Teoretická informatika v kurzech logiky (zejména v kurzech pro filozofické obory)

Martin Víta

Katedra logiky Filozofické fakulty Univerzity Karlovy v Praze
e-mail: martin.vita@vol.cz

Abstrakt

Vyčíslitelnost, formální jazyky a automaty patří stejně tak jako elementy výpočetní složitosti mezi základní kameny vzdělání v oboru logika. V naprosté většině kurzů logiky na filozofických oborech jsou však tato témata z nejrůznějších důvodů poněkud opomíjena. V tomto příspěvku si ukážeme alternativní přístup k výuce logiky spočívající v obohacení obvyklé linie výkladu o některé pojmy z oblasti automatů a gramatik. Mnohé významné logické koncepty (např. korektnost nebo úplnost) lze totiž velice průhledným a přímočarým způsobem demonstrovat právě v kontextech teoretické informatiky. Dále představíme několik softwarových nástrojů sloužících k interaktivní výuce vyčíslitelnosti, resp. Turingových strojů, a naznačíme jejich možné využití v rámci kurzů logiky.

Základní kurzy logiky jsou standardní součástí mnoha studijních oborů vyučovaných na filozofických fakultách. Pohledem do sylabů zjistíme, že student by měl v těchto kurzech získat především:

- základní algoritmické dovednosti (vyplývání ve výrokové logice, řešení sylogismů),
- zkušenosti s prací ve formálních systémech (odlišení práce uvnitř a vně systému),
- znalosti významných logických konceptů, prostředků a (meta)vět logiky (korektnost, úplnost, rozhodnutelnost, self-reference, diagonalizace, ...),
- povědomí o vztahu logiky a ostatních disciplín,
- přiměřené informace o současném stavu oboru.

Tyto body lze označit za cíle základních kurzů logiky. Je přitom patrné, že každý z nich má přirozený teoreticko-informatický kontext.

V první části tohoto příspěvku bych chtěl představit alternativní přístup k výuce logiky v pregraduálním studiu, který spočívá v rozšíření obsahu kurzů o několik pojmů teoretické informatiky – především z teorie formálních jazyků a vyčíslitelnosti. Domnívám se, že tato cesta umožní nejen preciznější formulaci některých logických pojmů (spolu s jejich algoritmickými aspekty), ale zejména studentům průzračnějším způsobem osvětlí podstatu – či spíše hlavní myšlenky – klíčových (meta)vět logiky. V neposlední řadě jde též o další příležitost vyzkoušet si práci uvnitř formálního systému a uvažování o něm. Získané poznatky jsou navíc dobře využitelné i v dalších oblastech, s nimiž se studenti filozofických fakult často setkávají: teorie mysli, kognitivní vědy, ale také například lingvistické disciplíny atp. Zároveň jde také o jakousi pozvánku do světa na pomezí logiky a informatiky, v němž se v současné době nachází řada otevřených problémů.

Výklad zmiňovaných partií se může vhodně prolínat s výkladem logiky již od naprostého začátku.

1 Formální jazyky a gramatiky

Pojmový aparát, který je zapotřebí k elementárnímu seznámení s teorií formálních jazyků, není naštěstí příliš rozsáhlý. Zahrnuje v podstatě jen pět základních definic: abeceda, slovo, jazyk, gramatika a jazyk generovaný danou gramatikou, viz např. [5] nebo on-line materiál [6]. Každou z těchto definic je dobré doplnit několika ilustrativními příklady, přičemž hlavním zdrojem může být právě výroková logika.

Ukažme si nyní, jak je možné začlenit toto téma do přednášek, resp. cvičení z logiky. Uvažujme např. gramatiku s množinou neterminálních symbolů $\{S\}$, množinou terminálních symbolů

$$\{p, q, r, \neg, \wedge, \vee, \rightarrow, \leftrightarrow, (,)\},$$

počátečním neterminálním symbolem S a pravidly:

$$S \rightarrow p \,|\, q \,|\, r$$
$$S \rightarrow \neg S \,|\, (S \wedge S) \,|\, (S \vee S) \,|\, (S \rightarrow S) \,|\, (S \leftrightarrow S)$$

Je na první pohled zřejmé, že tato gramatika generuje právě všechna slova, která jsou podle definice správně vytvořenými formulemi výrokové logiky nad výrokovými proměnnými p, q, r.

Vzniká zde přirozený prostor pro experimentování: studenti mohou zkusmo s použitím této gramatiky generovat různá slova nebo mohou k dané správně utvořené formuli výrokové logiky zkonstruovat její derivaci anebo třeba zdůvodnit, proč k danému řetězci symbolů, které není správně utvořenou formulí, neexistuje v této gramatice příslušná derivace (*zde stojíme mimo systém a uvažujeme o něm zvnějšku*).

Studenti si tak postupně osvojují práci uvnitř a vně daného formálního systému a uvědomují si, co to znamená aplikovat pravidla v rámci nějaké hry.

Výše uvedená gramatika může být východiskem pro další práci. Rigorózní definice správně utvořené formule se obvykle doplňuje úmluvou, že krajní závorky je možné vynechat. Tento fakt není v naší gramatice zachycen: naše původní gramatika tedy sice generuje slova, která patří do množiny správně utvořených formulí výrokové logiky, ale negeneruje je *všechny*. Nabízí se proto otázka, jak tuto gramatiku upravit, aby naši úmluvu respektovala.

Gramatika generující pouze správně utvořené formule s vynechanými krajními závorkami může vypadat např. takto (mezi neterminální symboly přibylo A):

$$S \to (A \wedge A) \,|\, (A \vee A) \,|\, A \to A \,|\, A \leftrightarrow A$$
$$A \to \neg A \,|\, (A \wedge A) \,|\, (A \vee A) \,|\, (A \to A) \,|\, (A \leftrightarrow A)$$
$$A \to p \,|\, q \,|\, r$$

Námi požadovaný jazyk je sjednocením jazyků generovaných těmito dvěma gramatikami. (Je nepříliš náročné vytvořit nyní gramatiku, která bude tento jazyk generovat „rovnou“.)

Běžnou praxí je rovněž vynechávání závorek v případě několikanásobných konjunkcí, resp. disjunkcí atp. To vše může být „materiálem“ například pro domácí cvičení. Další zdroje inspirace pro práci s gramatikami mohou být různé varianty zápisu formulí (polská notace, reverzní polská notace) nebo třeba jazyk termů aritmetiky, který později využijeme při výkladu predikátové logiky.

V takovýchto situacích je naprosto přirozené klást si otázky, zda daná gramatika vygeneruje pouze slova, která náleží do daného jazyka, a naopak, zda každé slovo daného jazyka se dá pomocí gramatiky vygenerovat. Je na první pohled zřejmé, že se jedná o korektnost a úplnost (ve smyslu teoretické lingvistiky); paralela ve světě logiky je naprosto přímočará. Tento jednoduchý příklad měl ilustrovat, že je možné seznamovat studenty s některými významnými logickými koncepty, aniž bychom k tomu explicitně potřebovali definice příslušných pojmů logiky.

Na tuto intuitivní představu pak můžeme později navázat při formulaci „skutečné“ věty o korektnosti a úplnosti pro výrokovou logiku. Hledáme-li logický kalkul, který by byl vůči nějaké dané sémantice úplný, snažíme se sestrojit systém axiomů a odvozovacích pravidel tak, aby takovýto systém generoval (přesněji umožňoval odvodit) všechny tautologie, a požadavek korektnosti vlastně říká, že nechceme, aby generoval jiné formule než tautologie. Jde vpodstatě o tutéž ideu, pouze v jiném kontextu.

Je zjevné, že se vstupem do oblasti formálních jazyků se stane jasnějším i význam dělení logiky na její syntaktickou a sémantickou stránku.

2 Algoritmy, Turingovy stroje a logika

Zhruba od dvacátých let 20. stol. hraje v logice klíčovou roli pojem algoritmu. Tento pojem se dá formalizovat mnoha ekvivalentními způsoby, např. pomocí částečně rekurzivních funkcí nebo Turingových strojů (TS). V úvodních kurzech logiky většinou padne informativní zmínka o nerozhodnutelnosti predikátové logiky v kontrastu s rozhodnutelností logiky výrokové, nicméně tato zmínka se v naprosté většině případů neopírá o patřičné zázemí teorie algoritmů. Nejinak je tomu u Gödelových vět.

Podobně jako v případě formálních jazyků a gramatik však stačí doplnit obsah kurzu logiky o několik málo definic a příkladů: jde konkrétně o definice Turingova stroje a jeho výpočtu. Zdrojem může být např. [7] nebo opět on-line materiál [6].

Po několika ukázkách již můžeme nechat studenty samostatně pracovat s tabulkou přechodové funkce a daným vstupem na pásce (opět si tak vyzkouší práci uvnitř a vně systému).[1] Přitom demonstrujeme „jevy“ jako zacyklení a zastavení.

Poté lze přistoupit k samotnému konstruování Turingových strojů, které realizují jednoduché operace. Vhodná cvičení k této části je možné najít např. v [4].

Po úvodním „praktickém“ seznámení s TS je možné poukázat na jeden významný fakt: TS pracují se slovy zapsanými na pásce – a konkrétní TS, resp. jeho výpočet, je možné do podoby slov zakódovat. Zde se poprvé mohou studenti setkat s jevem self-reference. Odtud už je jen krok ke konstrukci univezálního Turingova stroje a zmínce o problému zastavení a pojmu nerozhodnutelnosti. Toto je další příklad, v němž se

[1] Tento přístup používá doc. Fiala ve své přednášce z analytické filozofie na FF ZČU. Studenti mají v jedné úloze zjistit, jakou funkci realizuje TS s danou přechodovou funkcí (jde o násobení přirozených čísel zapsaných v unárním kódování), viz [3].

studenti mohou setkat s významnými logickými koncepty mimo svět logiky. Zároveň se jedná o oblast, která je podnětná pro ty, kteří se chtějí zabývat teorií a filozofií mysli.

Na základě kapitoly o Turingových strojích je možné na konci kurzu neformálně vyložit ideu časové a prostorové složitosti algoritmů a zmínit se např. o tématu důkazové složitosti. Takto je možné studentům zprostředkovat vhled do jedné z oblastí logiky, která se v současné době široce rozvíjí.

3 Simulátory Turingových strojů

Samotný výpočet Turingova stroje je záležitost poměrně neintuitivní a jeho provádění na papíře může být někdy časově náročné. Existují ale nejrůznější softwarové nástroje, pomocí nichž lze jak výpočet, tak i proces konstruování TS vizualizovat. A právě tyto nástroje budou obsahem druhé části příspěvku.

Simulátory TS zpravidla umožňují:

- interaktivně, většinou pomocí myši a dialogových oken, editovat programy pro TS,
- sledování běhu vytvořeného programu (stav pásky, vnitřní stav TS),
- trasování a debuggování jednotlivých programů.

Programem v tomto kontextu míníme sadu instrukcí pro TS, která realizuje daný algoritmus.

Simulátory TS se dají z hlediska správy a interakce s uživatelem rozdělit do dvou základních skupin:

1. **Standalone aplikace** – ty se na daný počítač obvykle instalují, jsou zpravidla závislé na platformě a poskytují většinou řádově větší funkcionalitu než ostatní nástroje. Mezi hlavní zástupce patří:
   - Turing's World: `http://www-csli.stanford.edu/hp/#Turing`
   - Visual Turing: `http://cheransoft.com/vturing/`
   - Deus Ex Machina: `http://www.ics.uci.edu/~savoiu/dem/`
   - JFLAP: `http://www.jflap.org/`
2. **On-line aplikace**, které se neinstalují (Flash animace, aplety, skripty prováděné na serveru atp.). Jsou dostupné přes www rozhraní, jejich možnosti jsou však spíše omezené; jako příklad uveďme:

- Animace z teoretické informatiky, VŠB–TUO:
  `http://www.cs.vsb.cz/jancar/TEORET-INF/ANIMACE/TuringuvStroj.html`
- Aplet Turing Machine Simulator:
  `http://ironphoenix.org/tril/tm/`
- Simulátor TS s webovým rozhraním:
  `http://infohost.nmt.edu/~prcm/turing/`
- Aplet: `http://www.turing.org.uk/turing/scrapbook/tmjava.html`

Dále se budeme zabývat pouze smulátory z první skupiny.[2]

4 Stručné seznámení s některými simulátory TS

Následující přehled je částečně převzat z článku [1], dále byly využity informace z dokumentace k jednotlivým programům.

4.1 Turing's World (3.0 for the Macintosh)

Jde patrně o nejpropracovanější z uvedených programů, nejedná se však o volně šiřitelný software – je distribuován spolu s publikací [2]. V současné době bohužel neexistuje verze pro platformu Windows ani Linux. Tento nástroj umožňuje vizálně vytvářet stavové diagramy TS, pracovat s „podprogramy" (submachines), nedeterminismem a jako jeden z mála umí vytvořit pro daný stavový diagram tabulku přechodové funkce (zobrazit přechodovou funkci jako sadu instrukcí). Při konstruování TS je možné vytvářet vlastní abecedu a užívat proměnných (wildcards). Samozřejmostí je možnost editace stavu pásky nebo její načítání ze souborů, trasování a debuggování TS atp.

4.2 Visual Turing

Volně šiřitelný simulátor dostupný pouze pro platformu Windows. Vyznačuje se komfortním uživatelským rozhraním, které připomíná vývojová prostředí pro tradiční programovací jazyky a nabízí bohaté mož-

[2] Pro úplnost bychom ještě mohli zmínit skupinu ostatních simulátorů – software v rozličných podobách určený k nejrůznějším účelům: Turingův stroj popsaný v jazyku Prolog (`http://ktiml.mff.cuni.cz/~bartak/computing/index.html`), dále nástroje ovládané z příkazové řádky atp. Svými cíli a funkcionalitou ovšem tyto programy spadají mimo rámec tohoto článku.

nosti editace projektu (mimo jiné např. vlastní abeceda, stav pásky, proměnné). TS v něm není reprezentován obvyklým stavovým diagramem, nýbrž pomocí jakéhosi modifikovaného vývojového diagramu. Umožňuje používání podprogramů, trasování a debuggování. Součástí je několik ilustrativních příkladů.

4.3 Deus Ex Machina

Víceúčelový volně šiřitelný nástroj určený pro výuku teorie automatů dostupný pro nejrůznější verze MS Windows (vč. 16-bit). Uživatelské rozhraní je jednoduché, ale velice přehledné. Obsahuje řadu řešených příkladů.

4.4 JFLAP (6.0)

Taktéž víceúčelový nástroj sloužící k výuce teorie automatů a gramatik, tedy nejen k práci s TS. Jde volně šiřitelný software kompletně napsaný v Javě – je tedy k dispozici pro naprostou většinu používaných platforem. Uživatelské rozhraní je skromnější, nicméně velice účelné. TS jsou reprezentovány podobně jako v prostředí Visual Turing pomocí modifikovaného vývojového diagramu. V tomto nástroji lze pracovat s nedeterminismem a také s několikapáskovými stroji. K tomuto nástroji existuje bohatá dokumentace na www stránkách.

5 Možnosti využití simulátorů TS ve výuce logiky

J. Barwise a J. Etchemendy ve své publikaci [2], která je dodávána spolu se softwarem Turing's World, uvádí několik cvičení s logickou tematikou (čísla 54–62). Jde o implementaci algoritmů (TS) na:

- rozpoznávání správně utvořených formulí výrokové logiky,
- sémanticky ekvivalentní transformace („zjednodušování", normální formy),
- určování pravdivostní hodnoty dané formule výrokové logiky při daném ohodnocení,
- zjišťování splnitelnosti dané formule,
- zjišťování tautologičnosti.

Tyto algoritmy bývají na přednáškách prezentovány v neformální podobě. Jejich implementace do podoby instrukcí pro Turingův stroj vyžaduje důkladné pochopení potřebné látky – přestože jde mnohdy spíše řešení „technických“ záležitostí, jedná se o vhodnou příležitost k uplatnění algoritmického (či spíše exaktního) myšlení. Studenti jsou hned v úvodu postaveni před problém reprezentace a kódování zadání, resp. výstupu, přičemž se objeví některé souvislosti, které nejsou zřetelné na první pohled, např. význam polské nebo obrácené polské notace, s nimiž se v některých úlohách pracuje snáze.

Zabývá-li se stejnou úlohou více studentů, je obvyklé, že se sejde větší množství různých řešení, což přirozeně vede k úvahám typu „které řešení je lepší?“ a dále (na neformální úrovni) k otázce složitosti algoritmů.

Při implementaci algoritmů v prostředí zmiňovaných simulátorů lze s výhodou použít podprogramů, které fungují jako samostatné celky, tj. řešit danou úlohu metodou „rozděl a panuj“. V případě rozsáhlejších algoritmů je možné zpracovávání jednotlivých částí (podprogramů) rozdělit mezi více lidí. Studenti tak mají možnost vyzkoušet si „fungování spolupráce v programátorském teamu“ a mají příležitost se naučit vhodně specifikovat vstupy a výstupy jednotlivých podprogramů. Konkrétním příkladem takového podprogramu, který se v kontextu výrokové logiky často využívá, je TS na generování všech možných ohodnocení n výrokových proměnných.

Využívání simulátorů TS má i svá úskalí: je zde riziko, že studenti začnou ztotožňovat Turingovy stroje s jejich vizualizacemi a vytratí se při tom důležitá idea self-reference. Nevýhodou je rovněž čas, který je nutný věnovat výkladu Turingových strojů, seznamování s konkrétním softwarovým nástrojem a někdy poněkud zdlouhavý proces konstrukce TS řešících dané úlohy. Tato nevýhoda se dá částečně eliminovat rozdělením jednotlivých částí tohoto tématu mezi přednášku a cvičení, přenecháním některých úloh studentům (domácí úkoly), dále poskytnutím dokumentace a zveřejněním dostatečného množství příkladů, které studenti mohou využít při řešení rozsáhlejších úloh jako podprogramy.

6 Závěrečné shrnutí

Cílem tohoto příspěvku bylo naznačit konkrétní způsob, jakým je možné doplnit obvyklý úvodní kurz logiky pro filozofické obory o témata z teoretické informatiky a jak při tom využít některých softwarových nástrojů simulujících Turingovy stroje. Tato témata se dají případně využít pro

přípravu samostatných volitelných seminářů z logiky. Domnívám se, že tento „alternativní" přístup k výuce základů logiky, v němž jde o demonstrování důležitých logických konceptů nejen uvnitř logiky samotné, by mohl přinést zajímavé výsledky. Ohlasy uvedené na konci publikace [2] to potvrzují.

Reference

[1] Martin, C. & Scheper, T., *Teaching Tools for Turing Machines*, 2003. `http://cmsdomino.brookes.ac.uk/tfm2003/papers/martin_scheper.pdf`

[2] Barwise, J. & Etchemendy, J., *Turing's World 3.0: An Introduction to Computability Theory*, CSLI, Stanford, CA 2003.

[3] Fiala, J., *Analytická filosofie. První čítanka*, Plzeň 1998.

[4] Švejdar, V., *Cvičení ke kursu Teorie algoritmů*, část I. `http://www1.cuni.cz/~svejdar/courses/cvalg1.pdf`

[5] Chytil, M., *Automaty a gramatiky*, SNTL, Praha 1984.

[6] Barták, R., *Automaty a gramatiky*, přednáška. `http://kti.mff.cuni.cz/~bartak/automaty/prednaska.html`

[7] Demuth, O. & Kryl, R. & Kučera A., *Teorie algoritmů I*, SPN, Praha 1989.