# A Graph Transformation Approach to the Modelling of Capital Markets

Nneka Ene[1], Maribel Fernández[1] and Bruno Pinaud[2]

[1] King's College London, UK
[2] University of Bordeaux, France

**Abstract.** We propose to use strategic port-graph rewriting as a visual modelling tool to analyse credit derivative markets. We illustrate the approach by specifying a basic "rational negligence" model in which investors may choose to trade securities without performing independent evaluations of the underlying assets. We show that our model is correct with respect to the equational model and can be used to simulate simple market behaviours. The model has been implemented within PORGY, a graph-based specification and simulation environment.

**Keywords:** graph rewriting, strategies, simulation, securitisation

## 1 Introduction

The sub-prime mortgage crisis of 2008 has heightened the need for more effective and transparent tools in the modelling of capital markets, as noted in [1]. In the Asset-Backed Securitisation space, *rational negligence* [2] has been identified as a behavioural pattern that led to weakening in the market: to reduce operational costs, transactions were performed without *proper due-diligence* checks. Ratings from credit agencies were found to be inaccurate especially in terms of underestimated default probabilities. This inaccuracy led to system states the DSGE (Dynamic Stochastic General Equilibrium) models[3] were unable to anticipate.

As an alternative to traditional top-down macro equilibrium models, Agent-Based Models (ABM) have been proposed, which examine behaviour at a micro-level [3]. In this paper we explore another micro-level approach: we seek to formalise the rational negligence theory using *graph rewriting*. We provide an example of application to illustrate the ideas. Our small example cannot prevent a future crisis but can be seen as a first step towards the development of alternative tools for the analysis of markets, which complement the current agent-based implementations.

Rewrite rules 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 [4]. Graph rewriting languages are well-suited to the study of the dynamic behaviour of

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

complex systems: their declarative nature and visual aspects facilitate the analysis of the processes of interest producing a shorter distance between mental picture and implementation; they can be used for rapid prototyping, to run system simulations, and, thanks to their formal semantics, also to reason about system properties.

We use *attributed port graphs*, that is, graphs where edges are connected to nodes at specific points called ports, and where attributes are attached to ports, nodes and edges. Attributed port graphs are useful in the development of graph models, due to their support of both topology (via ports and edges) and data (via attributes). To control the rewriting process, we use *strategies* that permit to select which rules to apply and where, including probabilistic rule applications. We present first a basic model of asset trading following a discretised equational model presented in [5], where the probability of asset toxicity, due diligence analysis cost and asset cost are fixed. We then briefly discuss a more general version of the model where stochasticity is introduced by using a probabilistic choice model of logit type.

*Summary of Contributions.* We identify rules and strategies that model basic asset-trading transactions, taking into account the rational negligence phenomenon [2, 5]. The model has been implemented in PORGY[4], an interactive, visual port graph rewriting tool. The graph rewriting approach we advocate produces flexible models that are easy to validate, experiment with and reason about. We illustrate it by showing the correctness of our graph rewrite rules and strategies with respect to the equations defining the rational negligence phenomenon, and using the implemented model to analyse simple market behaviours.

*Overview.* We first recall key notions on securitisation and graph rewriting in Section 2. Section 3 describes the proposed approach to the modelling of securitisation, including a short description of rules and associated strategies. Section 4 examines key properties of the model (formal definitions and proofs are omitted due to space restrictions). We finally conclude, discuss related work and briefly outline future plans in Section 5.

## 2 Background

### 2.1 Asset-Backed Securities

As defined in [1] "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". We now examine informally some of the entities and processes that characterise this space [5]. *Assets* represent loans to clients or obligors who make regular installment payments to

[4] `http://porgy.labri.fr`

the originator to clear their debts. 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 general, an ABS (*asset-backed security*), or simply asset if there is no ambiguity, is any securitisation issue backed by consumer loans, car loans, credit cards, etc.

In the core rational negligence model [2], 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 $\mathcal{U}_w$ is a function of $z$. According to [2], $\mathcal{U}_w(z)$ 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 $w'$:

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

  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 will 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 the value of $\mathcal{U}(1) - \mathcal{U}(0)$. If it is positive, then negligence is better, otherwise diligence is better. Note that

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

Following [2], in this paper we study the behaviour produced by the trading of one asset since this is sufficient to perform validations against equivalent DSGE analyses. The goal 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 negligent behaviour in their handling of the purchase of a particular asset.

### 2.2 Port Graph Rewriting

There are many different kinds of graph transformation systems. In this paper we focus on labelled port graphs [6], which have been used in various domains (see, e.g., [7, 8] for applications in biochemistry and social networks). Due to space restrictions, in this section we briefly recall the main notions of port graph rewriting and refer the reader to [6] for formal definitions and examples.

Intuitively 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, including a mandatory attribute *Name* that characterises the type of the node, port or edge. For example, in our model we have nodes named $B$ (representing banks), $A$ (representing an asset), etc. In general, attributes describe properties of nodes, ports and edges.

Port graphs are transformed by applying port graph rewrite rules. The system we use to implement our port graph rewriting system is PORGY [6], where labels are records, i.e., lists of attribute-value pairs. The values can be concrete (numbers, Booleans, etc.) or abstract (expressions in a term algebra, which may contain variables).

The port graph in Figure 1 depicts a toy ABS secondary market universe represented by a community of banks ($B$), one of which owns a tradeable asset ($A$), together with a global environment represented by the nodes $Z$ and Change. The edge between $A$ and $B$ represents ownership, it originates from a port in $B$ called $O$ (meaning that $B$ owns $A$). Transactions between banks, representing buy-sell requests and associated operations are specified by means of rewrite rules.

A *port graph rewrite rule* $L \Rightarrow_C R$ can itself be seen as a port graph consisting of two port graphs $L$ and $R$ together with an "arrow" node. Intuitively, the pattern, $L$, is used to identify subgraphs (redexes) in a given 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 specify a partial morphism between the ports in $L$ and $R$, following the single push-out approach [9] to graph rewriting (see [6] for more details).

Figure 2 displays one of the rules used in our model and implemented in PORGY. The arrow-node edges (depicted in red in the diagram) are used during rewriting to redirect edges that arrive to ports in the redex from outside: these edges are redirected towards the corresponding ports in the copy of the right-hand side that will replace the redex, ensuring that no edges are left dangling.

In PORGY, we can specify how the attributes are updated in the right-hand side of the rule by means of an "algorithm tab". As an example, we show in Figure 3 the tab associated with the *beginanalysis* rule detailed in Table 3 (see also the Appendix). The algorithm tab permits the definition of the values of new attributes, possibly using attributes of left-hand side elements. For example, the node Theta created in the right-hand side of rule *beginanalysis* has attributes U1, U0 and DeltaU1U0, which are updated as indicated in Figure 3, using attributes of the nodes $A$ and $Z$ in the left-hand side of the rule.

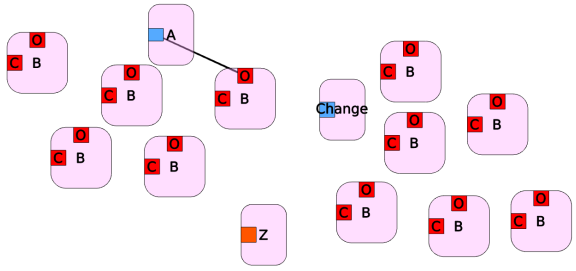



**Fig. 1.** *Sample Port-graph: Model's Starting Graph*

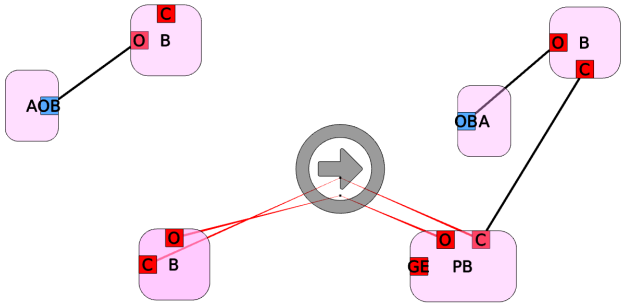



**Fig. 2.** *Sample Rule*

For a given graph, several different rewriting steps 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. 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).

PORGY's strategy language allows us to control the way derivations are generated. We can specify not only the rule to be used in a rewriting step, but also the position where the rule should (or should not) be applied. For the latter, PORGY implements *located graphs*, which are port-graphs with two distinguished sub-

```
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. 3.** Sample Algorithm Tab Script

graphs P (Position subgraph, the focus of rewriting) and Q (Banned subgraph, where rewriting steps are forbidden). The keywords `crtGraph`, `crtPos`, `crtBan` in the strategy language denote, respectively the current graph being rewritten and its Position and Banned subgraphs. For example, the strategy expression `setPos(crtGraph)` sets the position graph as the full current graph, so rules can apply anywhere. 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 $\Pi$. $while(S)[(n)]do(S')$ 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 $Id$).

PORGY [6] offers an in-built strategy editor, a navigable derivation tree widget, and widgets for the creation of rules and graphs (see Figures 4 and 5). By navigating on the tree and zooming on different nodes, we can see the various stages in the simulation; if we click on the black arrows in the derivation tree we can see which rule has been applied and identify the cause of the change in the model state.

## 3 The ABS-GTS Model

In this section we give a high-level description of a graph-based model of the ABS process as specified by the equations given in Section 2.1. Asset-transfer transactions are modelled using a combination of global and local data, as explained above: the global state includes $Z$ (an indicator of market behaviour obtained as the average value of each individual bank's approach, represented by the local variable $z$ and not to be confused with the global value $Z$) and a $Change$ indicator, to detect whether the market has reached a stable state. See Tables 1 and 2 for a description of the nodes used.

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. The asset trading model sits at the top level of the model hierarchy. It is non-deterministic in nature. Below this system, also able to handle asset pricing and valuation issues, lie several subsystems that model origination, structuring of the deal, SPV transfers and profitability of the sale, and therefore aid in enforcing internal checks. Alternative designs are possible, for example, banks could be organised in clusters, with local copies of $Z$, using algorithms borrowed from social networks [8] to model propagation of negligent/diligent

| 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 1.** *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 2.** *Ports in each kind of node*

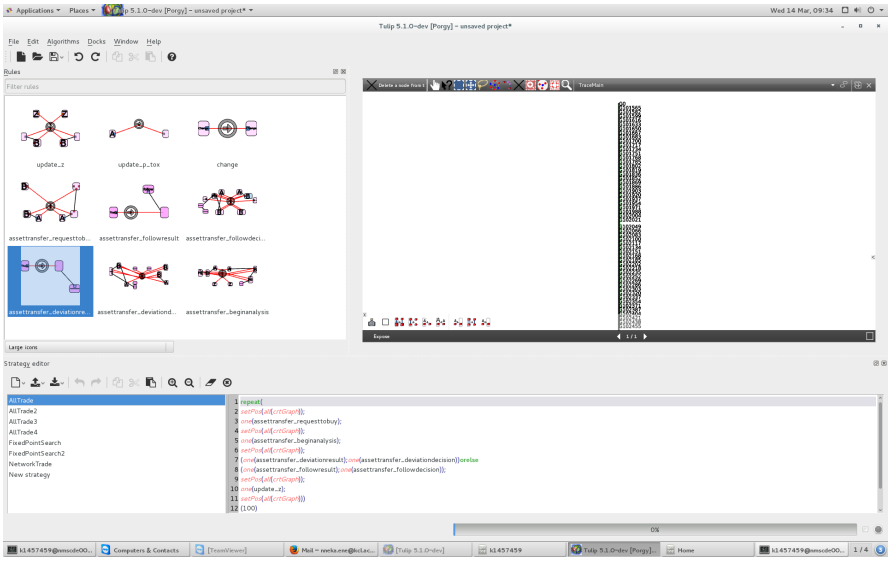


**Fig. 4.** *PORGY in action:* A portion of a branch in a derivation tree (top right window). Bottom window: strategy widget, left window: rules widget containing all the rules that drive the system. The diagram has been minimized in keeping with space restrictions. A zoom-in on the derivation tree can be found in Figure 5. Sample strategies can be seen in Section 3 as with descriptors associated with the rule thumbnails in the rules widget

behaviour. The exploration of these models is left for future work. In the rest of the paper we focus on the top tier level, which is where the 'rational negligence' phenomenon can be observed.

Reduction strategies define sub-graphs to be selected for evaluation and which rules should be applied. The starting state of the model is the graph shown in Figure 1 and it is from this point that the derivation tree begins to undergo construction as the execution strategy calls 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 3 (see also the diagrams in Figures 7 to 14 of the Appendix, omitted here due to space constraints).

A basic description of the strategy *AllTrade* is as follows: Line 1 specifies that rules will apply anywhere in the current graph. Line 2 starts a trading cycle: each iteration corresponds to one transaction (asset transfer operation), with the rules described in Table 3: the owner of the asset offers the asset to another bank (rule *requesttobuy*; the strategy operator `one` that controls the application of the rule *requesttobuy* ensures fair, nondeterministic choice of buyer); the potential buyer then begins the analysis in line 3 to decide whether or not to follow the negligence rule. It does this by computing the profitability of choices as described in Section 2.1 using rule *beginanalysis*. If diligence is more profitable the deviation

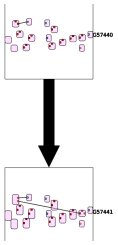


**Fig. 5.** *PORGY in action:* A zoom-in on the derivation tree in Figure 4. The square boxes are nodes in the derivation tree: they contain graphs, and the black arrow represents the application of a rewrite rule.

```
1 #AllTrade#;
2 while(match(change))do(
3   one(change);
4   #AllTrade#)
```

**Strategy 1:** *FixedPointSearch*

rules will apply, otherwise the bank follows the negligence rule (see the `orelse` in lines 4 and 5). The rule *updatez* used in Line 7 updates the global $Z$ as described in Table 3. We repeat $k$ times in order to give all banks an opportunity to trade. Strategy 1 controls the full execution: *AllTrade* is iterated until there are no changes in the agent behaviours (i.e., as long as the *change* rule can be applied).

A variant of strategy *AllTrade* replaces the `orelse` operator by a `ppick` operator, 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 [3] 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:

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

where $i$ is the current agent number and $\mathcal{B}$ is the intensity of choice parameter that controls the ease at which fixed point is reached (as specified in [3]). Levels of toxicity, asset value and due diligence cost are parameters of the simulation, which can be easily changed in our model by updating values in the attributes of bank and asset nodes.

| Name of Rule | Description |
|---|---|
| requesttobuy | Sends a request-to-buy message to a random bank $B$ changing the name of this node to $PB$ ($PotentialBuyer$) |
| beginanalysis | Computes profitability $\mathcal{U}(1), \mathcal{U}(0)$ of $PB$, generating a node *Theta* with attribute $DeltaU1U0 = \mathcal{U}(1) - \mathcal{U}(0)$ |
| updatez | Updates the attribute $Z$ in node Z. The new value in $Z$ is $(Z * (k-1) + z(PB))/k$ |
| followresult | Applies if $DeltaU1U0 \geq 0$. As additional visualisation support, it generates a *follow* node if more profitable to not do a full risk analysis. |
| deviationresult | Applies if $DeltaU1U0 < 0$. 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 to follow the 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 3.** *Rewrite Rules*

```
1 setPos(crtGraph);
2 repeat(one(requesttobuy);
3   one(beginanalysis);
4   (one(deviationresult);one(deviationdecision)) orelse
5   (one(followresult);one(followdecision))
6   setPos(crtGraph);
7   one(updatez))(k)
```

**Strategy 2:** *AllTrade*

## 4 Model Properties

Following a base case validation in which test results (see Figure 6 where average Z value is plotted versus depth of the simulation) line up with results from a traditional ABM simulation given in [2], we examine parameters and parameter values that simulate different economic scenarios. A natural question arises: What events could have mitigated or further instigated the 2008 crisis? By increasing toxicity values for example we can take into account the increase in interest rates that led to increased default rates and the 2008 crisis. Our experiments show that when toxicity is increased (attribute $p$ in node $A$) the system reaches a stable state where all banks perform independent risk analysis, as expected.

First, we show that our implementation is correct with respect to the equational semantics given in Section 2.1. This ensures that our model captures the ABS process of interest, and predictions from the ABS models under the same conditions coincide with the predictions produced by our system. Proofs are omitted due to space constraints and can be found in the Appendix.

**Lemma 1.** *Strategies 1 (FixedPointSearch) and 2 (AllTrade) never fail*[5]*.*

**Theorem 1 (Correctness).** *The graph-based model defined by the initial state, rewrite rules and strategies defined above is correct with respect to the equationally defined ABS process (see Section 2.1). More precisely, the graphs generated by the application of the rewrite rules with the given strategy represent states reached by the system governed by the equational ABS model.*

**Theorem 2 (Completeness).** *The graph-based model defined by the initial state, rewrite rules and strategies defined above is complete with respect to stability as specified by the equational ABS model (see Section 2.1). More precisely, if the equational model reaches a stable state, so does our model.*

Theorems 1 and 2 ensure that our model reaches a stable state if and only if the ABS equational model (see Section 2.1 and [2]) reaches the same stable state.

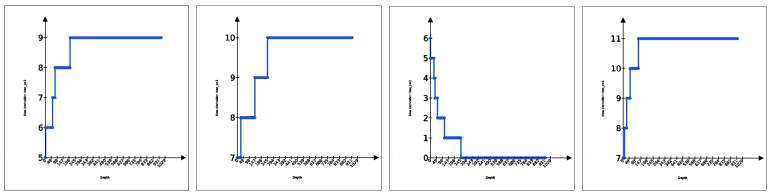
**Theorem 3 (Termination).** *The graph program consisting of the initial graph, rewrite rules and strategy described above terminates.*

More generally, if the rule *updatez* also changes the values of the asset attributes (reflecting changes in risk analysis cost, toxicity and asset value) then the graph program terminates if and only if stable state is reached.

*Experiments and Analysis.* Here we report results of experiments, where we run the model with different parameter values, i.e., different values for the probability of toxicity, due diligence cost, initial z, asset value, etc. 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 which in turn will generate stability. An illustration of this can be seen in Figure 6(c) and notice that given high due diligence costs Figures 6(b) and 6(e) highlight a negligent approach whereas Figures 6(c) and 6(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 6(l).

**Result 1 (Negligent equilibrium)** *If in the initial graph $Z \approx 1$, then the system arrives at negligent equilibrium (i.e., a result that reflects a community decision to no longer perform due diligence checks on a particular asset) even when the asset has high probability of toxicity.*

[5] A strategy fails if it attempts to apply a rule that is not applicable.

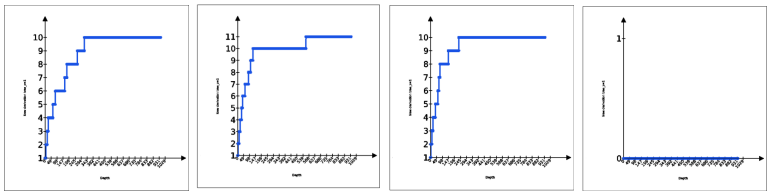


(a) Low Toxicity, High Due Diligence Cost,Mixture of Diligent and Negligent Banks

(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

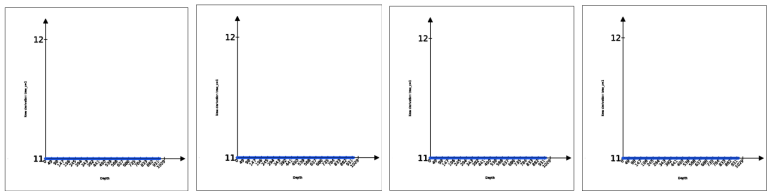


(e) High Toxicity, High Due Diligence Cost, Diligent Banks

(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. 6.** *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)

Explanation: For high $p$ the profitability equation outlined in section 2.1 reduces to: $\mathcal{U}(1) - \mathcal{U}(0) \approx Z - c + x_w$ given that the difference between $\mathcal{U}(1)$ and expected profit when the rule is not followed (i.e. $\mathcal{U}(0)$) is $p(Z - c) + x_w$. This linear equation will be computed at each iteration of the repeat loop. The result must be positive given that $c$ and $x_w$ are both positive constants smaller than 1.

Similarly, we observe that if $p$ is high but in the initial graph the majority of banks are deviating from the negligence rule, then the system reaches a due diligence stable state. We can also infer from the model conditions to avoid a market crash, such as the one below (that although not feasible in a real market, is valid in equational models).

**Result 2 (Indefinite propagation)** *A continuous increase in the number of negligent bank agents means that a market crash can be postponed.*

Explanation: A continuous increase in the number of agents used in calculating the average current sentiment, $Z$, as outlined in section 2.1 and as computed by PORGY, means that the value of $Z$ used in deciding whether or not to perform an independent risk analysis can remain unchanged.

**Result 3 (Dangerous Equilibrium)** *A negligent stable state/equilibrium can be reached despite high toxicity under certain circumstances (high number of negligent banks).*

Explanation: A sensitivity analysis shows that for a certain range of high due diligence cost values, negligent equilibrium can be obtained despite toxicity values noted by the base case as high, see Figure 6 (j and l).

*Additional Experiments.* The results obtained with the basic experiments performed so far suggest that the graph rewriting approach, and in particular the derivation tree provided by PORGY, could be used to get insights beyond simulation runs. For example, the derivation tree could be used to search for states with specific properties, or to identify the occurrence of specific events (e.g., the first application of a specific rule). More meaningful analyses could be carried out by experts in the area, such as calculating propagation speeds (i.e., number of steps it takes for rule sentiment to be adopted by all agents relative to the size of network or the rate of change of average sentiment within different environments), taking into the account the pay-down factor of the loans supporting the asset and the expected contractual degradation of the asset itself, etc.

## 5 Related Work and Conclusions

Graph Transformation Systems (GTSs) have been used as a modelling framework in many areas: for example, RuleBENDER[6] is a rule-based simulation tool for the modelling of biochemical systems and is compatible with ordinary and partial

[6] `http://www.rulebender.org`

differential equation projects written in the BioNetGen Language [7]; Kappa [10] is a rule-based language for modelling protein interaction networks; and more generally, graph transformation has been used to outline the semantics of domain specific modelling languages (DSMLs) [4].

General purpose agent-based simulation tools and platforms[7] 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. Other tools and languages like Stratego, Maude and ELAN [11] support a pure term-writing approach which in the case of Maude is augmented by probabilistic features. The visual, declarative nature of GTSs is thus welcome in the cases where users seek to primarily focus on describing what the system should accomplish, and is especially useful for the analysis of complex systems in interactive environments.

A benchmark analysing the differences between several GTS tools has been developed by Varro et al. [12]. Among other available GTS tools, we can cite GROOVE [13], a graph-based model checker for object oriented systems; AGG (the Attributed Graph Grammar System) [14], a graph-based language for the transformation of attributed graphs that comes with a visual programming environment; PROGRES (Programmed Graph Rewriting Systems) [15] that offers backtracking and nondeterministic contracts; GrGen (Graph Rewrite Generator) [16] that uses attributed typed multigraphs and includes features such as Java/C code generation, and GP [17], a graph programming language, where users can define rules and strategy expressions, with support for conditional rewriting. PORGY [18, 8], the implementation environment used in this work, has previously been used to model social networks and biochemical processes, where non-determinism, backtracking, positioning constructs, and probabilistic rule application are also key features.

We have shown that strategic port-graph rewriting provides a basis for the design and implementation of models of the rational negligence phenomenon. Whilst ABMs rely on the internal processing of its agents, GTSs provide at each point in time a holistic view of the system state and a visual trace of the specific rules that trigger specific behaviours. In future, we hope to further develop the model using hierarchical graphs [19] to be able to capture all tiers of the model, and also generalise the rules to permit dynamic changes in key attributes such as asset toxicity and costs.

## References


1. Markose, S., Dong, Y., Oluwasegun, B.: A multi-agent model of RMBS, credit risk transfer in banks and financial stability: Implications of the subprime crisis (2008)
2. Anand, K., Kirman, A., Marsili, M.: Epidemics of rules, rational negligence and market crashes. The European Journal of Finance **19**(5) (2013) 438–447
3. Farmer, J., Gallegati, M., Hommes, C., Kirman, A., Ormerod, P., Cincotti, S., Sanchez, A., Helbing, D.: A complex systems approach to constructing better


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

models for managing financial markets and the economy. The European Physical Journal Special Topics **214**(1) (2012) 295–324
4. 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
5. Gorton, G., Metrick, A.: Securitization. Working Paper 18611, National Bureau of Economic Research (December 2012)
6. Fernández, M., Kirchner, H., Pinaud, B.: Strategic Port Graph Rewriting: an Interactive Modelling Framework. Research report (2017) https://hal.inria.fr/hal-01251871.
7. Smith, A.M., Xu, W., Sun, Y., Faeder, J.R., Marai, G.: RuleBender: integrated modeling, simulation and visualization for rule-based intracellular biochemistry. BMC Bioinformatics **13**(8) (2012)
8. 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
9. Löwe, M.: Algebraic approach to single-pushout graph transformation. Theor. Comput. Sci. **109**(1&2) (1993) 181–224
10. Krivine, J., Danos, V., Benecke, A.: Modelling epigenetic information maintenance: A Kappa tutorial. In: Computer Aided Verification, 21st International Conference, CAV 2009, Grenoble, France, June 26 - July 2, 2009. Proceedings. (2009) 17–32
11. 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).
12. Varro, G., Schurr, A., Varro, D.: Benchmarking for graph transformation. In: 2005 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC'05). (Sept 2005) 79–88
13. Ghamarian, A.H., de Mol, M., Rensink, A., Zambon, E., Zimakova, M.: Modelling and analysis using GROOVE. STTT **14**(1) (2012) 15–40
14. Taentzer, G.: AGG: A graph transformation environment for modeling and validation of software. In: Applications of Graph Transformations with Industrial Relevance. Volume 3062 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2004) 446–453
15. Schürr, A., Winter, A.J., Zündorf, A.: The PROGRES approach: Language and environment. In: Handbook of graph grammars and computing by graph transformation, World Scientific Publishing Co., Inc. (1999) 487–550
16. Geiß, R., Kroll, M.: GrGen.NET: A fast, expressive, and general purpose graph rewrite tool. In: Applications of Graph Transformations with Industrial Relevance, Third International Symposium, AGTIVE 2007, Kassel, Germany, October 10-12, 2007, Revised Selected and Invited Papers. (2007) 568–569
17. Plump, D. In: The Graph Programming Language GP. Springer Berlin Heidelberg, Berlin, Heidelberg (2009) 99–122
18. Andrei, O., Fernández, M., Kirchner, H., Melançon, G., Namet, O., Pinaud, B.: Porgy: Strategy-driven interactive transformation of graphs. In: TERMGRAPH. (2011) 54–68
19. 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

# A Appendix

## A.1 Model Properties Continued

The theorems given in Section 4 are stated below along with their proofs:

**Lemma 1** *Strategies 1 (FixedPointSearch) and 2 (AllTrade) never fail.*

*Proof.* Strategy 2 (*AllTrade*) sets the position for rewriting (a *setPos* command cannot fail), and then executes a command of the form $repeat(S)(k)$, which according to PORGY's semantics [6] iterates the strategy $S$ while it succeeds and always returns *id* (i.e., it can never fail). The number $k$ indicates the maximum number of iterations of the loop. Since *AllTrade* cannot fail, strategy *FixedPointSearch* can only fail if the rule *change* in the body of the while loop fails, which is impossible due to the condition in the "while" (there is at least one match for *change*). This completes the proof.

**Theorem 1 (Correctness)** *The graph-based model defined by the initial state, rewrite rules and strategies defined above is correct with respect to the equationally defined ABS process in [2]. More precisely, the graphs generated by the application of the rewrite rules with the given strategy represent states reached by the system governed by the equational ABS model.*

*Proof.* We show that one trading transaction in our system corresponds to one trading transaction in the equational model. Let $w$ be the bank that owns the asset (i.e., the bank linked by an edge to the asset), and let $w'$ be the randomly chosen potential buyer (selected by the application of the rule *requesttobuy*). Rule *beginanalysis* computes the value of the projected profitability made by $w'$ following and not following the negligence rule using the attributes p-tox, c-val and dd-cost (i.e., probability of toxicity, current value and due diligence cost) in the asset, which correspond to the values of $c$, $x$ and $p$ in the equational model. It computes the difference between $\mathcal{U}_w(1)$ and $\mathcal{U}_w(0)$ using the equations given in section 2.1 and stores it in the attribute $DeltaU1U0$. The result of this computation is the values specified by the equations:

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

and

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

The best response, given by $\mathcal{U}(1) - \mathcal{U}(0)$, is accurately computed in the attribute $DeltaU1U0$ of node *Theta* (the indicator value of best choice). The strategy ensures that the potential buyer selects the most profitable choice (lines 5-6 of Strategy 2), and the rule *updatez* recomputes the global Z value as outlined in Table 3, as follows:

$$Z_{i+1} = \frac{Z_i * (k - 1) + z(w')}{k}$$

which gives the average value specified in the equational model.

$$Z = (\frac{1}{k_i}) \sum_{j \in N_i} z(w)$$

where $k_i$ is the number of selling agents and $z(w')$ is the estimated sentiment of a buyer (indicating whether it has decided to follow or not the rule). $N_i$ the set of agents in the universe. The case in which $|N_i|$ and $k_i$ coincide (represented in our model by the value $k$) is a centralised system.

**Theorem 2 (Completeness)** *The graph-based model defined by the initial state, rewrite rules and strategies defined above is complete with respect to stability as specified by the equational ABS model in [2] (see Section 2.1), assuming all the banks are given the opportunity to trade. More precisely, if the equational model reaches a stable state, so does our model.*

*Proof.* (Sketch) The transactions of the equational ABS model are mimicked by the iterations of the "repeat" in our strategy. A stable state in the equational model is reached when banks do not change their approach to negligence, which corresponds to absence of "Change" in our model: the *Change* flag is updated as required when rules $followdecision$ and *deviationdecision* are applied (see Table 3).

**Theorem 3 (Termination)** *The graph program consisting of the initial graph, rewrite rules and strategy described above terminates.*

More generally, if the rule UpdateZ also changes the values of the asset attributes (reflecting changes in risk analysis cost, toxicity and asset value) then the graph program terminates if and only if stable state is reached.

*Proof.* A state is stable if no bank has changed its mind regarding its negligence choice when given an opportunity to trade. If stable state has been reached, there is no change after executing *AllTrade* hence the while loop found in line 2 of Strategy 1 stops. Conversely if our strategy terminates then the *change* rule does not apply since this is the condition to exit the while, hence no bank has changed its behaviour in *AllTrade* (stability has been reached). Thus, the graph program terminates if and only if the initial graph reaches a stable state.

Moreover, if the parameters of the asset do not change during the simulation then the program is guaranteed to terminate, because in this case $Z$ is monotonic (once a bank decides towards diligence or negligence, the rest follow the trend). Thus, in this simple case, the program terminates.

### A.2 Rewrite Rules

Diagrams in Figures 7 to 14 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.

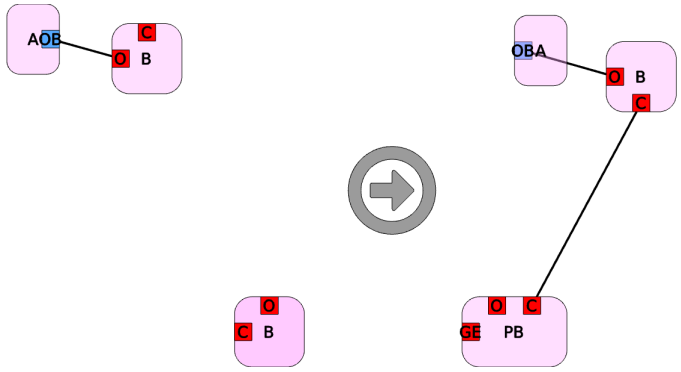



**Fig. 7.** *Request to Buy.* Sends a request-to-buy message to a random bank B changing the name of this node to PB (PotentialBuyer)

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

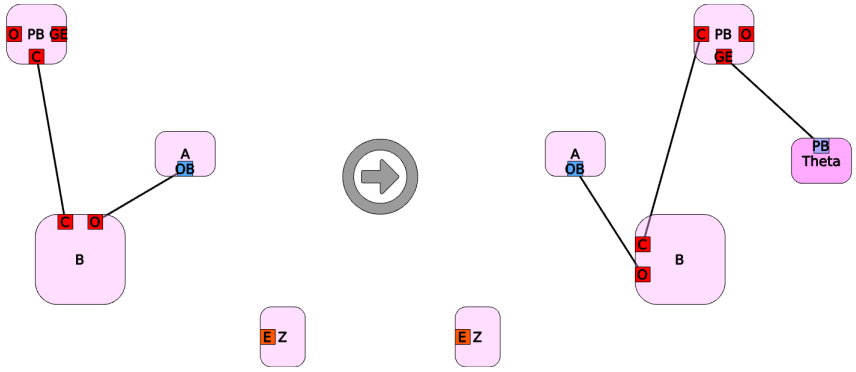



**Fig. 8.** *Begin Analysis.* Computes profitability U1, U0 of PB, generating a node Theta with attribute DeltaU1U0 = U1 - U0


**Fig. 9.** *Follow Result.* Applies if DeltaU1U0 $\geq 0$. As additional visualisation support, it generates a follow node, which indicates that it is more profitable to follow the negligence rule and not do a full risk analysis

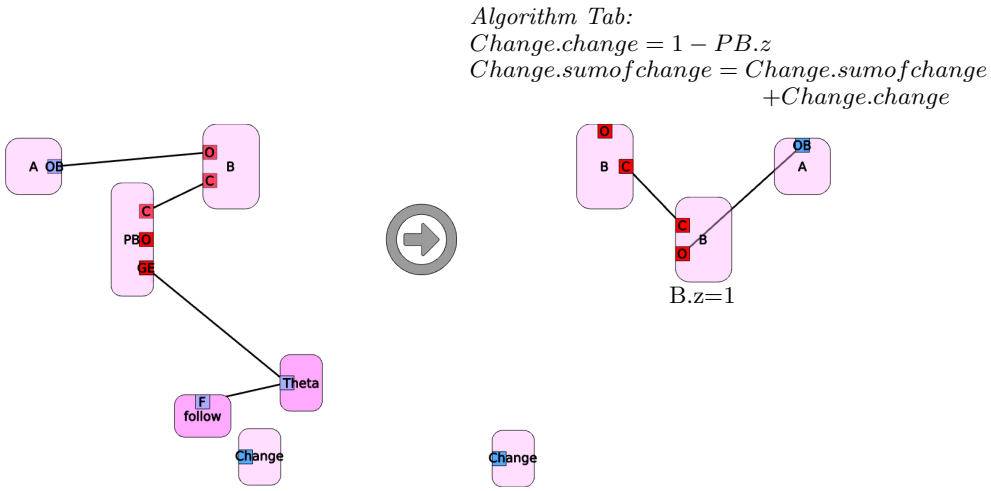



**Fig. 10.** *Follow Decision.* Transfers asset and prepares for a new transaction (i.e. cleans up after the decision to follow the negligence rule), setting buyer's attribute z to 1 (which indicates negligent behaviour) and increasing the Change counter sumofchange if there was a change in the bank's behaviour (i.e., if the new value of z is different from PB's value).

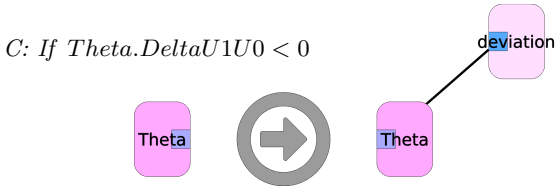



**Fig. 11.** *Deviation Result.* Applies if DeltaU1U0 $< 0$. As additional visualisation support, it generates a deviation node if more profitable to do a full risk analysis (deviating from the negligence rule).

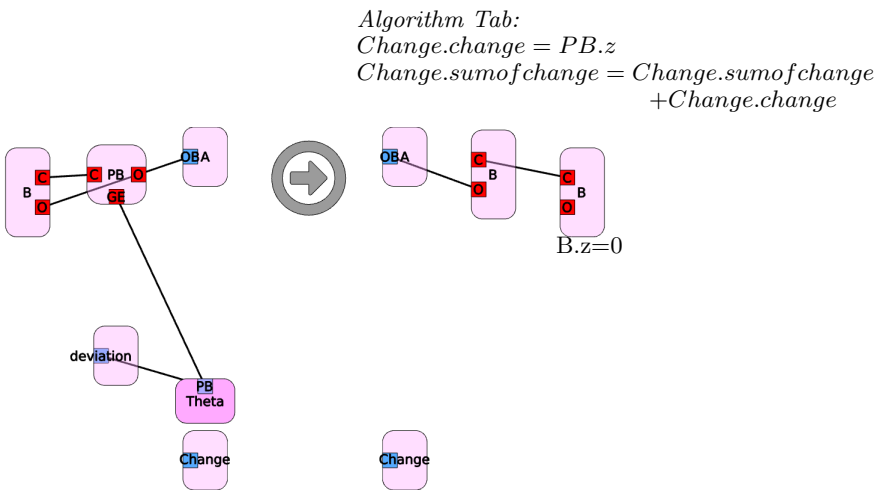



**Fig. 12.** *Deviation Decision.* Transfers asset and prepares for a new transaction (i.e. cleans up after the decision to deviate from the negligence rule), setting buyer's attribute z to 0, and increasing the Change counter sumofchange if there was a change in the bank's behaviour (i.e., if the new value of z is different from PB's value).

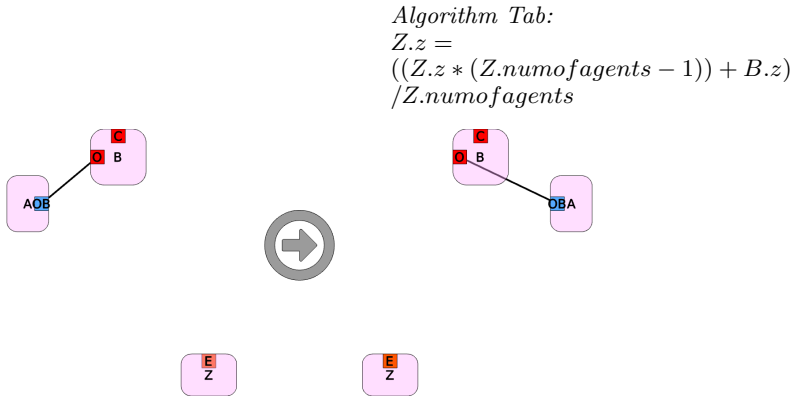



**Fig. 13.** *Update Z.* Updates the attribute z in node Z (which represents the average of the values of z in each bank). The new value in Z is (Z * (k - 1) + z(PB))/k

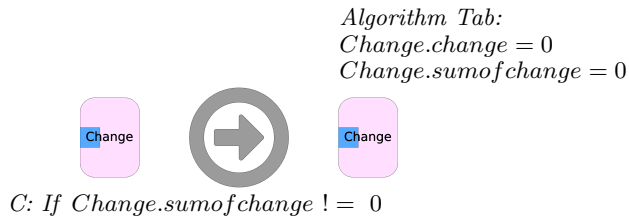



**Fig. 14.** *Change.* Sets the Change counter back to 0 if greater than 0