# On the Reactive Nature of Financial Networks

Silvia Crafa    Daniele Varacca

Università di Padova, Italy    LACL, Université Paris Est-Créteil, France

**Abstract.** We propose to model financial networks as distributed and reactive systems. Both financial institutions and financial contracts such as loans, assets or derivatives, are embodied as concurrent entities. These entities operate in parallel and dynamically update their state as a reaction to events issued by other entities. e.g. a macroeconomic shock or a bank default. The model is given in terms of an architectural pattern that can be directly implemented as an Actor system. We show how this reactive model can be used to study the systemic risk of financial networks and to design and conduct very flexible stress tests that include macroeconomic shocks and derivative contracts such as Credit Default Swaps.

## 1 Introduction

Business relationships between financial institutions, such as inter-banking borrowing and lending activities, have been often represented in terms of networks. For instance [CMS13,CM16,Cap16,BPK$^{+}$12] rely on weighted directed graphs whose nodes represent institutions and links represent financial dependencies. This network model has been used to analyse global properties of the financial systems such as the systemic risk, that is the fragility of the system when exposed to the contagious effects of distressed financial institutions.

Despite the success of these methodologies, a graph remains a static structure, whereas a financial system has an highly dynamic nature. A graph can then represent just a snapshot of the system at a given time. Therefore modeling, and thinking, in terms of graphs, naturally shifts the view towards the structure of the system rather than its dynamics/behavior. Accordingly, the current literature points out a number of structural properties, i.e. network-based measures of connectivity and risk such as counterparty susceptibility, local network fragility, feedback centrality (DebtRank). Furthermore, a graph is a very abstract model, that is not well suited to systematically incorporate additional information. For instance, given a weighted graph of inter-banking borrowing and lending, adding information about derivatives contracts depending on defaulted loans, requires multi-layered models such as the superimposal of additional graphs [CM16].

In this work, we propose to model financial systems as distributed and reactive systems. The key point is recognizing that financial institutions operate in parallel and they dynamically (and frequently) set up, trade and terminate contracts and other forms of assets. Sometimes trading decisions are taken independently, but often contracts are started/closed/modified *as a reaction* to other actions, e.g. a reaction to a macroeconomic stress. Default contagion is another source of reactivity, where the path (i.e., the sequence of events) that leads from an initial bank default to other bank defaults might be relevant to devise a recovery strategy. Therefore we argue that modeling the financial system as a reactive system better highlights properties that can be difficult to capture in terms of graphs, such as safety or liveness invariants of the system behavior, but also compositionality and nondeterminism. Moreover, this opens the way to reuse in the financial context the rich toolbox of analysis and verification techniques developed for digital concurrent systems.

Modeling and analysis techniques from (reactive) concurrent systems have been successfully applied to biological systems, which also have a highly concurrent and dynamic nature (e.g. [Kri17,DHK12,CH09,BDPP15]). However, while modeling biochemical systems requires quantitative and probabilistic variants of process calculi, we argue that financial systems naturally suit the plain Actor Model[Agh86]. Indeed, a financial network can be viewed as a set of interactive *entities*, that asynchronously react to external events, e.g. market or credit events, by modifying their behavior according to their internal policy and locally stored information (strong locality and isolation principles). There will not be always necessary to incorporate probabilistic reactions: the internal policy of a bank may consider stochastic algorithms, but the fact that a bank reacts to a market event according to its internal policy is deterministic. Moreover, each financial institution and each financial asset have a well-defined identity, e.g., the legal name, or an entry in a mortgage register, while biochemical reactions depend on the concentration rate of molecules or on absence of inhibitor molecules.

Besides being a natural and elegant approach for modeling financial networks, using Actors has the following advantages:

1. it better highlights properties that can be difficult to capture in terms of graphs, especially properties dealing with the nondeterministic dynamic of the systems and the effect of timing on (strategic) reactions to events;
2. since Actors systems can be directly implemented using mainstream programming languages, this model allows the straightforward development of very flexible and interactive simulation tools that support the analysis in a test-oriented style, which is hardly possible with other more complex mathematical simulation methods;
3. the model, being just based on entities that react to events issued by other entities, is very abstract but, differently form graphs, it easily scales to more concrete situations by simply $(i)$ refining the information encapsulated into entities and $(ii)$ by enlarging the set of events the entities react to.

In the rest of the paper we illustrate how to model a financial network as a reactive concurrent system. The model is given in terms of an architectural pattern that can be directly implemented as an Actor system. Accordingly, we refer to FinMarkSim® , our software tool written in Erlang, that allows the user to design and run very flexible, feasible and scalable stress tests over the financial system entered as input. Building on the financial literature, we show how this reactive model can be used to study the systemic risk of financial networks, even in presence of macroeconomic shocks. We finally illustrate how to extend the basic architectural pattern so to model a refined scenario where financial institutions' portfolios also contain Credit Default Swaps derivatives.

We mention that some of the work presented in this article is the subject of a recent patent application the content of which cannot be fully disclosed.

## 2 Banks, Assets and Loans

### 2.1 The Architectural Structure

The architectural structure we devise is based on two kinds of entities: financial *institutions* (e.g., banks, finance companies, brokerage companies, development agencies) and *contracts*, which generally refer to any financial agreement (e.g., loans, credits, bonds, swaps like CDS) and generally any kind of assets. Considering contracts at the same abstraction level of institutions, rather than being just part of the institutions state, allows for a better systemic account of the financial system. Indeed, decoupling contracts from institutions allows more flexibility in the modeling and a deeper observation of the system's reactions, especially in terms of the happens-before effects due to asynchrony and nondeterminism. It also opens the way to model smart contracts, which are contracts encoded as software so that contractual obligations are automatically self-enforced.

Actors, that is entities in the model, interact *by exchanging messages*: the sending of a message can be thought of as the communication that an event has occurred, so that the receiver of the message can *react* to the received message by modifying its behaviour according to its internal policy. As in the standard Actor Model, we rely just on asynchronous and point-to-point communication, while synchronous messages and broadcast could be added later. Notice that asynchrony seems to be particularly well-suited to financial interactions where counterparties tend to take their time to react to requests, e.g., a request of payment can be fulfilled after some delay, either because the debtor needs time to collect liquidity, or because of strategic or market reasons. We will see that exposing such a delay in message responses allows the model to capture complex scenarios.

Since the model we devise complies with the standard Actor Model, we directly implemented it as a software tool written in the Erlang programming language, so to actually run simulations of reactive financial networks as a system of Erlang Actors. Below we illustrate the architectural design of the FinMarkSim® tool using a pseudo-code notation that should be easy to understand also for readers that are not familiar with actor-based programming languages.

### 2.2 The Entities

Each entity in the system is characterized by:

- the *identity*, that identifies the institution or the contract, but that it is also used in contracts to legally refer to contract parties and to reference entities, e.g. the reference to the loan in a CDS;
- the *state*, that is a representation of the internal information, e.g., the assets of a bank or the value of a loan. The content of the internal state is private to the owner entity, that is it can be queried and modified only by the owner entity itself;
- the *interface*, that expresses how it is possible to interact with that entity. More precisely it lists the set of messages the entity reacts to, i.e., the events that can be notified to the entity so that it can modify its behavior.

```
Institution  identity
state =
  liquidity    : Money
  assets       : List[Asset]
  loans        : List[Loan]
  cdsPortfolio : List[Cds]

interface =
  default                           triggers an immediate default
  checkValue                 calculate value and possibly default
  addContract(c:Contract)     add the contract c:loan,asset,cds
  payCds(b:Institution,n:Money)         pay the notional n to b
  addLiquidity(n:Money)          modify liquidity, n can be <0
```

```
Loan identity
state =
   value     : Money
   creditor  : Institution
   debtor   : Institution

interface =
   default(who:Institution)          who notifies its default
```

```
Asset identity
state =
   value  : Money
   owner  : Institution

interface =
   change(r:Rate)                modify the value of the asset
```

**Fig. 1.** FinMarkSim$^{®}$ architecture

More concretely, Figure 1 defines the structure of loan, asset and institution actors. All the entities have a unique identity, that in the tool corresponds to the Pid of the Erlang process. The internal state of a financial institution informs on the current liquidity (essentially the Tier 1 capital, representing its capacity for absorbing losses, see e.g. [CMS13]) and the current list of contracts held by the institution. At the moment we consider three kinds of contracts: external assets, loans and CDSs, but more can be added just by refining the state of institutions. Each asset is an actor whose state holds information about the current value of the asset and a reference to its owner. A loan actor records in its state a reference to the debtor and the creditor institutions, together with the amount of borrowed money, corresponding to the loan's value. Therefore, the list of loans contained in the institution's state contains a reference to any loan for which the institution is either the creditor or the debtor. We postpone the discussion of CDSs to the next section. Future work could consider refining definition of loans by

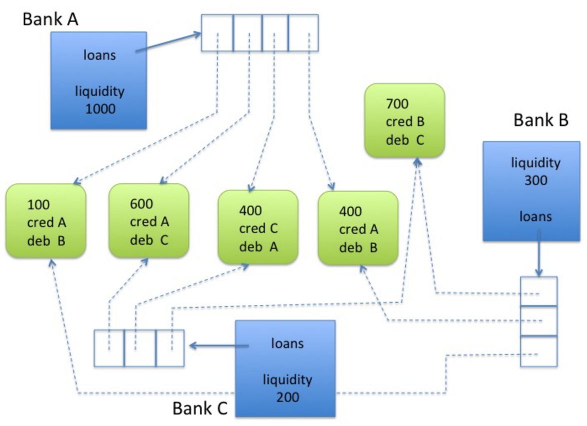


**Fig. 2.** A financial system

also considering interest rates. At the moment we keep a simple but nontrivial scenario considering both interbank exposures and external assets so to implement stress tests which take into account macroeconomic shocks.

In line with the literature, we consider defaults as generated by institution *insolvency*, that is when debts exceed the sum of credits, external assets and institution's liquidity.

**Definition 1 (Institution's value).** *The value (a.k.a. the net worth) of the institution $i$, written $value(i)$, is the sum of the liquidity of $i$, the values of its external assets and the value of its loans, where loans issued as the creditor have positive value while loans issued as the debtor have negative value. When $value(i) < 0$ institution $i$ becomes insolvent.*

*Remark 1.* It is easy to see that all the information encoded in the network model of [CMS13] can be encoded as an actor system structured as in Figure 1. For instance, the in-degree and the out-degree of each node $i$ in the graph model of [CMS13], that is the number of $i$'s debtors and creditors, correspond to the number of loans listed in $i$'s attribute `loans` for which $i$ is the creditor, resp. debtor, institution. The weight $e_{ij}$ of an edge $(i, j)$ in the network model corresponds to the `value` attribute of the loan actor whose state records $i$ as the creditor and $j$ as the debtor institutions. We also remark that graph models appearing in the literature usually assume that there is at most a single edge between a given pair of nodes $i, j$, and the weight of that edge is the sum of the values of all the loans established between $i$ and $j$. We do not require the same assumption: there can be multiple loan actors that have the same state (hence the same creditor/debtor) but with different identities.

As an example, Figure 2 shows a financial system with three institutions, no external assets and a number of loans. Bank $B$ has three contracts: the credit to the bank $C$, and two loans corresponding to two debts with bank $A$. Therefore, the value of $B$ is

$value(B) = 300 + 700 - 400 - 100 = 500$. Notice that each loan object is referred by a pair of institutions, accordingly, it is listed both in the creditor's and debtor's state.

The actors' interfaces given in Figure 1 are very simple: the messages `default` and `checkValue` deal with the default procedure which is explained below; the explanations of the asset's `change` message and the message `payCds` are postponed to the explanation of the stress test and the CDSs mechanisms. The remaining institution's messages `addContract` and `addLiquidity` are used to induce the corresponding modification of an institution's internal state.

## 3 The Dynamics of the Model

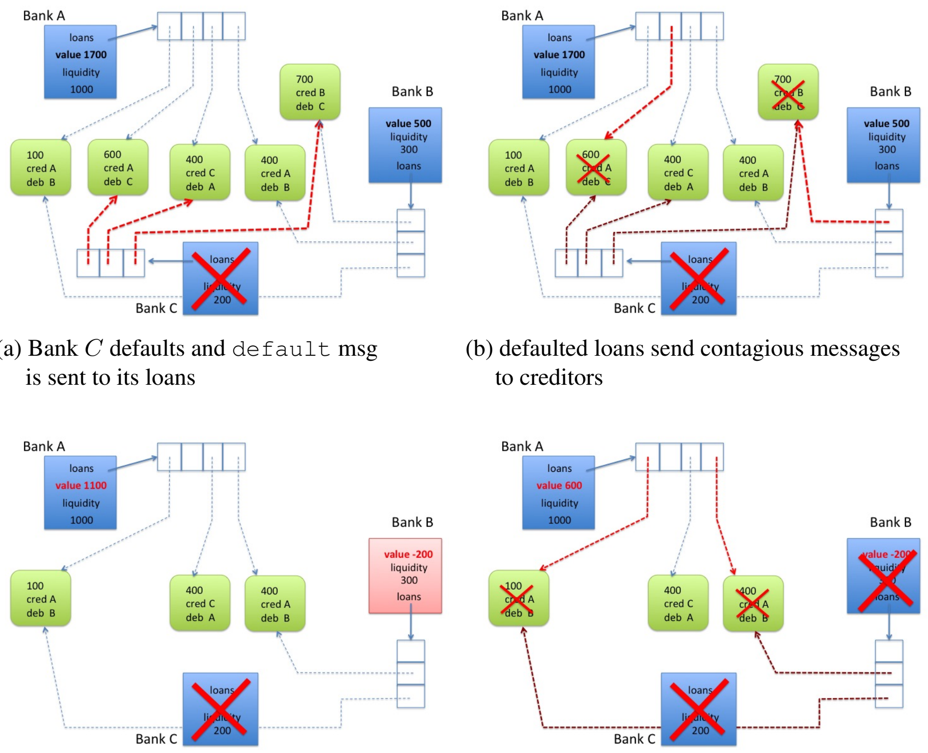
### 3.1 Default procedure

The default procedure can be initiated in two ways: a `default` message can be directly sent to an institution to trigger its immediate and unconditional default. A second possibility is to send a `checkValue` message, which is handled by the institution by calculating its internal value and, in case of a negative value, (the liabilities are higher than the assets plus the liquidity) declaring its default. Defaults can also occur as a consequence of a macroeconomic shock that decreases the value of (some) asset causing some institution to become insolvent. We will discuss this scenario in the next section, when explaining stress tests, here we focus on the default procedure that starts when an institution becomes insolvent.

When an institution $i$ defaults, a number of actions are triggered, modeling the systemic reaction to the default event. More precisely,

- $i$ sends a `default(i)` message to all the loans in its list informing them about its failure;
- $i$ updates its state and its interface to empty: this is for simplicity, but we could keep the info about default so to model the behavior of defaulted institutions;
- all the loans that received the `default` message from $i$ *react (in parallel)* by checking whether $i$ stands for the loan's creditor or debtor. If the defaulted $i$ is the creditor institution, nothing happens (but we could refine this in case of need). On the other hand, if $i$ is the debtor institution, then the default of $i$ implies the default of this loan, thus triggering a new *credit event*. Therefore the current loan's value is updated to 0 and a message is sent to the loan's creditor, say $j$, so to propagate the *contagion*. We model such a contagious message as a `checkValue` message (sent to $j$), which then causes the institution $j$ to internally check its value and possibly recursively default along the same procedure of $i$. Notice indeed that when $j$ computes its value as a consequence of the `checkValue` message received from one of its loans, it will for sure compute a reduced value since one of its debtors has just defaulted and the corresponding loan (which sent the contagious message) has been already set to 0.

This recursive procedure actually models contagion via an insolvency cascade. As an example, Figure 3 shows a systemic contagion where the default of Bank $B$ is induced by the initial default of Bank $C$. Notice that credits issued to a bank that defaults are not deleted, for future recovery computation, e.g. the credit $C$ gave to $A$ survives $C$'s default in Figure 3(c-d).



(a) Bank $C$ defaults and `default` msg is sent to its loans

(b) defaulted loans send contagious messages to creditors

(c) Bank $B$ suffers illiquidity and defaults

(d) Bank $A$ is well capitalized and survives

**Fig. 3.** An example of systemic contagion by illiquidity cascade

## 3.2 Systemic Risk in the literature

Since our model subsumes a graph structure, we can account for a number of measures put forward in the literature that study the systemic risk of financial networks. We mention here only a few, as examples:

**Loss cascade** Let $V$ be the set of institutions in the network, then in [CMS13] the authors define the loss cascade $(c_{|V|-1}(j), j \in V)$ as the remaining liquidity of the institution $j$ once all counterparty losses have been accounted for, i.e. when all the credit events generated by an initial default have been handled. In [CMS13] the values $(c_{|V|-1}(j), j \in V)$ are computed with $|V|$ iterations; our model requires a different approach since the loss cascade propagates concurrently and nondeterministically. For instance, let $A$ be a bank that has a debt with both $B_1$ and $B_2$, and $B_1$ has a debt with $C_1$ while $B_2$ has a debt with $C_2$, and both $C_1$ and $C_2$ have a debt with $D$. If all these banks are not well capitalized, the default of $A$ may lead to the default of the whole system. With the procedure devised in [CMS13] the loss cascade is deterministic: initially only $A$ is insolvent, then also $B_1$ and $B_2$, at the third step $C_1$ and $C_2$ become insolvent too, and in the final step also $D$ defaults. In that procedure the iterations are just a computation algorithm. Instead, in our model the default contagion starting form $A$ propagates concurrently along the *"causal paths"* $A - B_1 - C_1 - D$ and $A - B_2 - C_2 - D$. In the end all the banks will be insolvent but in some run $C_1$ (resp. $C_2$) may default before $B_2$ (resp. $B_1$), while in other run it may be the opposite, revealing that the defaults of $C_1$ and $B_2$ (resp. $C_2$ and $B_1$) are independent.

**Default Impact** The Default Impact of the institution $i$ is defined as the total loss capital induced by the default of $i$: $DI(i) = \Sigma_{j \neq i}\, c_0(j) - c_{|V|-1}(j)$. Default Impact focuses on the loss this initial default inflicts to the rest of the network: it thus measures the loss due to contagion. As for the loss cascade, the default impact measure can be obtained also in our model by computing the values of the institutions at the end of the default procedure.

**Group DebtRank** In [BPK$^+$12] the authors propose a measure to estimate the impact on the whole system due to a shock hitting the financial network. More precisely, *Group DebtRank* measures the impact of the simultaneous default or distress of several nodes in the network. As for all measures based on the graph structure, Group DebtRank can be computed in the same way in our model. We can model the initial default of a group of banks by initially sending the `default` message in parallel to all the institutions in the target group. The initial defaults will not be simultaneous and the execution will nondeterministically interleave the initial defaults with the systemic propagation of the contagions. Exposing the steps of the default contagion will be useful not only to define static global properties of the net, but also to reason about its dynamics, possibly devising strategic recovery methods such as choosing between modifying assets and increasing the liquidity of an institution.

The default procedure of FinMarkSim® follows the default cascade dynamics, that is no distress/contagion is propagated if it is below a given threshold, actually, if the distress in not enough to produce a local default. This procedure could be easily refined to propagate any level of distress along the lines of the DebtRank procedure. However, we decided to model the systemic propagation of defaults and rather use assets and market shocks to model other forms of distress propagation in the network. Finally, we remark again that the notions studied in the literature are used *to globally reason* on the network *structure*, e.g., to measure the systemic risk of the network nodes. In this work we focus instead on *modelling* the net as a reactive system, keeping the analysis of the system for future work, by possibly resorting to formal methods developed in the literature about concurrent systems.

## 3.3 Macroeconomic shock

A more realistic scenario for studying systemic risk requires to integrate contagion effects with macroeconomic shocks. Indeed, economic downturns produce losses in asset values (e.g. bonds, stocks, real estates), hence losses in institutions' values, thus making the financial network less resilient and contributing to the amplification of the magnitude of contagion. Accordingly, in [CMS13,BCDG16] the authors observe that correlated market shocks may increase the proportion of contagious exposures considerably (up to fifteen times), so ignoring market risk when assessing contagion effects can lead to a serious underestimation of the extent of default contagion.

In [CMS13] macroeconomic shocks are introduced as a (negative) random variable that decreases the institution liquidity with a severity that depends on the capital and the creditworthiness of the institution, that is institution with higher default probabilities are more affected by a macroeconomic shock. The systemic risk is then measured in terms of the *Contagion Index* of a given institution $i$, which is defined as the Default Impact of $i$ when the network is subject to a shock.

Our tool FinMarkSim® allows a very flexible modeling of macroeconomic shocks by means of the interface of the Asset Actor (cf. Figure 1). A `change(r)` message can be sent to assets, where `r` is a parameter determining the severity of the asset shock. We can choose to stress a single asset, a specific set of assets (e.g., those related to a specific market sector, or those held by a specific group of banks) or all the assets in the system. Moreover, as in [CMS13] we can study the impact of specific defaults *after* an asset shock, but in our model we can even let the macroeconomic shock happen *during* the default contagion. This last scenario is more realistic because real default procedures take time to complete and they induce market shocks in a sort of feedforward loop. We thus draw attention to the fact that modeling a financial network as a reactive concurrent system allows a very flexible and expressive design of stress-tests, that in combination with a straightforward and scalable implementation, effectively allows to monitor the systemic risk of a financial system in a test-oriented style.

## 4 Adding the Credit Default Swaps

While in the previous section we showed that we can recover several measures that were proposed in the literature, we argue here that our model can go further. The example we chose to deal with concerns Credit Default Swaps.

Credit Default Swaps (CDSs) are financial contracts that have been introduced to manage the credit risk. Given a loan $L$, the creditor can buy a CDS over the loan $L$ to be protected from the risk that the debtor will not honor its debt. Therefore, if the debtor defaults, the seller of the CDS will pay $L$'s creditor an amount agreed when the contract was created. It has been observed ([Con10]) that the high volume of CDSs had a relevant impact on the recent financial global crises, thus it is important to consider CDSs in the study of systemic risk of financial networks. The literature relies again on graph models, which show their limits when additional information such as CDS derivatives must be incorporated. In [PCB14] publicly available information about the CDS spread is used to reconstruct a financial network whose edges model the interdependencies among institutions over which CDS are issued. Stress tests and systemic risk measures defined in Section 3.2 are then applied to this net, which actually considers just CDS contracts. Other approaches simply refine the weight of the edges of a financial graph to aggregate the value of a loan with its corresponding derivative contract. However, in practice the dynamic of CDSs is detached from that of their underlying loans, i.e. a CDS can be sold, expire, or modify its spread, thus it is important to model the change in CDS exposures triggered by a credit event. Therefore, [CM16] put forward a multi-layered graph model where the dependencies due to derivative contracts belong to a different layer w.r.t. borrowing and landing exposures.

On the contrary, our approach based on a reactive system very smoothly allows to refine the analysis of a financial system by also incorporating information about CDSs. The key point here is the architectural structure we put forward in Section 2.1, in particular the fact that contracts entities are at the same abstraction level as institution entities. It is indeed sufficient to introduce a new type of actors corresponding to CDSs contracts, and refine the interface (and the corresponding reactive behavior) of other entities. More precisely, as anticipated in Figure 1, we let the state of institutions contain a list of CDSs called `cdsPortfolio` to represent the CDSs the institution bought and/or sold. The following pseudo-code defines a CDS actor that keeps in its state all the necessary data; we remind that the *spread* is the market price of the CDS while the *notional* is the amount of money the seller must give to the buyer if the *obligor* defaults.

```
Cds  identity
state = {
  spread    : Money
  notional  : Money
  buyer     : Institution
  seller    : Institution
  obligor   : Institution
}
interface = {
  default  the obligor send this msg to trigger the credit event
}
```

When an institution defaults, the procedure in Section 3.1 is refined so to send a `default` message not only to its loans, but also to all the CDSs for which it is the obligor. When a CDS entity receives a `default` message from its obligor entity, it sends a `payCds` message to the institution corresponding to the protection seller so that it pays the notional to the institution corresponding to the protection buyer, and finally expires. We model the payment by sending the `addLiquidity` message to the target institution and decreasing the liquidity in the seller's local state. Notice that, when receiving the `payCds` message, the protection seller may not be able to honor the CDS since its value may be not enough to cover the CDS notional, thus in this case the protection seller institution defaults, possibly starting an insolvency cascade.

*Modeling Central Counterparties* It has been observed ([Con10,CM16]) that the CDS market is significantly concentrated on five-ten top institution names. Moreover, clearinghouses, or central counterparties (CCP), have been proposed as a solution for mitigating counterparty risk and preventing default contagion in the CDS market. More precisely, a CCP acts as the buyer to every seller and seller to every buyer of protection, thereby isolating each participant from the default of other participants. The systemic risk of a financial network then crucially depends on capital and liquidity requirements for large CCP, whose (some degree of) transparency has been subject of various regulatory initiatives both in US and in Europe.

As an example, consider the scenario in Figure 3 with an additional CCP, and assume that the CCP sold to $B$ a CDS over its loan with $C$ and bought from $A$ a CDS over its credit to $C$, both with a notional of 300. Now when bank $C$ defaults, the two CDSs are triggered as a consequence of the credit event sent by $C$ to the CCP[1]. Therefore when the banks $A$ and $B$ re-check their value after the default of their loans **and** the payment of CDSs, they find values different from those in Figure 3 (c), in particular the CDS payment saves the bank $B$ from the default contagion.

It is important to observe that since the communication system is asynchronous, asking for a (CDS) payment has not an immediate effect: time is passing between the moment where the seller is asked to pay the buyer and the moment where the buyer gets the notional, and other things may happen during this time. For instance the buyer may have to use its liquidity to deal with a stress situation, and not having yet at its disposal the CDS notional it is expecting may increase the stress and possibly lead to default. This would be the case of Bank B in the example before, which needs a prompt payment of its CDS in order to break up the default contagion.

We claim that this is an important issue in real financial systems, where expected payments may be delayed. Our model naturally captures it by means of asynchronous concurrent communication and nondeterminism, while this phenomenon is difficult to account in graph-based models. Accordingly, the FinMarkSim® tool keeps track of the interleaving between default contagion and CDS payments, so to record (in each run) which institutions are saved from default due to a payment of CDS, and which one defaulted because of a delayed payment.

## 5 Conclusions

We have shown how financial networks can be productively modeled as distributed and reactive systems based on the Actor Model. Compared to the related literature, where financial analyses are based on mathematical methods on top of graphs, our computational model brings in a number of advantages due to the nature of concurrent systems. First of all, the model easily accounts for the dynamic of the net and the contagious effects of credit events such as a bank default or a macroeconomic asset shock. Moreover, the strong compositionality and encapsulation principles effectively support the design, the refactoring and the refinement of the financial system under observation, as shown by the case of CDS derivatives. Several extensions are possible, and easy to incorporate, as well as other measures of systemic risks (e.g. *Counterparty Susceptibility* and *Local Network Fraility* in [CMS13]).

A further fundamental aspect of this model is the fact that it brings to the fore the interleaving of events, thus exposing causally related events (more properly, happens-before relations) such as contagion propagation paths. The theory and practice of concurrent systems have developed a rich toolbox of analysis and verification techniques to reason about nondeterministic executions; their application to the analysis of financial networks will be an interesting direction for future investigations. We remark that, differently from concurrent systems, mathematical models usually resort to probability

[1] FinMarkSim® models CCPs as a specific type of institution actors which accept a specific message that can be used to notify the default of some institution.

to account for nondeterministic effects such as the unpredictable delay of an expected payment in a stressed scenario. Moreover, mathematical models usually describe and *simulate* relationships between quantities and variations between such relationships, while computational models encode and *execute* the fine-grained behavior of the system under observations.

The approach that is mostly related to our is that of Agent-Based Computational Economics [TJ06], where economic processes are modeled as dynamic systems of interacting agents. Indeed the Actor Model shares some similarity with Agent Systems. However we think that Actors are a specific instance of Agent System, that does not need the full generality of dynamic systems and the powerful mathematical methods behind them, such as game theory. We proved the effectiveness of the actor-based approach with our FinMarkSim® software tool, that implements very flexible and scalable stress-tests that allows to monitor the systemic risk of a financial system in a test-oriented style. We finally mention that our model, as customary in the financial research literature ([Cap16]), suffers form lack of real financial data, which are only partially disclosed by financial institutions and supervisory authorities. The usual approach is to indirectly reconstruct or generate missing data on the network structure, or to access specific datasets. For instance, the paper [CMS13] rely on real data as one of the authors works for Brasil's Central Bank. We plan to contact the institutions to ask for real data.

## References

[Agh86] G Agha. *Actors: A Model of Concurrent Computation in Distributed Systems*. MIT PRess, Cambridge, MA, USA, 1986.

[BCDG16] S Battiston, G Caldarelli, M D'Errico, and S. Gurciullo. Leveraging the network: a stress-test framework based on debtrank. *Statistics & Risk Modeling*, 33(3-4):117–138, 2016.

[BDPP15] G Boniolo, M D'Agostino, M Piazza, and G Pulcini. Adding logic to the toolbox of molecular biology. *European Journal for Philosophy of Science*, 5(3):399–417, 2015.

[BPK$^+$12] S Battiston, M Puliga, R Kaushik, P Tasca, and Caldarelli G. Debtrank: Too central to fail? financial networks, the fed and systemic risk. *Scientific Reports*, 2(541), 2012.

[Cap16] A Capponi. Systemic risk, policies, and data needs. *Tutorials in Operations Research*, pages 185–206, 2016.

[CH09] F Ciocchetta and J Hillston. Bio-pepa: A framework for the modelling and analysis of biological systems. *Theor. Comput. Sci.*, 410(33-34):3065–3084, 2009.

[CM16] R Cont and A Minca. Credit default swaps and systemic risk. *Annals of Operations Research*, 247:523–547, 2016.

[CMS13] R Cont, A Moussa, and EB Santos. *Network structure and systemic risk in banking systems.*, pages 327–367. Cambridge University Press, 2013.

[Con10] R. Cont. Credit default swaps and financial stability. *Financial Stability Review*, 14:35–43, 2010.

[DHK12] T Damgaard, E Højsgaard, and J Krivine. Formal cellular machinery. *Electr. Notes Theor. Comput. Sci.*, 284:55–74, 2012.

[Kri17] J Krivine. Systems biology. *SIGLOG News*, 4(3):43–61, 2017.

[PCB14] M Puliga, G Caldarelli, and S Battiston. Credit default swaps networks and systemic risk. *Scientific Reports*, 4(6822), 2014.

[TJ06] L Tesfatsion and K Judd, editors. *Handbook of Computational Economics. Agent-Based Computational Economics*. North Holland, 2006.