MATEMATICKÉ STRUKTURY V INFORMATICE (MI-MSI, 2+1)

jan.stary@fit.cvut.cz

Předběžný zápis probíhá do 23.11.

Denotační sémantika programovacích jazyků, datové typy jako svazy. Matematický základ funkcionálního programování a model λ-kalkulu. Úvod do teorie kategorií, která je v této oblasti standardním nástrojem.

Vhodným souběhem je *Funkcionání a logické programování* (`MI-FLP`) pro souvislost s funkcionálními jazyky a programování v jazyce `LISP`, založeném na λ-kalkulu. Vhodným doplňkem je přednáška *Vyčíslitelnost* (`MI-VYC`), která studuje obecný pojem algoritmu a rekurze.

Zdrojový kód programu je text v jakémsi formálním jazyce. Tak jako ostatní jazyky, ať už přirozené či umělé, mají i programovací jazyky svou *syntax* a svou *sémantiku*.

Syntax stanovuje gramatická pravidla: které řetězce považujeme vůbec za slova a věty (jména proměnných, příkazy, ...), a jakými způsoby můžeme z jednoduchých výrazů skládat složitější. Sémantika pak přiřazuje výrazům jazyka *význam*, resp. ptá se, co tyto výrazy „znamenají“. Co „znamená“ zdrojový kód programu?

Jedna možná odpověď se nabízí: „významem“ zdrojového kódu je sada instrukcí, která vzejde z kompilace; tyto instrukce bude vykonávat nějaký předem známý procesor, na kterém každá z těchto instrukcí znamená jistou předem známou operaci v paměti. To je základem tzv. *operační sémantiky* programovacích jazyků.

Stejně přirozená je však i následující námitka: v takové sémantice je „význam“ kódu závislý na překladači, procesoru, a konkrétní representaci dat v paměti. Existují ale přeci programy běžící na různých procesorech s různými instrukčními sadami, napsané v různých programovacích jazycích, které v nějakém dobrém smyslu „dělají totéž“ — přestože na nejnižší úrovni se jedná o jiné operace nad jinou representací dat. Je přirozené požadovat, aby „význam“ takových programů byl tentýž. Takový požadavek znamená, že „význam“ programu by neměl být závislý na překladači a architektuře, ba ani na jazyce. Popíšeme Scottovu *denotační sémantiku*, která je matematickým modelem takového „významu“.

Zavedeme algebraické a topologické pojmy potřebné k takovému popisu: uspořádané množiny, svazy, monotónní zobrazení, pevné body, konvergence ve svazech; topologické prostory, topologickou konvergenci, uzávěry, oddělovací vlastnosti, báze, spojitost. Pak popíšeme datové typy jako *spojité svazy* opatřené jistou topologií, a procedury jakožto spojitá zobrazení mezi datovými typy. Předvedeme existenci svazu (datového typu), který je isomorfní se svazem funkcí na sobě. To je první známý model λ-kalkulu jakožto základu funkcionálního programování: data jsou funkce a funkce jsou data.

Pokračujeme potom úvodem do *teorie kategorií*, s důrazem na souvislosti s informatikou. Jedná se o velmi obecný pohled na „svět“ matematiky, potažmo informatiky: *kategorie* je třída *objektů* (prostorů, algeber, datových typů, grafů, ...) a *morfismů* mezi nimi (spojité funkce, homomorfismy, monotónní zobrazení, ...). Budeme zkoumat kategorie, které jsou dobrým rámcem právě pro budování denotační sémantiky (kategorie spojitých svazů je jednou z nich).