# Kapitola: Aplikácie viet o rekurzii

V tomto kuse textu si ukážeme, že veta o rekurzii (nech už uvažujeme hociktorú jej formu) je skutočne silným tvrdením – viaceré iné výsledky, ktoré sme si v priebehu semestra ukazovali, z nej priamo vyplývajú.

Technické detaily použitia vety o rekurzii nebudeme rozpisovať. Vo všetkých prípadoch ju použijeme rovnako: využijeme, že každý program si vie zostrojiť svoj zdrojový kód a prípadne ho potom ďalej používať.

## 1.1 Problém zastavenia

Z vety o rekurzii priamo vyplýva, že množina $HALT$ nie je rekurzívna.

Sporom. Nech existuje rekurzívna funkcia $h$ rozhodujúca príslušnosť do $HALT$. Potom zostrojíme nasledovný program:

```
def oklam(x):
    do premennej p zostroj zdrojový kód programu oklam
    spusti h na p
    ak h akceptoval, zacykli sa, inak akceptuj
```

Tento program si teda zistí, čo si o ňom samom myslí $h$, a následne naschvál spraví naopak – čo je spor s predpokladom, že $h$ rozhoduje príslušnosť do $HALT$.

## 1.2 Minimálne Turingove stroje

Vo všeobecnosti je nájdenie najkratšieho programu riešiaceho daný problém ťažké. Aby sme mali istotu, že konkrétny program je najkratší možný, potrebovali by sme v princípe overiť, že žiaden kratší program nie je s tým naším ekvivalentný.

Pomocou vety o rekurzii to teraz formálne dokážeme. Presnejšie, ukážeme, že nie je ani čiastočne rozhodnuteľné, či je daný Turingov stroj minimálny. (Rovnaký dôkaz by sa samozrejme dal spraviť v ľubovoľnom inom modeli vypočítateľnosti, v ktorom platí veta o rekurzii – špeciálne teda v ľubovoľnom Turing-complete modeli.)

Sporom. Ak by tvrdenie platilo, tak existuje generujúci TS $g$, ktorý generuje množinu všetkých minimálnych TS. Zostrojíme teraz nový TS $p$, ktorý bude robiť nasledovné:

```
TS p na vstupe w:
    zostrojí svoj zdrojový kód
    simuluje g, kým nevygeneruje nejaký TS q s kódom dlhším ako p
    simuluje q na vstupe w
```

Keďže množina minimálnych TS je zjavne nekonečná – ku každému rekurzívne vyčísliteľnému jazyku existuje aspoň jeden minimálny TS. Bez ohľadu na to, aký dlhý náš TS $p$ vlastne bude, časom musí $g$ nutne vygenerovať nejaký TS $q$. No ale tým pádom sme práve dostali spor s minimálnosťou $q$, keďže $p$ rozpoznáva presne ten istý jazyk ako $q$ a je od neho kratší.

Nezaškodí si podrobnejšie rozmyslieť, ako bude $p$ vlastne vyzerať. Jeho konštrukciu začneme od konca. Na to, aby sme vedeli simulovať $q$ na $w$, bude súčasťou $p$ nejaký konkrétny univerzálny TS. Ten dostane niekde na páske zapísané $q$, inde $w$, a bude po tej páske potom behať a postupne napr. zostrojovať postupnosť konfigurácií $q$ počas výpočtu na $w$.

Pred touto časťou programu bude $p$ simulovať $g$. Súčasťou programu $p$ bude teda program $g$ a trocha réžie okolo neho – kontrolujeme dĺžky slov, ktoré vygeneroval, porovnávame ich s konkrétnym slovom, ktoré máme zapísané niekde na páske, a keď je výstup $g$ od nášho slova dlhší, beh $g$ zastavíme.

No a keď už máme toto celé, tak spravíme konštrukciu z vety o rekurzii, ktorá nám na začiatku na pásku vyrobí zdrojový kód $p$. Podobne ako u programov, ktoré vypisovali samé seba, sa touto konštrukciou dĺžka $p$ približne zdvojnásobí.

Na záver tejto časti uvedieme ešte jedno pozorovanie. Platí totiž ešte silnejšie tvrdenie. Nielen, že množina minimálnych TS nie je rekurzívne vyčísliteľná, dokonca ani žiadna jej nekonečná podmnožina nie je rekurzívne vyčísliteľná. Inými slovami, neexistuje žiadny program $g'$, ktorý bude do nekonečna generovať navzájom rôzne minimálne TS – a to ani vtedy, ak nepožadujeme, aby vygeneroval všetky.

Dôkaz je zjavný: stačí v dôkaze pôvodného tvrdenia namiesto $g$ použiť $g'$.

## 1.3 Riceova veta

Prvá Riceova veta hovorí, že každá netriviálna vlastnosť rekurzívne vyčísliteľných jazykov (resp. čiastočne rekurzívnych funkcií) je nerozhodnuteľná.

Inými slovami, ak existujú nejaké rekurzívne vyčísliteľné jazyky, ktoré danú vlastnosť majú a iné, ktoré ju nemajú, tak sa nedá napísať rekurzívny program $P$ (napr. Turingov stroj čo vždy zastane), ktorý na vstup dostane program a rozhodne o ňom, či ním rozpoznávaný jazyk danú vlastnosť má alebo nemá.

Príkladom takejto vlastnosti môže byť napríklad „rovnosť $\Sigma^*$“ u jazyka (resp. „rovnosť $\mathbb{N}$“ u množiny čísel), konečnosť, prázdnosť, obsahovanie konkrétneho prvku, atď. Jedným z mnohých dôsledkov Riceovej vety je napr. tvrdenie „nie je rozhodnuteľné o danom programe povedať, či existuje vstup, na ktorom zastane“.

Riceova veta tiež vyplýva z vety o rekurzii. Dôkaz je veľmi podobný tým, ktoré sme videli vyššie: Sporom. Nech pre nejakú vlastnosť taký $P$ existuje. Keďže je netriviálna, existujú aj program $Q^+$ ktorým rozpoznávaný jazyk tú vlastnosť má a program $Q^-$, ktorým rozpoznávaný jazyk našu vlastnosť nemá.

Zostrojíme teraz nový program $R$, ktorý spraví nasledovné

```
R na vstupe w:
    zostrojí svoj zdrojový kód
    opýta sa P, čo si o ňom myslí
    ak P odpovedal kladne, R simuluje Q- na vstupe w
    ak P odpovedal záporne, R simuluje Q+ na vstupe w
```

No a opäť máme hľadaný spor: nech by $P$ na $R$ odpovedal akokoľvek, $R$ zakaždým zabezpečí, že sa $P$ pomýlil.

## 1.4 Goedelova veta

Majme formálny systém dostatočne silný na to, aby sme v ňom vedeli formulovať tvrdenia o programoch. (Napr. naša známa aritmetika prirodzených čísel so sčítaním a násobením.)

K ľubovoľnému programu $P$, vstupu $v$ a „zápisu celého výpočtu“ $z$ teda vieme zostrojiť výraz predstavujúci výrok „$z$ je zápisom akceptačného výpočtu $P$ na $v$“. A následne teda vieme zostrojiť aj výraz $\exists z$(tamten výraz). A tento zodpovedá výroku „program $P$ akceptuje vstup $v$“.

Navyše predpokladajme, že množina *dokázateľných* tvrdení v tomto formálnom systéme je rekurzívne vyčísliteľná. Existuje teda čiastočne rekurzívny predikát $D$, ktorý akceptuje dokázateľné tvrdenia a do nekonečna počíta pre ostatné tvrdenia.

Ukážeme, že pomocou vety o rekurzii vieme, podobne ako Goedel, explicitne zostrojiť tvrdenie, ktoré je pravdivé a je nedokázateľné (lebo to samo o sebe tvrdí).

Zostrojme program $G$, ktorý spraví nasledovné:

```
G na ľubovoľnom vstupe:
    zostrojí svoj zdrojový kód
    keď už pozná ten, vie zostrojiť reťazec predstavujúci výrok "G NEakceptuje prázdny vstup"
    spustí predikát D na práve zostrojenom reťazci
    ak D akceptuje, aj G akceptuje (a inak do nekonečna simuluje D)
```

No a toto je ono. Máme explicitne zostrojený $G$, vieme teda aj explicitne zostrojiť reťazec, ktorý vyrobí v druhom kroku. A tento reťazec $r$ je náš hľadaný reťazec.

Prečo je to tak? Reťazec $r$ zodpovedá pravdivému výroku vtedy, keď $G$ neakceptuje prázdny vstup. Ak by $D$ akceptoval $r$ (teda by $r$ bol dokázateľný), tak by podľa našej konštrukcie $G$ prázdny vstup akceptoval (a teda

by $r$ nebol pravdivý). Keďže ale v našom formálnom systéme predpokladáme, že dokázateľné tvrdenia musia byť pravdivé, toto nemôže nastať.

Tým sme práve ukázali, že $D$ nemôže $r$ akceptovať. Reťazec $r$ teda nie je dokázateľný. A zároveň podľa našej konštrukcie $G$ prázdny vstup skutočne neakceptuje, $r$ teda zodpovedá pravdivému tvrdeniu.