Detektor isomorfismů grafů a podgrafů

Autorizovaný software

Dokumentace

Ing. Jiří Zuzaňák

11. prosince 2009

Abstrakt

V tomto textu bude popsán nástroj, navržený za účelem detekce isomorfismů grafů a podgrafů v daném hostujícím grafu. Tento nástroj byl vykázán jako autorizovaný software v roce 2009. Nástroj je možné použít jako jednoúčelový program pro zpracování grafů reprezentovaných textovými soubory, nebo jako knihovnu umožňující integrovat popisované funkce do jiného programu.

Samotné hledání isomorfismů grafů se provádí na základě vytvoření tzv. grafového automatu, který není ničím jiným než na základě hledaných grafů zkonstruovaným rozhodovacím stromem, pomocí kterého se popisované isomorfismy hledají. Detektor byl vytvořen za účelem detekce isomorfismů grafů v rámci přepisovacího systému, který je součástí nástroje pro práci s grafovými gramatikami.

Autorizovaný software je určen ke stažení bez jakýchkoli omezení (viz. licence), s tím že při použití zdrojových kódů je nutné zachovat jméno atora a instituce.

1 Úvod

Hledání isomorfismu podgrafu je NP-úplný problém, na který se dnes zaměřuje velké množství výzkumníků, a to hlavně z toho důvodu že grafy a grafová reprezentace představují silný nástroj pro popis téměř libovolné topologické struktury. O isomorfismu grafů (tj. nikoli podgrafu) není dnes ještě rozhodnuto zda se jedná o NP-úplný problém.

Teorie grafových přepisovacích systémů (založená především na uzlově a hranově přepisovacích systémech) se dnes hojně aplikuje v překladačích programovacích jazyků pro reprezentaci a hlavně optimalizaci grafů toku programu a grafů popisujících výrazy těchto jazyků (převážně pomocí stromů).

Další možností využití grafových přepisovacích systému a tím pádem i grafových gramatik je jejich aplikace při popisu a rozpoznávání struktury složitých objektů, jakými může být například obsah obrazu[1], objekt který je součástí obrazu, nebo jakákoli jiná topologii obsahující struktura.

[1]Nad obrazem se provede segmentace a detekce objektů, z jejichž výsledků se vytvoří graf. Nad tímto grafem je pak možné provádět pomocí nadefinované grafové gramatiky různé operace (interpretace obsahu obrazu, reprezentace znalostí získaných z obrazu)

2 Základní popis softwaru

Hlavní součástí vykazovaného autorizovaného softwaru je program umožňující načítání grafů popsaných v jazyce `dot` (viz dále), nad kterými je možné provádět popsanou detekci isomorfismů.

Postup práce se softwarem je následující: uživatel pomocí jednoduchého příkazového interpretu načte množinu grafů jako hledané grafy (grafy, které se hledají v hostujícím grafu). K jednotlivým hledaným grafům je nutné přiřadit mapování označující vrcholy hledaných grafů jako propojovací (join). Dále je nutné načíst hostující graf, ve kterém se isomorfismy hledají. Poté se pomocí dvojice příkazů vygeneruje z hledaných grafů automat, který se aplikuje na hostující graf. Výsledek hledání isomorfismů je možné vytisknout ve zvoleném formátu na standardní výstup.

2.1 Jazyk `dot` používaný pro popis grafů

Pro načítání a export grafů v textovém formátu je využíván jazyk `dot`, který je součástí software pro rozvržení a vizualizaci grafů `graphviz`. Jazyk umožňuje na základě jednoduché syntaxe pojmenovat vrcholy grafu identifikátory, které jsou poté využity při popisu hran grafu. K jednotlivým vrcholům a hranám je možné přiřadit i jejich ohodnocení ve formě čísla, nebo znakového řetězce. Následuje příklad jednoduchého grafu v jazyce `dot`.

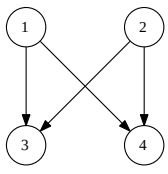
```
digraph G {

   node_1 [ label = "1" ]
   node_2 [ label = "2" ]
   node_3 [ label = "3" ]
   node_4 [ label = "4" ]

   /* edge_0 */ node_1 -> node_3 [ label = "" ]
   /* edge_1 */ node_1 -> node_4 [ label = "" ]

   /* edge_2 */ node_2 -> node_3 [ label = "" ]
   /* edge_3 */ node_2 -> node_4 [ label = "" ]
}
```

Graf popsaný výše uvedeným kódem je zobrazen na Obr. 1. Zmiňovaný graf obsahuje čtyři vrcholy ohodnocené čísly a čtyři hrany nadefinované mezi těmito vrcholy.



Obrázek 1: Graf popsaný výše uvedeným kódem v jazyce `dot`

Z důvodu potřeby identifikace grafových hran pomocí jednoznačných identifikátorů, byla do syntaxe jazyka `dot` v rámci jeho překladače přidána podpora načítání identifikátorů hran zapsaných v komentářích, čímž zůstala zachována zpětná kompatibilita s nástrojem `graphviz`.

3 Detekce isomorfismů

Graf je reprezentován dvěma množinami, množinou vrcholů a množinou hran. Množina vrcholů je složená z prvků tvořených ohodnocením vrcholu a seznamem hran incidenčních k tomuto vrcholu (hrany jsou v této množině reprezentovány indexy do množiny hran). Množina hran obsahuje prvky tvořené ohodnocením hrany, zdrojovým vrcholem a cílovým vrcholem hrany.

Množiny jsou reprezentovány pomocí struktury implementující abstraktní datový typ `red-black tree`, upravený takovým způsobem, aby bylo možné přiřadit jednotlivým prvkům množiny jednoznačný neměnný index.

Na základě načtených (nebo jakýmkoli jiným způsobem předaných) hledaných grafů je vytvořen grafový automat. Grafový automat reprezentuje rozhodovací strukturu, na základě které se hledají mapování grafových vrcholů z hledaného na hostující graf. Když automat při aplikaci na daný hostující graf přejde do "konečného" stavu (stavu, ve kterém se rozhodne o rozpoznání grafu), znamená to že byla nalezena kostra daného grafu. Nad takovou kostrou (vrcholovým mapováním) je potřeba nalézt všechny možné hranové mapování (permutace hran) z podgrafu na hostující graf. Zmíněné hranové mapování nemusí nutně existovat[2]. Součet takovýchto hranových permutací přes všechny nalezené vrcholové mapování odpovídá výslednému počtu isomorfismů hledaného grafu v hostujícím grafu.

3.1 Popis výsledných isomorfismů

Výsledné isomorfismy v rámci jednoho hledaného grafu jsou popsány množinou vrcholových a hranových mapování z indexů vrcholů a hran hledaného grafu na indexy vrcholů a hran hostujícího grafu. Detektor umožňuje dva základní způsoby interpretace nalezených isomorfismů. Prvním z nich je vytištění mapování indexů na standardní výstup a druhým způsobem je tisk zdrojového souboru v jazyce `dot`, po jehož "vysázení" je v hostujícím grafu zvýrazněn hledaný graf včetně indexového mapování mezi těmito grafy.

4 Příkazový interpret

Binární soubor, který vznikne po překladu zdrojových souborů reprezentuje jednoduchý příkazový interpret, umožňující snadné hledání isomorfismů v grafech. Vstupní grafy by měly být popsány v textových souborech v jazyce `dot`. Následuje seznam příkazů, které program interpretuje za účelem popisu zpracování grafů.

- `clean` - uvolní data reprezentované klíčovým slovem z paměti. Uvolní se také závislé datové struktury. Například při uvolnění hledaných grafů se uvolní i vytvořený grafový automat.
- `load` - načte z předaného souboru buď hledaný graf (searched), nebo graf, ve kterém se detekují isomorfismy hledaných grafů (host)

[2] V takovém případě se nalezené vrcholové mapování vyřadí z množiny výsledných isomorfismů

```
load searched "../examples/graphs/sub_3.dot";
load host "../examples/graphs/host.dot";
```

- `join vertices` - definování vrcholů hledaného grafu jako propojovacích. Tento příkaz umožňuje tří základní operace: nastavit všechny vrcholy jako propojovací, zrušit nastavení všech vrcholů jako propojovacích a poslední možností je výčet propojovacích vrcholů uzavřený ve složených závorkách. Vždy se pracuje s posledně načteným hledaným grafem.

```
join vertices all;
join vertices none;
join vertices {node_1, node_2, node_3};
```

- `create automata` - vygeneruje grafový automat na základě aktuálně načtených hledaných grafů a nastavení jejich propojovacích vrcholů. Vygenerovaný automat je poté použit k detekci isomorfismů
- `create isomorphisms` - nalezne isomorfismy grafů a podgrafů v hostujícím grafu a výsledky hledání uchová ve vhodných datových strukturách
- `print` - na základě předaného klíčového slova vytiskne na standardní výstup obsah: jednotlivých hledaných grafů, hostujícího grafu, vytvořeného grafového automatu, nebo nalezených grafových isomorfismů

```
print searched all;
print searched 0;
print host;
print automata;
print automata dot;
print isomorphisms all;
print isomorphisms dot all;
print isomorphisms 0;
print isomorphisms dot 0;
```

Následuje seznam klíčových slov zastupujících používané struktury aplikace. Klíčové slova jsou používány jako zástupné symboly v jednotlivých příkazech na vstupu programu.

- `all` - zastupuje všechny datové struktury programu. Množina zastoupených struktur může být závislá na aktuálním kontextu, ve kterém bylo klíčové slovo použito
- `searched` - hledané grafy, nebo jeden zvolený hledaný graf (v tomto případě je použit poslední načtený graf, nebo je požadovaný graf odkázán jeho indexem)
- `join vertices` - propojovací vrcholy. Pokud je vrchol nadefinován jako propojovací, je s ním nakládáno jako s vrcholem, ve kterém je podgraf napojen do hostujícího grafu. Jinými slovy, pokud není vrchol propojovací neexistuje v hostujícím grafu hrana jdoucí z/do tohoto vrcholu, aniž by existovala v hledaném podgrafu. Pokud není žádný z vrcholů hledaného grafu propojovací, tak jediné nalezené isomorfismy budou vždy kompletně isomorfní s celou komponentou hostujícího grafu

- `host` - hostující (mateřský) graf, graf ve kterém se hledají popsané isomorfismy
- `automata` - grafový automat. Struktura vytvořená za účelem hledání isomorfismů v hostujícím grafu
- `isomorphisms` - nalezené isomorfismy grafů a mapování jednotlivých podgrafů do hostujícího grafu. Detektor nalezne kompletní mapování tj. mapování vrcholů a také hran grafů

4.1 Použití programu

Program funguje po spuštění bez parametrů jako jednoduchý příkazový interpret, který načítá jednotlivé příkazy a tyto následně interpretuje. V případě že je programu z příkazové řádky předán argument, je tento považován za jméno souboru obsahujícího vstupní řetězec příkazové interpretu, který se po načtení provede. Příklady základních operací a hledání isomorfismů na ukázkových grafech je možné nalézt v adresáři `examples`.

Následují příklady možností jak s přeloženým programem pracovat

- Interaktivní příkazový interpret. Po spuštění pomocí názvu programu `./main`, vyzve uživatele pomocí promptu `>>>` k zadání prvního příkazu
- Přesměrování obsahu souboru na standardní vstup programu. Program rozpozná že vstup nepochází z terminálu a zbytečně nezobrazuje načítané symboly. Příklad spouštění programu: `./main < input.gic`, nebo `cat input.gic | ./main`, případně jiné možnosti přesměrování standardního vstupu v daném operačním systému
- Předání souboru obsahujícího příkazy jako argumentu. Program načte obsah souboru a tento poté v jednom rozkladovém kroku interpretuje. Příklad spouštění programu: `./main input.gic`

4.2 Příklad vstupu

Následující kód je možné přesměrovat na standardní vstup popisovaného programu, nebo předat název souboru obsahujícího tento text jako jeho parametr.

```
# - clean all structures -
clean all;

# - load searched graphs from dot files -
load searched "../examples/graphs/sub_0.dot";
join vertices all;
load searched "../examples/graphs/sub_1.dot";
join vertices all;
load searched "../examples/graphs/sub_2.dot";
join vertices all;
load searched "../examples/graphs/sub_3.dot";
join vertices all;

# - load host graph from dot files -
load host "../examples/graphs/host.dot";

# - create graph automata, and find graph isomorphisms -
create automata;
create isomorphisms;
```

```
# - print isomorphisms to standard output -
print isomorphisms all;
```

Uvedený příklad definuje následující kroky. První příkaz uvolní z paměti programu všechny data, tj. vyprázdní (uvolní paměť) všechny datové struktury programu. V následujících krocích jsou načteny hledané grafy a všechny jejich vrcholy jsou nastaveny jako propojovací. Po hledaných grafech je načten hostující graf. Pomocí příkazů `create automata` a `create isomorphisms`, je vygenerován grafový automat a tento je aplikován na v předchozích krocích načtený hostující graf. V posledním kroku jsou na standardní výstup vytisknuty informace o všech nalezených podgrafech.

5 Implementace a překlad programu

Program je implementován v jazyce `C/C++` s použitím standardních knihoven operačního systému. Program byl navržen a testován na Unixovém operačním systému a proto je primárně určen pro tyto operační systémy. Po drobných úpravách (knihovna `readline`) je možné použít autorizovaný software i na jiných operačních systémech.

Detektor isomorfismů je možné přeložit pomocí příkazu `make`, který na základě souboru `Makefile` vytvoří výsledný binární soubor. Pro překlad je nutné aby v systému byl obsažen interpret jazyka `Perl`, který se používá při kompilaci programu generujícího šablony jež využívá popisovaný detektor. Překlad proběhne ve dvou nezávislých krocích, kde v prvním z nich se přeloží generátor šablon `process` a v druhém kroku se vytvoří binární soubor požadovaného detektoru isomorfismů.

6 Závěr

Program vykazovaný jako autorizovaný software je možné použít jako standalone aplikaci (jednoduchý příkazový interpret), nebo využít jeho zdrojové kódy v rámci jiného programu. Autorizovaný software slouží k detekci isomorfismů grafů v orientovaných ohodnocených i neohodnocených grafech. Software obsahuje rozhraní pro jednoduchou práci s grafy popsanými v jazyce `dot` a umožňuje zápis výsledků detekce, také v tomto formátu.