DynamicGraph

mid-term report for IB013 Logic Programming

Andrej Krejčír, Vladimír Štill, Martin Ukrop

April 7, 2013

1 User interface

DynamicGraph provides interactive command line interface. After running the compiled binary (or calling the top level predicate `dynamicGraph/0`), basic program info is displayed followed by the internal command prompt (`|:  `). All the commands are using Prolog predicate syntax - i. e. they must always end with a dot character (`.`) with optional parameters in parentheses before it. Commands are divided into several categories.

The interactive interface holds several global settings that can be altered by specific commands. These settings include currently loaded graph, boundaries of the time interval and status of the *Graphviz* support. Settings are stored as facts dynamically added to program database.

1.1 Graph manipulating commands

- `graphLoad/1`, `gl/1`
  Load graph from specified file. Syntax of the file contents is checked, if it is not correct, file is not loaded.
- `graphGenerate/1`, `gg/1`
  Generate new graph according to settings in specified file. Syntax of the configuration file is checked, if incorrect, the graph is not generated. After generation, graph is automatically saved to file `<graphname>.pl` to current folder.
- `graphGenerate/0`, `gg/0`
  Generate new graph, parameters are set interactively. Syntax of the values from user are checked, if incorrect, the graph is not generated. After generation, graph is automatically saved to file `<graphname>.pl` to current folder.
- `graph/0`, `g/0`
  Display info about currently loaded graph - name, number or nodes and number of edges.

1.2 Time commands

- `timeBegin/1`, `tb/1`
  Set beginning of time interval. Time is provided in time stamp format. Input format is checked.
- `timeBegin/0`, `tb/0`
  Set first edge occurence as beginning of time interval.

- `timeEnd/1`, `te/1`
  Set end of time interval. Time is provided in time stamp format. Input format is checked.
- `timeEnd/0`, `te/0`
  Set last edge occurence as beginning of time interval.
- `timeInterval/2`, `ti/2`
  Set beginning/end of time interval. Equivalent to setting beginning and end separately. Time is provided in time stamp format. Input format is checked.
- `timeInterval/0`, `ti/0`
  Set time interval from first to last edge occurrence. Equivalent to using `timeBegin/0` with `timeEnd/0`.
- `timeMoment/1`, `tm/1`
  Set both beginning/end of time interval to the same value. Equivalent to using `timeBegin/1` and `timeEnd/1` with the same time stamp.
- `time/0`, `t/0`
  Display currently set time interval (beginning, end).

1.3 Graphviz output commands

- `graphviz/1`, `gv/1`
  Enable *Graphviz* outputs to specified folder. Folder existance is not checked. All commands providing *Graphviz* functionality will output to this folder. To change the output folder, run `graphviz/1` again with new parameter.
- `graphvizOff/0`, `gvo/0`
  Disable *Graphviz* outputs. No following commands will create any *Graphviz* files.

1.4 Graph statistics commands

- `statsNodes/0`, `sn/0`
  Display statistics for nodes. For the set time interval, the node with the most and the least (zero included) edges is determined. Displayed is the node name and edge count. In case of multiple results, only the first is displayed.
  Note: Edges existing in any point of the time interval are considered.
- `statsEdges/0`, `se/0`
  Display statistics for edges. For the set time interval, the edge with the longest and the shortest (but non-zero) duration is determined. Displayed is the edge beginning node, end node and time interval. In case of multiple results, only the first is displayed.
- `statsComponents/0`, `sc/0`
  Display the components at the beginning of the time interval. For each component, its nodes and node count are displayed. If *Graphviz* support is enabled, graph file for this moment is generated and saved. Graph file name consists of graph name and selected time moment.
- `statsProgress/0`, `sp/0`
  Display times when graph is changing (edge is added or removed). If *Graphviz* support is enabled, graph file for each of these moments is generated. Graph file names consists of graph name and time moment of the change.
  Note: As a precaution, if the number of files is greater than 10, confirmation is required

from the user.

- `statsAnalyseNode/1`, `san/1`
  Analyse given node specified by its name. All edges to this node in the selected time interval are displayed along with their count.
- `statsMaxComponent/0`, `smc/0`
  Determines the maximal component from all the time moments in the selected interval. For the result component, its nodes, edges and size is displayed. In case of multiple results, only the first is displayed.

1.5 Miscellaneous other commands

- `help/0`, `h/0`
  Display the list of commands with short description.
- `quit/0`, `q/0`
  Quit DynamicGraph session.

2 File specification

2.1 Graph file

- synax, syntax timestamp
- vrcholy `a-zA-Z0-9` stringy zacinajuce malym pismenom
- bez hran a->a
- trvanie hran je >=0
- neE 2 hrany medzi a,b v 1 okamihu

2.2 Graph generator configuration file

- soubor může obsahovat více deklarací grafu, pak jsou generovány postupně
- deklarace začíná klauzulí `create_graph( Filename ).`
- pokračuje až po další klauzuli `create_graph( Filename2 ).`, nebo `po end_of_file.`
- příslušný typ generátoru je automaticky detekován.

More graphs? Valid Prolog file? For detailed description of the predicates, see subsection 4.1. Example of the valid graph generator configuration file follows.

```
% gen_graph.pl
graphGenerate( 'test01.graph', 'test01.dot' ).
vertices( 16 ).
edges( 8, 150 ).
newEdge( _, 0.5 ).
removeEdge( _, _, 0.5 ).
duration( 2012-4-5+14:00, 2012-4-7+22:00 ).

graphGenerate( 'test02.graph' ).
vertices( 256 ).
edges( 0, ( 256 * 255 ) / 2 ).
newEdge( 0, 0.9 ).
```

```
newEdge( Time, 0.8 ) :- Time mod 60 =:= 0, !.
newEdge( _Time, 0.1 ).
removeEdge( _Time, Duration, Probability ) :-
    Duration >= 120,
    !,
    Probability = 1.
removeEdge( _Time, Duration, Probability ) :-
    Probability is duration / 120.
duration( 1970-1-1+00:00, 2038-1-19+3:14 ).

end_of_file.
```

2.3 Graphviz files

DynamicGraph provides graphical output using the *Graphviz* library. If this output is enabled (predicate `graphviz/1`), DynamicGraph generated `.dot` files, that can be displayed using the *Dot viewer* from the *Graphviz* library. For further information of *Graphviz* and full syntax of `.dot` files, consult http://www.graphviz.org/.

3 Computing graph statistics

Solution analysis. To be added.

3.1 Random numbers

- library(random)
- načtení: use_module(library(random)).
- https://www.fi.muni.cz/~hanka/sicstus/doc/html/sicstus/lib_002drandom.html#lib_002drandom

4 Graph generator

4.1 Graph generator settings

Graph generator requires the following parameters to be set:

- počet vrcholů – `vertices( V )`.
  - $V > 0$
- interval počtu hran `edges( Min, Max )` v každém časovém okamžiku
  - musí platit `Min` $\leq$ `Max` $\wedge Max \leq \frac{V(V-1)}{2}$
- počet vrcholů
- predikát `newEdge( Time, Probability )`, který pro daný čas vrací pravděpodobnost vzniku hrany mezi libovonlými nespojenými vrcholy
  - musí platit `Min` $\leq$ `Max`
- predikát `removeEdge( Time, Duration, Probability )`, který pro daný čas a dobu aktuálního trvání hrany, vrací pravděpodobnost, že hrana v tomto časovém okamžiku zanikne

- musí platit $0 \leq$ `Probability` ≤ 1, floating point
- doba trvání grafu `duration( BeginTime, EndTime )`
  - `BeginTime` $\leq$ `EndTime`
  - `BeginTime, EndTime :: DateTime`

4.2 Graph generator modes

Settings for graph generator can be set in different 2 modes:

- **Interactive mode**
  This mode is activated when using `graphGenerate/0`. User is prompted for all settings one-by-one. Input values are checked for correct syntax. For an example of interactive graph generation, see section 5.
- **Batch mode**
  This mode is activated when using `graphGenerate/1`. Settings are loaded from external file specified in parameter. The file syntax is checked – if incorrect, the user is informed and the graph is not generated. For detailed syntax of this file, see subsection 2.2.

4.3 Parsing the settings

- pokud generátor očekává na dané pozici výstupní číselnou hodnotu, je nutné aby hodnota byla aritmeticky vyhodnotitelná (tedy se může jednat o číslo, nebo aritmetický výraz – viz `edges/2` v příkladech).
- v přípdě, že vstup nebude korektní, tedy bude například kombinovat predikáty z různých generátorů, generátor ohlásí chybu a odmítne generovat
- v případě, že je dodán nekorektní vstup a je tak zjištěno po začátku generování (rozšířený generátor), generátor vypíše varování a ukončí generování v daném okamžiku (vrátí dosud vygenerovaný graf).
- obdobně, v případě, že selže některý z uživatelem dodaných predikátů generátor vypíše varování a korektně ukončí generování v daném okamžiku
- v případě dávkového vstupu se chyby řeší pro každý graf jednotlivě

4.4 graph generation

1: $Q_A \leftarrow$ queue of all possible edges (without time – pairs of vertices)
2: **for all** minute $T \in \langle BeginTime, EndTime \rangle$ **do**
3: **for all** edge $E \in Graph(T)$ **do**
4: call *removeEdge* with current time T and current duration of E
5: remove E with *Probability*
6: **if** removed **then** push to Q_A
7: **end if**
8: **end for**
9: call $newEdge(T, Probability)$.
10: $Q_a \leftarrow emptyqueue$
11: **while** not empty $_A$ **do**
12: **if** $currentedgecount == Max$ **then**
13: push all edges from Q_a to Q_A

```
14:            continue to next minute
15:        end if
16:        E = pop(Q_A)
17:        with Probability add E to graph
18:        if not added then push add E to Q_a
19:        end if
20:        if thenQ_A is empty and currentedgecount < Min
21:            while current edge count < Min do
22:                E = pop(Q_a)
23:                with Probability add E to graph
24:                if not added then push E to Q_a
25:                end if
26:            end while
27:        end if
28:    end while
29:    swap(Q_A, Q_a)                                  ▷ Q_A was empty
30: end for
```

5 Example CLI session

```
# ================== DynamicGraph ====================
# =========== time-dependent graph analyser ===========
# Authors: Andrej Krejcir, Martin Ukrop, Vladimir Still
#
# graph: <not loaded>
# time interval: <not set>
#
# Write 'help.' to display help summary.
|: heeelp.
# Error: 'heeelp' is not a valid command.
# Write 'help.' to display help summary.
|: help.
# === List of commands
# All commands are prolog predicates and must end with a dot.
# Shorter command equivalents are written in parentheses.
#
# == Graph manipulating commands
# graphLoad/1           (gl/1)  load graph from specified file
# graphGenerate/1       (gg/1)  generate new graph (settings from file)
# graphGenerate/0       (gg/0)  generate new graph (interactive)
# graph/0               (g/0)   display info about currently loaded graph
#
# == Time commands
# timeBegin/1           (tb/1)  set beginning of time interval
# timeBegin/0           (tb/0)  set first edge occurence as beginning
# timeEnd/1             (te/1)  set end of time interval
```

```
# timeEnd/0             (te/0)  set last edge occurence as end
# timeInterval/2        (ti/2)  set beginning/end of time interval
# timeInterval/0        (ti/0)  set time from first to last edge occurence
# timeMoment/1          (tm/1)  set both beginning/end of time interval
# time/0                (t/0)   display currently set time
#
# == Graphviz output commands
# graphviz/1            (gv/1)  enable Graphviz outputs to specified folder
# graphvizOff/0         (gvo/0) disable Graphviz outputs
#
# == Graph statistics commands
# statsNodes/0          (sn/0)  display statistics for nodes
# statsEdges/0          (se/0)  display statistics for edges
# statsComponents/0     (sc/0)  display stats for components at the beginning
# statsProgress/0       (sp/0)  display times when graph is changing
# statsAnalyseNode/1    (san/1) analyse given node
# statsMaxComponent/0   (smc/0) find maximal component
#
# == Other
# help/0                (h/0)   display the list of commands
# quit/0                (q/0)   quit DynamicGraph session
|: graphLoad('mygraph1.pl').
# Loading graph 'mygraph1.pl' ... done.
# Graph name: mygraph
# Node count: 21
# Edge count: 53
|: graphviz('graphs').
# Graphviz enabled (directory 'graphs').
|: graphGenerate.
# name( Name )?
|: test01
# Nodes( Number )?
|: 16
# edges( Min, Max )?
|: 8, 150
# newEdge( Probability )?
|: 0.5
# removeEdge( Probability )?
|: 0.5
# duration( BeginTime, EndTime )?
|: 2012-4-5+14:00, 2012-4-7+22:00
# generating 'test01.pl' ... done.
# generating 'graphs/test01.dot' ... done.
# Graph name: test01
# Node count: 16
# Edge count: 104
|: statsNodes.
```

```
# Error: time not set.
|: timeMoment(2013-04-06+19:17).
# Interval beginng set: 2013-04-06 19:17
# Interval end set: 2013-04-06 19:17
|: statsMaxComponent.
# Maximal component: { a, b, x, ds, aab}
# Component size: 5
# First occurence: 2013-04-06 19:17
|: quit.
```