Implementation of a Port-graph Model for Finance

Nneka Ene

King's College London, UK

Abstract. In this paper we examine the process involved in the design and implementation of a port-graph model to be used for the analysis of an agent-based *rational negligence* model. Rational negligence describes the phenomenon that occurred during the financial crisis of 2008 whereby investors chose to trade asset-backed securities without performing independent evaluations of the underlying assets. This has contributed to motivating the search for more effective and transparent tools in the modelling of the capital markets.
In this paper I propose to use a visual declarative language, based on strategic port-graph rewriting, as a visual modelling tool to analyse an asset-backed securitisation market.

Keywords: graph rewriting systems, strategy languages, simulation, securitisation

1 Introduction

In this paper we examine the process involved in the design and implementation of a port-graph model to be used for the analysis of an agent-based *rational negligence* model. Rational negligence describes the phenomenon that occurred during the financial crisis of 2008 whereby investors chose to trade asset-backed securities without performing independent evaluations of the underlying assets. This proposal is motivated by the interest an analyst or policy-maker might have in analysing whether or not the purchase of a particular class of asset-backed security ought, going forward, to be subjected to a full due-diligence.

By replacing more traditional Dynamic Stochastic General Equilibrium models[1] with heterogenous, proactive, agent-based models able to produce more realistic representations, a system that supports rapid prototyping, is able to run system simulations, and, thanks to its formal semantics, also reason about system properties, can be produced. Turning to a declarative port-graph transformation system, able to model the dynamic behaviour of complex systems, given that its declarative nature and visual aspects produces a shorter distance between mental picture and implementation, facilitate the analysis of processes of interest. Such a tool is able to convert a black box model into a white box and also greatly provide a platform of extensive flexibility as shall be seen later.

[1] https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1468-0106.2012.00579.x

There are non-visual elements, such as an inbuilt strategy language, but details of the states produced by resulting strategy programs are highlighted in a visual trace/derivation tree.

In addition the rewrite rules that drive graph transformation systems are an intuitive and natural way of expressing dynamic, structural changes which are generally more difficult to model in traditional simulation approaches where the structure of the model is usually fixed [1]. Port-graphs are in addition particularly useful in the development of more concise graph models given support of both topology and data at the same time. Over many of the extant declarative languages used within agent-based simulation tools, this approach, given its visual declarative nature, provides more conceptualization support. The port-graph structure allows users to more easily describe node relationships, and given the aforementioned dual-natured support, port-graph transformation systems are able to offer a lot of flexibility in the structuring of rules, rule strategies and model starting positions to obtain desired results.

Contributions. We outline the typical specification of a graph program that can be used to represent the workings of a small agent-based system and include broad implementation details highlighting design choices and alternatives that could otherwise have guided our approach. We also carry out a thorough validation of the model with respect to its equational specification producing a base case that provides an effective platform for incrementally increasing the complexity and scope of the model. In our previous work; [2], [3] and [4], we have focused on the details of general approach (as opposed to our current evaluation of specification, design and implementation), a hierarchical port-graph extension and a more realistic model extension respectively.

Overview. This paper is organised as follows: We briefly examine the general structure of a simple Port-graph Transformation System and equational semantics of the chosen Rational Negligence model in Section 2. Details of actual implementation can be found in Section 3 and Section 4 examines the results of key tests and checks on the system. We finally conclude and briefly outline future plans in Section 5.

2 Background

2.1 Port-graph Transformation Systems

A port graph is a graph where nodes have explicit connection points, called ports, and edges are attached to ports. Nodes, ports and edges are labelled by a set of attributes describing properties such as rule side, colour, shape, etc. Port graphs are transformed by applying port graph rewrite rules. We refer to [6] for a formal definition of labelled port graphs, where labels are records, i.e., lists of pairs attribute-value. The values can be concrete (numbers, Booleans, etc.) or abstract (expressions in a term algebra, which may contain variables). A *port graph rewrite rule* $L \Rightarrow_C R$ can itself be seen as an attributed port graph consisting of two port graphs L and R together with an "arrow" node. The pattern, L, is used to identify subgraphs (redexes) in a given starting graph which should be replaced by an instance of the right-hand side, R, provided the condition C holds. The arrow node may itself have ports and edges that connect it to L and R; these edges outline a partial morphism between the ports in L and R, following the single push-out approach [7] to graph rewriting. More precisely:

Definition 1 (Port-graph Rewrite Rule). *[8] A port graph rewrite rule $L \Rightarrow_C R$ is a port graph consisting of two subgraphs L and R together with a node (called arrow node) that captures the correspondence between the ports of L and the ports of R, and includes a condition C that will be checked at matching time. More precisely, each of the ports in the arrow node has an attribute Type, which can have three different values: bridge, wire and blackhole, values that indicate how a rewriting step using this rule should affect the edges that connect the redex to the rest of the graph and that satisfy the following conditions:*

1. *A port of type bridge must have edges connecting it to L and to R (one edge to L and one or more to R).*
2. *A port of type blackhole must have edges connecting it only to L (at least one edge).*
3. *A port of type wire must have exactly two edges connecting to L and no edge connecting to R.*

Let X and Y be two port graphs over the same signature ∇. A port graph morphism $g\colon X \to Y$ maps nodes, ports and edges of X to those of Y such that the attachment of ports and the edge connections are preserved, and all attributes are preserved except for variables in X, which must be instantiated in Y. Intuitively, the morphism identifies a subgraph of Y that is equal to X except at positions where X has variables (at those positions Y could have any value).

Definition 2 (Port Graph Morphism). *[8] Given two labelled port graphs $X = (V_X, P_X, E_X, \mathcal{L}_X)$ and $Y = (V_Y, P_Y, E_Y, \mathcal{L}_Y)$ over the same signature ∇, a morphism f from X to Y, denoted $f : X \to Y$, is a family of injective functions $\langle f_V : V_X \to V_Y, f_p : P_X \to P_Y, f_E : E_X \to E_Y \rangle$ and instantiation functions $f_1 : X_{\mathcal{A}} \to \nabla_{\mathcal{A}}, f_2 : X_{\mathcal{V}} \to \nabla_{\mathcal{V}}$ such that:*

1. *$f_V : V_X \to V_Y$ is a mapping from the set of nodes of X to the set of nodes of Y such that if $n \in V_X then f_1(f_2(\mathcal{L}_X(n))) = \mathcal{L}_Y(f_V(n))$*
2. *$f_P : P_X \to P_Y$ is a mapping from the set of ports of X to the set of ports of Y such that if $p \in P_G$ then $f_1(f_2(\mathcal{L}_X(p))) = \mathcal{L}_Y(f_P(p))$ and $f_V(\mathcal{L}_X(p).Attach) = \mathcal{L}_Y(f_P(p)).Attach$*
3. *$f_E : E_X \to E_Y$ is a mapping from the set of edges of X to the set of edges of Y such that if $e \in E_G$ then $f_1(f_2(\mathcal{L}_X(e))) = \mathcal{L}_Y(f_E(e))$ and $f_P(\mathcal{L}_X(e).Connection) = \mathcal{L}_Y(f_E(e)).Connection$ (i.e., the morphism preserves the edge connections)*

We denote by $g(X)$ the subgraph of Y consisting of the set of nodes, ports and edges that are images of nodes, ports and edges in X. This definition ensures that each corresponding pair of nodes, ports and edges between X and Y have the same set of attribute labels and associated values, except at positions where there are variables. When using this definition to define matching, we will only allow the use of variable labels on one of the graphs: X will be the graph on the left-hand side of the rewrite rule, which may include variable labels, and Y will be the graph to be rewritten, without variables.

Let G be a port graph. A *rewrite step* $G \Rightarrow H$ via the port graph rewrite rule $L \Rightarrow_C R$ is obtained by replacing in G a subgraph $g(L)$ by $g(R)$, where g is a morphism from L to G satisfying C. More precisely:

Definition 3 (Match). *[8] Let $L \Rightarrow R$ be a port graph rewrite rule and G a port graph. We say a match $g(L)$ of the left-hand side (i.e., a redex) is found if: there is a port graph morphism g from L to G (hence $g(L)$ is a subgraph of G), C holds, and for each port in L that is not connected to the arrow node, its corresponding port in $g(L)$ must not be an extremity in the set of edges of $G - g(L)$. This last point ensures that ports in L that are not connected to the arrow node are mapped to ports in $g(L)$ that have no edges connecting them with ports outside the redex, thus ensuring that there will be no dangling edges when $g(L)$ is replaced by $g(R)$.*

For a given graph, several outcomes on application of a rule may be possible (due to the intrinsic non-determinism of rewriting). Strategies in rewriting systems are a means of controlling the creation of rewriting steps and improving rewriting opportunities. A sequence of rewriting steps is called a *derivation*. A *derivation tree* is a collection of derivations with a common root. Intuitively, the derivation tree is a representation of the possible evolutions of the system starting from a given initial state (each derivation provides a trace, which can be used to analyse and reason about the behaviour of system). In PORGY [8], the strategy language allows us to control the way derivations are generated. The strategy expression `setPos(crtGraph)` sets the position graph as the full current graph. If T is a rule, then the strategy $one(T)$ randomly selects one possible occurence of a match of rule T in the current graph G, which should superpose the position subgraph P but not superpose the banned subgraph Q. This strategy fails if the rule cannot be applied. Id and $Fail$ denote success and failure, respectively. The strategy expression $match(T)$ is used to check if the rule T can be applied (i.e., if there is a match for the left hand side of the rule in the current graph) but does not apply the rule. $(S)orelse(S')$ tries strategy S and if it fails then tries to apply S'. If both strategies fail then the whole statement fails. $ppick(T_1, \ldots, T_n, \Pi)$ selects one of the transformations $T_1, \ldots T_n$ according to the given probability distribution Π. $while(S)[(n)]do(S')$ executes strategy S' (not exceeding n iterations if the optional parameter n is specified) while S succeeds. $repeat(S)[max\ n]$ repeatedly executes a strategy S, not exceeding n times. It can never fail (when S fails, it returns Id).

2.2 The Rational Negligence Model.

As defined in [9] "Securitisation is the process of converting cash flows arising from underlying assets or debts/receivables (typically illiquid such as corporate loans, mortgages, car loans and credit cards receivables) due to the originator into a smoothed liquid marketable repayment stream" and this ensures that the originator can raise asset-backed finance through loans or the issuance of debt securities also known as assets. An originator is any financial intermediary with a portfolio of assets on its balance sheet. In a securitisation, assets are selected, pooled and transferred to a tax neutral, liquidation-efficient (i.e bankruptcy avoiding), Special Purpose Vehicle (SPV), who funds them by issuing securities.

In the core rational negligence model [10], the profit $\mathcal{U}_w$ expected by an agent (e.g., a bank) w from trading an asset depends on whether or not w follows the *negligence rule*, i.e., the rule of not performing independent risk assessment. Let z be a binary variable indicating whether or not the agent is following the negligence rule, then $\mathcal{U}_w(z)$ can be characterised by the following equations, where p is the probability of asset toxicity, Z is the average of all z's in the domain, c is the cost of purchasing an asset (note that the payoff from successfully reselling the asset is normalised to unity), x_w is the cost of performing a complete risk analysis, k is the number of trading partners of the seller bank and $\mathcal{N}_i$ is the set of agents.

- Expected profit for w when following the negligence rule, i.e., when $z(w) = 1$, if w buys an asset and then tries to sell it to w':

  $$\mathcal{U}_w(1) = -p(1 - z(w'))c + [1 - p(1 - z(w'))](1 - c) = 1 - p(1 - Z)c$$

  This is because if the asset is toxic then w will loose c if w' checks, and will have a profit of $1 - c$ if w' does not check. Of course w does not know a priori whether w' will or not follow the rule, but it can estimate $z(w')$ as the average of all the values of z in the system, Z. Note that when $p = 0$ the profit is 1 as expected.
- Similarly, the expected profit for w when the rule is not followed, i.e., $z(w) = 0$, is defined by:

  $$\mathcal{U}_w(0) = (1 - p)(1 - c) - x_w$$

  This is because if the asset is toxic, then w will not buy it (losing only x_w), but if it is not toxic then it will resell it with a profit of $1 - c - x_w$.

So the best response of agent w to a buying request is determined by:

$$\mathcal{U}(1) - \mathcal{U}(0) = p(Z - c) + x_w = p\left(\frac{1}{k}\sum_{j \in \mathcal{N}i} z_j - c\right) + x_w$$

Following [10], we implement a model that mimicks the transactions that follow the trading of one asset since this is sufficient to perform validations against equivalent DSGE analyses. The goal of the model is to study the evolution of the system till *fixed point* or *stable state* is reached i.e., in this case, a state such that all potential buyers in the universe of discourse no longer alternate between diligent and non-diligent behaviour in their handling of the purchase of a particular asset.

3 Implementation

Following a design process similar to that outlined in [11], we chose to model asset-transfer transactions using a combination of global and local data, and a global state, the Z node (an indicator of market behaviour obtained as the average value of each individual bank's approach, represented by the local lower-case attribute z and not to be confused with the global value Z). A *Change* indicator node is used within a rule to detect whether the market has reached a stable state. Tables 2 and 3 contain a description of the nodes used. Alternative designs are possible, highlighting the flexibility of the approach: For example, local copies of the z-attribute could be used to propagate negligent/diligent behaviour using propagation algorithms borrowed from social network models whereby information is transmitted, or in this case received, based on the actions of neighbours or neighbours-of-neighbours or other clusters [12]. The details of this alternative model shall be contained in the full paper and we adhere to current design in order to arrive at a base case that most accurately matches the details of that provided in the equational model.

We represent the full ABS universe hierarchically as several initial graphs. Port graph rewriting rules and strategies are used to control the step-wise evolution of the graphs and to create a derivation tree that can be used for plotting and analysing parameter values. The asset trading model sits at the top level of the model hierarchy and it is non-deterministic in nature. Below this system, also able to handle asset pricing and valuation issues, lie several deterministic subsystems that model origination, structuring of the deal, SPV transfers and profitability of the sale.

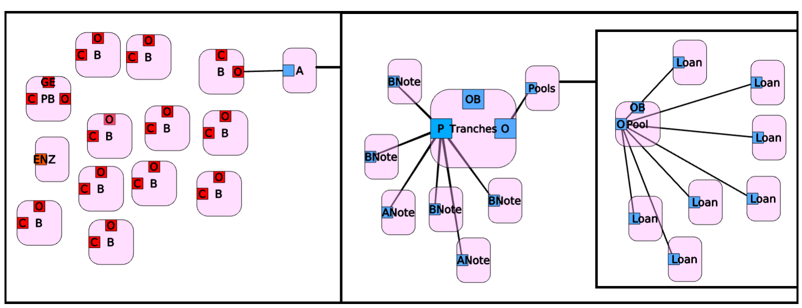


Fig. 1. *All Tiers Flattened and Condensed*

Tables 2 and 3 describe the nodes in the system and their ports respectively:

After the creation of a comprehensive set of rules, reduction strategies were created that defined the sub-graphs to be selected for evaluation and which rules should be applied to the starting state of the model and it is from this point that the derivation tree begins to undergo construction as the execution strategy calls

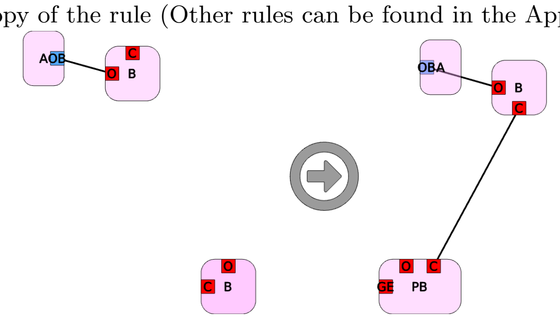
Name of Rule	Description
requesttobuy	Sends a request-to-buy message to a random bank B changing the name of this node to PB ($\mathit{PotentialBuyer}$) A copy of the rule (Other rules can be found in the Appendix) 
beginanalysis	Computes profitability $\mathcal{U}(1)$, $\mathcal{U}(0)$ of PB, generating a node *Theta* with attribute $\mathit{DeltaU1U0} = \mathcal{U}(1) - \mathcal{U}(0)$
updatez	Updates the attribute Z in node Z. The new value in Z is $(Z * (k-1) + z(PB))/k$
followresult	Applies if $\mathit{DeltaU1U0} \geq 0$. As additional visualisation support, it generates a *follow* node if more profitable to not do a full risk analysis
deviationresult	Applies if $\mathit{DeltaU1U0} < 0$ As additional visualisation support, it generates a *deviation* node if more profitable to do a full risk analysis
followdecision	Transfers asset and prepares for a new transaction (i.e. cleans up after the decision negligence rule), updating bank's attribute z, updating the *Change* counter if necessary
deviationdecision	Transfers asset and prepares for a new transaction (i.e. cleans up after the decision to deviate from the negligence rule), updating bank's attribute z, updating the *Change* counter if necessary
change	Sets the *Change* counter back to 0 if greater than 0

Table 1. *Rewrite Rules*

Entity	Attribute	Description
Bank/Potential Buyer (B/PB)	Payoff (payoff)	Returns from re-selling an asset
	z	Indicates whether or not, as a rule, the institution performs independent risk analyses
	Bank ID (b_id)	Bank identifier
Asset	Current Value (c_val)	Cost of purchasing an asset
	Probability of Toxicity (p_tox)	An asset is toxic if the borrowers of the underlying loans are likely to default or are in default
	Actualised Toxicity (a_tox)	Current toxicity level
	Perception (pe)	External rating of the asset by rating agencies
	Due Diligence Cost (ddcost)	Full cost of an independent risk assessment
Change	change	Change in bank approach
	Sum of change (sumofchange)	Sums all changes in a current cycle
Z	z	Represents the global average z
	Number of Iterations (numofiterations)	Counter that keeps track of AllTrade iterations
	Number of Agents (numofagents)	Variable that keeps track of number of banks
Theta	U1	Profitability of being negligent
	U0	Profitability of being diligent
	DeltaU1U0	Difference between U1 and U0

Table 2. *Nodes and Attributes*

Entity	Ports	Description
Bank	O (Owns)	Edges attached to this port highlight assets owned by the bank
	C (Contacts)	Communication channel with another bank
Asset	OB (Owned_by)	Connects the asset to its current owner
Z	EN (Environment)	Global entity that tracks current average sentiment
PotentialBuyer	O (Owns)	Links to assets owned by the bank
	C (Contacts)	Communication channel with another bank
	GE (Generates)	Declares a relationship with an analysis node
Change	CH (change)	Counter that keeps track of behaviour changes
Theta	PB (Produced_by)	Entity that produces this computation helper

Table 3. *Ports in each kind of node*

```
#AllTrade#;
while(match(change))do(
  one(change);
  #AllTrade#)
```

Strategy 1: *FixedPointSearch*

```
setPos(crtGraph);
repeat(one(requesttobuy);
  one(beginanalysis);
  (one(deviationresult);one(deviationdecision)) orelse
  (one(followresult);one(followdecision))
  setPos(crtGraph);
  one(updatez))(k)
```

Strategy 2: *AllTrade*

on rules that create step-wise transformations. Specifically, the asset transfer processes are governed by the strategies *AllTrade* and *FixedPointSearch* (see Strategies 1 and 2 below), using 8 rewrite rules summarised in Table 1 (see also the diagrams in Figures 3 to 10 of the Appendix, omitted here due to space constraints).

Also highlighting flexibility it is worth noting that a variant of strategy *AllTrade* can simply replace the `orelse` operator by a `ppick` operator, and then begin to model probabilistic choice of logit type between following or deviating from the negligence rule. The probability distribution used in this case implements the stochastic "trembles" described in [14] and can be written within our strategy environment as follows:

```
ppick(followResult, deviationResult, udfLogitModel)
```

where `udfLogitModel` is a function that reads the profitability of being negligent or diligent (attributes U1 and U0 in the node Theta of the graph produced by the relevant rule) and returns the following values as a list:

$$\frac{\exp^{\mathcal{B}U_i(z=1)}}{\exp^{\mathcal{B}U_i(z=1)} + \exp^{\mathcal{B}U_i(z=0)}} \quad and \quad 1 - \left(\frac{\exp^{\mathcal{B}U_i(z=1)}}{\exp^{\mathcal{B}U_i(z=1)} + \exp^{\mathcal{B}U_i(z=0)}}\right)$$

where i is the current agent number and $\mathcal{B}$ is the intensity of choice parameter that controls the ease by which fixed point is reached (as specified in [14]).

4 Testing

A successful base case validation has seen test results (see Figure 2 where average Z count value is plotted versus depth of the simulation) line up with results from the more traditional ABM simulation given in [10]). In particular, for high values of p (that is, high probability of toxicity), we observe the expected result when the initial state contains a mixture of negligent and diligent agents: a sharp drop in Z, corresponding to a sharp switch in average approach (i.e., more banks decide to perform independent analysis), which in turn will generate stability. An illustration of this can be seen in Figure 2(c) and notice that given high due diligence costs Figures 2(b) and 2(e) highlight a negligent approach whereas Figures 2(c) and 2(h) reflect the favouring of a diligent approach. However, even for high toxicity, if the initial state is a set of negligent agents, the model reaches equilibrium without switching approach as seen in Figure 2(l).

5 Some Related Work and Conclusion

Fundamentally, general purpose agent-based simulation tools and platforms[2] like JAS, Netlogo, AgentBuilder, Swarm, MASON, Repast, SeSAm, GAMA and INGENIAS Development Kit, support an imperative object-oriented approach to model development, facilitating the modular approach to coding. EMERALD and JADE middleware integrate a declarative approach but without any visualization support. Other tools and languages like Stratego, Maude and ELAN [5] support a pure term-writing approach which in the case of Maude is augmented by probabilistic features. The visual, declarative nature of graph transformation systems are thus welcome in the cases where users seek to primarily focus on describing what the system should accomplish in terms of problem domain versus the how, and maintain strong conceptualization support that can subsume the details of spatial and topological constraints.

We have shown that strategic port-graph rewriting provides a basis for the design and implementation of a multi-level graph model able to capture the inner workings of the sub-prime secondary securitisation market in a manner that reflects the aforementioned rational negligence phenomenon and that provides optional operational support.

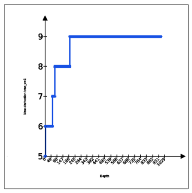
We observed that a declarative approach is much easier to program and maintain, and the incremental manner in which development was approached (e.g. coarse-grained rules tested before finer optimizations), in addition to the modular nature of development, eliminated many coding bugs. Interacting with the system was not convoluted in anyway and being able to view the states generated by each rule within the derivation tree generated was useful.

In future, we hope to further develop the hierarchical model to be able to capture all details of the full securitisation life-cycle and cater for more dynamically to changing parameters.

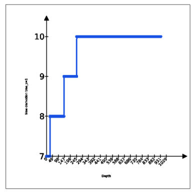
References

1. de Lara, J., Guerra, E., Boronat, A., Heckel, R., Torrini, P.: Domain-specific discrete event modelling and simulation using graph transformation. Software and System Modeling **13**(1) (2014) 209–238
2. Nneka Ene, M.F., Pinaud, B.: A graph transformation approach to the modelling of capital markets. Submitted for publication.

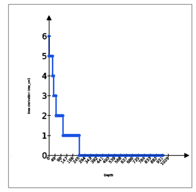
[2] `http://jasss.soc.surrey.ac.uk/18/1/11.html`



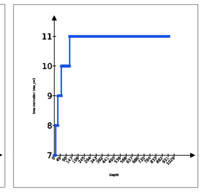
(a) Low Toxicity, High Due Diligence Cost,Mixture of Diligent and Negligent Banks



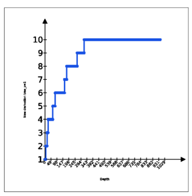
(b) High Toxicity, High Due Diligence Cost, Mixture of Diligent and Negligent Banks



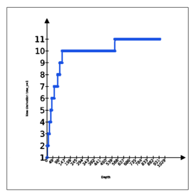
(c) High Toxicity, Low Due Diligence Cost, Mixture of Diligent and Negligent Banks



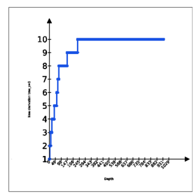
(d) Low Toxicity, Low Due Diligence Cost, Mixture of Diligent and Negligent Banks



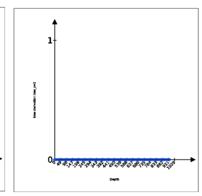
(e) High Toxicity, High Due Diligence Cost, Diligent Banks



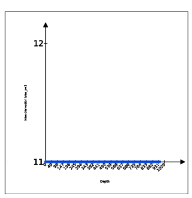
(f) Low Toxicity, High Due Diligence Cost, Diligent Banks



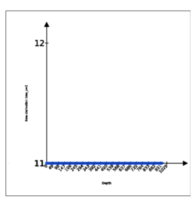
(g) Low Toxicity, Low Due Diligence Cost, Diligent Banks



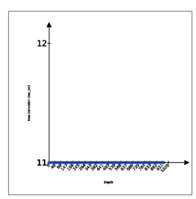
(h) High Toxicity, Low Due Diligence Cost, Diligent Banks



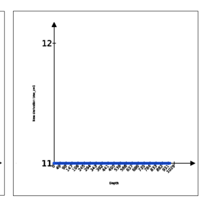
(i) Low Toxicity, High Due Diligence Cost, Negligent Banks



(j) High Toxicity, High Due Diligence Cost, Negligent Banks



(k) Low Toxicity, Low Due Diligence Cost, Negligent Banks



(l) High Toxicity, Low Due Diligence Cost, Negligent Banks

Fig. 2. *Experiment Results.* (y-axis: Count of the number of negligent banks. The intersection of x and y axes in the case of a starting universe of purely diligent banks corresponds to the co-ordinates (0,0) as opposed to (11,0) in the case where we begin with negligent banks. Curves tending upwards reflect a negligent equilibrium result)

3. Ene, N.C., Fernández, M., Pinaud, B.: Attributed hierarchical port graphs and applications. In: Proceedings Fourth International Workshop on Rewriting Techniques for Program Transformations and Evaluation, WPTE@FSCD 2017, Oxford, UK, 8th September 2017. (2017) 2–19
4. Ene, N.C.: Analysis of attributed hierarchical graphs. Submitted for publication.
5. Mart-Oliet, N., Meseguer, J., Verdejo, A.: Towards a strategy language for maude. Electronic Notes in Theoretical Computer Science **117** (2005) 417 – 441 Proceedings of the Fifth International Workshop on Rewriting Logic and Its Applications (WRLA 2004).
6. Fernández, M., Kirchner, H., Pinaud, B.: Strategic Port Graph Rewriting: An Interactive Modelling and Analysis Framework. Research report, Inria (January 2016)
7. Löwe, M.: Algebraic approach to single-pushout graph transformation. Theor. Comput. Sci. **109**(1&2) (1993) 181–224
8. Fernández, M., Kirchner, H., Pinaud, B.: Strategic Port Graph Rewriting: an Interactive Modelling Framework. Research report (2017) https://hal.inria.fr/hal-01251871.
9. Markose, S., Dong, Y., Oluwasegun, B.: An multi-agent model of rmbs, credit risk transfer in banks and financial stability: Implications of the subprime crisis (2008)
10. Anand, K., Kirman, A., Marsili, M.: Epidemics of rules, rational negligence and market crashes. The European Journal of Finance **19**(5) (2013) 438–447
11. Depke, R., Heckel, R., Küster, J.M.: Formal agent-oriented modeling with {UML} and graph transformation. Science of Computer Programming **44**(2) (2002) 229 – 252 Special Issue on Applications of Graph Transformations (GRATRA 2000).
12. Vallet, J., Kirchner, H., Pinaud, B., Melançon, G.: A visual analytics approach to compare propagation models in social networks. In Rensink, A., Zambon, E., eds.: Proc. Graphs as Models, GaM 2015. Volume 181 of EPTCS. (2015) 65–79
13. Norling, E., Powell, C.R., Edmonds, B.: Cross-disciplinary views on modelling complex systems. In: Multi-Agent-Based Simulation IX, International Workshop, MABS 2008, Estoril, Portugal, May 12-13, 2008, Revised Selected Papers. (2008) 183–194
14. Farmer, J., Gallegati, M., Hommes, C., Kirman, A., Ormerod, P., Cincotti, S., Sanchez, A., Helbing, D.: A complex systems approach to constructing better models for managing financial markets and the economy. The European Physical Journal Special Topics **214**(1) (2012) 295–324

A Appendix

A.1 Rewrite Rules

Diagrams in Figures 3 to 10 provide overviews on the rules described in the main section without the red arrow-node edges in order to achieve a more user-friendly viewing. Displaying the red arrow-node edges is optional in PORGY.

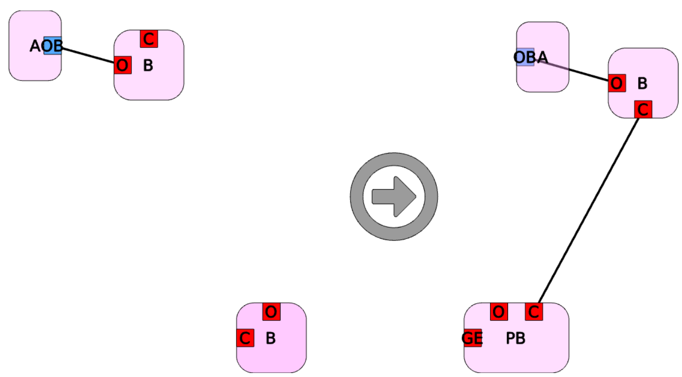


Fig. 3. *Request to Buy.* Sends a request-to-buy message to a random bank B changing the name of this node to PB (PotentialBuyer)

Algorithm Tab:
$Theta.U1 = 1 - (A.p\_tox(1 - Z.z)A.c\_val)$
$Theta.U0 = (1 - A.p\_tox)(1 - A.c\_val) - A.ddcost$
$Theta.DeltaU1U0 = Theta.U1 - Theta.U0$

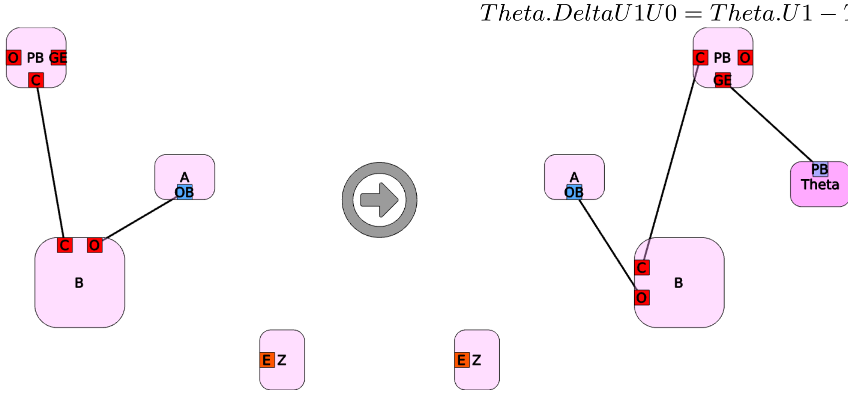


Fig. 4. *Begin Analysis.* Computes profitability U1, U0 of PB, generating a node Theta with attribute DeltaU1U0 = U1 - U0

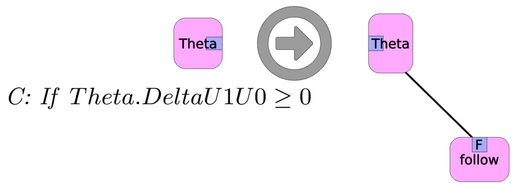


Fig. 5. *Follow Result.* Applies if DeltaU1U0 ≥ 0. As additional visualisation support, it generates a follow node if more profitable to not do a full risk analysis

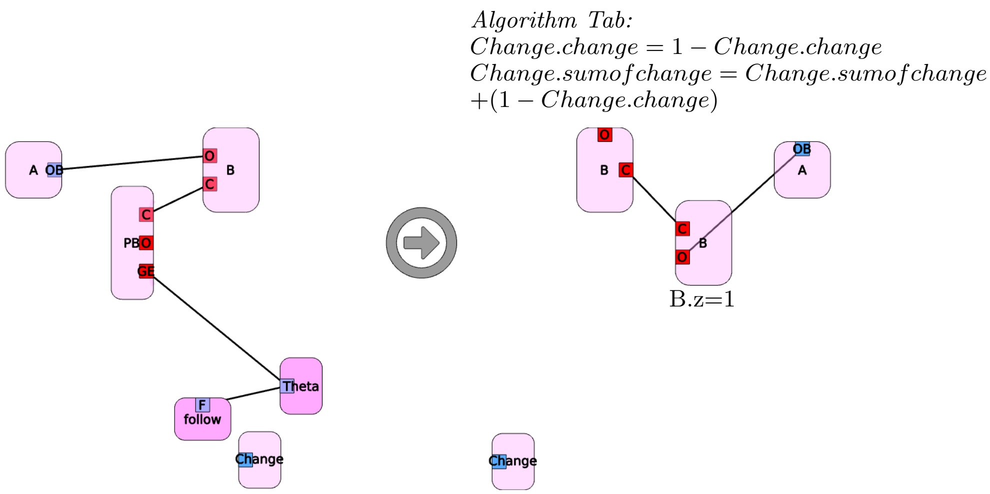


Fig. 6. *Follow Decision.* Transfers asset and prepares for a new transaction (i.e. cleans up after the decision to follow the negligence rule), updating bank's attribute z, updating the Change counter if necessary.

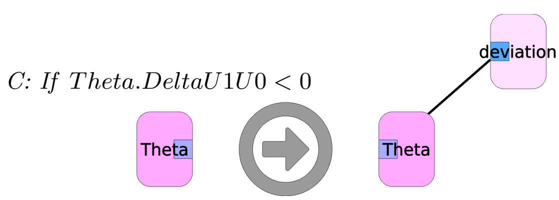


Fig. 7. *Deviation Result.* Applies if DeltaU1U0 < 0. As additional visualisation support, it generates a deviation node if more profitable to do a full risk analysis

Algorithm Tab:
$Change.change = 1 - Change.change$
$Change.sumofchange = Change.sumofchange$
$+(1 - Change.change)$

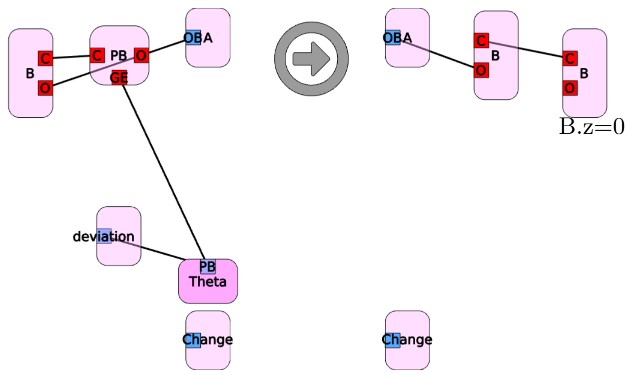


Fig. 8. *Deviation Decision.* Transfers asset and prepares for a new transaction (i.e. cleans up after the decision to deviate from the negligence rule), updating bank's attribute z, updating the Change counter if necessary

Algorithm Tab:
$Z.z =$
$((Z.z * (Z.numofagents - 1)) + B.z)$
$/Z.numofagents$

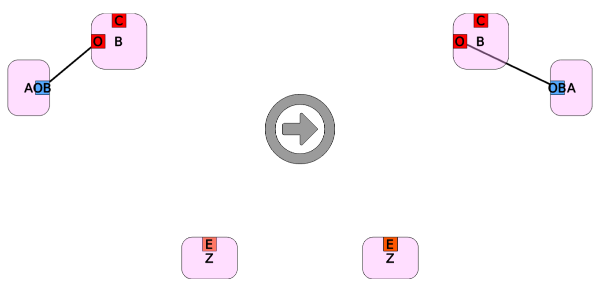
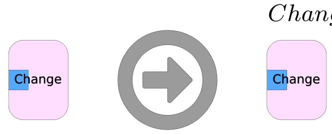


Fig. 9. *Update Z.* Updates the attribute Z in node Z. The new value in Z is (Z * (k - 1) + z(PB))/k

Algorithm Tab:
$Change.change = 0$
$Change.sumofchange = 0$



C: If $Change.sumofchange \mathrel{!=} 0$

Fig. 10. *Change.* Sets the Change counter back to 0 if greater than 0