Derivative-Based Financial Network Clearing Transition From Intractability to Computable Solutions

Ioannidis, Stavros

Awarding institution:
King's College London

Derivative-based Financial Network Clearing: Transition from Intractability to Computational Solutions

Stavros D. Ioannidis

A thesis submitted for the degree of

Doctor of Philosophy

King's College London

Acknowledgements

I would like to express my deepest gratitude to my supervisors Bart de Keijzer and Carmine Ventre for supporting me and sharing their research experience with me. Their valuable guidance and the freedom they gave me to work at my own pace allowed me to become scientifically more mature over time. I am very thankful to them for the opportunity they gave me, for introducing me to this thesis' subject of study and for our collaboration. I am also very grateful to Kousha Etessami for his elaborative email on the complexity class FIXP and Argyris Deligkas for pointing me to the PURE-CIRCUIT problem. I also want to express my gratitude to my masters' supervisor Ioannis Caragiannis, for introducing me to the TCS community.

I would like to thank Anastasia for the love and enthusiasm she brings to our family. Her serenity and love for books inspire me throughout my life. I extend my gratitude to Dimosthenis and Alexandros for generously opening their home to me, for their companionship, and for the journey we shared to Oxford. For their friendship, memories and fun times, I would like to thank all my friends in Greece and London and especially my friends from Thessaloniki for the unforgettable trip to Ikaria.

I would like to acknowledge my parents for their invaluable gifts of life and education. I want to express my deepest gratitude for their unwavering love, support and dedication. No words can fully convey my feelings of appreciation and love for them. My father Dimitris showed me the beauty of mathematics, language, literature, history and life. I would like to thank him for his noble and kind example of living life. If it were not for him, I would have not taken this path in life. I want to thank my mother, Despoina for the dedication she showed in raising me in her young years and for the passion for poetry and art that I know I inherit from her. I dedicate the effort I putted in this work to my parents, Despoina and Dimitris. Σας ευχαριστώ από καρδιάς για όλα τα δώρα που μού έχετε προσφέρει.

All those moments will be lost in time, like tears in rain.

Blade Runner

Department of Informatics
King's College London

Derivative-based Financial Network Clearing: Transitioning from Intractability to Computational Solutions

A thesis submitted for the degree of
Doctor of Philosophy

Stavros D. Ioannidis

Supervisors:

Bart de Keijzer
Senior Lecturer, *King's College London*

Carmine Ventre
Professor, *King's College London*

Thesis committee:

Argyris Deligkas
Senior Lecturer, *Royal Holloway of London*

Kousha Etessami
Professor, *University of Edinburgh*

Abstract

Financial networks model a set of financial entities, such as firms or banks, interconnected by monetary obligations. Recent work has introduced to this model a class of obligations called Credit Default Swaps "CDS", a well-known type of financial derivative. The main computational challenge in financial systems is the "*clearing*" problem. This problem involves the task of determining insolvent firms and quantifying their exposure to systemic risk. The technical term used to describe this exposure is the "*clearing recovery rate*". In essence, the "*clearing*" problem involves computing the *clearing recovery rates* of all financial institutions in a given network.

It is established that the "*clearing*" problem, is computationally tractable when dealing with simple debt contracts [EN01], while existing techniques show PPAD-hardness for finding a weak approximate ϵ-*solution* when CDSes are involved, within an unspecified small range for ϵ [SSB17b]. This thesis addresses the "*clearing*" problem in financial networks containing simple debt contracts and credit default swaps and presents results on the following aspects of the problem:

- ***Exact Computation***: We establish that the exact computation version of the problem is FIXP-complete. We infer FIXP_a-completeness for finding a strongly (or "near") approximate solution as a direct consequence.

- ***Approximation Strength***: We establish an improved explicit inapproximability bound for computing weak (or "almost") approximate solutions.

- ***Algorithms and Heuristic***: We focus on two meaningful restrictions of the class of financial networks motivated by regulations: (i) the presence of a central clearing authority; and, (ii) the restriction to *covered* CDSes introduced in [SSB20]. We provide the following results.

  i) The PPAD-hardness for weak approximation persists when central clearing authorities are introduced.

  ii) An optimisation-based method for solving the "*clearing*" problem with central clearing authorities.

  iii) A simple polynomial-time algorithm when the two restrictions hold simultaneously.

Additionally we identify necessary structural conditions of the financial system that suffice for numerically irrational solutions to emerge. In the absence of these conditions, we study the

complexity of finding an exact solution, which we show to be a problem close to, albeit outside of, PPAD.

We also present supplementary results on the computational complexity of the "*clearing*" problem in financial networks with derivatives, whenever payment priorities among creditors are applied. This practically relevant model has only been studied from a game-theoretic standpoint. Specifically, we examine the "*clearing*" problem whenever firms pay according to a Singleton Liability Priority list and prove that it is also FIXP-complete. Finally, we provide a number of NP-hardness results for the task of selecting the optimal priority list, i.e, the list that optimises specific objectives of interest.

The aim of this thesis is to contribute to the literature on the "*clearing*" problem with debt contracts and credit default swaps. In this direction we present progress in existing knowledge as well as new results that establish the clearing problem as an important computational challenge at the intersection of finance and computation, having both theoretical and practical interest. To facilitate this connection, we highlight the progress made on the *"clearing"* problem and how this progress aligns with the progress marked in the field along the period of publishing these results. This work arranges thematically and provides a unified presentation of the material published in the following research papers:

- *Strong Approximations and Irrationality in Financial Networks with Derivatives* [IDKV22]
- *Financial Networks with Singleton Liability Priorities* [IDKV23b]
- *Clearing Financial Networks with Derivatives: From Intractability to Algorithms* [IdKV23a]

Contents

List of Figures

Chapter 1

Introduction

1.1 The Subject of Study

The International Monetary Fund claims that the global financial crisis (GFC) of 2007 has had long lasting consequences, including loss of growth, large public debt and even a decline of fertility rates, see [IMF]. Consequently, the need to assess the *systemic risk* of financial networks cannot be overstated. Below are just some exemplar questions that are paramount to the study of systemic risk in finance:

- How does the collapse of an established financial institution affect the stability of a financial market?
- Can a bank's exposure to such a contagion risk be efficiently measured by clearing authorities?
- What factors should legislatures consider when formulating economic policies?

If banks at risk of defaults could be easily identified in the complex network of financial obligations, then spread could be preemptively avoided with appropriate countermeasures such as bailouts from central banks or regulators.

In this context, financial networks have emerged as the framework of reference. These networks are modelled as graphs where vertices represent banks (or, more generally, financial institutions) and weighted arcs (u, v) model economic commitments from bank u to bank v. Each bank has also some assets external to the network, that can be used to pay its liabilities. The central computational challenge for financial networks, that underpins many, if not all, questions considered in the literature in this area, is the "*clearing*" problem. In this problem the task is to find a *clearing recovery rate vector*. This amounts to computing the percentage of the liabilities that each bank should pay if the system had to be cleared at once – intuitively, the smaller the ratio, the more exposed the bank is to systemic risk due to defaults in the network. If this ratio is bigger than 1 for a bank, then it will be able to pay its dues – in this case, we simply set its rate to 1. The banks that are in default have rates smaller than 1.

In their seminal work, Eisenberg and Noe [EN01], establish the mathematical formulation of the "*clearing*" problem. Their approach, which constitutes the baseline of research in this area, initiates the study of the problem from a fixed-point-computation perspective. The upshot of their work, is that *clearing recovery rate vectors* correspond to fixed points of a certain function f, whenever the liabilities are simple contracts requiring the payment of a certain value (known as notional) unconditionally. More importantly, they provide a polynomial time algorithm for computing the *clearing recovery rate vector* in case of networks with simple debt contracts.

However, Eisenberg and Noe's model ignores the issue of *financial derivatives* that may be present in the system. The deregulation allowing banks to invest in these products is considered by many as one of the triggers of the GFC. The introduction of *financial derivatives* to financial networks is due to Schuldenzucker, Seuken and Battiston [SSB20], where the focus is on a simple yet widely used class of conditional obligations known as credit default swaps (CDSes), the idea being to "swap" or offset a bank's credit risk with that of another institution. More specifically, a CDS has three entities: a creditor v, a debtor u and a reference bank z — u agrees to pay v a certain amount whenever z defaults. Whilst CDSes were conceived in the early 1990s [SM06] as a way to protect v from the insolvency of z for direct liabilities (i.e., a (v, z)-arc in the network), they quickly became a speculative tool to bet against the creditworthiness of the reference entity and have in fact been widely used both as a hedging strategy against the infamous collateralise debt obligations, whose collapse contributed to the GFC, and pure speculation during the subsequent Eurozone crisis.

The "*clearing*" problem in the presence of credit default swaps is somewhat less well-characterised. Similarly to the simple case, the "*clearing*" problem takes the form of a fixed-point computation over a more complex function f that always admits fixed points. The main observation for the new model on top of its involved combinatorial structure, is the numerical irrationality of the clearing recovery rates; a combination that puts the exact solution of the problem out of reach. To correctly define the computational question in this case, one then needs to resort to a notion of approximation.

Proposed approximation notions in the literature involve the computation of either a 'weak' (almost) fixed point x such that $|x - f(x)| < \epsilon$ along all dimensions or a 'strong' (near) fixed point y satisfying $|y - z| < \epsilon$ where z is such that $z = f(z)$ (that is, z is a fixed point). Both versions can be appealing for systemic risk in financial networks.

The main computational result regarding the "*clearing*" problem with credit default swaps was proved by Schuldenzucker, Seuken and Battiston [SSB17b] and establishes that there exists a small unspecified constant ϵ, for which computing *weak* approximations is PPAD-complete, whenever the banks pay proportionally i.e, they pay the same fraction of each liability, where this fraction is computed by dividing the banks' total assets by total debts. The result is based on a reduction from the ϵ-**GENERALISED-CIRCUIT**; a problem introduced in [CDT09] and proved in [Rub15] to be PPAD-hard for some unknown small constant ϵ. This statement excludes the possibility of constructing a polynomial time approximation scheme (PTAS) under the widely believed complexity-theoretic assumption that PPAD-hard problems do not have polynomial time algorithms.

1.2 Contribution

We enhance the study of the "*clearing*" problem for financial networks with credit default swaps from two complementary viewpoints; computational complexity and algorithms for exact computable solutions.

We argue that weak approximations, initially motivated and studied in [SSB17b], can be misleading in this domain, as the objective under this criterion is to find an "almost" fixed point (i.e., a point which is not too far removed from its image under the function). The proportion of liabilities that a bank has to pay provided by this estimate concept, might be very far off one (out of possibly multiple) actual rate thus changing the amount of bailout needed or even whether a bank needs rescue in the first place. A more useful (but more difficult) objective is to obtain a strong approximation, that is, a point that is geometrically close to an actual fixed point of the function. Such a risk estimate would be actionable for a regulator, as the error could be measured in terms of irrelevant decimal places. Furthermore, the banks themselves would accept the rate when the strong approximation guarantee is negligible, whereas weak approximations could significantly misrepresent their income and are subject to be challenged, legally or otherwise.

We settle the computational complexity of computing strong approximations to the "*clearing*" problem by showing that the problem is FIXP-complete (Sections 3.1.1, 3.1.2). In our reduction, we provide a series of financial network gadgets (Section 3.1.3), that are able to compute arithmetic operations over recovery rates. Not many FIXP-complete problems are known, although there are a few important natural such problems (three-or-more-player Nash equilibrium being a notable example [EY10]). Interestingly, more FIXP-complete results have been published during the period of writing this work (see Literature). Hardness reductions for this class tend to be rather technically involved and not straightforward. The hardness reduction that we provide here indeed has some technical obstacles as well, although it is quite natural at a high level.

In [SSB17b] the authors established that computing weak approximate fixed point for the clearing problem is PPAD-complete. Our work complements the work of [SSB17b] on the computational complexity of weakly approximate fixed points for the "*clearing*" problem with credit default swaps by establishing that computing *strong* approximate fixed points is harder than computing *weakly* approximate fixed points, which holds due to PPAD being equal to the class Linear-FIXP, which is a restriction of FIXP, and this makes PPAD (indirectly) a subclass of FIXP (see Section 3.1.1).

Main Theorem 1 (cf. Theorem 3) *Computing a clearing vector (*strong *approximate clearing vector) in a financial network with debt contracts and credit default swaps is FIXP-complete (FIXP$_a$-complete).*

Furthermore, we want to understand whether there are values of ϵ for which computing a suitable ϵ-approximation of the *clearing recovery rates* can be done in polynomial time. In particular, we are interested in the intractability of the problem; for which values of ϵ do the hardness results carry over? We exploit the connection of the "*clearing*" problem to the PPAD complexity class, along with the recent advancements in the area of total function search problems TFNP [MP91], regarding constant inapproximability. We study *weak* approximations. It was previously established that computing weak ϵ-approximate clearing vectors is PPAD-hard for a small unknown constant ϵ [SSB17b]. We prove the following result.

Main Theorem 2 (cf. Theorem 6) Computing ϵ-*weak* approximations of clearing recovery rates is PPAD-hard for $\epsilon \leq \frac{3-\sqrt{5}}{16} \approx 0.048$.

The proof leverages **PURE-CIRCUIT** – a toolkit introduced by Deligkas, Fearnley, Hollender and Melissourgos [DFHM22] – for showing constant inapproximability in PPAD. The problem features a set of variables which can take one of three different values 0, 1 or garbage and a circuit constructed from connecting gates of the following three types: NOT, OR and PURIFY. The first two basically follow boolean logic whereas the PURIFY gate has one input and two outputs and makes sure to duplicate the pure bit in input or produce at least one pure bit in output if the input is garbage. The main result of [DFHM22] is that **PURE-CIRCUIT** is PPAD-complete.

We construct a direct reduction from **PURE-CIRCUIT** to the "*clearing*" problem. The proof is technically involved and can be divided in the following parts:

- An encoding of the recovery rate space $[0, 1]$ into three disjoint subintervals that intuitively map recovery rates to either a pure bit or garbage.

- The construction of three "financial gates" whose behaviour under *weak* approximate clearing vectors must simulate the function of **PURE-CIRCUIT** gates after decoding the recovery rates of specific banks.

Subsequently, we evaluate the effectiveness of policies introduced in the wake of economic crises, the introduction of central CDS debtors (CCDs), a construct reminiscent of a *central clearing counterparty* (CCP), and the *ban of naked CDSes.* Both the regulatory frameworks in EU and US require the use of a CCP for a large part of the over-the-counter derivatives market [SSB20]. This means that if two banks u and v want to sign a derivative (CDS in our context) they need to involve a CCP which will sign one CDS contract with u and another CDS with v. A CCP is typically so well capitalised that it can absorb shocks in the market. A central CDS debtor is a CCP which is not creditor of any CDS contract, a strengthening of the current notion of CCP where the risk of the original CDS from u to v is not only absorbed downstream towards v but also upstream vis-a-vis u (e.g., the original CDS is only substituted with a CDS between the CCP and v whereas the liability with u is a simple debt contract). Such a policy requirement restricts quite significantly the network structure and rules out the hope for efficient computation of (good approximations of) clearing rates. Interestingly, under the presences of CCDs, all clearing vectors are rational. Unfortunately, our inapproximability result applies also to the case in which CCDs ought to be used.

We propose an approach for computing exact clearing vectors for financial networks that involve CCDs. We devise a Mixed-Binary Linear Program (Section 4.1.1), where within this program, there is a combination of real-valued variables for the recovery rates and binary decision variables tailored to indicate a bank's solvency status. The feasible solutions of this program correspond to the clearing recovery rate vectors of the network. This framework is highly adaptable for optimising any linear objective function of interest related to the clearing vector. An immediate implication is an exponential-time algorithm for computing clearing recovery rate vectors when CCDs are mandated.

Main Theorem 3 (cf. Theorem 8) The task of computing an exact clearing recovery rate vector that optimises a given linear objective function (whose variables are the recovery rates) in the setting of a financial network that contains both debt and CDS contracts where moreover there exist one CCD, i.e, when a CCD is mandated, admits an exponential time algorithm.

A *naked CDS* [SSB20] is a purely speculative contract; its counterparties do not have any other interest in the reference bank if not the CDS itself. However, regulators could ban their existence and only allow to buy a CDS if a corresponding (debt) exposure exists–i.e., to only have so-called *covered* CDS [SSB20]. We exploit the topological structure of financial networks that contain only covered CDS contracts and CCDs to compute an exact clearing recovery rate vector in polynomial time.

Main Theorem 4 (cf. Theorem 9) The task of computing an exact clearing recovery rate vector in the setting of financial networks that contain debt and only covered CDS contracts, where moreover there exist one CCD, admits a polynomial time algorithm.

The FIXP_a-hardness of the *strong* approximation problem indicates that there is an additional numerical aspect contributing to the hardness of the problem, which is not present in the *weak* approximation problem (where the hardness is of a combinatorial nature, due to the reducibility to the end-of-the-line problem which is canonical to PPAD). For the *strong* approximation problem, the nature of the underlying function for which we want to find the fixed points requires, in particular, the multiplication operation, which ultimately accounts for irrationality and super-polynomial numerical precision being necessary in order to derive whether a given point is a *strong* approximation to a clearing vector.

Additionally, we turn our attention towards numerically irrational solutions with the goal to determine the source of irrationality and understand when it is possible to compute the clearing recovery rate vector exactly in the form of rational numbers. We identify a structural property of cycles in an opportunely enriched network that leads to a financial network with a unique clearing recovery rate vector whose coordinates contain irrational real values. This property exactly differentiates the CDSes that produce and propagate irrationality of the recovery rates, that we call "switched on", from those that do not, termed "switched off". We prove the following characterisation of irrationality.

> **Main Theorem 5 (cf. Theorems 10,11)** *If the financial network has only "switched on" CDSes in a cycle* and *the cycle cannot be shortcut with paths of length at most three*[1] *then there exist rational values for debt and asset values for which the recovery rate vector is unique and irrational. Conversely, if every cycle of the financial network does not have any "switched on" CDSes then we can compute rational recovery rates in a polynomial number of operations, provided that we have oracle access to* PPAD*.*

The proof of irrationality uses a type of graph "algebra" (that is, a set of network fragments and an operation on them) that is able to generate all the possible cycle like structures with the property above, which uncovers a connection between the network structure of the "*clearing*" problem and the roots of non-linear equations.

For the opposite direction, we provide an algorithm that exploits the acyclic structure of financial networks with solely "switched off" CDSes. This algorithm iteratively computes the recovery rates of each strongly connected component of the network. We show that even for the simpler topologies of the financial system under consideration, the problem remains PPAD-hard, hence the need for the oracle access to PPAD.

[1]The length-at-most-three condition is restated in the form of a more refined condition in the respective technical Sections that lead to this result.

1.3 Significance

This thesis builds upon and enhances a set of previously established complexity results published in [SSB17b]. In addition to the algorithmic results presented in [SSB17a], we establish new algorithmic methods for certain restrictions of the "*clearing*" problem when credit default swaps are involved.

In reference to Main result 1, the FIXP-completeness of the problem is an important result that was missing from the literature. It provides the complete complexity characterisation of the problem with immediate impact on the complexity of approximate solutions. Essentially it establishes the central role of the "*clearing*" problem with CDSes in the FIXP class, an important class related to Game Theory [EY10].

What is notable is that proving FIXP-completeness results is based on *SL*-reductions, a methodology initially suggested for establishing hardness. Here, the significance of our result is more technical, in the sense that our presented reduction for the FIXP-hardness is somewhat more direct than in previous work we are aware of, in the sense that it starts from the algebraic circuit defined by an arbitrary problem in FIXP and employs two main steps: We firstly force the outputs of all gates in the circuit to be in the unit cube, by essentially borrowing arguments from [EY10], after which we produce a series of network gadgets that preserve gate-wise the computations of said circuit.

The results of Main Theorem 2, concerned with the intractability of the "*clearing*" problem, follow the advancements regarding hardness of approximation in PPAD marked by the introduction of the **PURE-CIRCUIT** problem [DFHM22]. This problem is received by the community as a novel tool replacing the ϵ-**GENERALISED-CIRCUIT** problem defined in [CDT09] in establishing inapproximability bounds for problems in PPAD. The importance of this progress, apart from the simplicity of the new tool, is the transition from proving existence of an unspecified small inapproximability bounds (that results when using ϵ-**GENERALISED-CIRCUIT** which was proved PPAD-hard in [Rub15]) to establishing explicit and higher inapproximability bounds when using **PURE-CIRCUIT**. The "*clearing*" problem with *credit default swaps* is proposed by the authors in the introduction of [DFHM22], to fall into the category of problems for which a transition from an undefined small constant inapproximability parameter to an explicit higher bound can be achieved via their proposed tool.

Main Theorem 2 provides the first explicit inapproximability bound for the "*clearing*" problem for financial networks with credit default swaps. The significance of the result is enhanced when comparing it with the state of the art bound. The initial method for proving PPAD-hardness of ϵ-weak approximate clearing vectors, established in [SSB17b], presents a reduction from the ϵ-**GENERALISED-CIRCUIT** to the clearing problem. The resulting statement suggests that there exist a constant $\epsilon > 0$, such that the task of computing ϵ-weak approximate clearing vectors is PPAD-hard. In our work, we make the state of the art bound explicit. First we suggest a chain reduction from **PURE-CIRCUIT** to ϵ-**GENERALISED-CIRCUIT** to the clearing problem that leads to a bound of $\epsilon < 1/150 \approx 0.0067$ (Section 3.2.3). Subsequently we present a direct reduction from the **PURE-CIRCUIT** problem to the *clearing* problem. In comparison to the state of the art, Theorem 6 is a seven-fold improvement. An important added value of this result is that it holds for any "reasonable" *payment scheme* (see Corollary 1).

Furthermore, the presented optimisation approach that leads us to Main Theorem 3, could be received as a versatile tool for proving improved lower bounds and upper bounds. The exponential Algorithm 1 proposes a clearing mechanism when central clearing authorities are present in the system. It is important to emphasise that the exponent in the algorithm's running time is contingent solely on the number of banks in the network and remains unaffected by the actual size of the parameters within the networks structure (see discussion under Algorithm 1).

We see the complexity and irrationality results of Main Theorem 5 as important analytical tools that legislators can use to regulate *financial derivatives.* We contribute to the ongoing debate in the US and Europe about whether speculative uses of CDSes should be banned. In particular our results support, from a computational point of view, the call to ban so-called "naked" CDSes (as already done by the EU for sovereign debt in the wake of the Eurozone crisis, see [EUB]). In a "naked" CDS its creditor and debtor have no direct liabilities with the reference entity which arguably makes such contracts purely speculative. The simple efficient algorithm we propose (see Algorithm 2), when high-capitalised clearing authorities are coupled with *covered CDSes* [SSB20, SSB17a], aligns with legislative policies on prohibiting the use of "naked CDSes" and previous work published in [SSB17b].

1.4 Technical and Conceptual Innovations

It is worth highlighting a specific technical challenge that we overcome in the proof of Theorem 3, as it sheds further light on FIXP, and in particular, on the operator basis of the algebraic circuits that are used to define the class. It is known that the circuit of problems in FIXP can be restricted without loss of generality to be built on the arithmetic basis $\{\max, +, *\}$ [EY10], whereas restricting the internal signals of the circuit to the unit cube (with the toolkit developed in [EY10]) needs some further operators, including $/$. For our optimisation problem to be in FIXP, we need the rather mild and realistic assumption that our instances are *non-degenerate* as defined in [SSB17b]. The function of which the fixed points define the recovery rates of non-degenerate instances is well defined, where the non-degeneracy is needed to avoid a division by 0. It turns out that non-degeneracy is incompatible with division being part of the FIXP operator basis, i.e., it seems difficult to build such a financial network that in any sense simulates a division of two signals in an algebraic circuit. To bypass this problem, our proof shows that it is possible to substitute $/$ in the basis with the square root operator, $\sqrt{\cdot}$, whilst keeping the function well defined. This substitution can be used to simulate division with constant large powers of 2, and this turns out to be sufficient to omit the $/$-operator (i.e., arbitrary division). This observation might be useful for other problems where division is problematic to either define the fixed point function, or the reduction.

The proposed reduction for the PPAD-hardness result of Theorem 6 is optimised, in the sense that it generates a family of possible encodings for the recovery rate values, upon which we choose the one that maximises the inapproximability parameter ϵ. The intricate structure of the proposed networks combined with the restriction of computations to intervals within the $[0, 1]$ range, significantly increases the complexity of the analysis. To bypass this obstacle we define and use a first-order language denoted as $L(\mathrm{R}, \mathrm{F}, \mathrm{C})$. This language is designated to handle combined arithmetic operations involving intervals and numbers within the range $[0, 1]$ and combines concise representations with a high level of expressiveness. The upshot of Theorem 6 is that if we were able to approximate the "*clearing*" problem better than (roughly) 0.048 then the encoding would guarantee that we could solve **PURE-CIRCUIT**.

Main Theorem 5 indirectly aims at characterising the "rational fragment" of FIXP. A couple of observations can be drawn. Firstly, our sufficiency conditions for irrationality suggest that any such characterisation needs to fully capture the connection between the fixed point condition and the rational root theorem; our proof currently exploits the cyclical structure of networks with "switched on" CDSes to define one particular quadratic equation with irrational roots. Whilst this captures a large class of instances, more work is needed to give a complete characterisation.

Secondly, our sufficiency conditions for rational solutions highlight a potential issue with their representation. Due to the operations in the arithmetic basis, most notably multiplication, these solutions can grow exponentially large (even though each call to the PPAD oracle returns solutions of size polynomial in their input). This observation establishes a novel connection between the Blum-Shub-Smale computational model [BCSS98] (wherein the size to store any real number is assumed to be unitary and standard arithmetic operations are executed in one time unit), the rational part of FIXP, and PPAD.

1.5 Literature

Systemic risk and contagion in financial networks have been studied extensively in the literature [AOTS15, EGJ14, GY15, HK12, HK16, HZHW12, CFS05, EGJ14, JP21, JP24]. Previous work models a financial network as a setup of interconnected nodes, representing economic firms, in an arc-weighted graph where arcs represent debt obligations from one firm toward another. Among the first papers on systemic risk in financial networks, Eisenberg and Noe [EN01], study the problem of finding a clearing payment vector for a financial system that contains only simple debt contract liabilities. They prove, applying Tarski's fixed point theorem, that such payment vectors always exist and provide a polynomial time algorithm for computing one. A variation of the original model by Eisenberg and Noe with the addition of default costs is presented by Rogers and Veraart in [RV13]. Concepts of financial systems with simple debt contracts were investigated in [SS21, PW21a, HW22]

Financial systems admitting both debt contracts and credit default swaps are introduced by Schuldenzucker, Seuken and Battinston in [SSB20]. They establish that clearing recovery rate vectors always exists when studying models in [EN01] whereas that's not the case for models with default costs in [RV13] where they establish that deciding whether one clearing vector exists is NP-hard presenting a reduction from CIRCUIT-SATISFIABILITY [Pap94a]. In the same paper, the authors establish that there might exist multiple such clearing recovery rate vectors. This situation leads to ambiguity regarding the financial state of the institutions. Subsequently in [SSB17b] the authors are interested in the problem of computing a clearing recovery rate vector in models where existence is guaranteed. Early in their paper, they construct a simple instance of a financial network whose solution is proved to be irrational, thus focusing on the problem of computing an approximate clearing recovery rate vector. Their main result is that almost-approximating the clearing recovery rate vector is PPAD-complete, meaning that no PTAS exists unless P = PPAD. The notion of *covered* CDS contracts is defined in [SSB20]. In [SSB17a] the authors provide a FPTAS for computing an approximate clearing vector in financial networks that contain only covered CDS contracts.

Work has been published on the computational complexity of finding the clearing recovery rate vector that maximises/minimises specific objective functions regarding the financial system. Papp and Wattenhofer in [PW22], establish that even if a financial regulator could efficiently compute the set of clearing recovery rate vectors for a financial system, it is still NP-hard to find the clearing vector that minimises the number of defaulting banks and the amount of unpaid debt in the system. Moreover they prove that finding the vector that is most preferable by the largest set of banks, the vector that is preferred by a specific bank as well as the vector with the best equity distribution is also NP-hard to approximate within some constant factor. Further studies on financial networks with CDSes are presented in [PW22, PW20, PW21b, LPT17].

Work has also been published on the incentives of banks in financial networks. The authors of [BHS20, HW22, KKZ21a, KKZ22, KKZ23] study the price of anarchy and stability in games where insolvent banks can strategically decide how to pay off their debts. The strategic aspects of modifying the structure of the network (by, e.g., writing off a debt) to a bank's advantage are considered in [PW22, KKZ22]. The strategic framework of prioritising debts in financial networks with CDSes is due to Papp and Wattenhofer [PW22].

The complexity of total search function problems TFNP was first considered by Megiddo and Papadimitriou [MP91] while the class PPAD was defined by Papadimitriou [Pap94b, Yan09] and gained significant attention from the work of Daskalakis, Goldberg and Papadimitriou [DGP09] and Chen, Deng and Teng [CDT09] for computing Nash equilibrium in strategic form games. The FIXP complexity class was introduced by Etessami and Yannakakis [EY10, Yan09] for studying *strong* approximation to Nash equilibrium in strategic form games. Advancements related to the FIXP class can be found in [EHMS14, HL18, BHH21, FGH$^+$23, FHHH21, GH21, IDKV22, IDKV23b, HL21]. **GENERALISED-CIRCUIT** was introduced by [CDT09] and used by Rubinstein [Rub15] to establish a first constant inapproximability method. Most recently Deligkas, Fearnley, Hollender and Melissourgos [DFHM22] suggested **PURE-CIRCUIT**, a tool for showing stronger constant inapproximability results for PPAD. Applications of the problem are presented in [DFHM22, DFHM23, IdKV23a, DH24, FGHK24].

Chapter 2

Preliminaries

Overview

This chapter presents the essential introductory concepts. In Section 2.1 we model financial networks involving derivatives as edge-weighted directed graphs, with nodes linked by two types of arcs that represent simple debt contracts or credit default swaps (CDS). We provide a high level overview of the primary payment schemes employed by economic entities to settle debts and present a model for capturing the dynamics of the system under these schemes. We conclude with the definition of the *clearing recovery rate vector*, which is the central notion of focus in this work.

In Section 2.2 we formally define the computational problem of finding a clearing vector in a financial network with debt contract and credit default swaps as **CDS-CLEARING** or **CDS-PRIORITY-CLEARING** depending on the applied payment scheme. A fixed point computation perspective on the problem emerges through a simple analysis, facilitating the transition from the financial aspect of the problem to the fixed point computation facet of the problem. In Section 2.2.2 we highlight the existence of instances that admit numerically irrational solutions. This motivates the study of two approximate notions for the problem presented in Section 2.2.3: *weak* and *strong*.

2.1 Financial Networks with Derivatives

2.1.1 Financial Networks

We denote by $N = \{1, \dots, n\}$ a set of n financial institutions, which we will call *banks*. Each bank i possesses some amount of non-negative *external assets*, denoted by $e_i \in \mathbb{Q}_{\geq 0}$ and let $e = (e_i)_{i \in N} = (e_1, \dots, e_n)$ be the vector of external assets. We consider two types of liabilities banks can have towards other banks: debt contracts and credit default swaps (CDSes).

Definition (Debt Contract)**.** *A* debt contract *between two banks requires one of the banks, named the* debtor *(or* writer*), to pay a certain amount to the other bank, named the* creditor *(or* holder*). The value that needs to be paid from the debtor $i \in N$ to the creditor $j \in N$ is denoted by $c_{i,j} \in \mathbb{Q}_{\geq 0}$. We denote by $\mathcal{DC}$ the set of all pairs of banks participating in a* debt contract*; if $(i, j) \in \mathcal{DC}$, then there exists a* debt contract *of value $c_{i,j}$ where i is the debtor and j is the creditor. No bank has a debt contract with itself.*

Definition (Credit Default Swap)**.** *A* credit default swap (CDS) *involves three banks, where a debtor owes money to a creditor, similarly to a* debt contract*. However, the amount of money owed is dependent on whether a third bank called* reference bank *is in default. A bank is in default when it has insufficient assets to pay its total amount of debts. More formally, bank i's* recovery rate *$r_i \in [0, 1]$ is the fraction of debts that the bank is able to pay off, given its total assets (defined as the sum of e_i and the payments it receives from other banks). Bank i is said to be in default if and only if $r_i < 1$. In case a reference bank $R \in N$ of a certain CDS is in default, i.e, $r_R < 1$, the debtor $i \in N$ of that CDS is obliged to pay the creditor $j \in N$ an amount of $(1 - r_R) \cdot c^R_{i,j}$, where $c^R_{i,j} \in \mathbb{Q}_{\geq 0}$ is a specified amount associated to the particular CDS between banks i, j, with reference R. We denote by $\mathcal{CDS}$ the set of all triplets of banks participating in a CDS contract; if $(i, j, R) \in \mathcal{CDS}$, then there exists a CDS contract where i is the debtor bank, j is the creditor, and R is the reference bank. In a CDS contract all three banks are distinct.*

The value $c_{i,j}$ (or $c^R_{i,j}$) of a debt contract (or CDS) is referred to as the *notional* of the contract. The notionals of all debt and CDS contracts are contained in a three-dimensional $(n \times n \times n)$ matrix c. For conventional purposes we assume that the contract notional $c_{i,j}$ of a debt contract from a bank i to a bank j is stored in the entry (i, j, i) of the matrix c and that the notional $c^R_{i,j}$ of a CDS contract from bank i to bank j in reference to bank R is stored in the (i, j, R) entry of c. Moreover from the debt contract definition it holds that $(i,i,i) = 0$ for all banks i. Finally since all banks in a CDS contract are distinct it holds that $(i,j,j) = 0$ for all banks i, j. For simplicity we will refer to the entry in the matrix c of a debt contract (i,j,i) simply as (i,j). This convention avoids any misconception since all banks in a CDS contract are distinct thus a tuple (i,j,i) can never correspond to a CDS contract. The non-existence of a debt contract or CDS is equivalent to having a contract with corresponding notional 0, and multiple contracts between identical pairs or triplets of banks can be merged into a single contract with notional equal to the sum of the individual contract's notionals. This brings us to the formal definition of a financial network (or system).

Definition 1 (**Financial Networks**). *A financial network (system) is a triplet (N, e, c), where $N = \{1, .., n\}$ is a set of banks, $e = (e_i)_{i \in N} \in \mathbb{Q}^n_{\geq 0}$ is the vector of external assets, and $c \in \mathbb{Q}^{n \times n \times n}_{\geq 0}$ is a three-dimensional matrix of contract notionals.*

To visualise a network $\mathcal{F} = (N, e, c)$, we use a coloured directed graph-like structure, which we call the *contract graph* as follows: We first construct a directed multigraph $G_{\mathcal{F}} = (V, A)$, where $V = N$ and A is the multiset-union of the set $A_0 = \{(i,j) \mid c_{i,j} \neq 0\}$ and the sets $A_k = \{(i,j) \mid c^k_{i,j} \neq 0\}$ for all $k \in N$. Furthermore, we colour each arc through a function $t : E \to \{\text{blue}, \text{orange}\}$, where $t(e) = \text{blue}$ iff $e \in A_0$ and $t(e) = \text{orange}$ otherwise. For all $(i, j, R) \in \mathcal{CDS}$ we draw a dotted orange line from node R to arc $(i,j) \in A_R$, to denote that R is the reference bank of the CDS between i and j. We are using the terms *bank* and *node* interchangeably.

In the resulting graphical notation, we label an arc with the notional of the corresponding contract and a node with the external assets of the corresponding bank, in green font. Below we present an example of a financial network illustrated in Figure 2.1.

Example 1. *Let $\mathcal{F} = (N, e, c)$ be a financial network where $N = \{1, 2, 3, 4, 5, 6\}$ is the set of Banks and $e = (1, 0, 0, 1, 0, 0)$ the external assets vector (i.e, the external assets of Bank 1 are $e_1 = 1$ and for Bank 3 we have $e_3 = 0$). The contract graph of the financial network is shown in Figure 2.1. Bank 1 is the debtor of two debt contracts, one towards Bank 2 and one towards Bank 3, represented by blue arcs. The notionals of these contracts are $c_{1,2} = 1$, $c_{1,3} = \frac{1}{2}$. Bank 3 is the debtor in one debt contract towards Bank 5, where $c_{3,5} = \frac{1}{2}$. Bank 4 is only a debtor towards Bank 6, where $c_{4,6} = \frac{1}{2}$. There are two CDSes in the system, one from Bank 2 to Bank 4 in reference to Bank 3 and one from Bank 5 to Bank 6 in reference to Bank 4. Formally $\mathcal{CDS} = \{(2,4,3), (5,6,4)\}$. For $(5,6,4)$ the notional is $c^4_{5,6} = 1$ and for $(2,4,3)$ the notional is $c^3_{2,4} = \frac{2}{3}$.*

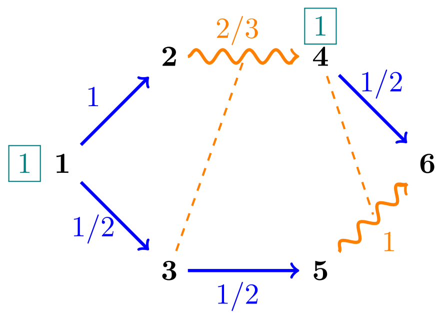


Figure 2.1: The contract graph of a system with six banks, four debt contracts and two CDSes.

2.1.2 Payment schemes

Banks pay their debts according to a prespecified payment scheme. We consider payment schemes that satisfy two fundamental conditions called Limited Liability and Absolute Priority introduced by Eisenberg and Noe is [EN01]. For simplicity we provide the informal definition of the two conditions as presented in [SSB17a][1].

(i.) *Limited Liability*: A bank with sufficient assets to pay all its debts is obliged to do so.

(ii.) *Absolute Priority*: A defaulting bank is obliged to submit all of its assets to its creditors.

The most studied payment scheme is the *Proportional payment scheme*, where each bank i submits the r_i proportion of each liability, leaving a $(1 - r_i)$ fraction of each liability unpaid.

Another scheme of interest is the *Singleton Liability Priority Payment-(SLPP)*, where for a given financial system each bank i defines a total order over its contracts. We denote the priority list of i as $P_i = (i_1 \mid i_2 \mid \ldots \mid i_{\text{outdeg}(i)})$, where i_k stands for the kth contract in the order, or kth *priority* of node i, and outdeg(i) denotes the outdegree of node i in the contact graph. The payments under this scheme are now formed through an iterative process where each bank pays off its liabilities, one after the other, according to the ordering given in its priority list. We denote by c_{i_k} the contract notional of the kth priority and denote by $\mathcal{P} = (P_1, \ldots, P_n)$ a profile of Singleton Liability Priority lists. We denote a financial system $\mathcal{F}$ coupled with a Singleton Liability Priority profile $\mathcal{P}$ as a pair $(\mathcal{F}, \mathcal{P})$. We provide an example that illustrates the scheme below.

[1]See page 10 of [SSB17a].

Example 2. *The financial network of Figure 2.2 consists of six banks, $N = \{1, 2, 3, 4, 5, 6\}$. Banks 2 and 5 have external assets $e_2 = e_5 = 1 - c$, for some constant $c \in (0, 1)$, while all other banks have zero external assets. The set of debt contracts is $\mathcal{DC} = \{(2, 3), (5, 4)\}$ and the set of credit default swaps is $\mathcal{CDS} = \{(2, 1, 5), (5, 6, 2)\}$. All contract notionals are set to 1. For example, $c_{2,3} = c^5_{2,1} = 1$. Bank 2 has two candidate Singleton Liability Priority lists, one is $P^1_2 = ((2, 3) \mid (2, 1, 5))$, where $2_1 = (2, 3)$ with contract notional $c_{2_1} = c_{2,1} = 1$ and $2_2 = (2, 1, 5)$ with contract notional $c_{2_2} = c^5_{2,1} = 1$. The other one is $P^2_2 = ((2, 1, 5) \mid (2, 3))$ where $2_1 = (2, 1, 5)$ with $c_{2_1} = c^5_{2,1} = 1$ and $2_2 = (2, 3)$ with $c_{2_2} = c_{2,3} = 1$. Symmetrically one can derive the lists for node 5.*

$$\mathcal{P} = (\underbrace{((2,3) \mid (2,1,5))}_{P^1_2}, \underbrace{((5,4) \mid (5,6,2))}_{P^1_5})$$

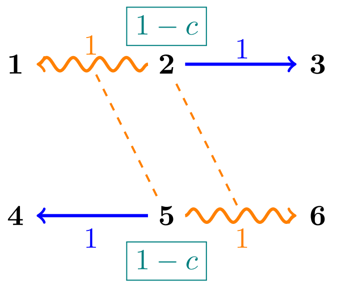


Figure 2.2: The contract graph of a financial network with the profile $\mathcal{P} = (P^1_2, P^1_5)$.

2.1.3 Clearing Recovery Rate Vectors

This thesis studies the task of computing for a given financial system, for each bank, the proportion of liabilities that it is able to pay. This proportion is captured by the notion of the *recovery rate* of a bank, introduced earlier in the definition of CDS contracts. For each bank i we associate a variable $r_i \in [0, 1]$, where $r_i = 1$, indicates that bank i can fully pay, while $r_i < 1$ indicates that i is in default and pays only the r_i proportion of liabilities. Recall that the amount a debtor i has to pay to creditor j in a CDS contract with reference bank R is given by $(1 - r_R) \cdot c^R_{i,j}$. Consequently, the liabilities of CDS debtor banks, are specified upon a given recovery rate vector.

In a technical sense, for any given vector $r \in [0, 1]^n$ representing recovery rates, it is possible to precisely define a financial institution's *dynamics*, i.e, liabilities, payments, and assets.

$\boxed{\textbf{Liabilities}}$ The liability of a bank $i \in N$ to a bank $j \in N$ under r is denoted by

$$l_{i,j}(r) = c_{i,j} + \sum_{k \in N} (1 - r_k) \cdot c^k_{i,j}.$$

That is we sum up the liabilities from all debt and CDS contracts between i and j. We denote by $l_i(r)$ the total liabilities of i:

$$l_i(r) = \sum_{j \in N} l_{i,j}(r). \tag{2.1}$$

In the *Singleton Liability Priority* payment scheme, we denote by $l_{i_k}(r)$ the kth *liability priority* of node i. Similarly if $i_k = (i,j) \in \mathcal{DC}$ for some $j \in N$, then $l_{i_k}(r) = c_{i,j}$ and if $i_k = (i,j,R) \in \mathcal{CDS}$ for some $j, R \in N$, then $l_{i_k}(r) = (1 - r_R) \cdot c^R_{i,j}$.

Proportional Payments The payment bank i submits to bank j under r is denoted by $p_{i,j}(r)$ and it holds that:

$$p_{i,j}(r) = r_i \cdot l_{i,j}(r). \tag{2.2}$$

The total payment that bank i submits to the network is

$$p_i(r) = r_i \cdot l_i(r).$$

Singleton Liability Priority Payments Bank i can fully pay its kth priority only if it has sufficient assets left after paying off the liabilities corresponding to priorities $i_1, \ldots, i_{k-1}$. We denote by $p_{i_k}(r)$ the payment of bank i to its kth priority, and by $a_i(r)$ its assets, which are defined as the external assets it possesses plus all incoming payments received from its debtors (see below for a more formal definition). Under the singleton liability priority list payment scheme:

$$p_{i_k}(r) = \max\left\{0, \min\left\{l_{i_k}(r), a_i(r) - \sum_{k' < k} l_{i_{k'}}(r)\right\}\right\}. \tag{2.3}$$

Similarly, we denote by $p_{i,j}(r)$ the payment of bank i to bank j under recovery rate vector r. For each $i, j \in N$ we define the set of contracts from i to j indexed by their priority position in the list of bank i as $\mathcal{C}^i_j = \{i_k \mid i_k$ is a contract with debtor i and creditor j where $k \leq (outdeg(i)\}$. The set $\mathcal{C}^i_j$ contains all different contracts with debtor the bank i and creditor the bank j. For example two such contracts might be two credit default swaps $i_1 = (i, j, R_1)$ and $i_2 = (i, j, R_2)$ with R_1 and R_2 being different banks of the network where the i_1 is contract is the first priority of bank i and i_2 is the second is the second priority of bank i. It holds that $p_{i,j}(r) = \sum_{i_k \in \mathcal{C}^i_j} p_{i_k}(r)$.

The total payment made by a bank is the sum of its individual payments to its priorities which is equal to the total sum of its payments to its creditors. Therefore, the following equations hold:

$$p_i(r) = \sum_{k=1}^{\text{outdeg}(i)} p_{i_k}(r) = \sum_{j \in N} p_{i,j}(r). \tag{2.4}$$

$\boxed{\textbf{Assets}}$ The assets of a bank i under r are the total amount of money it possesses through its external assets and incoming payments submitted by other banks. The payment function is adapted regarding the applied payment scheme.

$$a_i(r) = e_i + \sum_{j \in N} p_{j,i}(r). \tag{2.5}$$

Although the dynamics are defined upon any arbitrary vector r, the significance of the recovery rate lies in representing the proportion of a bank's liabilities that it can settle using its assets. Thus we are interested in vectors where r_i is 1 if i's assets exceed its liabilities, and otherwise equal to the ratio of assets by liabilities. We call such vectors clearing recovery rate vectors.

Definition 2 (**Clearing recovery rate vector-*CRRV***)**.** *Given a financial system $\mathcal{F} = (N, e, c)$, a recovery rate vector r is called clearing if and only if for all banks $i \in N$,*

$$r_i = \begin{cases} \min\left(1, \dfrac{a_i(r)}{l_i(r)}\right), & \text{if } l_i(r) > 0 \\ 1, & \text{if } l_i(r) = 0 \end{cases} \tag{2.6}$$

where the dynamics adjust according to the corresponding payment scheme.

To demonstrate the above definition we compute the *clearing vector* for the financial networks presented in Examples 1 and 2.

Example 1 (continued)**.** *We compute the clearing recovery rate vector for the system in Figure 2.1. For Bank* 1 *it holds that $a_1(r) = e_1 = 1$ and $l_1(r) = l_{1,2}(r)+l_{1,3}(r) = c_{1,2}+c_{1,3} = 1+1/2 = 3/2$, thus from Equation (2.6) we get that $r_1 = \min\{1, 2/3\} = 2/3$. For Bank* 2 *we have that $a_2(r) = p_{1,2}(r) + e_2 = r_1 \cdot l_{1,2}(r) + e_2 = 2/3$. Also $l_2(r) = (1-r_3) \cdot c^3_{2,4} = (2/3) \cdot (1-r_3)$. So to compute r_2 we first need to compute r_3. For Bank 3 it holds that $a_3(r) = p_{1,3}(r) + e_3 = r_1 \cdot l_{1,3}(r) + e_3 = (2/3) \cdot (1/2) = 1/3$, and $l_3(r) = c_{3,5} = 1/2$, finally $r_3 = \min\{1, a_3(r)/l_3(r)\} = \min\{1, 2/3\} = 2/3$. Since Bank 3 is in default, the CDS* $(2, 4, 3)$ *is activated and counts towards the liabilities of Bank 2, so that*

$l_2(r) = (1-r_3)\cdot c^3_{2,4} = (2/3)\cdot(1-2/3) = 2/9$. Finally $r_2 = \min\{1, a_2(r)/l_2(r)\} = \min\{1,3\} = 1$, i.e., Bank 2 can fully pay its liabilities. For Bank 4 it holds that $a_4(r) = p_{2,4}(r)+e_4 = r_2\cdot(1-r_3)\cdot c^3_{2,4}+1 = 1\cdot 2/3\cdot(1-2/3)+1 = 11/9$, $l_4(r) = 1/2$ and $r_4 = \min\{1, a_4(r)/l_4(r)\} = 1$. Bank 4 is not in default, so the CDS $(5,6,4)$ is not activated and Bank 5 has no liability towards Bank 6.

Example 2 (continued)**.** *Let $c = 1/4$ in Figure 2.2. Let $\mathcal{P} = (P^1_2 = ((2,3) \mid (2,1,5)), P^1_5 = ((5,4) \mid (5,6,2)))$. Both Banks 2 and 5 receive no payment from any other Bank thus their assets are defined as $a_2 = e_2 = 1-c$ and $a_5 = e_5 = 1-c$. For Bank 2, given P^1_2, we get that $l_{2_1} = l_{2,1} = c_{2_1} = c_{2,3} = 1$ and $l_{2_2} = l_{2,1} = (1-r_5)\cdot c^5_{2,1} = (1-r_5)$, thus the total liabilities for Bank 2 are $l_2 = l_{2_1}+l_{2_2} = 2-r_5$. For Bank 5 we get that $l_{5_1} = l_{5,4} = c_{5_1} = c_{5,4} = 1$ and $l_{5_2} = l_{5,6} = c_{5_2} = (1-r_2)\cdot c^2_{5,6} = 1-r_2$, thus the total liabilities for Bank 5 are $l_5 = l_{5_1} + l_{5_2} = 2 - r_2$. Let us compute the CRRV. By (2.6) it must be $r_2 = \min\{1, a_2(r)/l_2(r)\} = \min\{1, (1-c)/(2-r_5)\}$ and $r_5 = \min\{1, a_5(r)/l_5(r)\} = \min\{1, (1-c)/(2-r_2)\}$. After solving this system we get that $r_2 = r_5 = 1-\sqrt{c}$ and since we assumed $c = 1/4$ we finally get that $r_2 = r_5 = 1/2$. For the payments of Bank 2, we know that $a_2 = 3/4$ and it first prioritises Bank 3 for which it has a liability of 1, thus it cannot fully pay off that liability and submits all of its assets to Bank 3, namely $p_{2_1} = p_{2,3} = 3/4$ and $p_{2_2} = p_{2,1} = 0$. The payments of Bank 5 are symmetrical.*

2.2 The Clearing Problem

2.2.1 CDS-CLEARING & CDS-PRIORITY-CLEARING

The clearing condition of Definition 2, forces an interdependence between the assets, liabilities, and clearing recovery rates of the banks in a financial system. This complex interrelationship among recovery rates makes computation of a clearing vector a non-trivial computational problem. At first instance, it is not even clear whether a clearing vector always exists, or whether there can be multiple clearing vectors.

The computational task of finding a *clearing vector* in a given financial system is generally referred to as the *clearing problem.* It is known that under the exclusive presence of debt contracts, the clearing problem can be solved in polynomial time [EN01]. We are interested in studying the complexity of the clearing problem under the addition of credit default swap contracts in the financial systems.

We define and study two variations of the clearing problem phrased as **CDS-CLEARING** and **CDS-PRIORITY-CLEARING**, where the term "CDS" signifies that the financial system under consideration contain **Credit Default Swaps**.

Problem 1 (**CDS-CLEARING**). ***Input:*** *A financial system* (N, e, c)*, where* $N = \{1, \cdots, n\}$ *is a set of* n *banks,* $e = (e_i)_{i \in N} \in \mathbb{Q}^n$ *is an* n*-dimensional vector of external assets, and* $c = n \times n \times n$ *is a 3-dimensional matrix containing all contract notionals.* ***Task:*** *Compute a clearing recovery rate vector* $r = (r_1, \cdots, r_n)$*, when the banks pay proportionally.*

Problem 2 (**CDS-PRIORITY-CLEARING**). ***Input:*** *A pair* $(\mathcal{F}, \mathcal{P})$*, where* $\mathcal{P}$ *is a profile of Singleton Liability Priority lists and* $\mathcal{F} = (N, e, c)$ *is a financial system where* $N = \{1, \cdots, n\}$ *is a set of* n *banks,* $e = (e_i)_{i \in N} \in \mathbb{Q}^n$ *is an* n*-dimensional vector of external assets, and* $c = n \times n \times n$ *is a 3-dimensional matrix that contains all contract notionals.* ***Task:*** *Compute a clearing recovery rate vector* r *for* $(\mathcal{F}, \mathcal{P})$*.*

Any clearing recovery rate vector $r = (r_i)_{i \in N}$ of an instance $\mathcal{F} \in$ **CDS-CLEARING** ($(\mathcal{F}, \mathcal{P}) \in$ **CDS-PRIORITY-CLEARING** respectively) constitutes a *solution*. We denote by $\text{Sol}(\mathcal{F})$ ($\text{Sol}((\mathcal{F}, \mathcal{P}))$ respectively) the set containing all solutions of $\mathcal{F}$ ($(\mathcal{F}, \mathcal{P})$ respectively).

It is important to realise that the solutions to instances of both problems, are essentially the fixed points of the function expressed at the right hand side of (2.6). Let $\mathcal{F} \in$ **CDS-CLEARING** and consider the function $f_{\mathcal{F}} : [0, 1]^n \mapsto [0, 1]^n$ defined at each coordinate $i \in [n]$ by

$$f_{\mathcal{F}}(r)_i = \frac{a_i(r)}{\max\{a_i(r), l_i(r)\}}\,^{2}. \tag{2.7}$$

Function $f = f_{\mathcal{F}}$ takes as input a recovery rate vector r and outputs a potentially different recovery rate vector $r' = f(r)$. It holds that $r' = r$ if and only if r is a *clearing vector* and hence a solution of $\mathcal{F}$. **CDS-CLEARING** asks, in essence, for computing a fixed point of a specific function. The case for **CDS-PRIORITY-CLEARING** is analogous.

In order to avoid situations of $0/0$ division in (2.7) and for the sake of compatibility with existing literature, we narrow the study of the presented problems to *non-degenerate* systems.

[2] Strictly speaking, $f_I(r)_i$ is well-defined only for nodes i that are not sinks (i.e. nodes with no outgoing arcs) with 0 external assets in the contract graph. Sink nodes have recovery rate 1, cf. (2.6). Hence, we implicitly exclude them from the definition of f_I. Their exclusion simply allows to bypass potential divisions by 0 in f_I while preserving its continuity.

Definition 3 (Non-degeneracy). *A financial system is* non-degenerate *if and only if*

1. *Every debtor in a credit default swap either has positive external assets or is the debtor in at least one debt contract with a positive notional. (Definition 4.2 [SSB17b])*

2. *Every bank that acts as a reference bank in some credit default swap is the debtor of at least one debt contract with a positive notional.*

The second condition was excluded from the initial non-degeneracy definition that is provided in [SSB17b, SSB17a], but is explicitly mentioned to always hold for every system in the presentation of the financial network model[3]*. Overall the non-degeneracy condition does not hold in a system only if there exist a credit default swap debtor that breaks the first condition.* ***In this thesis, we consistently assume that all financial systems are non-degenerate.***

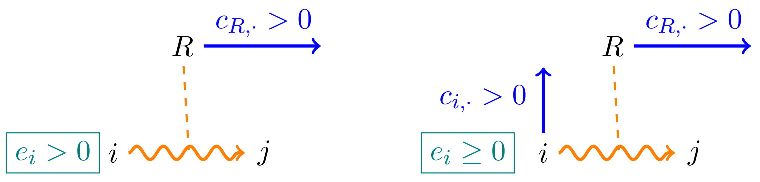


Figure 2.3: Topology of non-degenerate networks.

The fact that there exists at least one fixed point for every non-degenerate financial network of **CDS-CLEARING** (**CDS-PRIORITY-CLEARING** respectively) is proved in [SSB17b] ([PW20] respectively).

Theorem 1 (Theorem 3.2 [SSB17b], Theorem 1 [PW20])**.** *Every instance $\mathcal{F} = (N, e, c)$ of* **CDS-CLEARING**/ $(\mathcal{F}, \mathcal{P})$ *of* **CDS-PRIORITY-CLEARING***, admits a clearing recovery rate vector.*

Theorem 1 classifies both problems of interest as total search problems-(TFNP [MP91]), i.e, problems for which for every instance a solution always exist. Moreover the solutions correspond to the fixed points of function (2.7). Therefore, the initial task of calculating a clearing vector is synonymous with the computational challenge of finding a fixed point of function (2.7), the existence of which is guaranteed.

[3](see page 9 of [SSB17a])

2.2.2 Irrational Solutions

An obstacle is immediately encountered if one intends to solve both problems under the Turing machine model of computation, since there exist instances where all *clearing vectors* contain *irrational components*, and thus cannot be encoded in binary.

Observation 1. *There exist instances of* **CDS-CLEARING** *and* **CDS-PRIORITY-CLEARING** *that admit irrational clearing vectors.*

Proof. The reader can observe that the *clearing vector* computation of the system in Example 2 is independent from the payment scheme. Moreover in Example 2 we showed that $r_2 = r_5 = 1 - \sqrt{c}$. Thus, it is clear that for many choices of $c \in (0, 1)$ (e.g., $c = 1/2$) the CRRV is irrational. $\square$

Observation 2. *There exists a pair $(\mathcal{F}, \mathcal{P})$ with an irrational CRRV and irrational payments.*

Proof. Take again Example 2 and fix $c = 1/3$. We have $e_2 = e_5 = 2/3$, $l_{2,3} = l_{5,6} = 2/3$ and $r_2 = r_5 = 1 - \sqrt{1/3}$. Now consider the singleton liability priority lists $P_2^2 = ((2, 1, 5) \mid (2, 3))$ and $P_5^2 = ((5, 6, 2) \mid (5, 4))$. Since node 2 prioritises the $(2, 1, 5)$ contract, it has to pay an amount of $1 - \sqrt{1/3}$ to node 1. Given that its total assets are $2/3$, it can fully pay this liability and so $p_{2,1} = 1 - \sqrt{1/3}$ and what is left is being paid to node 3. Symmetrically, one can compute that $p_{5,6} = 1 - \sqrt{1/3}$. $\square$

Observation 3. *There exists a pair $(\mathcal{F}, \mathcal{P})$ with an irrational CRRV and rational payments.*

Proof. Consider Example 2 once more and fix $c = 1/2$. This yields $e_2 = e_5 = 1/2$, $l_{2,3} = l_{5,6} = 1/2$ and $r_2 = r_5 = 1 - \sqrt{1/2}$. Consider the profile $P_2^2 = ((2, 1, 5) \mid (2, 3))$ and $P_5^2 = ((5, 6, 2) \mid (5, 4))$. Since bank 2 prioritises the $(2, 1, 5)$ contract, it has to pay an amount of $1 - \sqrt{1/2}$ to node 1 but only possesses total assets of $1/2$. Thus it cannot fully pay this liability, meaning that $p_{2,1} = 1/2$. Symmetrically, we can compute that $p_{5,6} = 1/2$. $\square$

A thorough exploration of irrational solutions and an examination of the reasons behind their emergence are explored in Chapter 5.

2.2.3 Approximation concepts

Systems with irrational solutions are common and not difficult to construct. Hence, while the problems' instances always have solutions, they are not rational in general, consequently computing *exact* solutions is inconvenient for study under the standard Turing Machine model. One could however, quite naturally study this problem under a real valued computation model like [BCSS98], under which any real number can be stored in a single unit of memory, and the basic arithmetic operation take a single time unit to execute.

The preceding discussion prompts us to explore approximate solution concepts. Let F be a continuous function that maps a compact convex set to itself ($f_{\mathcal{F}}$ in (2.7) is such a function), and let $\epsilon > 0$ be a small constant.

Definition. *A* **weak ϵ-approximate fixed point** *of F is a point x such that its image is within a distance ϵ of x, i.e., $\|x - F(x)\|_\infty < \epsilon$.*

Definition. *A* **strong ϵ-approximate fixed point** *of F is a point x that is within a distance ϵ near a fixed point of F, i.e., $\exists x' : F(x') = x' \wedge \|x' - x\|_\infty < \epsilon$.*

Previous work on the *clearing problem* with credit default swaps has motivated the study of weak ϵ-approximate fixed points of function (2.7), termed as *ϵ-approximately clearing vectors* or simply *ϵ-solutions* (Definition 4.1 [SSB17b]). For motivation and thorough exploration and of the *ϵ-solution* concept, we directly address the reader to Appendix B of [SSB17b]. If we choose to clear a financial system that is assumed to have a unique clearing vector by means of a weak approximate clearing vector of function (2.7) or simply an ϵ-solution, we then may severely misrepresent the actual financial state of a bank. Strong approximations do not suffer from this problem, as the equity of all banks after clearing is indeed accurate (up to ϵ accuracy). While we legitimise the significance of strong approximations through an observation that weakly approximate solutions may "severely" misrepresent the actual financial state of an institution, it is also the case that a weak ϵ-approximate fixed point is generally not located close to an actual fixed point [4]. Conversely, despite the fact that the strong approximate solution is near the exact solution, it is impossible to verify whether a given point satisfies this approximation notion.

We demonstrate these claims with an example that supports and motivates the research on strong approximations.

[4]In the footnote of page 14 of [SSB17a], the authors mention that ϵ-solutions essentially may be far from an actual fixed point. Moreover they suggest that constructing a financial network that satisfies this property is not hard to construct but do not present any such example. In Example 3 we provide such an instance.

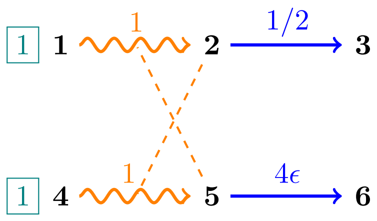


Figure 2.4: Financial system of Example 3

Example 3. *Consider the financial system depicted in Figure 2.4. Assume that* $e_1 = 1$, $e_2 = 0$, $e_4 = 1$, $e_5 = 0$. *Also* $c^5_{1,2} = c^2_{4,5} = 1$, $c_{2,3} = 1/2$, $c_{5,6} = 4\epsilon$, *for a parameter* $\epsilon > 0$.

Observe that $r = (1, 1, 1, 1, 0, 1)$ *is an exact clearing vector for the system in Figure 2.4. This can be verified since,* $f_1(r) = f_4(r) = 1$, *and* $f_2(r) = \min\{1, \frac{(1-r_5)}{1/2}\} = 1$, $f_5(r) = \min\{1, \frac{(1-r_2)}{4\epsilon}\} = 0$. *Since* $r = f(r)$, *the point* $r = (1, 1, 1, 1, 0, 1)$ *is an exact fixed point independently of the choice for the parameter* ϵ.

Assume that $\epsilon \geq \frac{1}{4}$, *then for the clearing vector holds that* $r_5 = \min(1, \frac{(1-r_2)}{4\epsilon})$ *and since* $\frac{(1-r_2)}{4\epsilon} \leq 1 - r_2 \leq 1$ *the clearing vector must satisfy that* $r_5 = \frac{1-r_2}{4\epsilon}$. *For Bank 2 under the clearing condition it must hold that* $r_2 = \min(1, 2 \cdot (1 - r_5))$. *If* $r_2 = 1$ *then* $r_5 = 0$ *which is the above exact clearing recovery rate vector. If* $r_2 = 2 \cdot (1 - r_5)$ *then* $r_5 = \frac{2r_5-1}{4\epsilon} \to 4\epsilon \cdot r_5 = 2 \cdot r_5 - 1$, *which means that* $r_5 = \frac{1}{2-4\epsilon}$ *but for* $\epsilon \in (\frac{1}{4}, \frac{1}{2})$ *it holds that* $r_5 > 1$ *which is a contradiction to the assumption that* $r_5 \leq 1$. *Thus for those values for the parameter* ϵ *the clearing vector is unique and equal to* $(1, 1, 1, 1, 0, 1)$.

Assume $r' = (1, 1 - 2\epsilon, 1, 1, 1/2 + \epsilon, 1)$. *It holds that* $f_2(r') = \min\{1, (1 - r'_5)/(1/2)\} = \min\{1, (1/2-\epsilon)/(1/2)\} = 1-2\epsilon$. *Moreover,* $f_5(r') = \min\{1, (1-r'_2)/4\epsilon\} = \min\{1, (1-1+2\epsilon)/4\epsilon\} = 1/2$. *Thus,* $\|r' - f(r')\|_\infty \leq \epsilon$ *which constitutes* $r' = (1, 1-2\epsilon, 1, 1, 1/2+\epsilon, 1)$ *a weakly* ϵ*-approximate fixed point.*

When comparing r' *to* r *though, we observe that the distance among the weakly* ϵ*-approximate clearing vector and the exact clearing vector is* $\|(1, 1, 1, 1, 0, 1) - (1, 1 - 2\epsilon, 1, 1, 1/2 + \epsilon, 1)\|_\infty > 1/2$. *This proves the claim that weakly* ϵ*-approximate clearing vectors may be very far from exact clearing vectors. On top of that a strong approximate clearing vector within distance 1/2 of the exact, is trivial to obtain by just by setting all components to 1/2.*

More importantly, notice that if $\epsilon \in (\frac{1}{4}, \frac{1}{2})$, *then as we computed earlier the clearing vector* $r = (1, 1, 1, 1, 0, 1)$ *is unique and the vector* r' *is a* $\frac{1}{4}$*-weakly approximate clearing vector. In that* *case though* $\|r - r'\|_\infty > 3/4$. *On top of that, the information we get on node 2 in* r' *is that* $r'_2 = 1 - 2\epsilon < 1$, *meaning that 2 is in default whereas actually 2 can fully pay its liabilities since* $r_2 = 1$. *This indicates that a weakly* ϵ*-approximate fixed point may contain misleading information about whether a bank is in default or not. Similar situations hold even for exact clearing vectors due to default ambiguity [SSB20] that arises when the clearing vector is not unique. The fact that for these specific values of the parameter* ϵ *the clearing vector is unique suggests that weak approximations are unreliable while motivating the study of strong approximations.*

Chapter 3

Computational Challenges

Overview

This chapter studies the complexity of **CDS-CLEARING**. First we provide a short introduction on *fixed point search problems*. In Section 3.1 we establish that computing exact solutions to instances of **CDS-CLEARING** is FIXP-complete and consequently computing *strong approximations* is FIXP_a- complete. A formal introduction of the complexity class FIXP is given in Section 3.1.1. Upon this introduction a full proof of the result is presented in Section 3.1.2. The main part of the proof is centered on the construction of systems that simulate the algebraic operations $\{+, -, *, /, \max, \min, \surd\}$. An analytic presentation of these financial gadgets is presented in Section 3.1.3 followed by a more technical analysis for the gadgets simulating $\{*, /\}$. An adaptation of these gadgets to certain priority profiles establishes FIXP-completeness of **CDS-PRIORITY-CLEARING**.

In Section 3.2 we define the *weak approximation version* of the problem as ϵ-**CDS-CLEARING** and prove that finding such approximate solutions is PPAD-hard for an explicit improved bound. The proof leverages the **PURE-CIRCUIT** problem, which is presented in Section 3.2.2. The chapter concludes in Section 3.2.4, where we identify two meaningful restrictions of the class of financial networks motivated by regulations: (i) the presence of central clearing authorities termed as central CDS debtors ($\mathcal{CCD}$); (ii) CDS debtors dedicated to fixed reference banks termed as dedicated CDS debtors. Both categories are subject to the same intractability result.

3.1 Computing Strong Approximations

A *fixed point problem* Π is defined as a search problem where every instance $I \in \Pi$ is associated with a continuous function $F_I : D_I \to D_I$ where $D_I \subseteq \mathbb{R}^n$ (for some $n \in \mathbb{N}$) is compact and convex, and the solutions of I are the fixed points of F_I. The class of functions $F = \{F_I \mid I \in \Pi\}$ (and the problem Π) is classified as

- ***polynomially computable*** if there is a polynomial q such that (i.) D_I is a convex polytope described by a set of at most $q(|I|)$ linear inequalities, each with coefficients of a size at most $q(|I|)$, and (ii.) For each vector x in $D_I \cap \mathbb{Q}^n$, the value $F_I(x)$ can be computed in time $q(|I| + \text{size}(x))$. By "size" of a rational number we mean the number of bits needed to represent the numerator and the denominator in binary.

- ***polynomially continuous*** if there is a polynomial q such that for each $I \in \Pi$, and rational $\epsilon > 0$, there is a rational δ of size $q(|I| + \text{size}(\epsilon))$ satisfying the following: for all $x, y \in D_I$ with $\|x - y\|_\infty < \delta$ it holds that $\|F_I(x) - F_I(y)\|_\infty < \epsilon$. Note that this condition defining polynomial continuity is the formally correct way of stating that the distance between any two points in the domain does not increase a lot (at most by a "polynomial amount") when taking their images under F_I.[1]

As previously stated, every fixed point function F associated with a search problem Π has a *weak* and a *strong* approximation version: In the weak approximation version we are given an instance I of Π and a rational $\epsilon > 0$, and we want to compute a weak ϵ-approximate fixed point for F_I. The strong approximation version is defined analogously.

Proposition 1 ([EY10], Proposition 2.2)**.** *Let Π be a fixed point problem and F_Π be an associated fixed point class of functions for Π.*

1. *If F_Π is polynomially continuous, then the weak approximation version of Π w.r.t F_Π polynomial-time reduces to the strong approximation version of Π.*

2. *If F_Π is polynomially continuous and polynomially computable, then the weak approximation version of Π w.r.t F_Π is in PPAD.* [2]

[1]The above collection of definitions is stated in a more extensive and refined way in [EY10].

[2]The reader might be familiar with PPAD, which is a complexity class introduced in [Pap94b]. Further below, we provide an indirect definition of PPAD in Proposition 2, through defining the complexity class Linear-FIXP.

It is established that function (2.7) associated with **CDS-CLEARING** is polynomially continuous under the *non-degeneracy* assumption (see Lemma C.1 [SSB17b]) and that the weak approximation version of **CDS-CLEARING** is PPAD-complete. Consequently due to Point 1 of Proposition 1, the strong approximation version of **CDS-CLEARING** is at least as hard as its weak approximation version, namely computing strong approximate clearing vectors is at least as hard as computing weak approximate clearing vectors.

Unfortunately as we will demonstrate later, we can construct financial systems capable of emulating multiplication operations on banks' recovery rates (see Figure 3.5). Consequently calculating a clearing vector may involve successive squaring operations, resulting in numbers with exponentially large bit representations relative to the input size. This is an indication that **CDS-CLEARING** cannot be polynomially computable. Hence, while it is true that the weak approximation version of **CDS-CLEARING** is in PPAD, Point 2 of Proposition 1 cannot be used to establish this fact and a separate proof, as provided in [SSB17b], is needed for PPAD-membership.

3.1.1 The Complexity Class FIXP

Etessami and Yannakakis in [EY10], define a framework for studying the complexity of fixed point computation problems, which considers both exact computation and approximate computation of the solutions to such problems. The authors introduce the complexity class FIXP, show that many fixed point problems can be shown to belong to this class, and establish that there exist FIXP-complete problems under a suitable notion of reducibility. **CDS-CLEARING** is, as we have seen, a fixed point problem, hence we consider the problem within the framework of [EY10].

We introduce in this section some fundamental notions related to exact and approximate computation of fixed points, and define the complexity class FIXP and some related classes.

Definition 4 (FIXP)**.** *The class FIXP consists of all fixed point problems Π for which for all $I \in \Pi$ the function $F_I : D_I \to D_I$ can be represented by an algebraic circuit C_I over the basis $A = \{+, -, *, \max, \min\}$, using rational constants, such that C_I computes F_I, and C_I can be constructed from I in time polynomial in $|I|$. In more detail, the circuit C_I is an acyclic directed graph over a set of gates, say $g_1, \ldots, g_m$ (in topological order), with the following properties. There are n gates in the first (bottom) layer of the circuit, called the* input *gates, where n is the number of arguments of F_I. Similarly there are n output gates in the last (top) layer of the circuit, called the* output gates. *The remaining nodes of the circuit each represent rational constants and arithmetic operations from the set A. Rational constant nodes have no incoming arcs, and each arithmetic operation node has two incoming arcs. The number of outgoing arcs from each node is not bounded. We may now label C_I's input nodes with an input $x \in D_I$ (coordinate-wise), and define the output of C_I by propagating the outputs of each of the nodes through the network via the arcs outgoing from each of the nodes. Naturally, an input node propagates its labeled value, a rational constant node propagates the associated rational number, and an arithmetic operation node takes the two values propagated through its incoming arcs as operands, and applies the corresponding operation on its operands, which it then propagates through its outgoing arcs. The circuit C_I is then said to compute F_I iff the output of C_I on input x equals $F_I(x)$ for all $x \in D_I$.*

Definition 5 (FIXP_a)**.** *The class FIXP_a is defined as the class of search problems that are the strong approximation version of some fixed point problem that belongs to FIXP.*

Definition 6 (Linear-FIXP)**.** *The class Linear-FIXP is a discrete total search problem class defined analogously to FIXP, but under the smaller arithmetic basis where only the gates $\{+, -\max, \min\}$ and multiplication by rational constants are used.*

The classes FIXP, Linear-FIXP, and FIXP_a admit complete problems. Hardness of a search problem Π for FIXP (resp. Linear-FIXP and FIXP_a) is defined through the existence of a polynomial time computable function $\rho : \Pi' \to \Pi$, for all $\Pi' \in$ FIXP (resp. FIXP_a), such that the solutions of I can be obtained from the solutions of $\rho(I)$ by applying a (polynomial-time computable) linear transformation on a subset of $\rho(I)$'s coordinates. This type of reduction is known as a *polynomial time SL-reduction.*

We mention a few important facts about these complexity classes that are relevant for understanding the results in this thesis. The following proposition states that strong approximations to problems in FIXP can be found in polynomial space.

Proposition 2 ([EY10], Proposition 4.2)**.** *$\mathsf{FIXP}_a \subseteq$ PSPACE.*

The following result shows that finding exact fixed points of fixed point functions expressible through only the operations $\{+, -, \max, \min\}$ and multiplication by rational constants, is equivalent to solving problems in PPAD.

Theorem 2 ([EY10], Theorem 5.4)**.** *Linear-FIXP = PPAD.*

Consequently, the solutions of instances in Linear-FIXP are always rationals of polynomial size.

Remark 1. *One detail that we have omitted (for simplicity) is that formally FIXP and Linear-FIXP are defined to also include their closures under polynomial time separable linear reductions-(SL-reductions) and polynomial time reductions respectively.*[3] *Thus, the traditional notion of polynomial-time reducibility is appropriate here. We remark furthermore that in the initial definition of FIXP, which was given in [EY10], the division operator and the operators $\sqrt[k]{\cdot}$ for $k \in \mathbb{N}$ were included in the arithmetic base. However, in the same paper the authors show that these can be dropped without altering the class. Similarly, the original definition of FIXP includes problems in which D_I is an arbitrary polytope with a polynomially computable set of constraints, but restricting this to $[0,1]^n$ turns out to not change the class.*

An informal understanding of how the hardness of FIXP compares to PPAD (or Linear-FIXP) is as follows: PPAD captures a type of computational hardness stemming from an essentially combinatorial source, as PPAD was originally defined as a class that captures the hardness of computing a solution to a graph-theoretical problem, where the solution is guaranteed to exist via a non-constructive parity argument. The class FIXP introduces on top of that a type of numerical hardness that emerges from the introduction of multiplication and division operations: These operations give rise to irrational exact solutions to these problems, and may independently also require the computation of rational numbers of very high precision or very high magnitude (which is enabled by the presence of the multiplication operation, by which one can perform successive squaring inside the associated algebraic circuit).

3.1.2 FIXP-Completeness of CDS-CLEARING

Our first main result shows that **CDS-CLEARING** and its strong approximation variant are FIXP_a-complete. The proof is based on reducing from the canonical FIXP-hard problem, defined in [EY10], of computing a fixed point of an algebraically specified Brouwer function given by an

[3]The term "*separable linear*" refers to the property that solutions of the reduced instance correspond to solutions of the original instance through a polynomial-time computable *linear transformation* of a subset of the coordinates of the reduced instance's solution. The authors of [EY10] require separable linearity for FIXP-reductions, but not for Linear-FIXP reductions. This is due to the fact that the fixed points of functions belonging to FIXP may be irrational, and linearly separable reductions preserve the approximation properties of approximate fixed points. For Linear-FIXP, there always exist at least one rational fixed point, and one is generally interested in exact computation for this class.

algebraic circuit. We show that we can take such an algebraic circuit (as described in [EY10]) and encode it in a direct way in the form of a financial system. Hence, our polynomial time hardness reduction is implicitly defined to work from any arbitrary fixed point problem in FIXP. The reduction is constructed by devising various financial network gadgets which enforce that certain banks in the system have recovery rates that are the result of applying one of the operators in FIXP's arithmetic base to the recovery rates of two other banks in the system. In other words, we can design our financial systems such that the interrelation between the recovery rates mimics a computation through an arbitrary algebraic circuit.

Theorem 3. **CDS-CLEARING** *is FIXP-complete, and its strong approximation version is FIXP$_a$-complete.*

Proof. Containment of **CDS-CLEARING** in FIXP is immediate: The clearing vectors for an instance $\mathcal{F} = (N, e, c) \in$ **CDS-CLEARING** are the fixed points of the function $f_{\mathcal{F}}$ defined coordinate-wise by

$$f_{\mathcal{F}}(r)_i = \frac{a_i(r)}{\max\{a_i(r), l_i(r)\}}$$

as in (2.7). The functions $l_i(r)$ and $a_i(r)$ are defined in (2.1) and (2.5), from which it is clear that $f_{\mathcal{F}}$ can be computed using a polynomial size algebraic circuit consisting only of $\{\max, +, *, -, /\}$ and rational constants. The non-degeneracy condition that holds for every financial system, prevents division by 0 for every instance $\mathcal{F}$. That holds because from statement (i) in Definition 3 if a bank i is a CDS debtor then both $a_i(r)$ and $l_i(r)$ cannot be equal to zero. Although not explicitly stated in our definitions, we can always assume that in the networks there are no isolated banks with zero assets and liabilities. Finally in the case where i is a sink bank, we point the reader to footnote 1 in page 21.

It is clear that the output of the algebraic circuit that computes $f_{\mathcal{F}}$ is well-defined for every $x \in [0,1]^n$. This shows that **CDS-CLEARING** is contained in FIXP and that its strong approximation version is contained in FIXP$_a$.

For the FIXP-hardness of the problem, let Π be an arbitrary problem in FIXP. We describe a polynomial-time reduction from Π to **CDS-CLEARING**. Let $I \in \Pi$ be an instance, let $F_I : [0,1]^n \to [0,1]^n$ be I's associated fixed point function, and let C_I be the algebraic circuit corresponding to F_I. As a pre-processing step, we convert C_I to an equivalent alternative circuit C'_I that satisfies that all the signals propagated by all gates in C'_I and all the used rational constants in C'_I are contained in the interval $[0, 1]$. The transformed circuit C'_I may contain two additional type of gates: Division gates and gates that compute the absolute value of the difference of two operands. We will refer to the latter type of gate as an *absolute difference gate*. The circuit C'_I will not contain any subtraction gates, and will not contain max and min gates either.

The transformation procedure for C_I follows the same approach of the transformation given in Theorem 4.3 of [EY10] where the 3-Player Nash equilibrium problem is proved FIXP-complete, and borrows some important ideas from there. Nonetheless, there are important differences in our transformation, starting with the fact that we use a different set of types of gates in our circuit.

First, transforming C_I into a circuit with only non-negative rational constants is trivial, since we can introduce subtraction gates combined with the rational constant 0 (which we will get rid of later on in the transformation process).

As a next step, we remove all min and max gates and replace them with addition, subtraction, multiplication, and absolute difference gates through the identities $\max\{a, b\} = (1/2) \cdot ((a + b) + |a - b|)$ and $\min\{a, b\} = (1/2) \cdot ((a + b) - |a - b|)$. Let the resulting circuit be C''_I.

The third step in the transformation is to move all the subtraction gates to the top of the circuit, right before the output gates, which results in exactly one subtraction gate being present for each output variable. This step is realised by representing each (potentially negative) signal s in C''_I that is propagated between two arithmetic gates, by two separate signals $s^+, s^- \geq 0$, which represent the positive and negative parts of s respectively, and are guaranteed to have values such that $s^+ - s^- = s$: Each arithmetic gate g_i in the circuit is replaced (sequentially, from the bottom of the circuit upward) by gates that operate on the separate positive and negative parts of the original input signals as follows.

Suppose g_i is a node outputting a constant rational value $c > 0$, then we may replace g_i by two constant nodes, one which outputs c, the positive part of the resulting separated signal, and one which outputs 0, representing the negative part of the signal.

Suppose next that g_i is an addition gate operating originally on the outputs s and t of two other gates. Then, we use the observation that for the two signals s and t, separated into positive and negative parts s^+, s^-, t^+, t^-, it holds that $(s^+ - s^-) + (t^+ - t^-) = (s^+ + t^+) - (s^- + t^-)$. Therefore, g_i can be replaced by two addition gates g_i^+ and g_i^-, where g_i^+ operates on signals s^+ and t^+, and g_i^- operates on gates s^- and t^-.

If g_i is a multiplication gate, note that for two inputs s and t, separated into positive and negative parts s^+, s^-, t^+, t^-, it holds that $(s^+ - s^-) \cdot (t^+ - t^-) = (s^+t^+ + s^-t^-) - (s^+t^- + s^-t^+)$, so we may similarly replace each multiplication gate with four multiplication gates and two addition gates accordingly.

When g_i is an absolute difference gate, we note that for two inputs s and t, separated into positive and negative parts s^+, s^-, t^+, t^-, it holds that $|(s^+ - s^-) - (t^+ - t^-)| = |(s^+ + t^-) - (s^- + t^+)| - 0$ so that we can replace g_i with two addition gates, one absolute difference gate, and one constant gate outputting 0 (representing the negative part of the resulting signal).

Lastly, when g_i is a subtraction gate in C''_I, we remove it and connect the gates that provide the inputs to g_i appropriately to the subsequent gates that g_i points to. The final subtraction gates introduced in the top layer of the circuit are directly connected to the output gates, and subtract the positive and negative parts of the final output signal, so as to generate the correct n outputs. Since the function F_I maps $[0, 1]^n$ to itself, and the resulting circuit is equivalent to the original circuit C_I, we now have a circuit where all signals in the circuit are non-negative, and where we have introduced the absolute difference operation as an additional type of gate into the circuit. Next, we perform a straightforward step that eliminates the remaining m subtraction gates altogether, by replacing them with absolute difference gates. This gives us an equivalent circuit, as we know that the result of the n subtractions at the top of the circuit are positive for all possible inputs. We call this resulting circuit without subtraction gates C'''_I.

The last step is to transform the circuit C'''_I into a circuit where all signals inside the circuit are in $[0, 1]$ for all possible inputs to the circuit. To do so, in case there are rational constants exceeding 1 present in the circuit, we first multiply all the rational constants by the inverse $c \in [0, 1]$ of the largest rational constant in the circuit. We add the appropriate gates at the start of the circuit that multiply all input gates by c, and we add division gates at the end of the circuit that divide the final signal by c, before being propagated to the output gates. Furthermore, for each multiplication gate, we add a division gate that divides its result by c. This results in a circuit where all rational constant nodes are in $[0, 1]$, and all internal signals of the original circuit are essentially scaled by a factor of c. The resulting circuit is thus equivalent to C'''_I.[4]

[4]Note that the division by c that we introduce right after every multiplication gate is needed because multiplying two scaled signals $c \cdot s$ and $c \cdot s'$ results in c^2ss', hence dividing this result by c results in the desired output signal css'.

Now that we have that all the rational constants are in $[0, 1]$, our final step is transforming the circuit further in such a way that all signals inside the circuit are contained in $[0, 1]$ as well. Observe that the magnitude of any signal in the circuit is at most doubly exponential in the size of C_I''' (this can be attained when a certain signal $s > 1$ goes through a chain of successive squaring operations, using multiplication gates). We thus perform the following final transformation to enforce that all signals stay in $[0, 1]$: Let d be a number, bounded by a polynomial in $|I|$, such that all signals in the circuit are strictly smaller than 2^{2^d} for every input in $[0, 1]^n$. We add to the start of the circuit an auxiliary circuit T that computes $t = 1/2^{2^d}$ by successively squaring the constant $1/2$ a number of d times, and we use T right after each input node and right after each rational constant node in order to multiply every input signal and every rational constant node by t. For each multiplication gate, we add a division gate such that its result is divided by t. Lastly, at the end of the circuit, we add a division gate just before each output gate, that divides the final signal by t. We now observe that each signal in the new circuit has effectively been multiplied by a factor of t, with exception of the signals directly propagated by the input gates, and directly entering the output gates, which still retain their original values.

This completes our pre-processing steps on the algebraic circuit, and we denote the resulting circuit by C_I'. The circuit C_I' has the desired properties that we are looking for: All signals in the circuit are in $[0, 1]$ regardless of the input. Furthermore, the applied transformation ensures that C_I' is equivalent to the original C_I and thus represents F_I. The circuit can be obtained in a polynomial number of computation steps from C_I and is thus polynomial-time computable from the instance I. The circuit C_I' consists of $\{+, *, /\}$-gates, as well as absolute difference gates that compute the absolute value of the difference of two inputs. Lastly, there are rational constant nodes in the circuit where all such constants are in $[0, 1]$. For notational convenience, in the remainder of the proof we may treat C_I' as the function F_I, hence we may write $C_I'(x) = y$ to denote $F_I(x) = y$.

In the remainder of the proof, we define our reduction ρ to **CDS-CLEARING**: We construct our instance $\rho(I)$ of **CDS-CLEARING** (i.e., a non-degenerate financial network) from the circuit C_I'. The instance $\rho(I)$ will have the property that its clearing vectors are in one-to-one correspondence with the fixed points of C_I', and that banks $1, \ldots, n$ in our construction correspond to the input gates of C_I'. More precisely, our construction is such that for each fixed point x of C_I', in the corresponding clearing vector r for $\rho(I)$ it holds that $(r_1, \ldots, r_n) = x$.

Our reduction works through a set of financial system *gadgets*, of which we prove that their recovery rates (under the clearing condition) must replicate the behaviour of each type of arithmetic operation that can occur in the circuit C'_I. This part of our hardness reduction shares similarities with the PPAD-hardness proof in [SSB17b] who also design a set of gadgets to replicate the behaviour a certain circuit, although this circuit is of a different type, and our gadgets account for an entirely different set of operations.

Each of our gadgets has one or two *input banks* that correspond to the input signals of one of the types of arithmetic gate, and there is an *output bank* that corresponds to the output signal of the gate. For each of the gadgets, it holds that the output bank must have a recovery rate that equals the result of applying the respective arithmetic operation on the recovery rates of the input banks.

In our financial system, these gadgets are then connected together according to the structure of the circuit C'_I: Output banks of (copies of) gadgets are connected to input banks of other gadgets through a single unit-cost debt contract, which mimics the propagation of a signal between two gates of the arithmetic circuit. This results in a financial system whose behaviour replicates the behaviour of the arithmetic circuit. The first layer of the financial system consists of n banks representing the n input nodes of the circuit, and the last layer of the financial system has n banks corresponding to the n output nodes of the circuit. As a final step in our reduction, the n output banks in the last layer are connected through a single unit-cost debt contract to the n input banks. This last step enforces the recovery rates of the input banks (i.e. banks $1, \dots, n$) are equal to the recovery rates of the last layer, under the clearing requirement. Consequently, any vector of clearing recovery rates for $\rho(I)$ must then correspond to a fixed point of C'_I, where the recovery rates of the first n banks in the system equal those of the final n banks, so that $C'_I(r_1, \dots, r_n) = (r_1, \dots, r_n)$, i.e., $(r_1, \dots, r_n)$ is a fixed point of C'_I.

The above paragraphs comprise a high-level description of our reduction. We will now proceed to discuss the details. We start with defining our gadgets, using our graphical notation. As stated, our gadgets each have one or two input banks, and one output bank. Each gadget represents certain arithmetic operations, and gadgets can be combined with each other to form the arithmetic basis $\{+, *, /\}$ and the absolute difference operation defined above.

We start with a straightforward addition gadget, named g_+, depicted in Figure 3.1. This gadget directly accounts for the addition operation in the arithmetic basis. In our figures, input banks are denoted by black arrows incoming from the left, and output banks correspond to black arrows pointing out of the bank. The black arrows represent connections to other gadgets, and these connections are always realised by a debt contract of unit cost, and are always from the output node of a gadget to an input node of another gadget. The assets that flow into an input gadget are thus always in $[0, 1]$ and all gadgets are designed such that the incoming flow of assets must equal the clearing recovery rate of a bank. For example: For both input banks of the addition gadget, their liabilities are equal to 1, given that their assets are equal to the incoming flow they receive, their recovery rate under any clearing vector (which must equal the ratio of assets and liabilities) is equal to the inflow along the black arc.

In the figures representing our gadgets, some of the nodes have been annotated with a red labelled formula in terms of the recovery rates of the input banks of the gadget, subject to the resulting values being in the interval $[0, 1]$. This can be seen for example in the output node of our addition gadget in Figure 3.1. Such a formula represents the value that a clearing recovery rate must satisfy for the respective node. It is straightforward to verify that the given formulas are correct for each of our gadgets.

Since all signals inside C'_I are guaranteed to be in $[0, 1]$ for all input vectors, our financial system gadgets can readily be used and connected to each other to construct a financial system that corresponds to C'_I, i.e., such that the clearing recovery rates of the input and output banks of each of the gadgets must correspond to each of the signals inside the circuit C'_I.

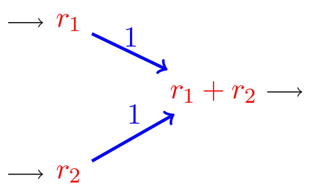


Figure 3.1: Addition gadget g_+.

The remaining gadgets are displayed in Figures 3.3 to 3.8 in the next section.

Besides gadgets for the necessary arithmetic operations, our reduction employs an additional *duplication gadget* g_{dup} that can be used to connect the output of a particular gadget to the input of more than one subsequent gadget. This gadget is displayed in Figure 3.3. Gadget g_{dup} furthermore has the convenient property that it can be used for multiplication by a rational $c \in [0,1]$: One of the output nodes has the recovery of the input node, while the other has this recovery rate multiplied by c. When choosing $c = 1$ the input recovery rate is effectively duplicated.

The gadgets for multiplication and division, g_* and $g_/$ are given in simplified form in Figures 3.5 and 3.4. These gadgets work as intended, but the problem with them is that they do not satisfy the non-degeneracy condition that we assume instances of **CDS-CLEARING** to have. It is possible to transform these into non-degenerate gadgets, but this process is technically detailed, in particular for the case of division: It requires replacing some of the divisions in the circuit C'_I by taking successive square-roots followed by successive squaring operations, where proper care has to be taken to ensure that the results of all these operations stay within the interval $[0,1]$. The non-degenerate versions of our gadgets (which interestingly includes a gadget that outputs square roots of input recovery rates), and the additional transformation of C'_I that is required for this, are described in the next section.

The absolute difference gadget g_{abs} is given in Figure 3.8. It uses g_{dup} and g_+, as sub-gadgets in its definition, as well as the sub-gadget $g_{\text{pos}-}$ (given in Figure 3.6) that takes two input recovery rates r_1 and r_2, and produces an output recovery rate of $\max\{r_1 - r_2, 0\}$.

Besides this set of gadgets that correspond to the arithmetic base of the circuit C'_I, in order to translate the circuit C'_I appropriately into a financial system, we also need the ability to generate rational constants as inputs to the gadgets (which correspond to the rational constant nodes of C'_I). This is straightforward: Any rational recovery rate $c \in [0,1]$ can be generated using a single node i with a single outgoing debt contract of unit cost, that can be pointed towards an input node of any gadget copy.

We provide some further financial system gadgets in Section 3.1.4. These gadgets demonstrate the ability of financial systems to simulate the max and min operations. We included these for the interested reader, but those gadgets are otherwise not used in our reduction, since C'_I does not have any min and max gates.

With the set of gadgets we have now defined, we can proceed to interconnect copies of gadgets to create financial systems of which the clearing recovery rates of the input and output gadgets respects an arbitrary algebraic computations over the arithmetic basis $\{+, *, /\}$, the absolute difference operator, and rational constants. In particular, we can connect n input nodes, say banks $\{1, \ldots, n\}$, to a network of gadgets that implements the algebraic circuit C'_I. We connect its n outputs, say banks $(m-n+1, \ldots, m)$ (where m is the number of banks in the resulting financial system) pairwise to the n input nodes, and we define the resulting financial system to be $\rho(I)$. It is clear that $\rho(I)$ can be constructed in polynomial time from C'_I, and since C'_I can be constructed in polynomial time from I, the financial system $\rho(I)$ takes polynomial time to compute.

Next, we show that clearing vectors of $\rho(I)$ correspond in a polynomial-time computable way (in this case: trivially) to fixed points of C'_I, thereby proving the correctness of the reduction and completing the proof. Let r be a clearing vector of $\rho(I)$. We show that $(r_1, \ldots, r_n)$ is a fixed point of I. Due to the fact that r is clearing, and by the definition of the gadgets, we know that the assets flowing into nodes $(1, \ldots, n)$ are respectively $(r_1, \ldots, r_n)$, and come from nodes $(m-n+1, \ldots, m)$. Hence, the recovery rates of nodes $(m-n+1, \ldots, m)$ are equal to $(r_1, \ldots, r_n)$. By construction of our network, (i.e., by correspondence of our gadgets and the way they are connected to each other, following the structure of the algebraic circuit C'_I), it must then hold that $C'_I(r_1, \ldots, r_n) = (r_1, \ldots, r_n)$. Hence, $(r_1, \ldots, r_n)$ is a fixed point of I.

This completes the proof, as the above is sufficient to establish FIXP-completeness. FIXP_a completeness of the strong approximation variant is immediate, since any strong ϵ-approximation of the recovery rate vector of $\rho(I)$ corresponds in the same manner to a strong ϵ-approximate fixed point of C'_I.

In addition, it is straightforward to see that fixed points of C'_I correspond to recovery rates of $\rho(I)$, so that indeed $\rho(I)$ captures the complete set of fixed points of C'_I: Let x be a fixed point of C'_I. Now construct a vector of recovery rates for $\rho(I)$ by setting $(r_1, \ldots, r_n) = x$ and compute the remaining recovery rates of the nodes inside the gadgets in the natural way, as described by the arithmetic operations that each of the gadgets represents. Since x is a fixed point of C'_I, the recovery rates of banks $(m-n+1, \ldots, m)$ will be set to x, which causes the recovery rates of banks $(1, \ldots, n)$ (and thereby also the entire financial system) to satisfy the clearing condition. □

In Section 3.1.4, we present a set of financial system gadgets that adapt to the framework of *Singleton Liability Priorities*. Substituting these gadgets with the ones used in the above reduction allows us to generalise the statement of Theorem 3 to **CDS-PRIORITY-CLEARING**.

Theorem 4. CDS-PRIORITY-CLEARING *is FIXP-complete, and its strong approximation version is* FIXP_a*-complete.*

3.1.3 Financial System Gadgets

In this section we present all contract graphs (Figures 3.2–3.15) and the detailed configuration for all the gadgets employed in the proof of Theorem 3.

$\boxed{\textbf{Inversion gadget } g_{\text{inv}}\textbf{:}}$ This is a basic gadget involved in the construction of more complicated systems. Bank 1 has zero external assets and receives payment r from the input bank. Since $l_{1,2}(r) = 1$ it holds that $r_1 = \min(1, r) = r$. For Bank 3 it holds that $a_3(r) = 1$ and $l_{3,4}(r) = 1 - r_1 = 1 - r$. Since $r_3 = \min(1, 1/(1-r)) = 1$, Bank 3 pays an amount of $1 - r$ to Bank 4 which in turn submits the same amount of money to the output bank.

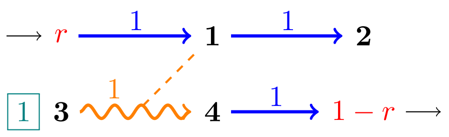


Figure 3.2: Inversion gadget g_{inv}.

$\boxed{\textbf{Duplication gadget } g_{dup} \textbf{ or } g_{c\cdot r}\textbf{:}}$ This gadget receives an input value r and outputs two values: r and $c \cdot r$, where c is a rational constant. Choosing $c = 1$ yields a duplication gadget that takes the input recovery rate r and outputs r as the two recovery rates of the output banks. We may also denote this gadget by $g_{c\cdot r}$, for a rational constant c this gadget is used to multiply the input by a constant c. The computation of the gadget up to Bank 4 is identical to the inversion gadget. For Bank 5 it holds that $a_5(r) = c$ and $l_{5,6}(r) = c \cdot (1 - r_3) = c \cdot (1 - r)$. Eventually since $r_5 = \min(1, 1/(1-r)) = 1$ Bank 5 pays Bank 6 an amount of $c \cdot (1 - r_3) = c \cdot r$, which in turn Bank 6 transfers to the output bank.

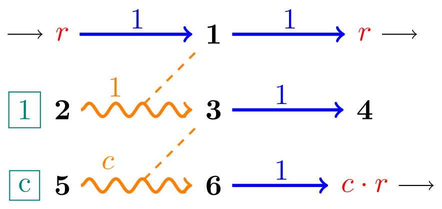


Figure 3.3: Gadget g_{dup}

$\boxed{\textbf{Degenerate Division gadget } g_{/}\textbf{:}}$ This gadget is degenerate because Bank 6 is a CDS debtor with zero external assets, thus it does not satisfy condition (i) of the non-degeneracy property. We present it for the sake of keeping the exposition simple, but it is not formally used in the reduction of Theorem 3, as this reduction should result in a non-degenerate financial system. This gadget has two input banks that receive a payment of r_2 and r_1 respectively. Notice that the sub-gadget that spans from the first input bank and banks $\{1, 2, 3, 4, 5\}$ is a g_{inv} gadget, thus $r_4 = 1 - r_2$. Bank 6 receives an incoming payment of r_1 and has liability $l_{6,7}(r) = r_2$, thus $r_6 = \min(1, r_1/r_2)$. The rest of the gadget constitutes a duplication sub-gadget where $r_{13} = \min(1, r_6) = \min(1, r_1/r_2)$. Notice that if $r_1 > r_2$ then the output bank has a recovery rate of 1 otherwise the output bank has a recovery rate of r_1/r_2.

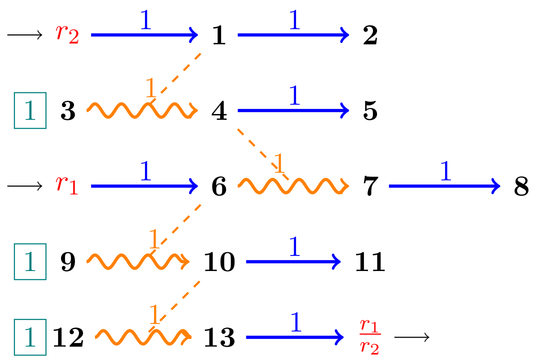


Figure 3.4: Division gadget $g_{/}$.

$\boxed{\textbf{Degenerate Multiplication gadget } g_*\text{:}}$ Similarly due to Bank 6, this gadget does not satisfy condition (i) of the non-degeneracy property. We present it for the sake of keeping the exposition simple, but it is not formally used in the reduction of Theorem 3, as this reduction should result in a non-degenerate financial system. Observe that $a_6(r) = r_2$ and $l_6(r) = l_{6,8}(r)+l_{6,7}(r) = 1-r_1+r_1 = 1$, thus $r_6 = \min(1, r_2) = r_2$. Finally $p_{6,7}(r) = r_6 \cdot (1 - r_4) \cdot 1 = r_2 \cdot r_1$, which is the amount of money transferred to the output bank.

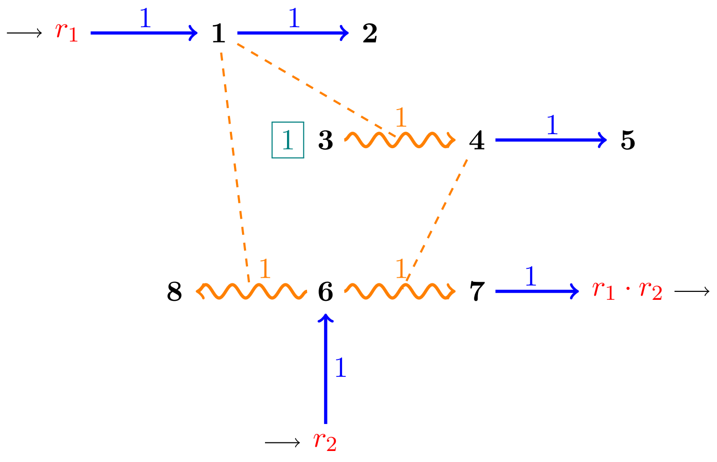


Figure 3.5: Multiplication gadget g_*..

$\boxed{\textbf{Positive subtraction gadget } g_{\text{pos}-}\text{:}}$ This gadget takes as input two recovery rate values r_1, r_2 and outputs the value $\max\{0, r_1 - r_2\}$. Bank 7 receives a payment of $1 - r_1$ from Bank 5 and r_2 from the second input bank, thus $r_7 = \min(1, 1 - r_1 + r_2)$. Bank 9 holds external assets of 1 it can always pay any generated liability. We proceed by a case analysis of r_7. If $r_7 < 1$ then $1 - r_1 + r_2 < 1$ which means that $r_1 > r_2$. Since $l_9(r) = 1 - r_7 = r_1 - r_2$ and $r_1 > r_2$, Bank 9 transfers an amount of $\max\{0, r_1 - r_2\}$ to the output bank. If $r_7 = 1$ then $1 - r_1 + r_2 \geq 1$ which means that $r_1 \leq r_2$ but also $l_9(r) = 0$. In that case Bank 9 pays 0 to the output bank and since $r_1 \leq r_2$ the expression $\max\{0, r_1 - r_2\}$ is again satisfied.

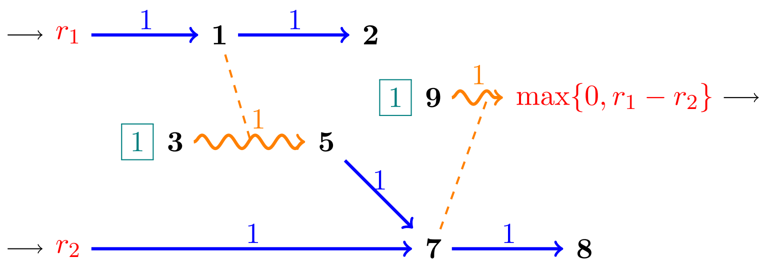


Figure 3.6: Positive subtraction gadget $g_{\text{pos}-}$.

$\boxed{\textbf{Absolute difference gadget } g_{\text{abs}}\text{:}}$ This gadget receives as input two payments of r_1 and r_2. First it uses two duplication gadgets on both r_1, r_2 and subsequently two positive subtraction gadgets where one receives an input of r_1, r_2 and the second receives an input of r_2, r_1. The positive subtraction gadgets compute $\max\{0, r_1 - r_2\}$ and $\max\{0, r_2 - r_2\}$ respectively. These two maxima are added together using an addition gadget, resulting in the desired output $|r_1 - r_2|$.

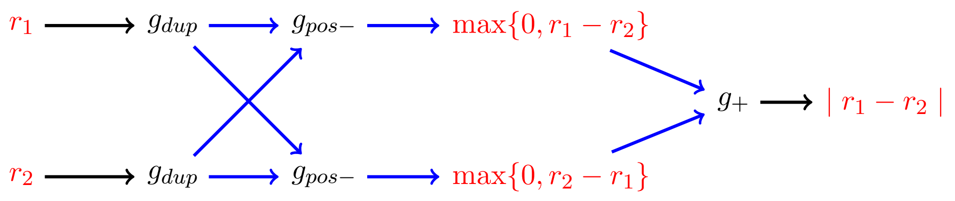


Figure 3.7: Absolute difference gadget g_{abs}: This is a compact representation of the gadget where the nodes labeled with a subscripted g have to be replaced by copies of the respective gadget in order to obtain the full financial network that defines the gadget.

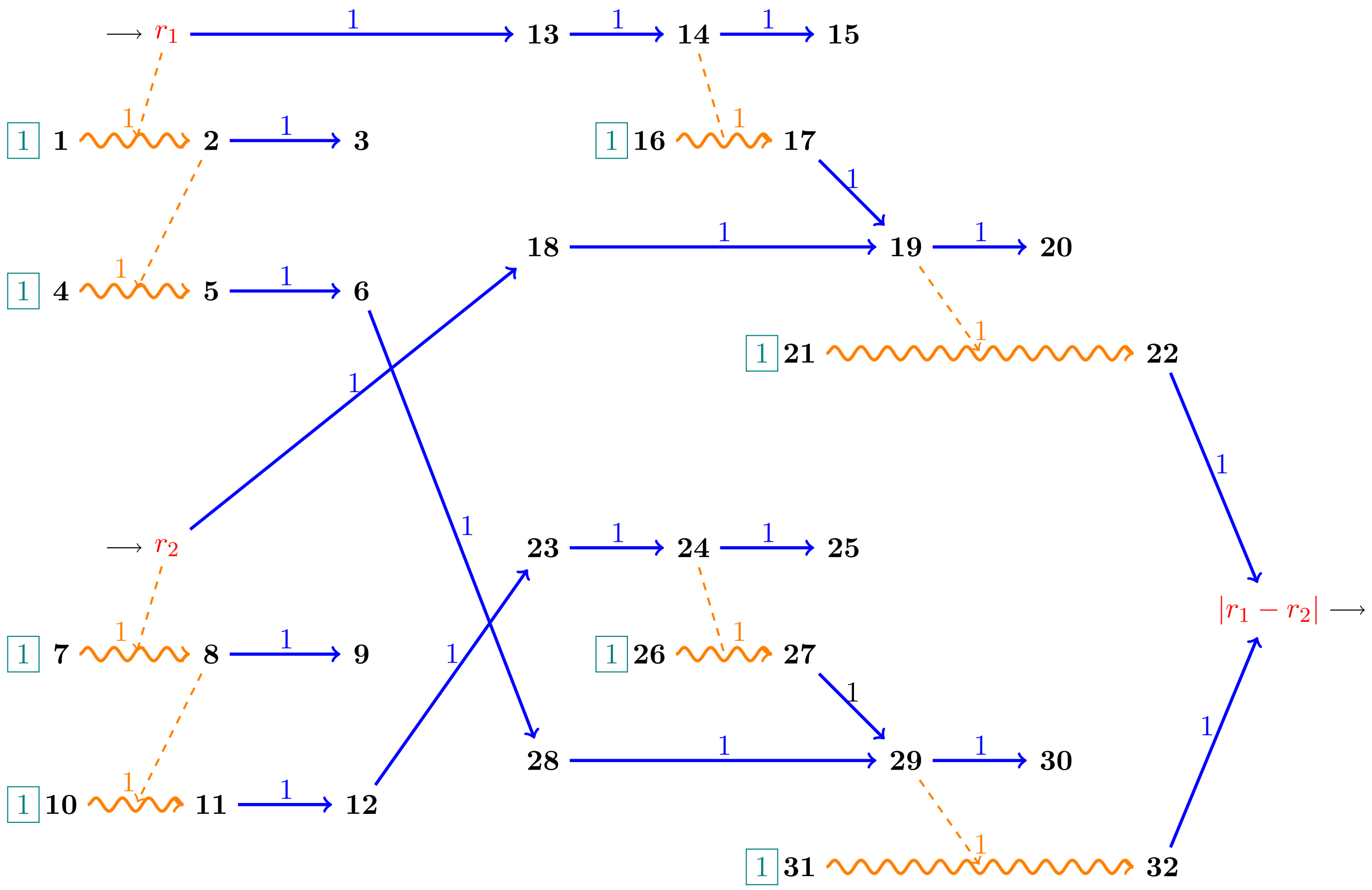


Figure 3.8: Absolute difference gadget g_{abs}: The full version of the gadget.

$\boxed{\textbf{Non-degenerate Multiplication and Division Gadgets:}}$ We describe how to adapt the hardness reduction of Theorem 3 in such a way that the financial system is non-degenerate. This is a rather technical detail and needs substantial work, yet it is needed for correctness of the reduction.

We present two non-degenerate versions of the multiplication gadget in Figures 3.9, 3.10. In Figure 3.9, observe that Bank 6 is a CDS debtor with positive external assets, thus it satisfies condition (i) of the non-degeneracy condition. In this gadget, the input recovery rates r_1 and r_2 are pre-processed by multiplying them both by 1/2, using the constant multiplication version of gadget g_{cr} in Figure 3.3 with $c = 1/2$. The gadget then generates the recovery rate $\frac{1}{4} \cdot r_1(1 + r_2) \in [0, 1]$, which can be post-processed using a positive subtraction gadget $g_{\text{pos}-}$ on input $\frac{1}{4} \cdot r_1(1 + r_2)$ and the value $\frac{1}{4} \cdot r_1$ (which can be generated by applying a $g_{\frac{1}{4} \cdot r}$ gadget to the input signal r_1) to output the value $\frac{1}{4} \cdot r_1 r_2$, which is then parsed as input to a $g_{4 \cdot r}$ gadget in order to obtain $r_1 \cdot r_2$.

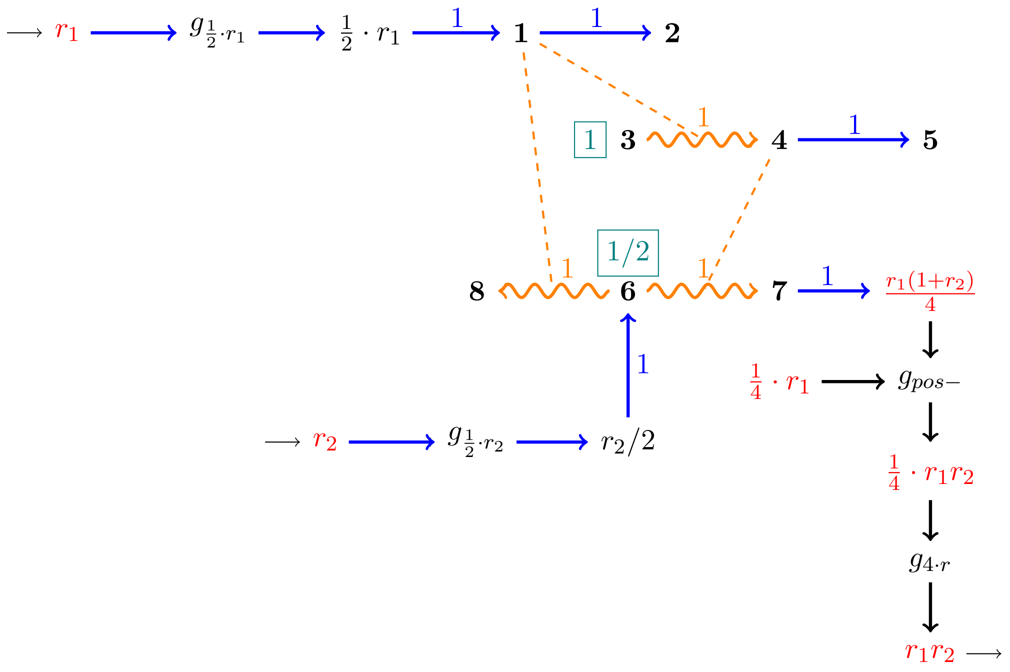


Figure 3.9: Compact representation of a non-degenerate version of the multiplication gadget g'_*.

In Figure 3.10 we present an alternative non-degenerate multiplication gadget where all contract notionals are assumed to be 1. In this gadget there are two input banks with incoming assets assumed to be r_1, r_2 and one output bank labelled x. Observe that Bank 8 is a CDS debtor with zero external assets but one positive debt contract, thus it satisfies condition (i) of the non-degenerate property. Bank 8 receives a payment of r_2 from the second input bank and a payment of r_{x} from Bank x, since the liability of x to 8 is equal to 1. The liability of Bank 8 is $l_8(r) = l_{8,9}(r) + l_{8,\mathrm{x}}(r) = 1 + 1 - r_5 = 1 + r_1$, so it holds that $r_8 = \min\left(1, \frac{a_8(r)}{l_8(r)}\right) = \min\left(1, \frac{r_2+r_{\mathrm{x}}}{1+r_1}\right)$.

If $\min\left(1, \frac{r_2+r_{\mathrm{x}}}{1+r_1}\right) = \frac{r_2+r_{\mathrm{x}}}{1+r_1}$, then Bank 8 submits a payment of $p_{8,x}(r) = r_8 \cdot (1 - r_5) = r_8 \cdot r_1 = \frac{r_2+r_x}{1+r_1} \cdot r_1$ to Bank x. For Bank x it holds that $r_{\mathrm{x}} = \min\left(1, p_{8,\mathrm{x}}(r)\right) = \min\left(1, \frac{r_2+r_x}{1+r_1} \cdot r_1\right)$, which by the hypothesis and since $r_1 \leq 1$ it must hold that $r_{\mathrm{x}} = \frac{r_2+r_x}{1+r_1} \cdot r_1$ where after calculations we get that $r_{\mathrm{x}} = r_1 \cdot r_2$.

If $\min\left(1, \frac{r_2+r_{\mathrm{x}}}{1+r_1}\right) = 1$, then $r_2 + r_{\mathrm{x}} \geq 1 + r_1$. Bank 8 submits a payment of $p_{8,\mathrm{x}}(r) = 1 - r_5 = r_1$ to Bank x, which in turns has a recovery rate vector of $r_{\mathrm{x}} = r_1$. This means that $r_2 \geq 1$ and thus r_2 must be 1 which means that the expression $r_x = r_1 \cdot r_2$ is satisfied.

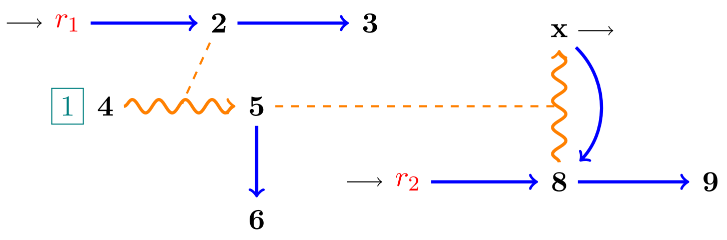


Figure 3.10: Alternative non-degenerate multiplication gadget where all contract notionals are assumed to be 1.

It is much more difficult to replace the degenerate gadget $g_/$. To do this, we will essentially get rid of division gates altogether in C_I', and replace each of them by a set of alternative operations that achieve the same result. We thus make a few modifications to C_I'. The first modification is that we alter the sub-circuit T, which was used to generate the value $t = 1/2^{2^d}$, which was used to scale all the signals in the circuit so that each signal is in $[0, 1]$ irrespective of the input vector. Here, d is a polynomial time computable value that satisfies that every signal in the original circuit C_I is at most 2^{2^d}. We adapt T such that the scaling factor it outputs is $1/(2^{1+2^d})$ instead of $1/2^{2^d}$. This can be done by adding one additional multiplication gate at the end of T. Let the output of the modified T be $t' = t/2$. After this modification, it holds that all signals inside the circuit are in $[0, 1/2]$.

We now observe that division gates are used in a very limited way in C_I' (with T modified as above).

- First, division is used for dividing by c, where c is a given explicit constant in the original circuit C_I. For such divisions, we can simply use our constant multiplication gadget g_{cr} to multiply by $1/c$, which is equivalent to dividing by c.

- Secondly, division is used to divide certain outputs of gates by t' (previously t), where $t' = 1/2^{1+2^d}$. This type of division happens in two cases.

    1. At every point in the circuit where two scaled signals $t'a$ and $t'b$ with $a, b \leq 2^{2^d}$ are multiplied with each other, resulting in the value $t't'ab$. This value is divided by t' in order to generate the signal $t'ab$, i.e., a scaled version of the signal ab in the original circuit.

    2. At the end of the circuit, where a scaled signal $t'a$ is divided by t' to produce an output signal of the original circuit, which is in $[0, 1]$.

  In the first case, let $x = t'ab$ and in the second case, let $x = a$, so that in both cases a number $t'x$ is divided by t' to result in x, and in both cases it holds that $x \in [0, 1]$. We replace the division $t'x/t'$ by a sequence of d gates that compute the square root of its input, followed by a multiplication by 2, followed by a sequence of d successive squaring multiplication gates, followed by a final multiplication by 2. This results in the correct value

$$((t'x)^{1/2^d} \cdot 2)^{2^d} \cdot 2 = (t'^{1/2^d} x^{1/2^d} \cdot 2)^{2^d} \cdot 2 = t'x \cdot 2^{2^d} \cdot 2 = \frac{1}{2} x \cdot 2 = x.$$

  It is furthermore straightforward to verify that, due to the order in which we apply our arithmetic operations, all of the values throughout this computation stay in the interval $[0, 1]$. We note that the adapted scaling factor $t' = t/2$ is needed because of the multiplication by 2 that is executed after the successive square roots and before the successive squaring.

The resulting circuit has no division gates anymore, so we do not need a division gadget in our reduction. Instead, now we need a square root gadget. Fortunately, it is possible to design this gadget in a non-degenerate way, although its construction is not very straightforward. It is presented in Figure 3.12, and it combines the two subgadgets given in Figures 3.2 and 3.11 which we provide separately for ease of understanding. The square root gadget may evidently output an irrational recovery rate.

$\boxed{\textbf{Square Root Gadget } g_{\sqrt{}}\textbf{:}}$ This gadget can be constructed in two parts. First we construct the gadget of Figure 3.11 which we call constant square root gadget $g_{\sqrt{}}$. This system has no input banks and one output bank denoted by y. Banks 2 and 5 have external assets of $1-c$, where c is assumed to be a constant in $[0,1]$. To compute the clearing vector we have to compute that $r_2 = \min\left(1, \frac{1-c}{2-r_5}\right)$ and $r_5 = \min\left(1, \frac{1-c}{2-r_2}\right)$. Observe that it is always the case that $1-c \leq 2-r_i$, for $i \in \{2,5\}$, thus in order to compute the clearing vector we have to solve the system $r_2 = \frac{1-c}{2-r_5}$ and $r_5 = \frac{1-c}{2-r_2}$, but these are exactly the computations in Example 2. So we know that under any clearing vector $r_2 = r_5 = 1-\sqrt{c}$, $c \in [0,1]$. Subsequently Bank x pays an amount of $1 - r_2 = \sqrt{c}$ to the output Bank y.

In order to construct the full square root gadget $g_{\sqrt{}}$, assuming a value r, we first use an inversion gadget g_{inv} to generate the value $1-r$ and subsequently a duplication gadget g_{dup} to duplicate the value $1-r$. We connect an output value $1-r$ of g_{dup} to Bank 2 and the other to 5 in a constant gadget $g_{\sqrt{c}}$, assuming $c=1$. Due to the selection of c both Banks 2 and 5 have zero external assets and receive a payment of $1-r$ from the output Banks of g_{dup}. From the analysis of $g_{\sqrt{c}}$ we know that the output value of Bank y is $\sqrt{r}$. The configuration of the square root gadget is illustrated in Figure 3.12

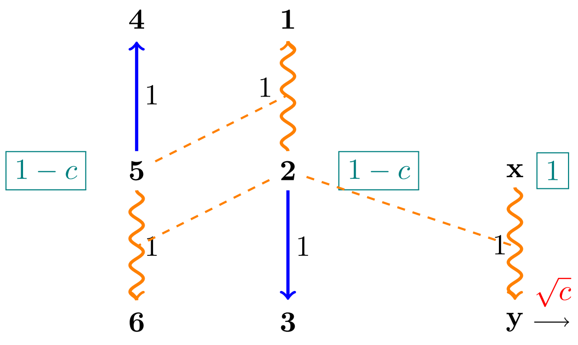


Figure 3.11: The square root constant gadget $g_{\sqrt{c}}$, $c \in [0,1]$ that has no input and outputs $\sqrt{c}$.

$$\begin{tikzcd} {} \arrow[r] & r \arrow[r] & g_{inv} \arrow[r] & g_{dup} \arrow[r, bend left] \arrow[r, bend right] & g_{\sqrt{c}} \end{tikzcd}$$

Figure 3.12: The square root gadget $g_{\sqrt{}}$.

A direct consequence of the above gadget is the connection of the default status of banks with the **SQRT-SUM** problem. An instance of **SQRT-SUM** consists of $n+1$ integers $d_1, d_2, ..., d_n, k$ and asks whether $\sum_{i=1}^{n} \sqrt{d_i} \leq k$. It is known that **SQRT-SUM** is solvable in PSPACE but it is unknown whether it is in P, or even in NP. In [Tiw92] it is shown that it can be solved in polynomial time in the unit-cost RAM model [Tiw92, Sch79, BMS81].

We show that for instances of both **CDS-CLEARING** and **CDS-PRIORITY-CLEARING**, determining for a given Bank i whether $r_i < 1$ (namely whether Bank i is in default) is at least as hard as solving the *square root sum* (**SQRT-SUM**) problem.

Proposition 3. *Given network $\mathcal{F}$ deciding whether a specific bank is in default is **SQRT-SUM**-hard.*

Proof. We prove the statement by reducing **CDS-CLEARING** from **SQRT-SUM**. Let $(d_1, ..d_n, k)$ be an instance of **SQRT-SUM**. Firstly, we note that in [BFHT85] it is shown that checking whether $\sum_{i=1}^{n} \sqrt{d_i} = k$ can be done in polynomial time. We check whether equality holds for our input first and proceed without loss of generality to the proof without minding equality.

We construct a financial system $\mathcal{F}$ as follows, first we construct n financial subnetworks which we refer to as *square root gadgets* and denote by $g_{i,\sqrt{\cdot}}$ the i-th square root gadget. Whenever referring to a node κ, belonging in a square root gadget $g_{i,\sqrt{\cdot}}$, we use the notation κ_i. Each square root gadget $g_{i,\sqrt{\cdot}}$ is augmented with two nodes x_i, y_i, and the CDS contract $(x_i, y_i, 2_i)$. Let $d_{\max} = \max_{i \in [n]} d_i$ and fix the external assets of nodes 2_i and 5_i to be $e_{2_i} = e_{5_i} = 1 - d_i/d_{\max}^2$. Moreover we let $e_{x_i} = 1$ and $e_{y_i} = 0$ and the CDS contract $(x_i, y_i, 2_i)$ have a notional of 1, i.e., $c_{x_i,y_i}^{2_i} = 1$. The n square root gadgets are all connected to a single node τ with $e_\tau = 0$, by n debt contracts $\{(y_i, \tau) : i \in [n]\}$. There is one further node τ' to which τ is connected through debt contract (τ, τ') with notional $c_{\tau,\tau'} = k/d_{\max}$. The construction is illustrated in Fig. 3.13.

We claim that this resulting financial system has a clearing recovery rate vector r with $r_\tau = 1$ if and only if $\sum_{i=1}^{n} \sqrt{d_i} \geq k$. From the analysis of the *square root gadget* in Figure 3.11, it follows that under any clearing recovery rate vector r, the recovery rate of node 2_i is $r_{2_i} = 1 - \sqrt{d_i}/d_{\max}$ for all $i \in [n]$. Since node 2_i is always in default (assuming all $d_i \neq 0$) the CDS $(x_i, y_i, 2_i)$ is activated and since node x_i can fully pay its liabilities, node y_i receives a payment of $1 - r_{2_i} = \sqrt{d_i}/d_{\max}$. This implies that τ receives a total payment of $(1/d_{\max}) \cdot \sum_{i \in [n]} \sqrt{d_i}$. Since τ has only one liability of $k/d_{\max}$, it holds that $r_\tau = 1$ if and only if the total payment that τ receives exceeds $k/d_{\max}$, i.e., if and only if $\sum_{i \in [n]} \sqrt{d_i} \geq k$, and this proves the claim. The construction in Figure 3.13

is independent of any priority profile which allows us to generalise the statement to instances of **CDS-PRIORITY-CLEARING**. □

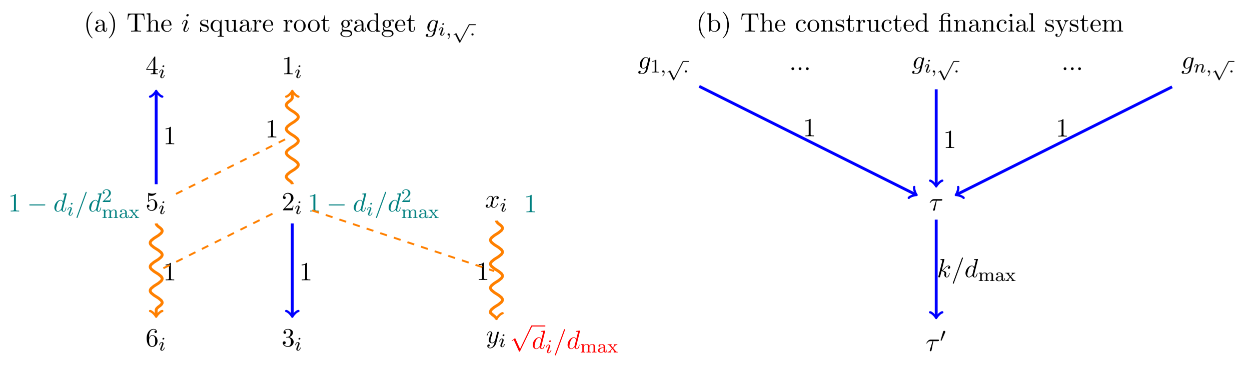


Figure 3.13: The financial system constructed from a given Square Root Sum instance.

The next gadgets illustrate that financial systems are able to naturally capture the max and min operations. However, these gadgets remain unused in the reduction of Theorem 3. $\boxed{\textbf{Maximum Gadget } g_{\max}\textbf{:}}$ We make use of the property $\max\{r_1, r_2\} = \frac{|r_1-r_2|+r_1+r_2}{2}$. This gadget receives two input values of r_1, r_2 and outputs a value of $\max\{r_1, r_2\}$. Figure 3.14 is a compact representation of the actual gadget, in the sense that is contains a set of sub-gadgets subscripted by g. First each input value r_1, r_2 enters a duplication gadget g_{dup}. Subsequently a copy of each value enters a gadget g_{abs} followed by a $g_{\frac{1}{2}r}$ gadget to maintain the value $\frac{|r_1-r_2|}{2}$. Another copy of each input value enter a $g_{\frac{1}{2}r}$ gadget respectively to maintain the values $\frac{r_1}{2}, \frac{r_2}{2}$. Finally two g_+ gadgets are employed to compute first the value $\frac{|r_1-r_2|+r_2}{2}$ and finally the value $\frac{|r_1-r_2|+r_1+r_2}{2}$.

$\boxed{\textbf{Minimum Gadget } g_{\min}\text{:}}$ We make use of the property $\min\{r_1, r_2\} = \frac{r_1+r_2-|r_1-r_2|}{2}$. Similarly to the maximum gadget, Figure 3.15 is a compact representation of the actual gadget. The configuration of the gadget resembles $g_{\max}$ in the sense that it generates the values $\frac{r_1+r_2}{2}, \frac{|r_1-r_2|}{2}$. The difference in this construction is that it uses a g_{pos-} gadget that takes as input the generated values $\frac{r_1+r_2}{2}, \frac{|r_1-r_2|}{2}$ and outputs the $\max\{0, \frac{r_1+r_2-|r_1-r_2|}{2}\}$, which is equivalent to $\max\{r_1, r_2\}$.

$$\begin{tikzcd}
 & & & r_1 \arrow[r, "1"] & g_{(1/2)r} \arrow[r, "1"] & \frac{r_1}{2} \arrow[rr, "1"] & & g_+ \arrow[r, "1"] & \max\{r_1, r_2\} \arrow[r] & {} \\
{} \arrow[r] & r_1 \arrow[r, "1"] & g_{\mathbf{dup}} \arrow[ur, "1"] \arrow[r, "1"] & r_1 \arrow[dr, "1"] & & & & & & \\
 & & & & g_{\mathbf{abs}} \arrow[r, "1"] & g_{(1/2)r} \arrow[r, "1"] & \frac{|r_1 - r_2|}{2} \arrow[r, "1"] \arrow[dd, "1"] & \frac{|r_1 - r_2| + r_2}{2} \arrow[uu, "1"] & & \\
{} \arrow[r] & r_2 \arrow[r, "1"] & g_{\mathbf{dup}} \arrow[r, "1"] \arrow[dr, "1"] & r_2 \arrow[ur, "1"] & & & & & & \\
 & & & r_2 \arrow[r, "1"] & g_{(1/2)r} \arrow[r, "1"] & \frac{r_2}{2} \arrow[r, "1"] & g_+ \arrow[uur, "1"] & & &
\end{tikzcd}$$

Figure 3.14: Maximum gadget $g_{\max}$, computing $\max\{r_1, r_2\}$. This is a compact representation where the nodes labeled with a subscripted g have to be replaced by copies of the respective gadgets, in order to obtain the full financial system defining the gadget.

$$\begin{tikzcd}
 & & & r_1 \arrow[r, "1"] & g_{(1/2)r} \arrow[rr, "1"] & & \frac{r_1}{2} \arrow[dr, "1"] & & & \\
{} \arrow[r] & r_1 \arrow[r, "1"] & g_{\mathbf{dup}} \arrow[ur, "1"] \arrow[r, "1"] & r_1 \arrow[dr, "1"] & & & & g_+ \arrow[r, "1"] & \frac{r_1 + r_2}{2} \arrow[ddd, "1"] & \\
 & & & & g_{\mathbf{abs}} \arrow[r, "1"] & g_{(1/2)r} \arrow[r, "1"] & \frac{|r_1 - r_2|}{2} \arrow[ddrr, "1"] & & & \\
{} \arrow[r] & r_2 \arrow[r, "1"] & g_{\mathbf{dup}} \arrow[r, "1"] \arrow[dr, "1"] & r_2 \arrow[ur, "1"] & & & & & & \\
 & & & r_2 \arrow[r, "1"] & g_{(1/2)r} \arrow[rr, "1"] & & \frac{r_2}{2} \arrow[uuur, "1"] & & g_{pos-}\{0, \frac{r_1 + r_2}{2} - \frac{|r_1 - r_2|}{2}\} \arrow[d, "1"] & \\
 & & & & & & & & \min\{r_1, r_2\} \arrow[r] & {}
\end{tikzcd}$$

Figure 3.15: Minimum gadget $g_{\min}$, computing $\min\{r_1, r_2\}$. This is a compact representation where the nodes labeled with a subscripted g have to be replaced by copies of the respective gadgets, in order to obtain the full financial system defining the gadget.

3.1.4 Financial System Gadgets for Singleton Liability Priorities

In Figures 3.17-3.20 we present a set of financial gadgets adapted to *Singleton Liability Priority* profiles that enable us to transfer the adapted reduction to financial systems with Credit Default Swaps and priority profiles and gain the same complexity result.

$\boxed{\textbf{Minimum Gadget } g_{\text{min}}\textbf{:}}$ This gadget computes the minimum of two input values r_1, r_2 in the framework of singleton liability priority profiles. Bank 6 pays according to the profile $P_6 = ((6,7,4) \mid (6,8))$ and receives a payment of r_1 from the second input and holds a liability towards Bank 7 of $l_{6,7}(r) = 1 - r_4 = r_2$. Bank 6 prioritises the CDS contract (6,7,4) and since it holds assets of r_1 it is evident that it pays Bank 7 an amount of r_1 if $r_1 \leq r_2$ and an amount r_2 if $r_1 \geq r_2$. Thus Bank 6 submits a payment of $\min(r_1, r_2)$ to Bank 7. Notice that compared to the gadget in Figure 3.15 this gadget's structure is significantly simpler.

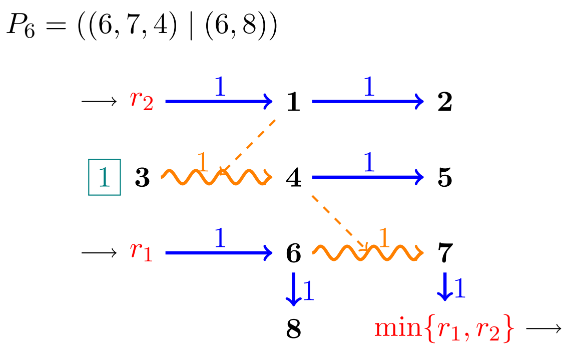


Figure 3.16: Minimum gadget g_{min}, computing $\min\{r_1, r_2\}$.

$\boxed{\textbf{Positive Subtraction Gadget } g_{pos-}\textbf{:}}$ This gadget is identical to the previous gadget with the difference that the output bank is Bank 8. It is not hard to see that given the analysis in the previous gadget Bank 6 after paying its first priority an amount of $\min r_1, r_2$ has to pay an amount of either 0 in case $r_1 - r_2 \leq 0$ or an amount of $r_1 - r_2$ in case $r_1 - r_2 \geq 0$. Evidently the amount of money Bank 8 receives from Bank 6 is $\max\{0, r_1 - r_2\}$ which is the output value that the gadget computes.

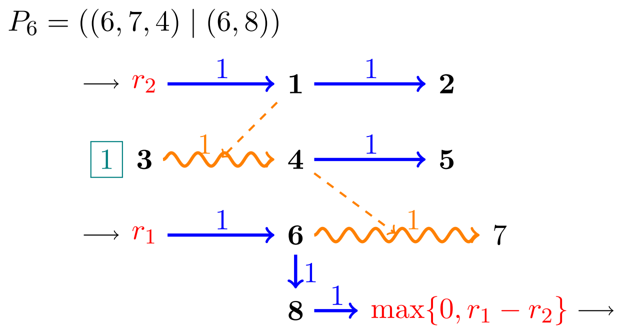
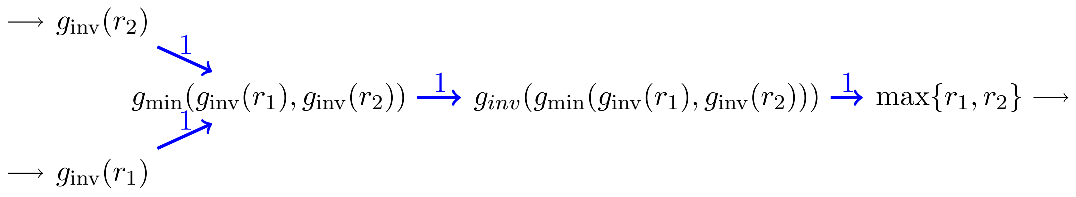


Figure 3.17: Positive subtraction gadget g_{pos-}, computing $\max\{0, r_1 - r_2\}$.

$\boxed{\textbf{Maximum gadget } g_{\max}\textbf{:}}$ This gadget computes the maximum value of two input recovery rate values r_1, r_2. The gadget is illustrated in Figure 3.18. We exploit the fact that $\max\{a, b\} = 1 - \min\{1 - a, 1 - b\}$ for $a, b \in [0, 1]$. Towards this direction the construction uses two inversion gadgets to compute the values $1 - r_1$ and $1 - r_2$. Subsequently the gadget appends the minimum gadget of Figure 3.17 to compute $\min\{1 - r_1, 1 - r_2\}$ and finally another inversion gadget to output the value $\max\{r_1, r_2\}$.



$g_{\text{inv}}(r_2)$
$g_{\text{inv}}(r_1)$
$g_{\min}(g_{\text{inv}}(r_1), g_{\text{inv}}(r_2))$
$g_{inv}(g_{\min}(g_{\text{inv}}(r_1), g_{\text{inv}}(r_2)))$
$\max\{r_1, r_2\}$
1

Figure 3.18: Maximum gadget $g_{\max}$, computing $\max\{r_1, r_2\}$

$\boxed{\textbf{Absolute difference gadget } g_{abs}\text{:}}$ This gadget uses the fact that $\max\{0, r_1 - r_2\} + \max\{0, r_2 - r_1\} = \mid r_1 - r_2 \mid$. The construction first uses two g_{pos-} gadgets on inputs (r_1, r_2) and (r_2, r_1) respectively to compute the values $\max\{0, r_1 - r_2\}$ and $\max\{0, r_2 - r_1\}$. Finally it employs an addition gadget g_+ to output the value $\mid r_1 - r_2 \mid$.

$$\begin{tikzcd}
{} \arrow[r] & g_{pos-}(r_1, r_2) \arrow[dr, "1"] & & & \\
& & g_+ \arrow[r, "1"] & {|r_1 - r_2|} \arrow[r] & {} \\
{} \arrow[r] & g_{pos-}(r_2, r_1) \arrow[ur, "1"] & & &
\end{tikzcd}$$

Figure 3.19: Absolute difference gadget g_{abs}.

$\boxed{\textbf{Multiplication gadget } g_*\textbf{:}}$ This construction is rather involved and makes use of various instances of multiplication-by-constant gadgets $g_{c\cdot r}$ and positive subtraction gadgets g_{pos-}. Some of the nodes have been annotated with expressions marked in red: These expressions are the recovery rates of some of the intermediate nodes in the gadget, in terms of the input nodes' recovery rates.

The purpose of Banks 1 to 19 is to compute the expression $\frac{r_1 r_2}{4} + \frac{r_1}{4} + \frac{1}{32} \cdot r_2 + \frac{1}{32}$. To verify that observe that Bank 6 has external assets 1 and total liability of $l_6(r) = l_{6,9}(r) + l_{6,7}(r) = 1 + 1 - r_4 = 1 + r_2$, thus its recovery rate is $r_6 = \frac{1}{1+r_2}$. Bank 13 has external assets of $\frac{1}{32}$, an incoming payment of $\frac{1}{4}r_1$ generated from the gadget $g_{\frac{1}{4}\cdot r_1}$ and liability of $l_{13}(r) = 1 - r_{11} = r_6 = \frac{1}{32}$, thus $r_{13} = \frac{1/4 \cdot r_1 + 1/32}{1/(1+r_2)} = \frac{r_1 \cdot r_2}{4} + \frac{r_1}{4} + \frac{r_2}{32} + \frac{1}{32}$. Bank 18 holds a liability of $l_{18}(r) = 1 - r_{16} = r_{13}$ and since it posses external assets of 1 it pays an amount of r_{13} to Bank 19. Bank 19 pays according to the profile $P_{19} = ((19,21) \mid (19,20))$, thus it prioritizes the debt contract of notional $1/32$ towards Bank 21. After paying Bank 21 the amount of $1/32$, Bank 19 submits the remaining amount $\frac{r_1}{4} + \frac{r_1 r_2}{4} + \frac{r_2}{32}$ to Bank 20. Next the construction employs a constant multiplication gadget $g_{\frac{1}{32}r_2}$ and transfers the output value $\frac{1}{32}r_2$ of this gadget along with the value r_{20} as inputs in a $g_{pos-}(r_{20}, g_{\frac{1}{32}r_2})$ gadget. Notice that in Figure 3.20 the second input signal r_2 is hidden in the compact representation of $g_{\frac{1}{32}r_2}$. The output value of $g_{pos-}(r_{20}, g_{\frac{1}{32}r_2})$ is $\frac{r_1}{4} + \frac{r_1 r_2}{4}$ which is transferred along with the value $\frac{1}{4}r_1$ that is generated by applying a $g_{\frac{1}{4}r_1}$ gadget as input values to another g_{pos-} gadget. The output of the later is the value $\frac{r_1 r_2}{4}$. Finally the signal $\frac{r_1 r_2}{4}$ is transferred as input to a g_{4r} gadget that eventually outputs the desired value $r_1 r_2$.

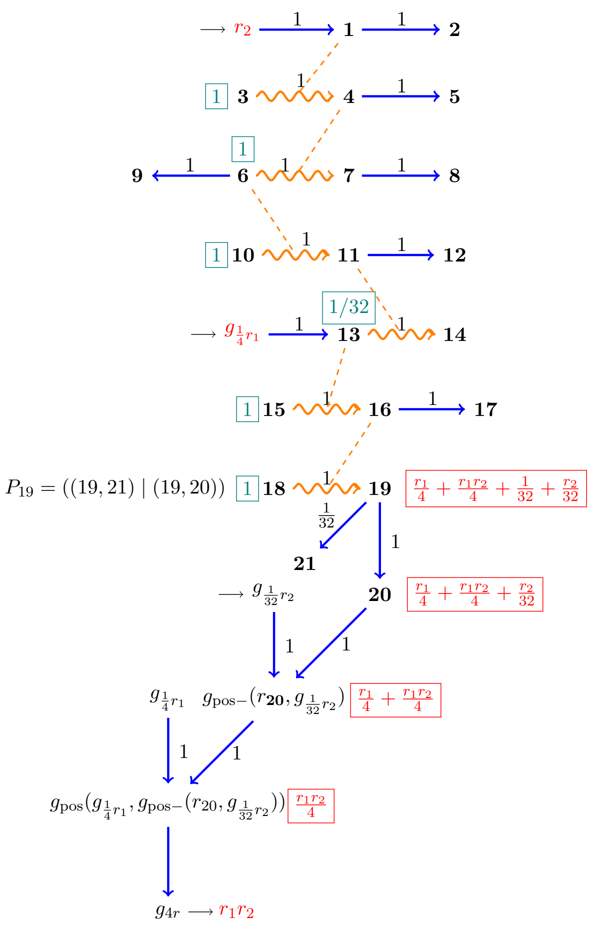

r_2
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
$\boxed{1}$
$\boxed{1/32}$
$g_{\frac{1}{4}r_1}$
$P_{19} = ((19,21) \mid (19,20))$
$\frac{1}{32}$
$\frac{r_1}{4} + \frac{r_1 r_2}{4} + \frac{1}{32} + \frac{r_2}{32}$
$\frac{r_1}{4} + \frac{r_1 r_2}{4} + \frac{r_2}{32}$
$g_{\frac{1}{32}r_2}$
$g_{\text{pos}-}(r_{\mathbf{20}}, g_{\frac{1}{32}r_2})$
$\frac{r_1}{4} + \frac{r_1 r_2}{4}$
$g_{\text{pos}}(g_{\frac{1}{4}r_1}, g_{\text{pos}-}(r_{20}, g_{\frac{1}{32}r_2}))$
$\frac{r_1 r_2}{4}$
g_{4r}
$r_1 r_2$

Figure 3.20: Multiplication gadget g_*: This gadget's construction is rather involved and makes use of various instances of multiplication-by-constant gadgets and positive subtraction gadgets. Some of the nodes have been annotated with expressions marked in red: These expressions are the recovery rates of some of the intermediate nodes in the gadget, in terms of the input nodes' recovery rates.

3.2 Computing Weak Approximations

3.2.1 ϵ-CDS-CLEARING

We proceed with the study of the *weak approximation* version of **CDS-CLEARING**. Given $\mathcal{F} \in$ **CDS-CLEARING** we define a *weak ϵ-approximate clearing recovery rate vector (ϵ-*CRRV*)* of $\mathcal{F}$ as follows:

Definition 7 (**Weak ϵ-approximate clearing recovery rate vector** (ϵ-*CRRV*))**.** *Given a financial network $\mathcal{F} = (N, e, c)$, a recovery rate vector $r \in [0,1]^n$ is called weak ϵ-approximate clearing iff for all banks $i \in N$, it holds that:*

(i.) *if $e_i > \sum_{j \in N \setminus \{i\}} \left(c_{i,j} + \sum_{k \in N \setminus \{i,j\}} c^k_{i,j} \right)$, then $r_i = 1$;*

(ii.) *otherwise, $\|r_i - f_{\mathcal{F}}(r_i)\|_\infty \leq \epsilon$, where $f_{\mathcal{F}}$ is the function induced by* (2.6) *with respect to instance $\mathcal{F}$ defined above.*

Condition *(i.)* guarantees that a bank with a "trivial" recovery rate value of 1 under any clearing vector, is indeed set to 1 under this notion of weak ϵ-approximate clearing. For those banks, r_i can be omitted from $f_{\mathcal{F}}$ as a variable, so that our notion of an ϵ-*CRRV* can strictly correspond to a weak ϵ-approximate fixed point of the function $f_{\mathcal{F}}$ in the standard sense. Condition *(ii)* suggests that the vector r is not far removed from satisfying the clearing condition. Remember though that, as we established in Section 2.2.3, this does not imply that r must be close to an actual clearing vector. For a more involved analysis on the weak approximation notion we point the reader to Section 2.4 of [SSB17a].

We define the *weak*-approximate version of **CDS-CLEARING** based on the above definition of ϵ-*CRRV* as ϵ-**CDS-CLEARING**.

Definition 8 (ϵ-**CDS-CLEARING**)**.** *Given a financial network $\mathcal{F} \in$* **CDS-CLEARING** *compute a weak ϵ-approximate clearing recovery rate vector (ϵ-CRRV).*

3.2.2 The PURE-CIRCUIT Problem

We use the problem **PURE-CIRCUIT** [DFHM22]. The problem features a set of variables which can take one of three different values 0, 1 or garbage and a circuit constructed from connecting gates of the following three types: NOT, OR and PURIFY. The first two basically follow boolean logic whereas the PURIFY gate has one input and two outputs and makes sure to duplicate the pure bit in input or produce at least one pure bit in output if the input is garbage.

Formally an instance $I = (V, G)$ of **PURE-CIRCUIT** consists of a set of gates G and variables V. The gates are represented by tuples $g = (T, u, w)$ or $g = (T, u, v, w)$, where T represents the gate type and $u, v, w \in V$ are either input or output variables, depending on T. Our analysis is based on the following gates:

- **NOT**-gate: $(\mathbf{NOT}, u, w)$, where u is the input variable and w is the output variable.

$$\begin{tikzcd} u \arrow[r] & \mathbf{NOT} \arrow[r] & w \end{tikzcd}$$

Figure 3.21: Graphical representation of the **NOT**-gate

- **OR**-gate: $(\mathbf{OR}, u, v, w)$, where u and v are the input variables and w is the output variable.

$$\begin{tikzcd} u \arrow[dr] & & v \arrow[dl] \\ & \mathbf{OR} \arrow[d] & \\ & w & \end{tikzcd}$$

Figure 3.22: Graphical representation of the **OR**-gate

- **PURIFY**-gate: $(\mathbf{PURIFY}, u, v, w)$, where u is the input and v, w are the output variables.

$$\begin{tikzcd} & u \arrow[d] & \\ v & \mathbf{PURIFY} \arrow[l] \arrow[r] & w \end{tikzcd}$$

Figure 3.23: Graphical representation of the **PURIFY**-gate

Each variable appearing in a gate g is assigned a value from the set $\{0, 1, \perp\}$, where the value $\perp$ is termed "garbage". We denote an assignment of values to variables by the function $\mathrm{x} : V \mapsto \{0, 1, \perp\}$. We are interested in assignments that satisfy the above gates.

- An assignment x satisfies gate $(\mathbf{NOT}, u, w)$ iff:
    1. $\mathrm{x}[u] = 0 \rightarrow \mathrm{x}[w] = 1$

2. $\mathrm{x}[u] = 1 \rightarrow \mathrm{x}[w] = 0$
3. $\mathrm{x}[u] = \perp \rightarrow \mathrm{x}[w] \in \{0, 1, \perp\}$

- An assignment x satisfies gate $(\mathbf{OR}, u, v, w)$ iff:
  1. $\mathrm{x}[u] = \mathrm{x}[v] = 0 \rightarrow \mathrm{x}[w] = 0$.
  2. $\mathrm{x}[u] = 1$ and $\mathrm{x}[v] \in \{0, 1, \perp\} \rightarrow \mathrm{x}[w] = 1$
  3. $\mathrm{x}[u] \in \{0, 1, \perp\}$ and $\mathrm{x}[v] = 1 \rightarrow \mathrm{x}[w] = 1$
  4. Else $\mathrm{x}[w] = \{0, 1, \perp\}$
- An assignment x satisfies gate $(\mathbf{PURIFY}, u, v, w)$ iff:
  1. $\mathrm{x}[u] = 0 \rightarrow \mathrm{x}[v] = \mathrm{x}[w] = 0$.
  2. $\mathrm{x}[u] = 1 \rightarrow \mathrm{x}[v] = \mathrm{x}[w] = 1$.
  3. $\mathrm{x}[u] = \perp \rightarrow \mathrm{x}[v] \in \{0, 1\}$ or $\mathrm{x}[w] \in \{0, 1\}$.

Below we define the version of **PURE-CIRCUIT** problem that we will use in our subsequent proof and illustrate it with an example.

Definition 9 (**PURE-CIRCUIT**). *An instance $I = (V, G)$ of* **PURE-CIRCUIT** *consists of a set of variables V, gates G of the form $g = (T, u, v, w)$ or $g = (T, u, w)$, where $T \in \{\mathbf{NOT}, \mathbf{OR}, \mathbf{PURIFY}\}$ and no two gates share the same output variable. A solution is an assignment $\mathrm{x} : V \mapsto \{0, 1, \perp\}$ that satisfies all gates in G.*

Example 4. *Consider the instance $I = (V, G)$ with $V = \{u, v, w, y\}$ and $G = \{(\boldsymbol{NOT}, u, v), (\boldsymbol{OR}, v, w, y), (\boldsymbol{PURIFY}, v, u, w)\}$. No variable is output to more than one gate, which constitutes I a valid instance. In any satisfying assignment x, neither $x[u]$ nor $x[v]$ can lie in $\{0, 1\}$, thus it must hold that $x[v] = x[u] = \perp$ and $x[w] \neq \perp$, otherwise the PURIFY gate is not satisfied. If we set $x[w] = 1$ then also $x[y] = 1$ due to the OR-gate. Therefore, a satisfying assignment is $x[u] = x[v] = \perp$ and $x[w] = x[y] = 1$ The graphical illustration of I is presented in Figure 3.24.*

Finally we will make use of the following theorem to prove our claims.

Theorem 5 ([DFHM22], Corollary 2.2). **PURE-CIRCUIT** *is PPAD-complete.*

3.2.3 PPAD-hardness of ϵ-CDS-CLEARING

State of the art bound: In [SSB17b], it is established that there exists a small constant ϵ for which computing an ϵ-*CRRV* is PPAD-*hard*. To prove that, the authors defined a variation (Definition

$$\begin{tikzcd}
y & \textbf{OR} \arrow[l] & v \arrow[l] \arrow[dl] \\
 & \textbf{PURIFY} \arrow[d] \arrow[dr] & \textbf{NOT} \arrow[u] \\
 & w \arrow[uu] & u \arrow[u]
\end{tikzcd}$$

Figure 3.24: The graphical illustration of the **PURE-CIRCUIT** instance I.

5.4 [SSB17b]) of the problem ϵ-**GENERALISED-CIRCUIT** that was defined in [CDT09] and was proved to be PPAD-*hard* in [Rub15] for some unknown constant ϵ. Subsequently they showed that there exists an $\epsilon > 0$ such that computing ϵ approximate solutions to circuits C' of this variation is PPAD-hard (Lemma 5.6 [SSB17b]). Next they connected their version with the original used in [Rub15] and showed that $\epsilon/5$ solutions to circuits C' of their version correspond to ϵ solutions to circuits C of the original version (Lemma 5.6 [SSB17b]). Additionally in Theorem 4.1 of [DFHM22] it is established that the original version of ϵ-**GENERALISED-CIRCUIT** used in [Rub15] is PPAD-hard for $\epsilon < 0.1$. Combined these facts imply that the version presented in [SSB17b] must be inapproximable up to any factor less than 1/50. Finally it is proved that $\epsilon/3$-*CRRVes* correspond to ϵ solutions for circuits C' of the new version of ϵ-**GENERALISED-CURCUIT** (Theorem 5.1 [SSB17b]), which combined with the inapproximability bound of [DFHM22] implies that computing ϵ-*CRRV*es is PPAD-hard for $\epsilon < 1/150$.

Improved Explicit bound: We apply a new method to provide the first explicit inapproximability bound for ϵ-**CDS-CLEARING**. We construct a direct reduction from **PURE-CIRCUIT** to ϵ-**CDS-CLEARING**. The proof is technically involved and divided in the following parts:

(i) A "hypothetical" encoding of the recovery rate space $[0, 1]$ into three disjoint subintervals that intuitively map recovery rates to either a pure bit or "garbage".

(ii) The construction of three "financial gates" whose behaviour under weak approximate clearing vectors must simulate the function of **PURE-CIRCUIT** gates after decoding the recovery rates of specific banks.

(iii) A reverse engineering process that fixes the initial encoding for the recovery rate values and results in the final specified constant ϵ.

The proposed reduction is optimised, in the sense that it generates a family of possible encodings for the recovery rate values, upon which we choose the one that maximises the inapproximability parameter ϵ. The structure of the proposed networks combined with the restriction of computations to intervals within the $[0,1]$ range, significantly increases the complexity of the analysis.

To bypass this obstacle we define a *first-order language* denoted as $L(\mathrm{R},\mathrm{F},\mathrm{C})$, designed to handle combined arithmetic operations involving intervals and numbers within the range $[0,1]$. This is a basic tool in our analysis since it combines concise representations with a high level of expressiveness. The upshot is that if we were able find in polynomial time, a solution to ϵ-**CDS-CLEARING** for ϵ less than (roughly) 0.048 then the encoding would generate a polynomial time algorithm for **PURE-CIRCUIT**. We proceed with the reduction.

The mapping m_δ. We construct a mapping of $[0,1]$ to $\{0,1,\perp\}$. The domain of this mapping reflects potential recovery rates of banks in a financial network, and these are mapped to values for variables of **PURE-CIRCUIT** instances. The mapping will be used to translate recovery rate vectors of a financial network to assignments x of a corresponding **PURE-CIRCUIT** instance. Fix δ to be any value in $(0,\frac{1}{2})$, and consider the following concrete mapping m_δ:

- If $r \in \left[0, \frac{1}{2}-\delta\right]$, then $m_\delta(r) = 0$;
- If $r \in \left(\frac{1}{2}-\delta, \frac{1}{2}+\delta\right)$, then $m_\delta(r) = \perp$;
- If $r \in \left[\frac{1}{2}+\delta, 1\right]$, then $m_\delta(r) = 1$.

We will use $m_\delta(r)$ throughout the remainder of the section. A concrete choice of δ will be defined later, as it will follow from analysis what the optimal choice of δ is (in the sense of achieving the strongest inapproximability result).

A first-order language[5]**.** Let $L(\mathbf{R},\mathbf{F},\mathbf{C})$ be a *first-order language* where

- $\mathbf{R} = \{\preceq\}$ is a *relation symbol set.*
- $\mathbf{F} = \{\pm, +, -, \cdot, /\}$ is a *function symbol set* that contains symbols that represent binary operations.
- $\mathbf{C} = \{\mathbf{0}, \mathbf{1}, \cdots, \mathbf{n}, \cdots\}$ is a *constant symbol set* that contains a corresponding constant symbol for each non-negative rational number.

[5]In defining the language we followed the notation in [Fit90].

We define the notion of a *term* following the standard definition within the context of first-order languages as follows.

- Any *variable symbol* χ is a *term.*
- Any *constant symbol* in $\mathbf{C} = \{\mathbf{0}, \mathbf{1}, \cdots\}$ is a *term.*
- If ϕ, ϕ' are *terms* and $\circ \in \mathbf{F}$ is a binary function symbol then $\phi \circ \phi'$ is a *term.*

We define the *model* $\mathbf{M} = \langle \mathbf{D}, \mathbf{I} \rangle$ for the above language to consist of the *domain* $\mathbf{D} = \{[x, y] | x, y \in \mathbb{R} \text{ and } x \leq y\} \cup \{\emptyset\}$ and the *interpretation* $\mathbf{I}$ that associates the relation symbol $\preceq$ to the relation:

$$\preceq^{\mathbf{I}} = \Big\{ (\mathbf{x}, \mathbf{y}) \mid \text{x and y are intervals where} \inf\{\mathbf{x}\} \leq \inf\{\mathbf{y}\} \text{ and } \sup\{\mathbf{x}\} \leq \sup\{\mathbf{y}\} \Big\}. \tag{3.1}$$

For notation simplicity we denote by $\mathbf{x}$ variables that represent intervals and by x variables that represent numbers. Let $\mathbf{A}$ be an assignment of variables to values in $\mathbf{D}$. We use the notation $x^{\mathbf{A}}$ to denote the value within the domain $\mathbf{D}$ that the variable x is assigned to under the assignment $\mathbf{A}$. Notice that x is a variable of the language, whereas $x^{\mathbf{A}}$ is a numerical value that belongs in the domain $\mathbf{D}$. When it is obvious from the context, we would rather use the notation $\mathbf{x}$ to also refer to the numerical value of $\mathbf{D}$ that $\mathbf{x}$ represents in order to prevent notational overflow.

For a *term* ϕ we recursively define $\phi^{\mathbf{I},\mathbf{A}}$, i.e, the value that the term ϕ represents under the interpretation $\mathbf{I}$ and the assignment $\mathbf{A}$ as follows

- If ϕ is a variable then $\phi^{\mathbf{I},\mathbf{A}} = \phi^{\mathbf{A}}$.
- If ϕ is a constant then w.l.o.g $\phi = \mathbf{c}$, where by default under any assignment $\mathbf{A}$, $\mathbf{c}^{\mathbf{A}} = c$. For example $(\mathbf{1/2})^{\mathbf{A}} = 1/2$
- If ϕ is none of the above then by the definition of *term* it holds that $\phi = \phi_1 \circ \phi_2$, where ϕ_1, ϕ_2 are terms. In that case $\phi^{\mathbf{I},\mathbf{A}} = (\phi_1^{\mathbf{I},\mathbf{A}}) \circ^{\mathbf{I}} (\phi_2^{\mathbf{I},\mathbf{A}})$.

From the above definition in order to completely define the first order language we have to interpret the binary function symbols in $\mathbf{F}$ on variables and constants.

- If x, y are variables that represent numbers or constant symbols of $\mathbf{D}$ under an assignment $\mathbf{A}$, then $(x \circ y)^{\mathbf{I},\mathbf{A}} = x^{\mathbf{A}} \circ y^{\mathbf{A}}$, where $\circ = \{+, -, *, /\}$. For example assume the term $(\mathbf{1} + x)$ and the assignment $\mathbf{A}$ under which the variable x is assigned to the value 1. Then $(\mathbf{1}+x)^{\mathbf{I},\mathbf{A}} = 1 + x^{\mathbf{A}} = 2$.

- If x and ϵ are variables assigned to real numbers with $\epsilon > 0$ under an assignment $\mathbf{A}$ then

$$(x \pm \epsilon)^{\mathbf{I},\mathbf{A}} = \begin{cases} \Big[x-\epsilon, x+\epsilon\Big] \cap \Big[0,1\Big], & \text{if } x \in \Big[0,1\Big] \\ \Big[1-\epsilon, 1\Big] \cap \Big[0,1\Big], & \text{if } x > 1 \\ \Big[0,\epsilon\Big] \cap \Big[0,1\Big], & \text{if } x < 0 \end{cases} \tag{3.2}$$

For example assume the assignment $\mathbf{A}$ where $x^{\mathbf{A}} = 2$ and $\epsilon^{\mathbf{A}} = 1/4$. Then $(x \pm \epsilon)^{\mathbf{I},\mathbf{A}} = [\frac{3}{4}, 1]$. Essentially operation $\pm^{\mathbf{I}}$ creates a closed ϵ-ball around the value of x, restricted to values that fall within $[0,1]$, with the quirk that the ball is centered to 1 if $x > 1$ and 0 if $x < 0$. The result of $\pm^{\mathbf{I}}$ on two numbers is always a range within $[0,1]$.

- If variable $\mathbf{x}$ is assigned to an interval within $[0,1]$ and $\epsilon > 0$ represents a real number under an assignment $\mathbf{A}$ then

$$(\mathbf{x} \pm \epsilon)^{\mathbf{I},\mathbf{A}} = \Big[\inf\{\mathbf{x}\} - \epsilon, \sup\{\mathbf{x}\} + \epsilon\Big] \cap \Big[0,1\Big] \tag{3.3}$$

For example if $\mathbf{x}^{\mathbf{A}} = [\frac{1}{4}, \frac{1}{2}]$ and $\epsilon^{\mathbf{A}} = \frac{1}{4}$ then $(\mathbf{x} \pm \epsilon)^{\mathbf{I},\mathbf{A}} = [0, \frac{3}{4}]$.

- If variable $\mathbf{x}$ is assigned to an interval within $[0,1]$ under an assignment $\mathbf{A}$ then

$$(\mathbf{1} - \mathbf{x})^{\mathbf{I},\mathbf{A}} = \Big[1 - \sup\{\mathbf{x}\}, 1 - \inf\{\mathbf{x}\}\Big] \tag{3.4}$$

For example if $\mathbf{x}^{\mathbf{A}} = [\frac{3}{4}, 1]$ then $(\mathbf{1} - \mathbf{x})^{\mathbf{I},\mathbf{A}} = [0, \frac{1}{4}]$.

- If $\mathbf{x} = x \pm \epsilon_1$ and $\mathbf{y} = y \pm \epsilon_2$ represent intervals within the range $[0,1]$ then under an assignment $\mathbf{A}$ where $\epsilon > 0$

$$(\mathbf{x} + \mathbf{y})^{\mathbf{I},\mathbf{A}} = \begin{cases} \Big[\inf\{\mathbf{x}\} + \inf\{\mathbf{y}\}, \sup\{\mathbf{x}\} + \sup\{\mathbf{y}\}\Big] \cap \Big[0,1\Big], & \text{if } x + y \in [0,1] \\ \Big[1 - (\epsilon_1 + \epsilon_2), 1)\Big] \cap \Big[0,1\Big], & \text{if } x + y > 1 \\ \Big[0, \epsilon_1 + \epsilon_2\Big] \cap \Big[0,1\Big], & \text{if } x + y < 0 \end{cases} \tag{3.5}$$

For example if $x^{\mathbf{A}} = [\frac{1}{4}, \frac{3}{4}]$ and $y^{\mathbf{A}} = [\frac{1}{3}, \frac{2}{3}]$ then $(\mathbf{x} + \mathbf{y})^{\mathbf{I},\mathbf{A}} = [\frac{7}{12}, 1]$

- If $\mathbf{x} = x \pm \epsilon$ represents an interval within $[0,1]$ and $\mathbf{l}$ is a constant symbol that represents a constant number bigger than or equal to 1, i.e, $\mathbf{l}^{\mathbf{A}} \geq 1$ then

$$(\mathbf{l} \cdot \mathbf{x})^{\mathbf{I},\mathbf{A}} = \begin{cases} \Big[l \cdot \inf\{\mathbf{x}\}, l \cdot \sup\{\mathbf{x}\}\Big] \cap \Big[0,1\Big], & \text{if } l \cdot x \in [0,1] \\ \Big[1 - l \cdot \epsilon, 1\Big] \cap \Big[0,1\Big], & \text{if } l \cdot x > 1 \\ \Big[0, l \cdot \epsilon\Big] \cap \Big[0,1\Big], & \text{if } l \cdot x < 0 \end{cases} \tag{3.6}$$

Finally we assume that all other terms expressed in cases of operations on symbols that are not captured in any of the above definitions are mapped by any assignment to the value $\emptyset$ of $\mathbf{D}$. Examples of such terms are $(\mathbf{x} \pm \mathbf{y})$ where the operation $\pm$ is not defined on intervals within $\mathbf{D}$ or $x * \mathbf{x}$ where x is a variable that represents a number and $\mathbf{x}$ is a variable that represents an interval. Observe that from all the above definitions it is evident that all operations expressed via the symbols within the $\mathbf{F}$ map intervals within $[0,1]$ to intervals within $[0,1]$.

We proceed by defining the notion of a *valid substitution* of a term τ with a term τ'.

Definition. *A substitution σ of a term τ with a term τ' denoted as $\sigma(\tau)$ is said to be* valid *if $(\sigma(\tau))^{\boldsymbol{I},\boldsymbol{A}} = \tau^{\boldsymbol{I},\boldsymbol{A}}$ for every assignment $\boldsymbol{A}$.*

Essentially a *valid* substitution between two terms implies that under the interpretation $\mathbf{I}$ and any assignment $\mathbf{A}$, both terms refer to the same value within the domain $\mathbf{D}$. Below we present a set of *valid substitutions* each with a discrete notation that we are about to use in our proofs.

Lemma 1. *The following are valid substitutions.* [6]

σ_{1-}*: If $\boldsymbol{x} = x \pm \epsilon$, then $\sigma_{1-}(1 - \boldsymbol{x}) = (1 - x) \pm \epsilon$.*

$\sigma_{\pm}$*: If $\boldsymbol{x} = x \pm \epsilon_1$ and $\epsilon_2 > 0$, then $\sigma_{\pm}(\boldsymbol{x} \pm \epsilon_2) = x \pm (\epsilon_1 + \epsilon_2)$.*

σ_{+}*: If $\boldsymbol{x} = (x \pm \epsilon_1)$ and $\boldsymbol{y} = (y \pm \epsilon_2)$, then $\sigma_{+}(\boldsymbol{x} + \boldsymbol{y}) = (x + y) \pm (\epsilon_1 + \epsilon_2)$.*

σ_{*}*: If $\boldsymbol{x} = x \pm \epsilon$ and $l \geq 1$, then $\sigma_{*}(l \cdot \boldsymbol{x}) = l \cdot x \pm l \cdot \epsilon$.*

Proof. For each substitution σ and a term τ, we will prove that $(\sigma(\tau))^{\mathbf{I},\mathbf{A}} = \tau^{\mathbf{I},\mathbf{A}}$, for any assignment $\mathbf{A}$. Namely the interpretation dictated by $\mathbf{I}$ of the terms connected via any of the above substitutions

[6] For simplicity we assume that the variables $\epsilon, \epsilon_1, \epsilon_2$ are always mapped to positive numbers and $\mathbf{l}$ is a constant that represents a rational number greater than or equal to one.

represent the same set of values from the domain $\mathbf{D}$. Notice that assignments where both terms are mapped to $\emptyset$, are trivially satisfied and, as such, will not be explicitly addressed in the subsequent analysis.

σ_{1-}: Let $\mathbf{k} = 1 - \mathbf{x}$. Combining the hypothesis with the definition of operation 3.4 we get that $\mathbf{k}$ represents a range of values within $[0, 1]$ and $\sup\{\mathbf{k}\} = 1 - \inf\{\mathbf{x}\}$ and $\inf\{\mathbf{k}\} = 1 - \sup\{\mathbf{x}\}$. If $x \in [0, 1]$ then

(a) $\sup\{\mathbf{k}\} = 1 - \max(0, x - \epsilon) = \min(1, (1 - x) + \epsilon)$.

(b) $\inf\{\mathbf{k}\} = 1 - \min(1, x + \epsilon) = \max(0, (1 - x) - \epsilon)$.

In the case where $x > 1$ then $1 - x < 0$ and

(a) $\sup\{\mathbf{k}\} = 1 - \max(0, (1 - \epsilon)) = \min(1, \epsilon)$

(b) $\inf\{\mathbf{k}\} = 1 - \min(1, 1 + \epsilon) = 0$

And if $x < 0$ then $1 - x > 1$ meaning that

(a) $\sup\{\mathbf{k}\} = 1 - \max(0, 1 - \epsilon) = 1$.

(b) $\inf\{\mathbf{k}\} = 1 - \min(1, \epsilon) = \max(0, 1 - \epsilon)$.

From the above analysis we conclude that,

$$(1 - \mathbf{x})^{\mathbf{I,A}} = \begin{cases} \Big[(1 - x) - \epsilon, (1 - x) + \epsilon\Big] \cap \Big[0, 1\Big], & \text{if } 1 - x \in [0, 1] \\ \Big[1 - \epsilon, 1\Big] \cap \Big[0, 1\Big], & \text{if } 1 - x > 1 \\ \Big[0, \epsilon\Big] \cap \Big[0, 1\Big], & \text{if } 1 - x < 0 \end{cases}$$

which proves the claim that if $\mathbf{x} = x \pm \epsilon$ then $\Big(1 - \mathbf{x}\Big)^{\mathbf{I,A}} = \Big((1 - x) \pm \epsilon\Big)^{\mathbf{I,A}}$, for any assignment $\mathbf{A}$. Consequently $\sigma_{1-}(1 - \mathbf{x}) = (1 - x) \pm \epsilon$ is a *valid substitution*.

$\sigma_{\pm}$: Let $\mathbf{k} = \mathbf{x} \pm \epsilon_2$. Combining the given hypothesis with the definition of the operation 3.3, it is evident that the variable $\mathbf{k}$ must represent a range of values within the interval $[0, 1]$. According to 3.3, we can determine that $\sup\{\mathbf{k}\} = \min(1, \sup\{\mathbf{x}\} + \epsilon_2)$ and $\inf\{\mathbf{k}\} = \max(0, \inf\{\mathbf{x}\} - \epsilon_2)$. Furthermore, considering the operation described in 3.2, if $x \in [0, 1]$, then:

(a) $\sup\{\mathbf{k}\} = \min(1, x + \epsilon_1 + \epsilon_2)$.

(b) $\inf\{\mathbf{k}\} = \max(0, x - \epsilon_1 - \epsilon_2)$.

In cases where $x > 1$, then $\sup\{\mathbf{x}\} = 1$ and $\inf\{\mathbf{x}\} = \max(0, 1 - \epsilon_1)$ which implies that:

(a) $\sup\{\mathbf{k}\} = 1$.

(b) $\inf\{\mathbf{k}\} = \max(0, 1 - \epsilon_1 - \epsilon_2)$.

And when $x < 0$, it holds that: $\sup\{\mathbf{x}\} = \min(1, \epsilon_1)$ and $\inf\{\mathbf{x}\} = 0$, which leads to the conclusion that

(a) $\sup\{\mathbf{k}\} = \min(1, \epsilon_1 + \epsilon_2)$.

(b) $\inf\{\mathbf{x}\} = 0$.

In summary of the preceding analysis, we can conclude that,

$$(\mathbf{x} \pm \epsilon_2)^{\mathbf{I},\mathbf{A}} = \begin{cases} \Big[x - (\epsilon_1 + \epsilon_2), x + (\epsilon_1 + \epsilon_2)\Big] \cap \Big[0, 1\Big], & \text{if } x \in [0,1] \\ \Big[1 - (\epsilon_1 + \epsilon_2), 1\Big] \cap \Big[0, 1\Big], & \text{if } x > 1 \\ \Big[0, \epsilon_1 + \epsilon_2\Big] \cap \Big[0, 1\Big], & \text{if } x < 0 \end{cases}$$

which proves the claim that if $\mathbf{x} = x \pm \epsilon_1$ then $\Big(\mathbf{x} \pm \epsilon_2\Big)^{\mathbf{I},\mathbf{A}} = \Big(x \pm (\epsilon_1 + \epsilon_2)\Big)^{\mathbf{I},\mathbf{A}}$. Consequently $\sigma_{\pm}(\mathbf{x} \pm \epsilon_2) = x \pm (\epsilon_1 + \epsilon_2)$ is a valid substitution.

σ_+: Consider $\mathbf{k} = \mathbf{x} + \mathbf{y}$. It's essential to note that, as per the definition in operation 3.2, both variables $\mathbf{x}$ and $\mathbf{y}$ represent intervals constrained to the range $[0, 1]$. This, in conjunction with operation 3.5, underscores that the addition of intervals is exclusively defined within the boundaries of $[0, 1]$.

In line with operation 3.5, when $x + y \in [0, 1]$, we observe that

(a) $\sup\{\mathbf{k}\} = \min(1, \sup\{\mathbf{x}\} + \sup\{\mathbf{y}\})$

(b) $\inf\{\mathbf{k}\} = \max(0, \inf\{\mathbf{x}\} + \inf\{\mathbf{y}\})$.

Whenever $x + y > 1$, then in line with the definition provided in operation 3.5, it holds that:

(a) $\sup\{\mathbf{k}\} = 1$

(b) $\inf\{\mathbf{k}\} = \max(0, 1 - (\epsilon_1 + \epsilon_2))$

Finally whenever $x + y < 1$:

(a) $\sup\{\mathbf{k}\} = \min(1, \epsilon_1 + \epsilon_2)$.

(b) $\inf\{\mathbf{k}\} = 0$.

Directly from 3.5 and by the above analysis we get that,

$$(\mathbf{x} + \mathbf{y})^{\mathbf{I,A}} = \begin{cases} \Big[(x+y) - (\epsilon_1 + \epsilon_2), (x+y) + (\epsilon_1 + \epsilon_2)\Big] \cap \Big[0, 1\Big], & \text{if } x + y \in [0, 1] \\ \Big[1 - (\epsilon_1 + \epsilon_2), 1\Big] \cap \Big[0, 1\Big], & \text{if } x > 1 \\ \Big[0, \epsilon_1 + \epsilon_2\Big] \cap \Big[0, 1\Big], & \text{if } x < 0 \end{cases}$$

which establishes that $\Big(\mathbf{x} + \mathbf{y}\Big)^{\mathbf{I,A}} = \Big((x+y) \pm (\epsilon_1 + \epsilon_2)\Big)^{\mathbf{I,A}}$. This implies that whenever $\mathbf{x} = x \pm \epsilon_1$ and $\mathbf{y} = y \pm \epsilon_2$ then $\sigma_+(\mathbf{x} + \mathbf{y}) = (x+y) \pm (\epsilon_1 + \epsilon_2)$ is a valid substitution.

σ_*: Assume that $\mathbf{k} = l \cdot \mathbf{x}$. The claim naturally stems out from the definition of operation (3.6). If $l \cdot x \in [0, 1]$, we have the following:

(a) $\sup\{\mathbf{k}\} = \min(1, l \cdot \sup\{\mathbf{x}\})$.

(b) $\inf\{\mathbf{k}\} = \max(0, l \cdot \inf\{\mathbf{x}\})$.

In situations where $l \cdot x > 1$, as defined in operation 3.6:

(a) $\sup\{\mathbf{k}\} = 1$

(b) $\inf\{\mathbf{k}\} = \max(0, 1 - l \cdot \epsilon)$

Conversely, if $l \cdot x < 0$, we have:

(a) $\sup\{\mathbf{k}\} = \min(1, l \cdot \epsilon)$

(b) $\inf\{\mathbf{k}\} = 0$

Summing up we get that:

$$(\mathbf{l} \cdot \mathbf{x})^{\mathbf{I,A}} = \begin{cases} \Big[l \cdot x - l \cdot \epsilon, l \cdot x + l \cdot \epsilon\Big] \cap \Big[0, 1\Big], & \text{if } l \cdot x \in [0, 1] \\ \Big[1 - l \cdot \epsilon, 1\Big] \cap \Big[0, 1\Big], & \text{if } l \cdot x > 1 \\ \Big[0, l \cdot \epsilon\Big] \cap \Big[0, 1\Big], & \text{if } l \cdot x < 0 \end{cases}$$

which establishes that if $\mathbf{x} = x \pm \epsilon$ and $\mathbf{l}$ represents a number greater than or equal to 1 then $\left(\mathbf{l} \cdot \mathbf{x}\right)^{\mathbf{I,A}} = \left(\mathbf{l} \cdot x \pm \mathbf{l} \cdot \epsilon\right)^{\mathbf{I,A}}$. This implies that $\sigma_*(\mathbf{l} \cdot \mathbf{x}) = \mathbf{l} \cdot x \pm \mathbf{l} \cdot \epsilon$ is a valid substitution.

□

The second Lemma which we make use in the subsequent proofs makes use of the following definition.

Definition. *Let $\boldsymbol{x}$ and $\boldsymbol{y}$ be variables that represent intervals within $[0,1]$. We say that the formula $\boldsymbol{y} \preceq \boldsymbol{x}$ is valid if $(\boldsymbol{y}^{\boldsymbol{I,A}}, \boldsymbol{x}^{\boldsymbol{I,A}}) \in \preceq^{\boldsymbol{I}}$, for all assignments $\boldsymbol{A}$.*

Remember that the interpretation of $\preceq$ is given in relation 3.1. The above definition essentially suggests that the formula $\preceq$ is true whenever the relation 3.1 holds.

Lemma 2. *If variables x, y represent numbers in $\boldsymbol{D}$ with $x \geq y$ and $\boldsymbol{x} = x \pm \epsilon$ and $\boldsymbol{y} = y \pm \epsilon$ for $\epsilon > 0$, then $\boldsymbol{y} \preceq \boldsymbol{x}$ is valid.*

Proof. Given the assumptions, we will prove that the pair of values $(\mathbf{y}^{\mathbf{I,A}}, \mathbf{x}^{\mathbf{I,A}})$ are contained in the relation (3.1).

When $x \in [0,1]$ and $x \geq y$ it is clear that $y \leq 1$. This leads to the following comparisons:

1. $\sup\{\mathbf{x}\} = \min(1, x+\epsilon) \geq \min(1, y+\epsilon) = \sup\{\mathbf{y}\}$.

2. $\inf\{\mathbf{x}\} = \max(0, x-\epsilon) \geq \max(0, y-\epsilon) = \inf\{\mathbf{y}\}$.

Now, when $x > 1$ and $x \geq y$, we have:

1. $\sup\{\mathbf{x}\} = 1 \geq \sup\{\mathbf{y}\}$.

2. $\inf\{\mathbf{x}\} = \max(0, 1-\epsilon)$. If $y > 1$, then $\inf\{\mathbf{y}\} = \max(0, 1-\epsilon)$, while otherwise $\inf\{\mathbf{y}\} = \max(0, y-\epsilon)$. In both cases, $\inf\{\mathbf{x}\} \geq \inf\{\mathbf{y}\}$.

Finally, if $y \leq x < 0$, according to (3.2), we have $\sup\{\mathbf{x}\} = \sup\{\mathbf{y}\}$ and $\inf\{\mathbf{x}\} = \inf\{\mathbf{y}\}$. In all scenarios, it is evident that $\sup\{\mathbf{x}\} \geq \sup\{\mathbf{y}\}$ and $\inf\{\mathbf{x}\} \geq \inf\{\mathbf{y}\}$, which, by definition, implies that $(\mathbf{y}^{\mathbf{I,A}}, \mathbf{x}^{\mathbf{I,A}}) \in \preceq^{\mathbf{I}}$. □

Gate simulation. For each gate $g \in \{\mathrm{NOT}, \mathrm{OR}, \mathrm{PURIFY}\}$ we construct a financial network $\mathcal{F}_g$. For each input variable u and output variable v of gate g, the financial network $\mathcal{F}_g$ contains (among others) a corresponding bank b_u and a corresponding bank b_v. We refer to a bank b_u of a financial network $\mathcal{F}_g$ as an *input bank* if u is an input variable of g, and we refer to bank b_u as an *output bank* if u is an output variable of g. Two notable characteristics of all $\mathcal{F}_g$ networks are that all CDS debtor banks possess enough external assets so that their recovery rate is equal to 1 in any ϵ-*CRRV* and no bank has more than one outgoing arc.

We proceed by describing each of the financial networks and we analyse the recovery rate of the output banks under any ϵ-*CRRV*, given a value of the input bank. We denote r_i the recovery rate of Bank i and $\mathbf{r}_i$ the range of the recovery rate values under ϵ-*CRRV*.

It is significant to realise that the error parameter $\epsilon > 0$, defines a range of values within which the recovery rate values for banks fall under an ϵ-CRRV. Definition 7 combined with the fact that all recovery rate values are restricted within $[0, 1]$, implies that obtaining an ϵ-CRRV for $\epsilon > \frac{1}{2}$ is achieved by setting $r_i = \frac{1}{2}$ for all banks i. Consequently we consider parameter values ϵ that fall within $(0, \frac{1}{2})$.

$\boxed{\textbf{NOT-gate.}}$ We simulate the gate (NOT, u, w) with the financial network $\mathcal{F}_{\text{NOT}}$, defined in Figure 3.25. Here, the input bank is b_u and the output bank is b_w.

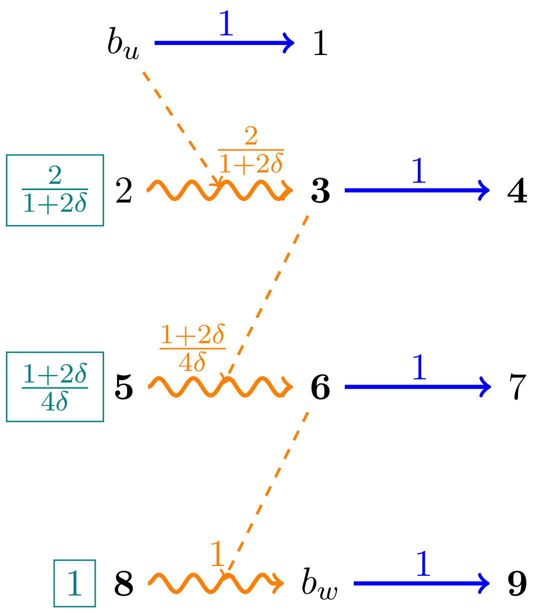


Figure 3.25: The financial network $\mathcal{F}_{\text{NOT}}$ that simulates a NOT-gate.

Assume that $r = (r_i)_{i \in N}$ is an ϵ-*CRRV* of $\mathcal{F}_{\text{NOT}}$. We will generate the atomic formulas of

$L(\mathrm{R}, \mathrm{F}, \mathrm{C})$ that, when interpreted through $\mathbf{I}$, defines the range of recovery rate values for the banks of $\mathcal{F}_{\mathrm{NOT}}$ under $r = (r_i)_{i \in N}$. Bank 3 receives a payment of $p_{2,3}(r) = \frac{2}{1+2\delta} \cdot [1 - r_{b_u}]$ from Bank 2 and holds a liability $l_{3,4}(r) = 1$ towards Bank 4. The recovery rate values for Bank 3 lie within the following range:

$$\mathrm{range}(r_3) = \Big[\min\Big(1, \frac{2}{1+2\delta} \cdot [1 - r_{b_u}]\Big) - \epsilon, \min\Big(1, \frac{2}{1+2\delta} \cdot [1 - r_{b_u}]\Big) + \epsilon\Big] \cap \Big[0, 1\Big] \tag{3.7}$$

Applying the proposed notation, the set of values mentioned above can be succinctly represented by the atomic formula $\mathbf{r}_3 = \frac{2}{1+2\delta} \cdot [1 - r_{b_u}] \pm \epsilon$. Notice that, as $r = (r_i)_{i \in N}$ is presumed to be an ϵ-*CRRV* of $\mathcal{F}_{\mathrm{NOT}}$, the payment received by Bank 6 from Bank 5 is contingent on the range of r_3. Specifically, each value ρ within $\mathrm{range}(r_3)$, establishes a corresponding payment, $p_{5,6}(r) = \frac{1+2\delta}{4\delta} \cdot [1 - \rho]$, from Bank 5 to Bank 6. Consequently, all possible recovery rate values for Bank 6 must lie within the range:

$$\mathrm{range}(r_6) = \bigcup_{\rho \in \mathrm{range}(r_3)} \Big[\min\Big(1, \frac{1+2\delta}{4\delta} \cdot [1 - \rho]\Big) - \epsilon, \min\Big(1, \frac{1+2\delta}{4\delta} \cdot [1 - \rho]\Big) + \epsilon\Big] \cap \Big[0, 1\Big] \tag{3.8}$$

which is represented by the atomic formula $\mathbf{r}_6 = \frac{1+2\delta}{4\delta} \cdot [1 - \mathbf{r}_3] \pm \epsilon$. Similarly, Bank b_w under the ϵ-*CRRV*, receives a payment of $p_{8,b_w}(r) = 1 - \rho$, where ρ is a value in $\mathrm{range}(r_6)$, thus we get

$$\mathrm{range}(r_{b_w}) = \bigcup_{\rho \in \mathrm{range}(r_6)} \Big[\min\Big(1, 1 - \rho\Big) - \epsilon, \min\Big(1, 1 - \rho\Big) + \epsilon\Big] \cap \Big[0, 1\Big] \tag{3.9}$$

which is generated by the atomic formula $\mathbf{r}_{b_w} = [1 - \mathbf{r}_6] \pm \epsilon$. To establish the connection between the range of recovery rate values of b_w w.r.t r_{b_u}, we apply a series of *valid substitutions* starting from the formula $\mathbf{r}_{b_w} = [1 - \mathbf{r}_6] \pm \epsilon$. We present the sequence of substitutions below.

$$\begin{aligned}
\mathbf{r}_{b_w} = [1 - \mathbf{r}_6] \pm \epsilon &= \left[1 - \left(\frac{1+2\delta}{4\delta} \cdot [1 - \mathbf{r}_3] \pm \epsilon\right)\right] \pm \epsilon \\
&= \left[1 - \left(\frac{1+2\delta}{4\delta} \cdot \underbrace{\left[1 - \frac{2}{1+2\delta} \cdot (1 - r_{b_u}) \pm \epsilon\right]}_{\mathbf{r}_3} \pm \epsilon\right)\right] \pm \epsilon \\
&\xrightarrow{\sigma_*(\frac{1+2\delta}{4\delta} \cdot \mathbf{r}_3)} \left[1 - \left(\underbrace{\left[\frac{1}{2\delta} \cdot r_{b_u} + \frac{-1+2\delta}{4\delta} \pm \frac{1+2\delta}{4\delta}\epsilon\right]}_{\mathbf{x}} \pm \epsilon\right)\right] \pm \epsilon \\
&\xrightarrow{\sigma_\pm(\mathbf{x} \pm \epsilon)} \left[1 - \underbrace{\left(\frac{1}{2\delta} \cdot r_{b_u} + \frac{-1+2\delta}{4\delta} \pm \frac{1+6\delta}{4\delta}\epsilon\right)}_{\mathbf{y}}\right] \pm \epsilon \\
&\xrightarrow{\sigma_{1-}(1-\mathbf{y})} \underbrace{\left[-\frac{1}{2\delta} \cdot r_{b_u} + \frac{1+2\delta}{4\delta} \pm \frac{1+6\delta}{4\delta}\epsilon\right]}_{\mathbf{z}} \pm \epsilon \\
&\xrightarrow{\sigma_\pm(\mathbf{z} \pm \epsilon)} -\frac{1}{2\delta} \cdot r_{b_u} + \frac{1+2\delta}{4\delta} \pm \frac{1+10\delta}{4\delta}\epsilon
\end{aligned}$$

Eventually the range for the recovery rate values of the output Bank b_w in any ϵ-*CRRV* is generated from an assignment $\mathbf{A}$ where $r_{b_u} \in [0, 1]$ and $\delta, \epsilon \in (0, \frac{1}{2})$ and by interpreting the following formula w.r.t. $\mathbf{I}$:

$$\mathbf{r}_{b_w} = -\frac{1}{2\delta} \cdot r_{b_u} + \frac{1+2\delta}{4\delta} \pm \frac{1+10\delta}{4\delta}\epsilon. \tag{3.10}$$

We proceed by computing the interval within which r_{b_w} belongs in any ϵ-*CRRV*, depending on the interval where the recovery rate of the input Bank r_{b_u} belongs according to the mapping m_δ. In all assignments $\mathbf{A}$ assumed in the proof of the claims, both δ and ϵ belong in $(0, \frac{1}{2})$.

Claim 1. *Consider the financial network $\mathcal{F}_{NOT}$ of Figure 3.25 and let $r = (r_i)_{i \in N}$ be an ϵ-CRRV:*

1. *If $r_{b_u} \in \left[0, \frac{1}{2} - \delta\right]$, then $r_{b_w} \in \left[1 - \frac{1+10\delta}{4\delta}\epsilon, 1\right]$.*
2. *If $r_{b_u} \in \left[\frac{1}{2} + \delta, 1\right]$, then $r_{b_w} \in \left[0, \frac{1+10\delta}{4\delta}\epsilon\right]$.*

Proof. We use Equation (3.10), that expresses range(r_{b_w}) in any ϵ-*CRRV* in terms of r_{b_u}.

1. In any assignment $\mathbf{A}$ of variables in 3.10 to values in $\mathbf{D}$ where $r_{b_u} \leq \frac{1}{2} - \delta$, from Lemma 2 and after calculations, it holds that $\left(1 \pm \frac{1+10\delta}{4\delta}\epsilon\right)^{\mathbf{I},\mathbf{A}} \preceq^{\mathbf{I}} \left(-\frac{1}{2\delta} \cdot r_{b_u} + \frac{1+2\delta}{4\delta} \pm \frac{1+10\delta}{4\delta}\epsilon\right)^{\mathbf{I},\mathbf{A}}$, namely $1 \pm \frac{1+10\delta}{4\delta}\epsilon \preceq \mathbf{r}_{b_w}$ is *valid.* Interpreting the last relation according to $\mathbf{I}$, from 3.2 and 3.1 we conclude that under any ϵ-*CRRV*, if $r_{b_u} \in \left[0, \frac{1}{2} - \delta\right]$ then $r_{b_w} \in \left[1 - \frac{1+10\delta}{4\delta}\epsilon, 1\right]$.
2. In any assignment $\mathbf{A}$ of variables in 3.10 to values in $\mathbf{D}$ where $r_{b_u} \geq \frac{1}{2} + \delta$, from Lemma 2 and after calculations, it holds that $\left(-\frac{1}{2\delta} \cdot r_{b_u} + \frac{1+2\delta}{4\delta} \pm \frac{1+10\delta}{4\delta}\epsilon\right)^{\mathbf{I},\mathbf{A}} \preceq^{\mathbf{I}} \left(0 \pm \frac{1+10\delta}{4\delta}\epsilon\right)^{\mathbf{I},\mathbf{A}}$,

namely $\mathbf{r}_{b_w} \preceq 0 \pm \frac{1+10\delta}{4\delta}\epsilon$ is *valid*. Applying $\mathbf{I}$ and from 3.2 and 3.1, under any ϵ-*CRRV* if $r_{b_u} \in \left[\frac{1}{2}+\delta, 1\right]$ then $r_{b_w} \in \left[0, \frac{1+10\delta}{4\delta}\epsilon\right]$ $\square$

$\boxed{\textbf{OR-gate.}}$ We simulate the gate (OR, u, v, w) with the financial network $\mathcal{F}_{\text{OR}}$ of Figure 3.26. The network consists of two input banks b_u and b_v corresponding to the input variables u, v respectively, and an output Bank b_w corresponding to the output variable w.

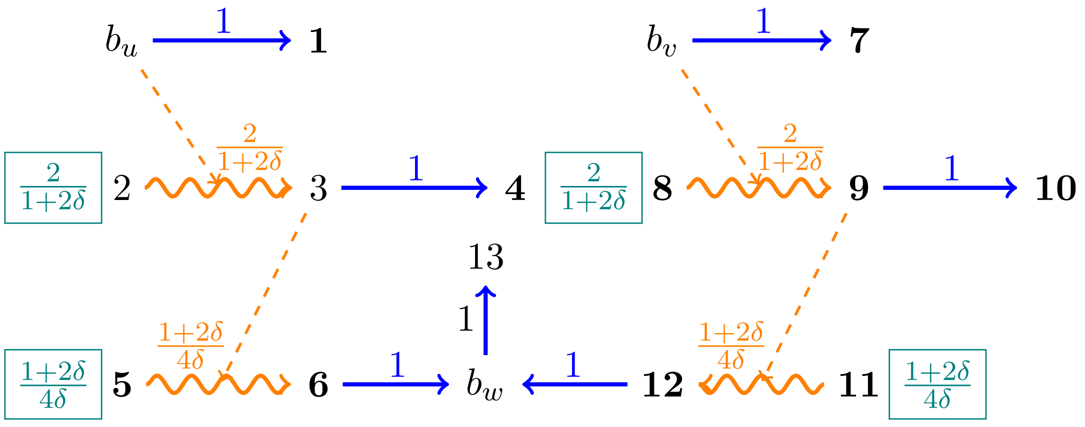


Figure 3.26: The financial network $\mathcal{F}_{\text{OR}}$ that simulates an OR-gate

Assume that $r = (r_i)_{i\in N}$ is an ϵ-*CRRV* of $\mathcal{F}_{\text{OR}}$ and let the recovery rates of the input banks b_u, b_v to be r_{b_u} and r_{b_v} respectively. The range of recovery rate values for Banks 3 and 9 in $\mathcal{F}_{\text{OR}}$ matches that of Bank 3 within the network $\mathcal{F}_{\text{NOT}}$. Symmetrically range(r_6) and range(r_{12}) in $\mathcal{F}_{\text{OR}}$ matches range(r_6) within the network $\mathcal{F}_{\text{NOT}}$. Both Bank 6 and Bank 12 submit to b_w payments of at most ρ_1, ρ_2 belonging to range(r_6) and range(r_{12}) respectively. Consequently the range of recovery rate values of the output Bank b_w is:

$$\text{range}(r_{b_w}) = \bigcup_{\substack{\rho_1\in\text{range}(r_6)\\ \rho_2\in\text{range}(r_{12})}} \left[\min\Big(1, \rho_1+\rho_2\Big) - \epsilon, \min\Big(1, \rho_1+\rho_2\Big) + \epsilon\right] \cap \Big[0, 1\Big] \tag{3.11}$$

which is generated from an assignment $\mathbf{A}$ where $r_{b_u} \in [0,1]$ and $\delta, \epsilon \in (0, \frac{1}{2})$ by applying $\mathbf{I}$ to the formula,

$$\mathbf{r}_{b_w} = (\mathbf{r}_6 + \mathbf{r}_{12}) \pm \epsilon \tag{3.12}$$

where for $\mathbf{r}_6$ (and similarly $\mathbf{r}_{12}$),

$$\mathbf{r}_6 = \frac{1+2\delta}{4\delta}\cdot[1-\mathbf{r}_3]\pm\epsilon = \Big(\frac{1+2\delta}{4\delta}\cdot\Big[1-\underbrace{\Big(\frac{2}{1+2\delta}\cdot[1-r_{b_u}]\pm\epsilon\Big)}_{\mathbf{r}_3}\Big]\Big)\pm\epsilon$$

$$\xrightarrow{\sigma_{1-(1-\mathbf{r}_3)}} \Big(\frac{1+2\delta}{4\delta}\cdot\underbrace{\Big[\frac{2}{1+2\delta}\cdot r_{b_u}+\frac{-1+2\delta}{1+2\delta}\pm\epsilon\Big]}_{\mathbf{x}}\Big)\pm\epsilon$$

$$\xrightarrow{\sigma_*(\frac{1+2\delta}{4\delta}\cdot\mathbf{x})} \underbrace{\Big(\frac{1}{2\delta}\cdot r_{b_u}+\frac{-1+2\delta}{4\delta}\pm\frac{1+2\delta}{4\delta}\epsilon\Big)}_{\mathbf{y}}\pm\epsilon$$

$$\xrightarrow{\sigma_\pm(\mathbf{y}\pm\epsilon)} \frac{1}{2\delta}\cdot r_{b_u}+\frac{-1+2\delta}{4\delta}\pm\frac{1+6\delta}{4\delta}\epsilon$$

We proceed by computing the interval where r_{b_w} belongs in any ϵ-*CRRV*, depending on the interval of the mapping m_δ where the recovery rate of the input Bank r_{b_u} belongs. In all assignments $\mathbf{A}$, both δ and ϵ belong in $(0,\frac{1}{2})$.

Claim 2. *Consider the financial network $\mathcal{F}_{OR}$ of Figure 3.26 and let r be an ϵ-CRRV.*

1. *If $r_{b_u}\in\Big[\frac{1}{2}+\delta,1\Big]$ or $r_{b_v}\in\Big[\frac{1}{2}+\delta,1\Big]$, then $r_{b_w}\in\Big[1-\frac{1+10\delta}{4\delta}\epsilon,1\Big]$.*
2. *If $r_{b_u},r_{b_v}\in\Big[0,\frac{1}{2}-\delta\Big]$, then $r_{b_w}\in\Big[0,\frac{1+8\delta}{2\delta}\epsilon\Big]$.*

Proof. We use (3.12), that expresses range(r_{b_w}) in any ϵ-*CRRV* in terms of r_{b_u},r_{b_v}.

1. In any assignment $\mathbf{A}$ of variables in 3.12 to values in $\mathbf{D}$ let w.l.o.g $r_{b_u}\geq\frac{1}{2}+\delta$. Since according to (3.12), $\mathbf{r}_{b_w}=(\mathbf{r}_6+\mathbf{r}_{12})\pm\epsilon$, it is easy to verify that $\Big((1\pm\frac{1+6\delta}{4\delta}\epsilon)\pm\epsilon\Big)^{\mathbf{I},\mathbf{A}}\preceq^{\mathbf{I}}\Big((\mathbf{r}_6+\mathbf{r}_{12})\pm\epsilon\Big)^{\mathbf{I},\mathbf{A}}$, where by applying substitution $\sigma_\pm$ we conclude that $1\pm\frac{1+10\delta}{4\delta}\epsilon\preceq\mathbf{r}_{b_w}$ is *valid.* This implies that under any ϵ-*CRRV*, if $r_{b_u}\in\Big[\frac{1}{2}+\delta,1\Big]$ then $r_{b_w}\in\Big[1-\frac{1+10\delta}{4\delta}\epsilon,1\Big]$.
2. In any assignment $\mathbf{A}$ of variables in (3.12) to values in $\mathbf{D}$ where both $r_{b_u},r_{b_v}\leq\frac{1}{2}-\delta$, it follows from the above analysis by similar reasoning that both $\mathbf{r}_6,\mathbf{r}_{12}\preceq 0\pm\frac{1+6\delta}{4\delta}\epsilon$ are valid. Consequently from (3.12) we can easily get that $\mathbf{r}_{b_w}\preceq(0\pm2\frac{1+6\delta}{4\delta}\epsilon)\pm\epsilon$, which by applying $\sigma_\pm$ implies that $\mathbf{r}_{b_w}\preceq 0\pm\frac{1+8\delta}{2\delta}\epsilon$ is *valid.* This establishes that under any ϵ-*CRRV*, if both $r_{b_u},r_{b_v}\in\Big[\frac{1}{2}+\delta,1\Big]$ then $r_{b_w}\in\Big[0,\frac{1+8\delta}{2\delta}\epsilon\Big]$. □

$\boxed{\textbf{PURIFY-gate.}}$ We simulate the gate (PURIFY, u,v,w) with the financial network of Figure 3.27. The network has one input bank denoted as b_u that corresponds to the input variable u and two output banks denoted as b_v,b_w corresponding to the output variables v,w respectively.

Assume that $r=(r)_{i\in N}$ is an ϵ-*CRRV* of $\mathcal{F}_{\text{PURIFY}}$ and let the recovery rate of the input Bank b_u be r_{b_u} and the recovery rates of the output banks b_u,b_w be r_{b_u},r_{b_w} respectively.

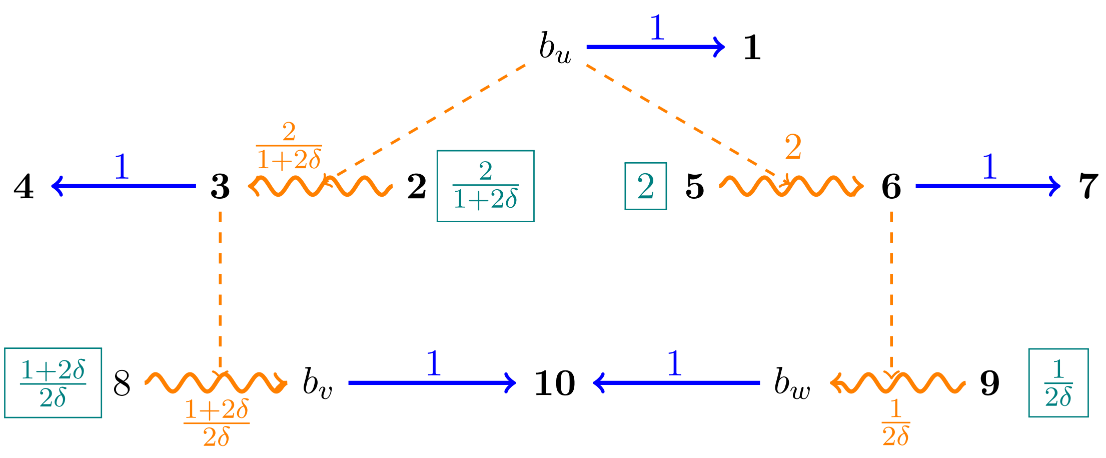


Figure 3.27: The financial network $\mathcal{F}_{\text{PURIFY}}$ that simulates a PURIFY-gate

- Let the *Left branch* be the subnetwork that spans nodes $\{b_u, 1, 2, 3, 4, 8, b_v, 10\}$.
- Let the *Right branch* be the subnetwork that spans nodes $\{b_u, 1, 5, 6, 7, 5, 9, b_w, 10\}$.

We proceed with analysing the intervals for the recovery rates r_{b_v}, r_{b_w} in any ϵ-*CRRV* as a function of r_{b_u}. Our analysis consists of two parts, one for each branch.

- ***Left* branch** : Bank 3 receives a payment of $\frac{2}{1+2\delta} \cdot [1 - r_{b_u}]$ from Bank 2 and holds a liability of 1 towards Bank 4. The recovery rate values for Bank 3 under $r = (r_i)_{i\in N}$ belong in the range:

$$\text{range}(r_3) = \Big[\min\Big(1, \frac{2}{1+2\delta} \cdot [1 - r_{b_u}]\Big) - \epsilon, \min\Big(1, \frac{2}{1+2\delta} \cdot [1 - r_{b_u}]\Big) + \epsilon\Big] \cap \Big[0, 1\Big] \quad (3.13)$$

which is represented by the atomic formula $\mathbf{r}_3 = \frac{2}{1+2\delta} \cdot [1 - r_3] \pm \epsilon$. Bank b_v receives a payment of $\frac{1+2\delta}{2\delta} \cdot [1 - \rho]$ from Bank 8, where $\rho \in \text{range}(r_3)$. The range of recovery rate values for Bank b_v is

$$\text{range}(r_{b_v}) = \bigcup_{\rho \in \text{range}(r_3)} \Big[\min\Big(1, \frac{1+2\delta}{2\delta} \cdot [1-\rho]\Big) - \epsilon, \min\Big(1, \frac{1+2\delta}{2\delta} \cdot [1-\rho]\Big) + \epsilon\Big] \cap \Big[0, 1\Big] \quad (3.14)$$

which is represented by the atomic formula $\mathbf{r}_{b_v} = \frac{1+2\delta}{2\delta} \cdot [1 - \mathbf{r}_3] \pm \epsilon$. The atomic formulas and the substitutions that generate the interval for the values of r_{b_v} under the ϵ-*CRRV* are listed below.

$$\mathbf{r}_{b_v} = \frac{1+2\delta}{2\delta} \cdot [1-\mathbf{r}_3] \pm \epsilon = \Big(\frac{1+2\delta}{2\delta} \cdot \Big[1 - \underbrace{\Big(\frac{2}{1+2\delta} \cdot [1-r_{b_u}] \pm \epsilon\Big)}_{\mathbf{r}_3}\Big]\Big) \pm \epsilon$$

$$\xrightarrow{\sigma_{1-(1-\mathbf{r}_3)}} \Big(\frac{1+2\delta}{2\delta} \cdot \Big[\underbrace{\frac{2}{1+2\delta} \cdot r_{b_u} + \frac{-1+2\delta}{1+2\delta} \pm \epsilon}_{\mathbf{x}}\Big]\Big) \pm \epsilon$$

$$\xrightarrow{\sigma_{*}(\frac{1+2\delta}{2\delta} \cdot \mathbf{x})} \underbrace{\Big(\frac{1}{\delta} \cdot r_{b_u} + \frac{-1+2\delta}{1+2\delta} \pm \frac{1+2\delta}{2\delta} \cdot \epsilon\Big)}_{\mathbf{y}} \pm \epsilon$$

$$\xrightarrow{\sigma_{\pm}(\mathbf{y} \pm \epsilon)} \frac{1}{\delta} \cdot r_{b_u} + \frac{-1+2\delta}{2\delta} \pm \frac{1+4\delta}{2\delta}\epsilon$$

From the above analysis, the range for the recovery rate values of the output Bank b_v in any ϵ-*CRRV* is generated from an assignment $\mathbf{A}$ where $r_{b_u} \in [0,1]$ and $\delta, \epsilon \in (0, \frac{1}{2})$ and the interpretation of the following formula according to $\mathbf{I}$:

$$\mathbf{r}_{b_v} = \frac{1}{\delta} \cdot r_{b_u} + \frac{-1+2\delta}{2\delta} \pm \frac{1+4\delta}{2\delta}\epsilon. \tag{3.15}$$

- ***Right* branch**: Bank 6 receives a payment of $2 \cdot [1 - r_{b_u}]$ from Bank 5 and since it holds a liability of 1 towards Bank 7 the range of recovery rate values is given by the following expression:

$$\text{range}(r_6) = \Big[\min\Big(1, 2 \cdot [1 - r_{b_u}]\Big) - \epsilon, \min\Big(1, 2 \cdot [1 - r_{b_u}]\Big) + \epsilon\Big] \cap \Big[0, 1\Big] \tag{3.16}$$

which is represented by the atomic formula $\mathbf{r}_6 = 2 \cdot [1 - r_3] \pm \epsilon$. Bank b_w receives a payment of $\frac{1}{2\delta} \cdot [1 - \rho]$ from Bank 9, where $\rho \in \text{range}(r_6)$. Consequently the range of recovery rate values for Bank b_w under the ϵ-*CRRV* $r = (r_i)_{i \in N}$ is:

$$\text{range}(r_{b_w}) = \bigcup_{\rho \in \text{range}(r_6)} \Big[\min\Big(1, \frac{1}{2\delta} \cdot [1-\rho]\Big) - \epsilon, \min\Big(1, \frac{1}{2\delta} \cdot [1-\rho]\Big) + \epsilon\Big] \cap \Big[0, 1\Big] \tag{3.17}$$

which is represented by the atomic formula $\mathbf{r}_{b_w} = \frac{1}{2\delta} \cdot [1 - \mathbf{r}_6] \pm \epsilon$. The atomic formulas and the substitutions that define the range where the values for r_{b_w} belong under the ϵ-*CRRV* are listed below.

$$\mathbf{r}_{b_w} = \frac{1}{2\delta}\cdot[1-\mathbf{r}_6] \pm \epsilon = \left(\frac{1}{2\delta}\cdot\left[1-\underbrace{\left(2\cdot[1-r_{b_u}]\pm\epsilon\right)}_{\mathbf{r}_6}\right]\right)\pm\epsilon$$

$$\xrightarrow{\sigma_{1-}(1-\mathbf{r}_6)} \left(\frac{1}{2\delta}\cdot\left[\underbrace{-1+2\cdot r_{b_u}\pm\epsilon}_{\mathbf{x}}\right]\right)\pm\epsilon$$

$$\xrightarrow{\sigma_*(\frac{1}{2\delta}\cdot\mathbf{x})} \underbrace{\left(\frac{1}{\delta}\cdot r_{b_u}-\frac{1}{2\delta}\pm\frac{1}{2\delta}\epsilon\right)}_{\mathbf{y}}\pm\epsilon$$

$$\xrightarrow{\sigma_\pm(\mathbf{y}\pm\epsilon)} \frac{1}{\delta}\cdot r_{b_u}-\frac{1}{2\delta}\pm\frac{1+2\delta}{2\delta}\epsilon$$

The range for the recovery rate values of the output Bank b_w in any ϵ-*CRRV* follows from an assignment $\mathbf{A}$ where $r_{b_u}\in[0,1]$ and $\delta,\epsilon\in(0,\frac{1}{2})$ and the interpretation of the following formula according to $\mathbf{I}$:

$$\mathbf{r}_{b_w} = \frac{1}{\delta}\cdot r_{b_u}-\frac{1}{2\delta}\pm\frac{1+2\delta}{2\delta}\epsilon. \tag{3.18}$$

We compute the range for r_{b_v}, r_{b_w} with respect to r_{b_u} are as follows:

Claim 3. *Consider the financial network $\mathcal{F}_{PURIFY}$ of Figure 3.27 and let r be an ϵ-CRRV.*

1. *If $r_{b_u}\in\left[0,\frac{1}{2}-\delta\right]$, then $r_{b_v}\in\left[0,\frac{1+4\delta}{2\delta}\epsilon\right]$ and $r_{b_w}\in\left[0,\frac{1+2\delta}{2\delta}\epsilon\right]$.*
2. *If $r_{b_u}\in\left[\frac{1}{2}+\delta,1\right]$, then $r_{b_v}\in\left[1-\frac{1+4\delta}{2\delta}\epsilon,1\right]$ and $r_{b_w}\in\left[1-\frac{1+2\delta}{2\delta}\epsilon,1\right]$.*
3. *If $r_{b_u}\in\left(\frac{1}{2}-\delta,\frac{1}{2}+\delta\right)$, then $r_{b_v}\in\left[1-\frac{1+4\delta}{2\delta}\epsilon,1\right]$ or $r_{b_w}\in\left[0,\frac{1+2\delta}{2\delta}\epsilon\right]$.*

Proof. Similar to the previous claims, the proof follows from a case analysis of (3.15) and (3.18) upon values of r_{b_u}.

1. In any assignment $\mathbf{A}$ of variables in (3.15) to values in $\mathbf{D}$ where $r_{b_u}\le\frac{1}{2}-\delta$, from Lemma 2 and after calculations we get that in any ϵ-*CRRV*, $\left(\frac{1}{\delta}\cdot r_{b_u}+\frac{-1+2\delta}{2\delta}\pm\frac{1+4\delta}{2\delta}\epsilon\right)^{\mathbf{I},\mathbf{A}}\preceq^{\mathbf{I}}\left(0\pm\frac{1+4\delta}{2\delta}\epsilon\right)^{\mathbf{I},\mathbf{A}}$, namely $\mathbf{r}_{b_v}\preceq 0\pm\frac{1+4\delta}{2\delta}\epsilon$ is *valid*. This establishes that if $r_{b_u}\in\left[0,\frac{1}{2}-\delta\right]$ then $r_{b_v}\in\left[0,\frac{1+4\delta}{2\delta}\epsilon\right]$. Similarly combining (3.18) with Lemma 2 and the fact that $r_{b_u}\le\frac{1}{2}$, we conclude that $\mathbf{r}_{b_w}\preceq 0\pm\frac{1+2\delta}{2\delta}\epsilon$ is *valid*. This establishes that whenever $r_{b_u}\in\left[0,\frac{1}{2}-\delta\right]$ then $r_{b_w}\in\left[0,\frac{1+2\delta}{2\delta}\epsilon\right]$.
2. In any assignment $\mathbf{A}$ of variables where $r_{b_u}\ge\frac{1}{2}+\delta$, trivially $r_{b_u}\ge\frac{1}{2}$ thus considering (3.15) and Lemma 2 and after calculations it holds that $\left(1\pm\frac{1+4\delta}{2\delta}\epsilon\right)^{\mathbf{I},\mathbf{A}}\preceq^{\mathbf{I}}\left(\frac{1}{\delta}r_{b_u}+\frac{-1+2\delta}{2\delta}\pm\frac{1+4\delta}{2\delta}\epsilon\right)^{\mathbf{I},\mathbf{A}}$, namely $1\pm\frac{1+4\delta}{2\delta}\epsilon\preceq\mathbf{r}_{b_v}$ is *valid*. This establishes that whenever $r_{b_u}\in\left[\frac{1}{2}+\delta,1\right]$ then $r_{b_v}\in\left[1-\frac{1+4\delta}{2\delta}\epsilon,1\right]$. Similarly combining the initial assumption with (3.18) and Lemma

2, after calculations it holds that $1 \pm \frac{1+2\delta}{2\delta}\epsilon \preceq \mathbf{r}_{b_w}$ is *valid.* This establishes that whenever $r_{b_u} \in \left[\frac{1}{2}+\delta, 1\right]$, then $r_{b_w} \in [1-\frac{1+2\delta}{2\delta}\epsilon, 1]$

3. Finally for assignments $\mathbf{A}$ where $r_{b_u} \in \left(\frac{1}{2}-\delta, \frac{1}{2}+\delta\right)$, if $r_{b_u} \leq \frac{1}{2}$ then Statement 1 already establishes that $\mathbf{r}_{b_w} \preceq 0 \pm \frac{1+2\delta}{2\delta}\epsilon$ is *valid*, while if $r_{b_u} \geq \frac{1}{2}$, then Statement 2 establishes that $1 \pm \frac{1+4\delta}{2\delta}\epsilon \preceq \mathbf{r}_{b_v}$ is *valid.* Consequently whenever $r_{b_u} \in \left(\frac{1}{2}-\delta, \frac{1}{2}+\delta\right)$ either $r_{b_u} \in \left[1-\frac{1+4\delta}{2\delta}\epsilon, 1\right]$ or $r_{b_w} \in \left[0, \frac{1+2\delta}{2\delta}\epsilon\right]$. □

Specifying δ. Claims 1, 2 and 3 establish for the networks $\mathcal{F}_{\text{NOT}}, \mathcal{F}_{\text{OR}}, \mathcal{F}_{\text{PURIFY}}$ respectively the intervals where the recovery rates of the output banks lie as a function of the recovery rates of the input banks, under any ϵ-*CRRV*. All these intervals either span right from 0 or left from 1. We want to find an assignment $\mathbf{A}$ for the value of δ such that under m_δ, the three networks $\mathcal{F}_{\text{NOT}}$, $\mathcal{F}_{\text{OR}}$, and $\mathcal{F}_{\text{PURIFY}}$ *simulate* the gates NOT, OR, and PURIFY respectively, where the notion of simulation is defined in the natural way: for any fixed input bank's recovery rate of the financial network, any ϵ-*CRRV* of the financial network must be mapped back by m_δ to a satisfying assignment on the inputs and outputs of the respective gate. Consequently, to determine the values of δ such that the $\mathcal{F}_{\text{NOT}}, \mathcal{F}_{\text{OR}}, \mathcal{F}_{\text{PURIFY}}$ networks correctly simulate the NOT-, OR-, PURIFY-gates respectively, we must set the parameter δ such that the recovery rates of the output banks of the three networks are contained in the appropriate intervals of m_δ. For example, considering assignments $\mathbf{A}$ where the values for δ satisfy $\left[0, \frac{1}{2}-\delta\right] \subset \left[0, \frac{1+2\delta}{2\delta}\epsilon\right]$, allows numbers $q \in \left(\frac{1}{2}-\delta, \frac{1+2\delta}{2\delta}\epsilon\right]$ to be mapped according to m_δ to a different value than 0, meaning that $\mathcal{F}_{\text{PURIFY}}$ would not correctly simulate the PURIFY-gate (Claim 3), because for input an encoded value 0, a satisfying assignment for the PURIFY-gate must output encoded values 0 for both output variables. Scanning through all intervals, it turns out the biggest interval is $\left[0, \frac{1+8\delta}{2\delta}\epsilon\right]$ from Statement 2 of Claim 2. Eventually to find a feasible choice of δ, we must consider all assignments $\mathbf{A}$ of pairs of values for the parameters δ, ϵ where

$$\frac{1+8\delta}{2\delta}\epsilon = \frac{1}{2} - \delta. \tag{3.19}$$

As we will show next for all assignments $\mathbf{A}$ in which a pair (δ, ϵ) satisfies (3.19), it holds that all three financial networks correctly simulate their respective gate under m_δ. Let (δ, ϵ) be any pair that satisfies (3.19).

- $\boxed{\textbf{NOT-gate}}$ Consider the network $\mathcal{F}_{\text{NOT}}$ of Figure 3.25 and let r be an ϵ-*CRRV*. From

Statement 1 of Claim 1 whenever $r_{b_u} \in \left[0, \frac{1}{2} - \delta\right]$ it holds that $r_{b_w} \in \left[1 - \frac{1+10\delta}{4\delta}\epsilon, 1\right]$ and since $\left[1 - \frac{1+10\delta}{4\delta}\epsilon, 1\right] \subset \left[1 - \frac{1+8\delta}{2\delta}\epsilon, 1\right]$ trivially $r_{b_w} \in \left[1 - \frac{1+8\delta}{2\delta}\epsilon, 1\right]$. From Statement 2 of the same Claim, whenever $r_{b_u} \in \left[\frac{1}{2} + \delta, 1\right]$, then $r_{b_w} \in \left[0, \frac{1+10\delta}{4\delta}\epsilon\right]$ and since $\left[0, \frac{1+10\delta}{4\delta}\epsilon\right] \subset \left[0, \frac{1+8\delta}{2\delta}\epsilon\right]$ trivially $r_{b_w} \in \left[0, \frac{1+8\delta}{2\delta}\epsilon\right]$. Consequently the mapping m_δ on any ϵ-*CRRV* of $\mathcal{F}_{\text{NOT}}$ generates an assignment x for which it holds that:

1. $\text{x}[u] = 0 \rightarrow \text{x}[w] = 1$
2. $\text{x}[u] = 1 \rightarrow \text{x}[w] = 0$

As satisfying assignments are indifferent with respect to $\perp$ values, the above argument suffices to establish that $\mathcal{F}_{\text{NOT}}$ correctly simulates a NOT-gate under m_δ.

- $\boxed{\textbf{OR-gate}}$ Consider the network $\mathcal{F}_{\text{OR}}$ of Figure 3.26 and let r be an ϵ-*CRRV*. Similarly from Statement 1 of Claim 2, the recovery rate for the output bank b_w lies in $\left[1 - \frac{1+8\delta}{2\delta}\epsilon, 1\right]$ if one of r_{b_u}, r_{b_v} lies in $\left[\frac{1}{2} + \delta, 1\right]$ and from Statement 2, $r_{b_w} \in \left[0, \frac{1+8\delta}{2\delta}\epsilon\right]$ if both $r_{b_u}, r_{b_v} \in \left[0, \frac{1}{2} - \delta\right]$. As a result the mapping m_δ on any ϵ-*CRRV* of $\mathcal{F}_{\text{OR}}$ generates an assignment x for which

  1. if $\text{x}[u] = \text{x}[v] = 0 \rightarrow \text{x}[w] = 0$
  2. if $\text{x}[u] = 1$ or $\text{x}[v] = 1 \rightarrow \text{x}[w] = 1$

  Checking the satisfying conditions in Section 3.1, we conclude that the above argument suffices to establish that $\mathcal{F}_{\text{OR}}$ correctly simulates the OR-gate under m_δ.

- $\boxed{\textbf{PURIFY-gate}}$ Consider the network of Figure 3.27, and let r be an ϵ-*CRRV*. From Statement 1 of Claim 3 whenever $r_{b_u} \in \left[0, \frac{1}{2} - \delta\right]$ it holds that $r_{b_v} \in \left[0, \frac{1+4\delta}{2\delta}\epsilon\right] \subset \left[0, \frac{1+8\delta}{2\delta}\epsilon\right]$ and $r_{b_w} \in \left[0, \frac{1+2\delta}{2\delta}\epsilon\right] \subset \left[0, \frac{1+8\delta}{2\delta}\epsilon\right]$. That means that the mapping m_δ generates an assignment x such that if $\text{x}[u] = 0$ then $\text{x}[v] = \text{x}[w] = 0$. From Statement 2, whenever $r_{b_u} \in \left[\frac{1}{2} + \delta, 1\right]$ then $r_{b_v} \in \left[1 - \frac{1+4\delta}{2\delta}\epsilon\right] \subset \left[1 - \frac{1+8\delta}{2\delta}\epsilon, 1\right]$ and $r_{b_w} \in \left[1 - \frac{1+2\delta}{2\delta}\epsilon, 1\right] \subset \left[1 - \frac{1+8\delta}{2\delta}\epsilon, 1\right]$, meaning m_δ generates an assignment x where if $\text{x}[u] = 1$ then $\text{x}[v] = \text{x}[w] = 1$. Finally from Statement 3 and using similar arguments, it is not hard to see that m_δ generates an assignment x such that whenever $\text{x}[u] = \perp$ then $\text{x}[v] = 1$ or $\text{x}[w] = 0$. So to conclude, from any ϵ-*CRRV* of $\mathcal{F}_{\text{PURIFY}}$, m_δ generates an assignment x where:

  1. $\text{x}[u] = 0 \rightarrow \text{x}[v] = \text{x}[w] = 0$.
  2. $\text{x}[u] = 1 \rightarrow \text{x}[v] = \text{x}[w] = 1$.
  3. $\text{x}[u] = \perp \rightarrow \text{x}[v] \in \{0, 1\}$ or $\text{x}[w] \in \{0, 1\}$.

These are exactly the conditions that hold in a satisfying assignment x of a PURIFY-gate.

Given an instance I of **PURE-CIRCUIT** that operates on gates NOT, OR and PURIFY, we construct a financial network $\mathcal{F}_I$ consisting of debt and CDS contracts and show that any ϵ-*CRRV* of $\mathcal{F}_I$ is mapped back to a satisfying assignment x of values to the variables of the original **PURE-CIRCUIT** instance I, by applying m_δ, such that δ and ϵ satisfies (3.19).

The reduction. Let $I = (V, G)$ be a **PURE-CIRCUIT** instance. The construction of the financial network $\mathcal{F}_I$ proceeds as follows: For each gate $g = (\text{NOT}, u, w) \in G$, we construct a copy of the financial network $\mathcal{F}_{\text{NOT}}$ of Figure 3.25. For each gate $g = (\text{OR}, u, v, w) \in G$, we construct a copy of the financial network $\mathcal{F}_{\text{OR}}$ of Figure 3.26. For each gate $g = (\text{PURIFY}, u, v, w) \in G$, we construct a copy of the financial network $\mathcal{F}_{\text{PURIFY}}$ of Figure 3.27.

The interconnection of the gates in a **PURE-CIRCUIT** instance rises from the fact that the gates may share variables. Remember though that by the definition of the **PURE-CIRCUIT** problem (see Definition 9), no variable can be the output of more than one gate, whereas a variable can be input to many gates. Consequently after the execution of the above steps we might end up with more than one bank to represent the same variable. Next we will show how to deal with these situations.

Without loss of generality, let χ be the output variable of gate g and the input variable of another gate g'. After replacing g and g' with their respective financial networks $\mathcal{F}_g$ and $\mathcal{F}_{g'}$, we are left with two banks representing variable χ, both denoted as b_χ. To connect $\mathcal{F}_g$ and $\mathcal{F}_{g'}$ and represent the interconnection between g and g' due to their shared variable χ, we merge the two b_χ banks into one bank, keeping the same notation b_χ. It's worth noting that, as constructed, every input and output bank holds one outgoing liability of notional 1. Therefore, both b_χ banks, which represented the common variable χ prior to merging, held one outgoing liability of notional 1. In order to maintain the recovery rate analysis presented earlier, it is important for the newly merged b_χ bank to retain only one outgoing liability of notional 1. To achieve this, we eliminate one of the two outgoing liabilities after merging.

We have shown how to generate a financial network $\mathcal{F}_I$ from a **PURE-CIRCUIT** instance I and presented a mapping m_δ to map back ϵ-*CRRV*es of $\mathcal{F}_I$ to satisfying assignments for I, where (δ, ϵ) is a pair that satisfies (3.19). The next theorem establishes our main result.

Theorem 6. ϵ-**CDS-CLEARING** *is PPAD-hard for* $\epsilon \leq \frac{3-\sqrt{5}}{16}$.

Proof. We reduce from **PURE-CIRCUIT** to ϵ-**CDS-CLEARING**. As established, all values of δ, ϵ that satisfy (3.19) generate mappings under which each $\mathcal{F}_g$ network correctly simulates gate $g \in \{\text{NOT}, \text{OR}, \text{PURIFY}\}$. To determine the parameter δ that generates the mapping that maximises the inapproximability parameter ϵ, we rewrite (3.19) as $\epsilon = \phi(\delta) = \frac{\delta\cdot(1-2\delta)}{1+8\delta}$. To maximise the inapproximability parameter ϵ, we have to find the maximum of the function $\phi(\delta)$ w.r.t δ, i.e, the value $\delta^* = \arg\max \frac{\delta\cdot(1-2\delta)}{1+8\delta}$, where $\delta^* \in (0, 1/2)$. This means we must compute the critical points of function $\phi(\delta)$, which in turn means finding the values δ for which $\phi'(\delta) = 0$, where $\phi'(\delta)$ is the first derivative of $\phi(\delta) = \frac{\delta\cdot(1-2\delta)}{1+8\delta}$. The first derivative of $\phi(\delta)$ is computed as $\frac{d}{d\delta}\big(\phi(\delta)\big) = \frac{(1+8\delta)\cdot(1-4\delta)-\delta\cdot(1-2\delta)\cdot 8}{(1+8\delta)^2}$. In order for $\frac{d}{d\delta}\big(\phi(\delta)\big) = 0$, we must solve $16\cdot\delta^2 + 4\cdot\delta - 1 = 0$. After solving this binomial we get that the critical point for function $\phi(\delta)$ is $\delta^* = \frac{\sqrt{5}-1}{8}$. Finally the maximum value for the inapproximability parameter ϵ is computed to be $\phi(\delta^*) = \frac{3-\sqrt{5}}{16} \approx 0.048$. Thus given an instance $I = (V, G)$ of **PURE-CIRCUIT**, we construct a network $\mathcal{F}_I$ of **CDS-CLEARING** as described above, where we set $\delta = \delta^*$. We can then map, in polynomial time ϵ-CRRVes of $\mathcal{F}_I$ to a satisfying assignment for I through mapping m_{δ^*}, i.e.:

(i.) If $r_{b_u} \leq \frac{1}{2} - \delta^*$ then $\text{x}[u] = 0$;

(ii.) if $r_{b_u} \geq \frac{1}{2} + \delta^*$ then $\text{x}[u] = 1$

(iii.) else $\text{x}[u] = \perp$.

This establishes the claim. □

Extension to other payment schemes. In the network that we constructed to establish the above theorem, each bank can be possibly a debtor to only one contract. This allows us to apply the presented analysis to a broader class of payments, including those satisfying *Limited Liability* and *Absolute Priority* (see Section 2.1.2).

Corollary 1. ϵ-**CDS-CLEARING** *is PPAD-hard for* $\epsilon \leq \frac{3-\sqrt{5}}{16}$ *under any payment scheme that satisfies the Limited Liability and Absolute Priority conditions.*

3.2.4 Central CDS debtors & Dedicated CDS debtors

In this final section, we evaluate the effectiveness of policies introduced in the wake of the 2008 economic crisis for reducing systemic risk in financial markets. We introduce **Central CDS debtors** (CCDs), a construct reminiscent of a *central clearing counterparty* (CCP).

Definition 10 (Central CDS debtor). *A financial network $\mathcal{F} = (N, e, c)$ satisfies the central CDS debtor property if the following conditions hold,*

1. $\exists i' \in N : \forall i, j, R \in N : c^R_{i,j} \neq 0 \to i = i'$, *i.e, all CDS contracts share the same debtor bank. We refer to this debtor as central CDS debtor and denote it by* $\mathcal{CCD}$.
2. $\forall i \in N : c_{\mathcal{CCD},i} = 0$, *i.e, there are no debt contracts where* $\mathcal{CCD}$ *is the debtor.*
3. $e_{\mathcal{CCD}} \geq \sum_{j,R \in N} c^R_{\mathcal{CCD},j}$, *i.e,* $\mathcal{CCD}$ *possess enough assets to fully pay off any of its potential liabilities.*

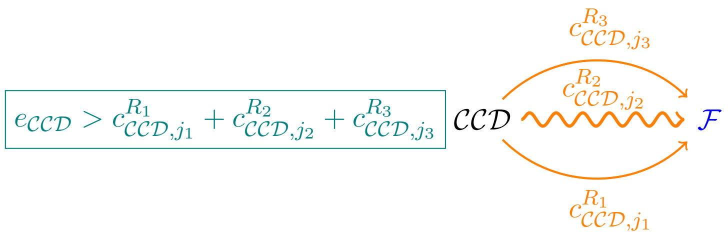

$\boxed{e_{\mathcal{CCD}} > c^{R_1}_{\mathcal{CCD},j_1} + c^{R_2}_{\mathcal{CCD},j_2} + c^{R_3}_{\mathcal{CCD},j_3}}$
$\mathcal{CCD}$
$\mathcal{F}$
$c^{R_3}_{\mathcal{CCD},j_3}$
$c^{R_2}_{\mathcal{CCD},j_2}$
$c^{R_1}_{\mathcal{CCD},j_1}$

Figure 3.28: A central CDS debtor $\mathcal{CCD}$ with three CDS contracts. The label $\mathcal{F}$ is used to indicate that the financial network $\mathcal{F}$ contains only simple debt contracts.

Essentially in the *Central CDS debtor* property, a single bank, assumed to have sufficient assets, is responsible for clearing the network's debt that is generated from the activation of credit default swaps. Despite this being a simple contract graph topology for a financial network with debt and CDS contracts, Theorem 6 still applies.

Corollary 2. ϵ**-CDS-CLEARING** *restricted to instances that satisfy the central CDS debtor property is PPAD-hard for* $\epsilon \leq \frac{3-\sqrt{5}}{16}$.

Proof. It can be verified that in the financial networks constructed in the reduction of Theorem 6, each CDS debtor satisfies Conditions 2 and 3 of the above Definition 10. We modify the reduction by (as a final step in the construction of the network) merging all CDS debtors into a single debtor node $\mathcal{CCD}$, whose external assets equals the sum of the external assets of merged debtors. The resulting network satisfies Definition 10. □

We show a similar result for another natural restriction on CDS debtors defined as *dedicated CDS debtors.* We will elaborate more on this class of instances in Section 5.2.

Definition 11 (**Dedicated CDS debtor**). *A financial network $\mathcal{F} = (N, e, c)$ satisfies the* dedicated CDS debtor property *iff for every CDS debtor bank $i \in N$,*

1. *$\forall j \in N : c_{i,j} = 0$, i.e, there are no debt contracts where i is the debtor.*
2. *$\exists R \in N : \forall j, k \in N : c^k_{i,j} \neq 0 \to k = R$, i.e, all CDSes for which i is the debtor share the same reference bank.*

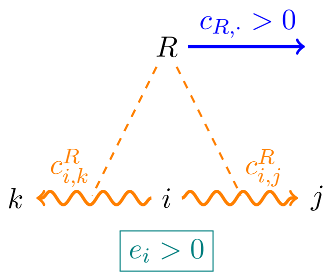


Figure 3.29: A dedicated CDS debtor i to R with two CDS contracts

Notice that in all financial networks illustrated in Figures 3.25, 3.26 and 3.27 the *dedicated CDS debtor* property is satisfied. From this observation we directly conclude the following statement

Corollary 3. *ϵ-**CDS-CLEARING** restricted to instances that satisfy the* dedicated CDS debtor *property is PPAD-hard for $\epsilon < \frac{3-\sqrt{5}}{16}$.*

An alternative proof of PPAD-hardness for instances that satisfy the *dedicated CDS debtor* property is given in Lemma 5.

Chapter 4

Algorithmic Approaches

Overview

This chapter addresses the computation of exact solutions for certain simpler network structures. We present an optimisation-based framework for computing clearing vectors, within the context of central CDS debtors (CCDs) and the general case. In Section 4.1.1 we present a Mixed-Binary Linear Program (MBLP) for systems with the central CDS debtor property. Within this program there is a combination of real-valued variables for the recovery rates and binary decision variables tailored to indicate a bank's solvency status. The feasible solutions of this program correspond to the clearing recovery rate vectors of the network. This framework is highly adaptable for optimising any linear objective function of interest related to the clearing vector. An immediate implication is an exponential-time algorithm for computing clearing recovery rate vectors when CCDs are mandated. Adapting the constraints to general instances, generates a Mixed-Binary Nonlinear Program (MBNLP) presented in Section 4.1.2 for **CDS-CLEARING**.

Motivated by EU regulations on the 2008 economic crisis, in Section 4.2 we evaluate the special types of credit default swaps called covered CDS introduced in [SSB20]. We establish that **CDS-CLEARING** restricted to instances with only covered CDSes and a central CDS debtor admit a polynomial time algorithm.

4.1 Optimisation-Based Computation of Clearing Vectors

4.1.1 Mixed-Binary Linear Program for Central CDS debtors

Assume a financial network $\mathcal{F} = (N \cup \{\mathcal{CCD}\}, e, c)$ that satisfies the central CDS debtor property. W.l.o.g assume that each bank in the network apart from the $\mathcal{CCD}$ has at least one positive liability[1]. We will formulate a Mixed-Binary Linear Program w.r.t $\mathcal{F}$ denoted as MBLP($\mathcal{F}$), whose feasible solution set is essentially Sol($\mathcal{F}$), i.e, all clearing recovery rate vectors of $\mathcal{F}$. The constraints of the program are formulated w.r.t the variable $z = (r_i, y_i)_{i \in N}$, where $r = (r_i)_{i \in N}$ represents a recovery rate vector with $r_i \in [0, 1]$ and $y = (y_i)_{i \in N}$ is a vector of binary decision variables one for each bank $i \in N$ with the following desired indication; For a given recovery rate vector r,

$$y_i = \begin{cases} 0, & \text{if } a_i(r) > l_i \\ 1, & \text{if } a_i(r) < l_i \\ 0 \text{ or } 1, & \text{if } a_i(r) = l_i. \end{cases} \tag{4.1}$$

Recall that by Definition 2.6, under any clearing vector $r = (r_i)_{i \in N}$, for each bank $i \in N$ it must hold that $r_i = \min(1, a_i(r)/l_i)$, where for the central CDS debtor framework,

$$a_i(r) = \left[e_i + \sum_{j \in N} r_j \cdot c_{j,i} + \sum_{k \in N} (1 - r_k) \cdot c^k_{\mathcal{CCD},i} \right] \tag{4.2}$$

$$l_i = \sum_{j \in N} c_{i,j}. \tag{4.3}$$

Remember that within networks adhering to the central CDS debtor framework, the liabilities of the banks remain fixed, regardless of the recovery rate vector.

The decision vector y serves the purpose of determining for each bank $i \in N$, which argument within the min operator is the correct choice, whenever the differentiation among the arguments affects the insolvency of bank i. Based on such a choice indicated by y, for each bank $i \in N$ the program should compute a feasible solution, if a recovery rate vector r that justifies the indicated choice exists. Essentially, y eliminates the need for the min operator in the fixed-point condition.

[1]We make this assumption to avoid technicalities for the sake of presentation. The program easily adapts to sink nodes.

Furthermore, we associate each bank $i \in N$, with the following constant

$$\mathrm{B}_i = \frac{1}{\sum_{j \in N} c_{i,j}} \cdot \left[e_i + \sum_{j \in N} c_{j,i} + \sum_{k \in N} c^k_{\mathcal{CCD},i} \right] + 1. \tag{4.4}$$

Clearly for each bank $i \in N$ and any recovery rate vector $r = (r_i)_{i \in N}$, $\min(1, a_i(r)/l_i) \leq \mathrm{B}_i$.

We proceed by presenting a Mixed-Binary Linear Program for computing exact clearing recovery rate vectors for a given financial network $\mathcal{F} = (N \cup \{\mathcal{CCD}\}, e, c)$ that satisfies the central CDS debtor property. At this stage we exclusively address the feasibility problem of just computing any clearing vector. Therefore, the program is presented without specifying an objective function, rather we assume any linear function $f(r)$, w.r.t a recovery rate vector $r = (r_i)_{i \in N}$.

$$\begin{aligned}
\text{Maximise / Minimise:} \quad & \text{Linear function } f(r) & \\
\text{Subject to:} \quad & \text{For each } i \in N & \\
& r_i \geq \frac{a_i(r)}{l_i} - \mathrm{B}_i \cdot (1 - y_i) & \text{(Constraint 1)} \\
& r_i \geq 1 - \mathrm{B}_i \cdot y_i & \text{(Constraint 2)} \\
& r_i \leq \frac{a_i(r)}{l_i} & \text{(Constraint 3)} \\
& r_i \in [0,1] & \text{(Constraint 4)} \\
& y_i \in \{0,1\} & \text{(Constraint 5)}
\end{aligned}$$

Figure 4.1: The Mixed-Binary Linear Program for *Central CDS debtors.*

Theorem 7. *Assume a financial network $\mathcal{F} = (N \cup \{\mathcal{CCD}\}, e, c)$ that satisfies the central CDS debtor property and construct the Mixed-Binary Linear Program in Figure 4.1 w.r.t. $\mathcal{F}$ denoted as $MBLP(\mathcal{F})$. It holds that*

i) If $z = (r_i, y_i)_{i \in N}$ is a feasible solution of $MBLP(\mathcal{F})$ then $(r_i)_{i \in N}$ is a clearing vector of $\mathcal{F}$.

ii) If $(r_i)_{i \in N}$ is a clearing vector of $\mathcal{F}$ then there exist a binary vector $(y_i)_{i \in N}$ s.t $z = (r_i, y_i)_{i \in N}$ is a feasible solution of $MBLP(\mathcal{F})$.

Proof. Assume the $MBLP(\mathcal{F})$ of Figure 4.1. The program's constraints contain a mixture of real valued variables $r_i \in [0,1]$ and binary valued decision variables $y_i \in \{0,1\}$ for each bank $i \in N$.

The linearity of all constraints w.r.t $(r_i)_{i\in N}$ and $(y_i)_{i\in N}$ is justified from equations (4.2), (4.3) and (4.4).

i) Let $z = (r_i, y_i)_{i\in N}$ be a feasible solution to the program in Figure 4.1 and w.l.o.g fix an index κ. If $y_\kappa = 0$ then from Constraint 2 and Constraint 4 we get that $r_\kappa = 1$, while subsequently from Constraint 3 it holds that $1 \le a_\kappa(r)/l_\kappa$. Note that Constraint 1 is trivially satisfied due to the choice of B_κ. If $y_\kappa = 1$, then from Constraint 1 and Constraint 3 it holds that $r_\kappa = a_\kappa(r)/l_\kappa$, while combining the latter with Constraint 4 implies that $a_\kappa(r)/l_\kappa \le 1$. Note that Constraint 2 is trivially satisfied due to the choice of B_κ. Consequently for both possible values of the binary variable y_κ, we showed that the satisfied constraints imply that $r_\kappa = \min(1, a_\kappa(r)/l_\kappa)$, thus establishing that the fixed point condition of Definition 2.6 is satisfied by $(r_i)_{i\in N}$. Therefore if $z = (r_i, y_i)_{i\in N}$ is a feasible solution of $MBLP(\mathcal{F})$ then $(r_i)_{i\in N}$ is a clearing vector of $\mathcal{F}$.

ii) Let $(r_i)_{i\in N}$ be a clearing vector of $\mathcal{F}$ and w.l.o.g fix an index κ. We will prove that the sole arrangement for y_κ that satisfies the constraints for κ, corresponds to the configuration implied by configration 4.1 w.r.t $(r_i)_{i\in N}$. Given that $(r_i)_{i\in N}$ is assumed to be a clearing vector, from Definition 2.6, it must hold that $r_\kappa = \min(1, a_\kappa(r)/l_\kappa)$. If $\min(1, a_\kappa(r)/l_\kappa) = 1$, then $r_\kappa = 1$. By setting $y_\kappa = 0$, both Constraint 2 and Constraint 4 are satisfied, while by assumption $1 \le a_\kappa(r)/l_\kappa$ thus Constraint 3 is also satisfied. Trivially Constraint 1 is satisfied under this choice of y_κ due to B_κ. If $\min(1, a_\kappa(r)/l_\kappa) = a_\kappa(r)/l_\kappa$, then $r_\kappa = a_\kappa(r)/l_\kappa$. Consequently by setting $y_\kappa = 1$, both Constraint 1 and Constraint 3 are satisfied, while Constraint 4 is also satisfied since by assumption $a_\kappa(r)/l_\kappa \le 1$. Trivially Constraint 2 is satisfied under this choice for y_κ. In summary we showed that for a given clearing vector $(r_i)_{i\in N}$ of $\mathcal{F}$, setting $(y_i)_{i\in N}$ according to configuration 4.1 constitutes $z = (r_i, y_i)_{i\in N}$ a feasible solution for $MBLP(\mathcal{F})$. □

If we could determine which configurations of the binary decision variable vector $y = (y_i)_{i\in N}$ can generate a clearing vector $r = (r_i)_{i\in N}$, then computing r simply comes down to solving a Linear Program. A direct consequence of the proposed Mixed-Binary Linear Program is an exponential-time algorithm for computing clearing recovery rate vectors for networks meeting the central CDS debtor property.

Theorem 8. CDS-CLEARING *restricted to instances that satisfy the central CDS debtor property admits an exponential time algorithm.*

Proof. Consider a financial network $\mathcal{F} = (N \cup \{\mathcal{CCD}\}, e, c)$, that satisfies the central CDS debtor property and construct the MBLP($\mathcal{F}$) as indicated in Figure 4.1. Based on the structure of MBLP($\mathcal{F}$) and the preceding discussion that states the linearity of the constraints, it is evident that when fixed values for the binary decision variable vector y are introduced, the formulation of Figure 4.1 transforms into a Linear Program w.r.t to the real variable vector $r = (r_i)_{i \in N}$. We denote the generated LP for a fixed vector y as LP($\mathcal{F}|y$). The algorithm iterates over all configurations of $y \in \{0,1\}^{|N|}$ one at a time and invokes any polynomial time algorithm designed for solving linear programs as a subroutine for solving LP($\mathcal{F}|y$). In case LP($\mathcal{F}|y$) is feasible the subroutine algorithm will compute and return a clearing recovery rate vector $r = (r_i)_{i \in N}$ and the algorithm will terminate. In case LP($\mathcal{F}|y$) is infeasible the subroutine algorithm returns nothing, the algorithm considers a new unprocessed configuration for y and repeats the execution of the steps described so far. The algorithm is illustrated below.

Algorithm 1 Exponential time algorithm for computing a clearing vector for *central CDS debtors.*

```
1: Let $\mathcal{F} = (N \cup \{\mathcal{CCD}\}, e, c)$ be the input network that satisfies the *central CDS debtor* property.

2: Construct the Mixed-Binary Linear Program for $\mathcal{F}$ as described in Figure 4.1.
3: **for** $y \in \{0,1\}^{|N|}$ **do**
4:   **if** LP($\mathcal{F}|y$) is feasible **then**
5:     Return the feasible point.
6:     Break.
7:   **end if**
8: **end for**
```

As **CDS-CLEARING** is a total search problem (cf Theorem 1) the existence of a clearing recovery rate vector is guaranteed. Therefore Algorithm 1 is guaranteed to output a clearing recovery rate vector. In the worst case the algorithm would have to execute a polynomial time subroutine on all $2^{|N|}$ possible configurations for vector y. This implies a running time of $\mathcal{O}(poly|\mathcal{F}|) \cdot 2^{|N|}$, where $|\mathcal{F}|$ is the bit length of the input. □

We intentionally avoided specifying a particular objective function. This deliberate choice was made to show the framework's versatility in optimising over a wide range of linear objectives tied to the clearing vector, which results, in an algorithmic scheme for computing the clearing vector, that effectively optimises any given linear objective function. Many concepts on proposed objective functions of interest have been addressed in the literature, but primarily from a computational hardness standpoint [PW22, IDKV23b]. In [PW22], the authors address ambiguity of CRRVs by highlighting how the multiplicity of CRRVs gives rise to optimisation problems whereby one attempts to select the appropriate clearing vector that optimises a desired linear objective. They prove a set of NP-hardness results regarding the choice of the clearing vector r that can satisfy certain objectives expressed as linear functions in r. The reductions therein are based on high capitalised CDS debtors[2] that can be merged into a single central CDS debtor. Our algorithm could be used to resolve this ambiguity for linear objective functions when **Central CDS debtors** are present.

Moreover, despite the exponential time complexity of the algorithm, the procedure is exponential solely in the number of banks in the network: The size of the coefficients in the input (i.e., contract notionals and external assets) have no impact on the exponent.

4.1.2 Mixed-Binary Nonlinear Program for CDS-CLEARING

By adapting the assets, liabilities, the constants $(\mathrm{B}_i)_{i\in N}$ and the binary decision variable y to the expressions

$$a_i(r) = \left[e_i + \sum_{j\in N} r_j \cdot c_{j,i} + \sum_{j,k\in N} r_j \cdot (1-r_k)\cdot c^k_{j,i}\right]$$

$$l_i(r) = \sum_{j,k\in N} (1-r_k)\cdot c_{i,j} + \sum_{j\in N} c_{i,j}$$

$$\mathrm{B}_i = \frac{1}{\sum_{j\in N} c_{i,j}} \cdot \left[e_i + \sum_{j\in N} c_{j,i} + \sum_{j,k\in N} c^k_{j,i}\right] + 1$$

we form a Mixed-Binary Nonlinear Program denoted as MBNLP($\mathcal{F}$) tailored for general instances $\mathcal{F}$ of **CDS-CLEARING**. The construction of the Mixed-Binary Nonlinear Program for general instances of **CDS-CLEARING** suggests heuristic-based approaches and the empirical study of the problem.

[2]CDS debtors with no debt contracts and ∞ external assets.

4.2 Central CDS debtors with covered CDSes

Section 3.2.4 identified **central** and **dedicated** CDS Debtors as two severe restrictions of **CDS-CLEARING** under which the inapproximability result of Theorem 6 remains to hold. We would further like to identify non-trivial and important instances whose solutions admit efficient algorithms.

A *naked* CDS (see Figure 4.2) is a purely speculative contract; its counterparties do not have any other interest in the reference bank if not the CDS itself. In [SSB17a] the authors construct a financial network that does not have any *naked* CDS contracts, yet it still admits irrational solutions [3]. Moreover the authors provide an FPTAS for computing an ϵ weak clearing vector when *naked* CDSes are absent from the financial network (Theorem 6 [SSB17a]). However, regulators could ban their existence and only allow to buy a CDS if a corresponding (debt) exposure exists – i.e., to only have so-called *covered* CDS.

Definition 12 (**Covered CDS** [SSB20, SSB17b])**.** *A credit default swap* (i, j, R) *is* covered *if* $c^R_{i,j} \leq c_{R,j}$.

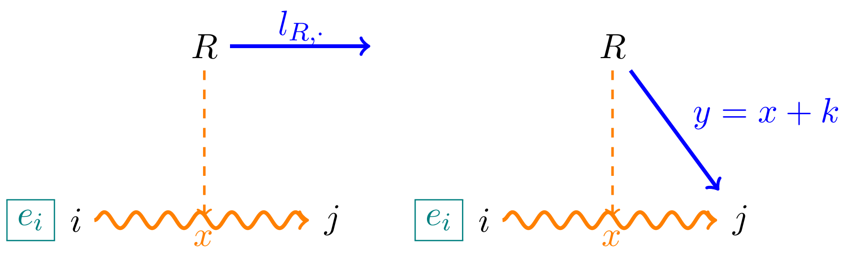


Figure 4.2: Topology of *naked* and *covered* CDS.

We exploit the topological structure of *covered* CDS and central CDS debtors and provide a polynomial time algorithm for computing exact clearing vectors in instances that contain both notions.

Theorem 9. CDS-CLEARING *restricted to instances that only contain* ***covered*** *CDSes and satisfy the central CDS debtor property admits a polynomial time algorithm.*

To prove this, first we perform a transformation step on each credit default swap in the given network, which locally modifies the network. Repeating this process on every CDS contract results

[3](Appendix A of [SSB17a])

in a financial network consisting only of debt contracts, which we can then solve by executing the polynomial-time algorithm presented in [EN01] that computes an exact clearing vector.

Network transformation. Let $\mathcal{F} = (N \cup \mathcal{CCD}, e, c)$ be a financial network that satisfies the central CDS debtor property where all CDS contracts are covered. Consider a CDS contract $(\mathcal{CCD}, j, R)$ in the network, let $x = c^R_{\mathcal{CCD},j}$, let $y = c_{R,j}$ and let k be such that $y = x + k$.

A *network transformation step* on $(\mathcal{CCD}, j, R)$ consists of the following consecutive operations.

1. Update the external assets of j to $e^*_j = e_j + x$;
2. Decrease the contract notional of the debt contract (R, j) to $c^*_{R,j} = k$;
3. Add a dummy node, which we call $\text{dummy}_{(R,j)}$;
4. Add a debt contract $(R, \text{dummy}_{(R,j)})$ with contract notional $c_{R,\text{dummy}_{(R,j)}} = x$;
5. Erase $(\mathcal{CCD}, j, R)$.

Figure 4.3 illustrates this transformation step.

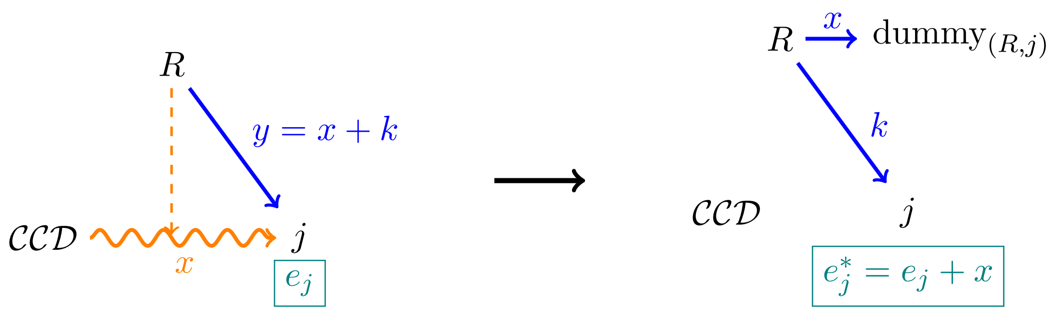


Figure 4.3: The reconfiguration of the dynamics after the removal of the covered CDS.

Let $\mathcal{F}$ be a financial network that satisfies the central CDS debtor property and consider the execution of a single transformation step on a covered CDS $(\mathcal{CCD}, j, R)$ of $\mathcal{F}$. Let r be a CRRV of $\mathcal{F}$, and let $\mathcal{F}'$ be the financial network after the transformation step. We define r' as the recovery rate vector of $\mathcal{F}'$ that is obtained from r by letting $r'_{\text{dummy}(R,j)} = 1$ and letting r' coincide with r for all other banks. Note that, by construction, the total liabilities of R are equal in $\mathcal{F}$ and $\mathcal{F}'$, and j's assets in $\mathcal{F}$ under r are equal to j''s assets in $\mathcal{F}'$ under r'. Hence, r' is a CRRV for $\mathcal{F}'$ under which every node's assets and liabilities remain unaffected with respect to $\mathcal{F}$ under r.

Thus, for a financial network that has only covered CDSes and satisfies the central CDS debtor property, it is possible to obtain an equivalent financial network without CDSes by repeatedly executing the above transformation step on each of the network's CDSes. The resulting CDS-free network then has essentially the same clearing recovery rate vectors as the original network, where the only difference is that the newly introduced dummy nodes always get assigned a recovery rate of 1.

A CRRV for the resulting CDS-free network can be found using e.g. the polynomial time algorithm of [EN01], hence by throwing away from that CRRV the coordinates of the introduced dummy nodes, we obtain a CRRV for the original network in polynomial time. This procedure is summarised as Algorithm 2 below.

Algorithm 2 Polynomial time algorithm for computing exact clearing vectors for instances with central CDS debtors and covered CDSes.

```
1: Let $\mathcal{F} = (N \cup \{\mathcal{CCD}\}, e, c)$ with $N = [n]$ be the input network with only covered CDSes, and
   the Central CDS debtor property.
2: for $j = 1$ to $[n]$ do
3:   for $R = 1$ to $[n]$ do
4:     if $c^R_{\mathcal{CCD},j} \neq 0$ then
5:       $e_j = e_j + c^R_{\mathcal{CCD},j}$
6:       $c_{R,j} = c_{R,j} - c^R_{\mathcal{CCD},j}$
7:       $N = N \cup \text{dummy}_{R,j}$
8:       $c_{R,\text{dummy}_{R,j}} = c^R_{\mathcal{CCD},j}$
9:       $c^R_{\mathcal{CCD},j} = 0$
10:    end if
11:  end for
12: end for
13: Run the polynomial time algorithm of [EN01] to obtain a CRRV $r$, and return $r$ restricted to
   $N \cup \{\mathcal{CCD}\}$.
```

From Lines 2 through 12 the algorithm repeatedly executes the network transformation step, consisting of Operations 1 through 5, on all CDS contracts of the form $(\mathcal{CCD}, j, R)$ of the input financial network. This results in a network $\mathcal{F}'$ that is composed entirely of debt contracts. The final line of the algorithm from Line 13 runs the polynomial time algorithm of [EN01] on $\mathcal{F}'$ to compute the clearing recovery rate vector, and throws away the coordinates corresponding to the introduced dummy nodes.

Lastly, we note that this polynomial time procedure also works correctly under a generalised version of the central CDS debtor property, where we allow multiple banks to be debtors of CDSes, but still require that these CDSes have an amount of external assets exceeding the sum of the notionals, akin to Conditions 2 and 3 of Definition 10. This reflects a setting where banks are risk averse and refuse (or are only authorised) to act as debtor in a CDS when it is certain that all liabilities resulting from these CDSes can be paid off through their assets a-priori.

Chapter 5

Rational and Irrational Solutions

Overview

In this chapter we investigate the existence of irrational solutions to instances of **CDS-CLEARING** and address the following question: Which structural conditions must exactly hold in a financial system for irrational clearing vectors to potentially exist? We aim to characterise such structural conditions of the contract graph independently of the relevant numerical values in these systems (i.e., the values of the external assets, and the notionals on the debt contracts and CDSes). Given a partially specified instance of a financial system where these numerical values are not specified, and thus only the debtors, creditors, and reference banks (in the case of CDSes) are provided to us for each contract in the instance, can we identify in which cases there are numerical coefficients (notionals c and external assets e) for the instance such that under these coefficients, no rational solutions exist?

In Section 5.1 we present a set of sufficient structural conditions that provide a partial answer to this question. We define a type of auxiliary coloured directed graph associated to a financial system for which we examine specific types of cycles and prove that the presence of such a cycle is a structurally sufficient condition for irrational solutions to arise, in the sense that we can then set the rational coefficients such that every clearing vector of the system is irrational. The proof procedure is technically involved, thus further explanatory text has been added in each subsection. In Section 5.2 we study the complementary goal of identifying under which conditions a rational solution must exist and formulate a second set of structural conditions that are close to the former irrationality conditions.

5.1 A Sufficient Structural Condition for Irrational Solutions

5.1.1 Switched Cycles

Assume an instance $I = (N, e, c)$ of **CDS-CLEARING** and let G_I be its contract graph. We construct an auxiliary coloured directed graph $G_{I,\text{aux}}$ as follows: We include all the arcs of the contract graph in $G_{I,\text{aux}}$, which retain their blue and orange colours. Furthermore, for every pair (i, j) such that there exists a CDS $(i, j, R) \in \mathcal{CDS}$ (for some $R \in N$), we add a red-coloured arc (R, i) to $G_{I,\text{aux}}$. Thus, in $G_{I,\text{aux}}$ there is at most one red, one orange, and one blue arc between every ordered pair of nodes. We refer to the resulting tricoloured directed graph as simply the *auxiliary graph* of I. An example of a financial system and its auxiliary graph is given in Figure 5.1.

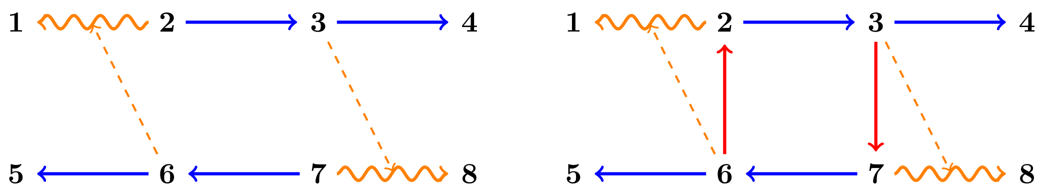


Figure 5.1: A financial system G_I represented as its contract graph, and the corresponding auxiliary graph $G_{I,\text{aux}}$. The dashed lines used in our original graphical notation are retained, for clarity. The coefficients along the arcs and on the nodes are omitted.

Definition 13. *A financial system is called* acyclic *if its auxiliary graph does not contain any directed cycle.*

The financial system of Figure 5.1 is not acyclic, since its auxiliary graph contains the cycle $(2, 3, 7, 6, 2)$. Acyclic financial systems turn out to be easy to analyse: the clearing recovery rate vector is always rational.

Proposition 4. *Every acyclic financial system has only rational solutions.*

Proof. Assume an acyclic financial system $I = (N, e, c)$ and let G and $G_{\text{aux}} = (N, A_{\text{aux}})$ be its contract graph and auxiliary graph respectively. Since I is acyclic, G_{aux} contains no cycles, meaning that for any $(i, j, R) \in \mathcal{CDS}$, there is no path from i to R. Since G_{aux} is a directed acyclic graph (DAG), we can rearrange its nodes in topological order in polynomial time (see e.g. [KT06]). Let

T be the topological ordering of G_{aux}. Without loss of generality, we assume that each node $i \in N$ is also the i'th node in T so that outgoing arcs of i point to higher-numbered nodes, and incoming arcs of i come from lower-numbered nodes.

We observe that the recovery rate r_i of any node $i \in N$ is determined by only the recovery rates of banks $j < i$, under the clearing condition, which holds because both the assets $a_i(r)$ and liabilities $l_i(r)$ under any clearing vector do not depend on any of the recovery rates $r_{i+1}, \dots, r_n$ (following (2.1) and (2.5)). This means that we can straightforwardly compute a clearing vector of recovery rates by iterating over the banks in topological order T, and using (2.1) and (2.5) to compute r_i from $r_1, \dots r_{i-1}$ in each iteration i.

In each iteration, the computation of the numerator and denominator of the recovery rate involves summations of terms containing multiplications, subtractions, and additions among recovery rates and rational constants. The numerator and denominator are thus rationals if all preceding recovery rates are rationals as well. Since the recovery rate computed in the first iteration is rational, by induction all computed recovery rates are rational. $\square$

This initial insight leads us to focus on specifying cyclic structures that might be capable of generating irrational solutions. The classical results of [EN01] show that financial systems without credit default swaps always have rational (and polynomial time computable) recovery rates. Thus, for irrational solutions to emerge, we know that the presence of just *any* cycle is not sufficient, and that CDSes must be involved.

Definition 14 (**Switched off/Switched on nodes**). *We define a node $i \in N$ as* **switched off** *if its number of incoming red arcs equals 1, and i does not have outgoing blue arcs (this implies that all CDSes of which i is the debtor share the same reference bank). We define a node $i \in N$ as* **switched on** *if one of the following holds:*

- *its number of incoming red arcs exceeds 1;*
- *its number of incoming red arcs equals 1 and there is at least one outgoing blue arc.*

We note that ***switched on*** and ***switched off*** nodes are not complements of each other: A node that is not a debtor in any CDS is neither switched on nor switched off. Figure 5.2 below illustrates these notions. Note also that ***switch off*** nodes are essentially what we previously define as dedicated CDS debtors. The purpose of renaming these nodes is made for the sake of presenting nodes as switches that either allow or stop irrational payments to flow through them.

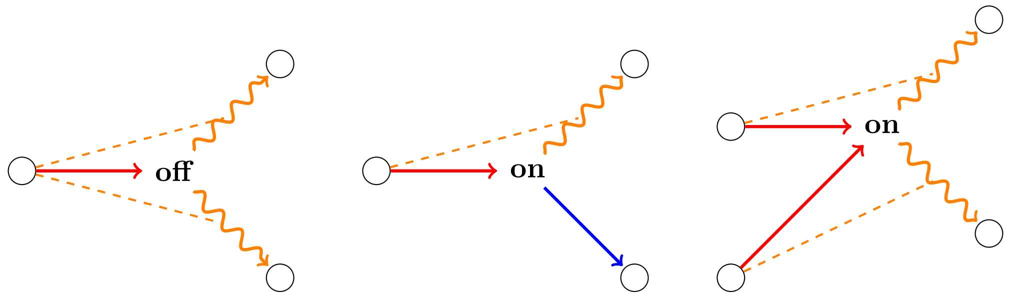


Figure 5.2: Illustration of switched off and switched on nodes.

Definition 15 (**Switched Cycles**)**.** *We call a cycle* red *if it has at least one red arc.*

- *A cycle C is called* **weakly switched** *if it is red, and for at least one red arc (u, v) in C, v is switched on.*
- *A cycle is called* **strongly switched** *if it is red, and for each red arc (u, v) in C, v is switched on.*

The notion of switched cycles is central to the occurrence of irrational solutions in financial systems. We will show that when a financial system I has a *strongly* switched cycle, and satisfies a certain additional technical condition (to be specified later), there exist rational coefficients for the financial system under which all clearing vectors of I are irrational. Conversely, in the subsequent section we show that in the absence of any *weakly* switched cycle, there exist rational clearing vectors of I regardless of its coefficients, and the computational problem of computing an exact fixed point in such instances lies in the complexity class $\mathsf{P}_{\mathbb{R}}^{\mathsf{PPAD}}$, where $\mathsf{P}_{\mathbb{R}}$ is the class of problems polynomial time solvable under the Blum-Shub-Smale model defined in [BCSS98] (see Corollary 4).

5.1.2 Rewriting Rules for Strongly Switched Cycles

In this section, we first present a framework for formulating strongly switched cycles, and we present various "rewriting rules" that capture an equivalence relation between strongly switched cycles in terms of their recovery rates. We begin by defining a set of primitive financial systems, represented in the form of their auxiliary graph, which we call *fragments*. These fragments lack coefficients:

They specify only some of the contracts present among a set of nodes, but do not contain any specification of the notionals on these contracts, and do not contain a specification of the external assets of any of the banks. Each of these fragments has a designated start and end node, and each of these fragments has the property that there is a unique path from the start to the end node. Furthermore, each fragment has the property that all nodes are directly connected to the unique path from start to end node.

We then define a binary operation that concatenates copies of fragments with each other, by identifying the start node of one copy of a fragment with the end node of a copy of another fragment. This results in auxiliary graphs of financial systems that are obtainable by "stringing" together fragments. We refer to graphs obtainable through this concatenation operation as *fragment strings*. A fragment string represents a path (starting at the start node of the first fragment and ending at the end node of the last fragment) along with some further neighbouring nodes that are directly attached to the main path through arcs.

In turn, the start and end nodes of the paths that can be formed in this way can be connected together: The end node of the last fragment in a fragment string can be identified with the start node of the first fragment in the fragment string, and this will create a (coefficientless) auxiliary graph of some financial system that contains a single cycle, along with some neighbouring nodes with arcs that are attached to the nodes in the main cycle. We call such a graph a *fragment cycle*.

We will explain below in detail how these fragments are defined and how to form cycles with them. Subsequently, we will equip each fragment with particular choices of coefficients (i.e., notionals and external assets) that we will later on show how to generate unique irrational solutions when composed into fragment cycles.

We study the resulting set of fragment cycle graphs, constructible through attaching such fragments to each other, with the following goal in mind: Given an arbitrary financial system I, we will show that if a certain subset of the graphs constructible from our set of fragments occurs as a subgraph in $G_{I,\text{aux}}$, then there exists a choice of rational coefficients for I such that all clearing vectors of I are irrational. This will yield us our intended structural characterisation of irrational financial systems. Therefore, the fragments we will define, and the graphs obtainable from stringing them together, should be interpreted as subgraphs of a larger financial system under consideration, and we want to determine whether this larger financial system is susceptible to having irrational clearing vectors.

Fragment Strings

We denote by $\mathcal{G}$ the set of all fragments that we will use. All fragments of $\mathcal{G}$ are defined in Figure 5.3, presented in our tricoloured graphical notation. Each fragment has designated start and end nodes, which are indicated by short incoming and outgoing black arrows, respectively. It can be verified that for each fragment indeed there is a single path from input to output node, as we claimed previously. Furthermore, as also claimed above, each node of any given fragment is either on this path, or directly connected to it by a single (incoming or outgoing) arc. Note that there are many fragments in $\mathcal{G}$ that are highly similar, and the names of our fragments are chosen such that highly similar fragments differ only in their superscripts: For example, the fragments g_1^a, g_1^b, g_1^c, and g_1^d all have in common that the main path from the input node to the output node consist of a red arc followed by an orange arc that corresponds to the same CDS as the red arc, and these four fragments only differ in the colours of two arcs attached to the main path (along with one or two external reference banks, denoted by c, c_1, or c_2). The two sets of fragments $\{g_2^a, g_2^b, g_2^c, g_2^d\}$ and $\{g_3^a, g_3^d\}$ are related in a similar fashion.

We define a binary merging operation on ordered pairs of fragments (a, b), where every pair (a, b) is mapped to a graph obtained by taking disjoint copies of a and b, and connecting the two copies together by identifying the end node of a with the start node of b. We define the new start node and end node of the resulting system to be the start node of the copy of a and the end node of the copy of b, respectively. We denote the result of the merge operation on fragments a and b symbolically by the notation ab. A *fragment string* is a fragment obtainable from fragments in $\mathcal{G}$ using any number of sequential applications of the merge operation. We let $\mathcal{GS}$ be the set of fragment strings (i.e., the closure of $\mathcal{G}$ under the *merge* operation).

We may turn any fragment string into a *fragment cycle* by identifying (i.e., connecting) its start node with its end node. We use the following symbolic notation for fragment cycles: For a fragment string $x_1x_2 \cdots x_{k-1}x_k$ of k fragments, the corresponding fragment cycle will be written symbolically by marking the first fragment x_1 as $\dot{x_1}$ and appending $\dot{x_1}$ to the end of the string, i.e., we write $\dot{x_1}x_2 \cdots x_{k-1}x_k\dot{x_1}$ to denote the fragment cycle corresponding to fragment string $x_1x_2 \cdots x_{k-1}x_k$.

Definition 16. *We define $\mathcal{GC}$ to be the set of all fragment cycles, i.e., the graphs $\dot{x}g_s\dot{x}$, where*

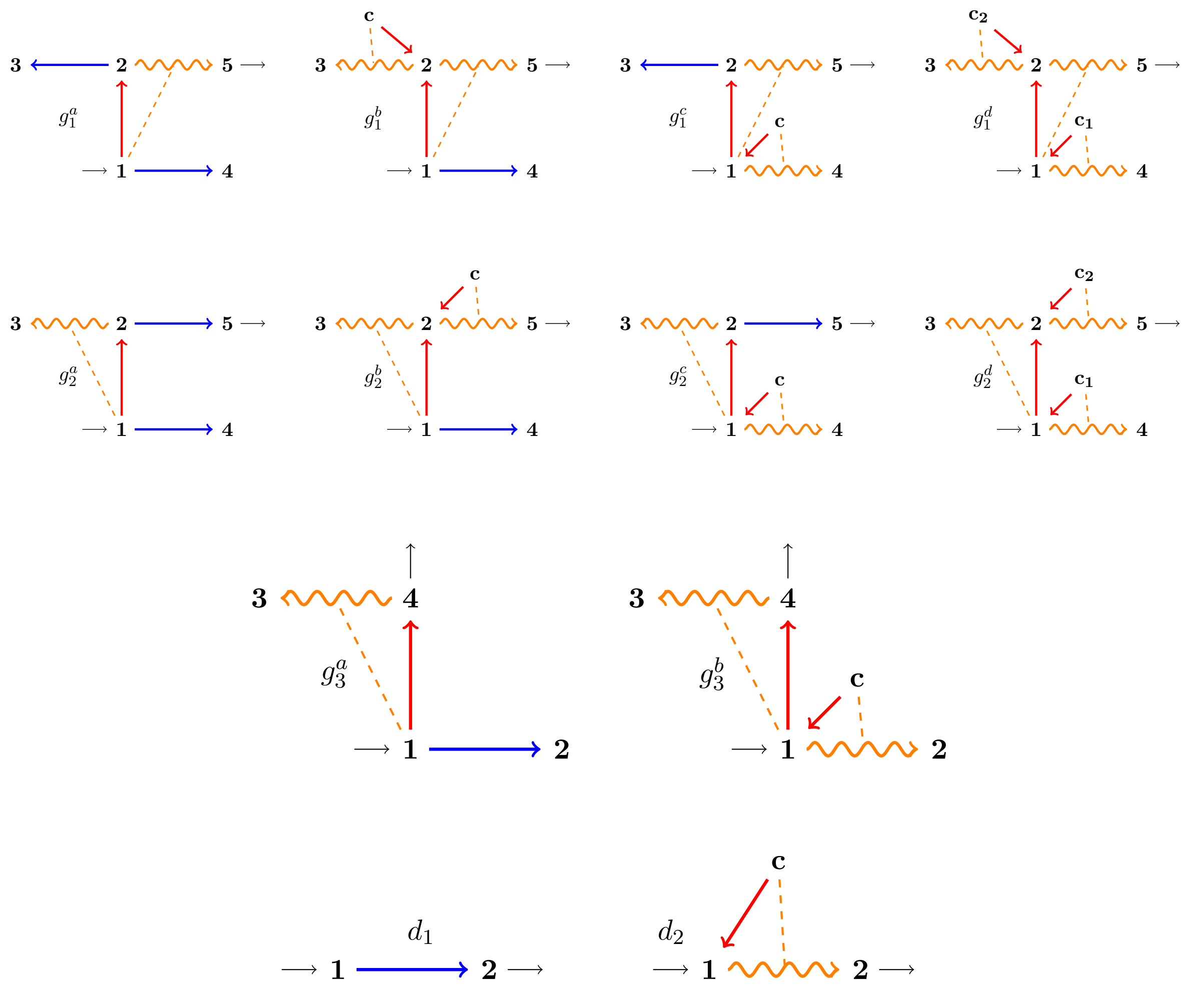


Figure 5.3: The fragments in $\mathcal{G}$. Each fragment is labeled with a name that we will use to refer to the individual fragments. We have $\mathcal{G} = \{g_1^a, g_1^b, g_1^c, g_1^d, g_2^a, g_2^b, g_2^c, g_2^d, g_3^a, g_3^b, d_1, d_2\}$

$x \in \mathcal{G}$ and $g_s \in \mathcal{GS}$.

Arithmetic Fragment Strings

A fragment in $\mathcal{G}$ can correspond to many concrete financial systems: The blue and yellow arcs in our fragments represent debt contracts, but their notionals are unspecified. Similarly, the nodes represent banks, but a specification of their external assets is not given. A fragment equipped with coefficients specifying these notionals and external assets will be referred to as an *arithmetic fragment.* Figure 5.4 presents the set of arithmetic versions of the fragments in $\mathcal{G}$ that we will be using. We use the notational convention that x' or x'' is an arithmetic version of a fragment $x \in \mathcal{G}$. For some of these arithmetic fragments, we have annotated the end nodes with red labels that indicate the assets of the end node under any clearing vector as a function of the recovery rate r of the start node of the fragment. Note that if the external assets of a bank are set to 0, our convention is to omit the green label at the respective node. Furthermore, all nodes labeled with c, c_1, and c_2 are assumed to have a recovery rate of 0, which is achieved by setting the external assets of such banks to 0 and setting the coefficients in the financial system (in which the fragment is embedded) such that c has a strictly positive liability.

Many of the arithmetic fragments in Figure 5.4 are highly similar, and this has been reflected in their names: For $i \in [3]$ and $j \in \{a, b, c, d\}$, the arithmetic fragment $g_i^{j'}$ differs from $g_i^{j''}$ in only a single contract's notional. Similarly to non-arithmetic fragment strings, we may string together arithmetic versions of fragment strings, and we may apply our symbolic notation to denote strings and cycles of arithmetic fragments.

Next, we study the recovery rates of strings and cycles of arithmetic fragments. We start with the following observations.

Observation 4. *Let x'_1 and x'_2 be any two consecutive arithmetic fragments in a string or cycle C of arithmetic fragments. Let r be the recovery rate of the start node of x'_1 under a clearing vector of C under the assumption that all nodes labeled with c, c_1, and c_2 have recovery rate 0.*

- *If $x'_1 \in \{g_i^{j'}, g_i^{j''} \ : \ i \in [2], j \in \{a, b, c, d\}\}$ and $x'_2 \in \{g_i^{j'} \ : \ i \in [2], j \in \{a, b, c, d\}\} \cup \{d'_1, d'_2\}$, then the recovery rate of the end node of x'_1, is $(1 - r)/(2 - r)$ or $1/(3 - r)$, as indicated by the red labels in Figure 5.4.*

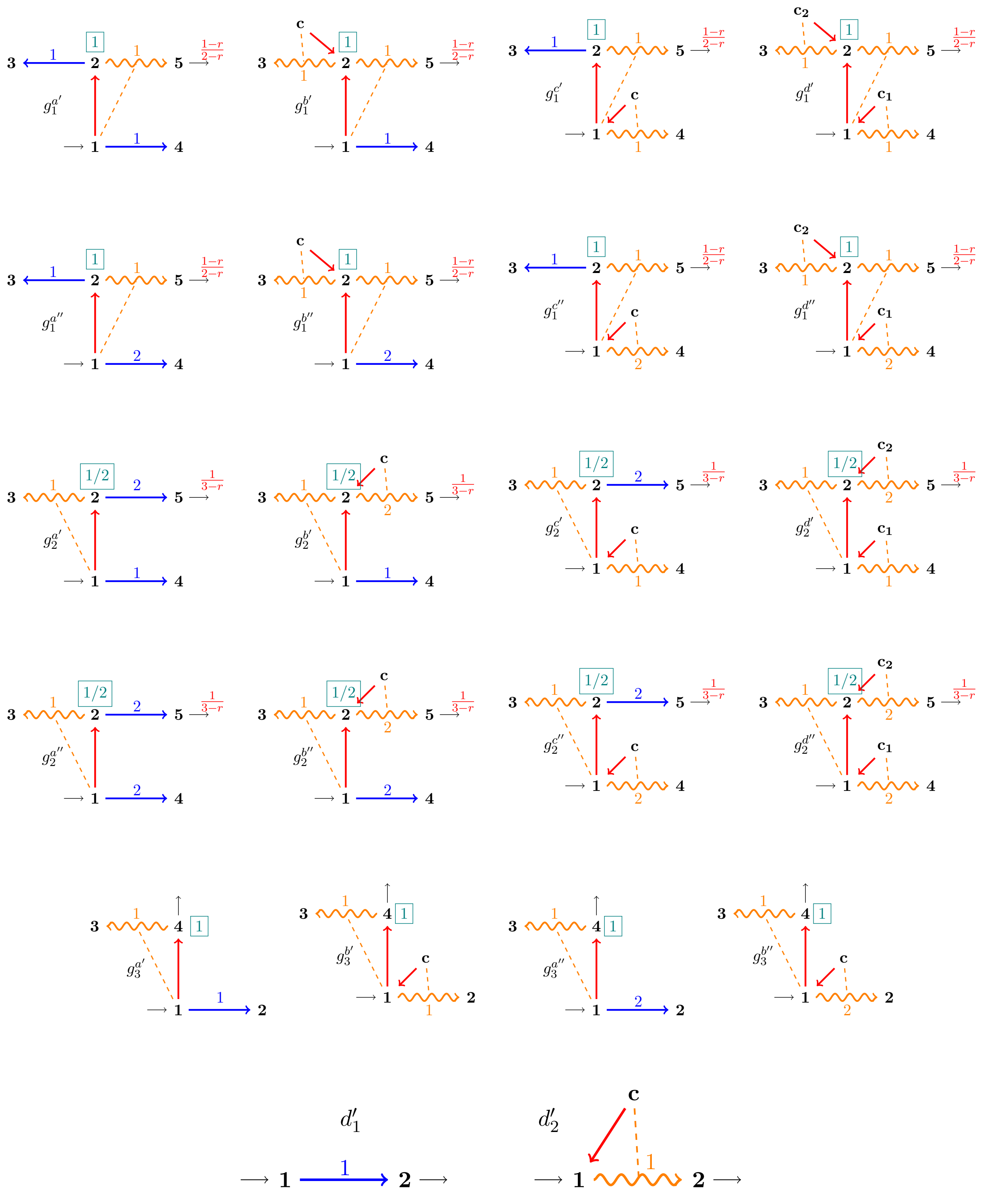


Figure 5.4: Full set of arithmetic fragments. All nodes labeled with c, c_1, and c_2, are assumed to have a recovery rate of 0, which is achieved by setting the external assets of c to 0 and setting the coefficients in the financial system (in which the fragment is embedded) such that c has a strictly positive liability.

- *If* $x_1' \in \{g_3^{j'}, g_3^{j''} \ : \ j \in \{a, b\}\}$ *and* $x_2' \in \{g_i^{j''} : i \in [3], j \in \{a, b, c, d\}\}$*, then the recovery rate of the end node of* x_1' *is* $1/(3-r)$*.*

Next we provide a notion of *equivalence* among fragment strings.

Definition 17 (**Equivalence**)**.** *Let* x_s^1*,* x_s^2 *be two arithmetic fragment strings. We say that* x_s^1 *and* x_s^2 *are* equivalent *iff the recovery rate of the end node of* x_s^1 *equals the recovery rate of the end node of* x_s^2 *for all possible choices* $r \in [0, 1]$ *of the recovery rate of the input node of* x_1^s *and* x_2^s *respectively, under the assumption that all nodes labeled with* c*,* c_1*, and* c_2 *have a recovery rate of* 0*.*

Using the notion of equivalence, we provide a set of *rewriting rules* for the symbolic formulations of our arithmetic fragment strings and cycles. We can use such rules to rewrite a given fragment string or cycle into an equivalent one that is simpler to analyse with respect to the output recovery rate. By repeatedly applying such rewriting rules, we enable ourselves to produce a family of equivalent reformulations of a given arithmetic fragment string or cycle, such that the clearing recovery rate vector (and in particular the output rate of the rewritten fragment string) can be proved to be irrational.

The rewriting rules will all be stated under the previously made assumption that nodes labeled with c, c_1, and c_2 in the fragments all have recovery rate 0.

Rule 0: In any arithmetic fragment string or cycle, we may replace an occurrence of a fragment $g_i^{j'}$, where $i \in [3]$ and $j \in \{a, b, c, d\}$, with the fragment $g_i^{a'}$. Similarly, we may replace an occurrence of a fragment $g_i^{j''}$ with the fragment $g_i^{a''}$. It is straightforward to see that the recovery rate of the end node in the replaced fragment has not changed as a function of the recovery rate of the start node, and therefore the resulting fragment string is equivalent to the original.

Rule 1: In any arithmetic fragment string or cycle, we may replace an occurrence of a fragment $g_2^{a'}$ (respectively $g_2^{a''}$) by $g_1^{a'}g_1^{a'}$ respectively $g_1^{a''}g_1^{a'}$ if the fragment $g_2^{a'}$ (or respectively $g_2^{a''}$) is followed by one of the fragments in $\{g_1^{a'}, g_2^{a'}, g_3^{a'}, d_1', d_2'\}$. This rewriting rule is valid because the recovery rate of the end node of $g_2^{a'}$ and $g_2^{a''}$ is equal to $1/(3-r)$, and the recovery rate of $g_1^{a'} g_1^{a'}$ (respectively $g_1^{a''} g_1^{a'}$) is given by

$$\frac{1-\frac{1-r}{2-r}}{2-\frac{1-r}{2-r}} = \frac{1}{3-r}.$$

Rule 2: In any arithmetic fragment string or cycle, we may replace an occurrence of a consecutive pair of fragments $g_3' g_i^{a''}$, where $g_3' \in \{g_3^{a'}, g_3^{a''}\}$, and $i \in [3]$, by the fragments $g_2^{a'} g_i^{a'}$. By Observation 4, the recovery rates of the end nodes of g_3' and $g_2^{a'}$ are identical under this substitution, under any clearing vector, so that the two fragment strings are equivalent.

Rule 3: In any arithmetic fragment string or cycle, we may remove an occurrence of d_1' or d_2'. This substitution is straightforward from the fact that both d_1' and d_2' just transfer the recovery rate from the start to the end node.

Example 5. *Given the fragment cycle $\dot{g}_1^a g_2^b d_1 d_2 \dot{g}_1^a$, we may choose the coefficients in order to obtain the arithmetic fragment cycle $\dot{g}_1^{a'} g_2^{b'} d_1' d_2' \dot{g}_1^{a'}$. The following two fragment cycles are then equivalent.*

1. *$\dot{g}_1^{a'} g_2^{a'} d_1' d_2' \dot{g}_1^{a}$ (by applying Rule 0),*
2. *$\dot{g}_1^{a'} g_2^{a'} \dot{g}_1^{a'}$ (by applying Rule 3),*
3. *$\dot{g}_1^{a'} g_1^{a'} g_1^{a'} \dot{g}_1^{a'}$ (by applying Rule 1).*

5.1.3 Irrationality of Strongly Switched Cycles

Consider any instance of a financial system I. Let $G_{I,\text{aux}}$ be its auxiliary graph, and suppose that this system has a strongly switched cycle. Then, this cycle is composed entirely of our fragments in Figure 5.3. This is formalised as follows. We will use the notation $V(G)$ and $E(G)$ to denote the set of vertices and the set of arcs of a directed graph G, respectively.

Definition 18. *Let G' be a fragment cycle, and let C' be the unique directed cycle in G'. The fragment cycle G' is said to* agree with a cycle C of $G_{I,\text{aux}}$ *iff there is a mapping $g : V(G') \to V(G_{I,aux})$ with the following properties:*

- *For all $(v, w) \in E(G')$, the arc $(g(v), g(w))$ is in $E(G_{I,aux})$ and has the same color as (v, w).*
- *g restricted to the domain $V(C')$ defines a bijection between $V(C')$ and $V(C)$.*

- *For each CDS (i, j, R) in G', $(g(i), g(j), g(R))$ is a CDS of $G_{I,aux}$.*

Note that the above points imply that g restricted to $V(C')$ defines an arc-color-preserving isomorphism between C' and C. However, this isomorphism property does not necessarily extend to node sets larger than C': nodes in $V(G')$ that are not in $V(C')$ may be mapped by g to the same vertex of $G_{I,aux}$.

Furthermore, we define the fragment cycle G' to simply *agree with a cycle C of $G_{I,aux}$, if G' agrees with cycle C of $G_{I,aux}$ through a mapping g for which it additionally holds that*

- *all nodes outside C' are mapped to vertices outside C,*
- *For every pair of nodes $\{u, v\} \subseteq V(G')$, where u is a node labeled with c, c_1, or c_2 (in Figure 5.4) and v is labeled with a number (in Figure 5.4), $g(u) \neq g(v)$, and*
- *for every node u of G' labeled with c, c_1, or c_2, $g(u)$ has an outgoing arc pointing towards a node not in C'.*

The notion of *simple* agreement defined above is a somewhat technical one. Informally stated, it is a mild condition that requires that the neighbouring nodes of C' are sufficiently "independent" from each other and from the cycle C, under the mapping g. This brings us to the definition of a *simple* strongly switched cycle.

Definition 19. *A cycle C of $G_{I,aux}$ is a* simple *strongly switched cycle iff C is strongly switched, and for each red arc (u, v) of C there are non-red arcs (u, u') and (v, v') such that $u', v' \notin C$. Furthermore, if (u, u') or (v, v') is orange, then the reference bank R of the corresponding CDS is not in C and R has an outgoing non-red arc pointing to a node not in C.*

From the definition of our fragments $\mathcal{G}$, it is straightforward to see that our fragments can represent any strongly switched cycle: If $G_{I,\text{aux}}$ has a strongly switched cycle C, then there is a fragment cycle G' consisting of fragments in $\mathcal{G}$ such that G' agrees with C of $G_{I,\text{aux}}$. Similarly, if $G_{I,\text{aux}}$ has a simple strongly switched cycle C, then there is a fragment cycle G' consisting of fragments in $\mathcal{G}$ such that G' simply agrees with C of $G_{I,\text{aux}}$. All nodes of C that are switched on correspond to the 2-labeled nodes of a g_2^j or g_1^j fragment, for some $j \in \{a, b, c, d\}$.

Next, we prove two lemmas that show that we can set the coefficients in any strongly switched fragment cycle such that the fragment cycle admits only irrational clearing recovery rates. We start by considering formulations consisting only of g_1^j fragments.

Lemma 3. *For all fragment cycles $C \in \mathcal{GC}$ consisting of only fragments in $\{g_1^j : j \in \{a,b,c,d\}\}$, there exist coefficients such that the clearing recovery rate vector of C is irrational (under the assumption that all nodes labeled with c,c_1, and c_2 have a recovery rate of* 0*).*

Proof. Consider a fragment cycle consisting exclusively of only fragments in $\{g_1^j : j \in \{a,b,c,d\}\}$. For all $j \in \{a,b,c,d\}$, fix the coefficients of all g_1^j fragments in the cycle to obtain the arithmetic version $g_1^{j'}$. Use rewriting Rule 0 to replace all $g_1^{j'}$ occurrences by $g_1^{a'}$. The resulting arithmetic fragment cycle consists of a number of consecutive copies of $g_1^{a'}$, say k of them. Consider now any clearing vector for the fragment cycle. denoting the $r \in [0,1]$. We prove by induction that the end node of the ith fragment has recovery rate equal to $(f_i - rf_{i-2})/(f_{i+2} - rf_i)$, where f_i is the ith Fibonacci number, with $f_0 = 0$.

As pointed out in Observation 4, the recovery rate of the end node of the first fragment equals $\frac{1-r}{2-r}$. Repeating this argument once, we obtain that the recovery rate of the end node of the second fragment equals

$$\frac{1-\frac{1-r}{2-r}}{2-\frac{1-r}{2-r}} = \frac{1}{3-r} = \frac{f_2 - f_0 r}{f_4 - f_2 r},$$

proving the base case.

Next, assume that the claim holds for the ith fragment in the fragment cycle. Taking the recovery rate of $(f_i - rf_{i-2})/(f_{i+2} - rf_i)$ as the recovery rate of the start node of fragment $i+1$, we obtain that the recovery rate of its end node is

$$\frac{f_{i+2} - rf_i - f_i + rf_{i-2}}{2f_{i+2} - 2rf_i - f_i + rf_{i-2}} = \frac{f_{i+1} - rf_{i-1}}{f_i + 2f_{i+1} - rf_{i-2} - 2rf_{i-1}} = \frac{f_{i+1} - rf_{i-1}}{f_{i+2} + f_{i+1} - r(f_i + f_{i-1})} = \frac{f_{i+1} - rf_{i-1}}{f_{i+3} - rf_{i+1}},$$

which establishes the inductive step.

We know that the end node of the last fragment in the fragment cycle has a recovery rate that coincides with the recovery rate r of the start node of the first fragment. Therefore, in a clearing vector of recovery rates, it holds that $r = \frac{f_k - rf_{k-2}}{f_{k+2} - rf_k}$ which is equivalent to solving the equation $r^2 f_k - (f_{k+2} + f_{k-2})r + f_k = 0$. Since $f_{k+2} + f_{k-2} = f_{k+1} + f_k + f_{k-2} = 2f_k + f_{k-1} + f_{k-2} = 3f_k$, computing the recovery rate of the initial node 1 comes down to solving the quadratic equation $r^2 - 3r + 1 = 0$. Solving this equation we obtain that the only solution in $[0,1]$ is $r = (3-\sqrt{5})/2$ which is irrational, thus the clearing recovery rate vector of the strongly switched arithmetic fragment cycle is irrational and is unique. □

The next lemma extends the above to a larger class of arithmetic fragments.

Lemma 4. *For all fragment cycles composed of fragments $\mathcal{G}$ in which every occurrence of a fragment in $\{g_3^j : j \in \{a,b\}\}$ is followed by a fragment in $\{g_i^j : i \in [2], j \in \{a,b,c,d\}\}$, there exist coefficients such that the clearing recovery rate vector of C is irrational (under the assumption that all nodes labeled with c, c_1, and c_2 have a recovery rate of 0).*

Proof. Consider any fragment cycle with the property described in the claim. Fix the coefficients of all fragments as follows.

- For a fragment $f = g_i^j, i \in [3], j \in \{a,b,c,d\}$ occurring in the cycle, if the fragment preceding it is in $\{g_3^j : j \in \{a,b\}\}$, turn f into the arithmetic fragment $g_i^{j''}$.
- For a fragment $f = g_i^j, i \in [3], j \in \{a,b,c,d\}$ occurring in the cycle, if the fragment preceding it is not in $\{g_3^j : j \in \{a,b\}\}$, turn f into the arithmetic fragment $g_i^{j'}$.
- Turn every occurrence of d_1 into d_1', and turn every occurrence of d_2 into d_2'.

Given the resulting arithmetic fragment cycle, we apply rewriting Rule 0 to all fragments in order to obtain a fragment cycle consisting only of arithmetic fragments in $\{g_1^{a'}, g_1^{a''}, g_2^{a'}, g_2^{a''}, g_3^{a'}, g_3^{a''}, d_1', d_2'\}$. We then use Rule 3 to remove all occurrences of d_1' and d_2' from the cycle, followed by Rule 2 to remove all occurrences of g_3 from the cycle, followed by Rule 1 to remove all occurrences of g_2 from the cycle, resulting in an arithmetic fragment cycle consisting of only a sequence of copies of g_1^a. Claim 3 now completes the proof. □

We may now use these last two claims to yield the main result of this section.

Theorem 10. *Let I be a non-degenerate financial system such that $G_{I,aux}$ has a simple strongly switched cycle. Then there exist rational coefficients for I such that all clearing vectors of I are irrational.*

Proof. Let C be a strongly switched cycle of $G_{I,\text{aux}}$ and let G' be a fragment cycle that simply agrees with C through a mapping g satisfying the conditions stated in Definition 18. By Lemma 4, there are notionals c and external assets e for G' such that all clearing vectors of G' are irrational, under the assumption that the nodes labeled with c, c_1, and c_2 have a recovery rate of 0. In $G_{I,\text{aux}}$, we can now set the notionals and external assets on the vertices and arcs such that they agree with c and e through the mapping g:

- For each $v \in V(G')$, we set the external assets of $g(v)$ to e_v.
- For each $(v, w) \in E(G')$ such that (v, w) is a blue arc, we set the notional on the contract $(g(v), g(w))$ to $c_{v,w}$.
- For each $(v, w) \in E(G')$ such that (v, w) is an orange arc, we set the notional on the contract $(g(v), g(w))$ to $c^R_{v,w}$, where R is the reference bank corresponding to the CDS arc (v, w).

This assignment of coefficients is well-defined by the properties of g stated in Definition 18 (i.e., there are no two arcs or vertices that get assigned multiple conflicting coefficients this way). We set the remaining coefficients of $G_{I,\text{aux}}$ (i.e., the coefficients on the arcs and vertices outside the image of g) as follows:

- We set the external assets to 0 for every node $v \in V(G_{I,\text{aux}})$ that is not in the image of g.
- We set the notional to 1 on every arc $(v, w) \in E(G_{I,\text{aux}})$ such that $g^{-1}(v)$ is a node labeled with c, and w is not in the image of g.
- We set the notional to 0 on every arc $(v, w) \in E(G_{I,\text{aux}})$ that does not satisfy the condition in the point above.

Note that by the simplicity property of the agreement between G' and C (see Definition 18), the second point in the above list ensures that for each node $v \in V(G')$ labeled with c, c_1 and c_2, the node $g(v)$ has a recovery rate of 0. Furthermore, if we denote by G'' the subgraph of G formed by the image of g (i.e., G'' is the projection of G' to $G_{I,\text{aux}}$ through g), we can see that the above setting of the coefficients outside of the image of g ensures that no payments flow from G'' to any node outside G'' under any clearing vector. It then follows by Lemma 4 and the simple agreement properties, that under this setting of the coefficient of $G_{I,\text{aux}}$, every clearing vector is irrational (and in particular these irrational recovery rates emerge in the nodes of G''). This establishes our claim. $\square$

5.2 Financial Systems with Guaranteed Rational Solutions

In the previous section, we identified a sufficient structural condition for the ability of a financial system to have irrational clearing vectors. In this section we investigate how close these conditions are to a characterisation, by attempting to answer the opposite question: Under which structural conditions are rational clearing vectors guaranteed to exist in a financial system? The answer to this relates again to the notion of switched cycles: We will show that if a given non-degenerate financial system does not possess any weakly switched cycle, then there must exist clearing vectors of the system that are rational. We investigate furthermore the computational complexity of finding a clearing vector in this case: Solutions can, informally stated, be computed by solving a linear number of PPAD-complete problems. This latter result is achieved through identifying a natural class of financial systems for which the problem of computing an exact fixed point is PPAD-complete.

The results in this section indicate that the structural conditions for irrationality formulated in the previous section do close in on a characterisation, although there is still a "gray area" left: For those instances of financial systems that do have weakly switched cycles, but do not have any simple strongly switched cycles, we are not yet able to determine by the structural interrelationships of the financial contracts whether these systems are likely to possess rational or irrational solutions. This forms an interesting remaining problem that we leave open.

The main result we will prove in this section is thus the following.

Theorem 11. *Let I be a non-degenerate financial system. If $G_{I,aux}$ does not have any weakly switched cycles, then all clearing vectors of I are rational.*

We start by showing that for financial networks that satisfy the**Dedicated CDS debtors**property (see Definition 11) a particular subclass of financial systems without weakly switched cycles, the clearing vector computation problem lies in Linear-FIXP, which is equal to PPAD (as per Theorem 2), and thus the clearing vectors of such financial system must have polynomial size rational coefficients.

Lemma 5. *(The exact computation version of)* **CDS-CLEARING** *restricted to non-degenerate financial systems with the dedicated CDS debtor property is PPAD-complete.*

Proof. Let $I = (N, e, c)$ be a non-degenerate financial system with the dedicated CDS debtor property.

Let $D \subseteq N$ be the set of all nodes that are a debtor in at least one CDS of I. We first show containment in Linear-FIXP, after which we will establish Linear-FIXP-hardness as well.

First, note that by non-degeneracy of the instance I, for all $i \in D$ the reference bank R of which i is the debtor in all its CDS contracts satisfies that $R \in N \setminus D$. We will show that the following function f' is in Linear-FIXP: The function $f' : \mathbb{R}^{|\mathcal{CDS}|} \times [0,1]^{|N \setminus D|} \to \mathbb{R}^{|\mathcal{CDS}|} \times [0,1]^{|N \setminus D|}$ maps a vector of payments p (one payment for each of the CDS contracts of I) combined with a vector of recovery rates r' for the nodes in $N \setminus D$, to vectors of the same dimensions. The fixed points of f' are those points (p, r') for which there is a clearing vector r such that r' coincides with r on $N \setminus D$, and p coincides with the payments through the CDS contracts under r. In other words, when comparing our original function f of (2.7) to f', the difference is that f' does not take the recovery rates of D as arguments, but instead takes the payments made by D as arguments. This change of fixed-point function is made in order to ensure that f' can be computed using only the operations $\{+, -, \min\}$ and multiplications by constants. Given a fixed point (p, r') of f', it is then easy to compute a corresponding fixed point of the original f, as we can compute the recovery rates of D from their payments p, given the recovery rates r' of the remaining nodes. This places the problem in Linear-FIXP, as the class is closed under polynomial-time reductions (see Remark 1), and hence also in PPAD by Theorem 2.

The function f' is defined as follows. Let (p, r') denote a vector of payments on the CDS contracts of I together with the recovery rates of $N \setminus D$. For a node $i \in N \setminus D$, f' defines the recovery rate

$$f'_i(p, r') = \min\left\{1, \frac{a_i(p, r')}{l_i(p, r')}\right\}, \text{ where } l_i(p, r') = \sum_{j \in N} c^{\emptyset}_{i,j}, \text{ and } a_i(p, r') = \sum_{j \in N \setminus D} r'_j c^{\emptyset}_{j,i} + \sum_{j \in D} p_{j,i}.$$

For a node $i \in D$, if there is a CDS contract with debtor i and creditor j, the payment on this contract specified by f' is defined as follows. Let $R \in N \setminus D$ be the unique reference bank of the CDSes in which i is the debtor.

$$f'_{i,j}(p, r') = \min\left\{1, \frac{a_i(p, r')}{l_i(p, r')}\right\}(1 - r'_R)c^R_{i,j} = \min\left\{(1 - r'_R)c^R_{i,j}, (1 - r'_R)c^R_{i,j}\frac{a_i(p, r')}{l_i(p, r')}\right\}, \tag{5.1}$$

where $a_i(p, r')$ are the assets of bank i,

$$a_i(p, r') = e_i + \sum_{k \in N \setminus D} r'_k c^{\emptyset}_{k,i} + \sum_{k \in D} p_{k,i},$$

and $l_i(p, r')$ denotes the liabilities of bank i, given by:

$$l_i(p, r') = (1 - r'_R) \sum_{k \in N} c^R_{i,k}.$$

We can therefore rewrite (5.1) to

$$f'_{i,j}(p, r') = \min\left\{\left(1 - r'_R\right) c^R_{i,j}, c^R_{i,j} \frac{e_i + \sum_{k \in N \setminus D} r'_k}{\sum_{k \in N} c^R_{i,k}}\right\},$$

which makes it clear that we can compute $f'_{i,j}(p)$ using $\{+, -, \min\}$ and multiplication-by-constant gates. This places the problem of computing a fixed point for f' (and in turn, for f) in Linear-FIXP and PPAD.

For PPAD-hardness, we simply refer to the reduction in [SSB17b], where it is proved that the weak approximation version of **CDS-CLEARING** is PPAD-complete: Their proof reduces instances of a known PPAD-hard problem into a non-degenerate instance of the weak approximation version of **CDS-CLEARING**, and the latter instance turns out to actually satisfy the dedicated CDS debtor property (where two trivial modifications need to be made to the *amplifier* and *sum* gadgets in the proof in [SSB17b]). This shows PPAD-hardness for computing a weakly approximate clearing vector in a financial system with the dedicated CDS debtor property. This establishes PPAD-hardness of the exact computation version of the problem as well, since weak approximation trivially reduces to exact computation. □

The above PPAD-completeness result (and more precisely the PPAD-membership part of the result), shows that non-degenerate instances with the dedicated CDS debtor property must have polynomial size rational solutions. We use this fact to prove Theorem 11.

Proof of Theorem 11. Consider the graph D that has as its nodes the strongly connected components (SCCs) of $G_{I,\text{aux}}$, and has an arc from a node S to a node T if and only if there exists an arc in $G_{I,\text{aux}}$ that runs from a node in S to a node in T. It is clear that D is a directed acyclic graph.

We will show that we can find a rational clearing vector for $G_{I,\text{aux}}$ by finding rational clearing vectors of the separate SCCs of the system. However, both the assets and the liabilities of the nodes in a given SCC might depend on the contracts from outside the SCC that point into the SCC. Similarly, the liabilities of the nodes in the SCC might depend on arcs pointing from the SCC to external nodes. We may overcome this problem by including the outward-pointing arcs of an SCC into the subinstances that we aim to solve for, and by iterating over the SCCs according to the topological order of D: That is, we first find clearing vectors to the set $\mathcal{S}_1$ of SCCs that have no incoming arc in D. For such SCCs, the assets and liabilities of the nodes are not influenced by external arcs pointing into the SCC. We subsequently find clearing rates for the set of SCCs $\mathcal{S}_2$ that succeed $\mathcal{S}_1$ in the topological order defined by D. In general, we define $\mathcal{S}_j$ inductively as the set of SCCs that directly succeed $\mathcal{S}_{j-1}$ in the topological order defined by D, and we iteratively find clearing rates to the set of SCCs $\mathcal{S}_j$, given the clearing rates computed for $\mathcal{S}_1, \ldots, \mathcal{S}_{j-1}$, until we have obtained a clearing vector covering all nodes in the system. A crucial observation that motivates this approach is that the absence of any weakly switched cycle of $G_{I,\text{aux}}$ causes all SCCs to satisfy the dedicated CDS debtor property, and that therefore the clearing vector computation problem considered in each iteration lies in PPAD. At each iteration, we are thus guaranteed that there are rational recovery rates, and finding them requires solving a PPAD-complete problem.

However, there are some details required to make this approach work, which we will address next.

For an SCC S of $G_{I,\text{aux}}$, define $N'(S)$ as the tricoloured subgraph of $G_{I,\text{aux}}$ formed by the set of all arcs that are going out of the nodes of S (i.e., $N'(S)$ consists of S itself and the set of arcs that point from S to nodes outside of S). The colouring of the arcs is defined to correspond directly to the colouring in $G_{I,\text{aux}}$. We treat $N'(S)$ as a vertex-labeled and arc-labeled graph where the labels represent the notionals c on the contracts and external assets e, according to the specification of I, in the usual way.

Note that $N'(S)$ does not in general define a valid financial subsystem of I, as there may be orange arcs in $N'(S)$ that do not have a corresponding red arc. The reason is that in $G_{I,\text{aux}}$ there may be such red arcs that point into S from outside S, and hence are not included in the subgraph $N'(S)$. Another problem is that $N'(S)$ may have red arcs for which the corresponding orange arc is outside $N'(S)$. We will next modify $N'(S)$ accordingly, where we will furthermore allow ourselves to overwrite the external assets of the nodes with other values, as well as the notionals on the aforementioned set of orange arcs. To that end, let $E_o(S)$ be the set of orange arcs in $N'(S)$ for which there is no corresponding red arc present in $N'(S)$ and let $E_r(S)$ be the set of red arcs in $N'(S)$ for which there is no corresponding orange arc present in $N'(S)$. We now define the subgraph $N(S, (e_i)_{i\in V(N(S))}, (c_e)_{e\in E_o(S)})$ which is obtained from $N'(S)$ by recolouring every orange arc $e \in E_o(S)$ into a blue arc and replacing its notional by c_e, removing every red arc in $E_r(S)$, and by replacing the external assets of every node $i \in V(N(S))$ by e_i.

Under this definition of $N(S, e, c)$, for any choice of non-negative vectors $e = (e_i)_{i\in V(N(S))}$ and $c = (c_e)_{e\in E_o(S)}$, we have the property $N(S)$ represents a valid financial system (i.e., where all orange arcs correspond to a CDS that is entirely contained in $N(S, e, c)$). Also, $N(S, e, c)$, has the dedicated CDS debtor property, which follows directly from $G_{I,\text{aux}}$ not having any weakly switched cycles. Therefore, from Lemma 5 it follows that $N(S, e, c)$ has a rational, polynomially sized clearing recovery rate vector, and finding one is PPAD-complete.

The algorithm by which we can find a rational vector of recovery rates now works as follows:

1. Compute the recovery rates $r_{\mathcal{S}_1}$ for all vertices in the SCCs $\mathcal{S}_1$. This is done by finding the recovery rates of the financial systems $N(S, e, c)$, where $S \in \mathcal{S}_1$ and e and c are defined as in $G_{I,\text{aux}}$.

2. Iterating over j, compute the recovery rates $r_{\mathcal{S}_j}$ given the recovery rates $r_{\mathcal{S}_1}, \ldots, r_{\mathcal{S}_{j-1}}$ for the SCCs preceding $\mathcal{S}_j$ (according to the topological order defined by j). This is done by finding the recovery rates of the financial systems $N(S, e', c')$, where $S \in \mathcal{S}_j$, and the coefficients e' and c' are defined as follows:

   - The external assets e'_i, for $i \in S$, is the sum of the original external assets e_i as defined in instance I, and all the payments that are made to i by nodes in the SCCs preceding S, under the recovery rates $r_{\mathcal{S}_1}, \ldots, r_{\mathcal{S}_{j-1}}$ computed so far.
   - The notional $c'_{i,j}$ on the blue arc $(i, j) \in E_o(S)$ of $N(S, e', c')$ (that is coloured orange in $G'_{I,\text{aux}}$), is set to $(1 - (r_{\mathcal{S}_{j-1}})_R)c^R_{i,j}$. Here R denotes the reference bank of the CDS of I that the orange arc (i, j) of $G_{I,\text{aux}}$ associates to.

Let r be the resulting vector of recovery rates for $G_{I,\text{aux}}$, i.e., r is obtained by combining the recovery rates $r_{\mathcal{S}_1}, r_{\mathcal{S}_2}, \ldots$ computed through the above procedure. It can be proved inductively that r is a clearing vector, by showing that for all nodes i, it holds that $f_i(r) = r$:

For an SCC $S \in \mathcal{S}_1$, all coefficients and arc colours in $N(S, e, c)$ correspond to those in $G_{I,\text{aux}}$. Therefore, for every node in $i \in V(S)$ it holds that $a_i(r)$ under $G_{I,\text{aux}}$ equals $a_i(r_{\mathcal{S}_1})$ under $N(S, e, c)$. Similarly $l_i(r)$ under $G_{I,\text{aux}}$ equals $l_i(r_{\mathcal{S}_1})$ under $N(S, e, c)$. Combining this with the fact that $f_i(r_{\mathcal{S}_1}) = (r_{\mathcal{S}_1})_i$ under $N(S, e, c)$, we conclude that it also holds that $f_i(r) = r$ under $G_{I,\text{aux}}$.

Next, we prove that under $G_{I,\text{aux}}$ it also holds that $f_i(r) = r$ for nodes in SCCs of $\mathcal{S}_j$, under the induction hypothesis that $f_i(r) = r$ for all nodes in SCCs of $\mathcal{S}_1 \cup \mathcal{S}_2 \cup \cdots$. Consider an SCC $S \in \mathcal{S}_j$ and a node $i \in V(S)$. First, we will establish that the payment along any arc of $N(S, e', c')$ under $r_{\mathcal{S}_j}$, equals the payment along the corresponding arc of $G_{I,\text{aux}}$ under r: This holds because (i.) all blue arcs in $N(S, e', c')$ that are not in $E_o(S)$ and are coming into i have the same notional as in $G_{I,\text{aux}}$, so that the payment made through such an arc is equal among both $N(S, e', c')$ and $G_{I,\text{aux}}$; (ii.) all orange arcs in $N(S, e', c')$ that associate to CDSes for which the reference bank is in S similarly have the same notional as in $G_{I,\text{aux}}$, so that the payment made through such an arc is equal among both $N(S, e', c')$ and $G_{I,\text{aux}}$ as well; (iii.) all blue arcs (i, i') in $N(S, e', c')$ that are in $E_o(S)$ are orange arcs under $G_{I,\text{aux}}$, however, the notional on this arc in $N(S, e', c')$ is set to $(1 - (r_{\mathcal{S}_{j-1}})_R)c^R_{i,i'} = (1 - r_R)c^R_{i,i'}$, where R is the reference bank of the CDS of which the orange arc (i, i') is part. Thus, the notional on the blue arc (i, i') under $N(S, e', c')$ equals the liability on the orange arc (i, i') under $G_{I,\text{aux}}$ and r. Therefore, the payment going through the blue arc (i, i') of $N(S, e', c')$ under $r_{\mathcal{S}_j}$ equals the payment going through the orange arc (i, i') of $G_{I,\text{aux}}$ under r.

This correspondence among payments in the two financial systems $N(S, e', c')$ (with recovery rates $r_{\mathcal{S}_j}$ and $G_{I,\text{aux}}$ (with recovery rates r) implies that $a_i(r)$ under $G_{I,\text{aux}}$ equals $a_i(r_{\mathcal{S}_j}$ under $N(S, e', c')$. Similarly $l_i(r)$ under $G_{I,\text{aux}}$ equals $l_i(r_{\mathcal{S}_j}$ under $N(S, e', c')$. Combining this with the fact that $f_i(r_{\mathcal{S}_j}) = (r_{\mathcal{S}_j})_i$ under $N(S, e', c')$, we conclude that it also holds that $f_i(r) = r$ under $G_{I,\text{aux}}$. □

The procedure outlined in the proof of Theorem 11 requires solving a PPAD-complete problem in each iteration, and the number of such iterations is at most linear in the instance size. Since solving each of these problems in PPAD yields a rational solution of size polynomial in the input, one might be tempted to think that the procedure in its entirety is capable of finding a polynomial size rational solution for any financial system that has no weakly switched cycles. Unfortunately, the latter is not true: Observe that in each iteration of the procedure, the PPAD-complete problem instance that is solved, is actually constructed using the rational recovery rate vectors that are computed in the preceding iterations. The coefficients in the PPAD-complete problem instance that is to be solved in any given iteration, are thus polynomially sized in the output recovery rates of the previous iteration. Altogether, this means that the coefficient sizes potentially grow by a polynomial factor in each iteration, and that the final recovery rates output by the procedure are potentially of exponential size.

Indeed, there are examples of financial systems without weakly switched cycles for which the rational clearing vector has recovery rates that require an exponential number of bits to write down. A simple example is obtained by taking some of the gadgets in the reduction used in our FIXP-completeness result (Theorem 3). By taking a duplication gadget (Figure 3.3) followed by a multiplication gadget (Figure 3.5, 3.10, or 3.9) that is connected to the two output nodes of the duplication gadget. We may then take multiple copies of these, and chain them together to form an acyclic financial system. If we now give the first node in the chain (i.e., the input node of the first duplication gadget) some small amount of positive external assets $c < 1$, this acyclic financial system essentially performs a sequence of successive squaring operations on the number c, under the unique clearing vector. The resulting recovery rates on the output nodes of the multiplication gadgets are then doubly exponentially small in magnitude, with respect to the number of squaring repetitions. Thus, the resulting clearing recovery rates require a number of bits that is exponential in the size of the financial system.

If one is willing to discard the complexity issues that arise from working with large-size rational numbers, it is possible to study the procedure in the proof of Theorem 11 in the Blum-Shub-Smale model of computation. Under this computational model, any real number takes one unit of space to store, regardless of its size. Moreover, standard arithmetic operations are assumed to take unit time.[1] The proof of Theorem 11 then implies that when one has oracle access to PPAD, it is possible to find rational clearing vectors in polynomial time under this model of computation. The class of problems polynomial time solvable under the Blum-Shub-Smale model is commonly denoted by $\mathsf{P}_{\mathbb{R}}$. Hence, we obtain the following corollary.

Corollary 4. *The exact computation version of* **CDS-CLEARING***, restricted to instances without weakly switched cycles, is in the complexity class* $\mathsf{P}_{\mathbb{R}}^{\mathsf{PPAD}}$.

[1]For a formal and more accurate definition of the Blum-Shub-Smale model, see the book [BCSS98].

Chapter 6

Hardness of Deciding Priority Profiles

Overview

This chapter considers the concept where a financial regulator determines which priority list each bank should get assigned, and is interested in assigning these in such a way that a specific objective is optimised for. In a financial network, each bank i can be assigned one of outdeg(i)! *Singleton Liability Priority lists*. Consequently the number of candidate priority profiles for a system is exponentially large in terms of its input size. This suggests that selecting from priority profiles to attain a particular objective could pose a computational challenge. We show that this problem is NP-hard for a set of natural choices of objective functions:[1]

1. Maximising the equity of a specific node (cf. Theorem 12);
2. Minimising the number of defaulting nodes (cf. Theorem 13);
3. Minimising the number of not fully paid liabilities (cf. Theorem 13);
4. Minimising the number of activated CDSes in the financial system (cf. Theorem 14);
5. Maximising the liquidity in the financial system (cf. Theorem 15).

[1]The authors in [PW22] establish similar NP-hardness results within the framework of networks with banks paying proportionally where specific financial structures termed as branching gadgets (see Figure 2 of page 5 in [PW22]) are present in the network and generate infinitely many clearing vectors. Their results stem from the task of finding the optimal clearing vector when banks pay proportionally, while our results stem from the task of finding the optimal priority profile.

6.1 Maximising the Equity of a Specific Bank

Theorem 12. *Finding a priority list profile that maximises the equity of a specific bank is NP-hard.*

Proof. We prove the theorem via a reduction from KNAPSACK. Let us be given a knapsack of capacity B and a set $S = \{a_1, \ldots, a_n\}$ of objects, having profit $\text{profit}(a_i)$ and size $\text{size}(a_i)$. Without loss of generality, we assume that $\text{size}(a_i)$, for all $a_i \in S$ as well as B are integer numbers. The aim is to identify a collection of objects that can fit inside the knapsack while maximising the overall profit.

We construct a financial network $\mathcal{F} = (N, e, c)$ as follows. We introduce a node 0 with external assets $e_0 = B$ and for each object $a_j \in S$ we introduce a corresponding node j and let node 0 hold a debt contract towards each node j with notional $c_{0,j} = \text{size}(a_j)$. Moreover we introduce a node τ with $e_\tau = 0$, and a node T with $e_T = 0$, which we refer to as the *terminal node.* We add a debt contract of notional $\text{size}(a_j)$ from each node j to node τ. For each node j we moreover introduce a *j-subnetwork*, consisting of:

- two nodes y_j and x_j with $e_{y_j} = 1, e_{x_j} = 0$,
- a CDS contract (y_j, x_j, j) with notional $c^j_{y_j,x_j} = \max_{a_j \in S}\{size(a_j)\}$,
- a node z_j towards which x_j holds a debt contract of notional $c_{x_j,z_j} = 1$.
- a node k_j with $e_{k_j} = \text{profit}(a_j)$ and an outgoing CDS (k_j, T, x_j) with notional $\text{profit}(a_j)$.

The construction of the j-subnetwork is illustrated in Fig. 6.1.

Assume an optimal solution to the original KNAPSACK instance and let OPT be the set of the objects a_j contained in it. We know that $\sum_{a_i \in \text{OPT}} \text{size}(a_i) \leq B$ and that $\sum_{a_i \in \text{OPT}} \text{profit}(a_i)$, is the maximum profit that can fit in the knapsack. We define the set $N = \{1, \ldots, n\}$, and $N_{\text{OPT}} = \{j \mid a_j \in \text{OPT}\}$ to be the set containing all nodes in $\mathcal{F}$ that correspond to objects contained in the optimal solution. Fix the singleton liability priority list for node i to be $\mathcal{P}_i = (\{N_{\text{OPT}}\} \mid \{N \setminus N_{\text{OPT}}\})$, meaning that i first prioritises all creditors in N_{OPT} in an arbitrary order and afterwards all other creditors in $N \setminus N_{\text{OPT}}$ again in an arbitrary order. Next we prove that under this profile, node T receives its maximum total assets. Observe that $\forall j \in N_{\text{OPT}}$, $p_{0j} = \text{size}(a_j)$ since $\sum_{j \in N_{\text{OPT}}} c_{0j} = \sum_{a_j \in \text{OPT}} \text{size}(a_j) \leq B = e_0$. Since every j node that corresponds to an object $a_j \in \text{OPT}$ receives $\text{size}(a_j)$, it can fully pay node τ, so $r_j = 1$. For all creditors $m \in N \setminus N_{\text{OPT}}$

it holds that $p_{0m} < \text{size}(a_m)$. So, $\forall j \in N_{\text{OPT}} : r_j = 1$ while $\forall m \in N \setminus N_{\text{OPT}} : r_m < 1$. Next we prove that for each $j \in N_{\text{OPT}}$, $p_{k_j,T} = \text{profit}(a_j)$ and for each $m \in N \setminus N_{\text{OPT}}$, $p_{k_m,T} = 0$. Take a $j \in N_{\text{OPT}}$, we know that $r_j = 1$, which implies that the CDS (y_j, x_j, j) is not activated thus $r_{x_j} = 0$ which in turn activates the CDS (k_j, T, x_j) where node k_j pays $\text{profit}(a_j)$ to node T. On the other side, for a $m \in N \setminus N_{\text{OPT}}$, it holds that $r_m < 1$, which means that the CDS (y_m, x_m, m) is activated and generates a liability of $\max_{a_i \in S}\{\text{size}(a_i)\} \cdot (1 - r_m)$ for node y_m. We prove that this liability is at least 1. For an object $a_m \notin \text{OPT}$, r_m indicates the proportion of $\text{size}(a_m)$ that fits in the available knapsack area unoccupied by the objects in OPT. Obviously for $a_m \notin \text{OPT}$, $\text{size}(a_m) > B - \text{size}(\text{OPT})$, otherwise $a_m \in \text{OPT}$ and $r_m \cdot \text{size}(a_m) + \text{size}(\text{OPT}) = B$. Since by assumption B and $\text{size}(a_j)$ for all $a_j \in S$ are integers, it holds that $\forall a_m \notin \text{OPT}$, $r_m \leq (\max_{a_k \in S}\{\text{size}(a_k)\} - 1)/(\max_{a_k \in S}\{\text{size}(a_k)\}$, so the generated liability for y_m is:

$$\begin{aligned} l^m_{y_m,x_m} &= \max_{a_k \in S}\{\text{size}(a_k)\} \cdot (1 - r_m) \\ &\geq \max_{a_k \in S}\{\text{size}(a_k)\} \cdot \left(1 - \frac{\max_{a_k \in S}\{\text{size}(a_k)\} - 1}{\max_{a_k \in S}\{\text{size}(a_k)\}}\right) = 1. \end{aligned}$$

So eventually, $\forall m \in N \setminus N_{\text{OPT}}$, $p_{y_m,x_m} = 1$. Now $r_{x_m} = 1$ thus the CDS (k_m, T, x_m) is not activated meaning that $p_{k_m,T} = 0$. From the above observations we conclude that the equity of T is $\sum_{j \in N_{\text{OPT}}} \text{profit}(a_j)$: node T receives money from all nodes that correspond to objects contained in OPT. We claim that this is the maximum equity node T can receive. If there exist a higher equity for T, then this must be generated from another profile $\mathcal{P}'_i$ that corresponds to a solution to the original KNAPSACK instance with higher profit than the optimal one which is a contradiction.

For the opposite direction assume $\mathcal{P}_0$ to be the profile of 0 that maximises T's equity. Let $A = \{a_j \mid p_{0j} = \text{size}(a_j)\}$ be the set of objects that corresponds to creditor nodes that 0 can fully pay. Obviously A can be computed in polynomial time from $\mathcal{P}_0$. We claim that A is an optimal solution to the original KNAPSACK instance. Assume that there exists another set A' such that $\sum_{a_j \in A'} \text{size}(a_j) \leq B$ and $\sum_{a_j \in A'} \text{profit}(a_j) > \sum_{a_j \in A} \text{profit}(a_j)$. Now node 0 could rearrange its priorities by prioritising all creditors j for which $a_j \in A'$. Doing so, 0 can fully pay all nodes j for which $a_j \in A'$ since $\sum_{a_j \in A'} \text{size}(a_j) \leq B = e_0$ and node T receives $\sum_{a_j \in A'} \text{profit}(a_j) > \sum_{a_j \in A} \text{profit}(a_j)$, a contradiction to the original assumption that $\sum_{a_j \in A} \text{profit}(a_j)$ is the maximum equity for node T. □ □

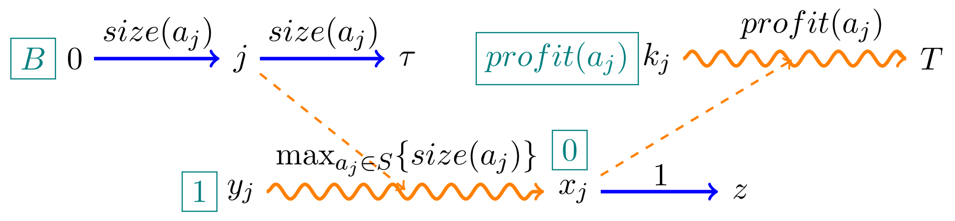


Figure 6.1: The j-subnetwork used in the proof of Theorem 12.

6.2 Minimising Defaulting Banks and Partially Paid Liabilities

Theorem 13. *Finding a priority list profile that minimises the number of defaulting banks/ the number of not fully paid liabilities is* NP*-hard.*

Proof. We prove both statements of the theorem via a reduction from the satisfiability problem (SAT), where we are given a boolean formula in conjunctive normal form, and have to determine whether there is a truth assignment to the variables that renders the formula true. Let $F = \bigwedge_{i=1}^{n} C_i$ be a SAT instance, where $C_1, \ldots, C_n$ are the clauses, and let V_F be the set of all variables that appear in F. We create a financial network from F as follows. For each variable $x \in V_F$ we construct a gadget that is refereed to as the x-subnetwork. Each x-subnetwork consists of four nodes, labeled as $i_x, x, \neg x, j_x$, where $e_{i_x} = 1$ and $e_x = e_{\neg x} = j_x = 0$, and of four debt contracts, $\mathcal{DC} = \{(i_x, x), (i_x, \neg x), (x, j_x), (\neg x, j_x)\}$ all with contract notionals equal to one. Moreover, the constructed financial network has n further nodes, labeled as $C_1, \ldots, C_n$, that correspond to each clause in F with $e_{C_1} = \cdots = e_{C_n} = 0$, and one terminal node labeled as τ, towards which each C_i holds a debt contract of notional one, i.e $c_{C_i,\tau} = 1$. Finally for each variable x of F, we construct two nodes labeled as k_x and $k_{\neg x}$ respectively, where e_{k_x} and $e_{k_{\neg x}}$ are equal to the number of occurrences of literals x and $\neg x$ in F, respectively. Whenever a literal l belongs to a clause C_i we construct the CDS $(k_l, C_i, \neg l)$ with $c^{\neg l}_{k_l, C_i} = 1$. An example of a network induced from $F = (x \vee y) \wedge (y \vee \neg y)$ is given in Fig. 6.2.

We now map a truth assignment $\mathrm{T} : V_F \mapsto \{\text{true}, \text{false}\}$ to a priority list profile $\mathcal{P}_{\mathrm{T}}$ as follows. If $\mathrm{T}(x) = \text{true}$, node i_x prioritises node x, and otherwise it prioritises node $\neg x$. All k_l nodes posses enough external assets to fully pay their debts under any priority list, so we can take an arbitrary list for those nodes, and all remaining nodes have at most one liability. Conversely, from a priority list profile $\mathcal{P}$ we induce the truth assignment $\mathrm{T}_{\mathcal{P}}$ as follows: if i_x prioritises x then $\mathrm{T}(x) = \text{true}$

otherwise $\mathrm{T}(x) = \text{false}$.

Let $\mathrm{T} : V_F \mapsto \{\text{true}, \text{false}\}$ be any truth assignment and consider the priority list profile $\mathcal{P}_\mathrm{T}$. In each x-subnetwork, node i_x can fully pay only its first priority, thus in each x-subnetwork there exist two defaulting nodes regardless of the choice of priority list of i_x, meaning that the minimum number of defaulting nodes in the induced financial system is $2|V_F|$. Similarly, each x-subnetwork has two not fully paid liabilities regardless of the choice of priority list of i_x. The only additional defaulting nodes might be the nodes $C_1, \ldots, C_n$, and the only additional not fully paid liabilities might be on the n debt contracts in which one of $C_1, \ldots, C_n$ is the debtor. Let us inspect which of the latter set of nodes are defaulting and which of the latter liabilities are not fully paid under $\mathcal{P}_\mathrm{T}$. If clause C_i is a clause that is not satisfied under T, then none of the CDSes involving node C_i are activated, and node C_i does not have any assets $\mathcal{P}_\mathrm{T}$. Since C_i has to pay 1 to τ, node C_i will be in default, and C_i's liability of 1 will not be paid. If clause C_i is a clause that is satisfied under T, then at least one CDS involving node C_i is activated, and since the reference bank in this CDS has recovery rate 0, node C_i will receive the CDS's full notional of 1, with which it can fully pay its liability of 1. Hence, in the latter case, node C_i is not in default.

Hence, for a truth assignment T, under $\mathcal{P}_\mathrm{T}$, the number of banks in default and the number of not fully paid liabilities are both equal to $2|V_F|$ plus the number of unsatisfied clauses. Since we argued above that restricting to the profiles

$$\{\mathcal{P}_\mathrm{T} \mid \mathrm{T} \text{ is a truth-assignment for } F\}$$

is without loss of generality, from finding the profile of priority lists minimising the number of defaulting banks or minimising the number of not fully paid liabilities in the constructed financial network, one can infer whether the formula F is satisfiable, which proves our claim. □

$F = \{x \vee \neg y\} \wedge \{y \vee \neg y\}$

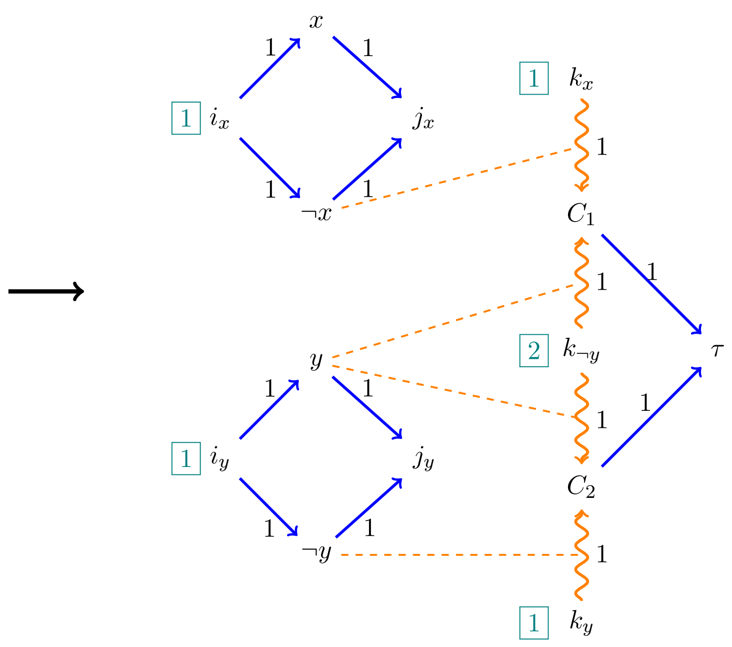


Figure 6.2: The financial system corresponding to the formula $F = \{x \vee \neg y\} \wedge \{y \vee \neg y\}$. This construction is used to prove the statements of Theorem 13.

6.3 Minimising Activated CDS Contracts

Theorem 14. *Finding a priority list profile that minimises the number of activated CDSes is* NP*-hard.*

Proof. We reduce from the problem of minimising the number of defaulting nodes, proved NP-hard in Theorem 13. Let $\mathcal{F}$ be a financial system and let m be the maximum number of CDS contracts that are issued upon the same reference bank. Consequently, there exists a node that is the reference bank in m CDSes and no other node is reference bank to more CDSes than that.

We construct a financial network $\mathcal{F}'$ by expanding $\mathcal{F}$ in the following way. For each node i appearing in $\mathcal{F}$, we add a number n_i of CDS contracts such that i is reference bank to exactly m CDSes. These contracts are of the form $(\alpha_i^1, \beta_i^1, i), \ldots, (\alpha_i^{n_i}, \beta_i^{n_i}, i)$, with notionals $c^i_{\alpha_i,\beta_i} = 1$, where α_i^j, β_i^j for $j \in [n_i]$, are newly introduced nodes with external assets $e_{\alpha_i} = 1$. We call the new contracts dummy CDSes. Eventually all nodes in $\mathcal{F}'$ are reference banks to exactly m CDSes. Fig. 6.3 shows an example of this expansion.

By this construction, it is straightforward to see that if under any priority list profile a set of banks default in $\mathcal{F}$, then in $\mathcal{F}'$, for each of these banks i, exactly m distinct CDSes activate in which i is the reference bank. Since the priority lists of $\mathcal{F}$ and $\mathcal{F}'$ are in one-to-one correspondence with each other, we conclude that finding the priority list profile minimising the number of activated CDSes in $\mathcal{F}'$ is equivalent to finding the priority list profile minimising the number of defaulting banks in $\mathcal{F}$. □ □

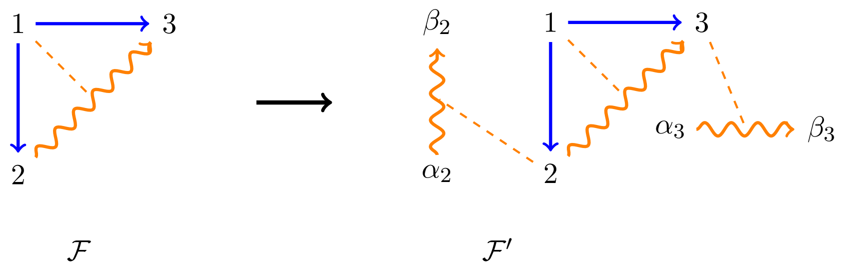


Figure 6.3: The initial financial system $\mathcal{F}$ and the constructed $\mathcal{F}'$ used is the proof of Theorem 14.

6.4 Maximising Systemic Liquidity

Systemic Liquidity. In [KKZ21b], the authors define the term *Systemic Liquidity* as the total amount of payments that are being transacted among the economic firms in the financial system under some clearing recovery rate vector. Given a financial system $\mathcal{F}$ we use the notation $\mathcal{L}^{\mathcal{P}}_{\mathcal{F}}(r)$ to denote the Systemic Liquidity of the system under the priority profile $\mathcal{P}$ and an assuming clearing recovery rate vector r and is defined as $\mathcal{L}^{\mathcal{P}}_{\mathcal{F}}(r) = \sum_{i\in N}\sum_{j\in N} p_{i,j}(r)$.

Theorem 15. *Finding a priority list profile that maximises the Systemic Liquidity is* NP*-hard.*

Proof. We prove the theorem by a reduction from the problem in Theorem 13. Given a financial system $\mathcal{F}$ we construct a modified financial system $\mathcal{F}'$ according to the following procedure:

- For each debt contract $(i, j) \in \mathcal{DC}_{\mathcal{F}}$ with contract notional $c_{i,j}$: Erase the debt contract (i, j). Add a new node denoted as τ_{ij} with $e_{\tau_{ij}} = 0$ and two new debt contracts (i, τ_{ij}) and $(\tau_{ij,}j)$ each with contract notional $c_{i,\tau_{ij}} = c_{\tau_{ij},j} = c_{i,j}$. Finally add two new nodes α_{ij} and β_{ij} with $e_{\alpha_{ij}} = 2c_{i,j}$ and construct the CDS contract $(\alpha_{ij}, \beta_{ij}, \tau_{ij})$ with contract notional $c^{\tau_{ij}}_{\alpha_{ij},\beta_{ij}} = c_{i,j}$. We refer to this construction as the τ_{ij}-gadget. It is illustrated in Fig. 6.4.

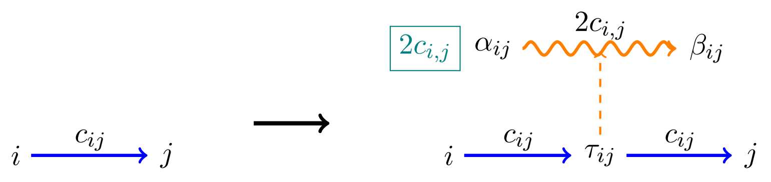


Figure 6.4: Transformation of an (i, j) contract that appears in $\mathcal{F}$ to a τ_{ij}-gadget appearing in $\mathcal{F}'_\tau$ and $\mathcal{F}'$.

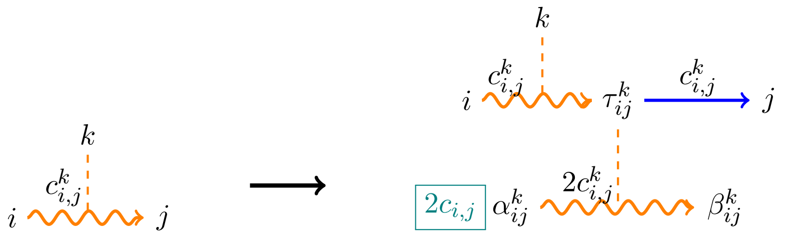


Figure 6.5: Transformation of an (i, j, k) contract that appears in $\mathcal{F}$ to a τ^k_{ij}-gadget appearing in $\mathcal{F}'_\tau$ and $\mathcal{F}'$.

- For each CDS contract $(i, j, k) \in \mathcal{CDS}_\mathcal{F}$ with contract notional $c^k_{i,j}$: Erase the CDS (i, j, k). Add a new node denoted as τ^k_{ij} with $e_{\tau^k_{ij}} = 0$ and construct the CDS contract (i, τ^k_{ij}, k) with contract notional $c^k_{i,\tau^k_{ij}} = c^k_{ij}$ and the debt contract (τ^k_{ij}, j) with contract notional $c_{\tau_{ij},j} = c^k_{i,j}$. Finally we add two new nodes α^k_{ij} and β^k_{ij} with $e_{\alpha^k_{ij}} = 2c^k_{i,j}$ and construct the CDS $(\alpha^k_{ij}, \beta^k_{ij}, \tau^k_{ij})$ with contract notional $c^{\tau^k_{ij}}_{\alpha^k_{ij},\beta^k_{ij}} = 2 \cdot c^k_{ij}$. We refer to this construction as the τ^k_{ij}-gadget. It is illustrated in Fig. 6.5.

- For each node $k \in N_\mathcal{F}$: We add five new nodes $\chi_k, \psi_k, \zeta_k, \alpha_k, \beta_k$ with $e_{\chi_k} = 1/(2 \mid N_\mathcal{F} \mid)$, $e_{\alpha_k} = 2$ and construct the CDS (χ_k, ψ_k, k) with $c^k_{\chi_k,\psi_k} = \infty^2$, the debt contract (ψ_k, ζ_k) with $c_{\psi_k,\zeta_k} = 1/(2 \mid N_\mathcal{F} \mid)$ and the CDS $(\alpha_k, \beta_k, \psi_k)$ with $c^{\psi_k}_{\alpha_k,\beta_k} = 2$. We refer to this construction as k-gadget. It is illustrated in Fig. 6.6.

For any priority profile $\mathcal{P}$ of $\mathcal{F}$ we define its 'natural extension' priority profile denoted as $\mathcal{P}^{\text{ext}}$ of $\mathcal{F}'$ as follows: For a node $i \in N_\mathcal{F}$ having a singleton liability priority list $\mathcal{P}_i$ we substitute each priority that corresponds to some debt contract $(i, j) \in \mathcal{DC}_\mathcal{F}$ with the debt contract $(i, \tau_{ij}) \in \mathcal{DC}_{\mathcal{F}'}$ and each priority that corresponds to some CDS contract $(i, j, k) \in \mathcal{CDS}_\mathcal{F}$ with the CDS contract

[2]Here we assume that a contract may have ∞ notional in the sense that whenever the debtor defaults it must submit all of its remaining assets through this contract. A similar assumption is made in [PW22, PW21b].

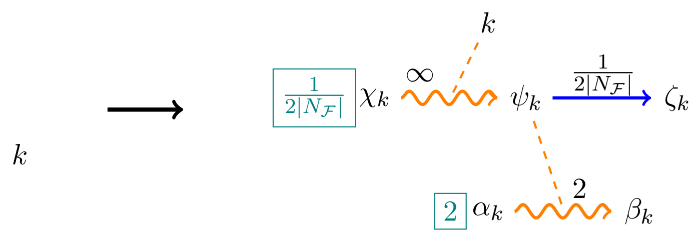


Figure 6.6: Addition of the k-gadget in $\mathcal{F}'$ for every node k of $\mathcal{F}$.

$(i, \tau_{ij}^k, k) \in \mathcal{CDS}_{\mathcal{F}'}$. For all other nodes appearing in the constructed gadgets, the priority lists are uniquely defined since they have only one outgoing edge. For example, if some node i pays in $\mathcal{F}$ according to $\mathcal{P}_i = ((i,j) \mid (i,j,k))$, then it pays in $\mathcal{F}'$ according to $\mathcal{P}_i^{\text{ext}} = ((i, \tau_{ij}) \mid (i, \tau_{ij}^k, k))$. Next we prove three useful claims that we use in our reduction.

Claim 4. *The 'natural extension' priority profile does not affect the recovery rate of nodes belonging in $N_{\mathcal{F}} \cap N_{\mathcal{F}'}$. Namely, if node i has recovery rate r_i under $\mathcal{P}$ in $\mathcal{F}$, then it has recovery rate r_i under $\mathcal{P}^{ext}$ in $\mathcal{F}'$.*

Proof. The amount of money transferred via a liability (i,j) in $\mathcal{F}$ under $\mathcal{P}$ are transferred from τ_{ij} to node j under $\mathcal{P}^{\text{ext}}$ in $\mathcal{F}'$, since node τ_{ij} has an incoming payment of at most $c_{i,j}$, a liability of $c_{i,j}$ and zero external assets. Similarly the amount of money that are being transferred via a CDS contract (i,j,k) in $\mathcal{F}$ under $\mathcal{P}$ are transferred from τ_{ij}^k to node j in $\mathcal{F}'$ under $\mathcal{P}^{\text{ext}}$ because τ_{ij}^k receives a payment of at most $(1 - r_k) \cdot c_{i,j}^k$, has a liability of $c_{i,j}^k$ and zero external assets. Also by the way we constructed $\mathcal{F}'$ the liabilities of all nodes appearing in $\mathcal{F}$ are unchanged. That means that the 'natural extension' profile $\mathcal{P}^{\text{ext}}$ of a profile $\mathcal{P}$ does not affect the assets and liabilities of nodes in $N_{\mathcal{F}} \cap N_{\mathcal{F}'}$, thus their recovery rate is the same. □ □

Next we denote by $\mathcal{F}'_\tau$ the financial system that is constructed from $\mathcal{F}$ only by adding the τ_{ij}-gadget and τ_{ij}^k-gadget. We denote by $\mathcal{P}_\tau$ the 'natural extension' of any priority profile $\mathcal{P}$ of $\mathcal{F}$ in $\mathcal{F}_\tau$. In the following claim we prove that the liquidity of $\mathcal{F}'_\tau$ is the same under any priority profile and any clearing recovery rate vector.

Claim 5. *For each priority profile $\mathcal{P}$ and each clearing recovery rate vector r, it holds that $\mathcal{L}^{\mathcal{P}}_{\mathcal{F}'_\tau}(r) = 2 \cdot \left(\sum_{i,j \in N_{\mathcal{F}}} c_{i,j} + \sum_{i,j,k \in N_{\mathcal{F}}} c_{i,j}^k \right)$.*

Proof. Assume a priority profile $\mathcal{P}$ of $\mathcal{F}$ and let $\mathcal{P}_\tau$ be its 'natural extension' profile in $\mathcal{F}_\tau$. First we will prove that each $(i,j) \in \mathcal{DC}_\mathcal{F}$ generates liquidity of $2 \cdot c_{i,j}$ in $(\mathcal{F}_\tau, \mathcal{P}_\tau)$. Assume $l \leq c_{i,j}$ to be the amount of money transferred via some debt contract (i,j) in $(\mathcal{F}, \mathcal{P})$. We distinguish two cases.

1. $l < c_{i,j}$. The recovery rate for node τ_{ij} is $r_{\tau_{ij}} = l/c_{i,j} < 1$. Thus the CDS $(\alpha_{ij}, \beta_{ij}, \tau_{ij})$ is activated and node α_{ij} owes $2 \cdot (1 - r_{\tau_{ij}}) \cdot c_{i,j} = 2 \cdot c_{i,j} - 2 \cdot l$ to node β_{ij} which can fully pay off. Edges (i, τ_{ij}) and (τ_{ij}, j) generate a liquidity of $2 \cdot l$ thus the generated liquidity in that case is $2 \cdot c_{i,j} - 2 \cdot l + 2 \cdot l = 2 \cdot c_{i,j}$.

2. $l = c_{i,j}$. The recovery rate of node τ_{ij} is $r_{\tau_{ij}} = 1$. Thus the $(\alpha_{ij}, \beta_{ij}, \tau_{ij})$ is not activated. The generated liquidity in that case arises only from edges (i, τ_{ij}) and (τ_{ij}, j) and is equal to $2c_{i,j}$.

Next we will prove that any CDS contract $(i,j,k) \in \mathcal{CDS}_\mathcal{F}$ generates a liquidity of $2 \cdot c^k_{i,j}$ in $(\mathcal{F}_\tau, \mathcal{P}_\tau)$. Assume a clearing recovery rate vector r and let $l \leq (1 - r_k) \cdot c^k_{i,j}$ to be the amount of money transferred via some CDS (i,j,k) in $(\mathcal{F}, \mathcal{P})$. We distinguish three cases.

1. $l = (1 - r_k) \cdot c^k_{i,j}$. The recovery rate for node τ_{ij} is $r_{\tau_{ij}} = l/c^k_{i,j} = 1 - r_k < 1$. Thus the CDS $(\alpha^k_{ij}, \beta^k_{ij}, \tau^k_{ij})$ is activated and node α^k_{ij} owes $2 \cdot (1 - r_{\tau^k_{ij}}) \cdot c^k_{i,j} = 2 \cdot r_k \cdot c^k_{i,j}$, which can fully pay off. Since edges $(i, \tau_{i,j})$ and (τ_{ij}, j) generate a liquidity of $2 \cdot (1 - r_k) \cdot c^k_{i,j}$ the total generated liquidity is $2 \cdot r_k \cdot c^k_{i,j} + 2 \cdot c^k_{i,j} - 2 \cdot r_k \cdot c^k_{i,j} = 2 \cdot c^k_{i,j}$.

2. $l < (1 - r_k) \cdot c^k_{i,j}$. Again $r_{\tau_{i,j}} = l/c^k_{i,j} < 1$. Thus $(\alpha^k_{ij}, \beta^k_{ij}, \tau^k_{ij})$ is activated and α^k_{ij} owes $2 \cdot c^k_{i,j} \cdot (1 - (l/c^k_{i,j})) = 2 \cdot c^k_{i,j} - 2 \cdot l$ to β^k_{ij}, which can fully pay. The total generated liquidity is $2 \cdot l + 2 \cdot c^k_{i,j} - 2 \cdot l = 2 \cdot c^k_{i,j}$.

3. $r_k = 1$. From Claim 4, we know that the recovery rate of any node does not change under $\mathcal{P}^{\text{ext}}$. It is not hard to see that the same holds under $\mathcal{P}_\tau$. If $r_k = 1$ then $r_{\tau^k_{ij}} = 0$ and the (i, τ^k_{ij}, k) is inactive. So node α^k_{ij} owes $2 \cdot c^k_{i,j}$ to β^k_{ij}, which can fully pay and the generated liquidity is $2 \cdot c^k_{i,j}$.

□

Notice that $\mathcal{F}'$ is actually $\mathcal{F}'_\tau$ with the addition of the k-gadgets, and since we proved that the systemic liquidity of $\mathcal{F}'_\tau$ is always the same under any priority profile it must be the case that the liquidity if $\mathcal{F}'$ depends on the generated liquidity in the k-gadgets.

Claim 6. *The generated liquidity from a k-gadget under any clearing recovery rate vector is* $1/(\mid N_{\mathcal{F}} \mid)$ *whenever* $r_k < 1$ *and* 2 *whenever* $r_k = 1$.

Proof. Assume some node k and let a clearing recovery rate vector with $r_k = 1$ in $\mathcal{F}$ under some profile $\mathcal{P}$. From Claim 4 we get that $r_k = 1$ in $\mathcal{F}'$ under $\mathcal{P}^{\text{ext}}$. Thus the CDS (χ_k, ψ_k, k) is inactive which means that $r_{\psi_k} = 0$ and node α_k owes 2 to node β_k which it can fully pay off and the generated liquidity in that case is 2. If $r_k < 1$ then the CDS (χ_k, ψ_k, k) is activated and since $c^k_{\chi_k,\psi_k} = \infty$ node χ_k must submit all of its assets to ψ_k. Now ψ_k can fully pay off ζ_k thus the $(\alpha_k, \beta_k, \psi_k)$ is inactive and the generated liquidity is $1/ \mid N_{\mathcal{F}} \mid$. □ □

Next we provide the reduction based on the above three claims. Assume $\mathcal{P}$ is the priority profile of $\mathcal{F}$ under which the number of defaulting nodes is minimised and let λ be that number. We construct the financial system $\mathcal{F}'$ as described above and consider the priority profile $\mathcal{P}^{\text{ext}}$. We will prove that under $\mathcal{P}^{\text{ext}}$ the liquidity in $\mathcal{F}'$ is maximised. From Claim 4 we know that the recovery rate of each node in $\mathcal{F}$ under $\mathcal{P}^{\text{ext}}$ is unchanged thus if a node is in default in $\mathcal{F}$ under $\mathcal{P}$ for some clearing recovery rate vector then it is in default in $\mathcal{F}'$ under $\mathcal{P}^{\text{ext}}$. That means that there exist exactly λ k-gadgets, which by Claim 6 generate liquidity $\lambda/ \mid N_{\mathcal{F}} \mid$, which (taking into consideration Claim 5) means that

$$\mathcal{L}^{\mathcal{P}^{\text{ext}}}_{\mathcal{F}'}(r) = 2 \cdot \left(\sum_{i,j \in N_{\mathcal{F}}} c_{i,j} + \sum_{i,j,k \in N_{\mathcal{F}}} c^k_{i,j} \right) + \frac{\lambda}{\mid N_{\mathcal{F}} \mid} + 2 \cdot (\mid N_{\mathcal{F}} \mid - \lambda).$$

Assume there exists another profile $\mathcal{Q}^{\text{ext}}$ such that $\mathcal{L}^{\mathcal{Q}^{\text{ext}}}_{\mathcal{F}'}(r) > \mathcal{L}^{\mathcal{P}^{\text{ext}}}_{\mathcal{F}'}(r)$. This means that there must exist another λ' such that

$$2 \cdot \left(\sum_{i,j \in N_{\mathcal{F}}} c_{i,j} + \sum_{i,j,k \in N_{\mathcal{F}}} c^k_{i,j} \right) + \frac{\lambda'}{\mid N_{\mathcal{F}} \mid} + 2 \cdot (\mid N_{\mathcal{F}} \mid - \lambda') >$$
$$2 \cdot \left(\sum_{i,j \in N_{\mathcal{F}}} c_{i,j} + \sum_{i,j,k \in N_{\mathcal{F}}} c^k_{i,j} \right) + \frac{\lambda}{\mid N_{\mathcal{F}} \mid} + 2 \cdot (\mid N_{\mathcal{F}} \mid - \lambda),$$

which is true if and only if $\lambda' < \lambda$. This means that the natural extension of the priority profile that minimises the number of defaulting nodes in $\mathcal{F}$ maximises the systemic liquidity in $\mathcal{F}'$ and vice versa. □ □

Chapter 7

Conclusion & Future Work

This thesis addresses research questions related to the study of systemic risk in financial networks containing simple debt contracts as well as more involved contracts called credit default swaps, a widely used and potentially disruptive class of derivatives. The primary result established in this work is settling the computational complexity of computing strong approximations (Section 2.2.3) of a bank's exposure to systemic risk, arguably an approximation notion of intrinsic interest to the financial industry.

Specifically, we defined and studied the problem **CDS-CLEARING** as well as an interesting byproduct of this problem that addresses priority payments among firms called **CDS-PRIORITY-CLEARING**. The main task for **CDS-CLEARING** is to compute a clearing recovery rate vector (see Definition 2), i.e, vectors that represent the actual proportion of liabilities that the banks can pay. The main presented result (Theorem 3) establishes that **CDS-CLEARING** is FIXP-complete and its strong approximation version is FIXP_a-complete. The methodology we followed in our proof follows the methodology that is presented in [EY10]. The main challenge was to simulate the algebraic circuit that computes the fixed points of a continuous function f in the domain $[0,1]^n$ that is constructed from the gates $\{\max, +, *\}$ and rational constants with a financial system comprising of debt and CDS contracts. The basis of our presented construction is to simulate the algebraic gates in the operator base $\{\max, +, *\}$ with special financial gadgets $g_+, g_*, g_{\max}$ (Section 3.1.3) whose configuration is structured so as to simulate their corresponding operation on recovery rate values.

Subsequently we defined and studied the problem ϵ-**CDS-CLEARING**, which stands for the version of the problem that is associated with the task of finding a weak approximate clearing vector alterantively termed as ϵ-solution (see Section 2.2.3). Previous work on this version of the problem published by [SSB17b] established that computing ϵ weak approximate solutions is PPAD-hard for some constant $\epsilon > 0$. Our results suggested an improved and explicit inapproximability bound for parameter ϵ that is approximately equal to 0.048 and stems out from directly reducing ϵ-**CDS-CLEARING** from the problem **PURE-CIRCUIT** introduced and proved PPAD-complete in [DFHM22].

In Chapter 4 we proposed an optimisation framework for computing exact solutions to instances of **CDS-CLEARING** that satisfy a certain condition called *central CDS debtors.* Specifically for a given financial network we devised a Mixed Binary Linear program whose feasible solutions correspond to exact clearing vectors of the given network. As a direct consequence of this program we get an exponential time algorithm for computing exact clearing vectors for instances that satisfy the *central CDS debtor* property. When coupling the *central CDS debtor* property with covered CDS, a certain kind of credit default swaps introduced in [SSB20], we show that there exist a polynomial time algorithm for computing exact clearing recovery rate vectors.

In Chapter 5, we presented results regarding the rational and irrational solutions for general instances of the examined problem. Specifically we established structural conditions that are sufficient for irrational/rational solutions to emerge. Finally in the last Chapter, we studied the framework where a financial regulator can determine priority payments for the banks in a financial system and established a set of NP-hardness results that stem out from the task of finding the optimal priority list for each bank so as to satisfy certain objective functions of interest tied to the financial network.

The presented work contributes in future theoretical studies on connections among financial systems and computation, as well as in constructing empirical and heuristic-based methods oriented towards financial applications. We propose a set of future research directions below.

- Whether the bound of Theorem 6 is tight is the main open question. Our intuition is that for higher bounds the financial gate analysis should induce less errors. This is achieved by reducing the number of CDS contracts in the presented reduction, which in our opinion is challenging.

- In light of the growing interest in exploring the smoothed complexity of problems associated with the class PPAD, another research direction could be to examine the smooth complexity of **CDS-CLEARING**.

- In absence of pure algorithmic methods, the empirical investigation of the problem can be initiated by referring to the presented programs in Chapter 5. Work in this line of research could contribute in clearing mechanisms and heuristic methods for monitoring systemic risk in complex financial networks as well as providing a better understanding of the problem.
- Finally an interesting challenge would be devising clearing algorithms based on reinforcement learning methods. This line of research would captivate the interest of both the finance and machine learning community.

Bibliography

[AOTS15] Daron Acemoglu, Asuman Ozdaglar, and Alireza Tahbaz-Salehi. Systemic risk and stability in financial networks. *American Economic Review*, 105(2):564–608, 2015.

[BCSS98] Lenore Blum, Felipe Cucker, Michael Shub, and Steve Smale. *Complexity and real computation*. Springer Science & Business Media, 1998.

[BFHT85] A. Borodin, R. Fagin, J. E. Hopcroft, and M. Tompa. Decreasing the nesting depth of expressions involving square roots. *Journal of Symbolic Computation*, 1(2):169–188, 1985.

[BHH21] Eleni Batziou, Kristoffer Arnsfelt Hansen, and Kasper Høgh. Strong approximate consensus halving and the borsuk-ulam theorem. In *ICALP*, volume 198 of *LIPIcs*, pages 24:1–24:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[BHS20] Nils Bertschinger, Martin Hoefer, and Daniel Schmand. Strategic payments in financial networks. In Thomas Vidick, editor, *11th Innovations in Theoretical Computer Science Conference, ITCS 2020, January 12-14, 2020, Seattle, Washington, USA*, volume 151 of *LIPIcs*, pages 46:1–46:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[BMS81] A. Bertoni, G. Mauri, and N. Sabadini. A characterization of the class of functions computable in polynomial time on random access machines. In *Proceedings of the Thirteenth Annual ACM Symposium on Theory of Computing (STOC 1981)*, pages 168–176. Association for Computing Machinery, 1981.

[CDT09] Xi Chen, Xiaotie Deng, and Shang-Hua Teng. Settling the complexity of computing two-player Nash equilibria. *J. ACM*, 56(3):14:1–14:57, 2009.

[CFS05] Rodrigo Cifuentes, Gianluigi Ferrucci, and Hyun Song Shin. Liquidity risk and contagion. *Journal of the European Economic association*, 3(2-3):556–566, 2005.

[DFHM22] Argyrios Deligkas, John Fearnley, Alexandros Hollender, and Themistoklis Melissourgos. Pure-circuit: Strong inapproximability for PPAD. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, Denver, CO, USA, October 31 - November 3, 2022*, pages 159–170. IEEE, 2022.

[DFHM23] Argyrios Deligkas, John Fearnley, Alexandros Hollender, and Themistoklis Melissourgos. Tight inapproximability for graphical games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 5600–5607, 2023.

[DGP09] Constantinos Daskalakis, Paul W Goldberg, and Christos H Papadimitriou. The complexity of computing a Nash equilibrium. *Communications of the ACM*, 52(2):89–97, 2009.

[DH24] Jérémi Do Dinh and Alexandros Hollender. Tight inapproximability of nash equilibria in public goods games. *CoRR*, abs/2402.14198, 2024.

[EGJ14] Matthew Elliott, Benjamin Golub, and Matthew O Jackson. Financial networks and contagion. *American Economic Review*, 104(10):3115–53, 2014.

[EHMS14] Kousha Etessami, Kristoffer Arnsfelt Hansen, Peter Bro Miltersen, and Troels Bjerre Sørensen. The complexity of approximating a trembling hand perfect equilibrium of a multi-player game in strategic form. In *Algorithmic Game Theory: 7th International Symposium, SAGT 2014, Haifa, Israel, September 30–October 2, 2014. Proceedings 7*, pages 231–243. Springer, 2014.

[EN01] Larry Eisenberg and Thomas H Noe. Systemic risk in financial systems. *Management Science*, 47(2):236–249, 2001.

[EUB] Euro-parliament bans 'naked' credit default swaps. EUBusiness. `https://www.eubusiness.com/news-eu/finance-economy-cds.dij`.

[EY10] Kousha Etessami and Mihalis Yannakakis. On the complexity of Nash equilibria and other fixed points. *SIAM J. Comput.*, 39(6):2531–2597, 2010.

[FGH$^+$23] Aris Filos-Ratsikas, Yiannis Giannakopoulos, Alexandros Hollender, Philip Lazos, and Diogo Poças. On the complexity of equilibrium computation in first-price auctions. *SIAM J. Comput.*, 52(1):80–131, 2023.

[FGHK24] Aris Filos-Ratsikas, Yiannis Giannakopoulos, Alexandros Hollender, and Charalampos Kokkalis. On the computation of equilibria in discrete first-price auctions. *CoRR*, abs/2402.12068, 2024.

[FHHH21] Aris Filos-Ratsikas, Kristoffer Arnsfelt Hansen, Kasper Høgh, and Alexandros Hollender. FIXP-membership via convex optimization: Games, cakes, and markets. In *FOCS*, pages 827–838. IEEE, 2021.

[Fit90] Melvin Fitting. *First-Order Logic and Automated Theorem Proving.* Texts and Monographs in Computer Science. Springer, 1990.

[GH21] Paul W. Goldberg and Alexandros Hollender. The hairy ball problem is PPAD-complete. *J. Comput. Syst. Sci.*, 122:34–62, 2021.

[GY15] Paul Glasserman and H Peyton Young. How likely is contagion in financial networks? *Journal of Banking & Finance*, 50:383–399, 2015.

[HK12] Sebastian Heise and Reimer Kühn. Derivatives and credit contagion in interconnected networks. *The European Physical Journal B*, 85(4):1–19, 2012.

[HK16] Brett Hemenway and Sanjeev Khanna. Sensitivity and computational complexity in financial networks. *Algorithmic Finance*, 5(3-4):95–110, 2016.

[HL18] Kristoffer Arnsfelt Hansen and Troels Bjerre Lund. Computational complexity of proper equilibrium. In *EC*, pages 113–130. ACM, 2018.

[HL21] Kristoffer Arnsfelt Hansen and Troels Bjerre Lund. Computational complexity of computing a quasi-proper equilibrium. In *International Symposium on Fundamentals of Computation Theory*, pages 259–271. Springer, 2021.

[HW22] Martin Hoefer and Lisa Wilhelmi. Seniorities and minimal clearing in financial network games. In *SAGT*, volume 13584 of *Lecture Notes in Computer Science*, pages 187–204. Springer, 2022.

[HZHW12] Daning Hu, J Leon Zhao, Zhimin Hua, and Michael CS Wong. Network-based modeling and analysis of systemic risk in banking systems. *MIS quarterly*, pages 1269–1291, 2012.

[IDKV22] Stavros D Ioannidis, Bart De Keijzer, and Carmine Ventre. Strong approximations and irrationality in financial networks with derivatives. In *49th International Colloquium on Automata, Languages, and Programming (ICALP 2022)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2022.

[IdKV23a] Stavros D. Ioannidis, Bart de Keijzer, and Carmine Ventre. Clearing financial networks with derivatives: From intractability to algorithms. *CoRR*, abs/2312.05139, 2023.

[IDKV23b] Stavros D Ioannidis, Bart De Keijzer, and Carmine Ventre. Financial networks with singleton liability priorities. *Theoretical Computer Science*, 963:113965, 2023.

[IMF] Lasting effects: The global economic recovery 10 years after the crisis. IMFBlogs. `https://blogs.imf.org/2018/10/03/lasting-effects-the-global-economic-recovery-10-years-after-the-crisis/`.

[JP21] Matthew O Jackson and Agathe Pernoud. Systemic risk in financial networks: A survey. *Annual Review of Economics*, 13:171–202, 2021.

[JP24] Matthew O Jackson and Agathe Pernoud. Credit freezes, equilibrium multiplicity, and optimal bailouts in financial networks. *The Review of Financial Studies*, page hhad096, 2024.

[KKZ21a] Panagiotis Kanellopoulos, Maria Kyropoulou, and Hao Zhou. Financial network games. In *Proceedings of the Second ACM International Conference on AI in Finance*, pages 1–9, 2021.

[KKZ21b] Panagiotis Kanellopoulos, Maria Kyropoulou, and Hao Zhou. Financial network games. In *ICAIF*, pages 26:1–26:9. Association for Computing Machinery, 2021.

[KKZ22] Panagiotis Kanellopoulos, Maria Kyropoulou, and Hao Zhou. Forgiving debt in financial network games. In *AAMAS*, pages 1651–1653. International Foundation for Autonomous Agents and Multiagent Systems (IFAAMAS), 2022.

[KKZ23] Panagiotis Kanellopoulos, Maria Kyropoulou, and Hao Zhou. Debt transfers in financial networks: Complexity and equilibria. In *AAMAS*, pages 260–268. ACM, 2023.

[KT06] Jon Kleinberg and Eva Tardos. *Algorithm design*. Pearson Education India, 2006.

[LPT17] Matt Leduc, Sebastian Poledna, and Stefan Thurner. Systemic risk management in financial networks with credit default swaps. *The Journal of Network Theory in Finance*, 5, 01 2017.

[MP91] Nimrod Megiddo and Christos H. Papadimitriou. On total functions, existence theorems and computational complexity. *Theor. Comput. Sci.*, 81(2):317–324, 1991.

[Pap94a] Christos H. Papadimitriou. *Computational complexity*. Addison-Wesley, 1994.

[Pap94b] Christos H Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. *Journal of Computer and system Sciences*, 48(3):498–532, 1994.

[PW20] Pál András Papp and Roger Wattenhofer. Network-aware strategies in financial systems. In *ICALP*, volume 168 of *LIPIcs*, pages 91:1–91:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[PW21a] Pál András Papp and Roger Wattenhofer. Debt swapping for risk mitigation in financial networks. In *EC*, pages 765–784. ACM, 2021.

[PW21b] Pál András Papp and Roger Wattenhofer. Sequential defaulting in financial networks. In *ITCS*, volume 185 of *LIPIcs*, pages 52:1–52:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[PW22] Pál András Papp and Roger Wattenhofer. Default ambiguity: finding the best solution to the clearing problem. In *Web and Internet Economics: 17th International Conference, WINE 2021, Potsdam, Germany, December 14–17, 2021, Proceedings*, pages 391–409. Springer, 2022.

[Rub15] Aviad Rubinstein. Inapproximability of Nash equilibrium. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 409–418, 2015.

[RV13] Leonard CG Rogers and Luitgard AM Veraart. Failure and rescue in an interbank network. *Management Science*, 59(4):882–898, 2013.

[Sch79] A. Schönhage. On the power of random access machines. In *8th International Colloquium on Automata, Languages and Programming (ICALP 1979)*, volume 71, pages 520–529. Springer, 1979.

[SM06] Charles Smithson and David Mengle. The promise of credit derivatives in nonfinancial corporations (and why it's failed to materialize). *Journal of Applied Corporate Finance*, 18(4):54–60, 2006.

[SS21] Steffen Schuldenzucker and Sven Seuken. Portfolio compression in financial networks: Incentives and systemic risk. *Available at SSRN 3135960*, 2021.

[SSB17a] Steffen Schuldenzucker, Sven Seuken, and Stefano Battiston. The computational complexity of clearing financial networks with credit default swaps. *CoRR*, abs/1710.01578, 2017.

[SSB17b] Steffen Schuldenzucker, Sven Seuken, and Stefano Battiston. Finding clearing payments in financial networks with credit default swaps is PPAD-complete. In *ITCS*, volume 67 of *LIPIcs*, pages 32:1–32:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.

[SSB20] Steffen Schuldenzucker, Sven Seuken, and Stefano Battiston. Default ambiguity: Credit default swaps create new systemic risks in financial networks. *Manag. Sci.*, 66(5):1981–1998, 2020.

[Tiw92] P. Tiwari. A problem that is easier to solve on the unit-cost algebraic RAM. *Journal of Complexity*, 8(4):393–397, 1992.

[Yan09] Mihalis Yannakakis. Equilibria, fixed points, and complexity classes. *Computer Science Review*, 3(2):71–85, 2009.