Graph Logic Model Framework for Predictive Linguistic Analysis

M. Charnine[1], I. Kobozeva[2], S Loesov[3] and I. Schagaev[4]
[1]Institute of Informatics Problems, 119333, Moscow, Vavilova St., 44/2, Russia
[2]Department of Theoretical and Applied Linguistics, Moscow State University, Leninskiye Gory, 1, Moscow, Russia, 119991
[3]Institute for Oriental and Classical Studies, Russian State University for the Humanities, Miusskaya Pl.6, korp. 1, 125993, GSP-3, Moscow, Russia
[4]IT-ACS Ltd, Shephall View, SG1 1RR, Stevenage, UK

Abstract - *In this paper we are trying to combine interests of authors and collaborators into one non-mutually exclusive concept and logic flow aiming to create a framework for searching of a migration of words from one cluster to another; this enables us to define a semantic shift well before it become obvious. We show how introduced graph-logic model can be applied for analysis of migration of the meaning of sentences indicating, quite often, a paradigm shifts. Using Artificial Intelligence as an example we illustrate development of AI from philosophy of mind to science, science fiction and technology, including games in science, technology, and further education. Several examples how proposed model with supportive searching framework applied in mentioned areas detecting evolving processes are presented*.

Keywords: A Graph Logic Model, Semantic search, Long-term trends, Google Books Ngram, Historical data, Predictive linguistic analysis

1 Introduction

During our previous research over migration of terms and areas in curriculum design [Bacon13] it was discovered that areas of research and knowledge in general are moving, changing, morphing and this "evolution" can be detected and even predicted. While practical use of this conception was already proved in number of papers [Bacon13], [Bacon14] and patent [Patent07] it is worth to investigate how it can be applied in linguistics, what are the limits and what it enables in terms of analysis and monitoring of migration of terminology and corresponding semantic shifts. At first we introduce a basic model – so called graph-logic model [Schagaev14], [Schagaev15] applying it for analysis of semantic shifts and migrations of terms. Every model that describes system initially using graph theory (GT) differentiates entities (nodes, vertices) and relations – (links, arcs, edges). GT model description is static; behavior aspect of the model is described separately by introducing rules and procedures allocated to nodes and links, such as activation, or algorithmic description of process to change state on a graph.

When rules for the graph path determination are applied algorithmically for every node then moves on the graph might be defined without contradiction.

The way of leaving/arriving to/from any node can be described with the help of logic operators (LO) from the basic {OR, AND, XOR}. LO might be chosen to apply to all and every node to define conditions of leaving/arriving. Say, if one applies "XOR" operator for the node leaving condition and chooses links along the graph, we are able to redraw the graph and actually mark activated links for each node, applying XOR rule for choosing only one link.

This scheme with some other restrictive conditions is used widely and known as Markov Process (MP). MP adds to the XOR rule (applied for every vertex) a normalization condition – either one adjacent link is activated or another, while probabilities – kind of "weights" - are used to "normalize" the chances of choosing one particular link (sum of probabilities to come out from a node equals 1, note that for incoming this condition does not stand).

To complete correct introduction of travel along the graph for XOR logic one has to introduce termination condition for this travel. Termination condition in MP graph as a whole is introduced by so-called aggregate state which must be reached and the condition that one of the nodes in a column is activated when the process is leaving i-th column (one of the states from there) and arriving to i+1-th column, as well as sum of probabilities in every column is equal to 1.

Again, MP works when "XOR" logic is applied to every node as a decision making rule to leave or to arrive the node. In system programming a using "XOR" operator can describe the mutually exclusive operations – i.e. concurrency, with separation of processes at the critical section entering. In

This work is partially supported by the RFFR, Grants 16-07-00756 and 15-07-06586

"classic" probability theory "XOR" logic applies when conditional probabilities are used. Models that use conditional probabilities require XOR operators for every node of process description by definition.

Let us consider another rule of leaving a node: AND logic. In this case semantically opposite graph model is introduced that describes transitions for every node at once, at the same time, instantly; when links exist from i-th node, say, to j-th, k-th,..., x-th nodes then all possible links are activated all together at the same time.

This logical condition assumes a parallel movement along the graph from any node to all connected nodes. We name this rule applied to a node as "AND" logic. Examples of the systems that are using AND logic for every node are:

- Broadcasting networking
- Salesman problem analysis,
- Diffusion processes in physics,
- Quantum effects model,
- Parallel calculations when hardware resources are unlimited, etc., etc.

Finally, "OR" logic might be applied for each and every node on the graph, assuming that only one or some links are selected, therefore flexible parallelism might be described – when it is not necessary to start everything at once.

Three natural questions arise here:

A. Can one apply various logic operators for various nodes?

B. Do conditions at output vertices dictate input conditions to other nodes?

C. How to separate logic applied? Do they affiliate to the vertex? Or Link?

It is clear that when every node has one input and one output it does not matter. On the contrary, if a graph has several links to or from its nodes it does...

Instinctive reply to question A is yes: what one does in parallel might be exclusive at the other end of the link; therefore an answer to question B is no. It is worth mentioning that conditions to leave a node (vertex) and arrive to another vertex should be attributed to edge not vertex, Fig.1

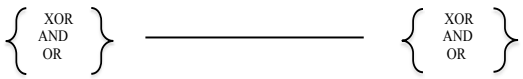

Figure 1. Logic of the edge

Their (logic operators) combination might be even more interesting: say leaving condition is "OR" but arriving condition is "AND" for each input - and we have Petri net described. To illustrate descriptive "power" of the combination of graph and logic models for behavior of graph let us draw a graph - Figure 2 - applying various logic operators for different nodes.

Let us describe Figure 2 in some details. Nodes (vertices) have output and input links. Those links that might be activated either one or another or both (node a links to node b and d are described in callout **ORo(b,d)** - that means that no order or imperative timing is required to move from vertex **a** (OR logic).

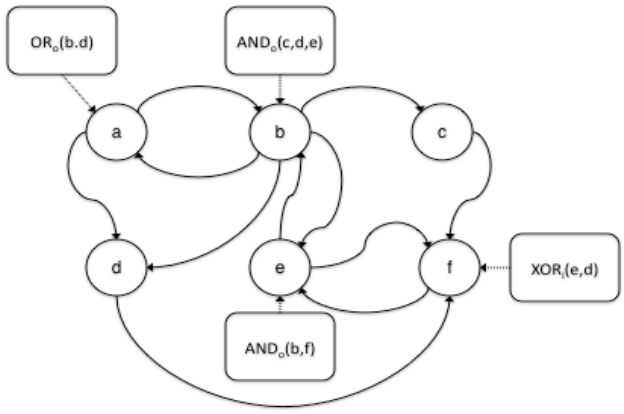

Figure 2. Various operators applied to leave and arrive

In turn, node **b** assumes parallel activation of links to nodes **d**, **e**, and **c** as it is described in special callout with operator **ANDo(c,d,e).** Note that link from node **b** to node **a** is not included in this list. Finally, input links that are required to be mutually exclusive at the node f are described by special callout as (XORi (e,d)). Again, note that incoming link from node c is not included as an input XOR operator of node f and thus might be analyzed separately.

The model proposed here in general is quite simple; it defines a graph behavior with various assumptions of leaving and arriving conditions for all vertices. This approach allows an analysis of large scale graph behavior in much more details and greater variety that the "standard" graph theory. Every node **x** of the graph such as Figure 5 thus might be described as a string:

x: AND-(j,j,k) AND+(l,m,n) OR-(p,q) OR+(r,s,t) XOR-(u,f) XOR+(g,h)

Minus "-" or "o" stand for every logic operator for output link, Plus "+" or "i" for every input link. Logic operators that have to be applied to various combinations of output links are explicitly presented in this notation.

Interestingly, leaving conditions do not obligatory match arriving ones: leaving one place together with all the rest such as parallel (AND-) calculations might be mutually exclusive at the arriving XOR – concurrency.