Contents

0.1 Úvod

0.2 Prirodzená intuionistická logika

0.2.1 Formalizovanie dôkazu

Dôkaz z teórie usporiadania. Tak ako je Program = Proof

Otázka ohľadne konzistentnosti dôkazu.

0.2.2 Prirodzená dedukcia

Theorem 1 (Výroková premenná, formula)**.** *Majme spočítateľnú množinu $\mathcal{X}$ výrokových premenných. Množina výrokov alebo formúl $\mathcal{A}$ generovanú nasledovnou gramatikou:*

$$A, B ::= X | A \implies B | A \wedge B | A \vee B | \neg A | \top | \bot \tag{1}$$

Kde $X \in \mathcal{X}$ reprezentuje výrokovú premennú, a $A, B \in \mathcal{A}$ výrok.

V prípade nasledovného výroku je precedencia $\neg$ vyššia ako $\vee$ alebo $\wedge$ a tá je vyššia ako $\implies$. Binárne operátory sú asociatívne sprava.

$$\neg A \wedge B \wedge C \implies A \vee B$$
$$(\neg A \wedge (B \wedge C)) \implies (A \vee B)$$

Theorem 2. *Kontextom(systém predpokladov) rozuemieme zoznam výrokov značených*

$$\Gamma = P_1, \ldots, P_n \tag{2}$$

Dedukciou nazývame dvojicu pozostávajúcu z kontextu a výroku.

$$\Gamma \vdash A \tag{3}$$

Výraz čítame ako A je možné dokázať zo systému predpokladov Γ.

Theorem 3. *Dedukčné pravidlo pozostáva z množiny dedukcií Γ_i ktoré nazývame prepokladom. Dolnú časť dedukčného pravidla Γ nazývame záverom.*

$$\frac{\Gamma_1 \vdash A_1 \quad \ldots \quad \Gamma_n \vdash A_n}{\Gamma \vdash A} \tag{4}$$

Pravidlá prirodzenej intuicionistickej logiky:

$$\frac{}{\Gamma, A, \Gamma' \vdash: A} \text{ (ax)}$$

$$\frac{\Gamma \vdash A \implies B \quad \Gamma \vdash A}{\Gamma : B} (\implies_E) \qquad \frac{\Gamma, A \vdash B}{\Gamma : B} \implies_I$$

$$\frac{\Gamma, A \vdash B}{\Gamma : A} (\wedge^l_E) \frac{\Gamma, A \vdash B}{\Gamma : B} (\wedge^r_E) \qquad \frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B} (\wedge_I)$$

$$\frac{\Gamma \vdash A \vee B \quad \Gamma, A \vdash C \quad \Gamma, B \vdash C}{\Gamma \vdash C} (\vee_E) \qquad \frac{\Gamma \vdash B}{\Gamma \vdash A \vee B} (\vee^r_I) \frac{\Gamma \vdash A}{\Gamma \vdash A \vee B} (\vee^l_I)$$

$$\frac{\Gamma \vdash \neg A \quad \Gamma \vdash A}{\Gamma \vdash \bot} (\neg_E) \qquad \frac{\Gamma, A \vdash \bot \quad \Gamma \vdash A}{\Gamma \vdash \neg A} (\neg_I)$$

$$\frac{\Gamma \vdash \bot}{\Gamma \vdash A} (\bot_E)$$

V prípade že tieto pravidlá čítame zhora nadol hovoríme o dedukcii. Ak čítame pravidlá zdola nahor hovoríme o indukčnom spôsobe.

Theorem 4. *Fragmentom intuionistickej logiky nazývame, systém ktorý dostaneme ak ho obmedzíme len na niektoré z predchádzajúcich pravidiel.*

Theorem 5. *Implikačným fragmentom intuionistickej logiky dostaneme v prípade ak formuly budú tvorené gramatikou*

$$A, B ::= X | A \implies B \tag{5}$$

a pravidlami (ax), ($\implies_E$), ($\implies_I$)

V prípade že chceme aby výrokove formuly korenšpondovali s typmi ktoré su prezentované neskôr. Ich booleova reprezentácia s hodnotami $1, 0$ je nahradená otázkou existencie prvkov v množine. V prípade implikácie o existencii funkcie v množine. Funkcie v programoch ale môžu mať pri rovnakých vstupoch a výstupoch mať rôznu výpočtovú zložitosť. Dôvod prečo by sme sa mali pozerať na dôkazy(podľa publikácie Gir11) v troch rovinách.

- 1. Booleovský - tvrdenia sú booleovské hodnoty, zaujímame sa o dokázateľnosť tvrdenia
- 2. Existenčný - tvrdenia sú množiny, aké funkcie môžu byť
- 3. Úmyselný/Zámerový(Intentional) - zaujímame sa o zložitosť vytvoreného dôkazu a ako sa zjednoduší cez (cut eliminitation)

0.2.3 Intuicionizmus

Jedným zo smerov matematickej filozofie týkajucej sa rozvoja teórie je konštruktivizmus. Konštruktivizmus hovorí o potrebe nájsť alebo zostrojiť matematický objekt k tomu aby bola dokázaná jeho existencia. Jeden z motivačných príkladov takéhoto prístupu je možnosť dokázania pravdivosti výroku $p \vee \neg p$ cez dôkaz sporom $\neg p$ ktorý nehovorí ako zostrojiť objekt p len o jeho existencii. Tento smer tvorí viacero "škôl" okrem iných finitizmus, predikativizmus, intuicionizmus. Intuitionizmus je teda konštruktívny prístup k matematike v duchu Brouwera(1881-1966) a Heytinga(1898-1980). Filozofickým základom tochto prístupu princíp že matematika je výtvorom mentálnej činnosti a nepozostáva z výsledkov formálnej manipulácie symbolov ktoré sú iba sekundárne. Jedným z princípov intuicionizmus je odmietnutie tvrdenia postulátu klasickej logiky a to zákona vylúčenia tretieho.

$$p \vee \neg p \tag{6}$$

Dôvodom je z konštruktívneho pohľadu nezmyselnosť uvažovania nad pravdivosťou výroku nezávisle od uvažovaného tvrdenia. Výrok je teda pravdivý ak existuje dôkaz o jeho pravidovsti a nepravdivé ak existuje dôkaz ktorý vedie k sporu.

- konjukcii $p \wedge q$ ako o výroku hovoriacom o existencii dôkazov p a zároveň q,
- disjunkcii $p \wedge q$ ako existencii konštrukcii dôkazu jedného z výrokov p, q,

- $p \implies q$ je metóda(funkcia) transformácie každej konštrukcie p k dôkazu q,
- neexistencie dôkazu nepravdivého tvrdenia, iba dôkazu ktorý vedie k sporu $p \implies \perp$
- konštrukcia $\neg p$ je metóda ktorá vytvorí každú konštrukciu p na neexistujúci objekt

konjukcii $A \wedge B$ ako $A \times B$ $A \vee B$ ako $A \sqcup B$ disjunktne zjednotenie $\neg A = A \implies \perp$ existencie kontrapríkladu

0.3 Lambda kalkulus

Theorem 6. *Majme nekonečnú množinu $\mathcal{X} = x, y, z, \ldots$ ktorých elementy nazývame premenné. Množinu Λ tvorenú λ-termínmy je potom generovaná nasledovnou gramatikou:*

$$t, u ::= x|tu|\lambda x.t \tag{7}$$

Význam jednotlivých termínov je

$$\begin{aligned} x &\text{ - je premennou} \\ tu &\text{ - je aplikáciou termínu } t \text{ s argumentom } u \\ \lambda x.t &\text{ - je abstrakciou } t \text{ nad } x \end{aligned}$$

Príklady lambda termínov:

$$\begin{aligned} &tx \\ &(\lambda y.\lambda x.ty)) \\ &(\lambda y.yx)(\lambda x.x) \\ &tuv = (tu)v \end{aligned}$$

Aplikácia λ-termínov je implicitne aplikovaná zľava.

Pri výraze

$$\lambda x.tx = \lambda x.(tx) \tag{8}$$

je precedencia aplikácie vyššia ako abstrakcia.

A abstrakciu s troma argumentmi je možné prepísať do troch po sebe nasledujúcich.

$$\lambda xyz.t = \lambda x.\lambda y.\lambda z.t \tag{9}$$

Theorem 7. *Premenná x sa vo výraze*

$$\lambda x.t \tag{10}$$

abstrakciou viaže na termín t. O premennej x hovoríme že je viazaná. O premenných ktoré nie sú viazané sú voľné.

$$\begin{aligned} VP(x) &= x \\ VP(\lambda x.t) &= VP(t) \setminus \{x\} \\ VP(tv) &= VP(t) \cup VP(v) \end{aligned}$$

Theorem 8. *Premenovaním nazývame nahradenie voľných premenných v termíne.*

$$t\{y/x\} \tag{11}$$

V termíne t je premenovaná premenná x za y.

0.3.1 α-ekvivalencia

Theorem 9. *O výrazov hovoríme že sú alfa-ekvivalentné ak sa výrazy rovnajú až na premenovanie.*

Theorem 10. *O substutícii hovoríme pri nahradení jednej premenej druhou.*

$$t[y/x] \tag{12}$$

Nahradenie je silnejšie a vieme nahradiť aj premmenné viazanné abstrakciou.

0.3.2 β-ekvivalencia

$$\frac{}{(\lambda x.t)u \to_\beta t[u/x]}\ (\beta_s) \qquad \frac{t \to_\beta t'}{(\lambda x.t)u \to_\beta t[u/x]}\ (\beta_\lambda)$$

$$\frac{t \to_\beta t'}{tu \to_\beta t'u}\ (\beta_l) \qquad \frac{u \to_\beta u'}{tu \to_\beta tu'}\ (\beta_r)$$

$$\cfrac{\cfrac{\cfrac{}{(\lambda y.y)x \to_\beta x}\ (\beta_s)}{(\lambda y.y)xz \to_\beta xz}\ (\beta_l)}{\lambda x.(\lambda y.y)xz \to_\beta \lambda x.xz}\ (\beta_\alpha) \tag{13}$$

Theorem 11. *Definujme rekurziu volania funkcie nasledovne*

$$f^0 x = x \tag{14}$$

$$f^n x = f(f^{n-1}x) \tag{15}$$

$$\tag{16}$$

Potom Churchove číslo c_n je λ-termín

$$c_n = \lambda s.\lambda z.s^n(z) \tag{17}$$

Prirodzené čísla je potom definovať

$$0 = \lambda fx.x$$

$$1 = \lambda fx.fx$$

$$1 = \lambda fx.f(fx)$$

$$2 = \lambda fx.f(f(fx))$$

$$\begin{aligned} succ(n) &= (\lambda nfx.f(nfx))(\lambda fx.f^n x) \\ &\to_\beta \lambda fx.f((\lambda fx.f^n x)fx) \\ &\to_\beta \lambda fx.f((\lambda x.f^n x)x) \\ &\to_\beta \lambda fx.f(f^n x) \\ &= \lambda fx.f^{n+1}x \\ &= n+1 \end{aligned}$$

Operáciu sčítania je potom možné definovať vykonať

Theorem 12. $f_+ = \lambda x.\lambda y.\lambda s.\lambda z.xs(ysz)$

Podobným spôsobom môžeme vytvoriť

Theorem 13.

$$\begin{aligned} True &= \lambda xy.x \\ False &= \lambda xy.y \end{aligned}$$

$$if = \lambda bxy.bxy$$

$$\begin{aligned} if \text{ True } tu = (\lambda bxy.bxy)(\lambda xy.x)tu &\to_\beta (\lambda xy.(\lambda xy.x)xy)tu \\ &\to_\beta (\lambda y.(\lambda xy.x)ty)u \\ &\to_\beta (\lambda xy.x)tu \\ &\to_\beta (\lambda y.t)u \\ &\to_\beta t \end{aligned}$$

Theorem 14. *Jednoduchý λ kalkulus je ekvivalentný výpočtovej sile turingovho stroja. Bez dôkazu*

0.4 Typovo jednoduchý λ-calculus

Typový lambda calculus je rozšírením jednoduchého o typy

Theorem 15. *Majme množinu U spočítateľnú nekonečnú abecedu obsahujúcu typové premenné. Potom množina Π obsahuje reťazce jednoduchých typov ktoré su generované gramatikov:*

$$\Pi ::= U|(\Pi \to \Pi) \tag{18}$$

Theorem 16. *Kontextom rozumieme množinu C tvoriacu*

$$x_1 : \tau_1, \ldots, x_n : \tau_n \tag{19}$$

kde $\tau_1, \ldots, \tau_n \in \Pi$ *a* $x_1, \ldots, x_n \in$ *Koobor kontextu je množina obsahujúca*

$$domain(\Gamma) = x_1, \ldots, x_n \tag{20}$$

Oboor kontextu je množina obsahujúca

$$range(\Gamma) = \tau \in \Pi | (x : \tau) \in \Gamma \tag{21}$$

Príklady generované gramatikou

- $\vdash \lambda x.x : \sigma \to \sigma$
- $\vdash \lambda x.\lambda y.x : \sigma \to \tau \to \sigma$
- $\vdash \lambda x.\lambda y.\lambda z.xz(yz) : (\sigma \to \tau \to \rho) \to (\rho \to \tau) \to \sigma \to \rho$

Theorem 17. *Postupnosť je trojica značená*

$$\Gamma \vdash t : A \tag{22}$$

tvorená kontextom Γ*,* λ*-termínom* t *a typom* A*.*

Termín t je typu A ak v kontexte Γ ak je postupnosť derivovateľná pomocou pravidiel:

- ax: v kontexte x je typu A
- $\xrightarrow{I}$: ak je x typu A, t je typu B, potom funkcia $\lambda x.t$ ktorá asociuje x t je typu $A \to B$
- $\xrightarrow{E}$: daná je funkcia t je typu $A \to B$ a argument u je typu A, vysledok aplikácia tu je typu B

$$\frac{}{\Gamma \vdash x : \Gamma(x)}\ \text{ax}$$

$$\frac{\Gamma, x : A \vdash t : B}{\Gamma \lambda x^A.t : A \to B} \xrightarrow{I} \qquad\qquad \frac{\Gamma \vdash t : A \to B \quad \Gamma \vdash u : A}{\Gamma \vdash tu : B} \xrightarrow{E}$$

0.5 Curry-Howardov izomorfizmus

Intuinistická logika	Typovo jednoduchý λ kalkulus
termín	dôkaz
typová premenná	propozičná premenná

Theorem 18. *Curry-Howard isomorphism*

- *If* $\Gamma \vdash M : \varphi$ *potom* $|\Gamma| \vdash \varphi$.
- *If* $\Gamma \vdash \varphi$ *potom existuje* $M \in \Lambda_\Pi$ *také že* $\Delta \vdash M : \varphi$*, kde* $\Delta = (x_\varphi : \varphi) | \varphi \in \Gamma$

0.6 Počítačom asistované dokazovanie

To com som vravel v prezentacii, historia, na zaciatku sa pouzivalo

0.7 Lean dokazovací asistent

Lean je dokazovací asistent ktorý bol vytvorený otvorený softvérový projekt Leonardom de Mourom v Microsoft Reasearch v roku 2013. Jazyk sa neustále vyvýja a momentálne sa nachádza vo štvrtej iterácii [4] zatiaľ čo komunitný projekt matematickej knižnice mathlib sa stále vyvíja v tretej verzii[3] vyvýjanej od roku 2017. Implementácia Lean-u je v jazyku C++ a jeho jadro má 8000 riadkov. Prostredie je dostupné pre operačné systémy Linux, Windows a Darwin. Interaktívne prostredie pre dokazovanie je podporované pre *Emacs* a *Visual Studio Code*.

Lean podobne ako *Coq* je založené na kalkule konštrukcií ktorý je zovšeobecnením teórie jednoduchých typov a teórii závislostných typov.

0.7.1 mathlib

Mathlib je komunitný projekt[5] ktorého cieľom je združovať matematickú teóriu implementovanú v Lean-e. Do projektu je možné jednoducho prispievať po udelení privilégií niektorým zo správcov repozitára a odobrením požiadavky na začlenenie kódu. Väčšina obsahu mathlibu obsahuje matematiku na vysokoškolskej úrovni. V dobre písania práce je súborová hierarchia teórie nasledovná:

```
algebra/
category_theory/
data/
geometry/
measure_theory/
probability_theory/
system/
algebraic_geometry/
combinatorics/
deprecated/
group_theory/
representation_theory/
algebraic_topology/
computability/
dynamics/
linear_algebra/
number_theory/
ring_theory/
analysis/
control/
field_theory/
logic/
order/
set_theory/
topology/
```

V kontraste s inými modernými dokazovacími asistentami má mathlib množstvo prispievateľov akademické vzdelanie v čistej matematike[6] čo ovplyvnilo aj jeho obsah.

0.7.2 Constracting proof

0.7.3 Forward proofs

assume

calc

fix

have

let

show

0.7.4 Backward proofs

cc

clear

exact

induction

intro

refl

refl

Inductive types

0.8 Teória usporiadania

V tejto kapitole sa budeme snažiť ukázať možnosti Lean-u a využitie už existujúcich definícii v mathlibe pre dokázanie viet týkajúcich sa teórie usporiadania. Pre tento účel je Lean ideálny z pohľadu našich možností definovania vlastností usporiadania, ktoré následne možno aplikovať na abstraktnú množinu objektov. Výsledný typ je potom odvodený na základe závislostných typov. Usporiadanie je jednoducho intuitívne uchopiteľná vlastnosť bez matematických preddispozícií. V každodennom živote porovnávame svoju výšku, čas, ktorý trval na vybehnutie do kopca alebo aj číselne neohodnotené, subjektívne merateľné objekty ako ktorý album od skupiny preferujem. Na otázky si potom vieme odpovedať "ja som vyšší", "zabehol si pomalšie" alebo tieto albumy sú neporovnateľné.

Teória usporiadania sa snaží tieto vlastnoti formálne definovať a rozvíjať ďalej otázkami ako, aké je horné celej množiny objektov. Existuje ohraničenie horné alebo dolné pre ľubovoľnú podmnožinu objektov? Pre stručnosť sa v rámcii definícii obmedzíme len na definíciu usporiadania ako relácie, čiže podmnožinu karteziánskeho súčinu dvoch množín.

Theorem 19. *Majme množinu P, potom usporiadanie alebo čiastočné usporiadanie na množine P je binárna relácia $\leq$ taká že, pre všetky $x, y, z \in P$*

- $x \leq x$ *vlastnoť reflexivity*
- $x \leq y$ *a* $y \leq x$ *implikuje* $x = y$ *antisymetria*
- $x \leq y$ *a* $y \leq z$ *impikuje* $x \leq z$ *tranzitivita*

Ideálnym nástrojom pre uvažovanie nad usporiadaním sú *Hasseho* diagramy. Ako príklad uvádzame diagram "kocky".

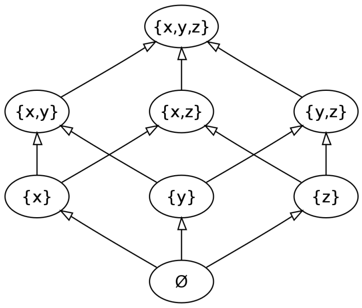

Figure 1: Usporiadania $\mathcal{P}(\{a, b, c\})$

Na obrázku je usporiadanie všetkých podmnožín trojprvkovej množiny $\{a, b, c\}$. Usporiadanie tvorí binárna relácia kardinality podmnožínm pričom porovnávame len podmnožiny obsahujúce spoločný prvok. Vyššie sú položené väčšie prvky a porovnateľnými prvkami považujeme len tie, ktoré sú "pokryté" jednosmernou cestou cez orientované hrany grafu.

V Leane je usporiadanie definované ako rozšírenie triedy predusporiadania, ktorá je reláciou, ktorá nemá oproti čiastočnému usporiadaniu vlastnosť antisymetrie.

```
class has_le       (α : Type u) := (le : α → α → Prop)
class has_lt       (α : Type u) := (lt : α → α → Prop)

class preorder (α : Type u) extends has_le α, has_lt α :=
(le_refl : ∀ a : α, a ≤ a)
(le_trans : ∀ a b c : α, a ≤ b → b ≤ c → a ≤ c)
(lt := λ a b, a ≤ b ∧ ¬ b ≤ a)
(lt_iff_le_not_le : ∀ a b : α, a < b ↔ (a ≤ b ∧ ¬ b ≤ a) . order_laws_tac) --
    toto chceme aj vysvetlit
```

Čiastočné usporiadanie je potom rozšírením predusporiadania o vlastnosť antysymetrie.

```
class partial_order (α : Type u) extends preorder α :=
(le_antisymm : ∀ a b : α, a ≤ b → b ≤ a → a = b)
```

DOPLN PRIKLAD NAJLEPSIE AK NAS USPORIADANY POSET Z PRIKLADU

Zväz

Zväz je usporiadaná množina, pre ktorú navyše platí, že pre každé 2 prvky a, b vieme nájsť prvok c, ktorý je ich jedinečným najmenším horným, respektíve(*supremum*) najväčším dolným ohraničením(*infinum*).

V prípade intervalu reálnych čísel je toto ohraničenie jednoducho predstaviteľné ako bod ohraničujúce množinu na číselnej osi. Ak ide o čiastočné usporiadanie, názov je pre tieto ohraničenia prvkov motivovaný zobrezním na grafe. *Spojenie* $\sqcup, \vee$ pre supremum, respektíve *priesek* $\sqcap, \wedge$ pre infinum. Popisnejším názvom pre zväz je preklad anglicky používaného názvu *lattice* "mriežka" tak isto motivovaná zobrazením takého usporiadania na grafe. Pri dokazovaní viet o zväzoch je často využívaná vlastnosť duality najmenšieho horného a duálne najväčšieho dolného ohraničenie pre druhú polovicu dôkazu. V prípade zväzu je táto vlastnosť využitá rovno v definícii zväzu ako spojenie duálnej definície suprémoveho a infinumového semizväzvu.

```
class has_sup (α : Type u) := (sup : α → α → α)
class has_inf (α : Type u) := (inf : α → α → α)

infix ⊔ := has_sup.sup
infix ⊓ := has_inf.inf

class semilattice_sup (α : Type u) extends has_sup α, partial_order α :=
(le_sup_left : ∀ a b : α, a ≤ a ⊔ b)
(le_sup_right : ∀ a b : α, b ≤ a ⊔ b)
(sup_le : ∀ a b c : α, a ≤ c → b ≤ c → a ⊔ b ≤ c)

class semilattice_inf (α : Type u) extends has_inf α, partial_order α :=
(inf_le_left : ∀ a b : α, a ⊓ b ≤ a)
(inf_le_right : ∀ a b : α, a ⊓ b ≤ b)
(le_inf : ∀ a b c : α, a ≤ b → a ≤ c → a ≤ b ⊓ c)

class lattice (α : Type u) extends semilattice_sup α, semilattice_inf α
```

Na nasledujúcich grafoch si ukážeme ako vyzerajú zväzy.

PRIDAT VLASTNY PRIKLAD, OPYTAT SA

```
def poset_nat : sublattice ℕ :=
    { carrier := {n : ℕ | 1 ≤ n},
    inf_mem :=
      ·begin
        intro a,
        intro b,
        intro a_set,
        intro b_set,
        simp at a_set,
        simp at b_set,
        simp,
        split,
        exact a_set,
        exact b_set,
      end,
    sup_mem := by finish,
}
```

Modulárne zväzy

V nasledujúcom úseku si ukážeme vetu týkajúcu sa špeciálneho typu zväzu s vlastnosťou modularity a ukážeme si formálny dôkaz a jej implementáciu v Leane, ktorú si podrobne rozoberieme.

O zväze L hovoríme, že je modulárny, v prípade, že má nasledujúcu vlastnosť.

$$(\forall x, y, z \in L)x \geq y \implies x \wedge (y \vee z) = (x \wedge y) \vee z$$

V Leane definovaný ako rozšírenie zväzu:

```
class modular_lattice(α : Type u) extends lattice α :=
  (modular_law: ∀ (x u v : α ), (x ≤ u) → u ⊓ (v ⊔ x) = (u ⊓ v) ⊔ x )
```

V nasledujúcom úseku si ukážeme vetu o modulárnom izomorfizme a podrobne si rozoberieme implementáciu jej dôkazu s obsahom prostredia v Leane.

TODO ZJEDNOTIT ZNACENIE DEFINICIE A LEAN-u

0.8.1 Modulárne zväzy

Theorem 20. Veta o izomorfizme modulárnych zväzov *Nech L je modulárnym zväzom a $a, b \in L$. Potom*

$$\varphi_b : x \mapsto x \wedge b, x \in [a, a \vee b], \tag{23}$$

Je izomorfizmom medzi intervalmi $[a, a \vee b]$ a $[a \wedge b, b]$. Inverzným izomorfizmom je

$$\psi_a : y \mapsto x \vee a, y \in [a \wedge b, b]. \tag{24}$$

Dôkaz. Stačí ukázať, že $\varphi_b\psi_a(y) = y$ pre všetky $x \in [a, a \vee b]$. Z duality vyplýva, že $\varphi_b\psi_a(y) = y$ pre všetky $y \in [a \wedge b, b]$, Majme $x \in [a, a \vee b]$. Potom $\psi_a\varphi_b = (x \wedge b) \vee a$ nerovnosť $a \leq x$ platí potom aj modularita

$$\varphi_a\psi_b(x) = (x \wedge b) \vee a = x \wedge (b \vee a) = x \tag{25}$$

pretože

$$x \leq a \vee b.$$ □

Predstavený dôkaz je znázornený na nasledujúcom grafe.

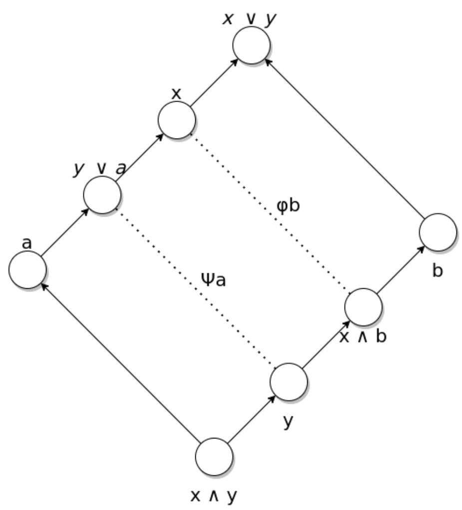

Figure 2: Izomorfizmus modulárneho zväzu

V prípade formálneho dôkazu sme sa mohli v časti dôkazu odkázať na dualitu. V prí návrhu dôkazu v Leane musíme ukázať dôkaz z ”oboch” strán.

```
theorem modular_lattice_isomorphism { α: Type u } [ modular_lattice α ]
{ u v w x y : α } :
  x ≤ u →
  x ≥ v →
  x ≥ u ⊓ v →
  x ≤ u ⊔ v →
  u ⊓ ( v ⊔ x ) = x ∧ (u ⊓ x) ⊔ v = x
  :=
  begin
          intros h1 h2 h3 h4,
          split,
          {
            rw modular_lattice.modular_law,
            exact sup_eq_right.mpr h3,
            exact h1
          },
          {
            rw inf_comm,
            rw ← modular_lattice.modular_law,
            exact inf_eq_left.mpr h4,
            exact h2
          }
  end
```

Začíname v taktickom móde prázdnou konštrukciou *begin* a *end* Interaktívne prostredie vyzerá nasledovne.

```
α: Type u
_inst_1: modular_lattice α
u v w x y : α
⊢ x ≤ u → x ≥ v → x ≥ u ⊓ v → x ≤ u ⊔ v → u ⊓ (v ⊔ x) = x ∧ u ⊓ x ⊔ v = x
```

Prvým krokom dôkazu je presunutie predpokladov zo sledu implikácii do prostredia pre ďalšiu prácu s nimi s označením $h1, h2, h3, h4$.

```
α: Type u
_inst_1: modular_lattice α
uvwxy: α
h1: x ≤ u
h2: x ≥ v
h3: x ≥ u ⊓ v
h4: x ≤ u ⊔ v
⊢ u ⊓ (v ⊔ x) = x ∧ u ⊓ x ⊔ v = x
```

Cieľ potom pozostáva z konjukcie, kde v druhej časti máme výraz implicitne ozátvorkovaný zľava. Výraz rozdelíme do dvoch podcieľov príkazom *split*, a pre lepšiu čitateľnosť ozátvorkujeme množinovými zátvorkami. Nachádzame sa v stave

```
begin
  intros h1 h2 h3 h4,
  split,
  {
  },
  {
  }
end
```

v ktorom nám lean ukazuje prostredie, kde musíme dokázať ľavú časť konjukcie.

```
⊢ u ⊓ (v ⊔ x) = x
```

Na cieľ použijeme z definície modulárneho zväzu vlastnosť modularity

```
(modular_law: ∀ (x u v : α ), (x ≤ u) → u ⊓ (v ⊔ x) = (u ⊓ v) ⊔ x )
```

a transformujeme prepíšeme cieľ cez príkaz

```
rw modular_lattice.modular_law,
```

na nasledujúci, kde má $u \sqcap v$ vyššiu precedenciu

```
⊢ u ⊓ v ⊔ x = x
```

Nasledujúca transformácia vyžaduje znalosť už dokázaných definícií, ktoré boli dokázané pre podkladové štruktúry. Použijeme nasledujúcu definíciu, ktorá vychádza z kontextu *semilattice_sup*.

```
% @[simp] theorem sup_eq_right : a ⊔ b = b ↔ a ≤ b :=    / TODO NEZABUDNUT
%  le_antisymm_iff.trans $ by simp [le_refl]           / ODKOMENTOVAT
```

Zaujímavosťou je, že si Lean dokáže substiuovať výraz $u \sqcap v$ za a z uvedeného výrazu. Pri použití vety dostávame ekvivalenciu, ktorá je definovaná ako štruktúra.

```
structure iff (a b : Prop) : Prop :=
    intro :: (mp : a → b)
             (mpr : b → a)
```

Z tejto štruktúry použijeme implikáciu smerujúca doľava nasledovne:

```
exact sup_eq_right.mpr h3,
```

Cieľ je teda transformovaný na:

```
⊢ x ≤ u
```

čo je už uvedený predpoklad $h1$. Týmto sme dokázali jeden z podcieľov. V tejto chvíli by sme sa v literatúre mohli odvolať na dualitu výrazov. V Leane musíme poskytnúť dôkaz aj o druhom cieli. Ideme dokázať

```
⊢ u ⊓ x ⊔ v = x
```

V tejto chvíli chceme znova použiť modularitu, leanu je, ale potrebné explicitne povedať, že chceme prepísať výraz nachádzajúci na pravej strane rovnosti pomocou symbolu ľavej šípky.

```
rw ← modular_lattice.modular_law,
```

Použijeme duálnu vetu duálnu k *sup_eq_right*.

```
@[simp] theorem inf_eq_left : a ⊓ b = a ↔ a ≤ b
```

a využijeme opačné predpoklady k predchádzajúcim $h2, h4$.

```
{
    rw ← modular_lattice.modular_law,
    exact inf_eq_left.mpr h4,
    exact h2
}
```

Po dokázaní druhého cieľa sme dokázali celú vetu. □

Bibliography

[1] Samuel Mimram, Program = Proof, Indenpendently published(July 3, 2020), ISBN-13: 979-8615591839

[2] Morten Heine B. Sørensen, Pawel Urzyczyn, Lectures on the Curry-Howard Isomorphism, Elsevier Science (April 4, 2013), ISBN-13 : 978-0444545961

[3] https://github.com/leanprover/lean

[4] https://github.com/leanprover/lean4

[5] https://github.com/leanprover-community/mathlib

[6] https://leanprover-community.github.io/papers/mathlib-paper.pdf

Slovicka na ktore nepoznam preklad a ne

- kalkul alebob kalkulus