OMICS A Journal of Integrative Biology
Volume 7, Number 1, 2003
© Mary Ann Liebert, Inc.

The Many Faces of Process Interaction Graphs: A Data Management Perspective

AMARNATH GUPTA and BERTRAM LUDÄSCHER

We use the expression "process interaction graph" (PIG) as a general purpose term to cover instances of network-like structures with specific semantics, including signal transduction networks, gene regulatory networks, metabolic pathways and so forth, because we believe that despite the differences in biological significance, they can be treated in a uniform data management framework. Aside from the fact that they are all graph-like entities, PIGs share a few common characteristics, such as follows:

- Regardless of what the nodes and edges represent, in many cases there is an inherent temporal order (perhaps of branching time) that can partition the network into phases. Often, as in the case of gene regulatory networks, a mutation produces an alternate graph which is structurally isomorphic to the normal- case, except that the temporal properties have changed. Other gene mutations produce regulatory graphs whose subgraphs corresponding to certain phases have been altered.
- The graphs very often represents objects and how they interact; while the interaction has a generic structure like

  reactants: some molecular elements *A* and *B*
  products: other molecular elements *C* and *D*
  occurs_at: some location *L*
  catalyzed_by: some enzyme *E*
  precondition: some first-order formula $\phi(\ldots)$
  equation(*T*): some equation of type *T* where *T* can be a chemical reaction formula or an ordinary dif ferential equation with rate constants.

  there is very often a wide heterogeneity in the kind of information represented in this general structure. For example, the location *L* can have could be "at −300 to −340 base pairs upstream of gene *G*" in gene regulatory networks, but "at the lipid sublayer of the cell membrane of cell *C*" for cell signaling. In the first case, the ***semantic type*** of location is a subsequence in a DNA sequence, while in the second, it is a member of an ontology describing cellular anatomical structures and relationships.
- Very often the objects in the primary graph are members of one or more DAG-structured (sometimes tree-structured) taxonomies such as the Gene Ontology, the Enzyme classification tree or the Yeast Functional Categories. In these cases, there is often a need to query the taxonomy graph and the PIG together, as though they were a compound graph, where the "join terms" between them are the node (i.e., gene or enzyme) names.
- The graphs are often not arbitrary, but have some "discipline" in their structure. For example, while bi-directional edges and cycles usually abound in the graphs, there are likely some natural constraints (like the number of reactions in one part of the graph) that put a bound on the lengths of the cycles.
- They often exhibit the need to represent a process or a reaction as an edge as well as a node. For example, one can have edges

San Diego Supercomputer Center, University of California, San Diego, La Jolla, California.

$$e1{:}A \xrightarrow{regulates} B$$
$$e2{:}C \xrightarrow{inhibits} e1$$

where e1 is a plain edge but e2 is "an edge to an edge" because it points to e1.

It is equally important to note that in addition, to being graph structures with interesting properties, the labels on the nodes and edges of these graph typically bear semantics that are important for a logical interpretation as well as semantic query processing. In our research, we typically use these graphs in four ways:

- As ***graph structures,*** we need to perform operations like(shortest) path finding, graph pattern matching, graph differencing based on homomorphic mappings between the two graphs, and so forth
- As ***logical entities,*** we need to perform closure-like operations but often with special rules that apply to the domain in question. For example, we may need to customize the definition of a transitive closure for a relation R in the following way:

  connected(R)(X,Y) **if**

  R(X,Z), R(Z,Y), dist(R)(X,Y) $<$ 4

  indirectly_connected(R)(X,Y) **if**

  R(X,Z), R(Z,Y), dist(R)(X,Y) $>$ 3, dist(R)(X,Y) $\leq$ 6

  maybe_connected(R)(X,Y) **if**

  R(X,Z), R(Z,Y), dist(R)(X,Y) $>$ 6

- We treat the graph as input to a ***network simulation engine*** and are developing a language to *construct* a Petri Net from such the properties of the interaction graph, such that the states in the reachability network of the Petri Net can itself be searched. For example, given an edge like

  $$A \xrightarrow{regulates} B \quad \textbf{if } absent(nutrient)$$

  one can create *bound* and *unbound* states (Petri Net places) for each of *A* and *B,* and connect them through a transition called *binding*. The condition *absent(nutrient)* can be modeled as an input place supplying to the same transition. Since enumerating all states of a Petri Net is a known complex problem, we are exploring methods to make the state-search procedure more manageable.
- As ***ontologies,*** we use the graphs as intermediary knowledge bases, that is, logical entities to define integrated view over two information sources. For example, a yeast gene information source, like MIPS and a yeast proteome database can be integrated, using a process interaction network that details the transcription and translation processes.

RESEARCH ISSUES

We outline a few research issues that we think are of general interest to the data management community:

- What are suitable query or DB-programming languages for PIGs? Due to their interesting structural as well as semantic properties, a combination of graph query languages (like Graphlog, GOOD, and even a more general purpose language F-Logic) and deductive approaches can be most promising to allow scientists to explore large graphs and elicit different kinds of "connections" between the objects and processes.
- Often graph algorithms use problem specific representations and in many cases, these algorithms are for the main memory. How can we develop representations and index structures that are general purpose and have been/can be implemented in secondary memory?

- How can we develop a language to describe the properties of the specific kinds of graph that appears in a domain such that the system uses it to choose one or more appropriate representations?
- Should the query architecture for solving general-purpose PIGs be a combination of specialized "co-processors," for example, one to evaluate queries on trees and DAGs, another to process queries on graphs with bounded-length cycles and few strong components? How will query evaluation be performed in such architectures?
- How can we use a deductive engine and a large-graph query engine simultaneously, especially when earlier attempts to put together two such systems for large semantic graphs have not been very successful?

Address reprint requests to:
Dr. Amarnath Gupta
University of California, San Diego
NPACI/SDSC
MC 0505
9500 Gilman Drive
La Jolla, CA 92093-0505

E-mail: gupta@sdsc.edu