Luboš Brim

Vyčislitelnost

Zápisy z přednášky zpracoval:
Jan Šerák

16. května 1995

Obsah

1 Úvod

1.1 Náš programovací jazyk

Dříve než začneme rozvíjet teorii vyčislitelnosti, nadefinujme si jakýsi „náš“ programovací jazyk. Pojmenujme si jej LANET (LANguage for Evalutional Theory)[1]. Jazyk LANET se skládá z těchto konstruktů:

- *identifikátory*, kterých je nekonečný počet
- *operátory:* `succ` (unární operátor následníka), `pred` (unární operátor předchůdce), `0` (nulární operátor, znamenající číslenou hodnotu 0).
- *relace* $\neq$, logický binární operátor.
- *přiřazení* $:=$, které lze použít pouze v těchto typech výrazů:
  - `X := 0`
  - `X := pred(Y)`
  - `X := succ(Y)`
- *sekvence:* `begin, end` s použitím typu `begin` $\delta_1;\ldots;\delta_n$ `end`
- *cyklus:* `while, do` s použitím typu `while X`$\neq$`Y do` δ.

Z výše uvedené definice jazyka LANET plyne, že umí pracovat pouze s přirozenými čísly. Množině programů tohoto jazyka se říká *while-programy*.

Dohoda: Bude-li v programu vystupovat k proměnných, budeme je značit $\mathtt{X}_1, \ldots, \mathtt{X}_k$.

1.2 Základní pojmy

Z dřívějších dob patrně víme, co je to *program* P o k proměnných. Nechť a_0 je k-rozměrný vektor nad $\mathbf{N}$. Víme, co je *výpočet* programu P pro a_0; taktéž víme, co znamená, že program *diverguje (neskončí)* pro a_0 a že program *končí* pro a_0 v n krocích a výstup je $a_n \in \mathbf{N}^k$.

Definice 1.1
j-ární sémantická funkce $\varphi_P : \mathbf{N}^j \longrightarrow \mathbf{N}$, kde P je program s k proměnnými, je definována:

- Je-li $k \geq j$, je $(a_1, \ldots, a_j)$ vstupní vektor pro program P, $\varphi_p(a_1, \ldots, a_j)$ je hodnota v proměnné `X1` po skončení P pro vstup $\underbrace{(a_1, \ldots, a_j, 0, \ldots, 0)}_{k}$.
- Je-li $k < j$, je $(a_1, \ldots, a_j)$ vstupní vektor pro program P, $\varphi_p(a_1, \ldots, a_j)$ je hodnota v proměnné `X1` po skončení P pro vstup $(a_1, \ldots, a_k)$.
- $\varphi_p(a_1, \ldots, a_j) = \perp$ (program diverguje).

Definice 1.2
Řekneme, že funkce $\psi : \mathbf{N}^j \longrightarrow \mathbf{N}$ je *(efektivně) vyčislitelná*, je-li $\psi = \varphi_P$ pro nějaký program P.

Churchova hypotéza: ψ je algoritmicky vyčislitelná $\Longleftrightarrow \psi$ je vyčislitelná.

[1] Toto označení jazyka zavádí autor tohoto dokumentu. Na přednáškách L. Brima se zůstává u poněkud nepohodlného a nic neříkajícího označení „náš“ jazyk.

2 Efektivní numerace programů

2.1 Standardní numerace

Motivací efektivní numerace programu je potřeba převádět programy na přirozená čísla a opačně. Je zřejmé, že to jde, nás však bude zajímat, zda lze tuto numeraci zalgoritmizovat.

Převod programu na přirozené číslo je vcelku jednoduchý. Mám-li program, mohu jeho text zakódovat (např. pomocí ASCII) do osmic binárních číslic. Zřetězením binárního zápisu textu programu dostanu binární zápis nějakého přirozeného čísla.

Opačný směr lze provést například takto: Máme přirozené číslo n, které zapíšeme v binární soustavě. Pokud počet cifer binárního rozvoje čísla n není dělitelý osmi, přiřadíme takovému číslu n program „prázdné“ funkce, např:

```
begin while x=x do x:=x end
```

Pokud je počet cifer binárního rozvoje čísla n dělitelný osmi, rozdělím jej na osmice a ty v nějakém předem známém kódu (např. ASCII) převedu na text. Tento získaný text je buď programem jazyka LANET a pak je to hledaný program P_n, nebo to není program jazyka LANET a číslu n přiřadím program prázdné funkce.

Oba tyto převody jsou jednoznačné.

Definice 2.1

Převodům mezi programy a přirozenými čísly (např. jak jsou uvedeny výše) se říká *aritmetizace syntaxe*. Podobně kódování matematických formulí se říká *Gödelizace*.

Definice 2.2 NUMERACE VYČISLITELNÝCH FUNKCÍ

Nechť máme seřazení všech programů pevné arity j do posloupnosti $P_0, P_1, \ldots$. Touto posloupností je definována také posloupnost vyčislitelných funkcí $\varphi_{P_0}, \varphi_{P_1}, \ldots$ vyčislovaných příslušnými programy. Takové posloupnosti se říká *numerace vyčislitelných funkcí*. *Standardní* numerace vyčislitelných funkcí arity 1 je posloupnost funkcí $\varphi_0, \varphi_1, \ldots$.

Jiný příklad kódování programů do přirozených čísel:

- `[begin end]` $\longrightarrow 1$
- `[x`$_i$ `:= 0]` $\longrightarrow 2^i$
- `[x`$_i$ `:= succ(x`$_j$`)]` $\longrightarrow 3^i \cdot 5^j$
- `[x`$_i$ `:= pred(x`$_j$`)]` $\longrightarrow 7^i \cdot 11^j$
- `[while x`$_i$ `= x`$_j$ `do` δ`]` $\longrightarrow 13^i \cdot 17^j \cdot 19^{[\delta]}$
- `[begin` $\delta_1; \ldots; \delta_n$ `end]` $\longrightarrow \mathrm{pr}(8)^{[\delta_1]} \cdots \mathrm{pr}(7+n)^{[\delta_n]}$, kde $\mathrm{pr}(i)$ je i-té liché prvočíslo.

2.2 Halting problem

Definice 2.3

Problém zastavení (Halting problem) je rozhodnutí, zda program, vyčíslující vyčislitelnou funkci, zastaví pro její vlastní index. Problém zastavení je jinými slovy funkce:

$$f(i) = \begin{cases} 1 & \varphi_i(i) \text{ je def} \\ 0 & \text{jinak} \end{cases}$$

Věta 2.4

Halting problem nelze algoritmicky rozhodnout. Jinými slovy funce f není vyčislitelná.

Důkaz: povedeme sporem. Předpokládejme, že f je vyčislitelná programem `HALT`. Nyní si nadefinujme funkci ψ takto:

$$\psi(i) = \begin{cases} \perp & f(i) = 1 \\ 1 & f(i) = 0 \end{cases}$$

Funkce ψ je samozřejmě vyčislitelná, např. programem `PSI`:

```
begin while HALT(X1) = 1 do X1 := X1; X1 := 1; end
```

Jelikož ψ je vyčislitelná funkce, existuje index e takový, že $\psi = \varphi_e$. Potom hodnota $\psi(e)$ bude taková:

$$\psi(e) = \begin{cases} \perp & \psi(e) \text{ je def} \\ 1 & \text{jinak} \end{cases}$$

Čímž jsme obdrželi kýžený spor. Program `HALT` proto neexistuje a tím pádem funkce f není vyčislitelná a Halting problem je nerozhodnutelný. □

2.3 Věta o numeraci

Věta 2.5 O NUMERACI (O UNIVERZÁLNÍ FUNKCI)
Pro $\forall j \geq 1$ existuje vyčislitelná funkce $\Phi : \mathbf{N}^{j+1} \longrightarrow \mathbf{N}$ taková, že pro $\forall e \in \mathbf{N}$, $\forall (a_1, \ldots, a_j) \in \mathbf{N}^j$ platí

$$\Phi(e, a_1, \ldots, a_j) = \varphi_e(a_1, \ldots, a_j)$$

. Funkce Φ se nazývá *univerzální* funkce pro standardní numeraci j-árních vyčislitelných funkcí.

Důkaz: Důkazem nám bude stačit náčrt programu, který vyčisluje funkci Φ:

1. Podle hodnoty e proměnné `X1` nalezneme (zkonstruujeme např. pomocí aritmetizace syntaxe) příslušný program P_e.
2. Simulujeme činnost programu P_e na základě jeho textu.
3. Do proměnné `X1` přiřadíme odpovídající výsledek.

□

Tvrzení 2.1
Pro $\forall j \geq 1$ existuje přirozené číslo r a totální vyčislitelná funkce *redukce* $red : \mathbf{N} \longrightarrow \mathbf{N}$ taková, že platí:

1. $\varphi_e^{(j)} = \varphi_{red(e)}^{(j)}$
2. Program $P_{red(e)}$ používá $j + r$ proměnných.

Důkaz: spočívá v kódování vektorů. Například všechny vektory z $\mathbf{N}^2$, uložené v této matici nekonečného řádu:

$$\begin{pmatrix} (0,0) & (0,1) & (0,2) & \cdots \\ (1,0) & (1,1) & (1,2) & \cdots \\ (2,0) & (2,1) & (2,2) & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

lze zakódovat po diagonálách do matice téhož řádu:

$$\begin{pmatrix} 0 & 2 & 5 & \cdots \\ 1 & 4 & 8 & \cdots \\ 3 & 7 & 12 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

Zde jsme užili kódovací funkce $\tau(i,j) = \frac{1}{2}(i+j)(i+j+1) + i$, která se nazývá *párující* funkce. Párující funkce τ i její projekce π_1 a π_2 jsou samozřejmě vyčislitelné funkce.

Tímto způsobem lze ukládat hodnoty dvou proměnných pouze do jedné proměnné a indukcí dostáváme požadované tvrzení. □

2.4 Věta o parametrizaci

Nyní nás bude zajímat tento problém: Mějme funkci $f(x,y)$, u níž zablokujeme argument x na nějaké konstantě c. Pak funkce $g(y) = f(c,y)$ vznikla parametrizací prvního argumentu funkce f.

Věta 2.6 O PARAMETRIZACI (SMN-VĚTA)
Existuje totální vyčislitelná funkce $s_n^m : \mathbf{N}^{m+1} \longrightarrow \mathbf{N}$ taková, že platí:

$$\varphi^{(n)}_{s_n^m(i,y_1,\ldots,y_m)}(z_1,\ldots,z_n) = \varphi_i^{(m+n)}(y_1,\ldots,y_m,z_1,\ldots,z_n)$$

Důkaz: Nechť máme nějakou vyčislitelnou funkci φ_i a ji vyčislující program P_i. Dále nechť máme program Q:

```
begin Xm+n := Xn; ...; Xm+1 := X1; X1 := y1; ...; Xn := yn; end;
```

Tento program nám přetransformuje vektor $(z_1,\ldots,z_n,0,\ldots,0)$ do vektoru $(y_1,\ldots,y_m,z_1,\ldots,z_n)$, kde $y_1,\ldots,y_m$ jsou konstanty. Program P_i' je pak:

```
begin Q; Pi; end
```

Tento program vyčisluje funkci $\varphi^{(n)}_{s_n^m(i,y_1,\ldots,y_m)}(z_1,\ldots,z_n)$, čímž jsme dokázali vyčislitelnost této funkce.

Nechť i' je index programu P_i'. Pak $i' = s_n^m(i,y_1,\ldots,y_n)$. Program P_i' umím napsat kdykoli, to znamená, že funkce s_n^m je totální. S odvoláním na hypotézu pana Churche, že umím pro libovolná m, n napsat program Q, je funkce s vyčislitelná. □

Na závěr ještě poznamenejme, že obdobné tvrzení platí i o skládání funkcí (sekvenci programů).

2.5 Totální numerace

Definice 2.7
Numerace μ množiny S je surjektivní funkce $\mu : \mathbf{N} \longrightarrow S$.

Definice 2.8 PROGRAMOVACÍ JAZYK
Nechť $\mathbf{P}^j$ je množina všech j-árních vyčislitelných funkcí. Dvojici $L = (TP, \varphi')$ nazveme *programovacím jazykem pro* $\mathbf{P}^1$, kde $\mathbf{N} \supseteq TP$ je množina programů a φ' je totální numerace množiny $\mathbf{P}^1$. TP nazýváme *syntax* a φ' *sémantika* programovacího jazyka[2].

V odstavci 2.1 jsme zkonstruovali $L = (\mathbf{N}, \varphi)$. Programovací systémy lze uspořádat: např. nechť $\varphi_0', \varphi_1', \ldots$ jsou funkce vyčislitelné konečnými automaty a nechť $\varphi_0'', \varphi_1'', \ldots$ jsou funkce vyčislitelné Turingovými stroji. Pak ke každé funkci φ_n' lze najít funkci φ_m'' takovou, že $\varphi_n' = \varphi_m''$. Opačně toto tvrzení neplatí. Z toho tedy plyne, že Turingovy stroje jsou silnější než konečné automaty. Podobně lze zatřídit všechny programové systémy.

Definice 2.9
Nechť máme μ_1, μ_2 resp. S_1, S_2. Řekneme, že μ_1 se *redukuje* na μ_2 (zapíšeme $\mu_1 \leq \mu_2$), jestliže existuje vyčislitelná funkce $r \in \mathbf{P}^1$ taková, že pro $\forall i \in \Re(\mu_1)$ [3] platí: $\mu_1(i) = \mu_2(r(i))$. Řekneme, že μ_1 a μ_2 jsou *ekvivalentní* numerace $(\mu_1 \equiv \mu_2) \Longleftrightarrow \mu_1 \leq \mu_2 \wedge \mu_1 \geq \mu_2$.

Věta 2.10
Nechť ψ, ψ' jsou totální numerace množiny $\mathbf{P}^1$. Pak:

1. Jestliže ψ splňuje větu o numeraci a ψ' splňuje větu o parametrizaci, pak ψ se redukuje na ψ' $(\psi \leq \psi')$.
2. $\psi \equiv \psi' \Longleftrightarrow \psi$ splňuje obě tyto věty současně.

Důkaz: Důkaz není uveden. □

Definice 2.11
Numerace ψ množiny $\mathbf{P}^1$ se nazývá *efektivní (přípustná)*, jestliže $\psi \equiv \varphi$.

Teze: Numerace ψ je intuitivně efektivní $\Longleftrightarrow$ je efektivní.

[2] $w \in TP \Longleftrightarrow \varphi'(w) \in \mathbf{P}^1$
[3] $\Re$ je definiční obor *(range)*, $\Im$ je obor hodnot *(image)*.

3 Nerozhodnutelné problémy

Věta 3.1
Existuje totální funkce $f : \mathbf{N} \longrightarrow \mathbf{N}$, která není vyčislitelná.

Důkaz: Uvažujme numeraci (ne nutně efektivní) všech vyčislitelných totálních funkcí $\mathbf{N} \longrightarrow \mathbf{N}$ takto: mám $\varphi_0, \varphi_1, \ldots$ a konstruuji posloupnost $f_1, f_2, \ldots$ totálních vyčislitelných funkcí tak, že f_{n+1} je první totální φ_j po f_n. Máme tedy posloupnost totálních vyčislitelných funkcí.

Nyní definujme totální funkci $f : \mathbf{N} \longrightarrow \mathbf{N}$ takto: $f(i) = f_i(i) + 1$. Funkce f je samozřejmě totální, ale od každé totální vyčislitelné funkce se liší $\Longrightarrow$ funkce f není vyčislitelná. □

Věta 3.2
. Nechť $f_0, f_1, \ldots$ je efektivní numerace, ve které f_i jsou totální vyčislitelné funkce $\mathbf{N} \longrightarrow \mathbf{N}$. Pak existuje totální vyčislitelná funkce $f : \mathbf{N} \longrightarrow \mathbf{N}$, která není v této numeraci. To znamená, že neexistuje efektivní numerace všech totálních vyčislitelných funkcí.

Důkaz: Existuje totální vyčislitelná funkce $g : \mathbf{N} \longrightarrow \mathbf{N}$ taková, že $f_i = \varphi_{g(i)}$. Definujeme funkci $f : \mathbf{N} \longrightarrow \mathbf{N}$ tak, že $f(i) = f_i(i) + 1$. Funkce f je totální vyčislitelná funkce, vyčisluje ji např. program:

```
begin X2 := g(X1); X1 := Φ(X2,X1); X1 := succ(X1); end
```

Tím je věta dokázána. □

3.1 Problém totálnosti

Věta 3.3
Problém totálnosti vyčislitelných funkcí je nerozhodnutelný. Jinými slovy neexistuje algoritmus, který pro daný index i rozhodne, zda je vyčislitelná funkce $\varphi_i : \mathbf{N} \longrightarrow \mathbf{N}$ totální.

Důkaz: Uvažujme totální funkci total(i), která nabývá hodnoty 1 je-li funkce φ_i totální a hodnoty 0 v opačném případě. Musíme ukázat, že funkce total není vyčislitelná. To dokážeme sporem.

Předpokládejme, že total je vyčislitelná. Pak můžeme definovat funkci $g : \mathbf{N} \longrightarrow \mathbf{N}$, tak že $g(0)$ je nejmenší i takové, že total$(i) = 1$, ... $g(n+1)$ je nejmenší i takové, že $i > g(n)$ a total$(i) = 1$. Tedy g je numerace totálních vyčislitelných funkcí a je vyčislitelná například programem:

```
begin
  X2 := succ(X1);
  X1 := 0;
  while X2 != 0 do
    begin
      while total(X1) = 0 do X1 := succ(X1);
      X2 := pred(X2);
      X1 := succ(X1);
    end;
  X1 := pred(X1);
end;
```

Tím jsme dostali posloupnost $\varphi_{g(0)}, \varphi_{g(1)}, \ldots$, což je efektivní numerace všech totálních vyčislitelných funkcí.

To je samozřejmě spor s větou 3.2. □

3.2 Problém verifikace

Věta 3.4
Problém verifikace je nerozhodnutelný. Jinými slovy neumím rozhodnout, zda daný program počítá danou funkci.

Důkaz: Můžeme „redukovat“ na problém verifikace identity. Nechť funkce $e(i)$ nabývá hodnoty 1, je-li funkce φ_i identitou, a hodnoty 0 v opačném případě. Musíme dokázat, že e není vyčislitelná.

To dokážeme tak, že redukujeme problém totálnosti na problém verifikace a dojdeme k závěru, že pokud bychom uměli rozhodnout verifikaci, uměli bychom rozhodnout také totálnost.

Redukci provedeme nalezením vyčislitelné funkce g takové, pro niž platí, že φ_i je totální $\iff$ $\varphi_{g(i)}$ je identita. Mějme tedy program $P_{g(i)}$:

```
begin X2 := Φ(i,X1) end
```

Je zřejmé, že funkce g je vyčislitelná. Program $P_{g(i)}$ vyčisluje funkci $\varphi_{g(i)}$, která vypadá takto:

$$\varphi_{g(i)}(j) = \begin{cases} j & \text{if } \varphi_i(j) \text{ is total} \\ \perp & \text{else} \end{cases}$$

Tedy $\varphi_{g(i)}$ je identita tehdy a jen tehdy, je-li φ_i totální. □

3.3 Problém ekvivalence

Věta 3.5

Problém ekvivalence je nerozhodnutelný. Jinými slovy neexistuje program, který pro i a j rozhodne, zda $\varphi_i = \varphi_j$.

Důkaz: Nejprve problém ekvivalence redukujeme na jednodušší problém ekvivalence funkce φ_i s funkcí φ_{j_0}, která je vyčislovaná programem P_{j_0}:

```
begin end
```

Tuto „redukci“ jsme provedli jen na intuitivní úrovni, protože pokud umím rozhodnout ekvivalenci funkcí φ_i a φ_j, umím rozhodnout i ekvivalenci funkcí φ_i a φ_{j_0}.

Tím dostáváme tvrzení, že funkce φ_i a φ_{j_0} jsou ekvivalentní tehdy a jen tehdy, když je funkce φ_i identitou. To však rozhodnout neumím, neumím tedy ani rozhodnout ekvivalenci s identitou, tím spíš neumím rozhodnout obecnou ekvivalenci. □

4 Vyčislistelné vlastnosti množin

4.1 Rekurzivní množiny

Definice 4.1
Nechť $A \subseteq \mathbf{N}^k$. Charakteristická funkce množiny A je funkce $\chi_A : \mathbf{N}^k \longrightarrow \{\texttt{true}, \texttt{false}\}$ taková, že:

$$\chi_A(x) = \begin{cases} \texttt{true} & \text{if } x \in A \\ \texttt{false} & \text{else} \end{cases}$$

Definice 4.2
Množina $A \subseteq \mathbf{N}^k$ je *rozhodnutelná (rekurzivní)*, jestliže její charakteristická funkce χ_A je totální vyčislitelná.

Příklad 4.1

- množina $\{(a,b) \mid b = a^2\}$ je rekurzivní
- množina $\{i \mid \varphi_i \text{ is total}\}$ není rekurzivní
- množina $\{(i,j) \mid \varphi_i = \varphi_j\}$ není rekurzivní

Lemma 4.3
Je-li množina A konečná nebo je-li množina $\bar{A}$ konečná, pak množina A je rekurzivní.

Důkaz: Triviálně konečnou množinu $|A| = n$ mohu rozhodovat jedním `ifem` s disjunkcí n porovnání v podmínce, podobně pro $\bar{A}$. □

Lemma 4.4
Je-li A rekurzivní, pak i $\bar{A}$ je rekurzivní.

Důkaz: Opět triviálně „znegováním“ programu charakteristické funkce χ_A dostanu program funkce $\chi_{\bar{A}}$. □

Lemma 4.5
Jsou-li A_1 a A_2 rekurzivní množiny, pak i $A_1 \cup A_2$ a $A_1 \cap A_2$ jsou rekurzivní.

Důkaz: je opět triviální. Do proměnných `X2, X3` uložím výsledky charakteristických funkcí χ_{A_1} a χ_{A_2} a potom do proměnné `X1` uložím výsledek `X2 or X3` resp. `X2 and X3`. □

4.2 Rekurzivně vyčislitelné množiny

Definice 4.6
Množina $A \subseteq \mathbf{N}$ je *parciálně rozhodnutelná (rekurzivně vyčislitelná, r. e., RE-množina)*, jestliže $A = \emptyset$ nebo $A = \Im(f)$ pro nějakou totálně vyčislitelnou funkci $f : \mathbf{N} \longrightarrow \mathbf{N}$. Taková funkce f se nazývá *numerující* funkce.

Příklad 4.2
Identita je numerující funkce pro $\mathbf{N}$.

Lemma 4.7
Každá rekurzivní množina A je rekurzivně vyčislitelná.

Důkaz: Případ $A = \emptyset$ je zřejmý. Nechť tedy $A \neq \emptyset$, pak existuje charakteristická funkce χ_A. Nechť $a_0 \notin A$. Pak funkce f zkonstruujeme takto:

$$f(n) = \begin{cases} n & \text{if } n \in A \\ a_0 & \text{else} \end{cases}$$

Funkce f je zřejmě vyčislitelná: např. programem

```
begin if not χA(X1) then X1 := a0; end
```

Tvrzení, že f je numerující funkcí množiny A plyne ihned z definice ($\Im(f) = A$). □

Důkaz lemmatu 4.7 dává návod na sestrojení programů – generátorů RE-množin. Nechť program P s k proměnnými vyčísluje numerující funkci $f : \mathbf{N} \longrightarrow \mathbf{N}$ množiny A. Pak program Q:

```
begin
  n := 0;
  while true do begin
    X1 := n; X2 := 0; ... Xk := 0;
    P;
    output(X1);
    n := succ(n);
  end;
end
```

generuje hodnoty $f(0), f(1), \ldots$. Když je f totální, je to bez problémů; f však nutně nemusí být totální, což přináší problém v tom, že neumím zjistit, zda je nějaká funkce totální.

Lemma 4.8
Funkce

$$Se(x,y,z) = \begin{cases} 1 & P_x \text{ stops for } y \text{ after maximum } z \text{ steps} \\ 0 & \text{else} \end{cases}$$

je totální vyčíslitelná.

Důkaz: Se je zřejmě totální. Vyčísluje ji program, který v numeraci najde program P_x a interpretuje jej pro vstup y po dobu nejvýše z kroků. □

Předchozí lemma nám dává návod, jak upravit generátor RE-množin Q:

```
begin
    n := 0;
    while true do begin
       x := π1(n); y := π2(n);
       #π1 a π2 projekce z kapitoly 2
       if Se(e,x,y) = 1 then begin
       #e je index programu P
          P(x);
          output(X1);
       end;
       n := succ(n);
    end;
end;
```

Nyní takto upravený program Q negeneruje řadu $f(0), f(1), \ldots$ takto uspořádaně, zato však vygeneruje všechny prvky množiny $\Im(f)$, i když f není totální.

Lemma 4.9
$K = \{i \mid \varphi_i(i) \neq \perp\}$ je rekurzivně vyčíslitelná, ale není rekurzívní.

Důkaz: Rekurzivní být nemůže, protože problém zastavení není rozhodnutelný. Množinu K však umím vygenerovat programem, který vychází z programu Q:

```
begin
    n := 0;
    while true do begin
       x := π1(n); y := π2(n);
       if Se(x,x,y) = 1 then output(x)
       n := succ(n);
    end;
end;
```

□

4.3 Další vlastnosti rekurzivně vyčislitelných množin

Obrázek 1: Třídy množin zhlediska vyčislitelnosti

Co jsme se zatím dověděli ukazuje obrázek 1. Nevíme, zda existuje nějaká třída množin, která je nadtřídou třídy RE-množin, nebo zda se třída RE-množin rovná třídě všech podmnožin $\mathbf{N}$. V tomto odstavci si ukážeme, že existuje množina, která není rekurzivně vyčislitelná. Dále si uvedeme užitečné vztahy mezi rekurzivními a rekurzivně vyčislitelnými množinami a užitečná kriteria RE-množin.

Lemma 4.10
Množina $\{i \mid \varphi_i : \mathbf{N} \longrightarrow \mathbf{N} \text{ is total}\}$ není rekurzivně vyčislitelná.

Důkaz: plyne z toho, že efektivní numerace je seznam. A jelikož efektivní numerace totálních funkcí neexistuje, nemůže být tato množina r. e. □

Lemma 4.11
Nechť $A \subseteq \mathbf{N}$. Pak platí ekvivalence: A je rekurzivní $\Longleftrightarrow$ A, $\bar{A}$ jsou rekurzivně vyčislitelné.

Důkaz:

$\Longrightarrow$: A je rekurzivní $\Longrightarrow$ $\bar{A}$ je rekurzivní $\Longrightarrow$ A i $\bar{A}$ jsou rekurzivně vyčislitelné.

$\Longleftarrow$: Je-li $A = \emptyset$ nebo $\bar{A} = \emptyset$, pak A je zřejmě rekurzivní. Nechť tedy $A \neq \emptyset$ a $\bar{A} \neq \emptyset$. Pak A, $\bar{A}$ jsou rekurzivně vyčislitelné $\Longleftrightarrow$ existují funkce f a g tak, že $A = \Im(f)$ a $\bar{A} = \Im(g)$ a navíc platí:

$$\begin{aligned} \Im(f) \cup \Im(g) &= \mathbf{N} \\ \Im(f) \cap \Im(g) &= \emptyset \end{aligned}$$

Jak tedy rozhodneme, zda $x \in A$? Program, který bude množinu A rozhodovat, bude generovat posloupnost $f(0), g(0), f(1), g(1), \ldots$ a v konečném čase dojde k prvku x. Pokud zjistí, že x je f-hodnota, pak odpoví $x \in A$; pokud zjistí, že x je g-hodnota, pak odpoví $x \notin A$.

Z existence tohoto programu ihned plyne rekurzivita A.

□

Důsledek 4.12
$\bar{K}$ (problém nezastavení) není rekurzivně vyčislitelná množina.

Důkaz: Přímo z lemmatu 4.11. □

Definice 4.13
Množina $A \subseteq \mathbf{N}$ se nazývá *rekurzivně vyčislitelná v rostoucím pořádku*, jestliže má rostoucí numerující funkci.

Lemma 4.14
Nekonečná množina $A \subseteq \mathbf{N}$ je rekurzivní $\Longleftrightarrow$ A je rekurzivně vyčislitelná v rostoucím pořádku.

Důkaz:

$\Longrightarrow$: Rostoucí posloupnost prvků $a_0, a_1, a_2, \ldots$ množiny A ($\chi_A(a_i) = \texttt{true}$), je rostoucí numerace množiny A.

$\Longleftarrow$: Nechť f je rostoucí numerující funkce množiny A. Nepřítel nám dal za úkol rozhodnout, zda $x \in A$. Začneme tedy generovat posloupnost $f(0), f(1), f(2), \ldots$. Pokud v této posloupnosti narazím na x, odpovím, že $x \in A$. Pokud dojdu k takovému i, že $f(i) > x$, vím, že již nemá cenu dál hledat, a odpovím, že $x \notin A$.

Výše popsaný algoritmus rozhoduje množinu A.

□

Důsledek 4.15
Každá nekonečná RE-množina má nekonečnou rekurzivní podmnožinu.

Důkaz: Stačí nám najít množinu $B \subseteq A$ rekurzivně vyčíslitelnou v rostoucím pořadí. Existuje numerující funkce f množiny A: $A = \Im(f)$. Množinu B můžeme generovat takto:

1. `output(f(0))`
2. `if f(1) > f(0) then output(f(1))`

   ...

n+1. `if f(n) > max(f(0),...,f(n-1)) then output(f(n))`

□

Věta 4.16

1. Množina A je RE-množina $\iff A = \Re(f)$, kde f je vyčíslitelná funkce.
2. Množina A je RE-množina $\iff A = \Im(f)$, kde f je vyčíslitelná funkce.

Důkaz:

1. $\Longrightarrow$: $A = \emptyset$ je zřejmé, protože $\emptyset$ je doménou prázdné funkce. Nechť $\emptyset \neq A = \Im(f)$, kde f je totální vyčíslitelná funkce. Definujeme funkci Θ:
$$\Theta(i) = \begin{cases} 1 & \text{if } i \in \Im(f) \\ \perp & \text{else} \end{cases}$$
Je nabíledni, že $\Re(\Theta) = \Im(f) = A$. Zbývá tedy jen dokázat např. konstrukcí programu, že funkce Θ je vyčíslitelná:
```
begin X2 := X1; X1 := 1; n:= 0;
    while X2 ≠ f(n) do n := succ(n);
end
```
$\Longleftarrow$: Nechť $A = \Re(\Theta)$ pro vyčíslitelnou funkci Θ. Je-li $\Re(\Theta) = \emptyset$, je tvrzení věty splněno triviálně. Nechť tedy $\Re(\Theta) \neq \emptyset$, tj. $\exists a_0 \in A$. Mějme program:
```
begin
    x := π1(X1); y := π2(X1);
    if výpočet Θ(x) končí během y kroků
       then X1 := x;
       else X1 := a0;
end
```
Označme funkci, kterou tento program vyčisluje, f. Pak zřejmě platí $\Im(f) = \Re(\Theta)$.

2. Důkaz tohoto tvrzení je zřejmý.

□

Připomeňme si standardní numeraci vyčíslitelných funkcí $\varphi_0, \varphi_1, \ldots$ Pak předchozí věta předurčuje existenci dvou standardních numerací rekurzivně vyčíslitelných funkcí:

1. $\Re(\varphi_0), \Re(\varphi_1), \ldots$
2. $\Im(\varphi_0), \Im(\varphi_1), \ldots$

Lemma 4.17
Existují totální vyčíslitelné funkce $t_1, t_2 : \mathbf{N} \longrightarrow \mathbf{N}$ takové, že pro $\forall i \in \mathbf{N}$ platí:
$$\begin{aligned} \Re(\varphi_i) &= \Im(\varphi_{t_1(i)}) \\ \Im(\varphi_i) &= \Re(\varphi_{t_2(i)}) \end{aligned}$$

Důkaz:

1. Nechť $t_1(i)$ je index programu:

   ```
   begin X2 := Φ(i,X1); end
   ```

   Tedy pro všechna a platí: program $P_{t_1(i)}$ končí pro $a \iff$ program P_i končí pro a. Rovnost $\Im(\varphi_{t_1(i)}) = \Re(\varphi_i)$ je zřejmě platná.

2. Nechť $t_2(i)$ je index programu:

   ```
   begin
       n := 0; X2 := succ(X1);
       while X1 ≠ X2 do begin
          x := π1(n); y := π2(n);
          if Se(i,x,y) = 1 then X2 := Φ(i,x);
          n := succ(n);
       end;
   end
   ```

   Tento program pro vstup a hledá b takové, že $a = \varphi_i(b)$. Tedy platí rovnost $\Im(\varphi_i) = \Re(\varphi_{t_2(i)})$.

□

Definice 4.18
Označme $W_i = \Re(\varphi_i)$. Posloupnost $W_0, W_1, \ldots$ nazýváme *standardní numerace RE-množin.*

Na závěr tohoto odstavce a celé kapitoly upozorněme na tento fakt:

- W_i je akceptovaná programem P_i (P_i zastaví pro libovolný prvek $x \in W_i$);
- W_i je generovaná programem $P_{t_1(i)}$.

5 Uzávěrové vlastnosti RE-množin

5.1 Uzávěry RE-množin

Definice 5.1
Nechť $A \subseteq \mathbf{N}$ a $\Theta : \mathbf{N} \longrightarrow \mathbf{N}$. *Parciální obraz* množiny A je množina $\Theta(A) = \{\Theta(a) \mid a \in \Re(\Theta) \wedge a \in A\}$ a *parciální vzor* množiny A je množina $\Theta^{-1}(A) = \{b \mid b \in \Re(\Theta) \wedge \Theta(b) \in A\}$.

Věta 5.2
Existují totální vyčislitelné funkce $g_1 : \mathbf{N}^2 \longrightarrow \mathbf{N}$, $g_2 : \mathbf{N}^2 \longrightarrow \mathbf{N}$ takové, že pro rekruzivně vyčislitelnou množinu W_j a vyčislitelnou funkci φ_i platí:

1. $\varphi_i(W_j) = W_{g_1(i,j)}$
2. $\varphi_i^{-1}(W_j) = W_{g_2(i,j)}$

Důkaz:

1. $\varphi_i(W_j) = \{\varphi_i(a) \mid a \in W_i \cap W_j\}$. Nechť číslo $\hat{g}(i,j)$ je index programu:

   ```
   begin X2 := Φ(j,X1); X1 := Φ(i,X1); end
   ```

   Tento program vyčisluje φ_i za předpokladu, že je pro vstup a definována hodnota $\varphi_j(a)$ i $\varphi_i(a)$.

   Potom tedy platí:

   $$\varphi_i(W_j) = \Im(\varphi_{\hat{g}(i,j)}) = \Re(\varphi_{t_2(\hat{g}(i,j))})$$

   a tedy funkce $g_1 = t_2 \circ \hat{g}$.

2. Platí, že $\varphi_i^{-1}(W_j) = \Re(\varphi_j \circ \varphi_i)$. Potom podle věty o parametrizaci (Věta 2.6) existuje funkce totální vyčislitelná funkce $g_2 : \mathbf{N} \longrightarrow \mathbf{N}$ taková, že $\varphi_{g_2(i,j)} = \varphi_j \circ \varphi_i$. To ovšem znamená:

   $$\varphi_i^{-1}(W_j) = \Re(\varphi_{g_2(i,j)}) = W_{g_2(i,j)}$$

□

Věta 5.3
RE-množiny nejsou uzavřené vzhledem k doplňku.

Důkaz: Problém zastavení K je RE-množina, problém nezastavení $\bar{K}$ není RE-množina. □

Věta 5.4
Existují totální vyčislitelné funkce $\mathsf{union} : \mathbf{N}^2 \longrightarrow \mathbf{N}$ a $\mathsf{intersect} : \mathbf{N}^2 \longrightarrow \mathbf{N}$ taková, že pro dané RE-množiny W_i a W_j platí:

$$\begin{aligned} W_i \cup W_j &= W_{\mathsf{union}(i,j)} \\ W_i \cap W_j &= W_{\mathsf{intersect}(i,j)} \end{aligned}$$

5.2 Projekce a sekce

Definice 5.5
Nechť $R \subseteq \mathbf{N}^k$ je k-ární relace. Pak množinu:

$$\{(a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_k) \mid \exists a_i \in \mathbf{N} : \ (a_1, \ldots, a_k) \in R\}$$

nazýváme *i-tou projekcí R.*

Nechť $b \in \mathbf{N}$. Pak množinu

$$\{(a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_k) \mid (a_1, \ldots, a_{i-1}, b, a_{i+1}, \ldots, a_k) \in R\}$$

nazveme *i-tou sekcí R pro b.*

Věta 5.6

1. Libovolná sekce rekurzívní relace je rekurzívní.
2. Kartézský součin $A \times B$ dvou RE-množin je RE-množina.
3. Libovolná sekce RE-relace je rekurzivně vyčislitelná.
4. Libovolná projekce RE-relace je rekurzivně vyčislitelná.

Příklad 5.1

Nechť $S \subseteq \mathbf{N}^3$ taková, že $(x,y,z) \in S \iff$ program P_x pro vstup y neudělá více než z kroků. Pak Se je charakteristická funkce relace S, která je totální vyčislitelná. Tedy S je rekurzivní.

3. projekce: $\{(x,y) \mid \exists z : (x,y,z) \in S\}$. Tato množina je však rovna $\{(x,y) \mid \varphi_x(y) \neq \perp\}$, což je zoběcněný halting problem. 3. projekce relace S tedy není rekurzivní.

Příklad 5.2

Nechť $A = \{i \mid 7 \in \Im(\varphi_i)\}$. A je množina programů, které někdy spočítají číslo 7; je tedy rekurzivně vyčislitelná. Dále nechť B je množina trojic (i,j,k) takových, že program P_i pro vstup j zastaví během k kroků a $\varphi_i(j) = 7$. Množina B je rekurzivní, tedy i rekurzivně vyčislitelná.

Nyní si uvědomme, že A je 2. a 3. projekcí B a obě množiny jsou RE-množiny.

Věta 5.7

Relace je rekurzivně vyčislitelná $\iff$ je projekcí rekurzivní relace.

Důkaz:

$\Longleftarrow$: zřejmé

$\Longrightarrow$: $W_i = \Re(\varphi_i)$ je množina takových a, že existuje k takové, že program P_i zastaví pro a během k kroků. Nechť relace $R(a,k)$ je množina a takových, že program P_i zastaví pro a během k kroků. Relace $R(a,k)$ je zřejmě rekurzivní.

Množina W_i je 2. projekcí relace $R(a,k)$ a je tedy rekurzivně vyčislitelná dle tvrzení předchozí věty.

□

Definice 5.8

j-ární relace $R \in \mathbf{N}$ se nazývá *diofantická*, jestliže existuje diofantická rovnice[4] $P(x_1,\ldots,x_j,y_1,\ldots,y_k) = 0$ o $j+k$ poměnných taková, že

$$(a_1,\ldots,a_j) \in R \iff \exists y_1,\ldots,y_k : \; P(a_1,\ldots,a_j,y_1,\ldots,y_k) = 0$$

Věta 5.9

Relace je rekurzivně vyčislitelná $\iff$ je diofantická.

[4] *Diofantická rovnice* je rovnice tvaru $P = 0$, kde P je polynom s celočíselnými koeficienty.

6 Riceova věta

6.1 Množiny respektující funkce

Definice 6.1
Množina $I \subseteq \mathbf{N}$ *respektuje funkce*, jestliže platí implikace:

$$i \in I,\ \varphi_i = \varphi_j \Longrightarrow j \in I$$

Poznámka 6.2
Zřejmě platí, že I respektuje funkce $\Longrightarrow \bar{I}$ respektuje funkce.

Poznámka 6.3
Množiny $\{i \mid 7 \in \Im(\varphi_i)\}$, $\{i \mid \varphi_i \text{ is total}\}$ a $\{i \mid \varphi_i = \emptyset\}$ respektují funkce.
Množina K nerespektuje funkce. Proč? Dejme tomu, že máme e takové, že $W_e = \{e\}$. Pak nechť existuje e' takové, že $e' \neq e$ a přitom $\varphi_{e'} = \varphi_e$. Platí, že $e \in K$ a přitom $e' \neq K$.

Věta 6.4
Nechť $I \subseteq \mathbf{N}$ respektuje funkce. Pak I je rekurzivní $\Longleftrightarrow I = \emptyset \vee I = \mathbf{N}$.

Důkaz: Nechť $\emptyset \neq I \neq \mathbf{N}$. Ukážeme sporem, že I není rekurzivní. Předpokládejme, že I obsahuje všechny indexy nějaké vyčislitelné funkce Θ různé od prázdné funkce μ. Pak $\bar{I}$ obsahuje všechny indexy pro μ.
Mějme dále program $P_{f(i)}$ takový:

```
begin X2 := Φ(i,i); X1 := Θ(X1); end
```

Program $P_{f(i)}$ vyčisluje funkci $\varphi_{f(i)}$:

$$\varphi_{f(i)} = \begin{cases} \Theta & \text{if } \varphi_i(i) \neq \perp \\ \mu & \text{else} \end{cases}$$

Zřejmě platí: $f(i) \in I \Longleftrightarrow \varphi_i(i)$ je definováno pro $\forall i \in \mathbf{N}$. Charakteristická funkce množiny I je funkce χ_I. Ta se však – aplikovaná na $f(i)$ – chová jistým speciálním způsobem:

$$\chi_I(f(i)) = \begin{cases} 1 & \text{if } \varphi_i(i) \neq \perp \\ 0 & \mathit{else} \end{cases}$$

a to pro $\forall i \in \mathbf{N}$. Na první pohled vidíme, že $\chi_I \circ f = \texttt{halt}$. Jelikož však f je totální vyčislitelná, měla by funkce $\chi_I \circ f$ bez problémů vyčislovat množinu I. Jelikož však není tato funkce vyčislitelná, není množina I rekurzivní. Došli jsme tedy ke kýženému sporu. □

Příklad 6.1
Následující množiny a jejich negace nejsou rekurzivní:

- $A_1 = \{i \mid i \text{ is total}\}$.
- $A_2 = \{i \mid i = f\}$, kde f je totální vyčislitelná funkce.
- $A_3 = \{i \mid i = \xi\}$, kde ξ je vyčislitelná funkce.
- $A_4 = \{i \mid a \in \Re(\varphi_i)\}$.
- $A_5 = \{i \mid \Re(\varphi_i) = \emptyset\}$.
- A_6 je množina všech i takových, že $\Re(\varphi_i)$ je konečná.
- $A_7 = \{i \mid a \in \Im(\varphi_i)\}$.
- A_8 je množina všech i takových, že $\Im(\varphi_i)$ je konečná.
- $A_9 = \{i \mid \Im(\varphi_i) = \mathbf{N}\}$.
- A_{10} je množina všech i takových, že φ_i je prosté.

- A_{11} je množina všech i takových, že φ_i je bijekce.

Nyní si rozšíříme pojem respektování funkcí:

Definice 6.5
Množina $J \subseteq \mathbf{N}^n$ respektuje funkce pokud platí: $(a_1, \ldots, a_n) \in J$, $\varphi_{a_1} = \varphi_{b_1}, \ldots, \varphi_{a_n} = \varphi_{b_n} \Longrightarrow (b_1, \ldots, b_n) \in J$.

Věta 6.6
Množina $J \subseteq \mathbf{N}^n$ respektuje funkce. Pak J je rekurzivní $\Longleftrightarrow J = \emptyset \vee J = \mathbf{N}^n$.

Důkaz: Pouze technicky se liší od důkazu věty 6.4. □

Můžeme tedy pokračovat ve vyjmenovávání typických nerekurzivních množinách:

Příklad 6.2

- $A_{12} = \{(i,j) \mid \varphi_i = \varphi_j\}$.
- $A_{13} = \{(i,j) \mid \forall a \in (\Re(\varphi_i) \cap \Re(\varphi_j)) : \varphi_i(a) < \varphi_j(a)\}$.
- $A_{14} = \{(i,j,k) \mid \varphi_i = \varphi_j \circ \varphi_k\}$.

Věta 6.7
Nechť $I \subseteq \mathbf{N}$ respektuje funkce. Nechť existuje vyčislitelná funkce Θ taková, že:

- $\{i \mid \varphi_i = \Theta\} \subseteq I$
- existuje vyčislitelná funkce $\hat{\Theta}$ tak, že $\Theta \leq \hat{\Theta}$[5] a $\{i \mid \varphi_i = \hat{\Theta}\} \subseteq \bar{I}$.

Pak množina I není rekurzivně vyčislitelná.

Důkaz: Nechť funkce $\xi : \mathbf{N}^2 \longrightarrow \mathbf{N}$ taková, že:

$$\xi(i,j) = \begin{cases} \Theta(j) & \text{if } \varphi_i(i) = \perp \\ \hat{\Theta}(j) & \text{else} \end{cases}$$

Funkce ξ je tedy vyčislitelná. Podle věty o parametrizaci 2.6 existuje totálně vyčislitelná funkce f: $\xi(i,j) = \varphi_{f(i)}(j)$. Tedy $f(i) \in I \Longleftrightarrow \varphi_i(i)$ není definovaná. Tedy $\bar{K} = f^{-1}(I)$. Pokud by tedy byla množina I rekurzivně vyčislitelná, musela by být K také RE-množina, což je samozřejmě spor. Tedy I není RE-množina, čímž končí důkaz. □

Příklad 6.3
Následující množiny nejsou rekurzivně vyčislitelné: $\bar{A_1}, \bar{A_2}, A_3, \bar{A_3}, \bar{A_4}, A_5, A_6, \bar{A_7}, A_8, \bar{A_9}, A_{10}, \bar{A_{11}}$.

Věta 6.8
Nechť $I \subseteq \mathbf{N}$ a nechť existuje vyčislitelná funkce Θ taková, že:

1. $\{i \mid \varphi_i = \Theta\} \subseteq I$.
2. Pro množiny Y všech i takových, že $\varphi_i \leq \Theta$ a přitom $\Re(\varphi_i)$ je konečná, platí $Y \subseteq \bar{I}$.

Pak I není RE-množina.

Důkaz: Mějme funkci $\mu : \mathbf{N}^2 \longrightarrow \mathbf{N}$, kdy $\mu(i,j)$ je rovno $\Theta(j)$, pokud program P_i nezastaví pro i během j kroků, a $\mu(i,j)$ je nedefinováno v opačném případě.

μ je vyčislitelná funkce a tedy existuje e: $\mu = \varphi_e$. Dle věty o parametrizaci 2.6 existuje totální vyčislitelná funkce f taková, že: $\mu(i,j) = \varphi_{f(i)}(j)$.

Zřejmě platí: $i \in K \Longleftrightarrow P_i$ zastaví pro $i \Longleftrightarrow P_i$ zastaví pro i během j kroků pro nějaké $j \Longleftrightarrow \varphi_{f(i)}(x) = \Theta$ pro $x < j$. Dále $\varphi_{f(i)}(x) = \perp$ pro $x \geq j \Longleftrightarrow \varphi_{f(i)} \leq \Theta$ a $\Re(\varphi_{f(i)})$ je konečná množina. Tedy $f(i) \in \bar{I}$.

$i \in \bar{K} \Longleftrightarrow P_i$ nezastaví pro $i \Longleftrightarrow \varphi_{f(i)} = \Theta$ a tedy $f(i) \in I$.

Jelikož $i \in \bar{K} \Longleftrightarrow f(i) \in I$, I není RE-množina. □

Příklad 6.4
Následující množiny nejsou rekurzivně vyčislitelné: $A_1, A_2, \bar{A_6}, \bar{A_8}, A_9, A_{11}$. Následující množiny jsou RE-množiny: $A_4, \bar{A_5}, A_7, \bar{A_{10}}$.

[5] Relaci $\leq$ aplikované na dvojici funkcí budeme rozumět jako *rozšíření funkcí.*

6.2 Redukovatelné množiny

Definice 6.9

Nechť f je totální vyčislitelná funkce; K a I nechť jsou množiny. Pokud platí ekvivalence: $i \in K \iff f(i) \in I$ řekneme, že množina K je *redukovatelná* na množinu I přes funkci f. Značíme $K \leq_m I$. Zřejmě relace $\leq_m$ je reflexivní a tranzitivní.

Lemma 6.10

Je-li A rekurzivně vyčislitelná, pak $A \leq_m K$.

Důkaz: Mějme funkci $\Theta : \mathbf{N}^2 \longrightarrow \mathbf{N}$:

$$\Theta(x,y) = \begin{cases} 1 & x \in A \\ \perp & x \notin A \end{cases}$$

Funkce Θ je vyčislitelná, tedy $\Theta = \varphi_e^{(2)}$ pro nějaké e. Podle věty o parametrizaci 2.6 existuje totálně vyčislitelná funkce $s_1^1 : \varphi_e^{(2)}(x,y) = \varphi_{s_1^1(e,x)}(y) = \varphi_{f(x)}(y)$. Tedy $\varphi_{f(x)}(f(x))$ je definováno $\iff x \in A$, což znamená, že $x \in A \iff f(x) \in K$. □

Předchozí lemma nám vlastně ukázalo, že problém zastavení (příslušnosti do množiny K) je alespoň tak těžký jako jakýkoli jiný rekurzivně vyčislitelný problém. Protože však K je rekurzivně vyčislitelná množina, je nejtěžší RE-množin. Říkáme, že K je *úplná* množina ve třídě RE-množin.

Lemma 6.11

Platí-li pro množinu A vztah $\bar{K} \leq_m A$, pak A není RE-množina.

Důkaz: $\bar{K} = f^{-1}(A)$. □

6.3 Produktivní a kreativní množiny

Definice 6.12

Množina A je *produktivní*, jestliže existuje vyčislitelná funkce Θ taková, že $\forall i : \; W_i \subseteq A \implies \Theta(i) \neq \perp$ a $\Theta(i) \in A - W_i$.

Takovou funkci Θ pak nazýváme *produktivní* funkce.

Definice 6.13

Množina B se nazývá *kreativní*, je-li rekurzivně vyčislitelná a zároveň $\bar{B}$ je produktivní. Funkce Θ se nazývá *produktivní funkce pro kreativní množinu* B.

Produktivní množina nemůže být RE-množina. Např. $\bar{K}$ je produktivní množina, její produktivní funkce $\Theta = \mathrm{id}$. Množina K je kreativní. Obecně však nemusí platit, že produktivní množiny mají kreativní doplňky (např. $\bar{A}_1$).

Lemma 6.14

Každá produktivní množina obsahuje nekonečnou RE-podmnožinu.

Důkaz: Nechť A je produktivní s produktivní funkcí Θ. Nechť i_0 je index prázdné množiny ($W_{i_0} = \emptyset$). $W_{i_0} \subseteq A$ a $\Theta(i_0) \in A - W_{i_0} = A$. Dále $W_{i_0} \neq W_{i_1} = \{\Theta(i_0)\} \subseteq A$ a $\Theta(i_1) \in A = W_{i_1}$. Tímto způsobem můžeme vygenerovat nekonečnou RE-množinu, aniž bychom vybočili ven z množiny A. □

Věta 6.15

Je-li A produktivní množina a $A \leq_m B$, pak i B je produktivní množina.

Důkaz: Mějme funkci f: $x \in A \iff f(x) \in B$. Dále nechť Θ je produktivní funkce množiny A. Platí, že $A = f^{-1}(B)$. Pro všechna i platí:

$$W_i \subseteq B \implies f^{-1}(W_i) \subseteq A$$

Existuje totální vyčislitelná funkce g taková, že $\forall i : \; f^{-1}(W_i) = W_{g(i)}$. Pak:

$$W_i \subseteq B \implies W_{g(i)} \subseteq A \quad \implies \quad (\Theta \circ g)(i) \in A - W_{g(i)} \tag{1}$$

$$(\Theta \circ g)(i) \notin W_{g(i)} \quad \iff \quad (f \circ \Theta \circ g)(i) \notin W_i \tag{2}$$

$$(\Theta \circ g)(i) \in A \quad \iff \quad (f \circ \Theta \circ g)(i) \in B \tag{3}$$

Tedy platí: $W_i \subseteq B \implies (f \circ \Theta \circ g)(i) \in B - W_i$. □

Lemma 6.16
Nechť A je kreativní a B je RE-množina. Pak platí implikace: $A \leq_m B \Longrightarrow B$ je kreativní.

Lemma 6.17
Je-li množina A produktivní, pak existuje totální vyčislitelná funkce $p : \mathbf{N} \longrightarrow \mathbf{N}$ taková, že A je produktivní s funkcí p.

Důkaz: Nechť Θ je produktivní funkce pro A. Existuje e taková, že $\Theta = \varphi_e$.
Mějme totální vyčislitelnou funkci $g : \mathbf{N}^2 \longrightarrow \mathbf{N}$ takovou, že program $P_{g(i,j)}$ je:

```
begin X2 := Φ(e,j); X2 := Φ(i,X1); end
```

Pak množina:

$$W_{g(i,j)} = \begin{cases} W_i & \text{if } \Theta(j) \neq \perp \\ \emptyset & \text{else} \end{cases}$$

Podle věty o rekurzi existuje totální vyčislitelná funkce f taková, že pro $\forall i : W_{f(i)} = W_{g(i,f(i))}$. Potom tedy:

$$W_{f(i)} = W_{g(i,f(i))} = \begin{cases} W_i & \text{if } (\Theta \circ f)(i) \neq \perp \\ \emptyset & \text{else} \end{cases}$$

Předpokládejme nyní, že $(\Theta \circ f)(i) = \perp$. Pak ovšem $W_{f(i)} = \emptyset \subset A$, přičemž A je produktivní, a $(\Theta \circ f)(i) \neq \perp$, což je spor. $(\Theta \circ f)(i)$ je tedy definováno pro $\forall i$ taková, že $W_{f(i)} = W_i$.
Tedy: $W_i \subset A \Longrightarrow W_{f(i)} \subset A \Longrightarrow (\Theta \circ f)(i) \in A - W_i$, tedy $\Theta \circ f$ je produktivní funkce pro A a my už jen položíme $p = \Theta \circ f$. □

Věta 6.18
Množina B je kreativní $\Longleftrightarrow$ B je RE-množina a pro libovolnou RE-množinu A platí $A \leq_m B$.

Důkaz:

$\Longleftarrow$: B je RE-množina a pro $\forall A$ platí $A \leq_m B$, tedy i pro $A = K$ (K je kreativní). Tedy i množina B je kreativní.

$\Longrightarrow$: Nechť máme kreativní množinu B. My ukážeme, že $K \leq_m B$, z čehož bude plynout, že libovolná množina $A \leq_m B$.

Máme B kreativní a tedy množina $\bar{B}$ je produktivní s totální funkcí, kterou si označíme p. Nyní definujme $g : \mathbf{N}^2 \longrightarrow \mathbf{N}$ tak, že program $P_{g(i,j)}$ vypadá takto:

```
begin X2 := Φ(i,i); X2 := Φ(e,j); while X1<>X2 do begin end; end
```

přičemž e je index funkce p ve standardní numeraci. Tedy:

$$W_{g(i,j)} = \begin{cases} \{p(j)\} & \text{if } i \in K \\ \emptyset & \text{else} \end{cases}$$

Podle věty o rekurzi existuje totální vyčislitelná funkce $f : \mathbf{N} \longrightarrow \mathbf{N}$ taková, že $\forall i$ platí:

$$W_{f(i)} = W_{g(i,j)} = \begin{cases} \{(p \circ f)(j)\} & \text{if } i \in K \\ \emptyset & \text{else} \end{cases}$$

Tedy platí:

- $i \in K \Longrightarrow (p \circ f)(i) \in W_{f(i)} \Longrightarrow W_{f(i)} \not\subset \bar{B} \Longrightarrow (p \circ f)(i) \in B$.
- $i \notin K \Longrightarrow W_{f(i)} = \emptyset \Longrightarrow W_{f(i)} \subset \bar{B} \Longrightarrow (p \circ f)(i) \in \bar{B}$.

a celkem tedy $i \in K \Longleftrightarrow (p \circ f)(i) \in B$.

□

Důsledek 6.19

1. K je kreativní, úplná množina. $\bar{K}$ je nejmenší produktivní množina.
2. $I \subset \mathbf{N}$ respektuje funkce, I je RE-množina, $\emptyset \neq I \neq \mathbf{N}$, pak I je kreativní. Všechny nerekurzivní RE-množiny, o kterých jsme mluvili[6], jsou kreativní.
3. Všechny množiny, o kterých jsme pomocí Riceovy věty dokázali, že nejsou RE-množiny, jsou produktivní.

6.4 Imunní a jednoduché množiny

Definice 6.20
Množina A se nazývá *imunní*, je-li nekonečná a neobsahuje-li RE-podmnožinu.
Množina A se nazývá *jednoduchá*, je-li RE-množina a její doplněk $\bar{A}$ je imunní.

Poznámka 6.21
Imunní množina, pokud existuje, není RE-množina ani produktivní množina.

Věta 6.22
Existuje jednoduchá množina.

Důkaz: Nechť B je RE-množina. Máme-li dokázat, že je jednoduchá, stačí ukázat, že:

1. B i $\bar{B}$ jsou nekonečné.
2. Každá nekonečná RE-množina má neprázdný průnik s množinou B.

Uvažujme proceduru pro numeraci RE-množin. Pro $\forall n$ položíme do množiny B první nalezený prvek z množiny W_n, který je větší než $2n$ (pokud existuje). Tedy množina B je efektivně generována. Přitom pro $\forall n$ může být v B nejvíce n čísel, která leží mimo interval $0, 1, \ldots, 2n$. Tedy B i $\bar{B}$ jsou nekonečné. Je-li W_n nekonečná RE-množina $\Longrightarrow$ existuje číslo $x > 2n$ takové, že $x \in W_n$. Tedy $B \cap W_n \neq \emptyset$. □

Důsledek 6.23
Existuje imunní množina.

6.5 Klasifikace problémů

1. Existují produktivní množiny, které mají produktivní doplněk (např. problém totálnosti).
2. Existují produktivní množiny s rekurzivně vyčislitelnými doplňky (např. problém nezastavení).
3. Existují kreativní množiny (nejtěžší mezi RE-množinami, např. problém zastavení).
4. Existují jednoduché množiny (tj. středně těžké mezi RE-množinami).

[6] Zde máme na mysli známe množiny A_i.

7 Relativizovaná teorie vyčislitelnosti

Definice 7.1 PROGRAM S ORÁKULEM

Nechť $B \subseteq \mathbf{N}$. *Program s orákulem* B je program, který může používat i příkazy `while` $x \in B$ `do` δ případně `while` $x \notin B$ `do` δ a B nemusí být rekurzivně vyčislitelná.

Program P s orákulem B vyčisluje funkci $\varphi_P^B : \mathbf{N}^k \longrightarrow \mathbf{N}$.

Definice 7.2

Mějme funkci $f : \mathbf{N}^k \longrightarrow \mathbf{N}$, $A \subseteq \mathbf{N}$. Řekneme, že f je *A-vyčislitelná*, jestliže existuje program P s orákulem A takový, že $f = \varphi_P^A$.

Věta 7.3 VĚTA O NUMERACI

Univerzální funkce $\Phi : \mathbf{N}^{j+1} \longrightarrow \mathbf{N}$ definovaná vztahem:

$$\Phi(e, a_1, \ldots, a_j) = \varphi_e^A(a_1, \ldots, a_j)$$

je A-vyčislitelná.

Věta 7.4 VĚTA O PARAMETRIZACI

Existuje totální vyčislitelná funkce $s_n^m : \mathbf{N}^{m+1} \longrightarrow \mathbf{N}$ taková, že:

$$\varphi_{s_n^m(i,g_1,\ldots,g_m)}^{A(n)}(z_1, \ldots, z_n) = \varphi_i^{A(n)}(g_1, \ldots, g_m, z_1, \ldots, z_n)$$

Definice 7.5

Množina $B \subseteq \mathbf{N}^k$ je *A-rekurzivní*, jestliže je její charakteristická funkce totální A-vyčislitelná.

Množina B je *A-rekurzivně vyčislitelná (A-RE-množina)*, jestliže existuje A-vyčislitelná funkce f taková, že $B = \Re(f)$.

V relativizované teorii vyčislitelnosti můžeme tedy definovat numerace:

- A-vyčislitelných funkcí: $\varphi_0^A, \varphi_1^A, \varphi_2^A, \ldots$
- A-RE-množin: $W_0^A, W_1^A, W_2^A, \ldots$ podle numerace A-vyčislitelných funkcí: $\Re(\varphi_0^A), \Re(\varphi_1^A), \Re(\varphi_2^A), \ldots$

Dále relativizovaný problém zastavení definujeme: $K^A = \{i \mid i \in W_i^A\}$.

Definice 7.6

Nechť $A, B \subseteq \mathbf{N}$. Řekneme, že *$A$ se T-redukuje na B* (píšeme $A \leq_T B$), jestliže A je B-rekurzivní množina.

$A \equiv_T B$: $(A \leq_T B) \wedge (B \leq_T A)$. Třídy ekvivalence $\equiv_T$ se nazývají *třídy Turingova stupně nerozhodnutelnosti*.

Přitom platí implikace $A \leq_m B \Longrightarrow A \leq_T B$. Opačná implikace neplatí. Např. $\bar{K} \not\leq_m K$, ale $\bar{K} \leq_T K$.

Označení: Zaveďme si označení:

$$A' = K^A = \{i \mid i \in W_i^A\}$$

$\emptyset_{\mathbf{N}} \in \mathbf{O}$, kde $\mathbf{O}$ jsou nejjednodušší problémy, tzv. stupně 0. Dále máme $\emptyset' = K \in \mathbf{O}'$ ($\emptyset' = \bar{K} \in \mathbf{O}'$). Ještě dále je $K' \in \mathbf{O}''$. K' je problém totálnosti (problém netotálnosti).

Věta 7.7 (VĚTA O REKURZI)

Nechť $f : \mathbf{N} \longrightarrow \mathbf{N}$ je totální vyčislitelná funkce. Pro libovolné j existuje n tak, že programy P_n a $P_{f(n)}$ počítají stejnou funkci (tj. $\varphi_n^{(j)} = \varphi_{f(n)}^{(j)}$).

Důkaz: Nechť funkce $g : \mathbf{N} \longrightarrow \mathbf{N}$ je taková, že program $P_{g(i)}$ je:

```
begin X2 := Φ(i,i); X1 := Φ(X2,X1); end
```

tedy první přiřazení spočítá $\varphi_i(i)$ a druhé přiřazení $\varphi_{\varphi_i(i)}(j)$.

g je zřejmě totální vyčislitelná funkce,

$$\varphi_{g(i)}(j) = \begin{cases} \varphi_{\varphi_i(i)}(j) & \text{if } \varphi_i(i) \text{ defined} \\ \perp & \text{else} \end{cases}$$

Pak $f \circ g = \varphi_m$ je totální vyčislitelná funkce a $\varphi_m(m)$ je definováno. Potom:

$$\varphi_{g(m)}(j) = \varphi_{\varphi_m(m)}(j) = \varphi_{f(g(m))}(j) = \varphi_{f(n)}(j)$$

kde $n = g(m)$. □

Věta 7.8
Je-li $f : \mathbf{N} \longrightarrow \mathbf{N}$ totální vyčislitelná funkce, pak existuje nekonečně mnoho n takových, že $\varphi_n^{(j)} = \varphi_{f(n)}^{(j)}$.

Důkaz: Nechť program L je takový, že $\varphi_l \neq \varphi_0, \ldots, \varphi_l \neq \varphi_k$, kde $\varphi_0, \ldots, \varphi_k$ je $k+1$ vyčislitelných funkcí, kde k je libovolné. Definujme totální vyčislitelnou funkci $g : \mathbf{N} \longrightarrow \mathbf{N}$ takto:

$$g(x) = \begin{cases} l & x \leq k \\ f(x) & x > k \end{cases}$$

g je samozřejmě vyčislitelná. n je pevný bod, je-li $n \leq k \Longrightarrow \varphi_{g(n)} = \varphi_l \neq \varphi_n$, což je spor, tedy $n > k$ a tím pádem $f(n) = g(n)$ a tedy n je pevný bod pro f a n může být libovolně velké. □