Jiří Zlatuška

Logika

Zápisy z přednášky zpracoval:
Jan Šerák

12. května 1995

Obsah

Seznam obrázků

1 Výroková logika

1.1 Uspořádání a strom

Definice 1.1

Částečné uspořádání je množina S spolu s binární relací $<$ na S a je tranzitivní, tj. $(x < y) \wedge (y < z) \Longrightarrow x < z$, a ireflexivní, tj. $x < x$ neplatí pro žádné x.

Definice 1.2

Částečné uspořádání $<$ je *lineární uspořádání (uspořádání)*, jestliže splňuje podmínku

$$(x < y) \vee (x = y) \vee (y < x)$$

pro $\forall x, y \in S$.

Definice 1.3

Lineární uspořádání je *dobré uspořádání*, když neobsahuje žádný nekonečný sestupný řetězec $(x_0, x_1, \ldots \in S$ tak, že $\cdots < x_1 < x_0)$.

Konvence:

- $x \leq y \Longleftrightarrow (x < y) \vee (x = y)$
- $x < y \Longleftrightarrow y > x$

Definice 1.4

Strom je množina T, jejíž prvky nazýváme *uzly*, částečně uspořádaná relací $<_T$ s jediným nejmenším prvkem, zvaným *kořen*. Předchůdci každého uzlu jsou dobře uspořádány relací $<_T$.

Cesta P ve stromu T je maximální lineárně uspořádaná podmnožina T.

Definice 1.5

Úroveň ve stromu T je definována indukcí takto:

- Nultá úroveň obsahuje kořen T.
- $(k+1)$-tá úroveň T obsahuje všechny bezprostřední následníky všech uzlů k-té úrovně.

Hloubka stromu T je maximální n takové, že strom T obsahuje uzel úrovně n. Existuje-li uzel úrovně n pro $\forall n \in \mathbf{N}$, pak strom T je nekonečný.

Jestliže každý uzel stromu T má nejvýše n následovníků, pak strom nazýváme *n-ární*. Pokud mají všechny uzly konečně mnoho bezprostředních následovníků, strom nazýváme *strom s konečným větvením*.

Věta 1.6 Königova věta

Jestliže strom s konečným větvením je nekonečný (má nekonečnou hloubku), pak obsahuje nekonečnou cestu.

Důkaz: Důkaz je v podstatě technický. Musíme si uvědomit, že do nekonečné cesty musíme vybírat takové uzly x, které jsou kořenem nekonečného podstromu stromu T. □

Definice 1.7

Označený strom T je strom T spolu s funkcí (nazýváme ji *označující funkce*), která přiřazuje každému uzlu nějaký objekt, který nazýváme *označení*.

1.2 Výroky, spojky a pravdivostní tabulky

Tento odstavec je v podstatě opakováním notoricky známých věcí. Neškodí si je však připomenout.

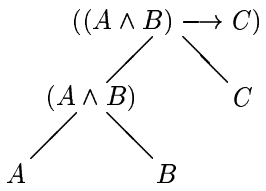
Definice 1.8
Jazyk výrokové logiky je tvořen:

- logické spojky: $\vee, \wedge, \neg, \longrightarrow, \longleftrightarrow$
- pomocné symboly: $(,)$
- výrokové symboly: $A, A_1, A_2, A_3, \ldots, B, B_1, B_2, B_3, \ldots, \ldots$

Definice 1.9
Výrok je definován indukcí:

1. výrokový symbol je výrok
2. jestliže α, β jsou výroky, pak $(\alpha \wedge \beta), (\alpha \vee \beta), (\neg\alpha), (\neg\beta), (\alpha \longrightarrow \beta), (\alpha \longleftrightarrow \beta)$ jsou výroky.
3. Řetězec symbolů je výrokem jen tehdy, pokud ho lze získat pomocí dvou předchozích pravidel.

Definice 1.10
Vytvářecí strom je strom, který „názorně" popisuje vznik výroku. Lépe to vystihuje obrázek: 1



Obrázek 1: Příklad vytvářecího stromu

Nyní si zopakujeme pravdivostní tabulky logických spojek:

α	β	$\alpha \wedge \beta$	$\alpha \vee \beta$	$\neg\alpha$	$\alpha \longrightarrow \beta$	$\alpha \longleftrightarrow \beta$
F	F	F	F	T	T	T
F	T	F	T	T	T	F
T	F	F	T	F	F	F
T	T	T	T	F	T	T

Definice 1.11
n-ární spojka je funkce σ, která přiřazuje výroku $\sigma(A_1, \ldots, A_n)$ ke každé n-tici výroků $A_1, \ldots, A_n$.
n-ární spojku nazýváme *pravdivostně funkcionální*, jestliže pravdivostní hodnota $\sigma(A_1, \ldots, A_n)$ je jednoznačně určena pravdivostními hodnotami $A_1, \ldots, A_n$.

Definice 1.12
Množina pravdivostně funkcionálních spojek je *postačující*, jestliže pro jakoukoli zadanou pravdivostní tabulku dokážeme najít výrok složený z těchto spojek, který má stejnou pravdivostní tabulku.

Věta 1.13
Množina $\{\neg, \wedge, \vee\}$ je postačující.

Důkaz: Je vcelku jednoduchý. Ukážeme si jej na příkladě. Máme nalézt vyjádření této neznámé ternární spojky:

A	B	C	?
T	T	T	T
T	T	F	F
T	F	T	F
T	F	F	F
F	T	T	T
F	T	F	F
F	F	T	F
F	F	F	T

Z pohledu na tuto tabulku vidíme, kdy hledaná funkce nabývá hodnot T (pravda). V našem případě jsou to 1., 5. a 8. řádek. Tedy výsledná funkce je logickým součtem podmínek, které je nutno splnit, aby hledaná funkce nabyla hodnoty pravda. Každá podmínka je konjunkcí požadavků na pravdivost všech výrokových symbolů A, B, C. Tedy hledaná funkce je:

$$? = (A \wedge B \wedge C) \vee (\neg A \wedge B \wedge C) \vee (\neg A \wedge \neg B \wedge \neg C)$$

□

Nyní si ukažme nám dosud neznámou *Shefferovu spojku* $|$:

α	β	$\alpha \mid \beta$
F	F	T
F	T	T
T	F	T
T	T	F

Věta 1.14
$\{\neg, \wedge\}, \{\neg, \vee\}, \{\neg, \longrightarrow\}, \{|\}$ jsou postačující množiny.

Definice 1.15
Pravdivostní přiřazení $\mathcal{A}$ je funkce, která přiřazuje každému výrokovému symbolu α jedinou pravdivostní hodnotu $\mathcal{A}(\alpha) \in \{T, F\}$.

Definice 1.16
Pravdivostní ohodnocení (valuace, interpretace) $\mathcal{V}$ je funkce, přiřazující každému výroku α jedinou pravdivostní hodnotu $\mathcal{V}(\alpha) \in \{T, F\}$.

Věta 1.17
Máme-li pravdivostní přiřazení $\mathcal{A}$, pak existuje jediné pravdivostní ohodnocení $\mathcal{V}$ takové, že $\mathcal{V}(\alpha) = \mathcal{A}(\alpha)$ pro libovolný výrokový symbol α.

Definice 1.18
Platná formule σ je taková výroková formule, že pro libovolnou valuaci $\mathcal{V}$ platí $\mathcal{V}(\sigma) = T$.
Formule σ je *důsledek* množiny formulí Σ (píšeme $\Sigma \models \sigma$), jestliže pro každou valuaci $\mathcal{V}$ platí:

$$\forall \tau \in \Sigma : \quad \mathcal{V}(\tau) = T \Longrightarrow \mathcal{V}(\sigma) = T$$

2 Tabulky ve výrokové logice

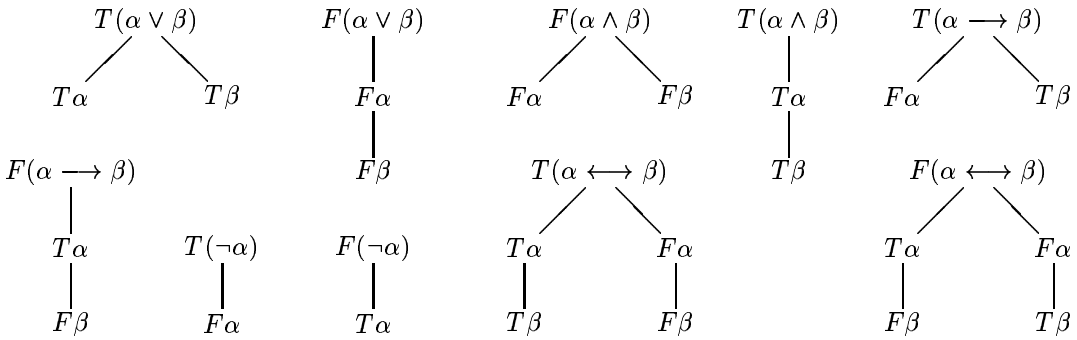
2.1 Tabulkové důkazy

Tabulkové důkazy, o nichž si zde budeme povídat, jsou stromy s označenými formulemi v uzlech. Probrat všechny možnosti pro důkaz formule není v podstatě možný (neuvažujeme matematickou proveditelnost, ale proveditelnost z hlediska automatizovaného dokazování formulí; zde uvedenou nemožnost chápeme jako extrémně vysokou složitost). Proto rozvíjíme strom s označenými formulemi tak, abychom dostali kontradikci.

Definice 2.1
Označená formule je $T\sigma$ nebo $F\sigma$ pro libovolnou formuli σ.

Definice 2.2
Atomické formule definujeme takto:

- FA a TA jsou atomické tabulky pro každý výrokový symbol A.
- Pokud jsou α, β výrokové formule, pak všechny atomické tabulky ukazuje obrázek 2.



Obrázek 2: Atomické tabulky

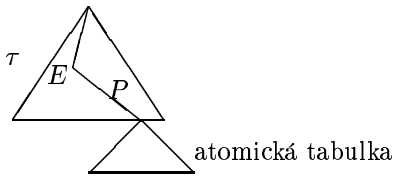
Definice 2.3
Konečnou tabulku definujeme takto:

- Každá atomická tabulka je konečná tabulka.
- Je-li τ konečná tabulka, P cesta v tabulce τ, E je uzel na P a τ' je tabulka, která vznikne z tabulky τ přidáním atomické tabulky s kořenem E na konec cesty P. Pak τ' je konečná tabulka.

Definice 2.4 TABULKA
Je-li $\tau_0, \tau_1, \ldots$ posloupnost konečných tabulek takových, že τ_{i+1} vznikne z tabulky τ_i postupem z předchozí definice, pak $\tau = \bigcup \tau_m$ je *tabulka*.

Příklad 2.1
Obrázek 3 ukazuje postup při tvorbě konečných tabulek.

Definice 2.5
Uzel E na cestě P je *redukovaný*, jestliže se na cestě P vyskytuje jako kořen atomické tabulky.
Cesta P je *kontradiktorická*, jestliže se na ní vyskytuje dvojice uzlů $T\alpha$ a $F\alpha$ pro nějaké α.
Tabulka τ se nazývá *ukončená*, jsou-li v ní na každé nekontradiktorické cestě všechny uzly redukované. Jinak je *neukončená*.
Tabulka τ se nazývá *kontradiktorická*, je-li v ní každá cesta kontradiktorická.

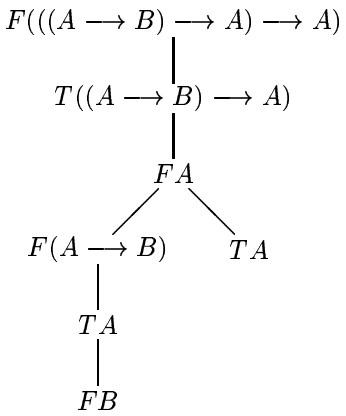


Obrázek 3: Definice konečné tabulky

Definice 2.6
Tabulkový důkaz formule α je kontradiktorická tabulka s kořenem $F\alpha$.
Tabulkové vyvrácení formule α je kontradiktorická tabulka s kořenem $T\alpha$.

Definice 2.7
Formule α je *tabulkově dokazatelná* (píšeme $\vdash \alpha$), jestliže existuje tabulkový důkaz formule α.

Příklad 2.2
Obrázek 4 ukazuje tabulkový důkaz $\vdash (((A \longrightarrow B) \longrightarrow A) \longrightarrow A)$.



Obrázek 4: Tabulkový důkaz $\vdash (((A \longrightarrow B) \longrightarrow A) \longrightarrow A)$

Definice 2.8
Úplná systematická tabulka (ÚST) je tabulka, vytvořená následujícím postupem:

- τ_0 je atomická tabulka s daným kořenem.
- Máme-li tabulku τ_m, pak nechť n je nejmenší úroveň v τ_m, na které existuje neredukovaný uzel na nekontradiktorické cestě. Buď E nejlevější uzel v této úrovni. Tabulka τ_{m+1} definujeme jako tabulku, ve které k τ_m přidáme atomickou tabulku s kořenem E na konec každé nekontradiktorické cesty procházející uzlem E.
- Úplná systematická tabulka je tabulka vzniklá jako sjednocení těchto tabulek:

$$\tau = \bigcup_m \tau_m$$

Lemma 2.9
Každá ÚST je ukončená.

Důkaz: Mám ÚST τ. Mám dokázat, že libovolný uzel E v ní je redukován. To je ale zřejmé, neboť uzel E leží někde na úrovni n v τ, nad ním a vlevo od něj je konečný počet m uzlů a tedy tabulka τ_{m+1} musí obsahovat už redukovaný uzel E. □

Lemma 2.10
Libovolná kontradiktorická tabulka τ je konečná (zejména tedy každá kontradiktorická ÚST je konečná).

Důkaz: Tvrzení lemmatu je důsledkem Königovy věty (věta 1.6), protože se jedná o konečně větvící se strom (konečné výrokové formule) s konečnými cestami (kontradiktorické cesty nejsou dále rozvíjeny). □

2.2 Korektnost a úplnost tabulkových důkazů

Korektnost je tvrzení $\vdash \alpha \Longrightarrow \models \alpha$. *Úplnost* je tvrzení $\vdash \alpha \Longleftarrow \models \alpha$.

Lemma 2.11
Nechť P je nekontradiktorická cesta v tabulce τ. Definujme valuaci $\mathcal{V}$ jako rozšíření přiřazení pravdivostních hodnot $\mathcal{A}$ daného tak, že $\mathcal{A}(p) = T$, pokud na P leží uzel Tp a $\mathcal{A}(p) = F$ v opačném případě. Pak $\mathcal{V}$ odpovídá všem prvkům na cestě P, tedy pro:

- $T\alpha$ na P, pak $\mathcal{V}(\alpha) = T$
- $F\alpha$ na P, pak $\mathcal{V}(\alpha) = F$

Důkaz: Je ideově bohatý asi tak jako marxismus. Provádí se indukcí vzhledem ke složitosti výrazu ≡ hloubce výrazu ≡ délce jeho generování. □

Důsledek 2.12
Tabulkové důkazy jsou korektní, tj. $\vdash \alpha \Longrightarrow \models \alpha$.

Důkaz: Tabulkovým důkazem formule α je kontradiktorická tabulka s kořenem $F\alpha$, tedy neexistuje $\mathcal{A}$ taková, že $\mathcal{V}(\alpha) = F$. Není-li $\models \alpha$, pak musí existovat valuace $\mathcal{V}$ taková, že $\mathcal{V}(\alpha) = F$. Toto je intuitivně naprosto jasné. Exaktně bychom se však museli pustit do rozboru všech logických spojek. □

Důsledek 2.13
Tabulkové důkazy jsou úplné, tj. $\models \alpha \Longrightarrow \vdash \alpha$.

Důkaz: $F\alpha$ rozvineme do ÚST. Dokážeme tvrzení sporem. Předpokládejme, že existuje nekontradiktorická cesta a přitom $\models \alpha$. Pak dle lemmatu 2.11 existuje přiřazení pravdivostních hodnot a odpovídající valuace $\mathcal{V}$, pro kterou platí $\mathcal{V}(\alpha) = F$, což je spor. □

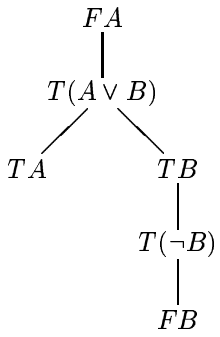
2.3 Dedukce z premis

Definice 2.14 (TABULKA Z PREMIS)
Buď Σ množina formulí. *Tabulka z premis* je tabulka definovaná tak, že:

1. Libovolná atomická tabulka je tabulka ze Σ.
2. Nechť τ je tabulka ze Σ, E je uzel na cestě P a τ' vznikne rozšířením τ tak, že na konec cesty P přidáme:
   (a) atomickou tabulku s kořenem E;
   (b) uzel $T\alpha$, $\alpha \in \Sigma$.

Definice 2.15
Tabulkový důkaz α z premis Σ je kontradiktorická tabulka ze Σ s kořenem $F\alpha$. Značíme $\Sigma \vdash \alpha$, jestliže existuje tabulkový důkaz formule α z premis Σ.

Příklad 2.3
Tabulkový důkaz $\{\neg B, A \vee B\} \vdash A$ uvádí obrázek 5.



Obrázek 5: Příklad tabulkového důkazu z premis

Definice 2.16 ÚST z premis
Nechť Σ je množina premis $\{\alpha_1, \alpha_2, \ldots\}$. Při tvorbě ÚST z premis střídáme dva kroky:

- totéž co u ÚST bez Σ (viz definice 2.8)
- V kroku $2n$ na konec každé nekontradiktorické cesty přidáme uzel $T\alpha_n$.

Věta 2.17
Každá ÚST z premis je konečná.

Věta 2.18 Korektnost
$\Sigma \vdash \alpha \implies \Sigma \models \alpha$

Důkaz: Totéž jako v důkaze korektnosti tabulkových důkazů, navíc stačí uvažovat valuace, které splňují Σ, odtud $T\alpha$. □

Věta 2.19 Úplnost
$\Sigma \models \alpha \implies \Sigma \vdash \alpha$

Důkaz: Opět totéž, navíc ještě ukončenost ÚST a fakt, že libovolná z premis může být použita. □

Věta 2.20 Kompaktnost
Platí-li $\Sigma \vdash \alpha$, pak existuje konečná $\Gamma \subseteq \Sigma$, pro kterou platí $\Gamma \vdash \alpha$.

Důkaz: Toto tvrzení je důsledek faktu, že v kontradiktorické ÚST může být použito jen konečně mnoho premis (z konečnosti ÚST). □

Věta 2.21
Nechť S je nekonečná množina formulí, pak platí: S je splnitelná $\iff$ je splnitelná každá její podmnožina.

Důkaz: Nechť $x_1, \ldots, x_i, \ldots \in S$.

$\Longrightarrow$: zřejmé

$\Longleftarrow$: Předpokládejme libovolné uspořádání prvků S do posloupnosti a z předpokladu splnitelnosti každé konečné podmnožiny množiny S platí splnitelnost každé množiny $\{x_i \mid i \leq n\}$ pro lib. $n \in \mathbf{N}$.

Vezmeme Tx_1 a expandujeme do ÚST. V této tabulce musí existovat aspoň jedna nekontradiktorická cesta. Nyní můžeme přidat Tx_2 na konec každé takové větve a znovu expandovat do ÚST.

Obecně v i-tém kroku můžeme vždy přidat x_{i+1} a takto dostáváme nekonečný strom. Z Königova lemmatu (věta 1.6) plyne existence nekonečné větve. To je větev, na níž leží všechny uzly Tx_i pro $\forall x_i \in S$ a tedy definuje interpretaci splňující S.

□

Definice 2.22
Říkáme, že formule x je *dedukovatelná* z množiny S, jestliže existuje konečný počet formulí $x_1, \ldots, x_n \in S$, pro které je formule $(x_1 \wedge x_2 \wedge \cdots \wedge x_n) \longrightarrow x$ tautologií.

Věta 2.23
Je-li x pravdivá ve všech valuacích, které splňují všechny formule z S, pak x je dedukovatelná z S.

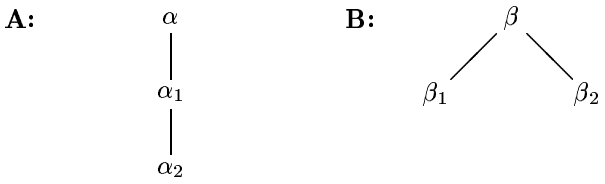
Důkaz: $S \cup \{\neg x\}$ je nesplnitelná (z předchozí věty). Existuje kontradiktorická ÚST s kořenem Fx, která je konečná. Tato tabulka je důkazem formule $(x_1 \wedge \cdots \wedge x_n) \longrightarrow x$. □

Poznámka 2.24
Odmyslíme-li si spojku $\longleftrightarrow$, máme dva druhy pravidel, nakreslené na obrázku 6. Typu **A** odpovídá tato tabulka:

α	α_1	α_2
$T(x \wedge y)$	Tx	Ty
$F(x \vee y)$	Fx	Fy
$F(x \longrightarrow y)$	Tx	Fy
$T(\neg x)$	Fx	Fx
$F(\neg x)$	Tx	Tx

a typu **B** odpovídá tato tabulka:

β	β_1	β_2
$F(x \wedge y)$	Fx	Fy
$T(x \vee y)$	Tx	Ty
$T(x \longrightarrow y)$	Fx	Ty



Obrázek 6: Typy pravidel

3 Rezoluce ve výrokové logice

Tuto záležitost objevil v roce 1961 J. A. Robinson. Zamyslel se a zjistil, že matematičtí logikové jsou trubky, když si myslí, že $X \wedge Y$ a $Y \wedge X$ jsou různé formule. Nyní si budeme myslet, že jsou to stejné formule a tedy v podstatě jako množiny výroků. Formule $X \wedge X$ tedy bude totožná s formulí X (obě jsou množiny $\{X\}$).

Pro tuto oblast budeme uvažovat úplnou množinu spojek $\{\wedge, \vee, \neg\}$. Spojku $\wedge$ budeme uvažovat v zobecněné formě:

$$< x_1, \ldots, x_n > \cong x_1 \wedge \cdots \wedge x_n$$

Podobně $\vee$ ve tvaru:

$$[x_1, \ldots, x_n] \cong x_1 \vee \cdots \vee x_n$$

Spojku $\neg$ budeme užívat ve tvaru $\bar{A}$. Dvojitá negace $\overline{\bar{A}}$ bude pro nás totéž jako samotná formule A.

3.1 Klauzulární tvar formulí

Definice 3.1

Literál l je výrokový symbol P nebo jeho negace $\neg P$, a to $\bar{l}$ pro $\neg P$ a l pro P. Literál s negací je *negativní*, bez ní je *pozitivní*.

Definice 3.2

Buď $\mathcal{M}$ konečná množina formulí. Definujeme další dva typy formulí:

- *zobecněná konjunkce* $< \mathcal{M} >$, pro kterou $\mathcal{V}(< \mathcal{M} >) = T$, jestliže pro $\forall \alpha \in \mathcal{M}$ je $\mathcal{V}(\alpha) = T$. $\mathcal{V}(<>) = T$. Označíme $<>$ jako `true`.
- *zobecněná disjunkce* $[\mathcal{M}]$, pro kterou $\mathcal{V}([\mathcal{M}]) = T$, jestliže $\exists \alpha \in \mathcal{M}$ taková, že $\mathcal{V}(\alpha) = T$. $\mathcal{V}([]) = F$. Označíme $[]$ jako `false`.

Notace: Formule $< \{x_1, \ldots, x_n\} >$ resp. $[\{x_1, \ldots, x_n\}]$ píšeme jako $< x_1, \ldots, x_n >$ resp. $[x_1, \ldots, x_n]$.

Definice 3.3

Klauzule C je zobecněná disjunkce množiny literálů.

Definice 3.4

Duální klauzule D je zobecněná konjunkce množiny literálů.

Definice 3.5

Formule S je formulí v *klauzulárním tvaru*, jedná-li se o zobecněnou konjunkci klauzulí, tj. $< C_1, \ldots, C_n >$.

Formule T je formulí v *duálním klauzulárním tvaru*, jedná-li se o zobecněnou disjunkci duálních klauzulí, tj. $< D_1, \ldots, D_n >$.

Definice 3.6

Přiřazení $\mathcal{A}$ je konzistentní množina literálů (tj. množina neobsahující zároveň pozitivní i negativní literál vytvořený z jednoho výrokového symbolu).

Úplné přiřazení je přiřazení, obsahující literál pro každý výrokový symbol.

Definice 3.7

(Úplné) přiřazení $\mathcal{A}$ *splňuje* formuli S v klauzulárním tvaru, jestliže pro $\forall C \in S$ platí $S \cap \mathcal{A} \neq \emptyset$.

Pojmy *splnitelnost / nesplnitelnost* jsou ekvivalentní pojmům existence / neexistence přiřazení, které formuli splňuje.

3.1.1 Alternativní notace

Tato notace je v podstatě à la Prolog. Mějme klauzuli $C \equiv [A_1, \ldots, A_m, \bar{B}_1, \ldots, \bar{B}_n]$. Je zřejmé, že C je formule $A_1 \vee \cdots \vee A_m \vee (B_1 \wedge \cdots \wedge B_n)$ a to je totéž jako $A_1 \vee \cdots \vee A_m \longleftarrow (B_1 \wedge \cdots \wedge B_n)$, pokud vezmeme spojku $\longleftarrow$ jako „obrácenou" implikaci, tj. $A \longleftarrow B \equiv B \longrightarrow A$. Programovací jazyk Prolog však používá místo znaku $\longleftarrow$ dvojznak :- a naše klauzule C nakonec vypadá $\underbrace{A_1, \ldots, A_m}_{\texttt{head}} :- \underbrace{B_1, \ldots, B_n}_{\texttt{body}}$. Slovně se tedy dá tato formule vyjádřit: Platí výroky tvořící hlavu (`head`), jestliže jsou splněny výroky tvořící tělo (`body`).

Rozlišujeme ještě tzv. *Hornovy klauzule*, což jsou takové klauzule, které mají v hlavě nejvýše jeden výrok. Mohou tudíž být:

- A nazývaná *fakt*. Výrok A prostě platí.
- $A \longleftarrow B_1, \ldots, B_n$ je *pravidlo*. Výrok A platí za předpokladu, že platí výroky $B_1, \ldots, B_n$.
- $\longleftarrow B_1, \ldots, B_n$ je *dotaz*, tj. „co plyne z toho, že platí $B_1, \ldots, B_n$?"

3.2 Převod formulí do (duálního) klauzulárního tvaru

Definice 3.8

Definujme relaci $\longrightarrow$ (od nynějška budeme spojku „implikace" značit znakem $\supset$) mezi formulemi následovně:

1. $< [(\alpha \wedge \beta)] > \longrightarrow < [\alpha], [\beta] >$ a $[< (\alpha \wedge \beta) >] \longrightarrow [< \alpha, \beta >]$
2. $< [\neg(\alpha \wedge \beta)] > \longrightarrow < [\neg\alpha, \neg\beta] >$ a $[< \neg(\alpha \wedge \beta) >] \longrightarrow [< \neg\alpha >, < \neg\beta >]$
3. $< [(\alpha \vee \beta)] > \longrightarrow < [\alpha, \beta] >$ a $[< (\alpha \vee \beta) >] \longrightarrow [< \alpha >, < \beta >]$
4. $< [\neg(\alpha \vee \beta)] > \longrightarrow < [\neg\alpha], [\neg\beta] >$ a $[< \neg(\alpha \vee \beta) >] \longrightarrow [< \neg\alpha, \neg\beta >]$
5. $< [(\alpha \supset \beta)] > \longrightarrow < [\neg\alpha, \beta] >$ a $[< (\alpha \supset \beta) >] \longrightarrow [< \neg\alpha >, < \beta >]$
6. $< [\neg(\alpha \supset \beta)] > \longrightarrow < [\alpha], [\neg\beta] >$ a $[< \neg(\alpha \supset \beta) >] \longrightarrow [< \alpha, \neg\beta >]$
7. $< [\alpha, \neg\alpha] > \longrightarrow <> =$ `true` a $[< \alpha, \neg\alpha >] \longrightarrow [] =$ `false`

Definice 3.9

1. Jestliže $< [M] > \longrightarrow < [N] >$, pak také

   $$< [M, M_1, \ldots, M_n], C_1, \ldots, C_m > \longrightarrow < [N, M_1, \ldots, M_n], C_1, \ldots, C_m >$$

   Jestliže $< [M] > \longrightarrow < [N_1], [N_2] >$, pak také

   $$< [M, M_1, \ldots, M_n], C_1, \ldots, C_m > \longrightarrow < [N_1, M_1, \ldots, M_n], [N_2, M_1, \ldots, M_n], C_1, \ldots, C_m >$$

2. Jestliže $[< M >] \longrightarrow [< N >]$, pak také

   $$[< M, M_1, \ldots, M_n >, C_1, \ldots, C_m] \longrightarrow [< N, M_1, \ldots, M_n >, C_1, \ldots, C_m]$$

   Jestliže $[< M >] \longrightarrow [< N_1 >, < N_2 >]$, pak také

   $$[< M, M_1, \ldots, M_n >, C_1, \ldots, C_m] \longrightarrow [< N_1, M_1, \ldots, M_n >, < N_2, M_1, \ldots, M_n >, C_1, \ldots, C_m]$$

3. Je-li $< [M] > \longrightarrow <>$, pak také

   $$< [M, M_1, \ldots, M_n], C_1, \ldots, C_m > \longrightarrow < C_1, \ldots, C_m >$$

4. Je-li $[< M >] \longrightarrow []$, pak také

$$[< M, M_1, \ldots, M_n >, D_1, \ldots, D_m] \longrightarrow [D_1, \ldots, D_m]$$

Algoritmus převodu do (duálního) klauzulárního tvaru spočívá v tom, že začneme od formule $< [M] >$ resp. $[< M >]$ a používáme pravidla pro tvorbu relace $\longrightarrow$ tak dlouho, dokud se s upravovanou formulí něco děje. Pokud ne, pak jsme již obdrželi kýžený (duální) klauzulární tvar.

Příklad 3.1

Máme převést formuli $A \vee (A \supset B)$ do klauzulárního tvaru. Provedeme to takto:

$$< [A \vee (A \supset B)] > \longrightarrow < [A, A \supset B] > \longrightarrow < [A, \bar{A}, B] > \longrightarrow <>$$

Nyní máme převést tutéž formuli do duálního klauzulárního tvaru. To provedeme naprosto stejně:

$$[< A \vee (A \supset B) >] \longrightarrow [< A >, < A \supset B >] \longrightarrow [< A >, < \bar{A} >, < B >]$$

Příklad 3.2

$$\begin{aligned} < [(A \vee (B \wedge C)) \supset (C \vee A)] > &\longrightarrow < [\neg(A \vee (B \wedge C)), (C \vee A)] > \longrightarrow \\ &\longrightarrow < [\bar{A}, (C \vee A)], [\neg(B \wedge C), (C \vee A)] > \longrightarrow \\ &\longrightarrow < [\bar{A}, C, A], [\bar{B}, \bar{C}, C, A] > \longrightarrow <> \end{aligned}$$

Příklad 3.3

Máme převést formuli $\neg((A \supset B) \vee \neg(C \supset B))$ do obou tvarů. Nejprve tedy do klauzulárního:

$$\begin{aligned} < [\neg((A \supset B) \vee \neg(C \supset B))] > &\longrightarrow < [\neg(A \supset B)], [C \supset B] > \longrightarrow \\ &\longrightarrow < [A], [\bar{B}], [\bar{C}, B] > \end{aligned}$$

A nyní do duálního klauzulárního:

$$\begin{aligned} [< \neg((A \supset B) \vee \neg(C \supset B)) >] &\longrightarrow [< \neg(A \supset B), C \supset B >] \longrightarrow \\ &\longrightarrow [< A, \bar{B}, (C \supset B) >] \longrightarrow \\ &\longrightarrow [< \bar{C}, A, \bar{B} >, < B, A, \bar{B} >] \longrightarrow [< A, \bar{B}, \bar{C} >] \end{aligned}$$

3.3 Formule v klauzulárním tvaru

V tomto odstavci si popíšeme metodu důkazů formulí v klauzulárním tvaru. Pod pojmem množina klauzulí budeme rozumět zobecněnou konjunkci těchto klauzulí. Množinou klauzulí S budeme tedy rozumět formuli v klauzulárním tvaru $< C \mid C \in S >$.

Definice 3.10

Je-li S množina klauzulí a l literál, pak S^l je definována:

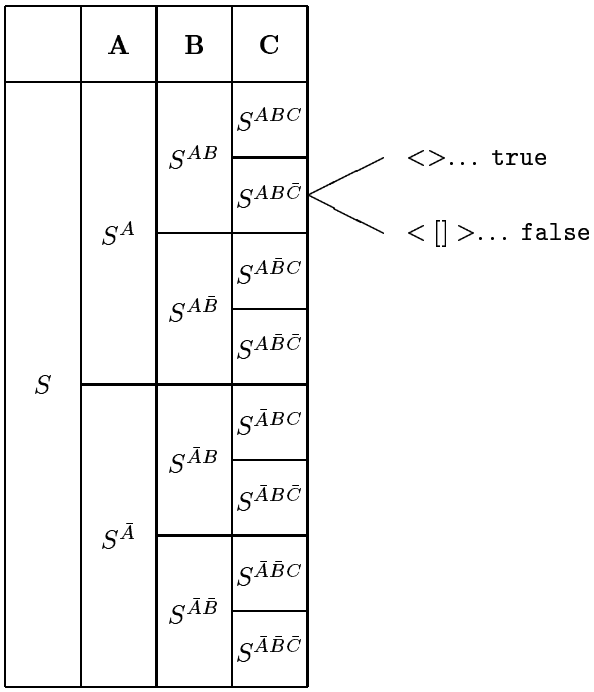
$$S^l = \{C - \{\bar{l}\} \mid C \in S, l \notin C\}$$

Máme-li množinu klauzulí $S =< [\ldots, l, \ldots], [\ldots], [\ldots] >$, pak první klauzule je jistě splněna (za předpokladu l). Pak tedy je zřejmé, že $S^l =< [\ldots], [\ldots] >$.

Obrázek 7 ilustruje indukci $\left(S^l\right)^m \equiv S^{lm}$.

Příklad 3.4

$$\begin{aligned} S &= < [A], [\bar{B}], [\bar{C}, B] > \\ S^A &= < [\bar{B}], [\bar{C}, B] > \\ S^{\bar{A}} &= < [], [\bar{B}], [\bar{C}, B] > \equiv < [] > \\ S^{AB} &= < [], [\bar{C}] > \equiv < [] > \\ S^{A\bar{B}} &= < [\bar{C}] > \\ S^{A\bar{B}C} &= < [] > \\ S^{A\bar{B}\bar{C}} &= <> \end{aligned}$$



Obrázek 7: Rozklad množiny klauzulí

Lemma 3.11
Buď l literál a S množina klauzulí. Platí: S je splnitelná, právě když je splnitelná buď S^l nebo $S^{\bar{l}}$.

Důkaz:

$\Longrightarrow$: Nechť máme přiřazení pravdivostních hodnot $\mathcal{A}$. Nechť $\mathcal{A} \models S$. Pak z definice platí pro $\forall C \in S$, že $C \cap \mathcal{A} \neq \emptyset$. To znamená že buď $l \in \mathcal{A}$ nebo (nonekvivalentně) $\bar{l} \in \mathcal{A}$.

- Nechť $l \in \mathcal{A}$. Ukážeme, že $\mathcal{A} \models S^l$.
  Pro $\forall C \in S$ platí: $C \cap \mathcal{A} \neq \emptyset$. Tedy:
  - $l \in C \Longrightarrow C \notin S^l$
  - $\bar{l} \in C \Longrightarrow C - \{\bar{l}\} \in S^l$ a přitom $\exists l' \in C, l' \in C \cap \mathcal{A}, l' \in C - \{\bar{l}\}$ a tedy i $C - \{\bar{l}\} \cap \mathcal{A} \neq \emptyset$.
  - $l, l' \in C, C \in S^l, C \cap \mathcal{A} \neq \emptyset$.

  Z těchto tří bodů tedy plyne, že $\mathcal{A} \models S^l$.
- $\bar{l} \in \mathcal{A}$ se dokáže symetricky vzhledem k l a $\bar{l}$.

$\Longleftarrow$: Předpokládejme, že $\mathcal{A} \models S^l$ (obdobně $\mathcal{A} \models S^{\bar{l}}$). Zvolme $\mathcal{A}' = (\mathcal{A} - \{\bar{l}\}) \cup \{l\}$. Musí platit $\mathcal{A}' \models S^l$. Ovšem máme $\mathcal{A}' \models S$, protože pro libovolnou klauzuli $C \in S$:

- $C \in S^l$ a bez problémů $C \cap \mathcal{A} \neq \emptyset$.
- $l \in C$, pak $C \cap \mathcal{A}' \neq \emptyset$.
- $C - \{\bar{l}\} \in S^l$, pak $C - \{\bar{l}\} \cap \mathcal{A}' = C \cap \mathcal{A}'$ $(\bar{l} \in C)$.

□

Důsledek 3.12
S je nesplnitelná, právě když S^l i $S^{\bar{l}}$ jsou nesplnitelné.

Definice 3.13
$UNSAT = \{S \mid S$ je nesplnitelná množina klauzulí $\}$.

Dohoda: Prázdnou klauzuli [] budeme někdy značit znakem $\Box$.

Věta 3.14
$UNSAT$ je nejmenší množina $\mathcal{M}$, která splňuje:

- Jestliže $\Box \in S$, pak $S \in \mathcal{M}$.
- Jestliže $S^l \in \mathcal{M}$ a $S^{\bar{l}} \in \mathcal{M}$, pak $S \in \mathcal{M}$.

(S je splnitelná, jestliže aspoň jedna z S^l, $S^{\bar{l}}$ je splnitelná, resp. S je nesplnitelná, právě když S^l i $S^{\bar{l}}$ jsou nesplnitelné.)

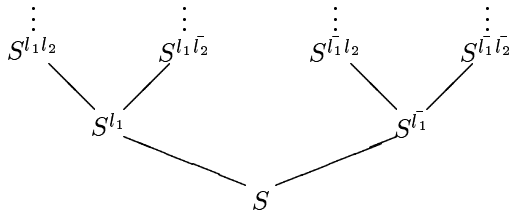
Důkaz:

a: $\mathcal{M} \subseteq UNSAT$

- $\star$ $S =< \ldots, \Box, \ldots >$, pak S je nesplnitelná.
- $\star$ $S^l \in UNSAT$, $S^{\bar{l}} \in UNSAT$, pak $S \in UNSAT$

Z výše uvedených dvou bodů máme $\mathcal{M} \subseteq UNSAT$.

- $UNSAT \subseteq \mathcal{M}$: Mějme $S \in UNSAT$. V S je konečný počet literálů, dejme tomu $l_1, \ldots, l_n$, $n \geq 0$:
  - $\star$ $n = 0$: tvrzení věty platí podle prvního bodu definice.
  - $\star$ $n > 0$: vybereme l_1 a uvažujeme $S^{l_1}, S^{\bar{l_1}} \in UNSAT$. Jestliže $S^{l_1} \notin \mathcal{M}$ (nebo $S^{\bar{l_1}} \notin \mathcal{M}$), pak existuje $S^{l_2}, S^{\bar{l_2}} \in \mathcal{M}$. Tímto způsobem rozvíjíme strom z obrázku 8.



Obrázek 8: Ilustrační stom k důkazu věty o množině UNSAT

Po konečném počtu kroků máme $S^{\delta_1 \ldots \delta_n}$, kde δ_i je buď l_i nebo $\bar{l}_i$. Pokud $S^{\delta_1 \ldots \delta_n} \neq <[]>$, pak literály $\delta_1 \ldots \delta_n$ definují přiřazení pravdivostních hodnot, které splňuje S, což je ovšem spor s předpokladem $S \in UNSAT$.

$\Box$

3.4 Rezoluce

Nyní již máme připraveny všechny prostředky k tomu, abychom si mohli začít povídat o mechanismu dokazování formulí v klauzulárním tvaru. Při této metodě dokazování vezmeme formuli S v klauzulárním tvaru a na ni postupně aplikujeme formální úpravy, které zachovávají splnitelnost formule S, a budeme zkoumat, zda dojdeme k $\Box$. Těmto formálním úpravám budeme říkat *rezoluce*.

Definice 3.15 REZOLUČNÍ PRAVIDLO
Buďte $C_1 = [p] \cup C_1'$ a $C_2 = [\bar{p}] \cup C_2'$ klauzule. Jejich *rezolventu* definujeme jako klauzuli $C = C_1' \cup C_2'$. C_1, C_2 nazveme *rodičovské klauzule*, C *dítě* a p *literál, na němž rezolvujeme*.

Výše popsané rezoluční pravidlo je vlastně ve svém skrytu pravidlo *modus ponens* známé z matematické logiky. Klauzule C_2 nám vlastně představuje pravidlo $p \supset C_2'$, což spolu s literálem p dává východiska pro pravidlo modus ponens, jehož výsledkem je klauzule C_2. Klauzulární tvar formulí nám však dovoluje toto pravidlo zobecnit, protože výsledky můžeme jednoduše sjednotit.

Definice 3.16

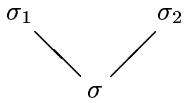
Rezoluční důkaz klauzule z množiny klauzulí S je konečná posloupnost klauzulí $C_1, \ldots, C_n = C$, kde pro $\forall i = 1, \ldots, n$ je $C_i \in S$ nebo rezolventou nějakých C_j a C_k, kde $j, k < i$.

Existuje-li rezoluční důkaz klauzule C z množiny klauzulí S, říkáme, že C je *rezolučně dokazatelná* z S. Píšeme $S \vdash_R C$.

Odvození prázdné klauzule $\square$ z S, $S \vdash_R \square$, se nazývá *vyvrácením* množiny klauzulí S.

Definice 3.17

Strom rezolučního důkazu klauzule C z množiny S je strom s C v kořeni, klauzulemi z S v listech, a kde každá část



znamená, že σ je rezolventa σ_1 a σ_2.

Příklad 3.5

Mějme literály / výrokové symboly: MgO, H_2, O_2, C, CO_2, H_2CO_3, H_2O. Mějme chemické rovnice:

$$\begin{aligned} MgO + H_2 &\longrightarrow Mg + H_2O \\ C + O_2 &\longrightarrow CO_2 \\ CO_2 + H_2O &\longrightarrow H_2CO_3 \end{aligned}$$

které nyní přepišme do formulí:

$$\begin{aligned} MgO \wedge H_2 &\supset Mg \wedge H_2O \\ C \wedge O_2 &\supset CO_2 \\ CO_2 \wedge H_2O &\supset H_2CO_3 \end{aligned}$$

a tyto převeďme do klauzulárního tvaru:

$$\begin{aligned} <[(MgO \wedge H_2) \supset (Mg \wedge H_2O)]> &\longrightarrow <[\neg(MgO \wedge H_2), (Mg \wedge H_2O)]> \longrightarrow \\ &\longrightarrow <[\overline{MgO}, \overline{H_2}, Mg], [\overline{MgO}, \overline{H_2}, H_2O]> \\ <[(C \wedge O_2) \supset CO_2]> &\longrightarrow <[\neg(C \wedge O_2), CO_2]> \longrightarrow <[\overline{C}, \overline{O_2}, CO_2]> \\ <[(CO_2 \wedge H_2O) \supset H_2CO_3]> &\longrightarrow <[\overline{CO_2}, \overline{H_2O}, H_2CO_3]> \end{aligned}$$

A nyní již můžeme rezolvovat a rezolvovat, jak to např. ukazuje obrázek 9.

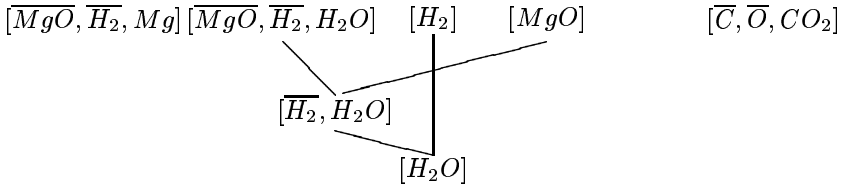
Lemma 3.18

Je-li množina $S =< [P \cup \{A\}], [\{\bar{A}\} \cup Q] >$ splnitelná, pak i $C \equiv [P \cup Q]$ je splnitelná a každá valuace splňující S splňuje i C.

Důkaz: Je-li S splnitelná, pak existuje přiřazení pravdivostních hodnot $\mathcal{A}$ takové, že $\mathcal{A} \models S$. Musí platit buď $\mathcal{A} \models A$ nebo $\mathcal{A} \models \bar{A}$:

- Pokud platí $\mathcal{A} \models A$, musí být i $\mathcal{A} \models Q$ (protože splňuje celou množinu S) a tedy $\mathcal{A} \models [P \cup Q]$.
- Analogicky pro $\mathcal{A} \models \bar{A}$.

Přitom navíc $\mathcal{A}$ definuje valuaci, která splňuje $[P \cup Q]$. $\square$



Obrázek 9: Příklad rezolucí v chemii

Věta 3.19 KOREKTNOST REZOLUCÍ
Existuje-li rezoluční vyvrácení množiny klauzulí S, pak S je nesplnitelná.

Důkaz: Plyne z definice vyvrácení a předchozího lemmatu. Mějme množinu S takovou, že se nám podařilo vyrezolvovat z ní $\square$. Dejme tomu, že existuje přiřazení pravdivostních hodnot $\mathcal{A}$ takové, že $\mathcal{A} \models S$. Pak ovšem taky $\mathcal{A} \models \square$, z čehož plyne, že takové $\mathcal{A}$ existovat nemůže. Čímž jsme dostali spor s tím, že S je splnitelná. $\square$

Důsledek 3.20
Existuje-li rezoluční strom s listy z množiny S a kořenem $\square$, pak S je nesplnitelná.

Důkaz: Plyne bezprostředně z korektnosti rezolucí. $\square$

Věta 3.21 ÚPLNOST REZOLUCÍ
Je-li S nesplnitelná, pak existuje rezoluční vyvrácení S.

Důkaz: Indukcí vzhledem k počtu literálů v množině S:

- žádný literál: $\square \in S$, zřejmé.
- n literálů: zvolíme jeden z nich, například l a víme, že S^l i $S^{\bar{l}}$ musí být prvky množiny UNSAT a obsahují $n-1$ literálů. Můžeme tedy předpokládat, že $S^l \vdash_R \square$ a $S^{\bar{l}} \vdash_R \square$. Z rezolučního stromu důkazu $S^l \vdash_R \square$ „přenesením" na klauzule množiny S dostaneme rezoluční strom důkazu $S \vdash_R X$, kde $X = \square$ nebo $X = [\bar{l}]$:
  - $X = \square$ jasné a konec důkazu.
  - $X = [\bar{l}]$: pak z důkazu $S^{\bar{l}} \vdash_R \square$ dostaneme stejným způsobem $S \vdash_R Y$, kde $Y = \square$ nebo $Y = [l]$. Je-li $Y = \square$, pak je to jasné a končím. Je-li $Y = [l]$, pak aplikuji jeden jediný rezoluční krok na X a Y (tedy $[\bar{l}]$ a $[l]$) a hned získávám jako jejich rezolventu $\square$ a tedy existuje rezoluční vyvrácení množiny S.

$\square$

3.5 Zjemění rezoluce

Zjeměním rezoluce budeme rozumět strategie tvorby rezolučního důkazu, aby byl co nejvíce efektivní (tj. aby co nejrychleji směřoval k $\square$, pokud je dokazovaná formule nesplnitelná). Jsou to zejména:

- Vyloučit z rezolucí všechny klauzule, které obsahují literál, který se v S vyskytuje jen v jedné paritě, (protože nemáme klauzuli s literálem v opačné paritě a tudíž nemůžeme tento literál vyrezolvovat a tudíž z klauzulí obsahujících tento literál nikdy nevyrezolvujeme $\square$).
- Odstranění tautologií, tj. klauzulí C takových, že existuje literál p : $p \in C \wedge \bar{p} \in C$ (např. klauzule: $[A, B, \bar{B}]$).
- Volba libovolného přiřazení pravdivostních hodnot $\mathcal{A}$ a odmítnutí rodičovských klauzulí, které jsou v něm splněny.

První strategie je zcela zřejmá včetně její korektnosti a úplnosti. Takže si v dalším odstavci budeme povídat o druhé strategii s odstraňováním tautologií.

3.5.1 T-rezoluce

Uvědomíme-li si, jak vypadají formule s tautologickými klauzulemi:

$$\{\underbrace{[\ldots],\ldots,[\ldots]}_{\text{tautologie}},\underbrace{\ldots}_{D}\}$$

pak protože každá tautologie obsahuje nějaký literál (dejme tomu p) v obou paritách, pak se tohoto literálu nemůžeme zbavit. Proto klauzule v množině D si musí při důkazu $\square$ „pomoci samy". Proč? Množinu tautologií můžeme brát jako množinu premis P a dokazovat formuli $P \supset \neg D$. Pokud je $P \supset \neg D \vdash_R \square$, pak musí být $D \vdash_R \square$, právě protože P je množina tautologií.

Definice 3.22
T-rezoluce jsou rezoluce, ve kterých žádná z rodičovských klauzulí není tautologií. Označíme $R^T(S)$ T-rezoluční uzávěr množiny S.

Věta 3.23 KOREKTNOST
Každé omezení rezoluce je korektní.

Důkaz: Je zřejmý. $\square$

Věta 3.24 ÚPLNOST T-REZOLUCE
Je-li $S \in UNSAT$, pak $\square \in R^T(S)$.

Důkaz: Tento důkaz můžeme vést jako důkaz úplnosti obecných rezolucí (důkaz věty 3.21). Stačí si pouze uvědomit, že při transformaci rezolučních stromů z důkazů množin S^l a $S^{\bar{l}}$ na důkazy množiny S do nich nepřidáme žádnou tautologii. Poslední důkazový krok (rezoluce klauzulí $[l]$ a $[\bar{l}]$) je rovněž T-rezoluce. $\square$

3.5.2 Sémantická rezoluce

Abychom si představili, co jsou sémantické rezoluce, představme si, že vybereme libovolnou interpretaci $\mathcal{A}$ a vyhodnotíme rodičovské klauzule rezoluce

$$\underbrace{C_0 \qquad\qquad C_1}_{C}$$

Pak platí implikace:

$$(\mathcal{A} \models C_0 \wedge \mathcal{A} \models C_1) \Longrightarrow \mathcal{A} \models C$$

Ovšem v *sémantických rezolucích* chceme takové interpretace $\mathcal{A}$ zakázat.

Definice 3.25
Buď $\mathcal{A}$ interpretace. *$\mathcal{A}$-rezoluce* je rezoluce, v níž vždy alespoň jedna z rodičovských klauzulí je v $\mathcal{A}$ nesplněná (tedy vyhodnocená do `false`). Označíme $R^{\mathcal{A}}(S)$ $\mathcal{A}$-rezoluční uzávěr množiny S.

Korektnost $\mathcal{A}$-rezoluce plyne již s tvrzení o korektnosti omezení rezoluce (věta 3.23).

Věta 3.26 ÚPLNOST SÉMANTICKÉ REZOLUCE
Pro libovolnou reprezentaci $\mathcal{A}$ a množinu klauzulí S platí: je-li $S \in UNSAT$, pak $\square \in R^{\mathcal{A}}(S)$.

Důkaz: Nepřítel nám zadal $\square \in R^{\mathcal{A}}(S^l)$ a $\square \in R^{\mathcal{A}}(S^{\bar{l}})$. My máme dokázat, že $\square \in R^{\mathcal{A}}(S)$.

Předpokládejme tedy $\mathcal{A}$-rezoluční stromy, které vyrezolvují $\square$ z formulí S^l (strom T_0) a $S^{\bar{l}}$ (strom T_1). Ztransformujeme nám již známým způsobem na $\mathcal{A}$-rezoluční stromy z formule S: $T_0 \longrightarrow T_0'$, resp. $T_1 \longrightarrow T_1'$.

Jeden z těchto stromů je opět $\mathcal{A}$-rezoluční a musí být buď $l \in \mathcal{A}$ nebo $\bar{l} \in \mathcal{A}$. Podle toho v příslušném stromě přidání $l/\bar{l}$ nezmění tak, aby platilo $\mathcal{A} \not\models C$ a přitom $\mathcal{A} \models C \cup [\sigma]$, kde σ je příslušné l nebo $\bar{l}$. Potom jeden ze stromů T_0' nebo T_1' musí být $\mathcal{A}$-rezoluční strom. Předpokládejme, že je to T_1' a nechť $C \in S^l$ je ze stromu T_0. Pak nastávají dvě možnosti:

- $C \in S$ a tedy není co řešit.
- $C \notin S$, pak klauzuli C odpovídá $C' \in S$, kde $C' = C \cup [l]$. Pak pomocí $\mathcal{A}$-rezolučního stromu T_1' vyrezolvujeme klauzuli $[\bar{l}]$ a s pomocí klauzule C' jedním $\mathcal{A}$-rezolučním krokem vyrezolvujeme C. To již máme C bez literálu l, z níž už můžeme ve stromě T_0 vyrezolvovat $\square$.

$\square$

3.6 Lineární rezoluce, Hornovské klauzule, Prologovská dedukce

Definice 3.27

- *Lineární (rezoluční) dedukce* klauzule $C \in S$, kde S je množina klauzulí, je posloupnost klauzulí $(C_0, B_0), \ldots, (C_n, B_n)$, $C_{n+1} = C$, kde C_i a B_j jsou klauzule a navíc platí:
  1. C_0 a všechna B_i jsou z množiny S nebo nějaké C_j takové, že $j < i$.
  2. Každá C_{i+1} je rezolventou C_i a B_i.
- Klauzule C je *lineárně odvoditelná* z S, jestliže existuje lineární dedukce klauzule C z S.
- Množinu S nazveme *množina vstupních klauzulí*, klauzule C_i nazveme *střední (průběžné,...) klauzule* a klauzule B_j nazveme *boční klauzule*.

Definice 3.28
Hornova klauzule je klauzule s nejvýše jedním pozitivním literálem.
Programová klauzule je klauzule s právě jedním pozitivním literálem.
Pravidlo je programová klauzule s negativními literály.
Fakt je programová klauzule bez negativních literálů.
Cíl je Hornova klauzule bez pozitivních literálů.

Lemma 3.29
Je-li S množina Hornových klauzulí nesplnitelná, pak S obsahuje alespoň jeden fakt a alespoň jeden cíl.

Důkaz: Využijeme důkaz úplnosti $\mathcal{A}$-rezoluce (důkaz věty 3.26). Uvažujme interpretaci $\mathcal{A}$, která obsahuje všechny výrokové symboly. Pak:

$\longrightarrow$: $\mathcal{A}$ splňuje každou programovou klauzuli a tedy v $\mathcal{A}$-rezolučním odvození má být použit cíl a tedy v množině S musí být cíl.

$\longrightarrow$: $\mathcal{A}$ splňuje každé pravidlo a dále splňuje každý cíl. V $\mathcal{A}$-rezolučním odvození musíme použít fakt, který musí být prvkem S.

□

Věta 3.30 Úplnost
Je-li S množina Hornových klauzulí, která je nesplnitelná, pak existuje lineární odvození □ z S.

Důkaz: Z kompaktnosti můžeme předpokládat, že S je konečná. Tvrzení dokážeme indukcí vzhledem k počtu literálů v množině S. Podle lemmatu 3.29 je v S alespoň jeden fakt, dejme tomu $[P]$. Uvažujme S^P. Každá klauzule v S^P je podmnožinou Hornovy klauzule. Množina S^P musí být rovněž nesplnitelná (kdyby $\mathcal{A} \models S^P$, pak by $(\mathcal{A} - \{\bar{P}\}) \cup \{P\} \models S$) a přitom má méně literálů než S.

Zbývá dokázat indukční předpoklad: existuje lineární vyvrácení S^P. Pokud máme lineární vyvrácení S^P, můžeme jej přetransformovat na lineární odvození klauzule $[\bar{P}]$ z množiny S. My ale víme, že $[P] \in S$ a můžeme jedním krokem vyrezolvovat □. □

Lemma 3.31
Je-li S množina Hornových klauzulí nesplnitelná, pak existuje rezoluční důkaz □ z S, ve kterém je v listech právě jedna cílová klauzule.

Důkaz: Lineární důkaz z důkazu předchozí věty (věty 3.30) má tuto vlastnost:

- cíl a cíl nemůže rezolvovat.
- cíl může rezolvovat s pravidlem nebo faktem, čímž dostaneme opět cíl nebo □.

□

Příklad 3.6

Definice 3.32
Nechť P je množina programových klauzulí a G cílová klauzule. *Lineární vstupní rezoluce (LI-rezoluce)* množiny $P \cup \{G\}$ je lineární vyvrácení $P \cup \{G\}$ začínající G s bočními klauzulemi z P.

Lemma 3.33
Je-li T množina Hornových klauzulí a G je cíl takový, že $T \cup \{G\} \in UNSAT$, ale $T \in SAT$[1], pak existuje lineární vstupní důkaz $\square$ z množiny $T \cup \{G\}$ začínající G.

Důkaz: Lineární rezoluční důkaz dokážeme z věty 3.30. $\square$

Věta 3.34 ÚPLNOST LI-REZOLUCE PRO HORNOVY KLAUZULE
Buď P množina programových klauzulí a G cílová klauzule. Je-li $S = P \cup \{G\} \in UNSAT$, pak existuje LI-rezoluční důkaz S.

Důkaz: Z předchozího lemmatu víme, že existuje vyvrácení začínající cílem G; po každém rezolučním kroku je rezolventa opět cílem a proto další rezoluční krok musí kombinovat tuto klauzuli s nějakou klauzulí z P. $\square$

Definice 3.35
Uspořádané klauzule (definite clauses) jsou konečné posloupnosti literálů.

Definice 3.36
Je-li $T \cup \{G\}$ dána jako množina uspořádaných klauzulí, pak *LD-rezoluční vyvrácení* $T \cup \{G\}$ je posloupnost $(G_0, C_0), \ldots, (G_n, C_n)$ uspořádaných klauzulí (G_i, C_i), kde $G \equiv G_0$ a $G_{n+1} \equiv \square$ a každá G_i je uspořádaná klauzule $\{\neg A_{i,0}, \ldots, \neg A_{i,i}, \ldots, \neg A_{i,n}\}$ a každá $C_i = \{B_i, \neg B_{i,0}, \ldots, \neg B_{i,m}\}$ je programová klauzule a pro každé $i < m$ existuje uspořádaná rezoluce G_i a C_i s rezolventou $\{\neg A_{i,0}, \ldots, \neg A_{i,k-1}, \neg B_{i,0}, \ldots, \neg B_{i,m}, \neg A_{i,k+1}, \ldots, \neg A_{i,n}\}$ a $A_{i,k} \equiv B_i$.

Lemma 3.37
Je-li $T \cup \{G\} \in UNSAT$, pak existuje LD-rezoluční vyvrácení $T \cup \{G\}$, které začíná klauzulí G.

Definice 3.38
Selekční pravidlo je funkce, která každé uspořádané klauzuli $\{A_1, \ldots, A_n\}$ přiřazuje číslo m, $1 \leq m \leq n$.

Definice 3.39
SLD-rezolučí vyvrácení množiny $P \cup \{G\}$ pomocí selekčního pravidla R je LD-rezoluční vyvrácení $(G_0, C_0), \ldots, (G_n, C_n)$, $G_0 \equiv G$ a $G_{n+1} \equiv \square$, kde $R(G_n)$ je index literálu z G_n rezolvovaný v n-tém kroku.

Věta 3.40 ÚPLNOST SLD-REZOLUCE PRO PROLOG
Je-li T splnitelná a $T \cup \{G\} \in UNSAT \implies$ existuje SLD-rezoluční vyvrácení $T \cup \{G\}$ pomocí R pro libovolné selekční pravidlo R.

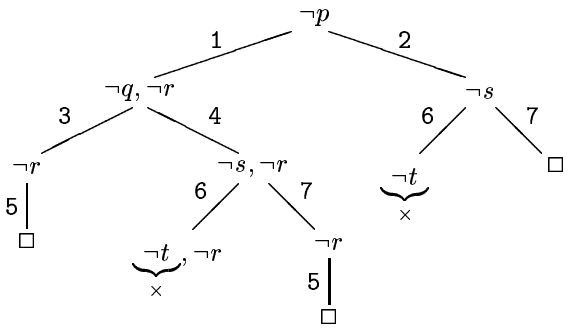
SLD-stromy: slouží k prohledávání stavového prostoru možných výběrů programových klauzulí. SLD-strom je strom, v jehož uzlech jsou průběžné rezolventy SLD-odvození a jehož bezprostřední následníci odpovídají rezolventám vzniklým použitím všech možných programových klauzulí. Větvení stromu reprezentují různé možnosti výběru programových klauzulí. V kořenu SLD-stromu je pak cíl, který startuje odvození.

Příklad 3.7
Mějme program:

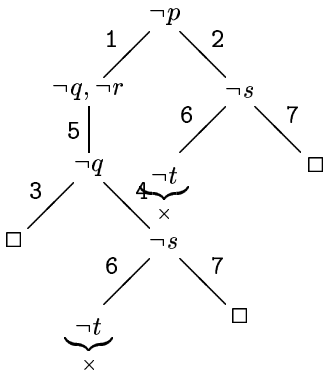
```
1.  p :- q,r.
2.  p :- s.
3.  q.
4.  q :- s.
5.  r.
6.  s :- t.
7.  s.
```

[1] Množinou SAT budeme mít na mysli množinový doplněk množiny $UNSAT$.

SLD-strom pro cíl ¬p a selekční pravidlo $R() = 1$ je uveden na obrázku 10 a SLD-strom pro ¬p a selekční pravidlo $R()$ =`poslední_literál` na obrázku 11.



Obrázek 10: SLD-strom k předešlému příkladu, $R() = 1$.



Obrázek 11: SLD-strom k předešlému příkladu, $R()$ =`poslední_literál`

Strategie prohledávání stromů:

- prohledávání do hloubky (≡ depth-first-search) – efektivní technika, datová struktura zásobník, z důvodu možnosti existence nekonečných větví není úplná.
- prohledávání do šířky (≡ breadth-first-search) – neefektivní (kombinatoricky exploduje počet stavů), ale je úplná.

3.7 Opakování

V tomto odstavci zadáme čtenáři domácí úlohu. Mějme tedy formuli:

„Jestliže nebude přijat nový zákon, bude stávka pokračovat, nebude-li přitom trvat déle než rok a neodstoupí-li ředitel."

Úkolem čtenáře je převést tuto formuli do klauzulárního tvaru a dokázat ji pomocí rezoluční metody.

4 Predikátová logika

Dosud jsme se zabývali automatickým dokazováním formulí ve výrokovém kalkulu. Nyní se však přesuneme na pole predikátové logiky. Tato kapitola je poněkud opakovací, ale doufáme, že nebude čtenáře nudit, neboť je krátká a ve svém závěru přinese i něco nového.

Než začneme s opakováním, poznamenejme, že predikátovou logiku budeme označovat zkratkou PL, případně PP (predikátový počet).

4.1 Opakování pojmů

Zásadním pojmem predikátové logiky je pojem predikátu. *Predikát* je n-ární relace, kterou značíme např. $p(\underbrace{\ldots}_{n})$.

Jazyk predikátové logiky tvoří *proměnné, konstanty, funkční symboly* $f(n)$ a *kvantifikátory* $\exists$, $\forall$.
Nad tímto jazykem pak definujeme pojmy *term* a *formule*. Do terminologie okolo formule patří pojmy jako:

- *podformule*,
- *výskyt proměnné*,
- *volné, vázané proměnné*,
- *uzavřené formule, podformule*,
- ...

To jsme si zopakovali syntaktické pojmy, nyní ještě dva sémantické:

- *obor interpretace (universum)*,
- *hodnota formule*.

4.2 Substituce proměnných

A nyní přichází jediná novinka této kapitoly. Je to substituce proměnných.

Definice 4.1
Substituce proměnných Θ je libovolné homomorfní zobrazení proměnných na termy (tj. zachovávající strukturu formule).

Pro substituci proměnných platí:

- $\Theta(E) = E$, kde E je konstanta.
- $\Theta(f(E_1, \ldots, E_n)) = f(\Theta(E_1), \ldots, \Theta(E_n))$, kde f je funkční n-ární symbol.
- $\Theta(E_1 \oplus E_2) = \Theta(E_1) \oplus \Theta(E_2)$, kde $\oplus$ je logická spojka ($\oplus \in \{\wedge, \vee, \Longrightarrow, \Longleftrightarrow\}$).

Dohoda 1 místo $\Theta(E)$ budeme substituci zapisovat $E\Theta$.

Dohoda 2 konečné substituce budeme zapisovat:

$$[x_1/E_1, \ldots, x_n/E_n]$$

a bude to znamenat, že term E_1 je substituován za proměnnou x_1 atd.

5 Tabulky v predikátové logice

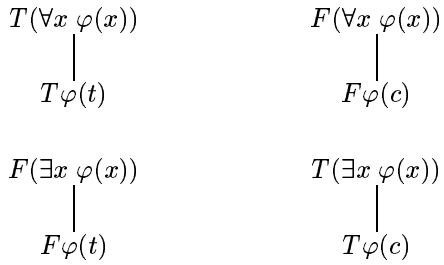
V této kapitole se budeme zabývat tabulkovými důkazy v predikátové logice. Jak záhy uvidíme, idea je takřka zcela totožná s ideou tabulkových důkazů ve výrokové logice. Musíme se jen technicky vypořádat s proměnnými.

5.1 Tabulkové důkazy

Definice 5.1 Atomická tabulka

Atomickou tabulku definujeme takto:

1. TA a FA jsou atomické tabulky pro libovolnou atomickou formuli A.
2. Atomické tabulky pro formule $TA_1 \wedge A_2$, $FA_1 \wedge A_2$, $TA_1 \vee A_2$ a $FA_1 \vee A_2$ definujeme stejně jako ve výrokové logice (viz obrázek 2).
3. Nechť t je libovolný základní term (tj. term bez proměnných) a nechť c je nový konstantní symbol. Pak obrázek 12 zobrazuje atomické tabulky kvantifikátorů.



Obrázek 12: Atomické tabulky kvantifikátorů

Definice 5.2 Konečné tabulky a tabulky

1. Každá atomická tabulka je *konečná tabulka*.
2. Je-li τ konečná tabulka, P je cesta v τ, E je uzel na P v τ a τ' se získá z τ přidáním atomické tabulky s kořenem E na konec cesty P, pak τ' je *konečná tabulka*.

   Požadavek nové konstanty v definici atomických tabulek kvantifikátorů znamená konstantu, která se nevyskytuje v žádném uzlu cesty P.
3. Je-li $\tau_0, \tau_1, \ldots$ posloupnost konečných tabulek, pak

$$\tau = \bigcup_i \tau_i$$

   je *tabulka*.

Poznámka 5.3

Přidání kořene atomické tabulky do tabulky je v případě predikátové logiky **podstatné** (pro atomické tabulky $T\,\forall$ a $F\,\exists$).

Definice 5.4

Cesta P je *kontradiktorická*, jestliže se na ní vyskytuje dvojice uzlů $T\alpha$ a $F\alpha$ pro nějaké α.

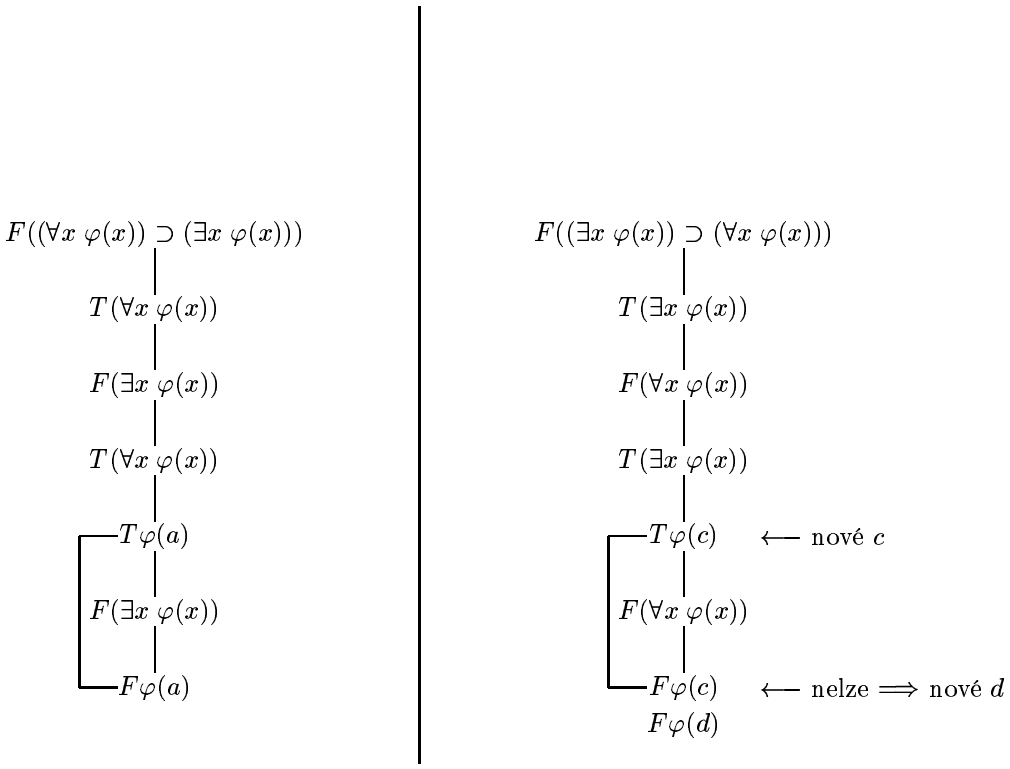
Tabulka τ se nazývá *kontradiktorická*, je-li v ní každá cesta kontradiktorická.

Tabulkový důkaz formule α je kontradiktorická tabulka s kořenem $F\alpha$.

Tabulkový důkaz α z premis Σ je kontradiktorická tabulka ze Σ s kořenem $F\alpha$. Značíme $\Sigma \vdash \alpha$, jestliže existuje tabulkový důkaz formule α z premis Σ.

Nekonzistentní množina formulí je taková množina S, že pro ni platí $S \vdash \alpha \wedge \neg\alpha$ pro nějaké α.

Příklad 5.1
Obrázek 13 ukazuje rozdíl mezi kontradiktorickou cestou a „zdánlivě" kontradiktorickou cestou.



Obrázek 13: Upřesnění pojmu kontradiktorické cesty v PL

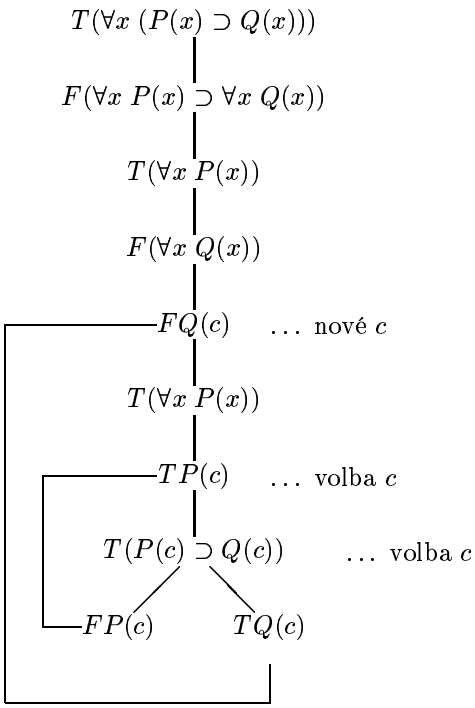
Příklad 5.2
Obrázek 14 nám ukazuje na složitější formuli jakousi „taktiku" jako nacházet kontradikce.
Na čtenáři necháme, aby si zkusil tabulkově dokázat formuli

$$(\forall x\ (P(x) \wedge Q(x)) \longleftrightarrow (\forall x\ P(x) \wedge \forall x\ Q(x)))$$

Jelikož je to první pokus, poradíme čtenáři, že je vhodné expandovat nejprve podle atomických tabulek, zavádějících nové konstanty a pak teprve použít pravidla, ve kterých vystupují obecné základní termy.

Definice 5.5 REDUKOVANÝ UZEL
Buď τ tabulka z množiny S, P cesta v τ, E uzel na P a w i-tý výskyt E na cestě P (i-tý uzel na P s označením w).

1. w je na P *redukovaný*, jestliže:
   (a) E není $T\ \forall x\ \varphi(x)$ ani $F\ \exists x\ \varphi(x)$ a E se vyskytuje na P v kořeni atomické tabulky.
   (b) E je tvaru $T\ \forall x\ \varphi(x)/F\ \exists x\ \varphi(x)$, $T\ \varphi(t_i)/F\ \varphi(t_i)$ je uzel na P a P obsahuje $i+1$-tý výskyt uzlu w.
2. Tabulka τ je *ukončená*, jestliže každý výskyt každého uzlu je redukovaný na každé nekontradiktorické cestě, která ho obsahuje (jinak je *neukončená*).



Obrázek 14: Příklad tabulkového důkazu

Definice 5.6 Úplná systematická tabulka z premis S

Konstruuje se stejně jako ve výrokové logice po úrovních (střídáme kroky expanze s přidáním $T\varphi$ pro $\varphi \in S$). Pro případy, kdy je atomická tabulka určena jednoznačně, případně se zavádí nová konstanta, to jde bez problémů. Pro tabulky s kořenem $T\ \forall x\ \varphi(x)$ nebo $F\ \exists x\ \varphi(x)$ a uzel w jako i-tý výskyt w na P přidáme tabulku:

$$\begin{array}{ccc} E & & E \\ | & \text{nebo} & | \\ T\varphi(t_i) & & F\varphi(t_i) \end{array}$$

s tím, že předpokládáme nějaké předem zvolené uspořádání všech termů v příslušném jazyce a i je index v tomto uspořádání.

Poznámka 5.7

Úplné systematické tabulky jsou obecně nekonečné, i když jsou ukončené.

Lemma 5.8

Každá ÚST je ukončená.

Důkaz: Důkaz vedeme jednoduchou úpravou důkazu z výrokové logiky (důkaz lemmatu 2.9). Platí, že

$$\tau_{UST} = \tau_0 \cup \tau_1 \cup \cdots \cup \tau_i \cup \cdots$$

a nad libovolným uzlem E (a vlevo od něj) je konečně mnoho uzlů a tedy v konečném počtu kroků dojde k redukci uzlu E. □

5.2 Korektnost a úplnost

Věta 5.9 Korektnost

Existuje-li tabulkový důkaz α z S (tj. $S \vdash \alpha$), pak $S \models \alpha$.

Důkaz: Spočívá v konstrukci interpretace, která souhlasí s každým prvkem na libovolné nekontradiktorické cestě. Nechť P je nekontradiktorická cesta. Indukcí vzhledem k její délce: předpokládejme, že pro cestu P to jde udělat; přidáme atomickou tabulku, která nezpůsobí kontradikci a ověříme, že příslušná interpretace splňující všechny uzly takto rozšířené cesty P' opět existuje. □

Věta 5.10
Je-li P nekontradiktorická cesta v úplné systematické tabulce τ z premis S s kořenem $F\alpha$, pak existuje interpretace nad vhodným universem, pro kterou je α nepravdivá a každá formule z S pravdivá.

Důkaz: viz úvaha v důkazu předchozí věty. Jedná se o konstrukci takové interpretace. □

Věta 5.11 Důsledek Königova lemmatu
Je-li ÚST kontradiktorická, je tato tabulka konečná.

Důkaz: Plyne z konstrukce ÚST. Je-li totiž kontradiktorická, pak neexistuje nekonečná větev. Každá cesta totiž musí být kontradiktorická a tudíž žádná není expandována do nekonečna. □

Kombinací věty 5.10 a důsledku Königova lemmatu 5.11 zde máme následující důsledek.

Důsledek 5.12
Pro každou formuli α a množinu formulí S platí právě jedno ze dvou následujících tvrzení:

- ÚST z S s kořenem $F\alpha$ je tabulkový důkaz formule α.
- Existuje nekontradiktorická cesta, která definuje interpretaci (a implicitně i universum), ve které je α nepravdivá a všechny formule z S pravdivé.

Tento důsledek má pro nás zcela zásadní význam, neboť všechna následující tvrzení plynou z tohoto důsledku.

Věta 5.13 Úplnost
Jestliže $S \models \alpha$, pak existuje tabulkový důkaz formule α z premis S.

Věta 5.14 Skolem – Löwenheimova
Je-li S splnitelná spočetná množina formulí predikátové logiky, pak existuje spočetný model.

[*Důkaz:* Probíráme universum a interpretaci ($\equiv$ model) z nekontradiktorické cesty generované z $S \Longrightarrow$ spočetně mnoho prvků, na kterých záleží. □
]

Věta 5.15 Kompaktnost
Je-li α dokazatelná z S, pak je dokazatelná i z nějaké její konečné podmnožiny.

Důkaz: Důkazem je konečná tabulka $\Longrightarrow$ byl použit konečný počet prvků z S. □

Věta 5.16 Jiná kompaktnost
S je splnitelná množina formulí $\Longleftrightarrow$ každá konečná podmnožina S je splnitelná.

Důkaz:

$\Longrightarrow$ je zřejmé.

$\Longleftarrow$ uvažujme tabulku dokazující formuli $F\alpha \wedge \neg\alpha$ z množiny S. Pak je tato tabulka konečná, právě když množina S je kontradiktorická.

□

6 Rezoluce v predikátové logice

6.1 Převod do klauzulárního tvaru

6.1.1 Kvantifikátory

Převod formulí do klauzulárního tvaru je naprosto stejný jako ve výrokové logice. Jediné s čím se musíme vypořádat jsou kvantifikátory. Abychom mohli z formule nadělat klauzule, musíme všechny kvantifikátory „vytknout" před formuli do tvaru

$$Qx_1 \dots Qx_n(\varphi)$$

kde φ je formule bez kvantifikátorů. Tento tvar formule nazýváme *prenexní tvar formulí*.

Lemma 6.1
Pro libovolný řetězec kvantifikátorů $\vec{Q}x = Q_1x_1 \dots Q_nx_n$, kde $Q_i \in \{\forall, \exists\}$ a formule φ, ψ jsou dokazatelné ekvivalence:

1. - $\vdash Q\vec{x}\ \neg\forall y\ \varphi \equiv \vec{Q}x\ \exists y\ \neg\varphi$
   - $\vdash Q\vec{x}\ \neg\exists y\ \varphi \equiv \vec{Q}x\ \forall y\ \neg\varphi$
2. - $\vdash Q\vec{x}\ (\forall y\ \varphi \vee \underbrace{\psi}) \equiv \vec{Q}x\ \forall z\ (\varphi[y/z] \vee \psi)$, kde z je nová proměnná.
   - $\vdash Q\vec{x}\ (\forall y\ \underbrace{\varphi}\ \vee\psi) \equiv \vec{Q}x\ \forall z\ (\varphi \vee \psi[y/z])$, kde z je nová proměnná.
   - $\vdash Q\vec{x}\ (\exists y\ \varphi \vee \underbrace{\psi}) \equiv \vec{Q}x\ \exists z\ (\varphi[y/z] \vee \psi)$, kde z je nová proměnná.
   - $\vdash Q\vec{x}\ (\exists y\ \underbrace{\varphi}\ \vee\psi) \equiv \vec{Q}x\ \exists z\ (\varphi \vee \psi[y/z])$, kde z je nová proměnná.
3. Zcela analogická pravidla jako pro spojku $\vee$ platí i pro spojku $\wedge$.

Důkaz: Například tabulkovým důkazem. □

Příklad 6.1

$$\begin{array}{c} \forall x\ \exists y\ P(x,y) \vee \neg\exists x\ \forall y\ Q(x,y) \\ \downarrow \\ \forall u\ \exists v\ \forall w\ \exists z\ (P(u,v) \vee \neg Q(w,z)) \end{array}$$

Důsledek 6.2
Libovolnou formuli predikátové logiky lze převést do ekvivalentní formule v prenexní normální formě, tj. $\vec{Q}x\ \varphi$, kde φ neobsahuje žádné kvantifikátory.

6.1.2 Funkce existenčního kvantifikátoru

Další věc, kterou pro získání klauzulí musíme provést je tzv. *skolemizace*, neboli odstranění existenčních kvantifikátorů.

Představme si, že máme splnitelnou formuli $\forall x\ \exists y\ P(x,y)$. Nyní máme funkční symbol $f(x) = y$, který si můžeme představit takto:

$$\begin{array}{lccccc} x: & 0 & 1 & \dots & n & \dots \\ & \downarrow & \downarrow & \dots & \downarrow & \dots \\ & P(0,y) & P(1,y) & \dots & P(n,y) & \dots \\ & \exists y? & \exists y? & \dots & \exists y? & \dots \end{array}$$

tedy je-li f nový funkční symbol, pak formule $\forall x\ P(x, f(x))$ je taky splnitelná. Takový funkční symbol f nazýváme *funkce existenčního kvantifikátoru*.

Tedy formuli $\forall x_1 \dots x_n\ \exists y\ \varphi$ upravíme na $\forall x_1 \dots x_n\ \varphi[y/f(x_1 \dots x_n)]$ s tím, že f bude nový funkční symbol.

Lemma 6.3
Pro libovolnou formuli $\varphi = \forall x_1 \dots \forall x_n\ \exists y\ \psi$ v jazyce $\mathcal{L}$ platí: formule φ a $\varphi' = \forall x_1 \dots \forall x_n\ \psi[y/f(x_1, \dots, x_n)]$, kde f je funkční symbol, který se nevyskytuje v jazyce $\mathcal{L}$, zachovává splnitelnost v jazyce $\mathcal{L}'$, což je jazyk vznikly přidáním symbolu f k funkčním symbolům jazyka $\mathcal{L}$.

Důkaz: Nechť φ' je splnitelná v $\mathcal{L}'$: $\mathcal{A}' \models \varphi'$ a $\mathcal{A}'(f(x_1, \ldots, x_n)) = a$, pak jistě existuje $\mathcal{A}$: $\mathcal{A} \models \varphi$ v $\mathcal{L}$ a $\mathcal{A}(y) = a$.

Naopak nechť φ je splnitelná v $\mathcal{L}$ a nechť máme množinu $\{\mathcal{A} \mid \mathcal{A} \models \varphi\}$, obsahující všechny takové interpretace. Tedy máme formuli $\forall x_1 \ldots \forall x_n\ \psi(y) \Longrightarrow \mathcal{A}'(f(x_1, \ldots, x_n)) = \mathtt{f}(\mathtt{x_1}, \ldots, \mathtt{x_n})$, kde $\mathtt{f}(\mathtt{x_1}, \ldots, \mathtt{x_n})$ je prvek Herbrandova universa, tedy $\varphi[y/f(x_1, \ldots, x_n)] \in \mathcal{A}' \Longrightarrow \mathcal{A}' \models \varphi'$ v jazyce $\mathcal{L}'$. □

Věta 6.4

Pro libovolnou formuli φ můžeme zkonstruovat obecnou formuli φ', tj. formuli, která obsahuje pouze všeobecně kvantifikované proměnné.

Důkaz: Důsledek předchozího lemmatu. Formule $\varphi \longrightarrow$ prenexní tvar $\longrightarrow$ zleva odstraňujeme $\exists$ náhradou skolemovskými funkcemi. □

Příklad 6.2

$$\begin{gathered}\forall u\ \exists v\ \forall w\ \exists z\ (P(u,v) \vee \neg Q(w,z)) \\ \downarrow \\ \forall u\ \forall w\ \exists z\ (P(u, f(u)) \vee \neg Q(w,z)) \\ \downarrow \\ \forall u\ \forall w\ (P(u, f(u)) \vee \neg Q(w, g(u,w)))\end{gathered}$$

6.2 Unifikace

Vzpomeňme si nyní, jak jsme prováděli rezoluce ve výrokové logice. Abychom totéž mohli provádět v predikátové logice musíme najít mechanismus, který by nám pomocí substitucí „ztotožnil" formule:

- $\neg P(x)$ a $P(y)$ pomocí změny proměnných;
- $\neg P(x)$ a $P(f(z))$ nalezením společné instance.

Tyto substituce musí být co nejobecnější, aby nezablokovaly další případné substituce. Takovou substituci nazveme *unifikátor*.

V literatuře [3] se na stranách 6 a dalších (kapitola 2.4 Unifikace) hovoří podrobně o celém problému unifikací. My jen vytáhneme to nejdůležitější. Dříve jsme se dohodli, že konečné substituce budeme značit $[x_1/\xi_1, \ldots, x_n/\xi_n]$.

Definice 6.5

(v textu [3] označená 2.10) *Unifikátor* množiny formulí S je substituce Θ taková, že $\mathrm{card}(S\Theta) = 1$, přičemž $S\Theta = \{\varphi\Theta \mid \varphi \in S\}$.

Definice 6.6

(v textu [3] označená 2.13) *Nejobecnější unifikátor* σ množiny S je takový unifikátor, že pro každý unifikátor Θ existuje vhodná substituce λ: $\Theta = \sigma\lambda$ (Θ je kompozicí zobrazení σ a λ).

Poznámka 6.7

Skládáme-li substituce $[x_1/\xi_1, \ldots, x_n/\xi_n][y_1/\xi_1, \ldots, y_m/\xi_m]$, pak pokud pro nějaké i: $x_i = y_i$ se pro toto i změna neprojeví.

Nyní stojíme před otázkou, zda existuje algoritmus hledající nejobecnější unifikátor, resp. rozhodující jeho existenci. O tom však v následujícím odstavci.

6.3 Robinsonův unifikační algoritmus

V tomto odstavci se opět budeme odvolávat na literaturu [3].

Předpokládejme, že bychom měli funkci $\mathtt{UNIFY}(A, B)$, která by vracela nejobecnější unifikátor dvou formulí. Pak bychom mohli jednoduše hledat nejobecnější unifikátor množiny formulí takto:

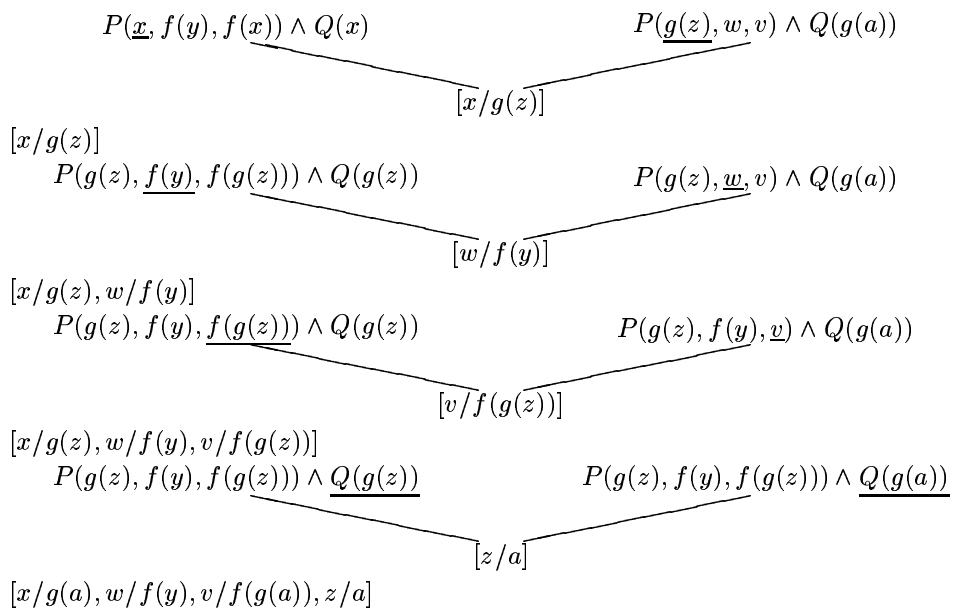
1. $\{A_1, \ldots, A_n\}$, $\mathtt{UNIFY}(A_1, A_2) \longrightarrow \Theta_1$.
2. $\{A_2\Theta_1, A_3, \ldots, A_n\}$, $\mathtt{UNIFY}(A_2\Theta_1, A_3) \longrightarrow \Theta_2$.

3. $\{A_3\Theta_2, A_4, \ldots, A_n\}$, $\texttt{UNIFY}(A_3\Theta_2, A_4) \longrightarrow \Theta_3$.

   ...

$(n-1)$. $\{A_{n-1}\Theta_{n-2}, A_n\}$, $\texttt{UNIFY}(A_{n-1}\Theta_{n-2}, A_n) \longrightarrow \Theta_{n-1}$; Θ_{n-1} je nejobecnější unifikátor celé množiny.

Jak bude postupovat nalezení nejobecnějšího unifikátoru dvou formulí nastiňuje následující obrázek 15.



Obrázek 15: Příklad unifikace $\texttt{UNIFY}(P(x, f(y), f(x)) \wedge Q(x), P(g(z), w, v) \wedge Q(g(a)))$

Definice 6.8
(v textu [3] označená 2.14) *Rozdíl* $\Delta(A, B)$ mezi výrazy definujeme induktivně takto: $\Delta(X, Y) =$

- $\emptyset$: pro $X = Y$ (strukturně totožné).
- $\Delta(X_1, Y_1) \cup \cdots \cup \Delta(X_n, Y_n)$, pro $X = f(X_1, \ldots, X_n)$ a $Y = f(Y_1, \ldots, Y_n)$, kde f je funkční nebo predikátový symbol nebo logická spojka.
- $\{(X, Y)\}$: v ostatních případech.

Příklad 6.3
Zde si uvedeme, jak by vypadal rozdíl formulí z obrázku 15.

$$\begin{aligned}\Delta(P(x, f(y), f(x)) \wedge Q(x), P(g(z), w, v) \wedge Q(g(a))) &= \Delta(P(x, f(y), f(x)), P(g(z), w, v)) \cup \Delta(Q(x), Q(g(a))) = \\ &= \Delta(x, g(z)) \cup \Delta(f(y), w) \cup \Delta(f(x), v) \cup \Delta(x, g(a)) = \\ &= \{(x, g(z)), (f(y), w), (f(x), v), (x, g(a))\}\end{aligned}$$

Definice 6.9 ZMENŠITELNÝ ROZDÍL
(v textu [3] označená 2.16) Rozdíl $\Delta(A, B)$ nazveme *zmenšitelný*, platí-li současně následující podmínky:

- $\Delta(A, B) \neq \emptyset$
- $\forall (U, V) \in \Delta(A, B)$ je v množině $\{U, V\}$ alespoň jedna proměnná, která se nevyskytuje ve druhé složce.

Je-li U proměnná, pak substituce $[U/V]$ se nazývá *zmenšením* rozdílu.

Příklad 6.4

- $\Delta(a, b)$, kde a, b jsou konstanty, je nezmenšitelný rozdíl.
- $\Delta(f(x), g(x))$ je nezmenšitelný rozdíl.

Nyní již máme připraveny všechny prostředky k tomu, abychom mohli napsat Robinsonův unifikační algoritmus:

```
UNIFY(A, B):
σ := ϵ;                                   {prázdná funkce}
while Δ(Aσ, Bσ) je zmenšitelný rozdíl
    do
       σ := σμ, kde μ je libovolná redukce rozdílu Δ(Aσ, Bσ)
    od
if Δ(Aσ, Bσ) = ∅
    then return(σ);
    else fail;
```

Věta 6.10
(v textu [3] označená 2.18) UNIFY(A, B) skončí a produkuje nejobecnější unifikátor.

Důkaz:

- ukončení: každá redukce rozdílu odstraní jednu proměnnou.
- nejobecnější unifikátor: viz text [3].

□

6.4 Rezoluce

Rezoluce v predikátové logice budou vypadat asi takto: máme rezolvovat formule $A \vee B$ a $\neg D \vee C$. Necháme si spočítat $\sigma = \mathtt{UNIFY}(B, D)$ a tedy máme $B\sigma = D\sigma$. Tedy můžeme z formulí $A\sigma \vee B\sigma$ a $\neg D\sigma \vee C\sigma$ vyrezolvovat formuli $A\sigma \vee C\sigma = (A \vee C)\sigma$.

Také lze z množin $\mathcal{A} \cup \{B\}$ a $\{\neg D\} \cup \mathcal{C}$ můžeme použitím téhož nejobecnějšího unifikátoru σ vyrezolvovat $\mathcal{A}\sigma \cup \mathcal{C}\sigma$. Přitom $\mathcal{A}\sigma = \{\alpha\sigma \mid \alpha \in \mathcal{A}\}$.

Dalším zobecněním našich úvah: nechť

$$\sigma = \mathtt{UNIFY}\{\underbrace{B_1, \ldots, B_m}_{\mathcal{B}}, \underbrace{\neg D_1, \ldots, \neg D_n}_{\mathcal{D}}\}$$

Pomocí σ můžeme vyrezolvovat z formulí $\mathcal{A} \cup \mathcal{B}$ a $\mathcal{D} \cup \mathcal{C}$ formuli $\mathcal{A}\sigma \cup \mathcal{C}\sigma$.

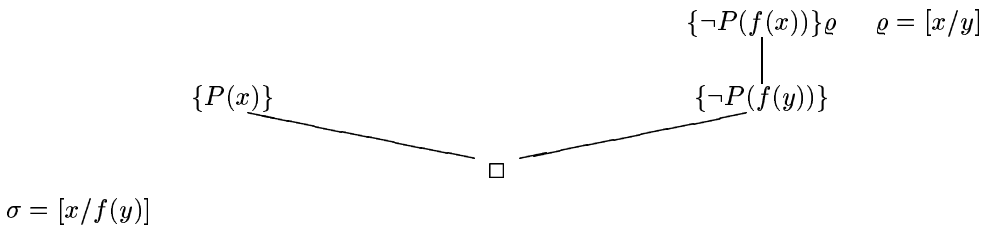
Pokud nyní máme formule $\forall x \ldots \mathcal{A} \cup \mathcal{B}$ a $\ldots \forall x\, \mathcal{D} \cup \mathcal{C}$. Nechť

$$\sigma = \mathtt{UNIFY}\{B_1, \ldots, B_m, \neg D_1\varrho, \ldots, \neg D_n\varrho\}$$

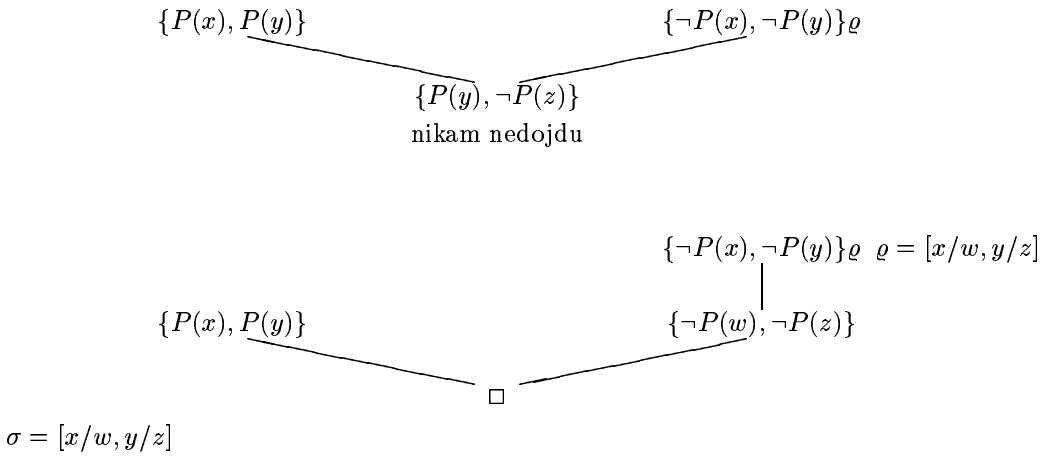
kde ϱ je přejmenování proměnných. Pomocí σ můžeme vyrezolvovat formuli $\mathcal{A}\sigma \cup \mathcal{C}\varrho\sigma$.

Definice 6.11
Mějme dvojici klauzulí C_1, C_2, v nichž můžeme přejmenovat proměnné pomocí substituce ϱ tak, že C_1 a $C_2\varrho$ nemají žádnou společnou proměnnou. Nechť $C_1 = C_1' \cup \{P(\vec{t_1}), \ldots, P(\vec{t_n})\}$ a $C_2\varrho = C_2' \cup \{\neg P(\vec{s_1}), \ldots, \neg P(\vec{s_m})\}$. Je-li σ nejobecnější unifikátor množiny $\{P(\vec{t_1}), \ldots, P(\vec{t_n}), \neg P(\vec{s_1}), \ldots, \neg P(\vec{s_m})\}$, pak klauzuli $C_1'\sigma \cup C_2'\sigma$ nazýváme *rezolventou* klauzulí C_1, C_2.

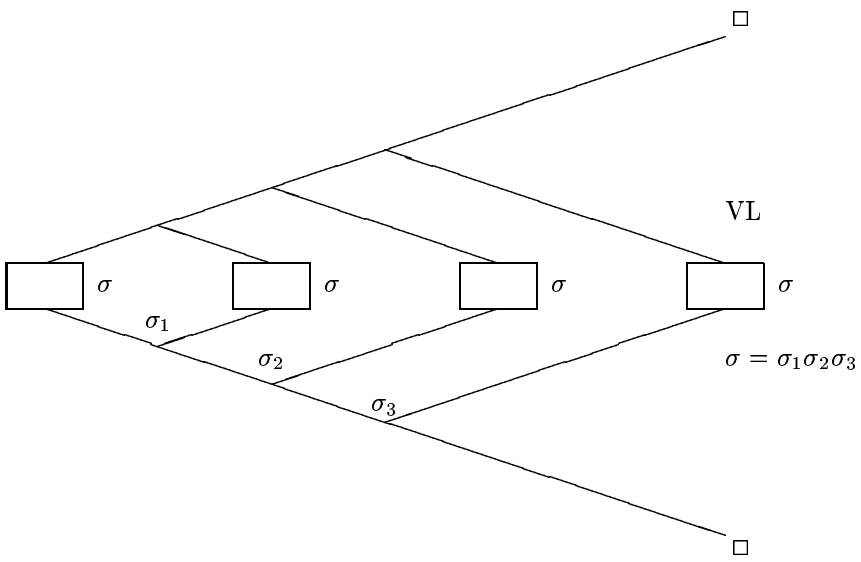
Definice 6.12
Rezoluční strom, *rezoluční důkaz*, *rezoluční vyvrácení*, ... se definují na základě pojmu rezolventy z předchozí definice.



Obrázek 16: Příklad přejmenování proměnných



Obrázek 17: Příklad faktorizace



Obrázek 18: Srovnání rezolucí ve VL a v PL

Příklad 6.5
Přejmenování proměnných při rezoluci klauzulí $\{P(x)\}$ a $\{\neg P(f(x))\}$ ukazuje obrazek 16.
Příklad *faktorizace* při rezoluci množin klauzulí $\{P(x), P(y)\}$ a $\{\neg P(x), \neg P(y)\}$ je zobrazen na obrázku 17.

Obrázek 18 ukazuje srovnání rezolucí ve výrokové logice a v predikátové logice.

Lemma 6.13 KOREKTNOST REZOLVENTY
Nechť C_1, C_2 jsou klauzule v predikátové logice a D jejich rezolventa, pak C_1 a C_2 jsou splnitelné $\Longleftrightarrow$ D je splnitelná.

Důkaz: Ponecháme jej čtenáři jako cvičení. □

Věta 6.14 KOREKTNOST REZOLUCE V PL
Jestliže existuje rezoluční vyvrácení množiny klauzulí S, pak S je nesplnitelná.

Úloha Pro zpestření zde uvedeme příklad od Lewise Carrola. Máme tato tvrzení:

1. *Nemám rád to, čeho se nedá použít jako mostu.*
2. *Všechno, o čem je možno napsat verše, bude pro mne dobrým dárkem.*
3. *Duha neudrží váhu vozu.*
4. *Co má sloužit jako most, musí vydržet váhu vozu.*
5. *Nevezmu jako dar nic, co nemám rád.*

Čtenář teď dostává za úkol odpovědět na tyto otázky:

? Odvoďte logický důsledek tvrzení 1. – 5., který neobsahuje kvantifikátor.

? Lze psát verše o duze?

6.5 Úplnost rezolucí

Úplnost rezolucí v predikátové logice se dokáže redukcí na výrokový případ.

Lemma 6.15 O POSUNUTÍ (obr. 18)
Jsou-li C_1', C_2' základní instance klauzulí C_1, C_2 definované $C_1' = C_1\Theta_1$ a $C_2' = C_2\Theta_2$ a nechť C' je rezolventa klauzulí C_1', C_2'. Pak existuje rezolventa C klauzulí C_1, C_2 taková, že C' je základní instance klauzule C.
Blíže myšlenku tohoto lemmatu zobrazuje obrázek 19.

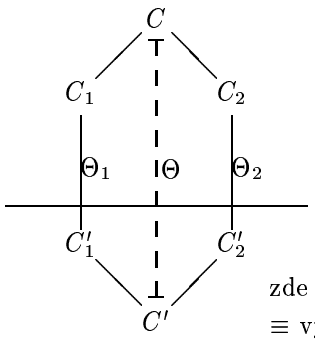
Důkaz: Důkaz je spíše technickou záležitostí, myšlenku samotnou ukazuje obrázek 19. Blíže je možno se dočíst v literatuře [3] na straně 10 a obrázku 7. □

Lemma 6.16
Nechť S je množina klauzulí v jazyce $\mathcal{L}$, S' množina všech základních instancí klauzulí z S prvky Herbrandova universa pro $\mathcal{L}$. Nechť C je libovolná klauzule jazyka $\mathcal{L}$, C' její základní instance v Herbrandově universu pro $\mathcal{L}$. Je-li T' rezoluční strom důkazu C' z množiny S', pak existuje rezoluční důkaz T klauzule C z množiny S a substituce Θ taková, že $T\Theta = T'$.

Příklad 6.6
Nechť $S = \{\{P(x)\}, \{\neg P(f(y))\}\}$. Z těchto dvou klauzulí pomocí substituce $[x/f(y)]$ můžeme ihned vyrezolvovat □. Množina S' vypadá takto:

$$\begin{aligned} S' \quad = \quad & \{P(a), P(b), P(c), \ldots \\ & P(f(a)), P(f(b)), P(f(c)), \ldots \\ & \ldots \\ & \neg P(f(a)), \neg P(f(b)), \neg P(f(c)), \ldots \\ & \neg P(f(f(a))), \neg P(f(f(b))), \neg P(f(f(c))), \ldots \\ & \ldots\} \end{aligned}$$

a v množině S' se už pohybujeme na poli výrokové logiky a máme kopec dvojic klauzulí, ze kterých můžeme □ ihned vyrezolvovat.



zde nejsou žádné proměnné, tedy
≡ výroková logika

Obrázek 19: Ilustrace lemmatu o posunutí

Důsledek 6.17
Úplnost rezoluční metody pro predikátovou logiku.

Literatura

[1] R. Smillian, *Logika prvého rádu*, ALFA

[2] J. Zlatuška, *Informační systémy* (1991)

[3] J. Zlatuška, *Logika, logické programování (unifikace) a paralelismus*, Zpravodaj ÚVT MU