Kapitola 4

Univerzální Turingův stroj a Nedeterministický Turingův stroj

4.1 Nedeterministický TS

Obdobně jako u konečných automatů zavedeme nedeterminismus.

Definice 14. *Nedeterministický Turingův stroj* (NTS) je dán stejnými složkami jako v definici 1 až bod

4. přechodová funkce $\delta : Q \times \Gamma \to 2^{Q \times \Gamma \times \{L,R\}}$.

Pojem konfigurace je u NTS zaveden stejně jako u deterministických TS. Malý rozdíl je v definici výpočtu.

Definice 15. Krok výpočtu NTS je definován jako binární relace na množině konfigurací: Nechť $(q, a_1 \dots a_n, i)$ je taková konfigurace T, kde $q \neq q_{\pm}, n \in \mathbb{N}, a_1, \dots, a_n \in \Gamma, i \leq n$.

(a) Je-li $1 \leq i \leq n$ a $\delta(q, a_i) \ni (q', b, \mathrm{L})$, pak

$$(q, a_1 \dots a_n, i) \vdash (q', a_1 \dots a_{i-1} b a_{i+1} \dots a_n, i-1).$$

(b) Je-li $\delta(q, a_1) \ni (q', b, \mathrm{L})$, pak

$$(q, a_1 \dots a_n, 0) \vdash (q', b a_2 \dots a_n, 0).$$

(c) Je-li $\delta(q, a_i) \ni (q', b, \mathrm{R})$, pak

$$(q, a_1 \dots a_n, i) \vdash (q', a_1 \dots a_{i-1} b a_{i+1} \dots a_n, i+1).$$

Definice kroku výpočtu NTS je tedy velmi podobná definici 15 kroku vypočtu TS. Podoné jsou i pojmy, které na tento pojem kroku výpočtu navazuji: *Výpočet* NTS definujeme jako tranzitivní, refexivní uzávěr relace $\vdash$. *Větví výpočtu* nazýváme sekvenci $c_1 \vdash c_2 \vdash \dots \vdash c_k$ takový, že platí, že $c_i \vdash c_{i+1}$ pro všechna $1 \leq i < k$. Bývá zvykem ztotožňovat pojmy výpočet a větev výpočtu. Větev výpočtu se nazývá *přijímající*, pokud je poslední konfigurace v zápise přijímající, a je zamítající pokud je poslední konfigurace v zápise *přijímající*.

NTS přijímá slovo w, pokud existuje alespoň jedna větev výpočtu nad w, která je přijímající. NTS zamítá slovo w, pokud všechny větve výpočtu nad tímto slovem konečné a žádná není přijímající.

Stejně jako u konečných automatů platí, že se zavedením nedeterminismu nezvýší výpočetní síla.

Věta 4. *Třída nedeterministických Turingových strojů je ekvivalentní s třídou deterministických Turingových strojů.*

4.2 Univerzální Turingův stroj

Pojmem *univerzální Turingův stroj* myslíme TS, který dokáže simulovat činnost jiného TS (například TS, jehož zakódování je součástí vstupního slova univerzálního TS). V této části textu si ukážeme, jak univerzální TS zkonstruovat.

Konstruujme TS U, který přijímá vstupní slova ve tvaru $[M, w]$, kde TS M přijímá w.

Univerzální TS U sestavíme jako třípáskový TS: Na první pásce bude zapsán kód $[M]$; na druhé pásce budeme uchovávat konfiguraci c TS M, na třetí pásce bude následná konfigurace TS M.

TS U pro vstupní slovo $[M, w]$, kde M je TS a w slovo nad vstupní abecedou M

1. Zapíše na druhou pásku kód $[c_0]$ iniciální konfigurace TS M (na první nechá $[M]$)
2. Pokud je konfigurace c, jejíž kód je na první pásce, přijímající, TS U přijme $[M, w]$; pokud je zamítající, U zamítne $[M, w]$; jinak pokračuje dalším bodem.
3. Najdi v $[M]$ přechod, který lze použít z c.
4. Vypočte novou konfiguraci c', t.ž. $c \vdash c'$, zapíše ji na druhou pásku. Pokračuje krokem 2.

Dále ukážeme, že můžeme sestrojit i (deterministický) TS U_N, který simuluje činnost nedeterministického TS.

TS U_N pro vstupní slovo $[M, w]$, kde M je NTS a w slovo nad vstupní abecedou M

1. Zapíše na druhou pásku kód $[c_0]$ iniciální konfigurace TS M (na první nechá $[M]$)
2. Pokud je některá konfigurace c, jejíž kód je na první pásce, přijímající, TS U_N přijme $[M, w]$; pokud je zamítající, U_N zamítne $[M, w]$; jinak pokračuje dalším bodem.
3. Pro každou konfiguraci c najdi v $[M]$ přechody, které lze použít z c.
4. Vypočte všechny nové konfigurace c', t.ž. $c \vdash c'$, zapíše je na druhou pásku. Pokračuje krokem 2.

Z faktu, že deterministický TS dokáže simulovat nedeterministický TS, vyplývá, že deterministický TS je výpočetně alespoň tak silný, jako nedeterministický. Protože deterministický TS je speciální případ nedeterministického TS, dostáváme že třída deterministických TS a třída nedeterministických TS jsou ekvivalentní.

4.3 Enumerator

Enumerator je varianta TS, která nemá žádný vstup, ale má navíc instrukci `print`, která, když je vyvolána vytiskne obsah pásky (až po první prázdný symbol). Jazykem enumeratoru rozumíme množinu všech slov, které vytiskne. Pro ilustraci viz obr. 4.3.

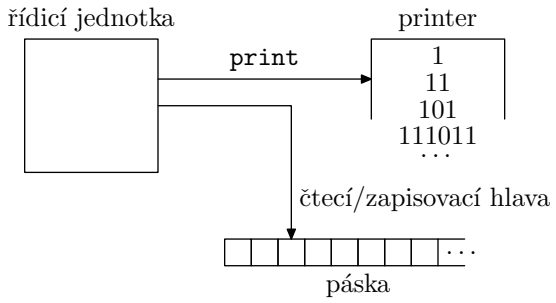
Enumerator můžeme formalizovat třeba tak, že přechodová funkce bude mít tvar

$$\delta : Q \times \Gamma \to Q \times \Gamma \times \{R, L\} \times \{\texttt{print}, \emptyset\}$$

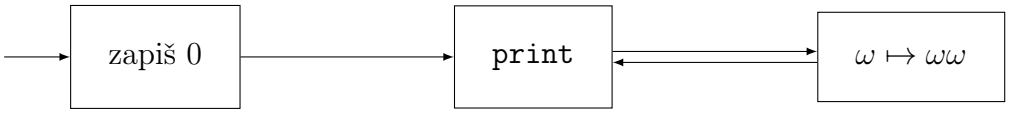
nebo tak, že určíme množinu stavů, do kterých, když enumerator přejde, nastane `print`.

Příklad 8. Pomocí stroje, který vyčísluje funkci $\omega \mapsto \omega\omega$ můžeme sestavit enumerator, který tiskne jazyk $\{0^{2^n} \mid n \in \mathbb{N}_0\}$. Viz obr. 4.2

Konkrétní formalizace nás zajímat nemusí. Důležitější pro nás bude fakt, že enumerator je výpočetně stejně silný jako klasický TS:



Obrázek 4.1: Schema enumeratoru



Obrázek 4.2: Enumerator tisknoucí jazyk $\{0^{2^n} \mid n \in \mathbb{N}_0\}$.

Věta 5. *Nechť $L \subseteq \Sigma^*$ je jazyk.*
(a) Jazyk L je rekurzivní, právě když existuje enumerator, který jej tiskne lexikografickém uspořádání.
(b) Jazyk L je částečně rekurzivní, právě když existuje enumerator, který jej tiskne.

Důkaz. (a) „⇒“: Nechť L je rekurzivní jazyk a M_L je TS, který rozhoduje L. Sestrojíme enumerátor E_L, který jej tiskne L v lexikografickém pořadí:

Enumerator E_L:

1. (na pásce je zapsáno $w = \varepsilon$)
2. simuluje činnost M_L pro w
3. pokud M_L přijme w, pak `print`
4. vypočte lexikografického následníka w, zapiš ho na pásku a pokračuje od bodu 2.

„⇐“: Nechť L je tisknut enumeratorem E_L v lexikografickém pořadí. Pokud je L konečný jazyk, pak ji jistě rekurzivní. Uvažujme tedy pouze případ, že L je nekonečný. Sestavíme TS M_L, který rozhoduje L:

TS M_L pro vstupní slovo w:

1. simuluje činnost E_L
2. pokaždé když E_L vytiskne slovo v, M_L ho porovná s w:
   - pokud $w = v$, M_L přijímá w,
   - pokud $w > v$, M_L zamítá w,
   - jinak pokračuje v simulaci E_L.

(b) „⇒“ Nechť L je částečně rekurzivní jazyk a M_L je TS, který ho přijímá. Sestrojíme E_L enumerator, který jej tiskne:

Enumerator E_L

1. Nastaví n na 0

2. Vygeneruje všechna slova délky maximálně n a pro každé z nich simuluje n kroků výpočtu TS M_L. Pokud některá simulace skončila přijetím, E_L vytiskne přijaté slovo.
3. Zvýší n o 1 a pokračuje krokem 2.

Zjevně právě slova jazyka L budou vytištěna enumeratorem E_L.

„$\Leftarrow$“ Nechť L je tisknut enumeratorem E_L v lexikografickém pořadí. Pokud je L konečný jazyk, pak ji jistě rekurzivní. Uvažujme tedy pouze případ, že L je nekonečný. Sestavíme TS M_L, který rozhoduje L:

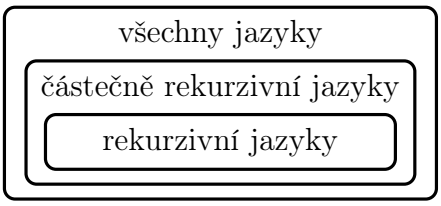
TS M_L pro vstupní slovo w:

1. simuluje činnost E_L
2. pokaždé když E_L vytiskne slovo v, M_L ho porovná s w:
   - pokud $w = v$, M_L přijímá w,
   - jinak pokračuje v simulaci E_L.

□

Kapitola 5

Jazyky Turingových strojů



Obrázek 5.1: Množiny jazyků TS

V kapitole 2 jsme definovali jazyky TS. Jazyky TS nám logicky rozdělily množinu všech jazyků na tři podmnožiny: rekurzivní jazyky, částečně rekurzivní jazyky, které nejsou rekurzivní, a jazyky, které nejsou ani částečně rekurzivní (viz Obr. 5.1). Zatím jsme si představili jen několik zástupců jazyků, které jsou rekurzivní. V této kapitole si představíme jazyky ze zbývajících dvou množin jazyků a budeme se zabývat vlastnostmi množin jazyků.

5.1 Jazyky, které nejsou částečně rekurzivní

Diagonální jazyk L_D je definován následovně

$$L_D = \{[M] \mid M \text{ je TS, který nepřijímá } [M]\}$$

Slovy, L_D je jazyk Turingových strojů, které nepřijímají vlastní kód.

Věta 6. *L_D není částečně rekurzivní.*

Důkaz. Tvrzení dokážeme sporem. Předpokládejme, že existuje TS D, který přijímá L_D. Uvažujme, jestli $[D] \in L_D$:

- Pokud $[D] \in L_D$, takže TS D přijímá $[D]$ (protože TS D podle předpokladu přijímá L_D), takže $[D] \notin L_D$ (dle definice jazyka L_D)
- Pokud $[D] \notin L_D$, takže TS D nepřijímá $[D]$, takže $[D] \in L_D$.

V obou větvích docházíme ke sporu, předpoklad je tedy mylný. □

Následující věta říká, že jazyk kódů TS, které přijímají prázdný jazyk, není částečně rekurzivní.

Věta 7. *Jazyk $L_\emptyset = \{[M] \mid M$ je TS, t.ž. $L(M) = \emptyset\}$ není částečně rekurzivní.*

Důkaz. TODO □

5.2 Jazyky, které jsou částečné rekurzivní, ale nejsou rekurzivní

Víme tedy, že existují jazyky, které jsou rekurzivní a víme také, že existují jazyky, nejsou částečně rekurzivní (L_D). V této sekci si představíme dva modelové zástupce jazyků, které jsou částečné rekurzivní, ale nejsou rekurzivní – konkrétně univerzální jazyk, a jazyk HALT[1].

Univerzální jazyk L_U je definován takto:

$$L_U = \{[M, w] \mid M \text{ je TS, který přijímá } w\}.$$

Věta 8. *L_U není rekurzivní.*

Důkaz. Toto tvrzení dokážeme opět sporem. Budeme předpokládat že L_U je rekurzivní, dojdeme ale k závěru, že v tom případě je i L_D rekurzivní.

Předpokládejme, že jazyk L_U je rekurzivní, a TS U' je TS, který ho rozhoduje. Sestavíme TS D:
TS D pro vstupní slovo $[M]$, kde M je TS:

1. Simuluje činnost TS U' pro vstupní slovo $[M, [M]]$.
2. Pokud tato simulace skončí přijetím, D *zamítá* $[M]$.
3. Pokud tato simulace skončí zamítnutím, D *přijímá* $[M]$.

O TS D víme, že:

(a) TS D nemůže cyklit: simulace v kroku 1. musí vždy skončit, protože U' necyklí, kroky 2. a 3. jsou jednoduché.

(b) TS D přijímá $[M]$, pokud U' zamítá $[M, [M]]$, což je v případě, že M nepřijímá $[M]$.

Z těchto dvou faktů vyplývá, že TS D rozhoduje jazyk L_D. To je ale spor tvrzením věty 6. Předpoklad tedy musí být mylný. □

Věta 9. *Jazyk L_U je částečně rekurzivní.*

Důkaz. Univerzální TS jej přijímá. □

Jazyk L_{HALT} je definován takto:

$$L_{\text{HALT}} = \{[M, w] \mid M \text{ je TS, který zastaví pro } w\}.$$

Věta 10. *Jazyk L_{HALT} není rekurzivní.*

Důkaz. Tvrzení opět dokážeme sporem. Budeme předpokládat, že L_{HALT} je rekurzivní, a ukážeme, že v tom případě by byl i L_U rekurzivní.

Předpokládejme, že L_{HALT} je rekurzivní, a že tedy existuje TS H, který jej rozhoduje. Sestavíme TS U':

TS U' pro vstupní slovo $[M, w]$, kde M je TS, a w je řetěz nad jeho abecedou

1. Vytvoří TS M' upravou kódu TS M tak, aby M' cyklil, kdykoli TS M zamítá (jinak se chová stejně).
2. Simuluje činnost TS H pro vstupní slovo $[M', w]$. Pokud simulace skončí přijetím TS U' *přijímá* $[M, w]$, v opačném případě TS U' *zamítá* $[M, w]$.

O TS D víme, že:

[1] V kontextu rozhodovacích problémů se tyto nazývají *problém přijetí TS* a *problém zastavení TS*

(a) TS U' nemůže cyklit: krok 1 je jednoduchá úprava, simulace v kroku 2. musí vždy skončit, protože H necyklí.

(b) TS U' přijímá $[M, w]$ právě když H přijímá $[M', w]$, což je právě v případě, že M' zastaví pro w a to je právě v případě, že M přijme w

Z těchto dvou faktů vyplývá, že U' rozhoduje L_U, což je spor s tvrzením Věty 8. Předpoklad je tedy mylný. □

Věta 11. *Jazyk L_{HALT} je částečně rekurzivní.*

Důkaz. Sestrojíme TS $T_{L_{\text{HALT}}}$, který přijímá L_{HALT}.

```
* DODELAT
```

□

5.3 Jazyky, které jsou rekurzivní

V předcházejících kapitolách jsme si ukázali několik jazyků, které jsou rozhodovány TS. V této sekci si ukážeme některé další, zajímavější.

Jazyk L_{HALTLBA} je definován takto.

$$L_{\text{HALTLBA}} = \{[M, w] \mid M \text{ je LBA, který zastaví pro } w\}.$$

Věta 12. *Jazyk L_{HALTLBA} je rekurzivní.*

Důkaz. Náznak: stačí sestrojit TS, který simuluje činnost LBA a navíc počítá simulované kroky. Protože LBA má pro použití konečný pevně daný počet políček pásky, které může pro zpracování w použít, existuje konečný počet konfigurací, ve kterých se může nacházet. Pokud je výpočet delší, než tento počet, LBA cyklí (musel být v nějaké konfiguraci alespoň dvakrát). TS, který simuluje činnost LBA může tedy snadno detekovat zacyklení. □

Jako důsledek této věty dostáváme následující větu:

Věta 13. *Každý jazyk přijímaný LBA je rekurzivní.*

Důkaz. Nechť X je TS, t.ž. $L(X) = L_{\text{HALTLBA}}$ a L je jazyk přijímaný LBA A. Sestrojíme následující TS:

TS M_{LBA} pro vstupní slovo w:

1. spustí TS X pro vstupní slovo $[A, w]$, pokud X zamítne $[A, w]$, M_{LBA} zamítne w.
2. simuluje činnost A pro w; pokud A přijme w, pak M_{LBA} přijme w, jinak M_{LBA} zamítne w.

□

5.4 Vlastnosti jazyků TS

Věta 14. *Je-li L jazyk přijímaný TS, a jeho doplněk $\overline{L}$ také jazyk přijímaný TS, pak L je jazyk rozhodovaný TS.*

Důkaz. Tvrzení dokážeme konstruktivním způsobem. Nechť M_L a $M_{\overline{L}}$ jsou TS, které přijímají L a $\overline{L}$. Sestavíme TS M, který rozhoduje L.

TS M pro vstupní slovo w:

1. Střídavě, po jednom kroku, simuluje činnost strojů M_L a $M_{\overline{L}}$ pro w.
2. Pokud M_L přijme w, pak M přijme w; Pokud $M_{\overline{L}}$ přijme w, pak M zamítne w.

Každé slovo w patří buďto do L nebo do $\overline{L}$. To znamená, že během konečného počtu kroků M_L nebo $M_{\overline{L}}$ přijme w. TS M se tedy nemůže během simulace zacyklit. TS M tedy necyklí a rozhoduje L. □

Poznámka 6. Věta 14 by se dala doplnit takto: Je-li L jazyk přijímaný TS, a $\overline{L}$ také jazyk přijímaný TS, pak L i $\overline{L}$ je jazyk rozhodovaný TS.

Věta 15. *Nechť L_1, L_2 jsou rekurzivní jazyky. Pak:*
(a) průnik $L_1 \cap L_2$ je rekurzivní jazyk,
(b) sjednocení $L_1 \cup L_2$ je rekurzivní jazyk,
(c) konkatenace $L_1L_2 := \{\alpha\beta \mid \alpha \in L_1, \beta \in L_2\}$ je rekurzivní jazyk,
(d) Kleeneho uzávěr $L^* := \{\varepsilon\} \cup \bigcup_{n\in\mathbb{N}} L^n$ je rekurzivní jazyk.
(e) doplněk $\overline{L}$ je rekurzivní jazyk.

Důkaz. (a) Nechť L_1, L_2 jsou rekurzivní jazyky a M_1, M_2 jsou Turingovy stroje, které rozhodují L_1 a L_2, tedy $L(M_1) = L_1$ a $L(M_2) = L_2$. Sestrojíme M tak, že $L(M) = L_1 \cap L_2$:

TS M pro vstupní slovo w:

1. TS M simuluje činnost TS M_1 pro vstupní slovo w.
   (a) Pokud M_1 zamítne w, TS M zamítne slovo w.
   (b) Pokud M_1 přijme w, TS M pokračuje bodem 2.
2. TS M simuluje činnost TS M_1 pro vstupní slovo w.
   (a) Pokud M_2 zamítne w, TS M zamítne slovo w.
   (b) Pokud M_2 přijme w, TS M zamítne slovo w.

TS M nikdy necyklí a přijímá slovo w právě když $w \in L(M_1)$ a $w \in L(M_2)$; rozhoduje tedy jazyk $L_1 \cap L_2$, a tento jazyk je tedy rekurzivní.

(b) podobný důkazu (a).

(c) Nechť L_1, L_2 jsou rekurzivní jazyky a M_1, M_2 jsou Turingovy stroje, které rozhodují L_1 a L_2, tedy $L(M_1) = L_1$ a $L(M_2) = L_2$. Sestrojíme M tak, že $L(M) = L_1L_2$:

TS M pro vstupní slovo $w := a_1a_2 \dots a_n$:

1. TS M pro $k = 0, 1, \dots n$ provádí následující činnost
   (a) rozdělí vstupní slovo w na dva podřetězce $w_1 := a_1 \dots a_k$ a $w_2 := a_{k+1} \dots a_n$.
   (b) simuluje činnost TS M_1 pro slovo w_1 a poté simuluje činnost TS M_2 pro slovo w_2. Pokud obě simulace skončily přijetím, TS M přijme slovo w, v opačném případě pokračuje dalším k.

2. Pokud pro žádné k nenastalo přijetí, TS M zamítá w.

TS M nikdy necyklí a přijímá slovo w právě když se dá rozdělit tak, aby $w_1 \in L(M_1)$ a $w_2 \in L(M_2)$; rozhoduje tedy jazyk L_1L_2, a tento jazyk je tedy rekurzivní.

(d) Nechť L je rekurzivní jazyk a M je TS, který rozhoduje L, tedy $L(M) = L$. Sestrojíme nedeterministický TS M' tak, že $L(M') = L^*$:

NTS M' pro vstupní slovo $w := a_1a_2 \dots a_n$:

1. Pokud $w = \varepsilon$, NTS M' přijímá.
2. NTS M' nedeterministicky zvolí $0 < m < n$ a nedeterministicky zvolí $1 < k_1 < k_2 < \dots < k_m < n$.
   (a) rozdělí vstupní slovo w na m podřetězců $w_i := a_{k_{i-1}} \dots a_{k_i}$.
   (b) simuluje činnost TS M pro slova w_i. Pokud všechny simulace skončily přijetím, TS M' přijme slovo w.
3. Pokud pro alespoň jedno w_i nenastalo přijetí, TS M zamítá w.

(e) Nechť M je TS, který rozhoduje L. TS $\overline{M}$ sestrojíme ze stroje M záměnou přijímacího a zamítacího stavu. TS $\overline{M}$ tedy přijímá slova, která M přijímá a naopak $\overline{M}$ zamítá slova, která M přijímá. TS $\overline{M}$ tedy rozhoduje $\overline{L}$. □

Věta 16. *Nechť L_1, L_2 jsou částečně rekurzivní jazyky. Pak:*
(a) průnik $L_1 \cap L_2$ je částečně rekurzivní jazyk,
(b) sjednocení $L_1 \cup L_2$ je částečně rekurzivní jazyk,
(c) konkatenace $L_1L_2 := \{\alpha\beta \mid \alpha \in L_1, \beta \in L_2\}$ je částečně rekurzivní jazyk,
(d) Kleeneho uzávěr $L^* := \{\varepsilon\} \cup \bigcup_{n\in\mathbb{N}} L^n$ je částečně rekurzivní jazyk.

Důkaz. (a) Náznak: Nechť L_1, L_2 jsou částečně rekurzivní jazyky a M_1, M_2 jsou Turingovy stroje, které přijímají L_1 a L_2, tedy $L(M_1) = L_1$ a $L(M_2) = L_2$. Sestrojíme M tak, že $L(M) = L_1 \cap L_2$ tak, že M bude pro vstupní slovo w střídavě (po kroku) simulovat výpočet M_1 pro w a výpočet M_2 pro w. Pokud dojde k přijetí w strojem M_1 nebo M_2, M přijme w.
(b) podobně jako (a)
(c) DODELAT
(d) DODELAT □

Jako důsledek Věty 14, Věty 15 a Věty 16 můžeme mít následující možnosti pro jazyky a jejich doplňky:
DODELAT

5.5 Redukce

Definice 16. Nechť $f : \Sigma^* \to \Sigma^*$ je vyčíslitelná funkce a nechť $L_1, L_2 \subseteq \Sigma^*$ jsou jazyky. Říkáme, že f redukuje L_1 na L_2, pokud pro $w \in L_1$ právě když $f(w) \in L_2$. Funkci f pak říkáme *redukce jazka L_1 na L_2*. Pokud existuje redukce jazyka L_1 na L_2, říkáme, že L_1 *je redukovatelný na L_2*. Tento fakt zapisujeme $L_1 \leq_\mathrm{r} L_2$.

Věta 17. *Nechť $L_1 \leq_\mathrm{r} L_2$, pak platí:*

- *pokud L_2 je rekurzivní, pak i L_1 je rekurzivní.*
- *pokud L_2 je částečně rekurzivní, pak i L_1 je částečně rekurzivní.*

Důkaz. Dokážeme pouze první tvrzení, důkaz druhého tvrzení je velmi podobný. Nechť R je TS provádějící redukci f z L_1 na L_2 a M_2 je TS rozhodující L_2. Sestrojíme M_1, který rozhoduje L_1,

TS M_1 pro vstupní slovo w:

1. spustí TS R pro w a vypočítá tak $f(w)$.
2. spustí TS M_2 pro $f(w)$; pokud TS M_2 přijme $f(w)$, TS M_1 přijme w, pokud TS M_2 zamítne $f(w)$, TS M_1 zamítne w.

Pokud $w \in L_1$, pak $f(w) \in L_2$ a proto M_2 přijme $f(w)$, takže M_1 přijme w. Pokud $w \notin L_1$, pak $f(w) \notin L_2$ a proto M_2 zamítne $f(w)$, takže M_1 zamítne w. A tedy M_1 rozhoduje L_1. □

Větu 17 využijeme spíše v pozměněném tvaru:

Věta 18. *Nechť $L_1 \leq_{\mathrm{r}} L_2$, pak platí:*

- *pokud L_1 není rekurzivní, pak ani L_2 není rekurzivní.*
- *pokud L_1 není částečně rekurzivní, pak ani L_2 není částečně rekurzivní.*

Důkaz. plyne přímo z věty 17. □

Věta 19. *$L_{ne} = \{[M] \mid L(M) \neq \emptyset\}$ není rekurzivní.*

Důkaz. Tvrzení dokážeme redukcí univerzálního jazyka L_U na L_{ne} a aplikací věty 18.

Redukci reprezentujeme pomocí TS R, který ji vyčísluje:

TS R pro vstupní slovo $[M, w]$, kde M je TS a w je vstupní slovo nad jeho vstupní abecedou, provádí:

1. Sestaví TS M', který pro vstupní slovo x provádí:
   (a) Simuluje činnost TS M pro vstupní slovo w.
   (b) Pokud M přijme w, pak M' přijme x.
   (c) Pokud M zamítne w, pak M' zamítne x.
2. Zapíše na pásku $[M']$ a zastaví.

TS R tedy provádí konverzi $[M, w]$ na $[M']$ tak, že platí:

- Pokud $[M, w] \in L$, tj. M nepřijímá w, pak M' přijímá jakékoli vstupní slovo x a tedy přijímá neprázdný jazyk, takže $[M'] \in L_{\neq}$.
- Pokud $[M, w] \notin L$, tj. M pro w cyklí nebo jej zamítá, pak M' buďto cyklí při simulaci M pro w, nebo zamítá x, protože M zamítlo w. Jakékoli vstupní slovo x je tedy zamítnuto a $L(M) = \emptyset$, takže $[M'] \in L_{\neq}$.

Tím jsme ukázali, že $L_U \leq_{\mathrm{r}} L_{ne}$, a z věty 18 dostáváme, že L_{ne} není rekurzivní. □

5.6 Postův problém přiřazení

Mějme množinu uspořádaných dvojic řetězů jako je například tato:

$$\left\{\frac{0}{100}, \frac{01}{00}, \frac{110}{11}\right\} \tag{5.1}$$

Řešením v sadě je sekvence dominových kostek, tak že konkatenace řetězců v horní části je rovna konkatenaci řetězců v dolní části. Dominové kostky se v této sekvenci mohou opakovat. Například řešení v sadě 5.1 je

$$\frac{110}{11}, \frac{01}{00}, \frac{110}{11}, \frac{0}{100},$$

protože konkatenací řetězců v horní i dolní části dostaneme stejný řetězec 110011100.

Postův problém přiřazení je pak definován takto:

Definice 17. *Postův problém přiřazení* je rozhodovací problém „Existuje v sadě S řešení?“
Postův problém přiřazení s inicializací je rozhodovací problém „Existuje v sadě S řešení začínající danou kostkou?“

Vyjádřeny jako jazyky:

- Postův problém přiřazení:

$$\mathrm{PPP} = \{[S] \mid S \text{ je sada, která má řešení}\}.$$

- Postův problém přiřazení s inicializací:

$$\mathrm{PPPinit} = \{[S, \frac{a}{b}] \mid S \text{ je sada, která má řešení začínající kostkou } \tfrac{a}{b}\}.$$

Věta 20. PPP *pro* $|\Sigma| \geq 2$ *není rekurzivní a* PPPinit *pro* $|\Sigma| \geq 2$ *není rekurzivní.*

Důkaz. Ukážeme, že $L_U \leq_\mathrm{r} \mathrm{PPPinit}$: Potřebujeme tedy sestavit TS R, který provádí konverzi $[M, w]$ na $[S, \frac{a}{b}]$, a to tak, že M přijme w, právě když S má řešení začínající kostkou $\frac{a}{b}$.

TS R pro $[M, w]$:

1. Sestaví stroj TS $T = \langle Q, \Sigma, \Gamma, \delta, q_\mathrm{start}, q_+, q_- \rangle$, který je ekvivalentní s M, ale nikdy se nepokusí přejet levý okraj pásky.
2. Ke stroji T sestaví následující sadu S:
   (a) Vloží $\frac{\#}{\#q_\mathrm{start} w_1 w_2 \ldots w_n \#}$ so S, kde $w_1 w_2 \ldots w_n = w$.
   (b) Pro každé $a, b \in \Gamma$ a každé $q, r \in Q$, kde $q \neq q_-$,

$$\text{pokud } \delta(q, a) = (r, b, R), \text{ vloží } \frac{qa}{br} \text{ do } S$$

   (c) Pro každé $a, b \in \Gamma$ a každé $q, r \in Q$, kde $q \neq q_-$,

$$\text{pokud } \delta(q, a) = (r, b, L), \text{ vloží } \frac{cqa}{cbr} \text{ do } S$$

   (d) Pro každé $a \in \Gamma$,

$$\text{vloží } \frac{a}{a} \text{ do } S$$

(e) Vloží $\frac{\#}{\#}$ a $\frac{\#}{\sqcup\#}$ do S.

(f) Pro každé $a \in \Gamma$

$$\text{vloží } \frac{aq_+}{q+} \text{ a } \frac{q_+a}{q_+} \text{ do } S$$

(g) Vloží $\frac{q_+\#\#}{\#}$ do S

3. zapíše na pásku $[S, \frac{\#}{\#q_{\text{start}}w_1w_2\ldots w_n\#}]$ a skončí

Pokud M přijímá w – tedy T přijímá w, existuje přijímající výpočet výpočet TS T nad w. Sestrojíme sadu S, ve které existuje jen řešení odpovídající tomuto výpočtu. Toto řešení bude mít v horní i dolní části celý výpočet TS T nad w.

Dále, ukážeme že PPPinit $\leq_{\text{r}}$ PPP: Potřebujeme tedy konvertovat $[S, \frac{a}{b}]$ na $[S']$, tak že sada S má řešení začínající $\frac{a}{b}$, právě když S' má řešení.

Pro řetězec x označme $(x_\#)$ řetězec, který vznikne z x tak, že se přidá symbol $\#$ za každý jeho symbol, a označme $(_\#x)$ řetězec, který vznikne z x tak, že se přidá symbol $\#$ před každý jeho symbol. Například $(001_\#) = 0\#0\#1\#$ a $(_\#001) = \#0\#0\#1$.

TS R pro vstupní slovo $[S, \frac{a}{b}]$:

1. Sestaví S' tak, že pro každou kostku $\frac{x}{y} \in S$ dáme $\frac{(x_\#)}{(_\#y)}$ do S'.
2. Přidá $\frac{\square}{\#\square}$ do S'.
3. Přidá $\frac{\#(a_\#)}{(_\#b)}$ do S'.
4. Zapíše $[S']$ a skončí.

Tedy pro každou kostku z S TS R vytvoří kostku tak, že v horním řetězci umístíme znak $\#$ před každý jeho symbol a v dolním řetězci umístíme znak $\#$ za každý jeho symbol. Dále se přidá $\frac{\square}{\#\square}$ do S', která má soužit jako „ukončovací“ kostka. V tomto okamžiku S§ nemá žádné řešení – chybí kostka, kterou by se dalo začít. V bodě 3 přidáváme kostku, kterou je možné začít a která odpovídá iniciální kostce $\frac{a}{b}$. Jakékoli řešení sady S' tedy musí začínat touto kostkou. Takové řešení existuje právě tehdy, když sada S má řešení začínající $\frac{a}{b}$. □

5.7 Úkoly k textu

1. Pomocí redukce popsané v důkazu Věty 20 vytvořte sadu pro PPP, která odpovídá výpočtu Turingova stroje M_1 nad slovem 0000.