

Knowledge Graphs & Reasoning in Concert: Automating Regulation in Corporate Economics

Luigi Bellomarini¹, Matteo Brandetti^{1,2}, Andrea Gentili¹, Rosario Laurendi¹ and
Davide Magnanimit^{1,3}

¹Bank of Italy, Italy

²TU Wien, Austria

³Politecnico di Milano, Italy

Abstract

Unravelling financial scenarios characterized by the interplay of many entities with complex interconnections is at the core of banking supervision. In particular, it involves applying technical regulations to large graphs of banks, financial intermediaries, and companies, to derive the knowledge needed for compliance checks. In such settings, modern notions of logic-based Knowledge Graphs (KGs) come to help. A KG is composed of an extensional component, the enterprise data, and an intensional component, a set of rules capturing the regulations, which are applied to infer a derived extensional component, thus enriching the original graph. State-of-the-art logical languages of the Datalog+/- family strike a good balance between expressive power and computational complexity, allowing for effective encoding of real-world scenarios.

This paper is about the experience of the Central Bank of Italy in automating banking supervision regulation with logic-based reasoning on KGs. In particular, we explain our design-to-production methodology through the specific problem of “concerted control”, that is, spotting the real decision makers in corporate graphs, a particularly challenging task in which the involved actors may orchestrate coordinated or collusive moves, and hide the real centres of interest. We illustrate the design methodology of the KG, discuss our encoding of the business rules, and assess the performance of their execution within Vadalog, a state-of-the-art reasoner.

Keywords

conceptual design, finance, knowledge graphs, Datalog+/-, knowledge representation and reasoning, banking supervision

1. Introduction

In the financial space, the application of regulatory prescriptions and compliance checks in the presence of large networks of entities with a complex interplay is particularly challenging. Specifically, banking supervision, i.e., the oversight ensuring that the banks follow the regulation, often entails unravelling such networks so as to compute the knowledge needed [1]. The typical supervision action follows a three-step process: (i) the financial authority is tasked with some regulation-related questions concerning a supervised entity, a bank or a financial intermediary; (ii) the regulation-related questions are translated into one or more data-related query to be answered thanks to an intelligent use of the enterprise data assets of the financial authority; (iii) the data-related queries are executed, integrated, and contribute to the construction of the business answer.

In these settings, logic-based notions of *Knowledge Graphs* [2] (KGs) are helpful. They consist of an *extensional component*, a database of facts integrating the enterprise data stores, and an *intensional component*, a set of logic-based rules capturing the regulation. The extensional component is modelled as a graph. The intensional component is expressed with a *Knowledge Representation and Reasoning* (KRR) formalism. Languages of the Datalog[±] family [3] such as VADALOG [4], achieve a good balance

Workshop on Logic Programming and Legal Reasoning (LPLR 2025) collocated within the 41st International Conference on Logic Programming (ICLP), September 12–13, 2025, Rende, Italy

✉ luigi.bellomarini@bancaditalia.it (L. Bellomarini); matteo.brandetti@bancaditalia.it (M. Brandetti); andrea.gentili@bancaditalia.it (A. Gentili); rosario.laurendi@bancaditalia.it (R. Laurendi); davide.magnanimit@bancaditalia.it (D. Magnanimit)

ORCID 0000-0001-6863-0162 (L. Bellomarini); 0009-0009-6328-4585 (M. Brandetti); 0000-0002-6824-826X (A. Gentili); 0000-0002-0567-0797 (R. Laurendi); 0000-0002-6560-8047 (D. Magnanimit)



© 2025 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0). The views and opinions expressed in this paper are those of the authors and do not necessarily reflect the official policy or position of Banca d'Italia.

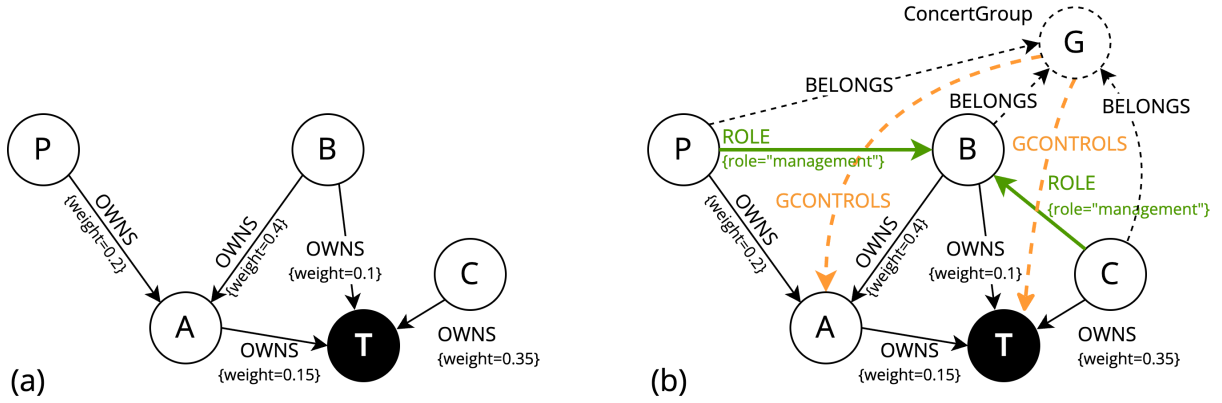


Figure 1: (a) A company ownership graph and (b) the notion of “concerted control” applied to it. Without concert, no control links can be present in the network, thus possibly eluding supervisory procedures.

between expressive power and computational complexity, offering *tractable query answering* while supporting *full recursion* and *existential quantification*, features that are essential for KG navigation and ontological reasoning, respectively. The application of the intensional component to the extensional component, through the so-called *reasoning process*, gives rise to the *derived extensional component*.

In the Central Bank of Italy, we have extensive experience in tackling regulation-intensive tasks by adopting logical KGs and we have developed methodologies, tools, and best practices to design, construct, and roll out into production KG-based solutions. The extensional component of the KG integrates the information assets of the Central Bank into a supervision KG. To obtain a comprehensive, easy-to-communicate, and sound extensional component, we introduced a *schema-design methodology* [5], which fosters the idea of a high-level conceptual understanding of the domain, featuring a straightforward conversion to an executable form, fit for the target system. The design of the rules and constraints to encode the regulation is not new in the literature [6, 7, e.g.]. It is intellectually stimulating and involves delving into the computer-theoretical internals of the entailed problems and capturing them in a logic-based formalism. We have developed experience with many problems revolving around the corporate economic area and, in particular, related to understanding who retains control over the decisions of banks and intermediaries [8].

Architectural challenges arise in the so-called *reasoning process*, where the regulation in VADALOG is applied to the extensional component, to implement steps (i)-(iii) of the supervision endeavour. A particularly relevant one is scalability, since many analyses often involve worldwide KGs, with hundreds of millions of nodes and even billions of edges. In this sense, we have recently developed systems for scalable reasoning in VADALOG, which exhibits very good performance in the settings at hand [9].

Concerted Control. In this paper, we want to walk the thin line between theory and practice and share our design-to-production experience and methodology. We will follow a by-example approach and showcase how the regulation-intensive problem of *concerted control* [10] is faced by (i) designing the extensional component of the KG; (ii) encoding the regulation in VADALOG in the intensional component of the KG; (iii) executing the regulation in the VADALOG system in a distributed setting.

Example 1. Consider Figure 1. Two entities, *P* and *B*, hold a minority stake (respectively 20% and 40%) of a company *A*. Individually, neither holding would be enough to control *A*, according to a majority threshold of 50%. In fact, *P* holds a managerial role in *B*. This suggests that *P* and *B* have a potential strategic alignment. Therefore, seemingly independent ownerships can “act in concert” and exert influence or even control *A*, summing their shares in it. Clearly, this definition recursively propagates through the graph: if a company *C* has an managerial role in *B*, coordinated interest can spread: in this case, the concert group *G* would indirectly control the shares held directly by the controlled company *A*, and concerting actors *B* and *C*, eventually gaining control of an intermediary *T* as well with 60% of the shares. ■

Contribution. Along the lines of concerted control, this paper provides a guided explanation of our methodology as well as pointers to more in-depth notions for the interested reader.

- We introduce **KG-Model**, our schema-design methodologies for KGs and apply it to concerted control.
- We **analyse and formalize the problem of concerted control**, providing its encoding in VADALOG.
- We discuss how the VADALOG system addresses the mentioned scenario in a **scalable setting**.

Overview. The remainder of the paper is organised as follows. In Section 2, we lay out the preliminary technical background of KGs and VADALOG. In Section 3, we design the schema to solve our problem. Section 4 is dedicated to the concerted control problem. Section 5 shows the VADALOG system in action in the setting at hand. Finally, in Section 6, we draw our conclusions.

2. Preliminaries

Let us now introduce the necessary preliminaries. First, we describe the technical background, then we introduce the company control problem, as an example.

Relational Foundations. A (*relational*) *schema* \mathbf{S} is a finite set of relation symbols (or *predicates*) with associated arity. A *term* is either a constant or a variable. An *atom* over \mathbf{S} is an expression of the form $R(\bar{v})$, where $R \in \mathbf{S}$ is of arity $n > 0$ and \bar{v} is an n -tuple of terms. A *database (instance)* over \mathbf{S} associates to each symbol in \mathbf{S} a relation of the respective arity over the domain of constants and nulls. The *extensional component* of a KG is a database instance.

Vadalog. A VADALOG program consists of a set of facts and *existential rules*, function-free Horn clauses of the form $\forall \bar{x} \forall \bar{y} (\varphi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z} \psi(\bar{x}, \bar{z}))$, where $\varphi(\bar{x}, \bar{y})$ (the *body*) and $\psi(\bar{x}, \bar{z})$ (the *head*) are conjunctions of atoms over the respective predicates and the arguments are vectors of variables and constants. It may also feature *equality-generating dependencies* (EGDs), first-order implications of the form $\forall \bar{x} (\varphi(\bar{x}) \rightarrow x_i = x_j)$, where $\varphi(\bar{x})$ is a conjunction of atoms and $x_i, x_j \in \bar{x}$. Existential quantification in rule heads introduces new fresh symbols called *labelled nulls*, which are placeholders for unknown values. EGDs assign values to labelled nulls or enforce their equivalence. This mechanism is fundamental for a broad class of reasoning tasks, including graph traversal, clustering, entity resolution, and data fusion [11]. Moreover, our applications call for multiple features that extend the declarative language. Among them, *aggregate functions*, namely *sum*, *prod*, *min*, *max* and *count*, as well as SQL-like grouping constructs, are particularly relevant. Important extensions also include *stratified* and *ground negations*, *negative constraints*, *negations as failure*, and *expressions* in rule bodies, modelled with *comparison* ($>$, $<$, \geq , \leq , \neq) and *algebraic* ($+$, $-$, $*$, $/$, etc.) operators. The intensional component of a KG consists of a set of rules Σ .

Reasoning and Query Answering. Intuitively speaking, an *ontological reasoning* task consists in answering a *conjunctive query* (CQ) Q over a database D , augmented with a set Σ of logical rules. A conjunctive query (CQ) Q over a schema \mathbf{S} is an implication $q(\mathbf{x}) \leftarrow \phi(\mathbf{x}, \mathbf{y})$, where $\phi(\mathbf{x}, \mathbf{y})$ is a conjunction of atoms, and $q(\mathbf{x})$ is an n -ary predicate not in \mathbf{S} . The semantics of a Datalog[±] program is usually defined in an operational way with an algorithmic tool known as the *chase procedure* [12], which enforces the satisfaction of a set of dependencies Σ over a database D , incrementally expanding D with facts entailed via the application of the rules over D , until all of them are *satisfied*.

Example 2. Consider the following majority-based definition Σ of the relationship of control between entities in a supervision KG.

$$\text{entity}(x) \rightarrow \text{control}(x, x). \quad (1)$$

$$\text{control}(x, z), \text{own}(z, y, w), \text{sum}(w, \langle z \rangle) > 0.5 \rightarrow \text{control}(x, y). \quad (2)$$

Every entity, be it a bank, a financial intermediary, or a person, exerts control on itself (Rule 1). When an entity x controls a set of entities z , which jointly own more than 50% of an entity y , then x controls y (Rule 2). Given a KG $\mathcal{G} = \langle D, \Sigma \rangle$, a possible reasoning task may consist in computing whether a company a controls a company b , i.e., if the CQ $\text{controls}(a, b)$ yields true. ■

In the rest of the paper, we shall focus on more complex notions of company control, which, however, are based on the simple one of Example 2.

3. Designing the Schema of the KG

In our supervision KG for the Bank of Italy, we aim at supporting forms of *concerted control*, which occurs when the decision power is exerted by multiple actors that “*act in concert*”, in a coordinated manner, either publicly exposed or covert.

We illustrate our KG design methodology in two phases: *schema design*, in which the new domain elements of the KG extensional and derived extensional component are devised; *business definition*, in which the intensional component is conceived through text analysis and culminates in drafting the resulting Datalog[±] rules. These phases are narrated through the eyes of two skilled analysts:

- | | |
|--|---|
| <p>🔗 The Graph Architect designs the conceptual model that defines the Knowledge Graph schema, ensuring that it accurately represents the domain, satisfies user requirements and matches modelling principles.</p> | <p>🔗 The Rule Engineer encodes normative prescriptions and computations as logic rules guaranteeing correctness and completeness of the formulation while ensuring scalable performance of the task.</p> |
|--|---|

A presentation of the schema design methodology is provided in Section 3.1 along with a view on the Bank of Italy KG. The design journey for the case of concerted control is narrated in Sections 3.2 (extensional component) and 3.3 (derived extensional component).

3.1. Knowledge Graphs Design Methodology

Let us explore the KG modelling part. The *extensional component* of the KG contains economic actors (e.g., banks and intermediaries, companies, individuals) connected by their participation in the shareholding capital. The *intensional component* captures the business knowledge, coming from the regulation, which is applied on the elements of the extensional component to produce a *derived extensional component*.

To design the KG, we use KGMODEL, a methodology inspired by model-driven software design [13]. In KGMODEL [5], the 🔗 *Graph Architect* designs a graph schema at a high level (the so-called *super schema*), by adopting generic constructs rendered through a *conceptual visual modelling language*, namely, *Graph Schema Language* (GSL). The schema can be represented as a GSL diagram, which adopts *visual graphemes*. A list of specific visual graphemes used in the super-schema of this paper is detailed in tabular form in Figure 2, along with their role in the system.

A View on the Bank of Italy KG. Figure 3 shows an essential portion of our KG, designed using our methodology (black-coloured section). The *extensional component* revolves around the notion of PERSON, which embodies any actor in the world relevant for the supervision domain. To better represent the domain, these entities are modelled in a *total disjoint specialization hierarchy*, which distinguishes PHYSICALPERSON and LEGALPERSON; the specialization is total, as no person can exist unless it is specifically physical or legal, and disjunct, as no physical person can be a legal entity, and vice versa. A LEGALPERSON in turn can be (totally and disjointly) a COMPANY (i.e., a legal person with a shareholding capital) or a NONCOMPANY (e.g., public entities like a Ministry or a municipality). A final more specific type of company is the SUPERVISEDINTERMEDIARY: it has a special meaning for the context and peculiar traits (e.g., AUTHORITYCODE); the specialization is non-total as there exist companies that are not supervised intermediaries, and partially disjoint, since a SUPERVISEDINTERMEDIARY can be a FINANCIALINTERMEDIARY. All persons in the domain have an identifying ID. Any PERSON may have shares in a COMPANY, modelled by the OWNS relationship with a PERCENTAGE attribute. The *derived extensional component* includes the CONTROLS relationship, which connects the controlling PERSON to the controlled COMPANY.

Problem Statement for Concerted Control. To kickstart our analysis, we begin from phrasing the problem through the words of Banking Supervision regulation [10]. A relevant excerpt is depicted in Figure 4. The problem of *concerted control* as a whole can be articulated in two regulatory pillars: *acting in concert*, the regulatory concept specifying the criteria a), b), and c) to identify two or more entities that cooperate toward a common goal; *group control*, the regulatory concept that aggregates the ownership contributions of the entities acting in concert, to determine whether they jointly trigger the



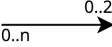
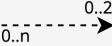






Extensional node		The general notion of a domain extensional entity. It should be used to represent any object that is characterized by its own identity, type, and distinguishing properties.
Intensional node		The general notion of a domain intensional entity.
Extensional edge		An extensional binary aggregation of two entities; it captures the existence of a relationship between domain concepts; cardinalities are encoded as min..max (natural numbers), where max can also be n, for "many". Tail cardinality indicates how many targets the relationship has; head cardinality, how many sources. For example, an individual can be parent of 0..n children and have 0..2 parents.
Intensional edge		An intensional binary aggregation of two entities.
Node/edge type		The specific type of an entity or relationship to distinguish different concepts in the domain. By convention, node types are indicated in PascalCase, edges in UPPER_SNAKE_CASE.
Total disjoint generalization		Captures a specialization→abstraction relationship existing between node entities. It is total (double head) when every instance of the parent is also an instance of a child, and disjoint (black) if the instances of the parent are instances of a single child. Example: any person is either physical or legal.
Non-total partial disjoint generalization		It is non-total (single head) when some instances of the parent are not instances of a child, and partial disjoint (white) if instances of the parent are instances of multiple children. Example: a company can be a FinancialIntermediary, a SupervisedIntermediary, both, or neither one.
Attribute		A property of an entity or a relationship; it should be used for any relevant domain object that does not have its own identity, but is part of a more general concept (e.g., name). They are depicted as lollipops attached to the object they refer to. By convention, attributes are indicated in camelCase.
Optional attribute		An optional property of an entity or a relationship.
Identifying attribute		A property used to uniquely identify the entity or relationship.

Figure 2: Tabular representation of the *super-schema* visual dictionary, with constructs used in this paper.

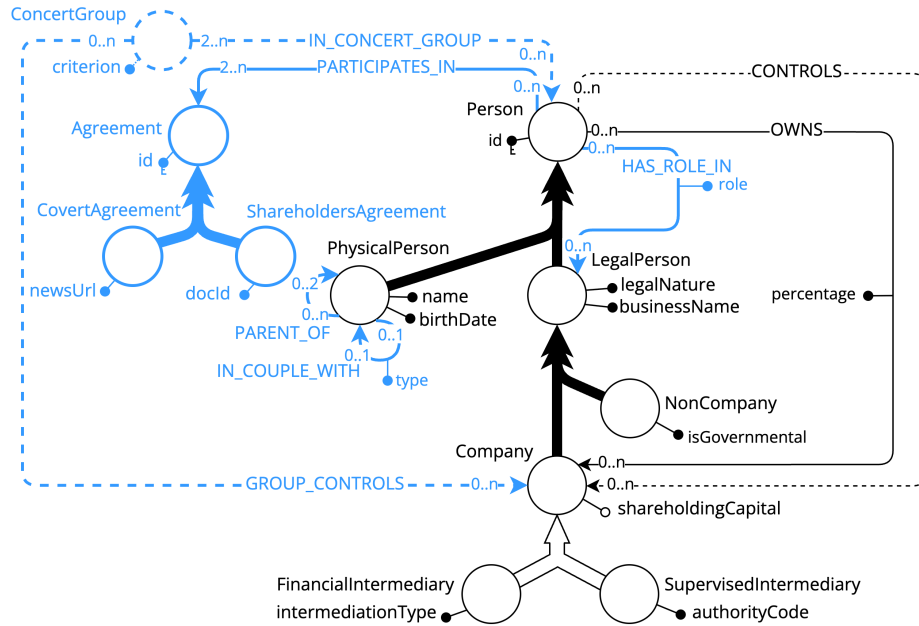


Figure 3: In black: a portion of the schema of the Bank of Italy Knowledge Graph. In blue: the obtained schema of the KG, representing the extensional and the derived extensional component of concerted control problem.

supervisory thresholds, which in turn entails additional notification and authorization procedures with the supervisory authorities [10]. We will focus on one of such thresholds, the *company control*.

3.2. Modelling the Extensional and Derived Extensional Component of the KG

We now narratively simulate the design of the Knowledge graph through each choice as taken by our *Graph Architect*, beginning with the *extensional component* of the KG.

«The assessment of whether the thresholds (...) have been reached or exceeded shall be carried out (...) with reference to the aggregate holdings in the supervised entity of the parties acting in concert (...)
The following parties are presumed to be acting in concert, even in the absence of share acquisitions and unless proven otherwise:

- a) parties adhering to an agreement, even if invalid or ineffective (...)
- b) a company and the persons who perform management and administrative functions within it;
- c) an individual and their spouse, the person to whom they are bound by a civil union or de facto cohabitation, as well as relatives and in-laws in the direct line and in the collateral line up to the second degree, including the children of the spouse or cohabiting partner.

(...) the situations referred to in points a) and b) above shall also be considered jointly.»

Figure 4: Definition of acting in concert in the regulatory framework [10]; unofficial translation of the excerpt by the authors. Portions not relevant to this paper have been replaced with “(...)”.

💡 **Agreements:** «Criterion a) of the regulation mentions agreements; the typical case would be shareholder agreements; since these agreements are domain concepts with their own identity, I can design them as `SHAREHOLDERSAGREEMENT` entities, connecting to it any entering `PERSON` through a `PARTICIPATES_IN` relationship. This approach yields tangible benefits: (i) an agreement can be entered by many persons, and any person can enter any number of agreements (ii) a `SHAREHOLDERSAGREEMENT` can have its own attributes (e.g., `DOCID`) to keep track of important information; (iii) a `SHAREHOLDERSAGREEMENT` makes it easy to navigate through connected participants; (iv) there is no impact on the existing schema.

Criterion a) mentions as well valid and invalid agreements. For example, there are cases—often not openly reported—where two or more actors coordinate to covertly acquire another company in a takeover [14]. Modelling should foresee such cases to avoid unwanted reworks in the future. As I need to design a single concept with different traits, I can use an abstraction to model a generic `AGREEMENT` entity with two specific cases: `SHAREHOLDERSAGREEMENT` and `COVERTAGREEMENT`.

Hence, I will introduce an `AGREEMENT` entity—connected to participating `PERSON` through a `PARTICIPATES_IN` relationship—with a disjoint specialization to represent `SHAREHOLDERSAGREEMENT` (e.g., characterized with a `DOCID` attribute) and `COVERTAGREEMENT` (e.g., with a `NEWSURL` attribute).»

💡 **Company Roles:** «The information that I want to preserve for criterion b) of the regulation is that a `PERSON` has a specific type of role in a `LEGALPERSON` (typically, in a company). I can: (i) create a `ROLE` entity with a `ROLE` attribute specifying role type (e.g., management), connected through a relationship to the `PERSON` holding the role and—through another relationship—to the `LEGALPERSON` where role is held; (ii) create a relationship `HAS_ROLE_IN` directly connecting the `PERSON` and the `LEGALPERSON`; (iii) introduce a specific `HAS_GUIDANCE_ROLE_IN` relationship for minimal satisfaction of regulatory requirements.

Solution (ii) appears more convenient than (i), as it allows to represent all needed information with a single construct, where (i) would use three. (iii) has several drawbacks as compared to (ii), because `HAS_GUIDANCE_ROLE_IN` poorly represents the domain: if today I choose to import only management and administrative roles, I hinder future analyses which can be done on other roles; if in the future we import other roles, then I would need to introduce many new relationships or to revert to (ii), thus changing the existing schema with consequences to the following computation stages (e.g., logic rules, visualization).

Hence, I will introduce a `HAS_ROLE_IN` relationship connecting the `PERSON` holding the role to the respective `LEGALPERSON`, with a `ROLE` attribute.»

💡 **Familiar Relationships:** «For satisfying criterion c), I want to design the relationships of `PHYSICALPERSON` entities. First, the disposition mentions spouse, implying marriage, but even civil union and de facto cohabitation. Creating an `IN_COUPLE_WITH` link between persons works well, as I can add an attribute on it, e.g., specifying couple type, and all information is represented. The regulation mentions also relatives. I can add a single catch-all `RELATED_TO` relationship with a `TYPE` attribute, abstracting all possible relationships relevant for the regulation (e.g. father-in-law), but: (i) it would capture many indirect relationships, thus opening to confusion in its usage by other applications on the KG that make

use of “relatedness”; (ii) we would lose the possibility of representing intermediate entities that justify the relationship (e.g., a mother, between grandfather and son); (iii) I would introduce a technical construct with no domain significance and hence no utility. To accurately represent our domain, I can create granular relationships (e.g., *SIBLING_OF*, *CHILD_OF*) and offload purely instrumental intermediate representations to the reasoner (e.g., *RELATED_TO*). Additionally, I should avoid introducing relationships which may be inferred, to prevent incoherences and duplications. Relationships up to the second degree are parents, grandparents and siblings, which can be derived from the foundational construct *PARENT_OF* alone.

Should I aggregate *PHYSICALPERSON* entities around a family hub node? The regulation admits affinities whose meaning is not universally mapped to “family” (e.g., cohabiting partner of the granddaughter). A *FAMILY* entity would introduce an ambiguous concept in the graph which can be confused and, as such, would end up with different variations for each domain (e.g., *SUPERVISIONCONCERTFAMILY*, *SUPERVISIONPOLITICALLYEXPOSEDFAMILY*, and so on). This excessive specialization suggests that *SUPERVISIONCONCERTFAMILY* is just another way of defining the concert group, which will have its own representation.

Hence, I will introduce a granular set of relationships between *PHYSICALPERSON* entities and avoid the *FAMILY* hub. As the regulation devises specific couples and relatives, I will introduce *IN_COUPLE_WITH* for couples—along with a *TYPE* attribute valued after couple situation— and *PARENT_OF* for relatives, while kinship and in-law relationships will be inferred from these extensional constructs.»

3.3. Modelling the derived extensional component of the KG

We now narrate the design of the derived extensional component of the KG.

☼ **Concert Groups:** «I can model group affiliation as a *GROUP* attribute of *PERSON*, populated with some *groupId*, but: (i) several situations do not warrant for meaningful group identifiers, indicating that the *GROUP* attribute is purely artificial; (ii) I would need to implement navigation using *GROUP* as a key, thus making network traversal harder; (iii) I would change existing schema for all *PERSON* nodes, which in practice encompass almost the entire population of the domain; (iv) I do not have a way to represent group attributes (e.g. *TYPE*) without adding other attributes to the *PERSON* node. Instead, I will devise autonomous *CONCERTGROUP* entities: (i) these groups are the subject of the regulation, so business users will benefit to “see” them distinctly; (ii) there is no need for an artificial group identifier; (iii) *PERSON* entities are connected to the group; (iv) group control gets a representation intuitively similar to company control. As a final consideration, the three criteria provided by the regulation involve different actors: criteria a) and b) target companies and individuals, while criterion c) targets *PHYSICALPERSON* entities. Hence, the entity that can be in a *CONCERTGROUP* is that at the highest level of modelling hierarchy, *PERSON*. Finally, the regulation defines that criteria a) and b) shall be considered jointly; while multiple options are available (e.g., introducing a supergroup), for the sake of clarity I will introduce a new mixed type for *CONCERTGROUP*.

Hence, I will model a *CONCERTGROUP* as an autonomous entity, to which a *PERSON* can belong through an *IN_CONCERT_GROUP* relationship. Since there are different criteria to identify concert, *CONCERTGROUP* will have a *CRITERION* attribute valued after its provenance: “agreement”, “role”, “family”, or “mixed”.»

☼ **Group Control:** «With the introduction of *CONCERTGROUP*, group control is straightforward to design as a relationship to a controlled entity. In this way, we can connect any group member to a controlled entity through a single-hop query.

Should I reuse the existing *CONTROLS* relationship or create a new one? In this case, *PERSON* and *CONCERTGROUP* would share a new *CONTROLS* relationship and thus they would be modelled by an abstraction, typically an *ACTOR* supernode with undefined traits, seriously compromising design integrity; second, the semantic of *CONTROLS* would change, propagating this change to existing rules and systems that use it downstream; finally, the group control is conceptually different from plain control, for example it is not transitive (effectively making it more similar to an ultimate control¹). These “smells” suggest that reusing *CONTROLS* should be avoided.

Hence, I will design group control using a new *GROUP_CONTROLS* relationship between controlling *CONCERTGROUP* and the controlled *COMPANY*, semantically equivalent to an ultimate control.»

¹The ultimate controller C of an entity A is an entity C that controls A and is not controlled by anyone.

The obtained graph schema, including all new extensional facts as well as the derived extensional facts—inferred by the rules of the intensional component that we are going to define in the next section—is shown in Figure 3 (blue-coloured section). All new entities and relationships in the domain are seamlessly integrated in the model at hand, with no changes to existing objects (black-coloured).

4. Concerted Control

In this Section, we model the *business definition* of concerted control, i.e., the *intensional component* of the Knowledge Graph. In the process, the \mathcal{Q} *Rule Engineer* encodes normative text into logic rules expressed in Datalog[±], mechanically following regulatory prescriptions and clearing natural language ambiguities to ensure correctness of the implementation. We will first encode the detection of concert groups (Section 4.1); then, we will show the computation of *group control* (Section 4.2), the ultimate goal of the task at hand. In order to enhance readability, in the rules, we will assume that the extensional component constructs introduced in the schema design phase (e.g., PARENT_OF) are input facts in database D with a corresponding predicate name in *camelCase* convention (e.g., parentOf). A brief discussion about correctness, complexity and scalability is in Section 4.3.

4.1. Concert Groups

We now encode the intensional component for the identification of concert groups.

Agreement Groups. The graph schema models agreements with constructs SHAREHOLDERSAGREEMENT and COVERTAGREEMENT; a person PARTICIPATES_IN an agreement. Given that both agreement types are equally valid for the regulation of concert groups, we have:

$$\text{shareholdersAgreement}(sa), \text{participatesIn}(s, sa) \rightarrow \exists g \text{ inAgreementGroup}(s, g, sa). \quad (3)$$

$$\text{covertAgreement}(ca), \text{participatesIn}(p, c) \rightarrow \exists g \text{ inAgreementGroup}(p, g, ca). \quad (4)$$

$$\text{person}(p), \text{inAgreementGroup}(p, g1, a), \text{inAgreementGroup}(p, g2, a) \rightarrow g1 = g2. \quad (5)$$

Rule 3 says that *if* shareholder s participates in a shareholder agreement sa *then* s participates in an agreement concert group g , in the context of sa . Rule 4 extends the notion to all covert agreements ca . Both rules introduce existentials in our reasoning, since g is a *labelled null*, i.e., an unknown value which is generated as a result of reasoning, assumed different for each activation of the rules. To ensure that actors participating in an agreement end up in the same group, Rule (5) uses EGDs: *if* a person p is in two groups $g1$ and $g2$ having entered agreement a , *then* $g1$ and $g2$ coincide.

Role Groups. For criterion b), a company and the persons who perform management and administrative functions are presumed to be acting in concert.

$$\text{hasRoleIn}(x, v, r), r = \text{"administration"} | \text{"management"} \rightarrow \text{hasGuidanceRoleIn}(x, v). \quad (6)$$

$$\text{hasGuidanceRoleIn}(x, v) \rightarrow \exists z \text{ inRoleGroup}(x, z, v), \text{inRoleGroup}(v, z, v). \quad (7)$$

We first infer a hasGuidanceRoleIn relationship by filtering administration or management role types (Rule 6). Then, for each guidance role of a person x in v there exists a concert group z such that x and v both belong to that group (Rule 7) due to the guidance of v . If there are more persons with guidance roles in the same company, they would end up into different concert groups. Figure 5 intuitively visualizes the reasoning process as a flower: different $\text{inRoleGroup}(p, g, c)$ relationships (petals) are inferred for each role that P_1 , P_1 and P_3 hold, bound to the subject company C (disk). Applying EGDs, the reasoner unifies those groups into one.

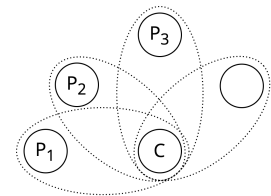


Figure 5: Unification of concert groups.

$$\text{inRoleGroup}(v, z, c), \text{company}(v), \text{hasGuidanceRoleIn}(k, v) \rightarrow \text{inRoleGroup}(k, z, c). \quad (8)$$

$$\text{company}(v), \text{inRoleGroup}(v, z1, c), \text{inRoleGroup}(v, z2, c) \rightarrow z1 = z2. \quad (9)$$

First, if k has a guidance role in v , which is in a role concert group z in the context of c , then k is in the same group (Rule 8). Hence, if company v is in two concert groups z_1 and z_2 in the context of c , then the two groups must be the same (Rule 9).

Family Groups. For criterion c), a thorough text examination can help understand how the group is devised. «An ① individual and ② their spouse, the ③ person to whom they are bound by a civil union or ④ de facto cohabitation, as well as ⑤ relatives and ⑥ in-laws in the ⑦ direct line and in the ⑧ collateral line up to the ⑨ second degree, including the ⑩ children of the spouse or ⑪ cohabiting partner.» The sentence starts with the pivoting entity ① “individual”, to which relationships ②-④ are referred; ⑩ and ⑪ instead are referred to the partner of ①. Parental relationships ⑦-⑨ are referred to the couple: the individual (⑤) and his/her partner (⑥). We first devise family ties, and then apply them to find family groups.

$$\text{in_couple_with}(p_1, p_2, t), t = \text{"marriage"} | \text{"civilunion"} | \text{"cohabitation"} \rightarrow \text{partner_to}(p_1, p_2). \quad (10)$$

$$\text{parent_of}(p_1, p_2) \rightarrow \text{related_to}(p_1, p_2), \text{related_to}(p_2, p_1). \quad (11)$$

$$\text{parent_of}(p_1, p_2), \text{parent_of}(p_2, p_3) \rightarrow \text{related_to}(p_1, p_3), \text{related_to}(p_3, p_1). \quad (12)$$

$$\text{parent_of}(p, s_1), \text{parent_of}(p, s_2), s_1 \neq s_2 \rightarrow \text{related_to}(s_1, s_2). \quad (13)$$

$$\text{spouse_of}(x, y), \text{related_to}(y, z) \rightarrow \text{in_law_to}(z, x). \quad (14)$$

We first devise the relationship $\text{partner_to}(p_1, p_2)$ specifically for cases covered by the legislation (②-④, Rule 10), since in the future more couple types can be added to database D . Then, we introduce relationship $\text{related_to}(p_1, p_2)$ (⑤) between persons p_1 and p_2 in the direct line and in the collateral line up to the second degree, upwards and downwards (⑦-⑨, for parents—Rule 11, for grandparents as parents of parents—Rule 12, and for siblings as children of the same parent—Rule 13). Relationship $\text{in_law_to}(p_1, p_2)$ links an individual p_1 and relatives p_2 of his/her spouse (⑥, Rule 14).

$$\text{partner_to}(x, y) \rightarrow \exists g \text{ inFamConcertGroup}(x, g, x), \text{inFamConcertGroup}(y, g, x). \quad (15)$$

$$\text{related_to}(x, y) \rightarrow \exists g \text{ inFamConcertGroup}(x, g, x), \text{inFamConcertGroup}(y, g, x). \quad (16)$$

$$\text{in_law_to}(x, y) \rightarrow \exists g \text{ inFamConcertGroup}(x, g, x), \text{inFamConcertGroup}(y, g, x). \quad (17)$$

$$\begin{aligned} &\text{spouse_to}(x, y), \text{parent_of}(y, c) \rightarrow \\ &\exists g \text{ inFamConcertGroup}(x, g, x), \text{inFamConcertGroup}(c, g, x). \end{aligned} \quad (18)$$

$$\begin{aligned} &\text{cohabiting_with}(x, y), \text{parent_of}(y, c) \rightarrow \\ &\exists g \text{ inFamConcertGroup}(x, g, x), \text{inFamConcertGroup}(c, g, x). \end{aligned} \quad (19)$$

$$\text{person}(x), \text{inFamConcertGroup}(x, g_1, p), \text{inFamConcertGroup}(x, g_2, p) \rightarrow g_1 = g_2. \quad (20)$$

We create a family group around a pivoting person x . First, x and his/her partner y are in a family group g (②-④, Rule 15); the same can be said for relatives of x (⑤ and ⑦-⑨, Rule 16) and his/her in-laws (⑥ and ⑦-⑨, Rule 17), the children of the spouse (⑩, Rule 18) and cohabiting partner (⑪, Rule 19). As these rules produce separate groups, we eventually unify them saying that if a person x is in groups g_1 and g_2 within the context of a family group of person p , then g_1 and g_2 coincide (Rules 20).

Mixed Groups. Finally, regulation in Figure 4 states that criteria a) and b) shall be considered jointly. This formulation is ambiguous, since there are multiple ways in which it can be applied (e.g., admitting an arbitrary combination of a)-b) groups in any number). While this is computable, especially using reasoning, it can produce unintended patterns, however out of the scope of this work. Hence, we assume that the regulation indicates the practical combination of exactly one case of criterion a) and one case of criterion b): we create a new mixed group around an entity that appears in both groups and we include all entities of those groups into it.

$$\text{inAgreementGroup}(p, _, a), \text{inRoleGroup}(p, _, c) \rightarrow \exists g \text{ inMixedGroup}(p, g, a, c). \quad (21)$$

$$\text{inMixedGroup}(p, g, a, c), \text{inAgreementGroup}(x, _, a), p \neq x \rightarrow \text{inMixedGroup}(x, g, a, c). \quad (22)$$

$$\text{inMixedGroup}(p, g, a, c), \text{inRoleGroup}(x, _, c), p \neq x \rightarrow \text{inMixedGroup}(x, g, a, c). \quad (23)$$

$$\text{inMixedGroup}(p_1, g_1, a, c), \text{inMixedGroup}(p_2, g_2, a, c), p_1 \neq p_2 \rightarrow g_1 = g_2. \quad (24)$$

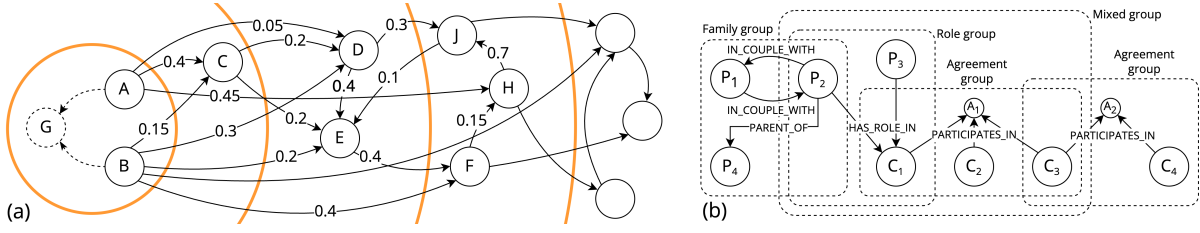


Figure 6: (a) Acting in concert unlocks possibilities of control at each recursion step, represented by an expanding control frontier centred on the concert group. Thanks to the ownerships of A and B , group G controls companies A - J downstream—and possibly others. (b) Example of all types of concert groups. For readability, relationship types crossing more edges are referred to all of them. Conforming to the regulation, the mixed group includes one role and one agreement group.

We first identify the pivoting entity of the mixed group, looking for person p , which is at the same time in an agreement group and in a role group, thus propagating the identifiers a and c of the original pivoting entities of the groups (Rule 21). Then, if a person x is in the same agreement with a person p that is in a mixed group g , then x is in g (Rule 22); the same holds for role groups (Rule 23). Finally, we unify the mixed group: if persons p_1 and p_2 are respectively in mixed groups g_1 and g_2 associated to the same agreement and role groups, then g_1 and g_2 coincide (Rule 24).

Concert Groups. Finally, the `INCONCERTGROUP` relation is derived by the union all the group assignments computed by the Rules 3–24. This can be expressed with rules of the form: “`inAgreementGroup($p, g, _$)` \rightarrow `inConcertGroup(p, g)`”. The remaining rules are omitted for brevity.

4.2. Group Control

Control over intermediaries shall be measured with reference to the aggregate holdings of the parties in a concert group: hence, we can think of it as an entity effectively acting on behalf of each of its members by inheriting their ownerships as if they were its own. This approach ensures that (i) we can leverage the existing formalization of control, (ii) we are accounting for every possible contribution of group members, and (iii) as we will see, that computation complexity is the same as company control, thus the reasoning problem is PTIME-complete. Given the implementation explained in Example 2, the majority-based control definition Σ can be easily extended to account for a concert group.

$$\text{inConcertGroup}(p, g), \text{owns}(p, y, w), \text{sum}(w) > 0.5 \rightarrow \text{groupControls}(g, y). \quad (25)$$

$$\text{groupControls}(g, z), \text{owns}(z, y, w), \text{sum}(w) > 0.5 \rightarrow \text{groupControls}(g, y). \quad (26)$$

As a base case, Rule 25 states that if a person p owns shares of y and is in group g , and the sum of that shares is more than 50%, then g controls y ; leveraging recursion, Rule 26 says that if a group g controls a set of entities z , which jointly own more than 50% of an entity y , then g controls y . A visual representation of the network effect of the recursion mechanism on control is in Figure 6a: since group G acts as if owning the shares held directly by A and B —which belong to G —then G owns more than 50% of C , controlling it; this unlocks the possibility of control over several other entities in the graph, as D can be controlled through joint shares of A , B and C ; with D , control cascades further in the system, extending to E and then down to F , H , J , opening more possibilities downstream. The *Rule Engineer* progresses further and implements the rules for `INCONCERTGROUP`, i.e., the criteria a)-c) stated by the regulation which lead to presume the existence of a concert group. Figure 6b depicts visually the type of groups. P_1 has partner P_2 , who in turn has a child P_4 ; according to criterion c), P_1 , P_2 and P_4 are in a family concert group. P_2 and P_3 have a role in company C_1 , hence P_2 , P_3 and C_1 are in a role concert group (criterion b)). C_1 , C_2 and C_3 participate in an agreement A_1 , and so they are in an agreement group, and so do C_3 and C_4 , in another group due to agreement A_2 (criterion a)). C_1 is in two groups: as the regulation states that criteria a) and b) shall be considered jointly, then there is a coordination among all involved parties P_2 , P_3 , C_1 , C_2 and C_3 . Also C_3 and P_2 are in two groups, but not under prescribed criteria (C_3 is in two agreement groups, P_2 in role and family groups).

4.3. Correctness, Complexity, and Scalability

We intuitively reflect on the correctness of our encoding of “concerted control” and briefly discuss the computational complexity of the problem, based on the formal analysis of proposed rules (Section 5 will focus instead on actual empirical simulations). For this, we structure the setting into two ontological reasoning tasks (see Section 2): *identification of concert groups* and *computation of group control*.

Correctness. The correctness of our VADALOG formalization descends by construction from the faithful encoding of the regulatory semantics. The rules for *concerted group identification* capture the propagation of control-relevant relationships between individuals and companies through a principled use of labelled nulls and equivalence rules. The EGDs reflect the legal principle that individuals connected via shared influence over a company should be treated as a single concerted group. Similarly, the rules for *concerted group control* extend standard control definitions by introducing synthetic nodes that aggregate ownership, aligning with how the regulation assesses indirect or collective control. The logic-based encoding ensures that the resulting groupings and control structures correspond exactly to the entities and relationships identified in legal interpretations.

Computational Complexity. The *identification of concert groups* (Rules 3–24) is encoded with a fragment of Datalog[±] known to guarantee decidability and PTIME-complete data complexity [11] of ontological reasoning. For example, the process of inferring role groups (Rules 6–9) assigns a fresh labelled null to each person–company pair in which the person holds a relevant role. These nulls are then propagated through people linked to the same company. If a person–company pair is assigned different group identifiers, the EGDs are applied to merge them. This procedure ensures termination and tractability: labelled nulls are introduced in a bounded and deterministic way, and propagated only along finite relational paths, with no arbitrary generation. Conceptually, the computation corresponds to identifying connected components in an undirected graph whose nodes are labelled nulls and whose edges represent equivalences induced by the EGDs. Since the number of nulls is finite and merges are transitively closed, the process is guaranteed to terminate and can be implemented efficiently using union–find data structures.

The *computation of group control* (Rules 25–26) can be viewed as a generalization of the classical *company control* problem, whose data complexity is also known to be PTIME-complete [8]. The extension consists in augmenting the ownership graph with artificial nodes that represent concerted groups. Each group is treated as an abstract company controlling—directly or indirectly—other companies based on aggregated ownership percentages.

The number of labelled nulls introduced to represent groups (i.e., artificial nodes) is polynomial in the size of the input (at most one labelled null is introduced for each KG relation). Therefore, the overall size of the augmented graph remains polynomial in the input size. As a result, the same fixpoint algorithm used for computing company control can be applied without modification. Hence, the concerted control computation inherits the same termination guarantees and polynomial-time complexity.

Scalability. Despite its theoretical tractability, computing concerted control on large KGs poses significant scalability challenges in practice. This is due to the potentially quadratic number of group–company pairs that must be evaluated. For instance, in a KG with 10,000 entities, there could be up to 100 million possible $\text{groupControls}(g, c)$ pairs. This explosion in the number of candidate control relations requires efficient distributed evaluation strategies.

5. Concert on Air

In this section, we present how the VADALOG system implements concerted control in a scalable manner. We first describe the distributed evaluation algorithm employed by VADALOG to solve the problem at scale, then, we discuss the performance of the system on a real-world KG.

To address the scalability challenges of the concerted control problem we adopt a practical and scalable approach that leverages the distributed reasoning capabilities of the VADALOG system [9].

Distributed Evaluation Model. To describe the evaluation model of the VADALOG system, we consider a shared-nothing architecture with independent workers that share no memory and can communicate via messages. We call *partition* a portion of memory that can be accessed by a single worker and we call *shuffling* the act of exchanging messages (i.e., facts) among workers. VADALOG’s computational model is based on the *Map Reduce* paradigm where every operation on predicates (i.e., relations containing facts) can be executed in a distributed fashion following three steps: (i) assigning a key to each fact in the predicate; (ii) shuffling the facts into the same partition based on equivalent keys, and (iii) applying a user-defined operation to the facts in the same partition to derive new facts.

The evaluation of VADALOG rules is based on a variant of *Semi-Naive Evaluation* (SNE) [15], adapted for VADALOG rules and designed for distributed execution. SNE is a bottom-up reasoning technique that incrementally materializes all derivable facts by repeatedly applying the rules in Σ to an input database D until a fixpoint is reached. In a distributed setting, VADALOG extends this approach to account for labelled nulls: at each iteration, only facts that are not structurally identical up to renaming of labelled nulls are considered new, and added to the corresponding predicate. This condition allows for the termination of every VADALOG set of rules evaluation.

To scale the reasoning process, VADALOG structures the SNE as a sequence of MapReduce steps. The input to the pipeline is the set of base facts in D and each step performs the following actions:

- Facts are shuffled across workers according to a partitioning criterion specific to the transformation required by the rule being applied.
- Each partition applies a user-defined transformation function that encodes rule logic (e.g., joins, projections, aggregations).
- The result is post-processed (e.g., to eliminate duplicates) and re-shuffled for the next stage.

If the rule set Σ includes recursion, a subset of these steps is repeated until no new facts are produced.

Rules Evaluation Examples. To concretely illustrate the evaluation process, we present example-based executions of Rules 25–26 and 6–9, as implemented in Algorithms 1 and 2. These algorithms rely on a sequence of relational transformations and auxiliary functions designed to efficiently compute fixpoint and enforce labelled nulls equalities during rule evaluation.

- *Projection, join, union, difference and labelled nulls creation:* $\Pi_{[]}^V$ denotes a *projection*, optionally involving the generation of *labelled nulls* for existential variables. The superscript V indicates that the operation is performed under the semantics adopted by VADALOG, where facts are considered duplicates if they are identical up to renaming of labelled nulls. \bowtie denotes the standard *natural join* between predicates, \cup denotes *union*, while ∇^V indicates a set difference under the semantics used by VADALOG, meaning that duplicated facts are filtered out during the difference operation.
- *Aggregations:* Aggregation operations (e.g., *sum*) are expressed using SQL-style *group-by with condition* notation, written as $AGG_{[g]}^{a,c}(P)$, where g are the group-by variables, a is the aggregation function, c is a constraint on the aggregated values (e.g., $\text{sum}(w) > 0.5$) and P is the predicate name on which the aggregation is applied.
- *Auxiliary functions:* `mergeTransitively` computes the transitive closure of equivalence classes of labelled nulls induced by the EGD enforcements. The function `updatePredicate` replaces labelled nulls in a predicate according to an equivalence mapping returned by `mergeTransitively`. `updateGroups` abstracts the incremental fixpoint update of a recursive predicate by applying new aggregated facts to the current state and returning only the newly derived facts.

Each of these operations can be naturally implemented using a MapReduce-style and in-memory execution model, as supported by distributed frameworks like *Apache SparkSQL* [16]. The pseudocode of Algorithms 1 and 2 follow. Note both of them use fixpoint loops (lines 5-9, 3-6, respectively), which encode the basic idea of SNE, where VADALOG rules are applied as long as they produce facts.

Algorithm 1 Evaluation of VADALOG Rules for Role Group Definition (Rules 6–9)

```
1: hasGuidanceRoleIn  $\leftarrow \Pi_{[x,v]}^V(\text{hasRoleIn}(x, v, r) \mid r \in \{\text{"administration"}, \text{"management"}\})$   $\triangleright$  Rule 6
2: inRoleGroup  $\leftarrow \Pi_{[x,z]}^V(\text{hasGuidanceRoleIn}(x, v)) \cup \Pi_{[v,z]}^V(\text{hasGuidanceRoleIn}(x, v))$   $\triangleright$  Rule 7
3:
4:  $\Delta$ inRoleGroup  $\leftarrow$  inRoleGroup  $\triangleright$  Initialize delta for recursive propagation (Rule 8)
5: repeat
6:   new  $\leftarrow \Pi_{[k,z,c]}^V(\Delta$ inRoleGroup( $v, z, c$ )  $\bowtie$  company( $v$ )  $\bowtie$  hasGuidanceRoleIn( $k, v$ ))
7:    $\Delta$ inRoleGroup  $\leftarrow$  new  $\nabla^V$ inRoleGroup  $\triangleright$  Keep only new facts
8:   inRoleGroup  $\leftarrow$  inRoleGroup  $\cup \Delta$ inRoleGroup  $\triangleright$  Accumulate derived facts
9: until  $\Delta$ inRoleGroup =  $\emptyset$ 
10:
11: eqPairs  $\leftarrow \Pi_{[z_1,z_2]}^V(\text{inRoleGroup}(v, z_1, c) \bowtie \text{inRoleGroup}(v, z_2, c) \bowtie \text{company}(v))$   $\triangleright$  Rule 9
12: equiv  $\leftarrow$  mergeTransitively(eqPairs)  $\triangleright$  Merge labelled nulls into equivalence classes
13: inRoleGroup  $\leftarrow$  updatePredicate(inRoleGroup, equiv)  $\triangleright$  Replace merged nulls
14: return inRoleGroup
```

Algorithm 2 Evaluation of VADALOG Rules for Concerted Group Control (Rules 25–26).

```
1: groupControls  $\leftarrow \text{AGG}_{[g,y]}^{\text{sum}(w)>0.5}(\text{inConcertGroup}(p, g) \bowtie \text{owns}(p, y, w))$   $\triangleright$  Rule 25
2:  $\Delta$ control  $\leftarrow$  groupControls  $\triangleright$  Set delta for recursion
3: repeat  $\triangleright$  Fixpoint loop for Rule 26
4:   aggOwnNext  $\leftarrow \text{AGG}_{[g,y]}^{\text{sum}(w)>0.5}(\Delta$ control( $g, z$ )  $\bowtie$  owns( $z, y, w$ ))  $\triangleright$  Infer indirect control
5:    $\Delta$ control  $\leftarrow$  updateGroups(groupControls, aggOwnNext)  $\triangleright$  Update groupControls and return the delta
6: until  $\Delta$ control =  $\emptyset$   $\triangleright$  Stop when no new facts are produced
7: return groupControls  $\triangleright$  Final group control
```

Experimental Evaluation. We conducted our experimental evaluation in VADALOG.

Setup. We used the supervision KG of the Bank of Italy. The extensional component used in the experiments comprise 182M nodes and more than 130M ownership and role relationships and has been built integrating several enterprise data assets, internal and external. The execution environment is a Spark cluster (version 3.3.2), with 4 distributed executors, each with 30 GiB of RAM.

We focused on the evaluation of Rules 6–9, which compute concert groups according to criterion b) of the regulation (dubbed as *identification of concert groups* like in Section 4.3), and Rules 25-26, which compute group controls (dubbed as *computation of group control*). In particular, the scalability of Algorithms 1 and 2 was evaluated by measuring the execution time with a growing level of parallelism (i.e., the number of cores per executor). The results have been averaged over three evaluations.

Results. The *identification of concert groups* produced more than 5k role groups, while *computation of group control* task discovered more than 2.2M groupControls(x, y) relationships.

The execution times, shown in Figure 7, consistently remained below 25 minutes, thus making the computation of concerted control feasible on the real graph. We observe a linear speed-up as we increase the number of cores assigned to each executor. This trend highlights the parallelization effectiveness in distributing the workload. Going over 16 cores per executor, the performance degrades, likely because of the data transfer overhead due to abrupt broader distribution and reduced memory locality. These results—executed on our internal distributed computing infrastructure—indicate that the rule-based approach is viable on the supervision KG, achieving runtimes compatible with the requirements of the industrial use case.

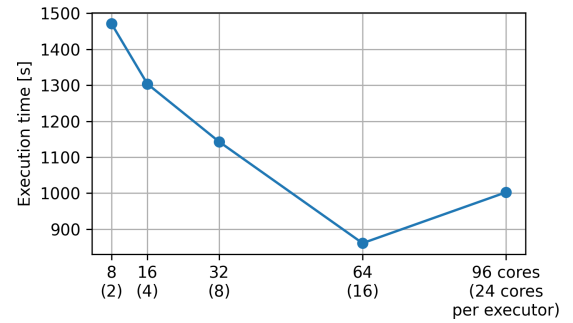


Figure 7: Execution times for computing concerted control on the real-world KG.

6. Conclusion

We presented a logic-based methodology for tackling regulation-intensive challenges in the legal domain. We used the motivating example of “acting in concert” in banking supervision, appreciating the shift of paradigm from the soft approach of natural language encoding of the regulations to the hard process of logic-based formalizations. Logic-based approaches offer a concrete tool to pursue regulatory simplification through optimization and restructuring. Logic-based Knowledge Graphs offer a natural and robust framework for legal reasoning on corporate structures. Our methodology, KGModel, introduces a graph super-schema that serves as an extensible and system-agnostic domain abstraction layer. The declarative formalisation of the business case through logic rules enables transparent, precise, and scalable compliance checks. Our experience at the Bank of Italy shows that such methodology offers a foundation for designing systems that are aligned with legal intent, responsive to change, and capable of supporting the explainability and accountability that modern governance demands.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

- [1] T. Baldazzi, L. Bellomarini, E. Sallinger, Reasoning over Financial Scenarios with the Vadalog System, in: EDBT, 2023.
- [2] L. Bellomarini, D. Fakhoury, G. Gottlob, E. Sallinger, Knowledge Graphs and Enterprise AI: the Promise of an Enabling Technology, in: ICDE, 2019.
- [3] A. Calì, G. Gottlob, T. Lukasiewicz, A General Datalog-based Framework for Tractable Query Answering over Ontologies, in: PODS, 2009.
- [4] L. Bellomarini, E. Sallinger, G. Gottlob, The Vadalog System: Datalog-based Reasoning for Knowledge Graphs, PVLDB (2018).
- [5] L. Bellomarini, A. Gentili, E. Laurenza, E. Sallinger, Model-independent Design of Knowledge Graphs - Lessons Learnt From Complex Financial Graphs, in: EDBT, 2022.
- [6] L. Robaldo, F. Pacenza, J. Zangari, R. Calejari, F. Calimeri, G. Siragusa, Efficient Compliance Checking of RDF Data, *Journal of Logic and Computation* 33 (2023) 1753–1776.
- [7] J. Anim, L. Robaldo, A. Z. Wyner, A SHACL-Based Approach for Enhancing Automated Compliance Checking with RDF Data, *Information* 15 (2024) 759.
- [8] A. Gulino, S. Ceri, G. Gottlob, E. Sallinger, L. Bellomarini, Distributed Company Control in Company Shareholding Graphs, in: ICDE, 2021.
- [9] L. Bellomarini, D. Benedetto, M. Brandetti, E. Sallinger, A. Vlad, The Vadalog Parallel System: Distributed Reasoning with Datalog+/-, *Proc. VLDB Endow.* 17 (2024) 4614–4626.
- [10] Banca d’Italia, Disposizioni della Banca d’Italia in Materia di Assetti Proprietari di Banche e Altri Intermediari, 2022. URL: <https://shorturl.at/D72NB>, 26 luglio 2022, Last accessed: 1 June 2025.
- [11] L. Bellomarini, D. Benedetto, M. Brandetti, E. Sallinger, Exploiting the Power of Equality-Generating Dependencies in Ontological Reasoning, *Proc. VLDB Endow.* 15 (2022) 3976–3988.
- [12] D. Maier, A. O. Mendelzon, Y. Sagiv, Testing Implications of Data Dependencies, *ACM TODS* 4 (1979) 455–469.
- [13] E. Evans, *Domain-Driven Design: Tackling Complexity in the Heart of Software*, Addison-Wesley, 2004.
- [14] 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* 141 (2022) 102073.
- [15] S. Abiteboul, R. Hull, V. Vianu, *Foundations of Databases*, Addison-Wesley, 1995.
- [16] S. Salloum, R. Dautov, X. Chen, P. X. Peng, J. Z. Huang, *Big data analytics on apache spark*, 2016.