS Information Set - informační sada

CI Common Information - společná informace

TI Total Information - celková informace

MI Missing Information - chybějící informace

EI Extra Information - přidaná informace

DI Different Information - rozdílná informace

ISIM Information Similarity - schodnost informace

IDIS Information Difference - rozdílnost informace

IDEC Decrase Information - ztrátovost informace

IINC Information Increase - zisk informace

TIQ Total Information Quantity - celkové množství informace

LUT Look-Up Table - vyhledávací tabulka

PAL Programmable Logic Arrat - programovatelné logické pole