Detektor isomorfismů grafů a podgrafů

Autorizovaný software

Příklad použití

Ing. Jiří Zuzaňák

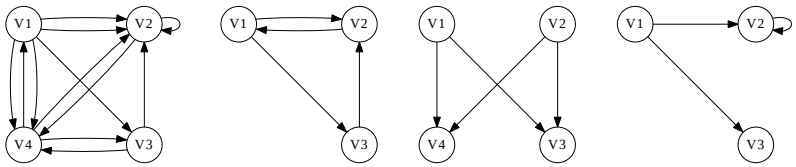
10. prosince 2009

Úvod

V tomto dokumentu je uveden příklad použití programu vykazovaného jako autorizovaný software v roce 2009. Vstupem v rámci uvedeného příkladu jsou čtyři grafy, kde tři z nich popisují hledané grafy a jeden graf hostující, ve kterém se hledané grafy detekují.

Načtení vstupních grafů

Grafy použité v tomto příkladu jsou zobrazeny na Obr. 1. Vstupní grafy, tj. tři hledané grafy a hostující graf jsou načteny následujícími bloky kódu. Příklady budou uváděny tak jako by je uživatel zadával v interaktivním režimu programu, tj. jejich součástí bude i reakce interpretu na jednotlivé příkazy.



Obrázek 1: Příklad hostujícího grafu a v něm hledaných podgrafů

- Uvolnění dosud používaných dynamických datových struktur

```
>>> # - clean program dynamic structures -
>>> clean all
INFO: searched graph set, host graph, graph automata and graph ↩
    isomorphisms has been cleared
>>>
```

- Načtení hledaných grafů a nastavení jejich propojovacích uzlů

```
>>> # - load searched graphs -
>>> load searched "graphs/sub_0.dot"
INFO: searched graph loaded: position: 0, vertex_cnt: 3, edge_cnt: 4
```

```
>>> join vertices all
INFO: all vertices of searched graph at position 0 was set as join
>>> load searched "graphs/sub_1.dot"
INFO: searched graph loaded: position: 1, vertex_cnt: 4, edge_cnt: 4
>>> join vertices all
INFO: all vertices of searched graph at position 1 was set as join
>>> load searched "graphs/sub_2.dot"
INFO: searched graph loaded: position: 2, vertex_cnt: 3, edge_cnt: 3
>>> join vertices all
INFO: all vertices of searched graph at position 2 was set as join
>>>
```

- Načtení hostujícího grafu, v kterém se detekují isomorfismy hledaných grafů

```
>>> # - load host graph -
>>> load host "graphs/host.dot"
INFO: host graph loaded: vertex_cnt: 4, edge_cnt: 12
>>>
```

Po provedení těchto příkazů jsou v paměti programu načteny požadované grafy, které je možné pomocí příkazu `print` vytisknout na standardní výstup a dále je zpracovávat detektorem isomorfismů.

- Tisk druhého z hledaných grafů je demonstrován v následujícím kódu.

```
# - print searched graph at position 1 -
>>> print searched 1
digraph G {
   rankdir = LR
   node [ shape = ellipse ]

   node_0 [ label = "" ]
   node_2 [ label = "" ]
   node_3 [ label = "" ]
   node_4 [ label = "" ]
   /* node_0_node_3 */ node_0 -> node_3 [ label = "" ]
   /* node_0_node_4 */ node_0 -> node_4 [ label = "" ]
   /* node_2_node_3 */ node_2 -> node_3 [ label = "" ]
   /* node_2_node_4 */ node_2 -> node_4 [ label = "" ]
}
INFO: searched graph at position 1, was printed to standard output
>>>
```

Nalezení isomorfismů

Pro nalezení isomorfismů je potřeba vytvořit na základě načtených grafů grafový automat. Následující bloky kódu popisují postup, kterým se tento automat vytvoří a následně použije.

- Vygenerování grafového automatu na základě hledaných grafů

```
>>> # - create graph automata from searched graphs -
>>> create automata
INFO: created graph automata (parser): state_cnt: 10
>>>
```

- Nalezení isomorfismů hledaných grafů (pomocí grafového automatu) v načteném hostujícím grafu

```
>>> # - find graph isomorphisms based on graph automata -
>>> create isomorphisms
INFO:
   searched graph 0, isomorphism count: 11
   searched graph 1, isomorphism count: 24
   searched graph 2, isomorphism count: 9
>>>
```

Příklad výstupu nalezených isomorfismů

Na základě struktur popisujících nalezené isomorfismy je možné výsledek vytisknout na standardní výstup programu.

- Příklad tisku vrcholových a hranových mapování z hledaného grafu na pozici 1 do hostujícího grafu

```
>>> print isomorphisms 1
 --- ISOMORPHISM 0 ---
VERTICES: 0000 - 0003, 0002 - 0000, 0003 - 0004, 0004 - 0002,
EDGES:    0000 - 0008, 0002 - 0009, 0003 - 0005, 0004 - 0000,

 --- ISOMORPHISM 1 ---
VERTICES: 0000 - 0003, 0002 - 0000, 0003 - 0004, 0004 - 0002,
EDGES:    0000 - 0008, 0002 - 0009, 0003 - 0004, 0004 - 0000,

...

INFO: isomorphisms of searched graph at position 1 was printed to ↩
     standard output
>>>
```

- Příklad tisku nalezených isomorfismů v jazyce dot, se zvýrazněním nalezených isomorfismů

```
print isomorphisms dot 1
digraph G {
   rankdir = LR
   node [ shape = ellipse ]

   node_0 [ color = red label = "0 > 0" ]
   node_2 [ color = red label = "4 > 2" ]
   node_3 [ color = red label = "3 > 3" ]

...

}
INFO: isomorphisms of searched graph at position 1 was printed to ↩
     standard output (in dot format)
>>>
```

Závěr

Pokud je standardní výstup programu přesměrován do souboru je možné takto získat isomorfismy zadaných grafů v cílovém hostujícím grafu. V případě generování kódu v jazyce `dot` je možné tyto isomorfismy jednoduše vizualizovat pomocí nástroje `graphviz`.