

Jazyky rekurzivně vyčíslitelné a jazyky rekurzivní

Rekurzivní vyčíslitelnost a rekurzivnost

- ❖ Turingův stroj se nazývá **úplný** (*total*), právě když se pro každý vstup zastaví.
- ❖ *Poznámka*: Nedeterministický Turingův stroj je **úplný**, právě když pro každý vstup je každá výpočetní větev konečná (tj. pro každý vstup vždy zastaví).

Definice 8.1 Jazyk $L \subseteq \Sigma^*$ se nazývá

- **rekurzivně vyčíslitelný**, jestliže $L = L(M)$ pro nějaký TS M ,
 - **rekurzivní**, jestliže $L = L(M)$ pro nějaký **úplný** TS M .
- ❖ Je-li M úplný Turingův stroj, pak říkáme, že M **rozhoduje jazyk** $L(M)$.
 - ❖ Ke každému **rekurzivnímu jazyku** existuje TS, který ho rozhoduje, tj. **zastaví pro každé vstupní slovo** – tento TS lze samozřejmě upravit tak, aby pro každý řetězec z daného jazyka zastavil s páskou $\Delta Y \Delta \Delta \dots$ a jinak zastavil s páskou $\Delta N \Delta \Delta \dots$.
 - ❖ TS přijímající **rekurzivně vyčíslitelný jazyk** L zastaví pro každé $w \in L$, ovšem pro $w \notin L$ může zastavit, ale také **může donekonečna cyklit**.

Rozhodovací problémy

- ❖ **Rozhodovací problém** (*decision problem*) P může být chápán jako funkce f_P s oborem hodnot $\{true, false\}$.
- ❖ **Rozhodovací problém je obvykle specifikován:**
 - definičním oborem A_P reprezentujícím množinu možných instancí problému (vstupů) a
 - podmnožinou $B_P \subseteq A_P$, $B_P = \{p \mid f_P(p) = true\}$ instancí, pro které je hodnota f_P rovna *true*.
- ❖ V teorii formálních jazyků **používáme ke kódování jednotlivých instancí problémů řetězce nad vhodnou abecedou Σ** . Pak je rozhodovací problém P přirozeně specifikován jazykem $L_P = \{w \in \Sigma^* \mid w = code(p), p \in B_P\}$, kde $code : A_P \rightarrow \Sigma^*$ je injektivní funkce, která přiřazuje instancím problému příslušný řetězec (nezávisle na f_P).

Příklad 8.1 Příklady rozhodovacích problémů:

- P_1 – orientovaný graf je silně souvislý.
 - P_2 – dvě bezkontextové gramatiky jsou ekvivalentní,
 - P_3 – n je prvočíslo.
- ❖ *Poznámka:* Dále budeme o rozhodovacích problémech hovořit jednoduše jako o problémech.

Rozhodování problémů TS

Definice 8.2 Necht' P je problém specifikovaný jazykem L_P nad Σ . Problém P nazveme:

- **rozhodnutelný**, pokud L_P je rekurzivní jazyk, tj. existuje TS, který L_P rozhoduje (přijme každý řetězec $w \in L_P$, a zamítne každý řetězec $w \in \Sigma^* \setminus L_P$),
- **nerozhodnutelný**, když není rozhodnutelný, a
- **částečně rozhodnutelný**, jestliže L_P je rekurzivně vyčíslitelný jazyk.

❖ **Poznámka:** Z definice 8.2 plyne, že každý rozhodnutelný problém je současně částečně rozhodnutelný, ale některé nerozhodnutelné problémy nejsou ani částečně rozhodnutelné.

TS a jazyky typu 0

Jazyky přijímané TS jsou typu 0

❖ Pro zápis konfigurace TS v řídicím stavu q a s konfigurací pásky $\Delta x \underline{y} z \Delta \dots$ zavedeme konvenci $[\Delta x q y z \Delta \dots]$.

Věta 8.1 Každý rekurzivně vyčíslitelný jazyk je jazykem typu 0.

Důkaz. * Necht' $L = L(M)$ pro nějaký TS $M = (Q, \Sigma, \Gamma, \delta, q_0, q_F)$. Sestrojíme gramatiku $G = (N, \Sigma, P, S)$ typu 0 takovou, že $L(G) = L(M)$. Gramatika G dovoluje vytvářet derivace odpovídající reverzi posloupnosti konfigurací TS M při přijetí $w \in L(M)$:

1. $N = \{S\} \cup Q \cup (\Gamma \setminus \Sigma) \cup \{[,]\}$ (množiny jsou po dvou disjunktní).
2. P je nejmenší množina obsahující následující pravidla:
 - (a) $S \rightarrow [q_f \Delta Y \Delta]$,
 - (b) $\Delta] \rightarrow \Delta \Delta]$ – doplnění Δ ,
 - (c) $qy \rightarrow px$, jestliže $\delta(p, x) = (q, y)$,
 - (d) $xq \rightarrow px$, jestliže $\delta(p, x) = (q, R)$,
 - (e) $qyx \rightarrow ypx$ pro každé $y \in \Gamma$, jestliže $\delta(p, x) = (q, L)$,
 - (f) $[q_0 \Delta \rightarrow \varepsilon, \Delta \Delta] \rightarrow \Delta]$, $\Delta] \rightarrow \varepsilon$ – zajištění $[q_0 \Delta w \Delta \dots \Delta] \xrightarrow[G]{+} w$.

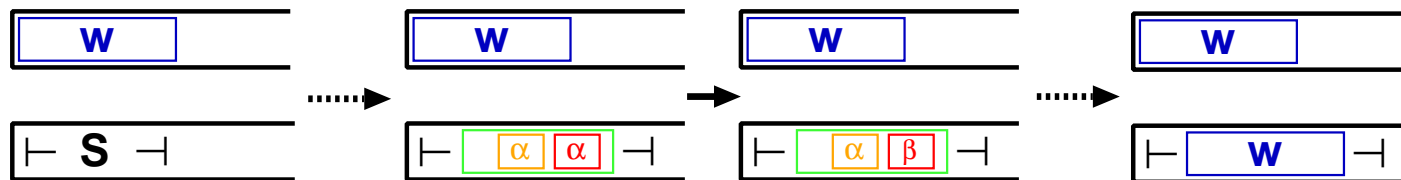
Snadno se nyní nahlédne, že $w \in L(M)$ právě tehdy, když existuje derivace $S \Rightarrow_G [q_F \Delta Y \Delta] \Rightarrow_G \dots \Rightarrow_G [q_0 \Delta w \Delta \dots] \Rightarrow_G \dots \Rightarrow_G w$, a že $L(G) = L(M)$. □ *

Jazyky typu 0 jsou přijímány TS

Věta 8.2 Každý jazyk typu 0 je přijímán nějakým TS (tj. je rekurzivně vyčíslitelný).

Důkaz. Necht' $L = L(G)$ pro $G = (N, \Sigma, P, S)$ je jazykem typu 0. Sestrojíme nedeterministický dvoupáskový TS M takový, že $L(G) = L(M)$:

- 1. páska obsahuje přijímaný vstupní řetězec w .
- Na 2. pásce se M pokouší pomocí simulace použití prepisovacích pravidel $(\alpha \rightarrow \beta) \in P$ vytvořit derivaci w :



1. Stroj nejprve umístí na 2. pásku symbol S .
2. Stroj opakovaně simuluje na 2. pásce provádění pravidel $(\alpha \rightarrow \beta) \in P$. Nedeterministicky zvolí pravidlo a také výskyt α na pásce. Při prepisu α na β , $|\alpha| \neq |\beta|$, může využít posuv části užitečného obsahu pásky vlevo či vpravo.
3. Stroj srovná finální obsah 2. pásky s 1. páskou. Shodují-li se, zastaví přechodem do q_F . Jinak posouvá hlavu doleva až do abnormálního zastavení.

Snadno se nyní nahlédne, že skutečně $L(G) = L(M)$. Navíc lze M podobně jako u vícepáskových DTS převést na jednopáskový NTS a ten dále na jednopáskový DTS. \square

Jazyky typu 0 = jazyky přijímané TS

Věta 8.3 *Třída jazyků přijímaných TS (neboli jazyků rekurzivně vyčíslitelných) je shodná se třídou jazyků typu 0.*

Důkaz. Důsledek dvou předchozích vět.



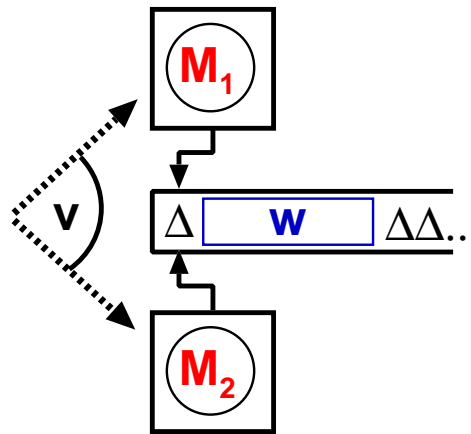
Vlastnosti jazyků rekurzivních a rekurzivně vyčíslitelných

Uzavřenost vůči \cup , \cap , \cdot a $*$

Věta 8.4 Třídy rekurzivních a rekurzivně vyčíslitelných jazyků jsou uzavřeny vůči operacím \cup , \cap , \cdot a $*$.

Důkaz. Nechť L_1, L_2 jsou jazyky přijímané TS M_1, M_2 . Zřejmě můžeme předpokládat, že množiny stavů TS M_1, M_2 jsou disjunktní.

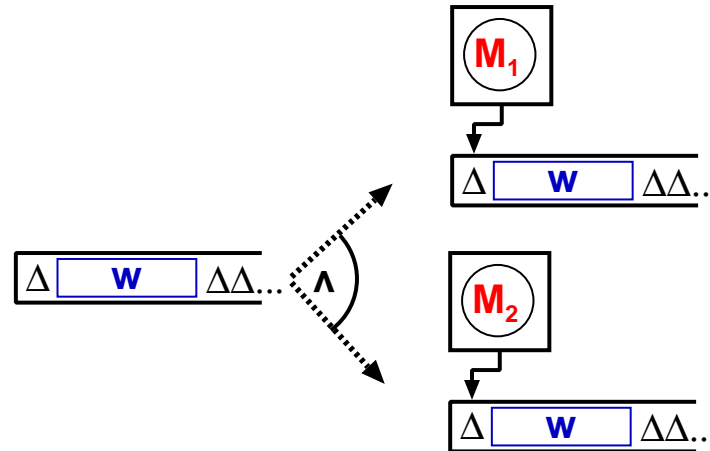
- NTS $M_{L_1 \cup L_2}$, $L(M_{L_1 \cup L_2}) = L_1 \cup L_2$, sestrojíme tak, že sjednotíme po složkách stroje M_1 a M_2 , zavedeme nový počáteční stav, z něj nedeterministické přechody přes Δ/Δ do obou původních počátečních stavů a sloučíme původní koncové stavy do jediného nového koncového stavu.



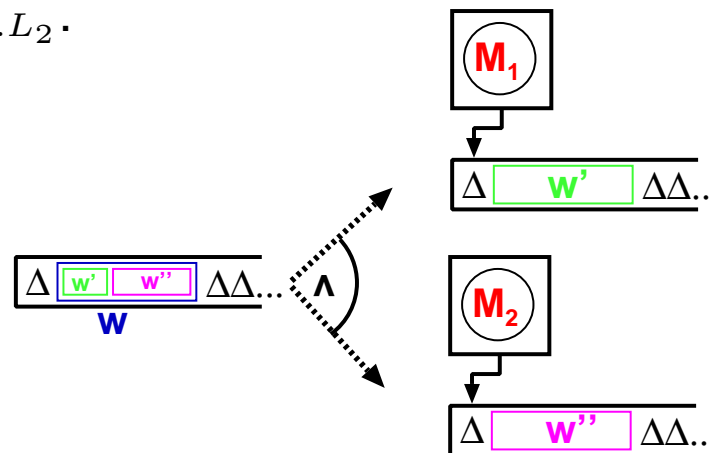
Důkaz pokračuje dále.

Pokračování důkazu.

- Třípáskový TS $M_{L_1 \cap L_2}$, $L(M_{L_1 \cap L_2}) = L_1 \cap L_2$, okopíruje vstup z první pásky na druhou, na ní simuluje stroj M_1 , pokud ten přijme, okopíruje vstup z první pásky na třetí, na ní simuluje stroj M_2 , pokud i ten přijme, přijme i stroj $M_{L_1 \cap L_2}$.



- Třípáskový NTS $M_{L_1.L_2}$, $L(M_{L_1.L_2}) = L_1.L_2$, okopíruje nedeterministicky zvolený prefix vstupu z první pásky na druhou, na ní simuluje stroj M_1 , pokud ten přijme, okopíruje zbytek vstupu z první pásky na třetí, na ní simuluje stroj M_2 , pokud i ten přijme, přijme i stroj $M_{L_1.L_2}$.



Důkaz pokračuje dále.

Pokračování důkazu.

- Dvoupáskový NTS $M_{L_1^*}$, $L(M_{L_1^*}) = L_1^*$, je zobecněním předchozího stroje: po částech kopíruje vstup z první pásky na druhou a na ní simuluje opakovaně stroj M_1 . Obsah druhé pásky má ohraničený speciálními značkami a po každé simulaci stroje M_1 ho smaže. Umožňuje samozřejmě posuv pravé značky dále doprava při nedostatku místa.

Jsou-li stroje M_1 a M_2 úplné, je možné vybudovat stroje podle výše uvedených pravidel také jako **úplné** (u $M_{L_1 \cup L_2}$, $M_{L_1 \cap L_2}$, $M_{L_1 \cdot L_2}$ je to okamžité, u $M_{L_1^*}$ nepřipustíme načítání prázdného podřetězce vstupu z 1. na 2. pásku – pouze umožníme jednorázově přijmout prázdný vstup). To dokazuje uzavřenost vůči uvedeným operacím také u **rekurzívních jazyků**.

□

(Ne)uzavřenost vůči komplementu

Věta 8.5 Třída rekurzivních jazyků je uzavřena vůči komplementu.

Důkaz. TS M přijímající rekurzivní jazyk L vždy zastaví. Snadno upravíme M na M' , který při nepřijetí řetězce vždy přejde do unikátního stavu q_{reject} . TS \overline{M} , $L(\overline{M}) = \overline{L}$, snadno dostaneme z M' záměnou q_F a q_{reject} . □

❖ Třída rekurzivně vyčíslitelných jazyků není uzavřena vůči komplementu!

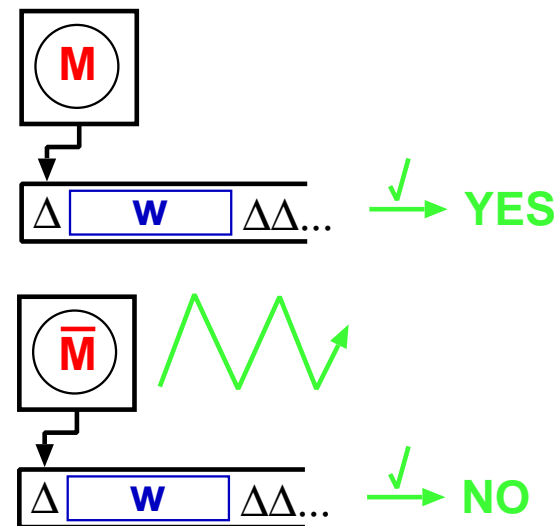
- Výše uvedené konstrukce nelze užít – cyklení zůstane zachováno.
- Důkaz neuzavřenosti bude uveden v dalších přednáškách.

Věta 8.6 Jsou-li L i \bar{L} rekurzívně vyčísitelné, pak jsou oba rekurzívni.

Důkaz.

Mějme M , $L(M) = L$, a \bar{M} , $L(\bar{M}) = \bar{L}$. Úplný TS přijímající L sestrojíme takto:

- Použijeme dvě pásy. Na jedné budeme simulovat M , na druhé \bar{M} . Simulace se bude provádět proloženě krok po kroku: krok M , krok \bar{M} , krok M , ...
- Přijmeme, právě když by přijal M , zamítneme abnormálním zastavením, právě když by přijal \bar{M} . Jedna z těchto situací určitě nastane v konečném počtu kroků.



Existence úplného TS pro \bar{L} plyne z uzavřenosti rekurzívniých jazyků vůči komplementu. □

❖ **Důsledkem výše uvedených vět** je mj. to, že pro L a \bar{L} musí vždy nastat jedna z následujících situací:

- L i \bar{L} jsou rekurzívni,
- L ani \bar{L} nejsou rekurzívně vyčísitelné,
- jeden z těchto jazyků je rekurzívně vyčísitelný, ale ne rekurzívni, druhý není rekurzívně vyčísitelný.

Lineárně omezené automaty

Lineárně omezené automaty

- ❖ **Lineárně omezený automat (LOA)** je **nedeterministický** TS, který nikdy neopustí tu část pásky, na níž je zapsán jeho vstup.
- ❖ Formálně můžeme LOA definovat jako NTS, který má v Γ speciální symbol, kterým unikátně označujeme pravý konec vstupu na pásce, přičemž tento symbol není možné přepsat, ani z něj provést posun doprava.
- ❖ **Deterministický LOA** můžeme přirozeně definovat jako (deterministický) TS, který nikdy neopustí část pásky se zapsaným vstupem.
- ❖ **Není známo, zda deterministický LOA je či není striktně slabší než LOA.**

LOA a kontextové jazyky

Věta 8.7 Třída jazyků, kterou lze generovat kontextovými gramatikami, odpovídá třídě jazyků, které lze přijímat LOA.

Důkaz.

- Uvážíme definici kontextových gramatik jako gramatik s pravidly v podobě $\alpha \rightarrow \beta$, kde $|\alpha| \leq |\beta|$, nebo $S \rightarrow \varepsilon$.
- *LOA \longrightarrow G1:
 - Použijeme podobnou konstrukci jako u TS \longrightarrow G0.
 - Na počátku vygenerujeme příslušný pracovní prostor, který se pak již nebude měnit: odpadá nekontextové pravidlo $\Delta\Delta] \rightarrow \Delta]$.
 - Užití nekontextových pravidel $[q_0\Delta \rightarrow \varepsilon$ a $\Delta] \rightarrow \varepsilon$ obejdeme (1) zavedením zvláštních koncových nonterminálů integrujících původní informaci a příznak, že se jedná o první/poslední symbol a (2) integrací symbolu reprezentujícího řídicí stav a pozici hlavy s následujícím páskovým symbolem.*
- G1 \longrightarrow LOA:
 - Použijeme podobnou konstrukci jako u G0 \longrightarrow TS s tím, že nepovolíme, aby rozsah druhé pásky někdy překročil rozsah první pásky.

□

Kontextové a rekurzivní jazyky

Věta 8.8 Každý kontextový jazyk je rekurzivní.

Důkaz. (Idea)

- Počet konfigurací, které se mohou objevit při přijímání w příslušným LOA M je vzhledem k nemožnosti zvětšovat pracovní prostor pásky konečný: lze shora ohraničit funkcí c^n pro vhodnou konstantu c – exponenciála plyne z nutnosti uvažovat výskyt všechny možných symbolů na všech místech pásky.
- Pro zápis libovolného čísla z intervalu $0, \dots, c^n - 1$ nikdy nebude třeba více než n symbolů, uijeme-li c -ární soustavu.
- Můžeme zkonstruovat úplný LOA ekvivalentní s M , který bude mít každý symbol na pásce strukturovaný jako dvojici:
 - S využitím 1. složek těchto dvojic simulujeme M .
 - V 2. složkách počítáme počet kroků; dojde-li k přetečení, odmítneme vstup.

□

Věta 8.9 Ne každý rekurzivní jazyk je kontextový.

Důkaz. (Idea) Lze užít techniku diagonalizace prezentovanou dále .

□

Vlastnosti kontextových jazyků

Věta 8.10 Třída kontextových jazyků je uzavřena vůči operacím \cup , \cap , \cdot , $*$ a komplementu.

Důkaz.

- Uzavřenost vůči \cup , \cap , \cdot a $*$ lze ukázat stejně jako u rekurzivně spočetných jazyků.
- Důkaz uzavřenosti vůči komplementu je značně komplikovaný (všimněme si, že LOA je *nedeterministický* a nelze tudíž užít konstrukce použité u rekurzivních jazyků). Jedná se o přímí důsledek Immerman-Szelepcényiho věty (více viz složitost, 20 let otevřený problém, vyřešen 1987, Gödelova cena 1995, zásadní československá stopa v teoretické informatice).

□

❖ Poznamenejme, že již víme, že u kontextových jazyků

- lze rozhodovat členství věty do jazyka (rekurzivnost) a
- nelze rozhodovat inkluzi jazyků (neplatí ani pro bezkontextové jazyky).

❖ Dále lze ukázat, že pro kontextové jazyky **nelze rozhodovat prázdnotu jazyka** (užije se redukce z Postova problému přiřazení – viz další přednášky).

Vztah vyčíslitelných funkcí a Turingových strojů

Základy teorie vyčíslitelných funkcí

Budeme se snažit identifikovat takové funkce, které jsou „spočítatelné“, tj. vyčíslitelné v obecném smyslu (bez ohledu na konkrétní výpočetní systém). Abychom snížili extrémní velikost třídy těchto funkcí, která je dána také varietou definičních oborů a oborů hodnot, omezíme se, uvažující možnost kódování, na funkce tvaru:

$$f : \mathbb{N}^m \rightarrow \mathbb{N}^n$$

kde $\mathbb{N} = \{0, 1, 2, \dots\}$, $m, n \in \mathbb{N}$

❖ Konvence: n -tici $(x_1, x_2, \dots, x_n) \in \mathbb{N}^n$ budeme označovat jako \bar{x}

❖ Klasifikace parciálních funkcí:

- *Totální funkce* – definovaná pro každý $\bar{x} \in \mathbb{N}^m$
- *Striktně parciální funkce* – $\exists \bar{x} \in \mathbb{N}^m : f(\bar{x}) = \perp$

Počáteční funkce

Hierarchie vyčíslitelných funkcí je založena na dostatečně elementárních tzv. *počátečních funkcích*, které tvoří „stavební kameny“ vyšších funkcí.

❖ Jsou to tyto funkce:

1. *Nulová funkce* (zero function): $\xi() = 0$
zobrazuje „prázdnou n -tici“ $\mapsto 0$

2. *Funkce následníka* (successor function): $\sigma : \mathbb{N} \rightarrow \mathbb{N}$
 $\sigma(x) = x + 1$

3. *Projekce* (projection): $\pi_k^n : \mathbb{N}^n \rightarrow \mathbb{N}$
Vyberá z n -tice k -tou složku, např.: $\pi_2^3(7, 6, 4) = 6$ a $\pi_1^2(5, 17) = 5$
Speciální případ: $\pi_0^n : \mathbb{N}^n \rightarrow \mathbb{N}^0$, tj. např. $\pi_0^3(1, 2, 3) = ()$

Vytváření složitějších funkcí

Nyní definujeme tři způsoby vytváření nových, složitějších funkcí:

1. *Kombinace*:

Kombinací dvou funkcí $f : \mathbb{N}^k \rightarrow \mathbb{N}^m$ a $g : \mathbb{N}^k \rightarrow \mathbb{N}^n$ získáme funkci, pro kterou:

$$\begin{aligned} f \times g &: \mathbb{N}^k \rightarrow \mathbb{N}^{m+n} \\ f \times g(\bar{x}) &= (f(\bar{x}), g(\bar{x})), \bar{x} \in \mathbb{N}^k \end{aligned}$$

Např.: $\pi_1^3 \times \pi_3^3(4, 12, 8) = (4, 8)$

2. *Kompozice*:

Kompozice dvou funkcí $f : \mathbb{N}^k \rightarrow \mathbb{N}^m$ a $g : \mathbb{N}^m \rightarrow \mathbb{N}^n$ je funkce, pro kterou:

$$\begin{aligned} g \circ f &: \mathbb{N}^k \rightarrow \mathbb{N}^n \\ g \circ f(\bar{x}) &= g(f(\bar{x})), \bar{x} \in \mathbb{N}^k \end{aligned}$$

Např.:

$$\sigma \circ \xi() = 1$$

$$\sigma \circ \sigma \circ \xi() = 2$$

3. *Primitivní rekurze:*

Příklad 8.2 Předpokládejme, že chceme definovat funkci násobení

$mult : \mathbb{N}^2 \rightarrow \mathbb{N}$.

$$mult(x, y) = \underbrace{x + \cdots + x}_y$$

Zřejmě:

(a) Pro $y = 0$ platí $x * 0 = 0$

(b) Pro $y > 0$ je výsledek $x + mult(x, y - 1)$

Takže funkci $mult$ můžeme definovat následujícím předpisem:

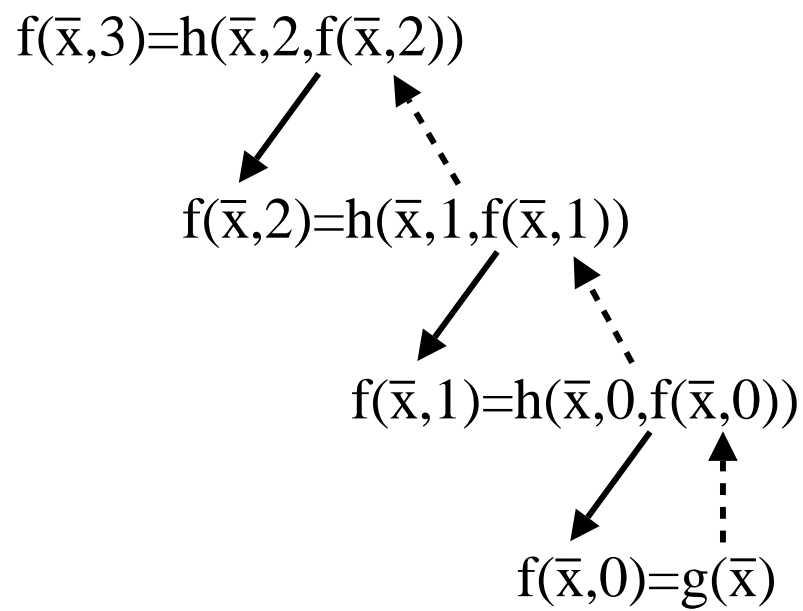
$$mult(x, 0) = 0$$

$$mult(x, y + 1) = x + mult(x, y)$$

Primitivní rekurze je technika, která umožňuje vytvořit funkci $f : \mathbb{N}^{k+1} \rightarrow \mathbb{N}^m$ na základě jiných dvou funkcí $g : \mathbb{N}^k \rightarrow \mathbb{N}^m$ a $h : \mathbb{N}^{k+m+1} \rightarrow \mathbb{N}^m$ rovnicemi:

$$\begin{aligned} f(\bar{x}, 0) &= g(\bar{x}) \\ f(\bar{x}, y + 1) &= h(\bar{x}, y, f(\bar{x}, y)), \bar{x} \in \mathbb{N}^k \end{aligned}$$

Ilustrace schématu vyčíslení (pro $y = 3$):



Primitivně rekurzivní funkce

Definice 8.3 *Třída primitivně rekurzivních funkcí* je třída totálních funkcí, které mohou být vytvořeny z počátečních funkcí konečnou aplikací:

- kombinace
- kompozice
- primitivní rekurze

❖ Charakteristika (bez důkazu): Primitivně rekurzivní funkce je taková, kterou lze zapsat jako počítačový program obsahující pouze konečné for cykly a žádné while cykly nebo skoky.

Funkce mimo primitivně rekurzivní funkce

Existují funkce, které jsou vyčíslitelné a nejsou primitivně rekurzivními funkcemi. Jsou to všechny striktně parciální funkce (jako *div*), ale i totální funkce.

Věta 8.11 Existuje totální funkce z \mathbb{N} do \mathbb{N} , která není primitivně rekurzivní.

Důkaz.

Definice funkcí, které jsou primitivně rekurzivní budeme chápat jako řetězce a můžeme je uspořádat v lexikografickém pořadí s označením $f_1, f_2, \dots, f_n, \dots$

Definujeme nyní funkci $f : \mathbb{N} \rightarrow \mathbb{N}$ tak, že $f(n) = f_n(n) + 1$ pro $\forall n \in \mathbb{N} \setminus \{0\}$. f je jasně totální a vyčíslitelná. f však není primitivně rekurzivní (kdyby byla, pak $f \equiv f_m$ pro nějaké $m \in \mathbb{N}$. Pak ale $f(m) = f_m(m)$ a ne $f_m(m) + 1$, jak vyžaduje definice funkce f).

□

Příkladem totální funkce, která není primitivně rekurzivní byla prezentována W. Ackermannem (1928) a nazývá se *Ackermannova funkce*. Je dána rovnicemi:

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

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

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

Parciálně rekurzivní funkce

K rozšíření třídy vyčíslitelných funkcí za totální vyčíslitelné funkce zavedeme techniku známou pod názvem *minimalizace*. Tato technika umožňuje vytvořit funkci $f : \mathbb{N}^n \rightarrow \mathbb{N}$ z jiné funkce $g : \mathbb{N}^{n+1} \rightarrow \mathbb{N}$ předpisem, v němž $f(\bar{x})$ je nejmenší y takové, že:

1. $g(\bar{x}, y) = 0$
2. $g(\bar{x}, z)$ je definována pro $\forall z < y, z \in \mathbb{N}$

❖ Funkce definovaná minimalizací je skutečně vyčíslitelná. Výpočet hodnoty $f(\bar{x})$ zahrnuje výpočet $g(\bar{x}, 0), g(\bar{x}, 1), \dots$ tak dlouho, pokud nedostaneme:

- $g(\bar{x}, y) = 0$ $(f(\bar{x}) = y)$
- $g(\bar{x}, z)$ je nedefinována $(f(\bar{x})$ je nedefinována)

Definice 8.4 *Třída parciálně rekurzivních funkcí* je třída parciálních funkcí, které mohou být vytvořeny z počátečních funkcí aplikací:

- kombinace
- kompozice
- primitivní rekurze
- minimalizace

Turingovsky vyčíslitelné funkce

Definice 8.5 Turingův stroj $M = (Q, \Sigma, \Gamma, \delta, q_0, q_F)$ *vyčísluje (počítá)* parciální funkci $f : \Sigma^{*m} \rightarrow \Sigma_1^{*n}$, $\Sigma_1 \subseteq \Gamma$, $\Delta \notin \Sigma_1$, jestliže pro každé $(w_1, w_2, \dots, w_m) \in \Sigma^{*m}$ a odpovídající počáteční konfiguraci $\underline{\Delta}w_1\Delta w_2\Delta \dots \Delta w_m\Delta^\omega$ stroj M :

1. v případě, že $f(w_1, \dots, w_m)$ je definována, pak M zastaví a páska obsahuje $\underline{\Delta}v_1\Delta v_2\Delta \dots \Delta v_n\Delta\Delta\Delta$, kde $(v_1, v_2, \dots, v_n) = f(w_1, \dots, w_m)$
2. v případě, že $f(w_1, \dots, w_m)$ není definována, M cyklí nebo zastaví abnormálně.

Parciální funkce, kterou může počítat nějaký Turingův stroj se nazývá funkcí *Turingovsky vyčíslitelnou*.

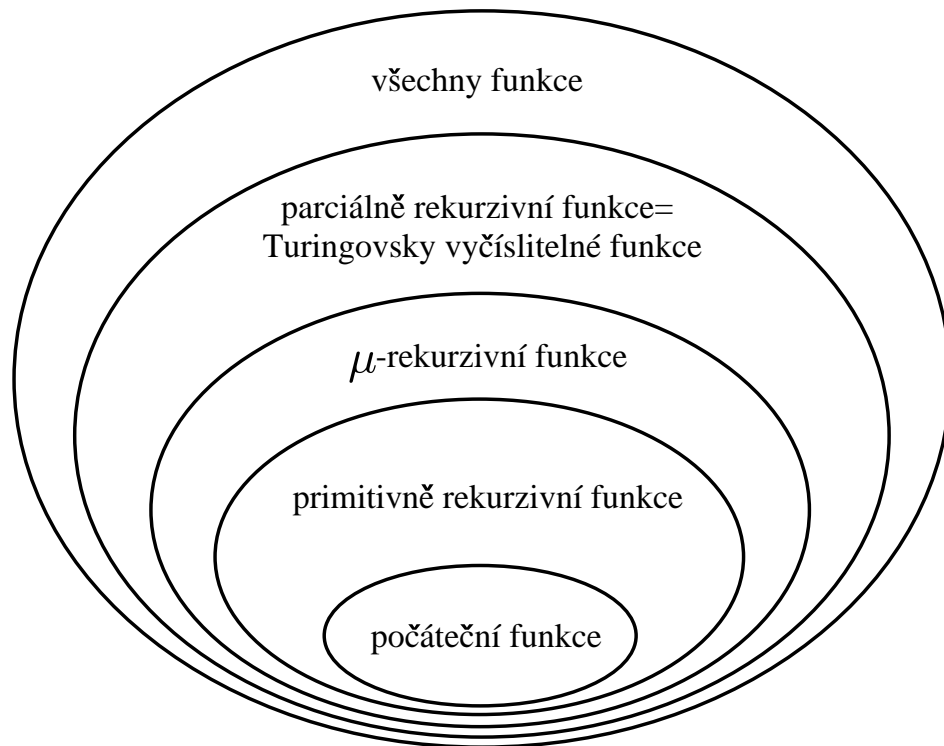
Věta 8.12 Každá parciálně rekurzivní funkce je Turingovsky vyčíslitelná.

Idea důkazu: počáteční funkce, kombinaci, kompozici, projekci, primitivní rekurzi i minimalizaci lze implementovat pomocí Turingova stroje.

Věta 8.13 Každý výpočetní proces prováděný Turingovým strojem je procesem vyčíslení nějaké parciálně rekurzivní funkce.

Idea důkazu: Definujme primitivně rekurzivní funkci provádějící k kroků daného TS. Využijeme techniku minimalizace k výpočtu počtu kroků nutných k zastavení TS (může vrátit nedefinováno).

Hierarchie funkcí



❖ Třída totálních vyčíslitelných funkcí se nazývá *μ -rekurzivní funkce*.

❖ Příklad funkce, která není Turingovsky vyčíslitelná:

Nechť L je libovolný jazyk.

$$\text{Funkce } f(w) = \begin{cases} |w| & \text{jestliže } w \in L \\ 0 & \text{jestliže } w \notin L \end{cases}$$