5.2 Asociační pravidla

IF-THEN konstrukce nalezneme ve všech programovacích jazycích, používají se i v běžné mluvě (nebude-li pršet, nezmoknem). Není tedy divu, že *pravidla* s touto syntaxí patří společně s rozhodovacími stromy k nejčastěji používaným prostředkům pro reprezentaci znalostí, ať už získaných od expertů, nebo vytvořených automatizovaně z dat.

Termín *asociační pravidla* široce zpopularizoval počátkem 90. let Agrawal [Agrawal a kol, 1993] v souvislosti s analýzou nákupního košíku. Při této analýze se zjišťuje, jaké druhy zboží si současně kupují zákazníci v supermarketech (např. pivo a párek). Jde tedy o hledání vzájemných vazeb (asociací) mezi různými položkami sortimentu prodejny. Přitom není upřednostňován žádný speciální druh zboží jako závěr pravidla. V tomto smyslu budeme chápat pravidla v této kapitole. O použití IF-THEN pravidel pro klasifikaci pojednává následující kapitola [1].

5.2.1 Základní charakteristiky pravidel

U pravidel vytvořených z dat nás obvykle zajímá, kolik příkladů splňuje předpoklad a kolik závěr pravidla, kolik příkladů splňuje předpoklad i závěr současně, kolik příkladů splňuje předpoklad a nesplňuje závěr…. Tedy, zajímá nás, jak pro pravidlo

$$\mathrm{Ant} \Rightarrow \mathrm{Suc},$$

kde *Ant* (předpoklad, levá strana pravidla, antecedent) i *Suc* (závěr, pravá strana pravidla, sukcedent) jsou kombinace kategorií [2] vypadá příslušná kontingenční tabulka. Pro *n* příkladů je její podoba uvedena v Tab. 1.

	Suc	**¬Suc**	**Σ**
Ant	a	b	r
¬Ant	c	d	s
Σ	k	l	n

Tab. 1 (Čtyřpolní) Kontingenční tabulka

kde $n(\mathrm{Ant} \wedge \mathrm{Suc}) = a$ je počet objektů[3] pokrytých současně předpokladem i závěrem,
$n(\mathrm{Ant} \wedge \neg\mathrm{Suc}) = b$ je počet objektů pokrytých předpokladem a nepokrytých závěrem,
$n(\neg\mathrm{Ant} \wedge \mathrm{Ssuc}) = c$ je počet příkladů nepokrytých předpokladem ale pokrytých závěrem,
$n(\neg\mathrm{Ant} \wedge \neg\mathrm{Suc}) = d$ je počet příkladů nepokrytých ani předpokladem ani závěrem.

$n(\mathrm{Ant}) = a+b = r$, $n(\neg\mathrm{Ant}) = c+d = s$, $n(\mathrm{Suc}) = a+c = k$, $n(\neg\mathrm{Suc}) = b+d = l$, $n = a+b+c+d$

Z těchto čísel (místo o počtu objektů pokrytých kombinací se někdy mluví o *četnosti* resp. *frekvenci* kombinace) můžeme počítat různé charakteristiky pravidel a kvantitativně tak hodnotit nalezené znalosti.

[1] Při použití pravidel pro klasifikaci je závěr pravidel vyhrazen pro cílový atribut určující zařazení do třídy.

[2] Numerické atributy se vždy musí diskretizovat před použitím algoritmu pro hledání asociačních pravidel.

[3] Někdy může být k objektům přiřazena „váha“ určující, kolikrát se má objekt počítat při výpočtu hodnot v tabulce. Váha vyjadřuje např. počet opakování objektu (při propojování tabulek) nebo to, že jde o částečný objekt vzniklý diskretizací nebo ošetřením chybějících hodnot (podrobnosti viz kapitola o předzpracování dat).

Základními charakteristikami asociačních pravidel v Agrawalově pojetí jsou *podpora* (*support*) a *spolehlivost* (*confidence*). Podpora je (absolutní resp. relativní[4]) počet objektů, splňujících předpoklad i závěr, tedy hodnota

$$a \qquad \text{resp.} \qquad P(Ant \wedge Suc) = \frac{a}{a + b + c + d}.$$

Spolehlivost (též nazývaná platnost – validity, konsistence – consistency, nebo správnost – accuracy) je vlastně podmíněná pravděpodobnost závěru pokud platí předpoklad, tedy

$$P(Suc \mid Ant) = \frac{a}{a + b}.$$

Další důležité charakteristiky jsou:

- absolutní resp. relativní počet objektů, které splňují předpoklad

$$a + b \quad \text{resp.} \qquad P(Ant) = \frac{a + b}{a + b + c + d}$$

- absolutní resp. relativní počet objektů, které splňují závěr

$$a + c \quad \text{resp.} \qquad P(Suc) = \frac{a + c}{a + b + c + d}$$

- *pokrytí* [5] *(coverage*), tj, podmíněná pravděpodobnost předpokladu pokud platí závěr

$$P(Ant \mid Suc) = \frac{a}{a + c}.$$

- *kvalita*, tj. vážený součet [6] spolehlivosti a pokrytí

$$\text{Kvalita} = w_1 \frac{a}{a + b} + w_2 \frac{a}{a + c}$$

  kde w_1 a w_2 se obvykle volí tak [7], aby $w_1 + w_2 = 1$, tedy např. $w_1 = 0.5$ a $w_2 = 0.5$ nebo $w_1 = 0.8$ a $w_2 = 0.2$.

Na základě platnosti a pokrytí lze dělit implikace do několika skupin [Holsheimer, Siebes, 1994]:

- *konzistentní pravidla* jsou pravidla s platností rovnou 1, levá strana implikace je postačující podmínkou pro splnění pravé strany,
- *úplná pravidla* jsou pravidla s pokrytím rovným 1, levá strana implikace je nutnou podmínkou pro splnění pravé strany,
- *deterministická pravidla* jsou pravidla s platností i pokrytím rovným 1, levá strana je nutnou a postačující podmínkou pro splnění pravé strany.

---

[4] Relativní četnosti kombinace K budeme chápat jako odhad pravděpodobnosti jejího výskytu v datech $P(K)$.

[5] Někteří autoři této charakteristice říkají úplnost (completeness).

[6] Přehled dalších způsobů, jak spočítat kvalitu pravidla lze nalézt v [Bruha, 1994].

[7] Zajímavou variantu tohoto přístupu uvádí Torgo ([Torgo, 1993]): $w_1 = 0.5 + \frac{1}{4} P(Suc \mid Ant)$ a $w_2 = 0.5 - \frac{1}{4} P(Ant \mid Suc)$.

V literatuře můžeme nalézt i další charakteristiky asociačních pravidel. Tak např. Kodratoff [Kodratoff, 1999] kromě výše zmíněné podpory *P(Ant∧Suc)*, kterou nazývá *popisn*á (descriptive support) a výše zmíněné spolehlivosti *P(Suc | Ant)*, kterou nazývá *popisn*á (descriptive confidence) uvádí ještě:

- kauzální podporu (causal support)

$$P(Ant \wedge Suc) + P(\neg Ant \wedge \neg Suc) = \frac{a + d}{a + b + c + d},$$

- kauzální spolehlivost (causal confidence)

$$\frac{1}{2} P(Suc \mid Ant) + \frac{1}{2} P(\neg Ant \mid \neg Suc) = \frac{1}{2}\frac{a}{a + b} + \frac{1}{2}\frac{d}{b + d},$$

- deskriptivní potvrzení (descriptive confirmation)

$$P(Ant \wedge Suc) - P(Ant \wedge \neg Suc) = \frac{a - b}{a + b + c + d},$$

- kauzální potvrzení (causal confirmation)

$$P(Ant \wedge Suc) + P(\neg Ant \wedge \neg Suc) - 2\,P(Ant \wedge \neg Suc) = \frac{a + d - 2b}{a + b + c + d},$$

- ujištění (conviction)

$$\frac{P(Ant) * P(\neg Suc)}{P(Ant \wedge \neg Suc)} = \frac{(a + b) * (b + d)}{d * (a + b + c + d)},$$

- zajímavost (interestingness)

$$\frac{P(Ant \wedge Suc)}{P(Ant) * P(Suc)} = \frac{a * (a + b + c + d)}{(a + b) * (a + c)},$$

- závislost (dependency)[8]

$$P(Suc \mid Ant) - P(Suc) = \frac{a}{a + b} - \frac{a + c}{a + b + c + d}.$$

Ze čtyřpolní tabulky tedy můžeme spočítat nejrůznější charakteristiky asociačního pravidla. Další vzorce počítané ze čtyřpolní tabulky, které ale budou tentokráte vyjadřovat různé typy pravidel, uvidíme v souvislosti s metodou *GUHA* (viz dále).

[8] Připomeňme ze statistiky, že pokud *Suc* a *Ant* jsou nezávislé, platí P(Suc | Ant) = P(Suc).

5.2.2 Generování kombinací

Základem všech algoritmů pro hledání asociačních pravidel je generování kombinací (konjunkcí) hodnot atributů. Při generování vlastně procházíme (prohledáváme) prostor všech přípustných konjunkcí [9]. Metod je několik:

- Do šířky
- Do hloubky
- Heuristicky

Jednotlivé metody budeme ilustrovat na příkladě vytváření konjunkcí z dat uvedených v Tab. 2. Pro zjednodušení zápisu kategorií v Obr. 1, Obr. 2 a Obr. 3 budeme kategorii zapisovat pořadovým číslem atributu a prvním písmenem hodnoty, tedy zápis *1v* bude znamenat kategorii *příjem(vysoký).*

klient	příjem	Konto	pohlaví	nezaměstnaný	úvěr
k1	vysoký	vysoké	žena	ne	ano
k2	vysoký	vysoké	muž	ne	ano
k3	nízký	nízké	muž	ne	ne
k4	nízký	vysoké	žena	ano	ano
k5	nízký	vysoké	muž	ano	ano
k6	nízký	nízké	žena	ano	ne
k7	vysoký	nízké	muž	ne	ano
k8	vysoký	nízké	žena	ano	ano
k9	nízký	střední	muž	ano	ne
k10	vysoký	střední	žena	ne	ano
k11	nízký	střední	žena	ano	ne
k12	nízký	střední	muž	ne	ano

Tab. 2 Data pro generování kombinací

Při generování *do šířky* se kombinace generují tak, že se nejprve vygenerují všechny kombinace délky jedna, pak všechny kombinace délky dvě, atd. Jde tedy o generování kombinací *podle délek,* kategorie jednoho atributu jsou přitom uspořádány podle abecedy (Obr. 1).

č.	četnost	Kombinace
1	7.00	1n
2	5.00	1v
3	4.00	2n
4	4.00	2s
5	4.00	2v
6	6.00	3m
7	6.00	3z
8	6.00	4a
9	6.00	4n
10	8.00	5a
11	4.00	5n
12	2.00	1n 2n

[9] V konjunkci se nesmí opakovat atributy.

13	3.00	1n 2s
14	2.00	1n 2v
15	4.00	1n 3m
16	3.00	1n 3z
17	5.00	1n 4a
18	2.00	1n 4n
19	3.00	1n 5a
20	4.00	1n 5n
21	2.00	1v 2n
22	1.00	1v 2s
23	2.00	1v 2v
24	2.00	1v 3m
323	*0.00*	*1v 2v 3z 4n 5n*

Obr. 1 Generování "do šířky"

Při generování *do hloubky* se vyjde od první kombinace délky jedna a ta se pak prodlužuje (vždy o první kategorii dalšího atributu) dokud to lze[10]. Nelze-li kombinaci prodloužit, změní se kategorie „posledního“ atributu. Nelze-li provést ani to (vyčerpaly se kategorie posledního atributu), kombinace se zkrátí a současně se změní poslední kategorie. Princip generování je ilustrován na Obr. 2. Kategorie jednoho atributu jsou opět uspořádány podle abecedy, kurzívou jsou vyznačeny kombinace s nulovou četností.

č.	Četnost	kombinace
1	7.00	1n
2	2.00	1n 2n
3	1.00	1n 2n 3m
4	*0.00*	*1n 2n 3m 4a*
5	*0.00*	*1n 2n 3m 4a 5a*
6	*0.00*	*1n 2n 3m 4a 5n*
7	1.00	1n 2n 3m 4n
8	*0.00*	*1n 2n 3m 4n 5a*
9	1.00	1n 2n 3m 4n 5n
10	*0.00*	*1n 2n 3m 5a*
11	1.00	1n 2n 3m 5n
12	1.00	1n 2n 3z
13	1.00	1n 2n 3z 4a
14	*0.00*	*1n 2n 3z 4a 5a*
15	1.00	1n 2n 3z 4a 5n
16	*0.00*	*1n 2n 3z 4n*
17	*0.00*	*1n 2n 3z 4n 5a*
18	*0.00*	*1n 2n 3z 4n 5n*
19	*0.00*	*1n 2n 3z 5a*
20	1.00	1n 2n 3z 5n
21	1.00	1n 2n 4a
22	*0.00*	*1n 2n 4a 5a*
23	1.00	1n 2n 4a 5n
24	1.00	1n 2n 4n
323	4.00	5n

Obr. 2 Generování "do hloubky"

[10] Prodlužování skončí při dosažení maximální zadané délky konjunkce nebo při vyčerpání atributů a jejich hodnot.

Oba výše zmíněné způsoby jsou „slepé“. Provádějí se pouze na základě seznamu hodnot atributů a neberou do úvahy vlastní data. Proto můžeme vygenerovat kombinace, které se nevyskytují v datech. V seznamu kombinací na Obr. 1 a Obr. 2, jsou takové kombinace zvýrazněny kursivou. Poslední zde uváděný způsob *generování podle četností* vytváří kombinace v pořadí podle jejich výskytu v datech. Jedná se o příklad heuristického prohledávání prostoru kombinací, kde heuristikou je „uvažuj kombinaci s nejvyšší četností". Při tomto způsobu generování se kombinace s nulovou četnosti objeví až na konci seznamu (Obr. 3).

č.	Četnost	kombinace
1	8.00	5a
2	7.00	1n
3	6.00	3m
4	6.00	3z
5	6.00	4a
6	6.00	4n
7	5.00	1v
8	5.00	1n 4a
9	5.00	4n 5a
10	5.00	1v 5a
11	4.00	2v
12	4.00	2s
13	4.00	2n
14	4.00	5n
15	4.00	3m 5a
16	4.00	1n 3m
17	4.00	3z 5a
18	4.00	3z 4a
19	4.00	3m 4n
20	4.00	1v 4n
21	4.00	2v 5a
22	4.00	1n 5n
23	4.00	1v 4n 5a
24	3.00	1n 5a
203	1.00	1v 2s 3z 4n 5a
	0.00	. . .
323	*0.00*	*1v 2n 3z 4n 5n*

Obr. 3 Generování podle četností

5.2.3 Počet kombinací

Generování kombinací je výpočetně náročný proces. S růstem počtu atributů značně roste počet možných konjunkcí. Odhad tohoto počtu nám umožní následující úvahy.

Jak velký je prostor kombinací, který se prohledává? Označme K_{A1}, K_{A2}, ..., K_{Am} počet kategorií atributu A_1, A_2, ..., A_m (m je počet atributů, ze kterých vytváříme kombinace). Pak

počet kombinací délky jedna je $\sum_{i=1}^{m} K_{Ai}$

počet kombinací délky dvě je $\sum_{i,j=1,\, i\neq j}^{m} (K_{Ai} \times K_{Aj})$

počet kombinací délky tři je $\sum_{i,j,k=1,\, i\neq j\neq k}^{m} (K_{Ai} \times K_{Aj} \times K_{Ak})$

. . .

Počet všech kombinací je pak [11]

$$\prod_{j=1}^{m} (1 + K_{Aj}) \; - \; 1$$

Pro data uvedená v Tab. 2 je podle tohoto vzorce počet kombinací délky 1 roven 11, počet kombinací délky 2 roven 48, počet kombinací délky 3 roven 104, počet kombinací délky 4 roven 112 a počet kombinací délky 5 roven 48. Celkový počet všech možných kombinací je tedy 323.

Výše uvedené úvahy se týkaly počtu všech možných kombinací bez ohledu na řídící parametry generování a na vstupní data. Počet generovaných kombinací samozřejmě klesne, požadujeme-li na výstupu pouze kombinace do určité délky[12] a s určitou minimální četností. Omezení dané vstupními daty se týká počtu kombinací, které budou mít nenulovou četnost. Řada (syntakticky) možných kombinací se v daných datech vůbec nemusí vyskytovat. To je i případ kombinací generovaných v předcházející podkapitole. Místo 323 syntakticky správných kombinací byly pro data z Tab. 2 nalezeny jen 203 kombinace s nenulovou četností.

Přes toto další redukování prohledávaného prostoru lze říci, že počet generovaných (a testovaných) kombinací je *exponenciálně závislý na počtu atributů*.

5.2.4 Algoritmus apriori

Nejznámějším algoritmem pro hledání asociačních pravidel je algoritmus *apriori*. Tento algoritmus navrhl R. Agrawal v souvislosti s analýzou nákupního košíku [Agrawal a kol., 1996]. Jádrem algoritmu je hledání často se opakujících množin položek (frequent itemsets). Jedná se kombinace (konjunkce) kategorií[13] které dosahují předem zadané četnosti (podpory *minsup*) v datech.

Při hledání kombinací délky k, které mají vysokou četnost se využívá toho, že již známe kombinace délky $k-1$. Při vytváření kombinace délky k spojujeme kombinace délky $k-1$. Jde tedy o generování kombinací „do šířky“. Přitom pro vytvoření jedné kombinace délky k požadujeme, aby všechny její podkombinace délky $k-1$ splňovaly požadavek na četnost. Tedy např. ze tříčlenných kombinací $\{A_1A_2A_3, A_1A_2A_4, A_1A_3A_4, A_1A_3A_5, A_2A_3A_4\}$ dosahujících požadované četnosti vytvoříme pouze

[11] Tento vztah byl již uveden v obecné kapitole o strojovém učení. Ve zjednodušeném případě, kdy je počet hodnot stejný pro všechny atributy (označme K), je počet kombinací délky jedna roven $K*N$, počet kombinací délky dva roven $K^2 * N*(N-1)/2$, počet kombinací délky tři roven $K^3 * N*(N-1)*(N-2)/6$. Počet všech kombinací je pak podle binomické věty $(1 + K)^N - 1$.

[12] Obvykle se kombinace generují od kratších délek.

[13] V případě analýzy nákupního košíku jsou kategorie typu *máslo(ano), chleba(ano)* apod., které můžeme stručněji zapisovat *máslo, chleba* a chápat jako položky zboží.

jedinou čtyřčlennou kombinaci $A_1A_2A_3A_4$. Kombinaci $A_1A_3A_4A_5$ sice lze vytvořit spojením $A_1A_3A_4$ a $A_1A_3A_5$, ale mezi tříčlennými kombinacemi chybí $A_1A_4A_5$ i $A_3A_4A_5$. Příslušný algoritmus je uveden na Obr. 4.

Algoritmus apriori

1. do L_1 přiřaď všechny kategorie, které dosahují alespoň požadované četnosti
2. polož $k=2$
3. dokud $L_{k-1} \neq \varnothing$
   3.1. pomocí funkce *apriori-gen* vygeneruj na základě L_{k-1} množinu kandidátů C_k
   3.2. do L_k zařaď ty kombinace z C_k, které dosáhly alespoň požadovanou četnost
   3.3. zvětš počítadlo k

Funkce apriori-gen(L_{k-1})

1. pro všechny dvojce kombinací $Comb_p$, $Comb_q$ z L_{k-1}
   1.1. pokud $Comb_p$ a $Comb_q$ se shodují v $k-2$ kategoriích přidej $Comb_p \curlywedge Comb_q$ do C_k
2. pro každou kombinaci $Comb$ z C_k
   2.1. pokud některá z jejich podkombinací délky $k-1$ není obsažena v L_{k-1} odstraň $Comb$ z C_k

Obr. 4 Algoritmus apriori

Po nalezení kombinací které vyhovují svou četností se vytvářejí asociační pravidla. Každá kombinace *Comb* se rozdělí na všechny možné dvojce podkombinací *Ant* a *Suc* takové, že $Suc = Comb - Ant$ [14]. Uvažované pravidlo $Ant \Rightarrow Suc$ má pak podporu, která se rovná četnosti kombinace *Comb*. Spolehlivost pravidla se spočítá jako podíl četností kombinací *Comb* a *Ant*. Četnost kombinace *Ant* přitom již známe. Vzhledem k tomu, že délka *Ant* je menší než délka *Comb*, byla kombinace *Ant* algoritmem apriori vygenerována dříve. Navíc víme, že četnost kombinace *Ant* je větší nebo rovna četnosti kombinace *Comb* [15].

Při vytváření pravidel se využívá skutečnosti, že četnost nadkombinace je nejvýše rovna četnosti kombinace:

$$n(Comb_1) \geq n(Comb_2) \quad \text{pro} \quad Comb_1 \succ Comb_2 .$$

Tedy je-li $Ant \succ Ant'$, je spolehlivost pravidla $Ant' \Rightarrow Comb - Ant'$ větší nebo rovna spolehlivosti pravidla $Ant \Rightarrow Comb - Ant$ [16].

Tudíž nevyhovuje-li zadané minimální spolehlivosti pravidlo $Ant' \Rightarrow Comb - Ant'$, nevyhoví ani žádné pravidlo $Ant \Rightarrow Comb - Ant$ kde $Ant \succ Ant'$. Podobně, aby mohlo zadanou minimální spolehlivost splnit pravidlo $Comb - Suc' \Rightarrow Suc'$, musí ji splnit všechna pravidla $Comb - Suc \Rightarrow Suc$, kde opět $Suc \succ Suc'$. Tedy např. vyhovuje-li pro kombinaci $Comb=A_1A_2A_3A_4$ pravidlo $A_1A_2 \Rightarrow A_3A_4$, budou vyhovovat i pravidla $A_1A_2A_3 \Rightarrow A_4$ a $A_1A_2A_4 \Rightarrow A_3$.

Začíná se tedy *Comb* rozdělovat tak, že *Suc* je nejprve tvořeno jednou položkou (kombinací délky 1). U těch pravidel, která dosáhla požadovanou spolehlivost se do *Suc* přidá další položka z *Ant* atd.

[14] Tedy *Ant* a *Suc* neobsahují stejnou kategorii ($Ant \cap Suc = \varnothing$) a zároveň $Ant \curlywedge Suc = Comb$.

[15] Kratší (méně omezující kombinaci) odpovídá alespoň tolik příkladů v datech jako kombinaci delší.

[16] Je-li Ant' nadkombinací kombinace *Ant*, je Ant' přísnější a tudíž ho splní méně příkladů. Vzhledem k tomu, že četnost Ant' je ve jmenovateli zlomku, přičemž čitatel zůstává stejný, je spolehlivost pravidla $Ant' \Rightarrow Comb - Amt'$ alespoň taková jako spolehlivost pravidla $Ant \Rightarrow Comb - Ant$.

5.2.5 Zobecněná asociační pravidla

Zboží, které si zákazník v supermarketu ukládá do košíku je součástí přirozené taxonomie. Obchod nabízí různé druhy nápojů, uzenin, hořčice apod. (Obr. 5). Takováto taxonomie se využívá při hledání *zobecněných* asociačních pravidel [Srikant, Agrawal, 1995]. Tato pravidla zachycují asociace mezi položkami na různé úrovni obecnosti (granularity) [17]. Zajímají nás tedy nejen pravidla na nejnižší úrovni hierarchie, např.

$$\text{lahůdkový párek} \Rightarrow \text{kremžská hořčice}$$

ale i obecnější (a tedy kompaktnější a snad srozumitelnější) pravidla využívající této taxonomie

$$\text{párek} \Rightarrow \text{hořčice}.$$

Zobecněné asociační pravidlo je tedy pravidlo ve tvaru

$$\text{Ant} \Rightarrow \text{Suc},$$

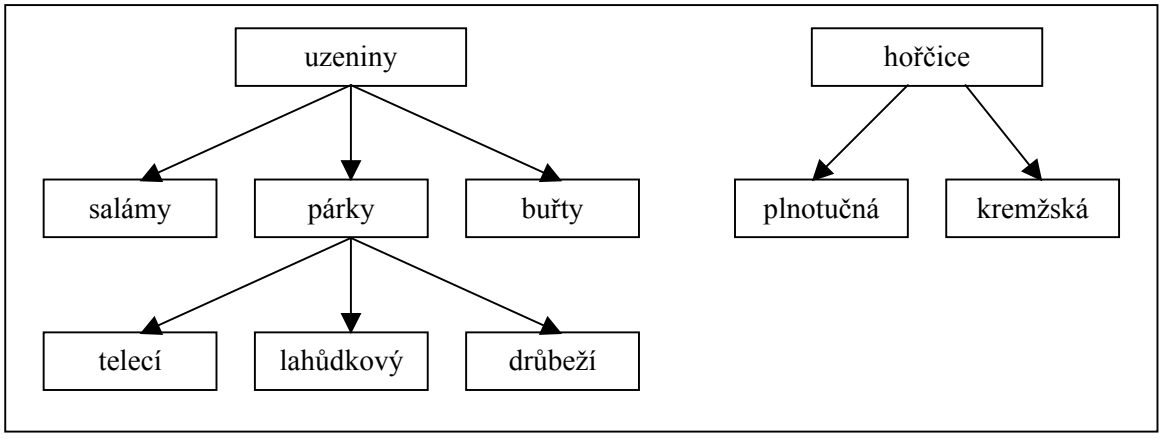
kde $Ant \cap Suc = \varnothing$ a žádná položka v *Suc* není předchůdcem žádné položky v *Ant* vzhledem k uvažované hierarchii [18]. Jinak se ale v pravidle objevují položky (kategorie) z různých úrovní hierarchie.

Problém při hledání zobecněných pravidel je ve vhodné volbě minimální požadované podpory (support). Je-li hodnota tohoto parametru příliš vysoká, nenaleznou se pravidla na nejnižší úrovni, není ani zaručeno, že nalezneme příslušné obecnější pravidlo (důvodem může být nízká spolehlivost takového pravidla). Je-li hodnota minimální požadované podpory příliš vysoká, dojde ke kombinatorické explozi, navíc budou v pravidlech figurovat položky z vyšších úrovní hierarchie ve všech možných vzájemných kombinacích. Platí totiž, že pokud má požadovanou podporu kombinace

$$\text{Ant} \wedge \text{Suc},$$

budou ji mít i kombinace vytvořené z obecnějších položek (předchůdců) položek *Ant* a *Suc*:

$$\text{Ant} \wedge \text{předchůdce(Suc)},\ \text{předchůdce(Ant)} \wedge \text{Suc, a předchůdce(Ant)} \wedge \text{předchůdce(Suc)}$$ [19]



Obr. 5 Taxonomie sortimentu zboží

[17] Jedná se o analogii s operacemi roll-up a drill-down používanými v systémech OLAP.

[18] Tato druhá podmínka eliminuje triviální pravidla typu *kremžská hořčice* $\Rightarrow$ *hořčice*. U „běžných" asociačních pravidel se podmínka neuplatní, protože neuvažujeme hierarchii hodnot atributů.

[19] *Předchůdce(X)* jakožto obecnější koncept bude pokrývat více příkladů než *X*.

Bude-li pravidlo

$$\text{Ant} \Rightarrow \text{Suc}$$

splňovat požadovanou podporu a spolehlivost, je pouze pro pravidlo

$$\text{Ant} \Rightarrow \text{předchůdce(Suc)}$$

zaručeno, že rovněž splní oba parametry. O spolehlivosti pravidel

$$\text{předchůdce(Ant)} \Rightarrow \text{Suc}\ , \quad \text{předchůdce(Ant)} \Rightarrow \text{předchůdce(Suc)}$$

nelze dopředu nic říci. To ilustruje jednoduchý příklad. V Tab. 3 jsou uvedeny údaje o nákupech (transakcích). Tab. 4 ukazuje položky s podporou alespoň 50%. V Tab. 5 jsou pak uvedena pravidla s podporou alespoň 50% a spolehlivostí alespoň 60%. Všimněme si, že chybí pravidlo *uzenina* $\Rightarrow$ *hořčice*, které nedosahuje požadované podpory.

nákup	položky
1	buřty
2	telecí párky
3	lahůdkové párky, kremžská hořčice
4	telecí párky, plnotučná hořčice

Tab. 3 Nákupy

položka	četnost
telecí párky	2
hořčice	2
párky	3
uzeniny	4

Tab. 4 Četnosti položek

pravidlo	podpora	spolehlivost
párek $\Rightarrow$ hořčice	50%	66%
hořčice $\Rightarrow$ párek	50%	100%
hořčice $\Rightarrow$ uzenina	50%	100%

Tab. 5 Zobecněná asociační pravidla

Jednou možností, jak se vypořádat s otázkou různé podpory na různých úrovních hierarchie je dynamicky měnit minimální požadovanou podporu v závislosti na úrovni hierarchie dané kombinace tak jak to popisují Chung a Lui. Vycházejí přitom z tzv. *stupně obecnosti g,* který je pro určitou hodnotu v hierarchii definován jako podíl počtu „listových" hodnot v podstromu uvažované hodnoty a počtu všech „listových" hodnot v dané hierarchii. Tedy např. pro hierarchii na Obr. 5 vlevo má hodnota *párky* stupeň obecnosti *3/5* a hodnota *telecí párky* stupeň obecnosti *0/5*. Stupeň obecnosti *g(Comb)* nějaké kombinace je pak dán minimem ze stupňů obecnosti jednotlivých kategorií (hodnot atributů) v kombinaci.

Při generování kombinací uživatel definuje parametry $minsup_0$ (požadovaná podpora pro kombinaci obsahující kategorii s nejnižším stupněm obecnosti $g=0$) a $minsup_1$ (požadovaná podpora pro kombinaci obsahující kategorii s nejvyšším stupni obecnosti $g=1$), $minsup_0 < minsup_1$, ze kterých se pak spočítá minimální požadovaná podpora libovolné kombinace jako ([Chung, Lui, 2000]):

$$\text{minsup(Comb)} = \text{minsup}_0 + (\text{minsup}_1 - \text{minsup}_0) \times \text{g(Comb)}\ .$$

Z kombinací dosahujících požadované podpory se pak vytvářejí asociační pravidla způsobem, který je popsán v předcházející podkapitole, tedy na základě rozkladu kombinace na dvě podkombinace.

5.2.6 Pravidla s výjimkami

Výsledkem algoritmů pro hledání asociačních pravidel bývá rozsáhlý seznam pravidel, které je nutno interpretovat. Vodítkem je obvykle nějakým způsobem definovaná zajímavost. Suzuki ve svých pracech (např. [Suzuki, 1997], [Suzuki, Kodratoff, 1998]) navrhuje považovat za zajímavá ta pravidla, která se vymykají ustáleným běžným představám (tzv. common sence).

Formálně je pravidlo s výjimkou definováno na základě trojice pravidel

$$\mathrm{Comb_A} \Rightarrow \mathrm{Suc}$$

$$\mathrm{Comb_A} \wedge \mathrm{Comb_B} \Rightarrow \neg\mathrm{Suc}$$

$$\mathrm{Comb_B} \Rightarrow \neg\mathrm{Suc}$$

kde první pravidlo odpovídá ustáleným představám (toto pravidlo má vysokou podporu i spolehlivost), druhé pravidlo je hledaná výjimka (toto pravidlo má nízkou podporu ale vysokou spolehlivost), a třetí pravidlo je takzvané referenční (má nízkou podporu a/nebo nízkou spolehlivost).

Příkladem pravidla s výjimkou může být následující trojce pravidel:

1. použité bezpečnostní pásy ⇒ přežití automobilové havárie
   (obecně uznávané pravidlo o účinnosti bezpečnostních pásů)

2. použité bezpečnostní pásy ∧ věk=předškolní ⇒ úmrtí při havárii
   (překvapivá výjimka, pro malé děti nejsou pásy vhodné)

3. věk=předškolní ⇒ úmrtí při havárii
   (referenční pravidlo, při haváriích umírá málo předškolních dětí)

Takovéto trojice je samozřejmě možné hledat až ve výsledném souboru pravidel. Suzuki ale nabízí alternativu; přímé generování. Hledání pravidla s výjimkou je řešeno jako simultánní hledání dvojce [obecné pravidlo, výjimka] reprezentované kombinacemi $[Comb_A, Comb_B]$ v prostoru všech dvojic kombinací. Implementovaný algoritmus prohledává tento prostor do hloubky. Prohledávání je řízeno parametry θ^{S}_1, θ^{S}_2, θ^{F}_1, θ^{F}_2, θ^{I}_2, které specifikují požadované kvantitativní charakteristiky nalezených pravidel [Suzuki, 1997]:

$$P(\mathrm{Comb_A}) \geq \theta^{S}_1,$$

$$P(\mathrm{Suc} \mid \mathrm{Comb_A}) \geq \theta^{F}_1,$$

$$P(\mathrm{Comb_A} \wedge \mathrm{Comb_B}) \geq \theta^{S}_2,$$

$$P(\neg\mathrm{Suc} \mid \mathrm{Comb_A} \wedge \mathrm{Comb_B}) \geq \theta^{F}_2,$$

$$P(\neg\mathrm{Suc} \mid \mathrm{Comb_B}) \leq \theta^{I}_2 .$$

Při prohledávání se využívá skutečnosti, že nesplní-li nějaká dvojice kombinací $[Comb_A, Comb_B]$ některý z výše uvedených požadavků, nesplní tento požadavek ani žádná dvojice $[Comb_{A'}, Comb_B]$ taková, že kombinace $Comb_{A'}$ je nadkombinací kombinace $Comb_A$, nebo kombinace $Comb_{B'}$ je nadkombinací kombinace $Comb_B$. V takovém případě se tedy může další prohledávání z uzlu $[Comb_A, Comb_B]$ ukončit.

5.2.7 Časové sekvence

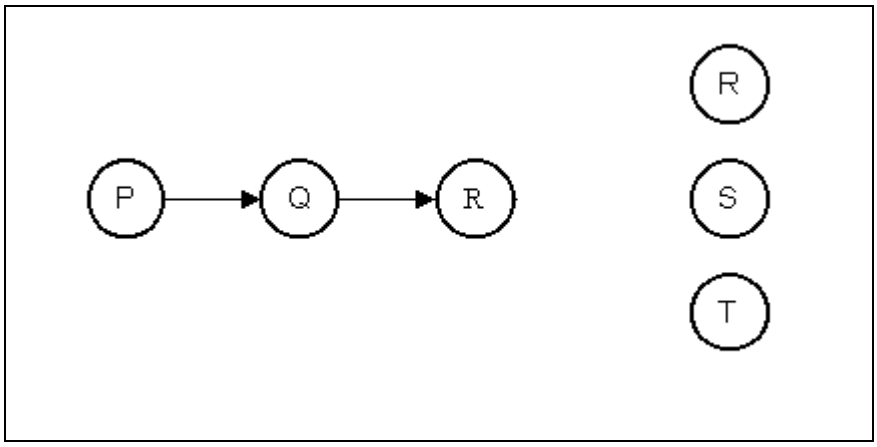
Většinou se asociační pravidla hledají v databázích, kde se neuvažuje s faktorem času. Záznamy v databázi prostě zachycují charakteristiky nějakých objektů. Někdy se ale ocitneme v situaci, že potřebujeme nalézt asociace mezi *událostmi*, které se odehrávají v různých časových okamžicích. Příkladem může být databáze poruch v telekomunikační síti.

Problematikou hledání opakujících se *epizod* v sekvenci nějakých událostí se zabývá např. Mannila [Mannila a kol., 1995]. Příkladem sekvence událostí může být:

(P, 123), (Q, 125), (S, 140), (P, 150), (R, 151),

(Q, 155), (S, 201), (P, 220), (S, 222), (Q, 225).

kde v zápise *(písmeno, číslo)* je písmeno název události a číslo je údaj o čase, ve kterém událost nastala[20].



Obr. 6 Sériová epizoda (vlevo) a paralelní epizoda (vpravo)

Epizodou pak může být "P se stane dříve než Q a Q se stane dříve než R". (tzv. sériová epizoda [21]), nebo "R, S a T se stanou současně" (tzv. paralelní epizoda) – viz Obr. 6. Z těchto dvou základních typů epizod lze skládat i epizody složitější. Epizoda je tedy částečně uspořádaná množina událostí. Při analýze časových sekvencí jde o to nalézt epizody, které se opakují dostatečně často.

Pokud pracujeme s konkretními časovými okamžiky, je pro hledání epizod podstatné zadání časového okna, uvnitř kterého se musí epizoda vyskytnout. Toto okno pevné délky w_d se posouvá po sekvenci událostí. Okno lze posouvat s krokem pevné délky (je-li délka časové sekvence T a krok rovný jedné, je počet všech oken, která musíme prozkoumat rovný $T-w_d+1$), nebo tak, že nové okno začíná s další událostí. Je-li například sekvencí událostí výše uvedený příklad a je-li $w_d=20$, budeme pro druhý způsob posouvání okna zkoumat okna

[P Q S], [Q S], [S P R Q], [P R Q], [R, Q], [Q], [S P], [P S Q].

[20] Někdy přesný časový údaj o události nepotřebujeme znát, protože nám stačí vědět pouze to, že některá událost předcházela události jiné. Příkladem takovýchto sekvencí mohou být opakovaná vyšetření pacientů v nemocnici (událostí je pak výsledek vyšetření nějakého pacienta a sekvencí je seznam výsledků vyšetření tohoto pacienta), nebo opakované nákupy v supermarketu (událostí je pak jeden nákup jednoho zákazníka a sekvencí je seznam nákupů tohoto zákazníka).

[21] V případě sériové epizody se mezi P, Q a R mohou vyskytnout i jiné události.

V těchto oknech tedy budeme hledat četné epizody. Opět vycházíme ze skutečnosti, že má-li pro okno dané délky dostatečnou četnost epizoda $P \rightarrow Q \rightarrow R$, mají dostatečnou četnost i epizody $P \rightarrow Q$, $Q \rightarrow R$ a $P \rightarrow R$. Můžeme tedy delší epizody skládat z kratších úseků (tzv. podepizod).

Pro hledání epizod které se vyskytují alespoň se zadanou četností lze použít algoritmus *aprioriAll* ([Agrawal, Srikant, 1995]), Tento algoritmus pracuje tak, že nejprve nalezne četné epizody tvořené jednou událostí, z nich skládá epizody tvořené dvěma událostmi a v rámci daného časového okna zjišťuje jejich četnosti. Pak se zjišťuje četnost potenciálně zajímavých epizod tvořených třemi událostmi, což jsou epizody takové, že všechny jejich "podepizody" délky dvě vyhovují svou četností [22], atd. Jak už název algoritmu napovídá, jedná se o variantu algoritmu *apriori* popsaného na jiném místě této kapitoly. Jediným větším rozdílem je to, že místo generování kombinací (u kterých nezáleží na pořadí kategorií) se nyní generují epizody událostí uspořádaných v čase.

Budeme-li ve výše uvedených oknech hledat epizody s četností alespoň 4, zjistíme, že z epizod délky jedna vyhovují epizody P (s četností 5), Q (s četností 7) a S (s četností 5). Z těchto epizod budeme vytvářet potenciálně vyhovující epizody délky dvě. Tyto epizody jsou spolu s četnostmi uvedeny v Tab. 6. Z této tabulky plyne, že žádná epizoda délky tři a více nebude vyhovovat požadavku na minimální četnost 4.

Epizoda	P→Q	Q→P	P→S	S→P	Q→S	S→Q
Četnost	4	0	2	2	2	2

Tab. 6 Četnost epizod délky dvě

Nalezené epizody lze použít pro predikci. Víme-li například, že epizoda $P \rightarrow Q \rightarrow R$ se objevuje ve 4% oken a že její podepizoda $P \rightarrow Q$ se objevuje ve 4.2%, můžeme na základě pozorování epizody $P \rightarrow Q$ predikovat (s pravděpodobností 0,95), že bude následovat událost R.

5.2.8 Více tabulek

Většina algoritmů pro dobývání znalostí z databází pracuje s jednou datovou tabulkou (maticí). Výjimkou nejsou ani algoritmy pro hledání asociačních pravidel. Přesto můžeme v literatuře nalézt přístupy, které tento problém řeší jinak, než „prostým" spojením všech relevantních tabulek do jedné v kroku předzpracování, tedy před vlastním výpočtem.

Jensen a Soparkar [Jensen, Soparkar, 2000] popisují postup, umožňující hledat kombinace s vysokou četností v datovém skladu organizovaném do hvězdy. O schematu hvězdy se píše v kapitole o databázích, zde jen připomeňme, že hvězda obsahuje jednu centrální tabulku faktů, ze které se odkazuje (pomocí tzv. cizích klíčů) do tabulek jednotlivých dimenzí. Příklad tří tabulek tvořících hvězdu ukazují Tab. 7, Tab. 8 a Tab. 9. Navržený způsob hledání četných kombinací pracuje ve dvou krocích:

1. hledání četných kombinací v tabulkách dimenzí,
2. spojení výsledků z jednotlivých tabulek dimenzí s využitím cizích klíčů v tabulce faktů.

[22] Například epizoda $P \rightarrow Q \rightarrow R$ má tři podepizody délky 2: $P \rightarrow Q$, $Q \rightarrow R$ a $P \rightarrow R$.

účet#	věk	pohlaví	PSČ
01	20..29	M	x
02	20..29	Z	x
03	30..39	M	z
04	30..39	Z	y

Tab. 7 Dimenze účet

karta#	typ	limit
A	junior	2000
B	junior	3000
C	classic	5000

Tab. 8 Dimenze karta

operace#	účet#	karta#	typ	částka
01	01	A	Výběr	1500
02	01	A	Platba	320
03	03	C	Výběr	3000
04	02	B	Výběr	1000
05	04	C	Platba	2500
06	04	C	Platba	1100

Tab. 9 Fakta o transakcích s kartami

V prvním kroku se nejprve se zjistí, kolikrát se každý záznam z tabulky dimenzí objeví v tabulce faktů. Ke každé tabulce dimenzí se tedy vytvoří vektor udávající "počet opakování" jednoho řádku tabulky dimenzí v tabulce faktů (Tab. 10). Tento vektor se využije při hledání četných kombinací v tabulkách dimenzí.

První krok ilustruje Tab. 7 a Tab. 8. V tabulce "účet" je kombinace *pohlaví(Z)* $\wedge$ *PSČ(y)* zastoupena jednou (má podporu 0.25), vezmeme-li do úvahy i tabulku transakce, bude mít tato kombinace podporu 0.33.

Účet#	01	02	03	04
Četnost	2	1	1	2

Karta#	A	B	C
Četnost	2	1	3

Tab. 10 Počet výskytů jednotlivých řádků tabulek dimenzí

Ve druhém kroku se pak spojí výsledky z jednotlivých tabulek za použití relací mezi tabulkami. Získáme tak vlastně kombinace "napříč" tabulkami, tedy např. *pohlaví(Z)* $\wedge$ *PSČ(y)* $\wedge$ *typ(classic)* . Tato kombinace má podporu 0.33 (vyskytuje se dvakrát), přitom kombinace *pohlaví(Z)* $\wedge$ *PSČ(y)* (z tabulky účet) má podporu 0.33 a kombinace *typ(classic)* (z tabulky karta) má podporu 0.5. Kombinace "napříč" tabulkami se tedy vytváří z podkombinací nalezených v prvním kroku.

Z výpočetního pohledu je popsaný postup opět inspirován algoritmem *apriori*.

5.2.9 Implikace, dvojité implikace a ekvivalence

Zatím jsme uvažovali asociační pravidla v Agrawalově pojetí, iterpretovatelná jako implikace

$$\mathrm{Ant} \Rightarrow \mathrm{Suc}.$$

Základními charakteristikami těchto pravidel jsou podpora $\frac{a}{a+b+c+d}$ a spolehlivost $\frac{a}{a+b}$.

Zajímavý obecnější pohled na typy pravidel (vztahů mezi předpokladem a závěrem opírající se o numerické charakterisitky odvozené z kontingenční tabulky) můžeme nalézt v původní české metodě *GUHA* (viz dále). Tato metoda pracuje s vlastní terminologií vycházející z predikátové logiky. Pro předpoklad se používá termín antecedent, pro závěr termín sukcedent a pro typ pravidla termín *(zobecněný) kvantifikátor.*

Proč zrovna kvantifikátor? Pojem kvantifikátor je běžně používán v matematice, kde se pracuje se dvěma kvantifikátory: obecným ($\forall$) a existenčním ($\exists$). Formule $\forall x T(x)$ je pravdivá, právě když všechna x splňují tvrzení T, formule $\exists x T(x)$ je pravdivá, právě když alespoň jedno x splňuje T. Pokud si pod x představíme příklady (objekty) v databázi a pod $T(x)$ implikaci $Ant(x) \Rightarrow Suc(x)$, bude formule $\forall x T(x)$ pravdivá, pokud $Ant \Rightarrow Suc$ bude konzistentní pravidlo. Odsud je již jen krůček k zobecnění pojmu kvantifikátor na situace, kdy požadujeme aby alespoň předem zadaný počet objektů splnil nějaké tvrzení vytvořené z kombinací Ant a Suc [23]. Budeme tedy zobecněný kvantifikátor chápat jako zobrazení z kontingenční tabulky na hodnoty 0 a 1. V tomto smyslu můžeme kontingenční (čtyřpolní) tabulku (a,b,c,d) takovou, že $F(a,b,c,d) \geq p$, kde

$$F(a,b,c,d) = \frac{a}{a+b}, \quad a \quad p \in [0,1]$$

chápat jako pravdivou formuli $Ant \sim_p Suc$, přičemž kvantifikátor $\sim_p$ odpovídá našemu známému (asociačnímu) pravidlu mezi Ant a Suc. Toto pravidlo má spolehlivost alespoň p.

Po tomto spíše terminologickém úvodu můžeme přejít k typologii vztahů mezi Ant a Suc. Vycházíme přitom z [Rauch,1998a,b] a [Iránek, 1999]. Bude-li funkce F definující kvantifikátor záviset pouze na hodnotách a a b z kontingenční tabulky, budeme psát $\sim(a,b)$, bude-li funkce F záviset na hodnotách a, b a c, budeme psát $\sim(a,b,c)$, bude-li F záviset na všech hodnotách, budeme psát $\sim(a,b,c,d)$ [24].

Nejprve uveďme základní typy pravidel (kvantifikátorů)[25]:

- základní implikace $Ant \Rightarrow Suc$,
  kde $\Rightarrow(a,b) = \frac{a}{a+b}$

- základní dvojitá implikace $Ant \Leftrightarrow Suc$,
  kde $\Leftrightarrow(a,b,c) = \frac{a}{a+b+c}$

- základní ekvivalence $Ant \equiv Suc$,
  kde $\equiv(a,b,c,d) = \frac{a+d}{a+b+c+d}$

Základní implikace $Ant \Rightarrow Suc$ (odpovídající Agrawalově asociačnímu pravidlu) je asymetrický vztah, zbývající dva vztahy jsou symetrické; $Ant \Leftrightarrow Suc$ je totéž jako $Suc \Leftrightarrow Ant$ a $Ant \equiv Suc$ je totéž jako $Suc \equiv Ant$. $Ant \Leftrightarrow Suc$ je ekvivalentní formuli $(Ant \Rightarrow Suc) \wedge (Suc \Rightarrow Ant)$, $Ant \equiv Suc$ měří vzájemnou závislost mezi Ant a Suc.

---

[23] Připomeňme, že v případě obecného kvantifikátoru to musí být všechny objekty, v případě existenčního kvantifikátoru alespoň jeden objekt.

[24] $\sim(a,b)$, $\sim(a,b,c)$ i $\sim(a,b,c,d)$ tedy bude v rozsahu $[0,1]$.

[25] Místo základní se rovněž říká fundovaná; tedy např. fundovaná implikace.

Uvedené vztahy (kvantifikátory) jsou nejjednoduššími představiteli různých tříd kvantifikátorů:

1. kvantifikátor $\sim(a,b)$ je *implikační*, právě když
$a' \geq a \wedge b' \leq b$ implikuje $\sim(a',b') \geq \sim(a,b)$

2. kvantifikátor $\sim(a,b,c)$ je *dvojitě implikační*, právě když
$a' \geq a \wedge b' \leq b \wedge c' \leq c$ implikuje $\sim(a',b',c') \geq \sim(a,b,c)$

3. kvantifikátor $\sim(a,b,c)$ je *∑-dvojitě implikační*, právě když
$a' \geq a \wedge b' + c' \leq b + c$ implikuje $\sim(a',b',c') \geq \sim(a,b,c)$

4. kvantifikátor $\sim(a,b,c,d)$ je *ekvivalenční*, právě když
$a' \geq a \wedge b' \leq b \wedge c' \leq c \wedge d' \geq d$ implikuje $\sim(a',b',c',d') \geq \sim(a,b,c,d)$

5. kvantifikátor $\sim(a,b,c,d)$ je *∑-ekvivalenční*, právě když
$a' + d' \geq a + d \wedge b' + c' \leq b + c$ implikuje $\sim(a',b',c',d') \geq \sim(a,b,c,d)$

6. kvantifikátor $\sim(a,b,c,d)$ je *fisherovský*, právě když
$a' \geq a \wedge d' \geq d \wedge |b' - c'| \geq |b - c|$ implikuje $\sim(a',b',c',d') \geq \sim(a,b,c,d)$

Z hlediska takto definovaných tříd je základní implikace implikační kvantifikátor, základní dvojitá implikace ∑-dvojitě implikační kvantifikátor a základní ekvivalence ∑-ekvivalenční kvantifikátor.

Jiným příkladem je trojice kvantifikátorů[26] motivovaných statistickým testem, který testuje nulovou hypotézu, že podmíněná pravděpodobnost vyjadřující vztah mezi *Ant* a *Suc* je větší nebo rovna danému *p* proti alternativní hypotéze že podmíněná pravděpodobnost je menší než *p*:

- horní kritická implikace $Ant \Rightarrow^{?}_{p} Suc$,

  kde $\Rightarrow^{?}_{p}(a,b) = \sum_{i=0}^{a} \frac{(a+b)!}{i!(a+b-i)!} p^i (1-p)^{a+b-i}$

- horní kritická dvojitá implikace $Ant \Leftrightarrow^{?}_{p} Suc$,

  kde $\Leftrightarrow^{?}_{p}(a,b,c) = \sum_{i=0}^{a} \frac{(a+b+c)!}{i!(a+b+c-i)!} p^i (1-p)^{a+b+c-i}$

- horní kritická ekvivalence $Ant \equiv^{?}_{p} Suc$,

  kde $\equiv^{?}_{p}(a,b,c,d) = \sum_{i=0}^{a} \frac{(a+b+c+d)!}{i!(a+b+c+d-i)!} p^i (1-p)^{a+b+c+d-i}$

Opět, horní kritická implikace je implikační kvantifikátor, horní kritická dvojitá implikace je ∑-dvojitě implikační kvantifikátor a horní kritická ekvivalence je ∑-ekvivalenční kvantifikátor.

[26] Všechny v této podkapitole uvedené kvantifikátory jsou implementovány v systému *4FT-Miner* (viz dále). Původní metoda *GUHA* používala (mimo jiné) ze zmíněných kvantifikátorů pouze základní implikaci a horní kritickou implikaci.

Můžeme si všimnout, že základní kvantifikátory jsou spolu úzce svázány:

$$\Leftrightarrow(a,b,c) = \Rightarrow(a,b+c)$$

$$\equiv(a,b,c,d) = \Leftrightarrow(a+d,b,c)$$

Analogickou vazbu můžeme pozorovat i mezi kvantifikátory založenými na statistickém testu. Obecně pak platí první vztah mezi kvantifikátorem implikačním a ∑-dvojitě implikačním, a druhý vztah mezi kvantifikátorem je ∑-dvojitě implikačním a kvantifikátorem ∑-ekvivalenčním. Tuto vazbu můžeme použít pro vytváření nových kvantifikátorů počínaje implikačním. Přitom takto vytvořený ∑-dvojitě implikační kvantifikátor bude přísnější[27] než implikační kvantifikátor a ∑-ekvivalenční kvantifikátor bude přísnější než ∑-dvojitě implikační kvantifikátor [Iványek, 1999].

5.2.10 Metoda GUHA

Zhruba 30 let před Agrawalem přišla s konceptem asociačních pravidel skupina českých vědců kolem P. Hájka [28]. Základní myšlenkou jejich metody *GUHA* (General Unary Hypotheses Automaton) bylo nalézt v datech všechny zajímavé souvislosti (hypotézy) a nabídnout je uživateli ([Hájek, Havránek, 1978], [Hájek a kol.,1983]).

V době svého vzniku, kdy se ještě nic netušilo o metodách dobývání znalostí, se *GUHA* řadila k metodám *exploračních analýzy dat.* Na rozdíl od konfirmační analýzy, kdy cílem bylo ověřit platnost konkrétní statistické hypotézy, při explorační analýze je cílem tyto hypotézy nejen testovat ale i vytvářet. Neboli, jak pravila dobová metafora, zatímco konfirmační analýza se dá přirovnat k chytání ryb na udici, metoda *GUHA* umožňuje výlov celého rybníka.

Jádrem GUHY je spojení metod pro generování hypotéz s metodami pro jejich (statistické) testování. Postupem času bylo formulováno několik typů hypotéz (a s tím souvisejících algoritmů pro jejich generování): vztahy mezi kombinacemi hodnot binárních atributů, korelace mezi numerickými atributy podmíněné kombinací kategoriálních atributů, nebo hledání zdrojů závislosti v nominálních datech. My se zaměříme pouze na jeden typ hypotéz a jeden algoritmus pro jejich generování a testování. Budeme tedy nadále hovořit o *GUHA* proceduře *4FT-Miner* vyvinuté na VŠE [Rauch, 1997] v návaznosti na původní *GUHA* proceduru *ASSOC* ([Hájek a kol.,1983]). Poznamenejme ještě na úvod, že jiná nová implementace procedury *ASSOC* vznikla pod názvem *GUHA+−* v Ústavu informatiky AV ČR [Honzíková, 1999].

Hypotézy generované a testované[29] procedurou *4FT-Miner* mají podobu

$$\text{Ant} \sim \text{Suc} / \text{Cond},$$

Kde *Ant* (antecedent), *Suc* (sukcedent) a *Cond* (podmínka) jsou konjunkce literálů a symbol ~ značí zobecněný kvantifikátor charakterizující typ vztahu mezi *Ant* a *Suc* na podmatici objektů, které splňují podmínku *Cond.*

---

[27] Kvatifikátor $\sim_1(a,b,c,d)$ je přísnější než kvantifikátor $\sim_2(a,b,c,d)$, právě když $\sim_1(a,b,c,d) < \sim_2(a,b,c,d)$.

[28] První česky publikovaná práce týkající se metody *GUHA* je Hájek P., Havel I, Chytil M.K.: Metoda GUHA automatického vyhledávání hypotéz, Kybernetika 2, (1996), 31-41. První mezinárodní publikací je pak Hájek P., Havel I, Chytil M.K.: The GUHA method of automatic hypotheses determination, Computing 1 (1966), 293--308.

[29] V terminologii metody *GUHA* se mluví o relevantních otázkách (to jsou hypotézy které vyhovují požadavkům na antecedent, sukcedent a podmínku, ale které ještě nebyly testovány v datech) a o relevantních tvrzeních (to jsou hypotézy podporované daty, tedy hypotézy, které vyhovují použitému kvantifikátoru.

Základním stavebním kamenem pro konstrukci hypotéz je takzvaný *literál* (pozitivní nebo negativní), definovaný jako *atribut(koeficient)* v případě *pozitivního literálu* resp. jako *¬atribut(koeficient)* v případě *negativního literálu* [30]. *Koeficient* (seznam hodnot atributu) pak může být:

- *podmnožina* omezené délky
  např. literál *město(Praha, Brno)* obsahuje podmnožinu délky 2,
- *interval* omezené délky
  např. literály *věk(nízký, střední)*, *věk(střední)*, *věk(střední, vysoký)* obsahují interval délky 1 až 2,
- *řez* (interval, obsahující krajní hodnotu) omezené délky
  např. literály *věk(nízký)*, *věk(nízký, střední)*, *věk(nízký, střední, vysoký)* obsahují dolní řez délky 1 až 3.

Kombinace *Ant*, *Suc* a *Cond* tak mají podstatně bohatší strukturu než jakou najdeme v jiných systémech pro hledání asociačních pravidel. *4FT-Miner* rovněž nabízí více typů vztahů; kvantifikátory implikační (Tab. 11), dvojitě implikační (Tab. 12), ekvivalenční (Tab. 13), Fisherovské (Tab. 14) a asociační (Tab. 15). Vztah platí, právě když hodnota funkce definující kvantifikátor splňuje příslušné podmínky.

Název	parametry	podmínka platnosti
Základní (fundovaná) implikace	$0 < p \leq 1$ $Base > 0$	$\frac{a}{a+b} \geq p \ \wedge \ a \geq Base$
Dolní kritická implikace	$0 < p \leq 1$ $0 < \alpha < 1$ $Base > 0$	$\sum_{i=a}^{a+b} \frac{(a+b)!}{i!(a+b-i)!} p^i(1-p)^{a+b-i} \leq \alpha \ \wedge \ a \geq Base$
Horní kritická implikace	$0 < p \leq 1$ $0 < \alpha < 1$ $Base > 0$	$\sum_{i=0}^{a} \frac{(a+b)!}{i!(a+b-i)!} p^i(1-p)^{a+b-i} > \alpha \ \wedge \ a \geq Base$

Tab. 11 Implikační kvantifikátory

název	parametry	podmínka platnosti
Základní (fundovaná) dvojitá implikace	$0 < p \leq 1$ $Base > 0$	$\frac{a}{a+b+c} \geq p \ \wedge \ a \geq Base$
Dolní kritická dvojitá implikace	$0 < p < 1$ $0 < \alpha < 1$ $Base > 0$	$\sum_{i=a}^{a+b+c} \frac{(a+b+c)!}{i!(a+b+c-i)!} p^i(1-p)^{a+b+c-i} \leq \alpha \ \wedge \ a \geq Base$
Horní kritická dvojitá implikace	$0 < p < 1$ $0 < \alpha < 1$ $Base > 0$	$\sum_{i=0}^{a} \frac{(a+b+c)!}{i!(a+b+c-i)!} p^i(1-p)^{a+b+c-i} > \alpha \ \wedge \ a \geq Base$

Tab. 12 Dvojitě implikační kvantifikátory

[30] Připomeňme si, že již dříve jsme definovali kategorii jako *Atribut(hodnota)*. Literál je tedy zobecněním kategorie. A naopak. Kategorie je pozitivní literál s koeficientem tvořeným podmnožinou délky jedna.

název	parametry	podmínka platnosti
Základní (fundovaná) ekvivalence	$0 < p \leq 1$ $Base > 0$	$\frac{a + d}{a + b + c + d} \geq p \ \wedge \ a \geq Base$
Dolní kritická ekvivalence	$0 < p < 1$ $0 < \alpha < 1$ $Base > 0$	$\sum_{i=a}^{a+b+c+d} \frac{(a + b + c + d)!}{i!(a + b + \ + d - i)!} p^i(1 - p)^{a+b+c+d-i} \leq \alpha \ \wedge \ a \geq Base$
Horní kritická ekvivalence	$0 < p < 1$ $0 < \alpha < 1$ $Base > 0$	$\sum_{i=0}^{a} \frac{(a + b + c + d)!}{i!(a + b + c + d - i)!} p^i(1 - p)^{a+b+c+d-i} > \alpha \ \wedge \ a \geq Base$
E-kvantifikátor	$0 < \delta < 1$	$a + b > 0 \ \wedge \ \frac{b}{a + b} < \delta \ \wedge \ c + d > 0 \ \wedge \ \frac{c}{c + d} < \delta$

Tab. 13 Ekvivalenční kvantifikátory

název	parametry	podmínka platnosti
Prosté vychýlení	$\delta > 0$ $Base > 0$	$ab > e^{\delta}cd \ \wedge \ a \geq Base$
Fisherův kvantifikátor	$0 < \alpha \leq 0.5$ $Base > 0$	$\sum_{i=a}^{\min(r,k)} \frac{r!s!k!l!}{n!i!(r-i)!(k-i)!(n-r-k-i)!} \leq \alpha \ \wedge \ a \geq Base$
Chi-kvadrát kvantifikátor	$0 < \alpha \leq 0.5$ $Base > 0$	$ad > bc \ \wedge \ \frac{n(ad - bc)}{klrs} > \alpha \ \wedge \ a \geq Base$

Tab. 14 Fisherovské kvantifikátory

název	parametry	podmínka platnosti
A-kvantifikátor	$0 < \gamma \leq 1$ $0 < \sigma \leq 1$	$\frac{a}{a + b} \geq \gamma \ \wedge \ \frac{a}{a + b + c + d} \geq \sigma$

Tab. 15 Asociační pravidla

Většinu těchto kvantifikátorů je možno nalézt v klasických pracech o metodě *GUHA* ([Hájek, Havránek, 1978], [Hájek a kol.,1983]), A-kvantifikátor odpovídá asociačnímu pravidlu s parametry spolehlivost a podpora a E-kvantifikátor lze nalézt v [Agrawal a kol., 1996].

Při spouštění procedury *4FT-Miner* se volí množina literálů[31] pro generování kombinací *Ant*, *Suc* a *Comb*, maximální délka těchto kombinací a typ a parametry kvantifikátoru. Chceme-li tedy pro naše data z Tab. 2 hledat všechna asociační pravidla se závěrem tvořeným nějakou kategorií a se spolehlivostí 0.9, můžeme spustit proceduru *4FT-Miner* s parametry:

[31] Zadává se atribut, možná podoba koeficientu, to zda je literál pozitivní, nebo negativní nebo obojí a to, zda se literál musí vyskytnout v kombinaci (tzv. základní literál), nebo může vyskytnout v kombinaci.

- kvantifikátor základní implikace, p=0.9, Base=5%,
- pro antecedent: max. délka 4, literály pouze pozitivní, vytvářeny pro všechny atributy v datech, koeficienty pouze jednoprvkové množiny (tedy literály jsou kategorie),
- pro sukcedent: max. délka 1, literály pouze pozitivní, vytvářeny pro všechny atributy v datech, jsou pouze pozitivní, koeficienty pouze jednoprvkové množiny (tedy literály jsou kategorie),
- podmínka nebude použita.

Pro uvedené zadání nalezne procedura *4FT-Miner* 206 pravidel. Při generování pravidla se nejprve vytvoří nějaký antecedent, k němu se pak naleznou všechny sukcedenty tak, aby pravidlo vyhovovalo zadaným parametrům. Při vytváření kombinací se postupuje do hloubky, literály jsou přitom uspořádány podle abecedy (podle názvů atributů resp. názvů hodnot). Část výpisu pravidel ukazuje Tab. 16. Všechna nalezená pravidla mají spolehlivost rovnu 1.

Konto(nízké) ∧ Nezaměstnaný(ano) ⇒ Pohlaví(žena)
Konto(nízké) ∧ Nezaměstnaný(ano) ∧ Pohlaví(žena) ∧ Příjem(nízký) ⇒ Úvěr(ne)
Konto(nízké) ∧ Nezaměstnaný(ano) ∧ Pohlaví(žena) ∧ Příjem(vysoký) ⇒ Úvěr(ano)
Konto(nízké) ∧ Nezaměstnaný(ano) ∧ Pohlaví(žena) ∧ Úvěr(ano) ⇒ Příjem(vysoký)
Konto(nízké) ∧ Nezaměstnaný(ano) ∧ Pohlaví(žena) ∧ Úvěr(ne) ⇒ Příjem(nízký)
Konto(nízké) ∧ Nezaměstnaný(ano) ∧ Příjem(nízký) ⇒ Úvěr(ne)
Konto(nízké) ∧ Nezaměstnaný(ano) ∧ Příjem(nízký) ⇒ Pohlaví(žena)
Konto(nízké) ∧ Nezaměstnaný(ano) ∧ Příjem(nízký) ∧ Úvěr(ne) ⇒ Pohlaví(žena)
Konto(nízké) ∧ Nezaměstnaný(ano) ∧ Příjem(vysoký) ⇒ Úvěr(ano)
Konto(nízké) ∧ Nezaměstnaný(ano) ∧ Příjem(vysoký) ⇒ Pohlaví(žena)
Konto(nízké) ∧ Nezaměstnaný(ano) ∧ Příjem(vysoký) ∧ Úvěr(ano) ⇒ Pohlaví(žena)
Konto(nízké) ∧ Nezaměstnaný(ano) ∧ Úvěr(ano) ⇒ Pohlaví(žena)
Konto(nízké) ∧ Nezaměstnaný(ano) ∧ Úvěr(ano) ⇒ Příjem(vysoký)
Konto(nízké) ∧ Nezaměstnaný(ano) ∧ Úvěr(ne) ⇒ Pohlaví(žena)
Konto(nízké) ∧ Nezaměstnaný(ano) ∧ Úvěr(ne) ⇒ Příjem(nízký)
Konto(nízké) ∧ Nezaměstnaný(ne) ⇒ Pohlaví(muž)
Konto(nízké) ∧ Nezaměstnaný(ne) ∧ Pohlaví(muž) ∧ Příjem(nízký) ⇒ Úvěr(ne)
Konto(nízké) ∧ Nezaměstnaný(ne) ∧ Pohlaví(muž) ∧ Příjem(vysoký) ⇒ Úvěr(ano)
Konto(nízké) ∧ Nezaměstnaný(ne) ∧ Pohlaví(muž) ∧ Úvěr(ano) ⇒ Příjem(vysoký)
Konto(nízké) ∧ Nezaměstnaný(ne) ∧ Pohlaví(muž) ∧ Úvěr(ne) ⇒ Příjem(nízký)
Konto(nízké) ∧ Nezaměstnaný(ne) ∧ Příjem(nízký) ⇒ Úvěr(ne)
Konto(nízké) ∧ Nezaměstnaný(ne) ∧ Příjem(nízký) ⇒ Pohlaví(muž)
Konto(nízké) ∧ Nezaměstnaný(ne) ∧ Příjem(nízký) ∧ Úvěr(ne) ⇒ Pohlaví(muž)
Konto(nízké) ∧ Nezaměstnaný(ne) ∧ Příjem(vysoký) ⇒ Úvěr(ano)
Konto(nízké) ∧ Nezaměstnaný(ne) ∧ Příjem(vysoký) ⇒ Pohlaví(muž)
Konto(nízké) ∧ Nezaměstnaný(ne) ∧ Příjem(vysoký) ∧ Úvěr(ano) ⇒ Pohlaví(muž)
Konto(nízké) ∧ Nezaměstnaný(ne) ∧ Úvěr(ano) ⇒ Pohlaví(muž)
Konto(nízké) ∧ Nezaměstnaný(ne) ∧ Úvěr(ano) ⇒ Příjem(vysoký)
Konto(nízké) ∧ Nezaměstnaný(ne) ∧ Úvěr(ne) ⇒ Pohlaví(muž)
Konto(nízké) ∧ Nezaměstnaný(ne) ∧ Úvěr(ne) ⇒ Příjem(nízký)
Konto(nízké) ∧ Pohlaví(muž) $\Rightarrow_{\emptyset}$ Nezaměstnaný(ne)
Konto(nízké) ∧ Pohlaví(muž) ∧ Příjem(nízký) ⇒ Úvěr(ne)
Konto(nízké) ∧ Pohlaví(muž) ∧ Příjem(nízký) ⇒ Nezaměstnaný(ne)
Konto(nízké) ∧ Pohlaví(muž) ∧ Příjem(nízký) ∧ Úvěr(ne) ⇒ Nezaměstnaný(ne)

Tab. 16 Asociační pravidla nalezená systémem 4FT-Miner

5.2.11 Kombinační analýza dat

Kombinační analýza dat (*KAD*) vychází z metody *GUHA*. Kombinační analýza dat byla inspirována zejména snahou po získání všech, v datech prokazatelných, hypotéz a formulací některých úloh, které řeší *GUHA* procedura *ASSOC* ([Hájek a kol, 1983]). Z pohledu GUHY představuje kombinační analýza dat jisté zjednodušení, neboť

- nabízí jednodušší podobu kombinací na levé a pravé straně vztahu; k dispozici je pouze konjunkce hodnot atributů,
- nabízí menší škálu možných vztahů; k dispozici je pouze implikace a dvojitá implikace v základní podobě,
- nabízí pevně zadané typy úloh.

Kombinační analýza umožňuje řešit následující úlohy:

- konkretní dotaz,
- kompletní úloha,
- analýza důsledků,
- analýza příčin.

V těchto úlohách jde o automatické generování a verifikování vzathů mezi dvěma kombinacemi. Pro dvě kombinace $Comb_1$ a $Comb_2$ hledá kombinační analýza základní implikace $Comb_1 \Rightarrow Comb_2$ případně základní dvojité implikace $Comb_1 \Leftrightarrow Comb_2$, které datech vyhovují zadaným požadavkům na četnost a spolehlivost. Zjišťujeme tedy (na základě čtyřpolní tabulky) hodnoty $\frac{a}{a+b}$ pro implikaci a $\frac{a}{a+b+c}$ pro dvojitou implikaci. Těmto hodnotám se v kombinační analýze říká *platnost vztahu* [32].

Nejjednodušší úlohou je *konkrétní dotaz*. Pro dvě zadané kombinace $Comb_1$ a $Comb_2$ se spočítají platnosti vztahů $Comb_1 \Rightarrow Comb_2$, $Comb_2 \Rightarrow Comb_1$ a $Comb_1 \Leftrightarrow Comb_2$. Tato úloha se uplatní, když chceme v datech získat argumenty pro potvrzení svých úvah o konkrétních kombinacích a vztazích, nebo chceme doplnit řešení některé další explorační úlohy informacemi o objektech, které splňují jednu nebo obě kombinace v nějakém zajímavém vztahu.

Podstatou *analýzy důsledků* je vyhledávání vztahů implikace, vyplývajících ze zadané výchozí kombinace *Comb*. Hledáme tedy vztahy typu

$$\text{Comb} \Rightarrow \ldots.$$

Tedy pro pevnou levou stranu generujeme kombinace na pravé straně. Tato úloha je vhodná, jestliže lze v datech některé kombinace označit za výchozí a úkolem je hledat důsledky jejich výskytu.

Podstatou *analýzy příčin* je vyhledávání vztahů implikace směřujících k zadané cílové kombinaci *Comb*. Tentokrát tedy hledáme vztahy typu

$$\ldots.. \Rightarrow \text{Comb}$$

[32] Platnost implikace je tedy totéž co spolehlivost (Agrawalova) asociačního pravidel nebo hodnota kvantifikátoru základní implikace v metodě *GUHA*. Platnost dvojité implikace odpovídá hodnotě kvantifikátoru základní dvojité implikace v metodě *GUHA*.

Tedy pro pevnou pravou stranu generujeme kombinace na levé straně. Tato úloha je vhodná, jestliže lze v datech některé kombinace označit za cílové a úkolem je hledat pro jejich výskyt příčiny.

Nejblíže k algoritmu apriori má *kompletní úloha.* Úkolem je nalézt všechny vztahy (implikace i dvojité implikace), které splňují zadané numerické charakteristiky maximální délka celého hledaného vztahu, minimální četnost celého hledaného vztahu (min počet objektů splňujících současně obě kombinace) a minimální platnost vztahu.

….. ? ….

Tato úloha je vhodná, pro první hrubou orientaci v neznámých datech, kdy si klademe otázku, zda existuje vůbec nějaká vazba mezi jednotlivými atributy. V každém kroku systém vygeneruje kombinaci *Comb*, rozdělí ji na všechny možné dvojice podkombincí $Comb_1$, $Comb_2$ tak, že $Comb = Comb_1 \wedge Comb_2$ a spočítá hodnoty základních implikací i základní dvojité implikace:

$$Comb_1 \Rightarrow Comb_2$$

$$Comb_2 \Rightarrow Comb_1$$

$$Comb_1 \Leftrightarrow Comb_2$$

Při generování kombinací *Comb* se v kombinační analýze postupuje podle četností. Algoritmus generování kombinací je uveden na Obr. 7. Generování je řízeno dvěma parametry; maximální požadovanou délkou kombinace l_{max} a minimální požadovanou četností kombinace n_{min}. Prodlužují se pouze ty kombinace, které jsou kratší než maximální požadovaná délka. Do *OPEN* se zařazují pouze kombinace, které mají četnost větší (a samozřejmě nenulovou), než je minimální požadovaná četnost. Generování podle četností byla dána přednost, protože:

- dříve generuje četnější (častěji se vyskytující) kombinace (a tedy i vztahy),
- dříve generuje spíše kratší kombinace (a tedy i vztahy) (přidáním kategorie do kombinace se zpřísní kritérium a tedy i sníží počet objektů, které ho splní).

Algoritmus generování kombinací

Inicializace

1. vytvoř *CAT* - seznam kategorií *A(v)* uspořádaný sestupně dle četnosti *n(A(v))*
2. přiřaď *OPEN := CAT*

Hlavní cyklus

1. dokud *OPEN* není prázdný seznam
   - 1.1. vezmi první kombinaci ze seznamu *OPEN* (označ ji *Comb*)
   - 1.2. je-li $l(Comb) < l_{max}$, pak
     - 1.2.1. pro každou kategorii *A(v)* ze seznamu *CAT* takovou, že:
       - atribut *A* se nevyskytuje v *Comb*
       - *A(v)* je v *CAT* před všemi kategoriemi z *Comb* - tedy platí, že četnost *n(A(v))* je větší nebo rovna četnosti *n(Comb)*
     - 1.2.1.1. generuj novou kombinaci $CombA = Comb \wedge A(v)$
     - 1.2.1.2. je-li $n(CombA) \geq n_{min}$ přidej *CombA* do seznamu *OPEN* za poslední kombinaci *Comb* takovou, že $n(Comb) \geq n(CombA)$
   - 1.3. odstraň *Comb* ze seznamu *OPEN*

Obr. 7 Algoritmus generování kombinací

Porovnáme-li tento algoritmus generování s dříve uvedeným algoritmem *apriori* zjistíme, že v obou případech se kombinace délky *k* vytváří z kratších kombinací majících dostatečnou četnost. Algoritmus *apriori* postupuje prostorem kombinací do šířky: použije *k-1* kombinací délky *k-1* lišících se od sebe vždy v jedné kategorii. Při kombinační analýze se postupuje prostorem kombinací heuristicky: použije se jedna dostatečně četná kombinace *Comb* délky *k-1* a jedna kombinace délky *1*; tato jednočlenná kombinace (kategorie *A(v)*) má přitom větší (nebo stejnou) četnost než je *n(Comb)*.

Pro naše data z Tab. 2 můžeme použít analýzu příčin pro nalezení všech implikací s pevně zadaným závěrem *Úvěr(ano)*. Pro parametry:

- max. délka antecedentu 4,
- min. četnost antecedentu 1,
- min. platnost (spolehlivost) implikací 0.9

nalezneme 46 implikací[33]. Všchny nalezené implikace mají platnost 1. Tab. 17 ukazuje výpis těchto implikací uspořádaný podle četnosti antecedentu (tedy v pořadí v jakém je antecedent generován). Výpis je, podobně jako výpis generování kombinací (Obr. 1, Obr. 2, Obr. 3), ve zkrácené podobě: kategorie je kódována číslem atributu a prvním písmenem hodnoty. Tedy první implikace *1v ⇒ 5a* znamená *Příjem(vysoký) ⇒ Úvěr(ano)*.

č.	n(Ant)	Implikace
1	5	.....1v => 5a
2	4	.....2v => 5a
3	4	....1v4n => 5a
4	3	....1v3z => 5a
5	2	....3z4n => 5a
6	2	....1v3m => 5a
7	2	....1n2v => 5a
8	2	....2v3m => 5a
9	2	....2v3z => 5a
10	2	....2v4a => 5a
11	2	....2v4n => 5a
12	2	....1v2v => 5a
13	2	....2s4n => 5a
14	2	....1v2n => 5a
15	2	..1v3m4n => 5a
16	2	..1v3z4n => 5a
17	2	..1n2v4a => 5a
18	2	..1v2v4n => 5a
19	1	....1v4a => 5a
20	1	....1v2s => 5a
21	1	..1n2v3m => 5a
22	1	..1n2v3z => 5a
23	1	..2v3m4a => 5a
24	1	..2v3z4a => 5a
25	1	..2v3m4n => 5a
26	1	..2v3z4n => 5a
27	1	..1v2v3m => 5a
28	1	..1v2v3z => 5a
29	1	..1n2s4n => 5a
30	1	..2s3m4n => 5a
31	1	..2s3z4n => 5a
32	1	..1v2n3m => 5a
33	1	..1v2n3z => 5a
34	1	..1v2n4a => 5a
35	1	..1v2n4n => 5a
36	1	1v2v3m4n => 5a
37	1	1v2v3z4n => 5a
38	1	..1v3z4a => 5a
39	1	..1v2s3z => 5a
40	1	..1v2s4n => 5a
41	1	1n2v3m4a => 5a
42	1	1n2v3z4a => 5a
43	1	1n2s3m4n => 5a
44	1	1v2n3z4a => 5a
45	1	1v2n3m4n => 5a
46	1	1v2s3z4n => 5a

Tab. 17 Implikce ..... ⇒ Úvěr(ano)

Kombinační analýza dat byla původně navržena a implementována jako samostatný systém *KAD* ([Ivánek, Stejskal, 1984]). V současnosti je součástí systému *KEX* [Berka, 1993].

5.

[33] Stejné výsledky bychom dostali i použitím procedury *4FT-Miner* s odpovídajícími parametry.

5.2.12 Chybějící hodnoty

Běžným způsobem, jak se vypořádat s chybějícími údaji v databázi je jejich ošetření ve fázi přípravy a předzpracování dat. O tomto přístupu je již zmínka v kapitole o rozhodovacích stromech. Na tomto místě zmiňme zajímavý přístup ošetření chybějících hodnot *během generování pravidel*, tak, jak to umožňuje *GUHA* a *4FT-Miner* [Rauch,1998b].

Charakteristiky konkrétního pravidla *Ant ~ Suc / Cond* (hodnoty funkce *F(a,b,c,d)* definující nějaký kvantifikátor) se počítají z odpovídající (čtyřpolní) kontingenční tabulky uvedené Tab. 1. V případě, že v datech nechybí žádná hodnota, je počet objektů v datech *n* roven *a+b+c+d.* V případě, že chybí hodnota některého atributu, který se vyskytuje v pravidle (v antecedentu nebo v sukcedentu), musíme místo čtyřpolní tabulky uvažovat tabulku devítípolní (Tab. 18)[34].

	S	**?S**	**¬S**	**Σ**
A	a´	i	b´	r
?A	o	m	p	
¬A	c´	j	d´	s
Σ	k		l	n

Tab. 18 (Devítipolní) kontingenční tabulka

Ošetření chybějících hodnot spočívá v tom, že se pro konkrétní pravidlo jeho devítípolní tabulka převede na čtyřpolní a teprve z této tabulky se počítají příslušné hodnoty. *GUHA* (a *4FT-Miner*) nabízí tři možnosti tohoto převodu (a tedy tři způsoby práce s chybějícími hodnotami): *konzervativní, optimistický* a *zabezpečený*. Přitom se bere do úvahy typ použitého kvantifikátoru, převod devítipolní tabulky na čtyřpolní tedy bude různý pro různé kvantifikátory.

Při *konzervativním* způsobu se objekty, u kterých chybí nějaký údaj, ignorují. Tedy

$$a=a´, b=b´, c=c´, d=d´.$$

Tato úprava je stejná pro všechny kvantifikárory.

Při *optimistickém* způsobu předpokládáme, že chybějící hodnoty podporují uvažovaný vztah mezi antecedentem a sukcedentem. V případě např. implikačních kvantifikátorů budeme tedy posilovat hodnotu *a*:

$$a=a´+i+o+m, b=b´,$$

v případě ekvivalenčních kvantifikátorů budeme posilovat hlavní diagonálu

$$a=a´+i+o+m, b=b´, c=c´, d=d´+j+p.$$

Při *zabezpečeném* způsobu předpokládáme, že chybějící hodnoty jsou v rozporu s uvažovaným vztahem mezi antecedentem a sukcedentem. V případě implikačních kvantifikátorů budeme tedy posilovat hodnotu *b*:

[34] Řádek *?A* odpovídá příkladům u kterých nevíme, zda jsou pokryty předpokladem (antecedenem), sloupec *?S* odpovídá příkladům u kterých nevíme, zda jsou pokryty závěrem (sukcedentem).

$$a=a', b=b'+i+m+p,$$

v případě ekvivalenčních kvantifikátorů budeme posilovat vedlejší diagonálu

$$a=a', b=b'+i+p+m, c=c'+j+o, d=d'.$$

Optimistický převod je (z hlediska uvažovaného kvantifikátoru) nejlepší možný, zabezpečený převod je nejhorší možný. Skutečná tabulka (kdybychom znali všechny chybějící hodnoty) bude někde mezi těmito extrémy

$$\sim_{zab}(a,b,c,d) \leq \sim_{skut}(a,b,c,d) \leq \sim_{opt}(a,b,c,d).$$

Můžeme tedy i z neúplných dat spočítat, v jakých mezích se bude pohybovat hodnota kvantifikátoru pro data bez chybějících hodnot.

Literatrura:

[Agrawal a kol., 1993] Agrawal,R. - Imielinski,T. – Swami,A.: Mining associations between sets of items in massive databases. In: Proc. of the ACM-SIGMOD 1993 Int Conference on Management of Data, Washington D.C., May 1993, 207-216.

[Agrawal, Srikant, 1995] Agrawal,R. - Srikant,R.: Mining sequential patterns. In: Proc. Int. Conf. On Data Engineering ICDE'95, Taiwan, 1995.

[Agrawal a kol., 1996] Agrawal,R. - Mannila,H. - Srikant,R. – Toivonen,H. – Verkamo, A.I.: Fast discovery of association rules. In: (Fayyad, Piatetsky-Shapiro, Smyth, Uthurusamy, eds.) Advances in Knowledge Discovery and Data Mining, AAAI/MIT Press, 1996, ISBN 0-262-56097-6.

[Berka, 1993] Berka,P.: Vybrané znalostní systémy, SAK, SAZE, KEX. Skripta VŠE, Praha 1993.

[Bruha, 1994] Bruha,I.: Combining rule qualities in a covering learning algorithm. In: (Nakhaeizadeh, Taylor) Proc: Machine Learning and Statistics, ECML'94 Mlnet Familiarization Workshop, Catania, 1994.

[Chung, Lui, 2000] Chung,F. – Lui,Ch.: A post-analysis framework for mining generalized association rules with multiple minimum supports. In: Proc. KDD-2000 Wshop on Post-Processing in Machine Learning and Data Mining.

[Hájek, Havránek, 1978] Hájek,P. – Havránek,T.: Mechanising Hypothesis Formation – Mathematical Foundations for a General Theory. Springer, 1978.

[Hájek a kol.,1983] Hájek,P. – Havránek,T. – Chytil,M.K.: Metoda GUHA. Automatická tvorba hypotéz. Academia, 1983.

[Hájek, Rauch, 1999] Hájek,P. – Rauch,J.: Logic and statistic for association rules and beyond. In: (Zytkow, Rauch eds.) Proc. 3rd European Conf. on Principles and Practice of KDD PKDD'99, Springer LNAI ,1999, 116-124.

[Holzheimer, Siebes, 1994] Holzheimer, M. – Siebes,A.: The search for knowledge in databases. Tech.Rep. CS-R9406, CWI, Amsterdam, 1994.

[Honzíková, 1999] Honzíková,Z.: GUHA+- user's guide, UI AV ČR, 1999.

[Ivánek, 1999] Ivánek,J.: On the correspondence between classes of implicational and equivalence quantifiers. . In: (Zytkow, Rauch eds.) Proc. 3rd European Conf. on Principles and Practice of KDD PKDD'99, Springer LNAI 1999.

[Ivánek, Stejskal, 1984] Ivánek,J. - Stejskal,B.: Combinational data analysis. In: Proc. COMPSTAT'84, Prague, 1984, s.95.

[Jensen, Soparkar, 2000] Jensen,V.C. – Soparkar,N.: Frequent itemset counting across multiple tables. In: Proc Pacific-Asian Conf. on KDD PAKDD 2000, Springer LNAI 1805, 2000, 49-61.

[Kodratoff, 1999] Kodratoff,I.: Comparing machine learning and knowledge discovery in databases. In: Lecture Notes from Machine Leaning and Applications, ACAI'99, Chania, Vol.1, 1999.

[Mannila a kol., 1995] Mannila,H. – Toivonen,H. – Verkamo,A.I.: Discovering frequent episodes in sequences. In: Proc. 1st Int. Conf. on Knowledge Discovery and Data Mining KDD-95, Montreal, 1995, 210-215.

[Rauch, 1997a] Rauch, J.: CD-ASSOC: Uvodní popis procedury. LISp-Report LISp-97-08, 1997.

[Rauch, 1997b] Rauch,J.: Logical calculi for Knowledge Discovery in Databases. In: (Komorowski, Zytkow eds) Proc. 1nd European Conf. on Principles of Data Mining and Knowledge Discovery PKDD'97, Springer, LNAI , 1997, 47-57.

[Rauch, 1998a] Rauch,J.: Classes of four-fold table quantifiers. In: (Quafafou, Zytkow eds) Proc. 2nd European Conf. on Principles of Data Mining and Knowledge Discovery PKDD'98, Springer, LNAI , 1998, 203-211.

[Rauch, 1998b] Rauch,J.: Four-fold table predicate calculi and missing information. In: Proc. Int. Conf. on Cmputational Intelligence and Neurosciences. 1998.

[Srikant, Agrawal, 1995] Srikant, R. – Agrawal,R.: Mining generalized association rules. In: Proc. 21st Conf. on Verly Large Databases VLDB´95, 1995.

[Suzuki, 1997] Suzuki,E.: Autonomous Discovery of Reliable Exception Rules. In: Proc. Int. Conf. on KDD KDD'97, 1997, 259-262.

[Suzuki, Kodratoff, 1998] Suzuki,E. – Kodrtoff,I.: Discovery of surprising exception rules based on intensity of implication. In: Proc. Principles and Practice of KDD PKDD'98, 1998.

[Torgo, 1993] Torgo,L.: Controlled redundancy in incremental rule learning. In: (Brazdil, ed.) Proc. European Conference on Machine Learning ECML'93, Springer, LNAI 667, 1993, 185-495.