The Company Control Problem, Reactively

Davide Magnanimi[1,2], Stefano Ceri[1], Luigi Bellomarini[2] and Davide Martinenghi[1]

[1]*Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Italy*
[2]*Banca d'Italia*

Abstract

The Company Control Problem (CCP) consists in understanding who exerts decision power in companies, and is of crucial interest to central banks, financial intelligence units, and market regulators. Computing control relationships is computationally expensive and involves traversing a shareholding graph that may comprise hundreds of millions of nodes and billions of edges and properties. This graph is highly volatile, thus it is unaffordable to entirely recompute the control relationships for each change. Our incremental, rule-based formalization of CCP offers a practical and scalable solution to the problem.

1. Introduction

Who exerts decision power in a company? Given a national or international network of millions of entities interconnected by billions of shareholding relationships, who controls—directly or indirectly—a given company? This question, known as the *Company Control Problem* (CCP) [1], is of great interest to banks, national central banks [2], financial intelligence units, regulatory and supervisory authorities, and Fintech firms. Indeed, it impacts their core business: loan granting processes, *creditworthiness evaluation* and *know-your-customer* onboarding procedures are hinged on company control, to spot any possible conflict of interest between borrowers and lenders. In *banking supervision* and *anti-money laundering*, the authorities need to identify the actual centers of power so as to accurately detect the drivers and the ultimate beneficiaries of financial misconducts. In *monetary policy* in the Eurosystem, authorities oversee the credit market by assessing the so-called *collateral eligibility* [3], that is, the independence of the collateral issuer with respect to the entities involved in monetary policy operations, in order to reduce the credit risk. Company control is a relevant topic also in the broader macroeconomic community [4, 5], where *market concentration* sparks recurring debates.

Industrial Scenario. The Bank of Italy holds a company network in the form of a *Company Knowledge Graph* (KG), stored as a *property graph* [6]. The nodes denote individuals and companies, and the edges represent shareholding relationships (or *ownerships*), with the respective percentage. Since the Bank of Italy operates as a regulator and supervisor in the European System of Central Banks, the graph, which was initially focussed on Italian companies, is being gradually expanded to span the entire EU-level space, with hundreds of millions of nodes and billions of edges and properties.
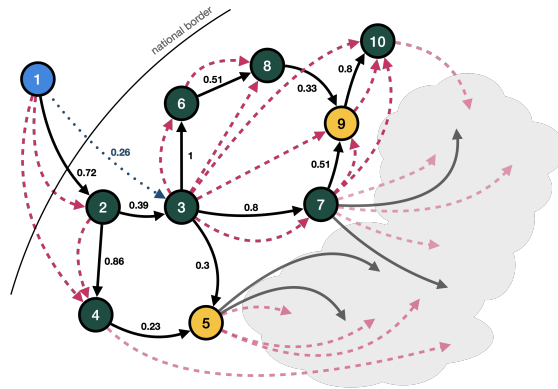
Figure 1: An excerpt from the KG. Yellow nodes are companies with a national strategic relevance; the blue node is a non-EU shareholder. Solid edges are existing ownership relationships; dashed-magenta edges are control relationships. The dotted-blue edge represents a candidate share acquisition.

A core element of the graph are *control edges*, which are derived "x *controls* y" relationships computed from the shareholding edges according to the following definition [1]:

A company x controls a company y if x (i) directly owns the majority of y shares, or (ii) controls a set of companies that, possibly together with x itself, jointly hold the majority of y.

A static pre-computation of control edges is not suitable for uprising economic and financial applications, which need to capture dynamic phenomena involving frequent changes in ownership edges. A timely "reactive" computation of control edges is crucial for a host of institutional services of the Bank of Italy, which call for immediate action once a change of control is detected. Also in the analytical perspective, timeliness in updating control edges matters, e.g., when analysts need to perform interactive "*what-if analysis*" to seize the systemic effect on control, by simulating changes of ownership.

As a matter of fact, a large category of different industrial scenarios share a common trait: a very limited number of changes in the ownership edges in a very small time interval cause changes in control relationships, potentially in distant areas of the KG, to be accounted for as efficiently as possible.

Example 1. *Consider now the excerpt in Figure 1 of our KG: an analyst wants to swiftly assess the impact of the acquisition of 26% of the shares of Company 3 undertaken by a non-EU company 1. In fact, it would change the controllers of Companies 5 and 9, which are of strategic national relevance, and potentially also affect others. Company 1 already controls 39% of Company 3, and, with a further 26% acquisition would gain control of the majority. Consequently, Company 1 would exert control over Company 7, the strategic 9, as well as 6, 8, and 10, already controlled by 3. Finally, via the controlled Companies 3 and 4, our non-EU shareholder would take over the strategic Company 5 as well and, with a cascade effect, also other companies controlled by 5, 7, 9, and 10.∎*

As of now, the Bank of Italy computes the control relationships between companies in a batch fashion: daily ownership variations are collected from specific data sources, the ownership edges are updated and all the control relationships are finally recomputed. Considering the current rate of thousands of changes per day, rapidly soaring for the EU-level role of the Bank,

not only is the batch approach incompatible with the mentioned on-line applications, but also causes longer and longer periods with outdated data, unacceptable in practice.

Goal. In this work, we consider a declarative specification of the CCP problem in Datalog [7, 8] extended with features of practical utility such as aggregation, negation, and inequalities, and introduce a reactive formulation of it as an *inference task* that exploits changes of ownership so as to locally and efficiently update control relationships. We propose a production-ready system that provides our analysts with a *reactive Knowledge Graph*, having continuously and incrementally updated control edges. We implemented our solution in the Vadalog System [9, 10], a state-of-the-art reasoner using the VADALOG language of the Datalog$^{\pm}$ family [11].

Contribution. We provide a *declarative, incremental, reasoning-based* approach to the computation of company control relationships over large and frequently updated Company Knowledge Graphs, which we name *reactive control computation*. In particular, we propose a **reactive logic-based formulation of the CCP**. We introduce specific predicates to reason on insertions and deletions of shareholding relationships in the graph and use them to incrementally update control edges. Our formulation leverages the topology of the graph to exploit forms of *update locality* and avoid massive cascade updates or full recomputation, as in the batch approach.

In an extended version of this work [12], we also validate our approach in many experimental settings comprising real snapshots of the Italian Company KG of the Bank of Italy as well as a large number of synthetic graphs, of various sizes and topologies.

2. The Company Control Problem

Before analyzing the CCP, let us first briefly present the Italian Company Knowledge Graph.

The Company Knowledge Graph contains detailed information about companies, individuals, corporate events, many other entities, and the relationships among them. In this work, we focus on entities and relationships regarding *ownership (of shares) relationships*, which connect subsidiaries to their shareholders (either other companies or individuals), and define an *ownership graph* $G = (V, E, L)$ as a directed, weighted graph in which V is the set of nodes representing such companies or individuals; E is the set of directed edges, with $E \subseteq V^2$, that represent shareholding relationships; and $L : E \to (0, 1]$ is a labeling function that assigns to each edge $e = (v, w) \in E$ a label $L(e)$ representing the percentage of total equity of company w held by the shareholder v. Clearly, for every node $w \in V$, we require that $\sum_{v \in pred_w} L((v, w)) \leq 1$, with $pred_w$ indicating the set of nodes $v \in V$ that are shareholders of w.

For our purposes, we model the ownership graph as a database $\mathcal{D}$ of facts $\mathsf{Own}(x, y, w)$, where x is a shareholder (company or individual), y is a company, and w is the owned share percentage. For example, the graph in Figure 1 would be encoded as $\mathcal{D} = \{\mathsf{own}(1, 2, 0.72),$ $\mathsf{own}(2, 4, 0.86), \mathsf{own}(2, 3, 0.39), \mathsf{own}(4, 5, 0.23), \ldots\}$.

We encode the CCP and our solution as an *inference task* in a Datalog-based formalism.

The Company Control Problem. Our definition of the CCP is majority-based and broadly accepted in the corporate economics community [5]. Let us focus on a portion of Figure 1. Shareholder 3 directly owns 100% and 80% of the total equity of 6 and 7, respectively. Therefore,

shareholder 3 obtains the control over them, as well as over the shares they own of other companies. In particular, 3 is able to exert control on firm 8, as 6 directly controls it. Consequently, by controlling 7 and 8, the shareholder 3 also controls the sum of the shares of 9 they own. As the sum is above 50%, it allows 3 to control company 9 as well.

The CCP has been studied in the database literature and proved to be quadratic, in its pairwise decision formulation, that is, given an ownership graph and two companies x and y from this graph, decide whether a control relationship between them holds [1]. In this paper, we will focus on the more general inference task of deriving all the control relationships. Thanks to its recursive nature, an elegant VADALOG formulation of CCP can be provided as follows [1]:

$$\mathsf{company}(x) \rightarrow \mathsf{control}(x, x) \tag{1}$$

$$\mathsf{control}(x, z), \mathsf{own}(z, y, w) \wedge v = \textit{sum}(w) \wedge v > 0.5 \rightarrow \mathsf{control}(x, y) \tag{2}$$

Given that every company x controls itself (Rule 1), by Rule 2, x controls y if the sum of the shares w of y owned by companies z, over all companies z controlled by x, is above 50%.

3. Reactive Company Control

Given the company KG, a *materialization* and *batch* approach to the CCP would consist in periodically deleting all the control edges, considering the most recent version of the graph, and computing a new materialization of the relationships—unaffordable in practice, as we have discussed. Conversely, in this section we introduce an incremental solution, which "*reacts*" to changes and updates the control edges.

Towards an incremental approach, we enrich our vocabulary of VADALOG atoms with adorned versions. In detail, given a set of changes of the KG performed in the interval $[t_1, t_2]$, for an atom ϱ, we denote as *newϱ* an atom binding to all facts for ϱ holding after t_2; as ϱ^+, an atom binding to all the facts for ϱ inserted during $[t_1, t_2]$; as ϱ^-, an atom binding to all the facts for ϱ that have been removed during $[t_1, t_2]$; finally, ϱ binds to all facts holding before t_1.

Then, in our context we will refer to the set of ownerships holding before (own) and after the update (newOwn); to those inserted (own$^+$) and removed (own$^-$) by the update; to the old materialization of the control relationship (control) and to the facts to be inserted (control$^+$) or deleted (control$^-$) to materialize the new one (newControl). For the sake of simplicity, we will refer to a generic update (interval) and will not explicitly refer to $[t_1, t_2]$ in the predicate syntax. We model modifications to share amounts as a deletion of the respective ownership fact followed by a re-insertion. Inserted or deleted nodes are represented by adding (own$^+$) resp. removing (own$^-$) all the ownership facts they participate in.

Our framework models the scenario at hand as an instance of the *materialization maintenance* problem [13, 14, 15, 16], in particular, as the task of maintaining the materialization of $\Sigma(\mathcal{D})$, where Σ is the company control program and $\mathcal{D}$ is the database encoding the KG. We adopt a *Delete/Rederive* (DRed) [17] strategy to handle deletions: we first delete the affected control facts, identifying them with an *overstimation/refinement* process. Then, we apply the insertions and extend $\mathcal{D}$ with all the consequences of the inserted facts.

More precisely, given $\mathcal{D}$ and a set of updates occurred during $[t_1, t_2]$ that can be referred to via adorned predicates, the materialization maintenance task is solved by two sets of Vadalog rules Σ_D and Σ_I with two associated inference tasks that, respectively, infer the sets $\mathbf{C}^- = \Sigma_D(\mathcal{D})$ of control facts to be deleted from $\mathcal{D}$, and the set $\mathbf{C}^+ = \Sigma_I(\mathcal{D} \setminus \mathbf{C}^-)$ to be added to $\mathcal{D} \setminus \mathbf{C}^-$. In total, $(\mathcal{D} \setminus \mathbf{C}^-) \cup \mathbf{C}^+$ is the updated version of $\Sigma(\mathcal{D})$.

3.1. Deletion rules (Σ_D)

The rules of Σ_D are conceptually organized in three groups that (i) identify the facts that are potentially deleted as a consequence of ownership changes; (ii) for each of those facts, try to identify alternative derivation paths; (iii) single out the facts deleted by group (i) and not reinserted by group (ii).

Group 1 - Potentially deleted controls. The rules of this group identify, by overestimation, the control facts potDControl that are potentially deleted as a consequence of ownership changes. This deletion event can take place in three scenarios, that we see next. For each scenario, we present the rationale, the Vadalog rules, and an explanatory snippet. Each snippet offers a visual representation of the rule application, where deleted facts are denoted by red edges.

- **Deletion of a direct ownership.** The deletion of a direct ownership $x \to y$ reduces the total shares of y controlled by x. Consequently, a control relationship from x to y may not hold anymore, as denoted by a new potDControl fact.

$$\mathsf{own}^-(x, y, q) \wedge \mathsf{control}(x, y) \to \mathsf{potDControl}(x, y) \tag{3}$$
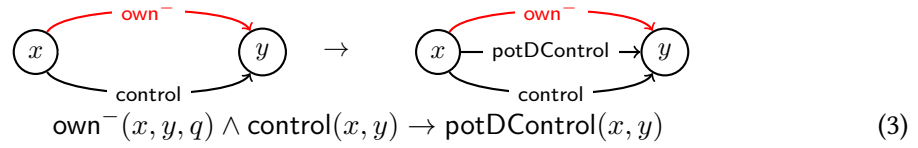
- **Deletion of an indirect ownership.** Let x control y via direct and indirect undertakings, e.g., via z. If the indirect undertaking is removed, x may lose its control on y.

$$\mathsf{control}(x, z) \wedge \mathsf{own}^-(z, y, q) \wedge \mathsf{control}(x, y) \to \mathsf{potDControl}(x, y) \tag{4}$$
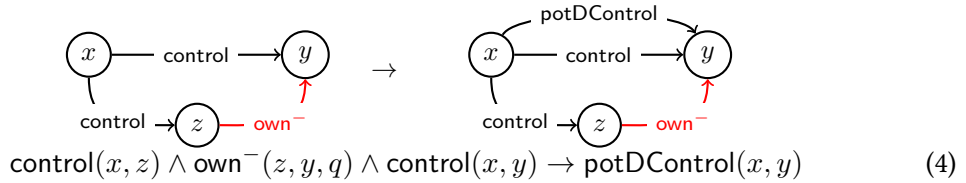
- **Recursive propagation.** If x controls y also via a controlled company z retaining shares of y, a potential deletion of the control from x to z could cause x to lose its control on y, recursively.

$$\mathsf{potDControl}(x, z) \wedge \mathsf{own}(z, y) \wedge \mathsf{control}(x, y) \to \mathsf{potDControl}(x, y) \tag{5}$$
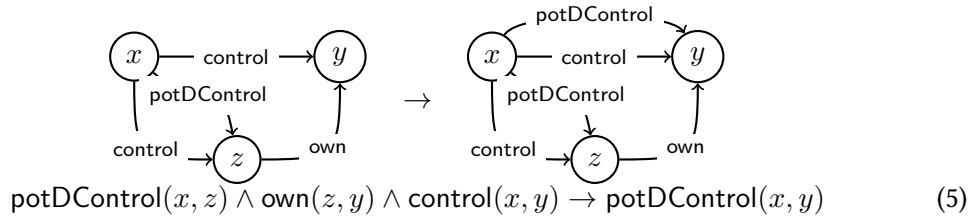
Group 2 - Recomputing affected controlled shares. The rules of this group seek for alternative derivations of the control relationships affected by ownership deletions.

In this group, we recompute direct and indirect control shares only for the pairs of nodes interested by a potDControl relationship, hence minimizing the portion of the traversed graph. To find alternative derivations, given a potentially deleted control, we collect all the shares q of y controlled by x via z into the predicate potDControlShare(x, z, y, q), so that they are eventually aggregated. Rules 6 and 7 together represent the base case of this aggregation; Rules 8 and 9, which are mutually recursive, represent the inductive case.

To start, we need to introduce an auxiliary predicate to consider only the ownerships that have not been deleted, i.e.: $\mathsf{own}(x, y, q) \land \neg\mathsf{own}^-(x, y, q) \to \mathsf{stillOwn}(x, y, q)$.

$$\mathsf{potDControl}(x, y) \land \mathsf{stillOwn}(x, y, q) \to \mathsf{potDControlShare}(x, y, y, q) \tag{6}$$

With Rule 7, we want to collect the shares of indirect controls. We consider the shares q for the pairs x and y connected by a potentially deleted control, in the case x certainly controls some other node z, i.e., $\mathsf{potDControl}(x, z)$ does not hold, and z still retains q shares of y.

$$\mathsf{potDControl}(x, y) \land \mathsf{stillOwn}(z, y, q) \land \mathsf{control}(x, z) \land \neg \mathsf{potDControl}(x, z)$$
$$\to \mathsf{potDControlShare}(x, z, y, q) \tag{7}$$

The goal of Rules 8 and 9 is to collect the pairs for which while potentially the control may have been deleted, in fact, it is not, as alternative derivations do exist. Rule 8 creates potStillControl facts, which witness that a control of x on y still exists if the sum of the newly derived controlled shares (those computed by Rules 6, 7, and 9) is above 0.5.

$$\mathsf{potDControlShare}(x, z, y, q) \land j = sum(q) \land j > 0.5 \to \mathsf{potStillControl}(x, y) \tag{8}$$

In Rule 9 we collect the shares that x controls of y via z, in the case the potentially deleted control between x and z instead still holds (as witnessed by potStillControl facts produced by Rule 8) as well as the ownership between z and y.

$$\mathsf{potDControl}(x, y) \land \mathsf{potStillControl}(x, z) \land \mathsf{stillOwn}(z, y, q) \to \mathsf{potDControlShare}(x, z, y, q) \tag{9}$$

Group 3 - Computing $\mathbf{C}^-$. We are now ready to single out the control facts to be removed in the absence of alternative derivations (Rule 10).

$$\mathsf{potDControl}(x, y) \land \neg \mathsf{potStillControl}(x, y) \to \mathsf{control}^-(x, y) \tag{10}$$

Example 2. *Let us consider the example in Figure 1 again and suppose Company* 6 *sells all its shares of* 8*, as captured by the fact* $\mathsf{own}^-(6, 8, 0.51)$. *In Group 1, we obtain* $\mathsf{potDControl}(6, 8)$ *and* $\mathsf{potDControl}(3, 8)$ *by Rules 3 and 4 respectively;* $\mathsf{potDControl}(3, 9)$ *and* $\mathsf{potDControl}(3, 10)$ *by Rule 5, respectively. When recomputing the affected control shares, Group 2 produces only* $\mathsf{potDControlShare}(3, 7, 9, 0.51)$ *(Rule 7) and, therefore,* $\mathsf{potStillControl}(3, 9)$ *(Rule 8). Company* 3 *still control* 9 *as the total controlled shares is above 50%. Rule 9 derives* $\mathsf{potDControlShare}(3, 9, 10, 0.8)$. *Again, another still holding control is derived by Rule 8, i.e.,* $\mathsf{potStillControl}(3, 10)$. *Note that the control facts of Company* 3 *on* 8 *and* 6 *on* 8 *do not exist any longer. Finally, Rule 10 produces the deleted controls* $\mathsf{control}^-(6, 8)$ *and* $\mathsf{control}^-(3, 8)$. ∎

Applying C⁻. Given the outcome of the inference task $\Sigma_D(\mathcal{D})$, Rule 11 removes the deleted control facts **C⁻** from $\mathcal{D}$. Note that we are actually performing a "virtual" deletion by producing facts for stillControl by set difference.
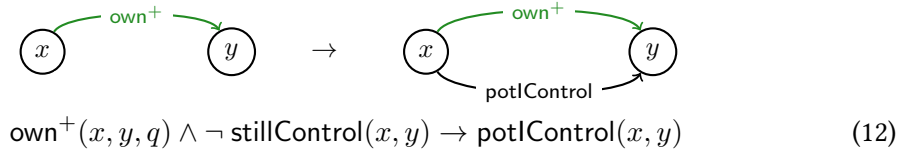
$$\text{control}(x,y) \wedge \neg\,\text{control}^-(x,y) \rightarrow \text{stillControl}(x,y) \tag{11}$$

3.2. Insertion rules (Σ_I)

The rules of Σ_I are organized in three groups that (i) identify the control facts that are potentially added due to new ownerships; (ii) for each of those new control candidates, compute the respective control shares; (iii) based on these shares, single out the actual new control facts.
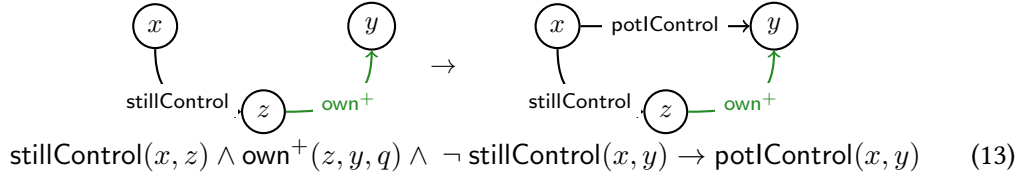
Group 1 - Potentially inserted controls. The rules of this group identify three scenarios in which it is possible to derive by overestimation new control facts potlControl whenever new ownership relationships (own⁺) or, recursively, new controls (control⁺) are added. Let us introduce the scenarios; we will use green edges to denote the added facts.

- **Insertion of a direct ownership.** The addition of a new ownership from x to y contributes to the total shares of y controlled by x and a new control may arise.
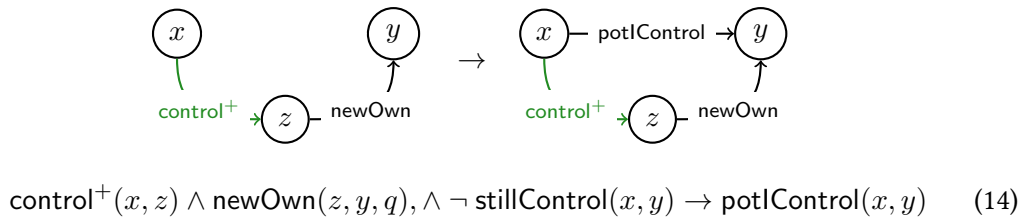

$$\text{own}^+(x,y,q) \wedge \neg\,\text{stillControl}(x,y) \rightarrow \text{potlControl}(x,y) \tag{12}$$

- **Insertion of an indirect ownership.** Let z be a company controlled by x. The addition of a new ownership from z to y contributes to the total shares of y that x controls, directly or indirectly, possibly resulting in a new control.

$$\text{stillControl}(x,z) \wedge \text{own}^+(z,y,q) \wedge \neg\,\text{stillControl}(x,y) \rightarrow \text{potlControl}(x,y) \tag{13}$$

- **Recursive propagation.** Let z own shares of y. A new control (control⁺) from x to z may contribute to the total amount of shares of y, directly and indirectly, controlled by x. Recursively, this may lead to a new control from x to y.

$$\text{control}^+(x,z) \wedge \text{newOwn}(z,y,q), \wedge \neg\,\text{stillControl}(x,y) \rightarrow \text{potlControl}(x,y) \tag{14}$$

Group 2 - Computation of new controlled shares. In this group, we collect all the direct (Rule 15) and indirect (Rule 16, 17) controlled shares in $\mathcal{D} \setminus \mathbf{C}^-$ between the pairs singled out by potIControl facts.

$$\text{potIControl}(x, y) \wedge \text{newOwn}(x, y, q) \rightarrow \text{potIControlShare}(x, y, y, q) \quad (15)$$

$$\text{potIControl}(x, y) \wedge \text{stillControl}(x, z) \wedge \text{newOwn}(z, y, q) \rightarrow \text{potIControlShare}(x, z, y, q) \quad (16)$$

$$\text{potIControl}(x, y) \wedge \text{control}^+(x, z) \wedge \text{newOwn}(z, y, q) \rightarrow \text{potIControlShare}(x, z, y, q) \quad (17)$$

Group 3 - Computing $\mathbf{C}^+$. Finally, the new controls (control^+) are derived by summarizing all the newly computed controlled shares and selecting only those above 0.5.

$$\text{potIControlShare}(x, z, y, q) \wedge j = sum(q) \wedge j > 0.5 \rightarrow \text{control}^+(x, y) \quad (18)$$

Note that the new controls are in turn fed into Rule 14 and 17.

Example 3. *Let us suppose the candidate share acquisition operation in Figure 1 is settled and the reactive approach is employed for updating controls. The fact* $\text{own}^+(1, 3, 0.26)$ *captures the update. We obtain* $\text{potIControl}(1, 3)$ *by Rule 12, and* $\text{potIControlShare}(1, 3, 3, 0.26)$ *and* $\text{potIControlShare}(1, 2, 3, 0.39)$ *by Rules 15 and 16, respectively. As the sum of the two computed controlled shares is above 50%, Rule 18 yields* $\text{control}^+(1, 3)$. *Then, by Rule 14 we generate* $\text{potIControl}(1, 5)$, $\text{potIControl}(1, 6)$, *and* $\text{potIControl}(1, 7)$. *By evaluating all the direct and indirect contributions in the recursive activation of rules, we finally also obtain* $\text{control}^+(1, 5)$, $\text{control}^+(1, 6)$, $\text{control}^+(1, 8)$, $\text{control}^+(1, 7)$, $\text{control}^+(1, 9)$, *and* $\text{control}^+(1, 10)$. ∎

Applying $\mathbf{C}^+$. Once the inference task $\Sigma_I(\mathcal{D} \setminus \mathbf{C}^-)$ has been completed, the obtained control edges $\mathbf{C}^+$ are added to $\mathcal{D} \setminus \mathbf{C}^-$ with Rules 19 and 20, via a new control predicate (newControl).

$$\text{stillControl}(x, y) \rightarrow \text{newControl}(x, y) \quad (19)$$

$$\text{control}^+(x, y) \rightarrow \text{newControl}(x, y) \quad (20)$$

4. Conclusion

We contributed a declarative, incremental, and reasoning-based formulation of the CCP, aimed at solving the problem of efficiently updating an existing materialization of control edges in the shareholding network held by the Bank of Italy. We put our logical formulation into action by implementing and running it within the VADALOG System for logical reasoning.

The framework, is currently in a pre-production stage, but has production ambitions, as we intend to deploy and use it for multiple applications of the Bank of Italy.

Acknowledgment. This work is part of Davide Magnanimi's Executive PhD program at Politecnico di Milano.

References

[1] A. Gulino, S. Ceri, G. Gottlob, E. Sallinger, L. Bellomarini, Distributed company control in company shareholding graphs, in: ICDE, 2021.

[2] L. Bellomarini, L. Bencivelli, C. Biancotti, L. Blasi, F. P. Conteduca, A. Gentili, R. Laurendi, D. Magnanimi, M. S. Zangrandi, F. Tonelli, S. Ceri, D. Benedetto, M. Nissl, E. Sallinger, Reasoning on company takeovers: From tactic to strategy, Data Knowledge Engineering (2022).

[3] Guideline of the european central bank of 20 september 2011 on monetary policy instruments and procedures of the eurosystem (ecb/2011/14), https://bit.ly/3VqqWtT, ????

[4] F. Barca, M. Becht, The control of corporate Europe, 2001.

[5] J. B. Glattfelder, Ownership networks and corporate control: mapping economic power in a globalized world, Ph.D. thesis, 2010.

[6] R. Angles, The property graph database model, in: AMW, CEUR Workshop Proceedings, 2018.

[7] S. Abiteboul, R. Hull, V. Vianu, Foundations of Databases, 1995.

[8] S. Ceri, G. Gottlob, L. Tanca, What you always wanted to know about datalog (and never dared to ask), TKDE (1989).

[9] L. Bellomarini, D. Benedetto, G. Gottlob, E. Sallinger, Vadalog: A modern architecture for automated reasoning with large knowledge graphs, Information Systems (2020).

[10] T. Baldazzi, L. Bellomarini, M. Gerschberger, A. Jami, D. Magnanimi, M. Nissl, A. Pavlovic, E. Sallinger, Vadalog: Overview, extensions and business applications, in: Reasoning Web. Causality, Explanations and Declarative Knowledge, 2022.

[11] A. Calì, G. Gottlob, A. Pieris, New expressive languages for ontological query answering, in: AAAI, 2011.

[12] D. Magnanimi, L. Bellomarini, S. Ceri, D. Martinenghi, Reactive company control in company knowledge graphs, in: ICDE, 2023. Best Industrial Paper.

[13] S. Ceri, J. Widom, Deriving production rules for incremental view maintenance, in: VLDB, 1991.

[14] B. Motik, Y. Nenov, R. E. F. Piro, I. Horrocks, Incremental update of datalog materialisation: the backward/forward algorithm, in: AAAI, 2015.

[15] M. Staudt, M. Jarke, Incremental maintenance of externally materialized views, in: VLDB, 1996.

[16] A. Gupta, I. S. Mumick, V. S. Subrahmanian, Maintaining views incrementally, in: SIGMOD, 1993.

[17] P. Hu, B. Motik, I. Horrocks, Optimised maintenance of datalog materialisations, in: AAAI, 2018.