66. Turingovy stroje (jazyky přijímané TS, varianty TS, lineárně omezené automaty, vyčíslitelné funkce).

Odkaz na video: [SZZ: Turingovy stroje a vyčíslitelné funkce]

Video pokrývá následující témata: *Deterministický Turingův stroj. Nedeterministický Turingův stroj. Úplný Turingův stroj. k-páskový Turingův stroj. Lineárně omezený automat. Jazyky přijímané Turingovými stroji. Převod nedeterministického Turingova stroje na deterministický Turingův stroj. Převod k-páskového Turingova stroje na jednopáskový Turingův stroj. Vyčíslitelné funkce. Počáteční funkce. Technika kombinace. Technika kompozice. Technika primitivní rekurze. Primitivně rekurzivní funkce. μ-rekurzivní funkce. Technika minimalizace. Parciálně rekurzivní funkce. Hierarchie funkcí.*

Deterministický Turingův stroj

Deterministický Turingův stroj je šestice $M = (Q, \Sigma, \Gamma, \delta, q_0, q_f)$, kde

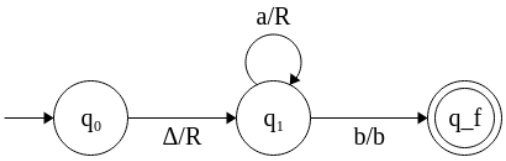
- Q je konečná, neprázdná množina stavů
- Σ je vstupní abeceda (konečná, neprázdná množina symbolů), přičemž $\Delta \notin \Sigma$
- Γ je pásková abeceda (konečná, neprázdná množina symbolů), přičemž $\Sigma \subset \Gamma \wedge \Delta \in \Gamma \wedge L, R \notin \Gamma$
- $\delta : (Q \setminus \{q_f\}) \times \Gamma \rightarrow Q \times (\Gamma \cup \{L, R\})$ je parciální přechodová funkce
- $q_0 \in Q$ je počáteční stav
- $q_f \in Q$ je koncový stav

Turingův stroj je výpočetní model, který podobně jako konečný automat slouží jako akceptor nějaké třídy jazyků. Lze rovněž využít pro vyčíslení funkce. Skládá se ze stavového prostoru, k němuž máme k dispozici nekonečnou paměť ve formě pásky, po níž je možné se libovolně pohybovat.

Turingův stroj zakreslíme grafově podobně jako konečný automat. Rozdíl je v tom, že využíváme právě jeden koncový stav a že přechody jsou ohodnoceny jiným způsobem. Zápis a/b poblíž přechodu, kde $a, b \in \Gamma$, značí, že pod hlavou stroje očekáváme symbol *a*, který bude po vykonání přechodu přepsán na *b*. Místo symbolu *b* se může v rámci přechodu vyskytovat jeden ze symbolů L, R, což značí vykonání kroku doleva/doprava na pásce (v tomto pořadí).

Na následujícím obrázku je vyobrazen Turingův stroj $M = (Q, \Sigma, \Gamma, \delta, q_0, q_f)$, kde

- $Q = \{q_0, q_1, q_f\}$
- $\Sigma = \{a, b\}$
- $\Gamma = \{a, b, \Delta\}$
- $\delta(q_0, \Delta) = (q_1, R)$, $\delta(q_1, a) = (q_1, R)$, $\delta(q_1, b) = (q_f, b)$



Ať $M = (Q, \Sigma, \Gamma, \delta, q_0, q_f)$ je Turingův stroj. **Konfigurace** Turingova stroje je trojice

$$(q, \gamma\Delta^\omega, n) \in Q \times \{\gamma\Delta^\omega \mid \gamma \in \Gamma^*\} \times \mathbb{N},$$

tedy stav výpočtu stroje, kde *q* značí stav, v němž se stroj momentálně nachází, *γ* je konečná předpona páskových symbolů (konečný prefix pásky) následovaná nekonečnou posloupností symbolů Δ a *n* je pozice čtecí hlavy na pásce.

Ať $M = (Q, \Sigma, \Gamma, \delta, q_0, q_f)$ je Turingův stroj. **Krok výpočtu** Turingova stroje je **nejmenší** binární relace mezi konfiguracemi $\vdash \subseteq (Q \times \{\gamma\Delta^\omega \mid \gamma \in \Gamma^*\} \times \mathbb{N}) \times (Q \times \{\gamma\Delta^\omega \mid \gamma \in \Gamma^*\} \times \mathbb{N})$, taková, že

$$\forall q, r \in Q : \forall n \in \mathbb{N} : \forall \alpha \in \Gamma^n : \forall \beta, \lambda \in \Gamma : \forall \gamma \in \Gamma^\omega :$$

- $\delta(q, \beta) = (r, R) \Rightarrow (q, \alpha\beta\gamma, n) \vdash (r, \alpha\beta\gamma, n + 1)$ **[krok doprava na pásce]**
- $(\delta(q, \beta) = (r, L) \wedge n > 0) \Rightarrow (q, \alpha\beta\gamma, n) \vdash (r, \alpha\beta\gamma, n - 1)$ **[krok doleva na pásce]**
- $\delta(q, \beta) = (r, \lambda) \Rightarrow (q, \alpha\beta\gamma, n) \vdash (r, \alpha\lambda\gamma, n)$ **[přepsání symbolu na pásce]**

Jinak řečeno, pásku si rozdělíme na tři části α, β, γ, přičemž část α má délku *n* symbolů, část β obsahuje jediný symbol a část γ je nekonečná. Hlava stroje je na pozici *n*, tedy stroj má pod hlavou právě symbol β (pásku indexujeme od nuly). Pokud je stroj ve stavu *q* a $\delta(q, \beta) = (r, R)$, pak se přesuneme do stavu *r*, páska zůstane nezměněná a *n* zvýšíme o 1, tedy hlava se posune doprava. Pokud je stroj ve stavu *q*, hlava není na indexu 0 (nejlevější buňka pásky) a $\delta(q, \beta) = (r, L)$, pak se přesuneme do stavu r, páska zůstane nezměněná a *n* snížíme o 1, tedy hlava se posune doleva. Pokud je stroj ve stavu *q* a $\delta(q, \beta) = (r, \lambda)$, pak se přesuneme do stavu *n*, symbol *β* pod hlavou změníme na symbol *λ* a neposuneme nikam hlavu stroje.

*Pozn.: Říkáme, že ⊢ je **nejmenší** binární relace splňující výše uvedené podmínky z toho důvodu, že jednotlivé podmínky jsou uvedeny ve tvaru implikace. Pokud je levá strana některé implikace splněna, pak dané dvě konfigurace jsou nutně v relaci ⊢. Pokud ovšem pro některé dvě konfigurace neplatí ani jedna z uvedených podmínek na levé straně implikace, není zřejmé, zda v relaci být mají, nebo ne. Výše uvedené podmínky tedy definují nekonečně mnoho různých relací ⊢. My z nich vybíráme tu **nejmenší** (vzhledem k relaci ⊆), neboť to je taková relace ⊢, která obsahuje právě ty dvojice konfigurací splňující výše uvedené podmínky a žádné jiné.*

Slovo nejmenší nebylo nutné uvádět u definici přechodové relace konečného/zásobníkového automatu, neboť v jeho případě jsme relaci $\vdash$ definovali pomocí ekvivalence.

Ať M = (Q, Σ, Γ, δ, q_0, q_f) je Turingův stroj. Mějme posloupnost konfigurací v relaci kroku výpočtu $\kappa_1 \vdash \kappa_2 \vdash \kappa_3 \vdash \dots$ Tato posloupnost může být:

- **nekonečná,** Turingův stroj **cyklí**
- **konečná,** Turingův stroj **zastaví**
  - **abnormálně**
    - pro poslední konfiguraci (q, αβγ, n), kde |α| = n, |β| = 1, není definováno δ(q, β), tedy stroj se zasekne, neboť nemůže provést žádný přechod a současně stav q není koncový
    - poslední konfigurace je (q, βγ, 0), kde |β| = 1, *q* není koncový stav a δ(q, β) = (r, L) pro nějaký stav *r*, tedy stroj se zasekne, neboť se nemůže z první pozice posunout doleva
  - **normálně**
    - poslední konfigurace je (q_f, γΔ$^\omega$, n), kde q_f je koncový stav

Ať M = (Q, Σ, Γ, δ, q_0, q_f) je Turingův stroj. **Jazyk přijímaný Turingovým strojem** je množina

$$L(M) = \{w \in \Sigma^* \mid (q_0, \Delta w \Delta^\omega, 0) \vdash^* (q_f, \gamma, n)\},$$

tedy množina takových řetězců, pro které platí, že pokud je umístíme na pásku (za jeden symbol Δ, na nějž umístíme hlavu stroje) a zbytek pásky bude obsahovat pouze symboly Δ, pak v konečném počtu kroků výpočtu dospějeme do koncového stavu, přičemž obsah pásky a pozice hlavy stroje budou libovolné. Pokud stroj na nějakém řetězci zastaví abnormálně, nebo cyklí, daný řetězec nepřijímá.

Varianty Turingových strojů

Nedeterministický Turingův stroj

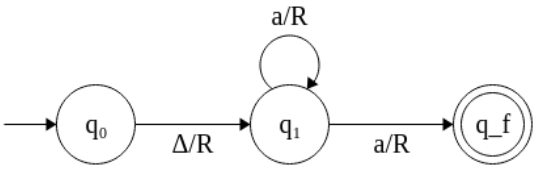
Nedeterministický Turingův stroj je šestice M = (Q, Σ, Γ, δ, q_0, q_f), kde

- Q je konečná, neprázdná množina stavů
- Σ je vstupní abeceda (konečná, neprázdná množina symbolů), přičemž $\Delta \notin \Sigma$
- Γ je pásková abeceda (konečná, neprázdná množina symbolů), přičemž $\Sigma \subset \Gamma \wedge \Delta \in \Gamma \wedge L, R \notin \Gamma$
- $\delta : (Q \setminus \{q_f\}) \times \Gamma \to 2^{Q \times (\Gamma \cup \{L, R\})}$ je parciální přechodová funkce
- $q_0 \in Q$ je počáteční stav
- $q_f \in Q$ je koncový stav

Na rozdíl od deterministického Turingova stroje se může stát, že z jednoho stavu vycházejí dva různé přechody, které vyžadují pod hlavou stroje stejný symbol. Za těchto okolností není zřejmé, který z těchto přechodů se má vykonat, vzniká zde možnost nedeterministické volby. Pro běh stroje nad jedním slovem zde tedy může existovat mnoho různých výpočetních větví.

Definice konfigurace, kroku výpočtu a jazyku přijímaného strojem zůstávají stejné. Pokud má tedy nějaký řetězec být strojem akceptován, je dostačující, aby alespoň jedna z výpočetních větví vedla do akceptujícího stavu.

V případě níže uvedeného Turingova stroje je nedeterminismus přítomný ve stavu q_1, kde není jednoznačně zřejmé, který z přechodů vykonat, pokud se pod hlavou stroje nachází symbol a.



k-páskový deterministický Turingův stroj

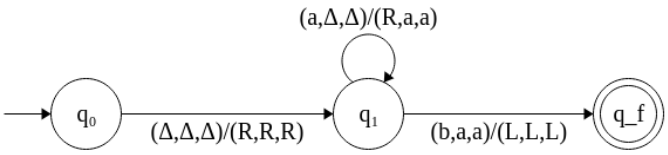
k-páskový deterministický Turingův stroj je (5+k)-tice $M = (Q, \Sigma, \Gamma_1, \Gamma_2, \ldots, \Gamma_k, \delta, q_0, q_f)$, kde

- Q je konečná, neprázdná množina stavů
- Σ je vstupní abeceda (konečná, neprázdná množina symbolů), přičemž $\Delta \notin \Sigma$
- $\Gamma_1, \Gamma_2, \ldots \Gamma_k$, jsou páskové abecedy (konečné, neprázdné množiny symbolů), přičemž $\Sigma \subset \Gamma_1, \Gamma_2, \ldots \Gamma_k \wedge \Delta \in \Gamma_1, \Gamma_2, \ldots \Gamma_k \wedge L, R \notin \Gamma_1, \Gamma_2, \ldots \Gamma_k$
- $\delta : (Q \setminus \{q_f\}) \times \Gamma_1 \times \Gamma_2 \times \ldots \times \Gamma_k \rightarrow Q \times (\Gamma_1 \cup \{L, R\}) \times (\Gamma_2 \cup \{L, R\}) \times \ldots \times (\Gamma_k \cup \{L, R\})$ je parciální přechodová funkce
- $q_0 \in Q$ je počáteční stav
- $q_f \in Q$ je koncový stav

Na rozdíl od jednopáskového deterministického Turingova stroje má k-páskový stroj k dispozici k pásek, příčemž každá z nich může mít vlastní páskovou abecedu. Při výpočtu lze s výhodou využít více pásek tak, že na každé pásce se uchovávají jiná data vyprodukovaná při výpočtu. Přechod je definován výchozím a cílovým stavem a odpovídajícími symboly pod každou hlavou stroje na všech páskách. Při provedení přechodu dojde také k odpovídající páskové operaci na každé z pásek.

Při zahájení běhu je vstup uložen na první pásce obvyklým způsobem, ostatní pásky jsou prázdné.

Vícepáskový Turingův stroj tedy umožňuje zpřehlednit a zrychlit výpočet. Je však třeba si uvědomit, že počet pásek je v rámci jednoho stroje vždy konstantní, tzn. nijak se nemění v závislosti na vstupu.



Očekávatelným způsobem lze definovat **k-páskový nedeterministický Turingův stroj.**

Úplný Turingův stroj

Ať M = (Q, Σ, Γ, δ, q_0, q_f) je Turingův stroj. Stroj M nazveme **úplným Turingovým strojem**, pokud pro každý svůj vstup vždy zastaví. Stroj tedy vždy v konečném počtu kroků výpočtu přejde do akceptujícího stavu, nebo zastaví abnormálním způsobem. Nikdy se však nestane, že by se zacyklil.

Lineárně omezený (ohraničený) automat

Ať M = (Q, Σ, Γ, δ, q_0, q_f) je Turingův stroj. Stroj M nazveme **lineárně omezeným (ohraničeným) automatem (LOA)**, pokud v rámci svého běhu nikdy neopustí tu část pásky, na níž je zapsán jeho vstup. Stroj může využívat konstantní počet pásek, ovšem i u každé z nich může využít pouze takovou jejich část, které odpovídá délce vstupního slova.

Alternativně lze lineárně omezený automat definovat jako Turingův stroj, který využívá maximálně takový počet buněk pásky, který je dán nějakou lineární funkcí f(|w|), kde |w| je délka vstupního slova Turingova stroje. Tyto definice jsou ekvivalentní, neboť konstantní počet pásek *k* uvedený v první definici lze chápat jako lineární člen této funkce $f(|w|) = k \cdot |w|$.

Transfromace variant Turingových strojů

1-páskový TS → k-páskový TS. Každý jednopáskový Turingův stroj je triviálně k-páskový. Není třeba provádět žádný převod.

DTS → NTS. Každý deterministický Turingův stroj je triviálně nedeterministický. Není třeba provádět žádný převod.

LOA →DLOA. K 17. červnu 2023 není známo, zda je možné každý nedeterministický lineárně omezený automat převést na ekvivalentní deterministický lineárně omezený automat.

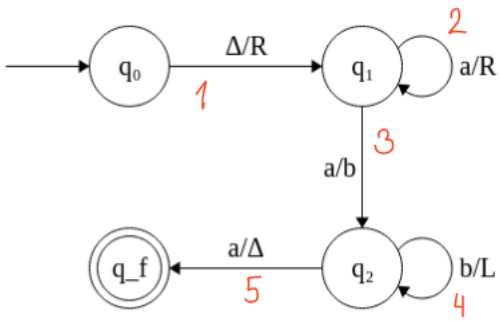
úplný TS → LOA. Nedeterministický lineárně omezený automat je slabší model než úplný Turingův stroj, tedy není možné každý úplný Turingův stroj převést na lineárně omezený automat.

obecný TS → úplný TS. Úplný Turingův stroj je slabší model než obecný Turingův stroj, tedy není možné každý obecný Turingův stroj převést na úplný Turingův stroj.

NTS →DTS. Každý nedeterministický Turingův stroj je možné převést na ekvivalentní deterministický Turingův stroj. Ukažme, jak by takový stroj pracoval. Postup je následující:

- Je zadán NTS M = (Q, Σ, Γ, δ, q_0, q_f) s *k* přechody a jeho vstup w ∈ Σ*
- Očíslujeme všechny přechody stroje M čísly z množiny {1, 2, …, k}
- Postupně generujeme veškeré posloupnosti čísel z množiny {1, 2, …, k} v lexikografickém pořadí, tzn. začneme posloupnostmi délky 1, dále pokračujeme posloupnostmi délky 2 apod.
- Pro každou vygenerovanou posloupnost otestujeme, zda by NTS s daným vstupem při provedení dané posloupnosti přechodů dospěl do koncového stavu. Pokud by nebylo možné některý přechod provést, nebo pokud by byly všechny přechody dané aktuální posloupností provedeny, ale nevedly by do koncového stavu, přejdeme k další posloupnosti
- Pokud by některá posloupnost přechodů vedla do koncového stavu, DTS akceptuje
- Takto DTS akceptuje tentýž jazyk jako NTS M, přičemž je deterministický, neboť přistupuje k procesu systematicky. **Stroj ovšem cyklí na všech řetězcích, které nepřijímá.**

Demonstrujme si tento postup na následujícím Turingově stroji se vstupem w = aab. Očíslujeme jednotlivé přechody. A generujeme posloupnosti **1, 2, 3, 4, 5, 11, 12, 13, 14, …**

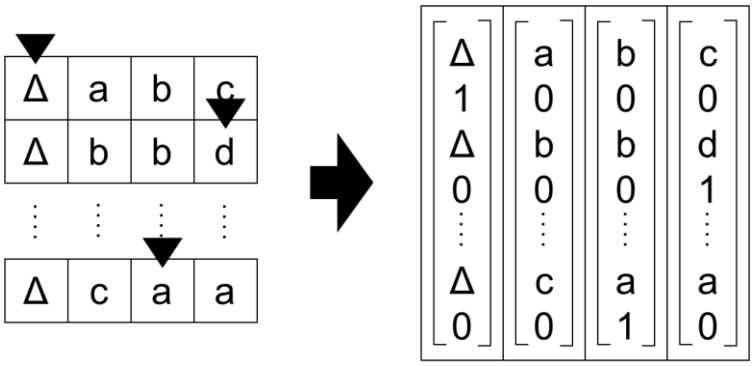


…, 12345. Posloupnost 12345 odpovídá akceptujícímu běhu, tedy deterministický stroj akceptuje.

$$(q_0, \Delta aab\Delta^\omega, 0) \vdash (q_1, \Delta aab\Delta^\omega, 1) \vdash (q_1, \Delta aab\Delta^\omega, 2) \vdash (q_2, \Delta abb\Delta^\omega, 2) \vdash (q_2, \Delta abb\Delta^\omega, 1) \vdash (q_f, \Delta\Delta bb\Delta^\omega, 1)$$

k-páskový TS → 1-páskový TS. Každý k-páskový Turingův stroj lze převést na ekvivalentní jednopáskový Turingův stroj. Podstatou tohoto převodu je transformace páskové abecedy. Ať M = (Q, Σ, Γ_1, Γ_2, …, Γ_k, δ, q_0, q_f) je k-páskový Turingův stroj. Z původních páskových abeced Γ_1, Γ_2, …, Γ_k vytvoříme jednu novou páskovou abecedu $\Gamma = \Gamma_1 \times \{0, 1\} \times \Gamma_2 \times \{0, 1\} \times \ldots \times \Gamma_k \times \{0, 1\}$

jakožto množinu 2k-tic, které jsou složeny ze znaků původních abeced a z příznaků 0 a 1, které udávají, zda na daném políčku k-páskového stroje byla odpovídající hlava stroje.



Pokud budeme chtít vykonat krok na tomto odpovídajícím jednopáskovém stroji, bude třeba projít celou pásku složenou z 2k-tic, abychom v rámci stavového řízení uchovali informaci o pozici hlav a symbolů pod hlavami. Díky tomu lze vyhodnotit, jaký přechod vykonat. Druhým průchodem pásky aktualizujeme jednotlivé 2k-tice, čímž provedeme přechod. Pro jeden krok původního stroje je tedy třeba několikrát projít pásku ekvivalentního jednopáskového stroje.

Jazyky přijímané Turingovými stroji

Jednotlivé typy Turingových strojů přijímají následující třídy jazyků:

- **lineárně omezené automaty**
  - třída **kontextových jazyků $\mathcal{L}_1$**
- **úplné Turingovy stroje**
  - třída **rekurzivních jazyků $\mathcal{L}_{REC}$**
- **obecné Turingovy stroje**
  - třída **rekurzivně vyčíslitelných jazyků $\mathcal{L}_{RE} = \mathcal{L}_0$**

Platí

$$\mathcal{L}_1 \subset \mathcal{L}_{REC} \subset \mathcal{L}_{RE}.$$

Vyčíslitelné funkce

Uvažujme o funkcích typu $f : \mathbb{N}^n \to \mathbb{N}^m$, tedy o funkcích pracujících s vektory přirozených čísel $(a_1, a_2, \dots a_k) = \bar{a}$. Každá taková funkce může spadat do jedné z následujících skupin:

- **totální funkce**
  - $\forall(a_1, a_2, \dots a_n) \in \mathbb{N}^n : \exists(b_1, b_2, \dots b_m) \in \mathbb{N}^m : f(a_1, a_2, \dots a_n) = (b_1, b_2, \dots b_m)$
- **striktně parciální funkce**
  - $\exists(a_1, a_2, \dots a_n) \in \mathbb{N}^n : f(a_1, a_2, \dots a_n)$ není definováno

Počáteční funkce

Počáteční funkce jsou základní, nejjednodušší vyčíslitelné funkce, z nichž bude možné vystavět další funkce.

- **nulová funkce**
  - **ξ :** $\mathbb{N}^0 \rightarrow \mathbb{N}^1$, kde $\xi() = 0$
  - funkce, která nepřijímá žádný argument a vrací konstantu 0. Konstantní funkce
  - $\xi() = 0$
- **funkce následníka**
  - **σ :** $\mathbb{N}^1 \rightarrow \mathbb{N}^1$, kde $\sigma(a) = a + 1$
  - funkce, která přijímá jako argument jedno číslo (vektor čísel o délce 1) a vrátí přirozené číslo o 1 vyšší
  - $\sigma(18) = 19$
- **projekce**
  - **$\pi_k^n : \mathbb{N}^n \rightarrow \mathbb{N}^1$**, kde $\pi_k^n(a_1, a_2, \dots a_k, \dots a_n) = a_k$
  - funkce, která přijímá vektor čísel o délce n a vrací číslo na pozici k (indexováno od 1 do n)
  - $\pi_2^4(10, 100, 1000, 10000) = 100$

Metody pro vytváření složitějších funkcí

Počáteční funkce je možné kombinovat do složitějších funkcí pomocí operací **kompozice** a **kombinace, primitivní rekurze** a **minimalizace**.

- **kombinace**
  - mějme dvě funkce $f : \mathbb{N}^k \rightarrow \mathbb{N}^m$, $g : \mathbb{N}^k \rightarrow \mathbb{N}^n$, kombinace je funkce typu $\times : \mathbb{N}^k \rightarrow \mathbb{N}^{m+n}$, která kombinuje funkce f, g
  - $f \times g\ (a_1, a_2, \dots, a_k) = (f(a_1, a_2, \dots, a_k), g(a_1, a_2, \dots, a_k))$
  - kombinace přijímá na vstup vektor délky *k*, aplikuje na něj funkce *f* i *g* a vrátí vektor výsledků obou těchto funkcí
  - $\pi_1^1 \times \sigma\ (18) = (\pi_1^1(18), \sigma(18)) = (18, 19)$
- **kompozice**
  - mějme dvě funkce $f : \mathbb{N}^k \rightarrow \mathbb{N}^m$, $g : \mathbb{N}^m \rightarrow \mathbb{N}^n$, kompozice je funkce typu $\circ : \mathbb{N}^k \rightarrow \mathbb{N}^n$, která skládá funkce f, g
  - $g \circ f\ (a_1, a_2, \dots, a_k) = g(f(a_1, a_2, \dots, a_k))$
  - kompozice na vstup přijímá vektor délky *k,* aplikuje na něj funkci *f* a na výsledek aplikuje funkci *g*
  - $\sigma \circ \sigma \circ \sigma \circ \sigma \circ \sigma \circ \sigma \circ \sigma \circ \xi() = 7$
  - $\pi_2^3 \circ ((\sigma \circ \xi()) \times (\sigma \circ \sigma \circ \xi()) \times (\sigma \circ \sigma \circ \sigma \circ \sigma \circ \sigma \circ \sigma \circ \xi())) = \pi_2^3(1, 2, 5) = 2$
  - $\sigma \circ \sigma \circ \sigma \circ \pi_2^4\ (10, 100, 1000, 10000) = \sigma \circ \sigma \circ \sigma(100) = \sigma \circ \sigma(101) = \sigma(102) = 103$
  - (v tomto posledním příkladu by vektor (10, 100, 1000, 10000) bylo možné rovněž definovat pomocí kompozice a kombinace, ovšem šlo by o velmi dlouhý zápis)
- **primitivní rekurze**

    - mějme dvě funkce g : $\mathbb{N}^k \rightarrow \mathbb{N}^m$, h : $\mathbb{N}^{k+m+1} \rightarrow \mathbb{N}^m$, pomocí techniky primitivní rekurze lze vytvořit funkci f : $\mathbb{N}^{k+1} \rightarrow \mathbb{N}^m$
    - f(ā, 0) = g(ā)
    - f(ā, y + 1) = h(ā, y, f(ā, y))
    - každou primitivní rekurzi lze převést na while cyklus s konečným, konstantním počtem provedení smyčky
    - poslední argument odpovídá počtu cyklů, které budou ještě provedeny, přičemž při každém dalším volání funkce f se tento argument snižuje, dokud nedosáhne 0
    - v poslední iteraci dojde pouze k vyhodnocení funkce g, která už nezávisí na dalším vyhodnocení funkce f
    - příkladem je funkce sčítání dvou čísel s přičtením kostanty 2, tzn. f : y = a + b + 2, definovaná takto:
        - f(a, 0) = σ ∘ σ(a)
        - f(a, x + 1) = σ ∘ f(a, x)
        - tzn. g(a) = σ ∘ σ(a), h(a, x, f(a, x)) = σ ∘ f(a, x)
        - f(2, 2) = σ ∘ f(2, 1) = σ ∘ σ ∘ f(2, 0) = σ ∘ σ ∘ σ ∘ σ(2) = 6, což odpovídá 2 + 2 + 2 = 6
- **minimalizace**
    - mějme funkci g : $\mathbb{N}^{k+1} \rightarrow \mathbb{N}$, pomocí techniky minimalizace lze vytvořit funkci f : $\mathbb{N}^k \rightarrow \mathbb{N}$
    - tyto funkce mohou být parciální
    - f(ā) = y, přičemž y je minimální přirozené číslo, které splňuje g(ā, y) = 0, kde navíc $\forall z \in \mathbb{N} : z < y$ a g(ā, z) je definováno
    - minimalizace odpovídá while cyklu s předem nedefinovaným počtem provedení smyčky
    - pro nalezení f(ā) je totiž třeba vyhodnotit g(ā, 0), g(ā, 1), g(ā, 2),… dokud nebude nalezeno vhodné y, kde g(ā, y) = 0
    - pokud funkce g není definovaná pro některé g(ā, z), přičemž dosud nebylo nalezeno vhodné y, f(ā) je rovněž nedefinováno
    - vyhodnocujeme-li g(ā, 0), g(ā, 1), g(ā, 2),... donekonečna, přičemž pro žádné y neplatí, že g(ā, y) = 0, ale všechny g(ā, z) jsou definované, pak toto odpovídá nekonečnému cyklení while cyklu

Primitivně rekurzivní funkce

Primitivně rekurzivní funkce jsou takové totální funkce, které je možné obdržet z počátečních funkcí konečnou aplikací:

- kombinace
- kompozice
- primitivní rekurze

μ-rekurzivní funkce

Jde o všechny totální funkce, které jsou vyčíslitelné. Každá primitivně rekurzivní funkce je μ-rekurzivní, ale existují μ-rekurzivní funkce, které nejsou primitivně rekurzivní. Příkladem je Ackermanova funkce definovaná jako

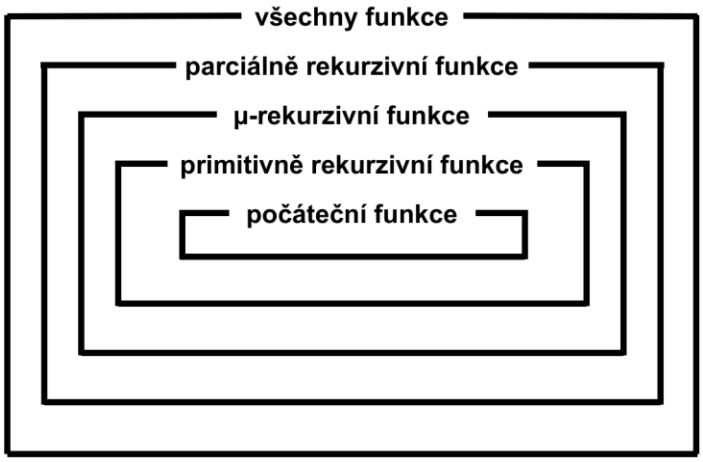
$$A(0, y) = y + 1$$
$$A(x + 1, 0) = A(x, 1)$$
$$A(x + 1, y + 1) = A(x, A(x + 1, y)).$$

Parciálně rekurzivní funkce

Jde o všechny (tedy nejen totální) funkce, které je možné získat z počátečních funkcí aplikací:

- kombinace
- kompozice
- primitivní rekurze
- minimalizace

Parciálně rekurzivní funkce mají výpočetní sílu Turingových strojů, tedy každou parciálně rekurzivní funkci je možné vyčíslit pomocí nějakého Turingova stroje a každý Turingův stroj provádí výpočet odpovídající nějaké parciálně rekurzivní funkci.



Pokud v textu najdete chybu, nebudete něčemu rozumět nebo budete mít dojem, že by bylo vhodné něco doplnit, kontaktujte na discordu uživatele kocotom.