Vyčíslitelnost

2. ledna 2013

Obsah

1 Úvod

Intuitivní teorie množin měla problémy – spory. Chce to tedy zavést teorii prvního řádu, která je bezesporná. Kdyby byla i úplná, tak by to bylo ještě lepší.

Když byla aritmetika s $+, \cdot$, tak se nedalo rozhodnout, jestli tam nějaká věta patří, nebo ne. Když tam bylo jen $+$, tak rozhodnutelnost dokázat šla.

Gödel dokázal, že rozumná teorie (pokud má něco „užitečného“), tak v ní nelze dokázat bezespornost. A je také nerozhodnutelná.

2 Turingovy stroje

Má nekonečnou pásku, na každé buňce jeden z konečně mnoha znaků, vnitřní stav z konečně mnoha stavů, rozhoduje o změně stavu, posunu a zápisu.

Obvykle se zapisuje jako pětice $T = (A, Q, \delta, \overline{q}, F)$.

- A je vnější abeceda
- Q je vnitřní abeceda
- $\delta : (Q - F) \times \{A \cup \{\Lambda\}\} \to \{-1, 0, 1\} \times \{A \cup \{A\}\}$ přechodová funkce, je v ní zadán program
- $\overline{q}$ počáteční stav
- F koncové stavy

Existují i jiné modifikace (konečné, Λ součást abecedy, etc).

Konfiguraci zapisuji obvykle jako $UqsV$, kde U, V jsou kusy před a za čtecí hlavou (které obsahují vše „zajímavé“ – prázdno do nekonečna můžu zahodit), q je stav a s je čtecí hlava.

Turingův stroj umí v zásadě všechno, ale programovat ho je složité. Proto lze vytvořit některé operace:

- Skládání – pustím dva stroje „za sebou“
- Větvení – jeden má dva koncové stavy, za jeden pověsím jeden další stroj, za druhý další.
- Jde pospojovat i cyklus.

2.1 Modifikace

Možnost nepovolit mu zůstat na místě – je stejně silný, můžu popojít tam a zase zpět.

Pokud se můžu pohybovat jen jedním směrem, degraduje to na konečný automat.

Pokud bude páska jen na jednu stranu, tak můžu zmergovat na liché/sudé, nebo kartézký součin abecedy (a udělat ji dvoustopou).

Když zavedu omezovače – okraje – umí poznat, kde končí, musí se posouvat.

Omezení činnosti – když mu povolím udělat jen dvě věci zároveň, tak mu to stačí. Dokonce stačí, když bude smět dělat jen jednu věc. Není jednoduché dokázat.

Omezení na počet písmen – stačí dvě různá $(\Lambda, 1)$ – budu kódovat sekvence.

Mít jen jeden aktivní stav nestačí – není se dle čeho rozhodnout. Jsou potřeba alespoň dva aktivní stavy.

Univerzální turingův stroj – bere kromě vstupu také program, ten interpretuje. Budeme tvrdit, že se programy podmíněně rovnají, pokud se zastavují v obou případech stejně a když se zastaví, dají stejné výsledky. Musím si pamatovat stavy, přecházet a prohledávat program. Poněkud pomalé.

2.2 Složitost

Obvykle se uvažuje prostorová a časová. Na počítání složitostí se hodí, protože je to jednoduchý model.

Můžeme uvažovat více pásek, čímž se to zrychlí. Počítá se, že složitost je alespoň délka vstupu, ale u prostorové to tak není – počítá se jedna vstupní páska (jen pro čtení) a jedna přepisovatelná páska.

2.3 Univerzální turingův stroj

Existuje univerzální turingův stroj, tj. takový, který dokáže simulovat libovolný jiný. Jeho vstupem je původní vstup (překódovaný do jeho abecedy) a program původního stroje. Má pásku, kde má vstup, kód stroje, ohraničení a oddělovače a odkládací místo (na stav, aktuálně přečtené políčko, etc). Na vstupu si označuji místo vstupu, kde je zrovna hlava.

Je potřeba všechno kopírovat po bitu a pobíhat, resp. hledat instrukce a podobně.

2.4 Halting problém

Máme daný program a data. Ptáme se, jestli se zastaví. Není algoritmicky rozhodnutelný – neexistuje turingův stroj, který se vždy zastaví a rozhodne. Použiji kantorovu diagonální metodu. Předpokládejme, že máme takový program. Vezmeme ho jako číslo, tedy musí umět být vstupem sama sebe.

Funkce $\varphi : \mathbb{N}^k \to \mathbb{N}$ je ***turingovsky vyčíslitelná***, jestli existuje turingův stroj, který ji vyčísluje. Nezáleží na kódování čísel.

3 Primitivně, částečně rekurzivní funkce

Přístup, který zvolil Göedel. Ve funkcionálním přístupu abstrahuje od implementace, nezajímáme se o to, jak se to dělá. Má to induktivní charakter. Máme axiomy a odvozovací pravidla.

Máme 3 druhy základních funkcí:

- Nulová funkce ($o(x) = 0$).
- Následník)$S(x) = x + 1$).
- Projekce ($I_n^j(x_1, x_2, \ldots, x_n) = x_j$).

Dále máme odvozovací pravidla:

- Operátor substituce – nahradí proměnnou za funkci.
- Operátor primitivní rekurze $R_n(f, g)$, kde f má $n - 1$ proměnných, g má $n + 1$ proměnných. $h(0, x_2, \ldots, x_n) = f(x_2, \ldots, x_n)$ a $h(y + 1, x_2, \ldots x_n) = g(y, h(y, x_2, \ldots, x_n))$. Tedy, máme inicializaci (f) a krok.
- Operátor minimalizace $M_n(f)$, f má $n+1$ proměnných. $h(x_1, \ldots, x_n) = (f(x_1, \ldots, x_n, z) = 0) \wedge \forall j < z f(x_1, \ldots, x_n, j) <> 0$.

Toto je ve všech rozumných jazycích jednoduše implementovatelné.

Třída primitivně rekurzivních funkcí je třída obsahující základní funkce a je uzavřená na první dva operátory.

Částečně rekurzivní funkce jsou uzavřené i na třetí operátor.

Obecné rekurzivní funkce musí být definované všude.

Tyto funkce – z jejich odvození – lze implementovat.

Odvození funkce je posloupnost funkcí, z nichž každá je buď primitivní, nebo vzniká z předchozí pomocí povolených operátorů. Samozřejmě je potřeba

přidat i kterou základní, nebo z kterých předchozích se tvoří, etc. Poslední je ta funkce. Odvození hraje roli programu.

Primitivně rekurzivní funkce jsou všude definované a efektivně vyčíslitelné – idukcí dle složitosti formule.

Tvrzení 1 *Primitivně rekurzivní jsou vlastní podmnožinou obecných rekurzivních a ty jsou vlastní podmnožinou částečných rekurzivních funkcí.*

Důkaz:
Neostré inkluze jsou zřejmě vidět.

Ackermanova funkce patří do obecně rekurzivních, ale není primitivně rekurzivní – for cyklus na ni nestačí.

☻

Příklad 1:

Součet je primitivní rekurzivní funkce. Stačí použít operátor primitivní rekurze.

☺

Primitivní rekurzivní funkce jsou právě všechny funkce, které jsou naprogramovatelné v programovacím jazyce s pomocí for cyklu. To, že ackermanova funkce není primitivně rekurzivní je zřejmé z toho, že nám nestačí konečně mnoho vnořených for-cyklů.

Množina M je ***rekurzivní***, pokud χ_M (charakteristická funkce) je obecně rekurzivní. Tedy, jsou to právě algoritmicky rozhodnutelné množiny.

Predikáty (vlastnosti, podmínky) – lze ztotožnit s množinami.

Tvrzení 2 *Spojky výrokového počtu zachovávají rekurzivitu (libovolnou) predikátů. Stejně tak omezená kvantifikace.*

4 Relativní vyčíslitelnost

Máme turingův stroj s orákulem, říká jestli něco je v množině B – vždy se zeptá, dostane odpověď.

Také jde říct tak, že se smí zeptat uživatele na ano/ne.

Lze definovat jako přidání další funkce C_B, což bude charakteristická funkce množiny B.

B-obecně rekurzivní funkce a tak podobně, jsou definované obvyklým způsobem.

Jsou ekvivalentní. Funkce jdou simulevat pomocí orákula na TS.

Opačně, máme pevně daný program, uděláme výpočtový strom, podle odpovědí. To může mít konečné i nekonečné větve. Zajímají nás končící výpočty. Budeme to prohlížet do šířky, kdykoliv bude konečná větev, tak ji vygeneruji.

V těch konečných se může zeptat jen na konečně mnoho věcí (to je kompaktnost výpočtů). Těch, kterých končí, je rekurzivně spočetně.

4.1 Částečně rekurzivní funkcionál

Je rekurzivně spočetná Φ množina trojic typu $\langle \sigma, x, y \rangle$ taková, že $\langle \sigma, x, y \rangle$ a $\langle \sigma^1, x, y^2 \rangle$ (kde σ je začátek σ^1), potom $y = y^1$.

σ je kus orákula potřebný pro výpočet, x je vstup, y je výstup. Když už se to rozhodlo, tak to nesmí měnit názor.

Definujeme, že $\Phi(\sigma)(x) \cong y \Leftrightarrow \langle \sigma, x, y \rangle \in \Phi$.

Můžu Φ aplikovat na množinu, získám funkci ($\Phi(B)$ je funkce). Numerace částečně rekurzivních funkcionálů. Můžu očíslovat programy.

Existuje primitivně rekurzivní funkce (regularizační), že $W_{\rho(z)} \subseteq W_z$, $W_{\rho(z)}$ je částečně rekurzivní funkce. A W_z částečně rekurzivní funkcionál, potom $W_z = W\rho(z)$.

Věta 1 *Existují primitivně rekurzivní funkce $\overline{S_m}$ takové, že $\Phi_z(B)(x_1, \ldots, x_m, y_1 \ldots y_m) \cong \Phi_{\overline{z,x_1,\ldots,x_m}}(B)(y_1, \ldots, y_m)$. Tedy, je to zacházení s programy bez závislosti na vstupu.*

Operace skoku = relativizovaný halting problem. Dá se to udělat úplně stejně, jako halting problem. Lze to iterovat.

Vlastnosti:

- A' je A-rekurzivně spočetná.
- A' není A-rekurzivní.
- $B \leq_1 A' \Leftrightarrow B$ je A-rekurzivně spočetná.
- A je B-rekurzivně spočetná a $B \leq_t C$, potom A je C-rekurzivně spočetná.

A je limitně vyčíslitelná, pokud nějakou dobu přeýšlíme a rozhodujeme, jestli tam prvek je, nebo není a nakonec se to stabilizuje.