Deduktivní databáze

PDB, 28.11.2015

Máme mnoho dat, ale nechceme používat dotazovací logiku SQL, ale chceme použít predikátovou logiku – nějak v tom přemýšlet.

Máme nějaká data a nějakou informaci musíme dopočítávat (odvozovat) z toho, co máme – celá informace není kompletní.

K tomu chceme používat prvky predikátové logiky – nebo spíše její části.

Proč ne prolog? Prolog pracuje se zpětným navracením a vždy nám nabídne jeden odvozený výsledek, ale my těch výsledků očekáváme spousty, protože pracujeme s databází. Tedy nechci použít programátorský přístup.

Můžu to udělat aplikačně – najdu rezoluční systém a ten zanesu do aplikace, lze i genericky, pro lepší přizpůsobivost.

Když máme databázi, tak některá data jsou přímo v databází a ta další jsou ta, která se odvozují – odvozovací pravidla atd.

Máme tři věci, které jsou uloženy v DDB:

1. Explicitní – surová data
2. Programy, které nám umožní odvodit něco z surových dat
3. Odvozená data – vznikají užitím odvozovacích pravidel, kdyby nám to nikdi neřekl, vypadali by jako ta klasická surová data

Rysy

- Obrovské množství dat
- Způsob vyhodnocení – odlišné od Prologu
- Lze zde udělat rekurzi, ale jazyk i tak nedokáže vše

Způsobů, jak docílit nějaké dedukce je více – shora dolů, zdola nahoru, ne vždy to ale vyjde.

Predikátová logika

- Syntaxe – viz temporální DB
- Existuje několik normálních forem:
    i. Prenexová forma
    ii. Skokem forma
- Obecná snaha u těchto forem je svázání proměnných, snaha o uzavřenost formule

Wffs + modus ponens + zobecnění nám bude stačit

Dokazování – existuje více způsobů

- Syntaktický přístup: Interferenční pravidla na axiomy – zespoda nahoru nebo shora dolů
- Sémantický přístup: jdeme přes vlastnost nesplnitelnosti, máme-li formuli OK a přidáme tu, která není OK, tak rychle ověříme, že to nemůže fungovat – na tomto principu funguje NO??? v prologu
    i. Hledáme interpretací / vyjádření, hledáme pro predikáty, volné proměnné, funktorům, … nějaké hodnoty, třeba přiřazujeme chování – interpretací je nekonečně mnoho, interpretace může být modelem

Logický důsledek

K – program

W – výsledek, odvozenina, dotaz zadávaný v prologu

W může být splitelná, pokud existuje alespoň jedna interpretace, která platí

Může být platná pokud platí pro všechny interpretace

…

Relační model nám přinesl pan Codd v roce 1970, pracujeme zde s datovými doménami, nad nimi tvoříme relační schéma, různé datové typy, můžeme je různě upravovat.

Databáze je pak nějaká podmnožina kartézského součinu, formálně se na to tedy díváme jako na množinu. I když teda mlžou se nám opakovat řádky.

Máme tu pojem klíče – kandidátní, složený, …

A máme tu normální formy – ta první pak byla zrušena objektovým přístupem k databázím

Relační algebra

- Selekce (sigma)
- Projekce (pí)
- Spojení (hvězdička)

Dotazovací jazyky – DML

- Nejvíce jsme zvyklí na SQL – dotaz může být otevřený, odpověď ano/ne, a nebo uzavřený dotaz, kdy výsledkem je tabulka, to nám umožnuje vnořovat ty SQL dotazy

Relační model nad logikou

- Podáváme se na to tak, že relace je jednoduchý predikát z logiky prvního prvního řádu, př. Osoba(123, „Karel“,35, 12 000). – záleží na pořadí
- Pokud existuje řádek v databází s těmito hodnotami – 35 let starý Karel má plat 12000, tak výsledek predikátu bude true
- Chci vylistovat všechny osoby: Osoba(_, n, u, p)
- Mladší osoby: Osoba(_, n, u, p), v > 14, v < 44.

- Starší osoby: Osoba(_, n, u, p), v > 44 → tento dotaz má potenciálně nekonečně mnoho výsledků, takže by takový dotaz nemusel být přijat, je třeba omezit i shora: v < 105

Způsob vyhodnocení (Význam) pravidel

Důkazní interpretace – syntaktická

- Mohu použít axiomy, explicitně uložené informace, nebo implicitní, odvozené
- nebo mohu použít negace

Důkaz při takovém postupu:

- Vezmeme všechny explicitně uložení pravidla a potom aplikujeme modus ponens na všechna odvozovací pravidla tak, abychom získali výsledek, dá se najít nějaká optimální cesta a my pak dostaneme množinu výsledků
- Uplatňuje se postup od axiomů k teorémům

Další možnost jak to vypočítávat:

Modelová interpretace – sémantická

- Predikát nám říká pravdu nebo nepravdu
- Modelem je seznam hodnot, ve kterých jsou všechna tato pravidla pravdivá a nezáleží na pořadí

Minimální x nejmenší model

- Minimální je takový, že neexistuje menší, může jich být více, jelikož nemusí být navzájem porovnatelné
- Nejmenší je jenom jeden

My budeme pracovat s minimálním modelem, když bude správná konstalace, tak bude jediný a stane se nejmenším.

Definice minimálního modelu: On je podmnožinou všech těch ostatních a neexistuje žádný jiný model takový, který by byl menší.

Můžeme udělat uspořádání modelů, to hledáme nějakou permutaci. Nás zajímá to, že z toho minimálního modelu můžeme odvodit všechny ostatní. Ten bude vlastně výsledkem, další nebudeme zmiňovat, protože už jsou z něho odvoditelná.

Tady je problém negace:

- P(x) :- r(x) and not q(x)
- Q(x) :- r(x) and not p(x)
- DB: {r(1)}
- Získáme dva minimální modely, toto způsobuje ta negace

Výpočetní interpretace

- Pracuje se SQL rezolucí, nevýhoda je, že nedává celou množinu jako výsledek, ale třeba jen její část

Vznik těchto databází byl, že byl prolog a nestačila kapacita databáze, tak to začali integrovat a upravovat. Casem se to dostalo do nějaké formy. Typicky pokud je to systém bez negací, tak sice není moc silný, ale výsledkem je jediný minimální model. Pokud je s negací, tak tyto systémy naleznou jenom jeden z těch minimálních modelů.

Vypadá to jako prolog.

Prolog jako takový vychází z Hornových klauzolí: B :- A1, A2, …, An znamená B <= (A1 and A2 and … and An)

Platí tu tranzitivita.

Příklad: databáze s elementárními obrázky, nebo fragmenty něčeho, ten obrázek obsahuje pozici, na které začíná atd.

Složitější obrázek můžu udělat tak, že z těch prvotních seskládám jiný – určitým posunem – to můžu udělat rekurzí.

Shrnutí

- Predikáty jsou relace
- My jsme schopni seskládat výsledný obrázek nebo tabulky tak, že používáne nejenom explicitně uložená data

Tvar atomických formulí

P (A1, … An)

- Mapujeme na sérii dotazů,
- Predikát má nějaké jméno, když se to mapuje na tabulku, tak se to musí krýt a shoda počtu atributů – tvrdě dodržujeme pořadí
- Predikát lze zúžit na
    i. Konstantu, porovnání na konstantu
    ii. Unifikace proměnných

Výpočer relací pro nerekurzivní pravidla

- Princip: úprava pravidel

Jazyk Datalog

- Kombinuje několik způsobů vyhodnocení
- Původní byl bez negace, ale s rekurzí
- Určitá omezení pro pravidla
- Nelze měnit odvozovací pravidla za běhu programu (rozdíl oproti Prologu)
-