Datalog → RA

Zoznámenie sa s projektom

Keďže sa na projekte Datalog → RA pracovalo už v predchádzajúcich rokoch, prvým krokom bolo samozrejme stiahnutie a naštudovanie jeho súčasného zdrojového kódu. Projekt som skompiloval a analýzou a experimentovaním som ho skúmal, až kým som v ňom nezískal dostatočný prehľad.

ProjectionTransformation

Ako jeden z prvých problematických aspektov súčasnej implementácia sa mi hneď javila trieda ProjectionTransformation, ktorá berie zoznam booleanov a každý Tuple transformuje tak, že vyberie iba stĺpce, v ktorých je hodnota zadaného zoznamu **true**. Toto nie je pre implementáciu datalogu dostatočne univerzálne – nedokázali by sme (jednoducho) implementovať výpočet dotazu p(X, Y) :− q(Y, X). Zoznam booleanov som teda nahradil zoznamom indexov. Trieda potom pri transformácií jednoducho vyberie zodpovedajúci prvok z transformovaného Tuple. Vieme takto vyjadriť výber ľubovoľných prvkov v ľubovoľnom poradí.

Parser

Začal som s implementáciou parsera datalogu, ktorý má nahradiť ručné volanie rutín, ktoré zostavujú strom výpočtu dotazu. Naprogramoval som väčšinu lexera a jednoduchý, rekurzívne zostupný parser, ktorý vie zatiaľ parsovať iba zoznamy faktov, t. j. pravidiel typu capuje(krcma1, alkohol1). Parser zároveň trackuje polohu v súbore a dokáže korektne označiť miesto syntaktickej chyby. Plná podpora datalogu bude samozrejme pridaná počas letného semestra.

load & save

Jednou z motivácií pre implementáciu aspoň takto zjednodušeného parsera boli príkazy load a save. Tie v súčasnosti pracujú s pomerne komplikovanou formou vstupu – adresárom so zoznamom súborov s príponou txt pomenovaných podľa relácie, ktorú obsahujú. Implementácia aspoň čiastočného parsera mi umožnila nahradiť načítavanie z adresárovej štruktúry jednoduchým načítavaním priamo z datalogového súboru. Ukladacia rutina bola taktiež modifikovaná, aby celú databázu uložila do jediného súboru.

~~Predicate & Query~~

Pôvodne som plánoval aj úpravy tried Predicate a Query, ktoré majú na starosti zostavujú strom výpočtu a samotný výpočet dotazu, ešte pred začatím parsera. Vyskúšal som niekoľko podôb modifikovaného rozhrania, ale nakoniec som sa rozhodol to zatiaľ odložiť. Najlepšia podoba tried bude pravdepodobne veľmi závislá na presnej podobe a štruktúre parsera, a zdá sa mi, že bude oveľa jednoduchšie „upratovať“ tieto triedy paralelne s jeho implementáciou.