Port Graphs, Rules and Strategies
for Dynamic Data Analytics

Hélène KIRCHNER

Warsaw, July 2015
In a world of connected data and objects

Social Networking Websites

Biological Network: Protein Interaction

Research Collaboration Network

Product Recommendation Network via Emails
Context

Systems that are

- distributed and connected in networks
- massive and heterogeneous
- dynamic (interactions, evolutions, ...)

Two domains examples:

- biological systems: protein interaction
- social networks: propagation analysis (spread of innovations, rumors, or diseases, ...)

*Porgy, an interactive visual environment for port graph transformation.*

Collaboration between Bordeaux (France) and King’s College London (UK)
Biochemical Network
Regulation of cell proliferation, transformation and survival

A biochemical network composed of different types of molecules at different concentrations who interact to maintain a regulation mechanism.

Wetlab experiments suggest:
1) the overall process is controlled through only four chemical reactions
2) the regulation works if there are alternating short periods of time where the concentration of a specific molecule called A increases, and others where it remains constant.
Introduction

Model design and validation

Formalisation of the different kinds of molecules
Formalisation of rules and in-silico simulation
Plot of the evolution of the concentration of A molecules: expected staircase shape
Social Network
Propagation analysis

Study propagation mechanism in social networks:
- Global phenomenon resulting from sequences of local events
- Different metrics used to measure the propagation evolution: objectives reached by the propagation may be speed, covering, ...
- Different propagation mathematical models exist. How to compare them?

Experiment:
Propagation (disease, rumor) initiated with a starting set (seed). Two models (probabilistic cascade, linear threshold) of propagation from the same set
White for the untouched nodes - Flashy green for the active nodes- Dark green for the visited nodes- Red for the inactive nodes.
Probabilistic cascade model simulation

Linear threshold model simulation
Probabilistic cascade model simulation

Linear threshold model simulation
Probabilistic cascade model simulation

Linear threshold model simulation
Probabilistic cascade model simulation

Linear threshold model simulation
Final situation
Social Network: Analytic Visualization

An experimental approach:
Run the model and observe how works the propagation and how the objective is reached.

- Build the network
- Design the propagation rules
  - Rules describe situations where an entity can influence its neighbours
- Simulate propagation with different mathematical models
- Compare the execution traces of these models, according to a chosen metrics
- Change parameters (ex: threshold level) and run again
Challenges for modelisation

**Big networks**: need for abstraction, patterns focusing on points of interests, views

**Dynamic Evolution**: simple transformations applying in parallel and triggered by events/time controlled versus autonomous behaviour

**Uncertainty**: probabilistic / stochastic issues

**Conflicts**: detection - overlapping rules,... resolution - precedence, choices, i.e. strategic issues

**Memory and Backtracking**: history, traces
In this talk

Concepts needed:

- **Graphs** to represent networks of data or objects
- **Rules** to deal with concurrent local transformations
- **Strategies** to express control versus autonomy
- **Located** graphs and rules; scope defining strategies to focus on points of interests
- **Strategy language and strategic programs**

and research topics.
Graphs and Port Graphs
Graph Data Bases (opposed to relational data bases) have gained wide interest:
- Facebook social graph maps the interconnections between users,
- Google knowledge graph describes the semantic links between people, places and objects,
- Twitter graph database software, FlockDB, represents the links between its members.
Graph Data Bases (opposed to relational data bases) have gained wide interest:
- Facebook social graph maps the interconnections between users,
- Google knowledge graph describes the semantic links between people, places and objects,
- Twitter graph database software, FlockDB, represents the links between its members.

In 2010, the Twitter FlockDB cluster stores 13+ billion edges and sustains peak traffic of 20,000 writes per second and 100,000 reads per second.
Challenges in Graph Data Bases

Graph data used in different domains:
- Social Networks analysis
- Supply chain management
- Web of data
- Protein Interaction analysis
- Recommendation Systems
- Fraud detection in financial systems

But yet various challenges:
- No standardized graph model: Un/directed graph, Plain/structured property graph, Mixed/Hyper/Multi graph
- No standardized graph query language
- Scalability is an issue: difficult to split them up into parts and distribute them across numerous machines
Port graphs

Inspired by protein-protein interactions [IbanescuBK03], [AndreiK07], \(\kappa\)-calculus [DanosL04], BioNetGen [BlinovYFH05]

Port graphs are graphs with multiple edges and loops, where:
- nodes have explicit connection points, called \textit{ports}.
- the edges attach to ports of nodes.
- nodes, edges and ports are \textit{labeled by a set of properties}, i.e. pairs (attribute,value).

Actually equivalent to usual labeled graphs, but with more structure.
Port graphs: examples

Figure: Some examples of port graphs
Compare to labelled property graphs

Figure: Neo4j Technical Introduction.
http://dist.neo4j.org/neo-technology-introduction.pdf
Biochemical network: AKAP model

6 Chemical species occurring in the AKAP model: scaffold protein AKAP; nucleotide cAMP; protein PKA; enzyme PDE8A1 with one phosphorylation site; protein Raf-1 with one site for phosphorylation; signal protein A.
Rules
Rules - Example: AKAP model

Four chemical reactions:

\( r_1 \) cAMP activates PKA through binding;
\( r_2 \) active PKA phosphorylates PDE and Raf on the same scaffold and becomes inactive;
\( r_3 \) phosphorylated PDE degrades free cAMP and becomes unphosphorylated as well as Raf at which point Raf sends an activation signal A;
\( r_4 \) unphosphorylated PDE degrades free cAMP
A general concept of rewriting

The syntactic structure

Words, Terms, Propositions, Logic formulas, Dags, Graphs, Structured Objects, Segments . . .

The pattern : rule

Expressed with $\Rightarrow$, variables, left-hand side, right-hand side, condition or constraint

The application mode

- match to select a redex (possibly modulo some axioms, constraints,...)
- instantiate variables
- replace
Port Graph Rewriting

$G$ rewrites to $G'$ using the rule $\ell : L \Rightarrow R$ if

- there is a morphism $g : L \rightarrow G$
  
  $g$ identifies a subgraph $H$ that is equal to $L$ except at positions where $L$ has variables

- and (with appropriate rewiring)

\[ G[H] = G[g(L)] \text{ and } G' = G[g(R)] \]

This *rewriting step* is denoted

\[ G \xrightarrow{g} L \Rightarrow_R G' \]

*The choices:* position(s), rule, matching substitution(s).
*Rewriting may be concurrent, probabilistic, constraint...*
Logical foundations

**Rewriting Logic** due to J. Meseguer [TCS92]

*Rewriting logic (RL) is a natural model of computation and an expressive semantic framework for concurrency, parallelism, communication, and interaction.*

http://wrla2012.lcc.uma.es/

**Rewriting Calculus** Introduced in 1998 by H. Cirstea and C. Kirchner

*The rho-calculus has been introduced as a general means to uniformly integrate rewriting and lambda calculus.*

http://rho.loria.fr/index.html
The Abstract Biochemical Calculus
Initiated in Oana Andrei’s Phd in 2007

A rewriting calculus that models an autonomous system as a biochemical program:
- collections of molecules (objects and rewrite rules)
- higher-order rewrite rules over molecules (that may introduce new rewrite rules in the behaviour of the system)
- strategies for modelling the system’s evolution

A visual representation via port graphs and an implementation is provided by the Porgy environment [AndreiFKMNP11].
Rewriting challenges in graph data bases

- *Finding subgraph isomorphisms* - NP hard problem
  Several algorithms (comparison in [Lee& all-VLDB2013]):
  - *exact* subgraph matching returning either *one* or *all* solutions
  - *approximate* subgraph matching.
Rewriting challenges in graph data bases

- **Finding subgraph isomorphisms** - NP hard problem
  Several algorithms (comparison in [Lee & all-VLDB2013]):
  - exact subgraph matching returning either *one* or *all* solutions
  - approximate subgraph matching.

- **Probabilistic / stochastic issues** - more and more useful to deal with uncertainties and stochastic models
  Different approaches:
  - stochastic simulation in biology: the $\kappa$ language ([Danos & all])
  - probabilistic (real-time) rewrite theories defined and implemented in Maude
  - probabilistic choice of rules to be fired in $\rho$-calculus ([Bournez & all])
  - a simplified version of probabilistic non-deterministic systems in Porgy
Strategies
**Derivation tree**

Given a set of rewrite rules $\mathcal{R}$, a *derivation*, or computation from $G$ is a sequence of rewriting steps

$$G \xrightarrow{\mathcal{R}} G' \xrightarrow{\mathcal{R}} G'' \rightarrow \mathcal{R} \ldots$$

The *derivation tree* of $G$, written $DT(G, \mathcal{R})$, is a labeled tree
- whose root is labeled by $G$,
- its children are all the derivation trees $DT(G_i, \mathcal{R})$ such that $G \xrightarrow{\mathcal{R}} G_i$.

The edges of the derivation tree are labeled with the rewrite rule and the morphism used in the corresponding rewrite step.

A derivation tree may be infinite, if there is an infinite reduction sequence out of $G$. 

A derivation tree
A derivation tree
Strategies are needed

**Rewrite rules describe local transformations**
By default, rules apply everywhere...

**What to do when only selected computations are of interest...?**

**Strategies describe the control** of rewrite rule application
Strategies allow to
- Order rules
- Repeat rules
- Make choices
Strategy description

Different facets

- A strategy is a set of proof terms in rewriting logic
- A strategy is a $\rho$-term in the $\rho$-calculus and an abstract molecule in the biochemical rewriting calculus
- A strategy is a (higher-order) function that can apply to other strategies [BKMR02].
- A strategy is a subset of the set of all rewriting derivations [KKK08].
- A strategy is a partial function that associates to a reduction-in-progress, the possible next steps in the reduction sequence in sequential path-building games [DoughertyWRS09]
In $\lambda$-calculus, a strategy is a map $F$ from terms to terms such that $t \mapsto F(t)$ (Barendregt [Bar84]).

In abstract reduction systems (ARS), a strategy is a sub-ARS having the same set of normal forms (van Oostrom and de Vrijer [Ter03]).

In program transformation, a strategy is a plan for achieving a complex transformation using a set of rules. (Visser [Vis05]).

...
Different classes of strategies

Consider a syntactic structure (graph) $G$ to rewrite with $\mathcal{R}$. A strategy is a subset of derivations from $G$.

**Strategic rewriting** derivations are selected derivations.

An extensional strategy selects a subset of the set of all derivations (finite or not).

An intensional strategy is a partial function that associates to a reduction-in-progress, the possible next steps in the reduction sequence.

A positional strategy chooses the positions in the syntactic structure where a rule or a set of rules can be applied either by traversing the syntactic structure, or by using annotations to select a set of positions.
Graph rewriting strategies

Specific case of positions

- Where to apply a rule in a graph?
- Top-down or bottom-up traversals do not make sense.
- Need for a strategy language which includes operators to select rules and the positions where the rules are applied, and also to change the positions along the derivation.
Graph rewriting strategies

PORGY solution (Porgy2011):
A \textit{located graph} $G^Q_P$ consists of a graph $G$, a subgraph $P$ of $G$ called the 	extit{position subgraph} and a subgraph $Q$ of $G$ called the \textit{banned subgraph}.

- Rewriting must take place fully or partially in $P$.
- No rewriting can happen fully or partially in $Q$. 
Located graphs and rules
Strategy language

A *strategy language* gives syntactic means to describe strategies. Two main purposes:

- build derivation step and derivations
- operationaly compute the next acceptable strategic steps.

**Some strategy languages**

- ASF+SDF, Stratego, Strafunski [http://strategoxt.org/Stratego/WebHome](http://strategoxt.org/Stratego/WebHome)
- Porgy [tulip.labri.fr/TulipDrupal/?q=porgy](tulip.labri.fr/TulipDrupal/?q=porgy)
Strategy language

Constructs

- Primitives: rule, Identity, Failure
- (Non-)Determinism: all, one
- Composition: sequences of composable steps
- Selection of branches in the derivation tree (first, orelse, try)
- Conditional and tests
- Recursive strategies and iterations (repeat, while)
- Exploiting the structure of objects (traversals)
- Focusing (select or ban)
Strategy challenges

Strategies have been used in various domains. Which cross-fertilization can we get between them?

A few ideas of useful properties:

- Automated deduction: Fairness (no crucial rule will be postponed forever)
- Functional and Logic Programming: Loop-freeness,
- Game theory: connection with game theory strategies, strategies with memories,
Strategic Programming
Strategic programming

A *strategic rewrite program* consists of a finite set of rewrite rules $\mathcal{R}$, a strategy expression $S$ (built from $\mathcal{R}$ using a strategy language $\mathcal{L}(\mathcal{R})$) and a given structure $G$.

Denoted $[S_{\mathcal{R}}, G]$, or simply $[S, G]$. The strategy expression $S$ is used to decide which rewrite steps should be performed on $G$.

Operational semantics described by a transition relation on multisets of strategic programs (cf. [Porgy]).
Strategic programming challenges

- Confluence, termination of strategic programs
- Completeness w.r.t. normal forms: which (computable) strategies are guaranteed to find a normal form for any term whenever it exists?
- Rules with conditions or constraints:
  - operational termination (defined as the absence of infinite proof trees), studied in [LucasMM2015] for conditional term rewriting (CTRS) systems, is different from the notion of termination (absence of infinite reduction sequences).
  - irreducible terms and normal forms are also different for CTRSs [LucasM2014].
Porgy implementation

The Porgy environment features:

- Design port graphs and port rules and visualise them.
- Interactive application of a rule on a port graph.
- Creating and running a strategy.
- Exploration and analysis of a derivation tree.
  - Tooltips (get information)
  - Small multiples and animation (show the evolution of the graph)
  - Histograms (to follow graph parameter over rewriting operations)

Lesson: visualization is important at the level of data and graphs, at the rule level, at the strategy level!
Conclusion

Open questions and more challenges

- Add a strategy language on a rule environment
- Hierarchical modelling of data and processes: graphs, rules, strategies
- Visualization challenges: on all components
- Proving properties of strategies and strategic programs
- Verification: concurrent game structures, model checking,...
- Runtime verification
- Autonomic computing
Conclusion

Topics at the intersection of
- formal specifications and models of complex systems
- programming with massive data
- proof search and deduction
- game theory and verification

For
Formal Structures for Computation and Deduction
Thanks

The results presented here are based on joint works
- on Elan and Tom from 1990 to 2008,
- and on Porgy since 2008.

Thanks to
- the Protheo team
- the Porgy team
- and colleagues from Maude

for many inspiring discussions on these topics!