A Two-tiered Approach for Knowledge Reasoning

Fernando Antonio Dantas Gomes Pinto[1], Jefferson de Barros Santos[2], Sérgio Lifschitz[1], Edward Hermann Haeusler[1]

[1]Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio)
Departamento de Informática
Rio de Janeiro – RJ – Brazil

[2]Fundação Getúlio Vargas (FGV)
Rio de Janeiro – RJ – Brazil

{fpinto, sergio, hermann}@inf.puc-rio.br, jefferson.santos@fgv.br

Abstract. *This article proposes an explicit two-tier approach for reasoning and querying on Knowledge Bases. It separates the conceptual fraction from the model elements, enabling clear connections between the components. This method may be more effective than known one-tier solutions. Besides a theoretical view, we explore a case study on actual data about legal knowledge to illustrate how well our strategy may work in practice.*

1. Introduction

Integrating knowledge and data in models often leads to conflicts between theoretical and universal assertions and the underlying logic of the models. For example, relational databases use models of the finite domain logic $L_{<\omega}$, while first-order logic (FOL) expresses integrity constraints. No FOL formula can express finiteness, while every database instance is finite. On the other hand, the logic $L_{<\omega}$ of the finite models is non-recursively enumerable and a non-conservative extension of FOL. In other scenarios, Knowledge Graphs (KG) represent data and knowledge in a rather ad-hoc way. OWL-based ontologies combine the TBOX, theoretical knowledge (KB), and the ABOX, which is its model-based part. As in the relational model, the logical reasoning tasks concerned the ABOX are different from those regarded to the ABOX. TBOX reasoning focuses on the schema or structure of the knowledge base (concepts and relationships), while ABOX reasoning focuses on the specific instances and their properties. The similarity with the above-mentioned relational database case is perfect. TBOX reasoning provides conclusions about the conceptual hierarchy and relationships, while ABOX provides conclusions about individual membership and properties. Their computational complexity is also very different, as is the case with relational databases. In both cases, relational and ontologies, the exclusively model-based reasoning is computationally easier than the conceptual one[1].

We shortly analyze and depict two main problems with this rather ad-hoc way of integrating the conceptual and the model-based or data-driven KB part: (1) consistency and completeness in Knowledge Representation (KR) and (2) unnecessary higher reasoning complexity on the represented knowledge. We propose in this paper a two-tiered approach for reasoning and querying on a KB turning the integration of the conceptual (part 1) of the KB with its model (data-driven, part 2) aspect. The explicit linking between

[1]The first is related to model-checking while the latter is related to logical consequence verification, i.e., LogSPACE vs PSPACE.

these parts allows us to connect the answer set of the queries made at the theoretical level to run as model-theoretical queries at the model-based level. We argue that this strategy is more efficient than one-tier logical solutions, including a mix of logic, data, and knowledge representation, such as KGs or Description Logic systems. Besides a theoretical view, we also explore a practical case study on legal knowledge representation obtained from public gazettes and the Law from the Union, State, or City.

2. Logical Models and Knowledge Representation

Modeling data along with a knowledge base involves structuring information to facilitate effective reasoning and retrieval. Some common approaches are: **(1) Relational Data Model** is a widely used approach to represent data through mathematical relations (or tables) in a database system. Attributes describe properties of entities and their corresponding relationships, forming the basis for database design. The underlying logic of the relational DBs is the First-Order Logic (FOL). The logical schema of a DB is the FOL non-logical language used, i.e., the set of predicate or relational symbols, functional and constant symbols. Predicates of arity one is used to express concepts, and those with arity two, to express the relationships or queries. FOL structures for a non-logical FOL language L are the DBs. Each structure can be regarded as an FOL model that also has to satisfy the integrity constraints (IC) of the DB. The IC is a set of FOL formulas that any DB instance or model should satisfy. For example, we need to express that any person should have one and only one mother in a schema about family relationships in the set of IC; **(2) Ontologies** formally represent knowledge by defining concepts, relationships, and axioms within a domain. Ontologies use a structured format to capture domain-specific knowledge and facilitate the sharing and integration of data. Technologies like the Ontology Web Language (OWL) support creating and managing ontologies. Description Logic (DL) supports reasoning on the ontologies if OWL-DL is used to express and prove the properties of interest that should hold in the ontology. This approach is also based on the dichotomy between conceptual terminology, i.e., theoretical knowledge or TBOX; instance-based or model-theoretical knowledge; **(3) Semantic Web Technologies** such as Resource Description Framework (RDF) and SPARQL Protocol and RDF Query Language (SPARQL), enable the representation of data and knowledge on the web in a machine-understandable format. They allow linking, querying, and reasoning over distributed and interconnected data sources; **(4) Graph-based models**, like property graphs and knowledge graphs, represent data as nodes (entities) connected by edges (relationships). These models capture complex relationships and hierarchies in data, making them suitable for representing knowledge bases with rich interconnections.

3. Dealing with Two Logic Formalisms

This paper discusses the main problems in integrating data and knowledge for logical systems such as those cited in Section 2. We use theoretical or universal assertions to refer to propositions that belong to a theory and model-based propositions when referring to facts held in a model. For example, the proposition that any person has one and only one father is theoretical, while the proposition that says that Jos is Maria's father is part of the corresponding model.

Following the literature on logic and epistemology of science, we denote by theory[2] any set of propositions Δ, expressed in some logic L that is closed by logical consequence, i.e., $\Delta = Cn(\Delta) = \{\varphi : \Delta \models_L \varphi\}$, where $\models_L$ is the logical consequence in the logic L and φ is an arbitrary proposition of L. In relational DBs, the underlying logic that expresses the theoretical and universal assertions does not fit very well with the underlying logic of the models. The logic of the queries is FOL, while the logic of the ICs and other theoretical propositions is the logic of finite domains, named here as $FOL_{<\omega}$. However, $FOL_{<\omega}$ does not recursively enumerate its valid assertions, and FOL recursively does it. In practice, there are rare cases where a theoretical assertion must be proved. They are generally model-checked together with the model-based assertions related to the DBs. Logically speaking, the most frequent theoretical integration with model-based assertions may be inconsistent or incomplete. So, they should be kept separate, even when sharing knowledge about the same specific domain.

Ontologies and OWL-based Semantic Web Technologies use TBOX to represent the theoretical assertions and formalization of the domain-specific ontology. The model-based assertions are formalized using nominals in the ABOX. We know that the computational complexity of TBOX reasoning in the acyclic case is PSPACE-complete and, with the general case, is EXPTIME-complete and hence intractable, see [Baader et al. 2007] for the details cited here we find a *jungle* of fragments of Description Logic with different complexities; they go from NP-complete to NEXPTime complexity.

The practical validation of ontologies with this union-only way of integrating model-based and theoretical assertions is not very efficient. The usual way of dealing with ABOX reasoning has been quite similar to FOL; the TBOX is instantiated with the elements in the ABOX, producing a greater TBOX, and the validation goes on this new TBOX. It is pretty much like checking satisfiability in first-order logic. Our approach advocates that a rather general and abstract query should be checked as a logical consequence regarding the TBOX, and in the very case that it is not a logical consequence, the abstract and formal counter-example is used to define a query that runs on the ABOX, providing, if it is the case, an answer set at a much lower computational complexity than the usual approach of using an augmented ABOX+TBOX. Non-OWL-DL KB having model-based assertions and graph-based KBs (KG) have a similar feature with the addition of model-based assertions, which multiplies by the size of these assertions the complexity of validation of the set of theoretical assertions, resulting in a very high complexity.

4. A Two-Tiered Logic KB Approach

When considering only the validation in the TBOX, i.e., with theoretical assertions only, the complexity is smaller than the overall complexity of TBOX+ABOX, since the size of the TBOX+ABOX is strictly higher than the TBOX size only. Besides that, to reason with formulas that uses individuals from the ABOX together with the concepts of the TBOX is known to be of a higher complexity. Similar behaviours occur with FOL and FLogic, the logic behind the RDF-Schema theory. Our approach keeps the logical manipulation of the conceptual theory and the model-based reasoning separate.

Using a formal query $\alpha(p_1, \ldots, p_k)$, i.e., a query written using formal parameters p_i, $i = 1 \ldots p_n$ represented by fresh constants. We then proceed to check whether

[2]This definition is formally taken as a logical theory. To consider mathematical theories, we should include mathematical axioms in the set of propositions.

$TBOX \vdash \alpha(p_1, \ldots, p_k)$. If the answer is positive, any substitution of the parameters by individuals in the ABOX also holds. On the contrary, if the answer is negative, the checking procedure will return an answer set indicating the atomic concepts holding and failure for each parameter regarding some atomic concept; this is what we call properly an answer set. This answer set will serve as a basis for performing a query that will run against the ABOX and produce the actual counter-examples in the model-based part of the theory. Observe that the query is made on a smaller set of formulas, the ABOX only, by a more efficient algorithm, the model-checker. Thus, our approach is computationally more efficient than running the reasoning on the ABOX+TBOX KB. To a logician, we would explain that we are using a form of Craig interpolation theorem [Craig 1957] to reduce the complexity of the reasoning. We consider a Knowledge Base with a Model representing Data, also named KB with Data, as a pair (A, Q) where A is a theory presentation, i.e., a set A of formulas of some logic L_T, representing the theory $Cn_{L_T}(A)$[3], and Q is a model in L_M, where L_M is the logic where the models come from. Q is a model of A, i.e. $Q \in Mod(A)$. $Cn(A)$ is the usual logical consequence operator. We do not need to specify Cn formally since we consider the mappings Mod and Th, and $Cn(A) = Th(Mod(A))$. To be more flexible, we might consider a mapping ϕ from non-logical languages in L_M to non-logical languages in L_T and the condition $\phi_\Sigma(Q) \in Mod_\Sigma(A)$. We also need a validation procedure P_T that, whenever we give a Σ-formula[4] α from L_T, P_T verifies whether $A \in Cn(\Sigma)(A)$ or not, providing the counter-examples belonging to the set CE_T, if $A \notin Cn(\Sigma)(A)$. CE_T is the set of all counter-examples to formulas in L_T. To complete the formalization, we need a mapping q_Σ from CE_T into queries (formulas with free variables in L_M) over Q, such that, for each $\gamma \in CE_T(A, \alpha)$, $q_\Sigma(\gamma) \not\subseteq Q$. The structure is then formed by $\langle (A, Q), P_T, q_\Sigma \rangle$.

A computational mechanism has been devised that integrates an SAT Solver with a query generator, henceforth referred to as the "framework". We consider a logic tool based on Description Logic L_{law} adapted to legal ontologies. We use a *Tableau System* [Fitting 1969] developed for reasoning on legal ontologies [Haeusler et al. 2010] [Alkmim et al. 2022]. The model-based part is represented in terms of a *graph* database structure. Consequently, we consider the Knowledge base with Model data in the pair (L_{law}, $Graph_m$) and the instance of our approach as $\langle (L_{law}, Graph_m), P_T, q_\Sigma \rangle$. This instance of our proposed approach will perform an audit process in a KG of the legal domain (laws). We illustrate with an example where and how it interconnects (integrates) the logic of L_{law} and a Graph DB to have inferences in a two-logic-based way, as explained in 4. Initially, we defined a set of Competency Questions (CQ)[5] extracted from various laws (municipal, state, and federal) together with business specialists in a chosen City in Brazil, available at the link: Competence Questions Report. These CQs are formulas in L_{law} representing the legislation's compliance rules over *real problems and data*. These rules define what should be valid in a knowledge base regarding its TBOX, the domain of interpretation. Thus, we can define the CQ as $\langle C, \sigma \rangle$ such that C is a question expressed in L_{law} language and σ is an answer to this question expressing counter-examples generated by specialists through a tableaux reasoner. The complete practical experiment of this case study is available in the link: Practical Exper-

[3]The operator Cn_{L_T} is $\{\alpha \in L_T : A \vdash \alpha\}$ as noted in the background section.
[4]Formulas written in the non-logical language of L_T.
[5]They are used in methodologies for validating ontology functional requirements[Bezerra et al. 2013].

iment Report. Its objective is to evaluate the quality of the framework in two aspects: *accuracy* in generating counter-models and *query coverage*. Both metrics summarize our approach's ability to make the connection between models (two-tier approach for reasoning). We are concerned with treating only the concepts and roles extracted from the L_{law} formulas and directly building them as elements of KG. We use a transpiler[6] to translate the grammar of the counter-model and transform it into SPARQL. In practice, L_{law} can be mapped to *RDF properties*, and *nominal "concepts"*, present in L_{law}, to *RDF object "values"*. Then, from RDF to SPARQL is also a direct transformation.

This instance of our approach uses three basic templates that helped construct SPARQL queries for graph databases: (1) The *first template* is applied according to the number of open branches in the tableau. To do this, the framework starts the construction process by associating each branch with a single complete SPARQL structure (`SELECT` clause, *projection variables* and `WHERE` clause). For example, if Tableau generates two counter-examples, this template will be used twice. In other words, we will have a query for each open branch; (2) The *second template* is for cases where the counter-model only presents variables signed with $\mathcal{T}$. The query is constructed by applying these variables to each `WHERE` clause of the subset of queries presented in the first template; (3) The *third template* is for cases where the counter-model presents variables signed with $\mathcal{F}$, so the query is built using the `FILTER NOT EXISTS` in the parameters of the `WHERE` clause. Then, this template is associated with the rules of the first and second templates. The case study presents an example from a federal law in Brazil, No. 9,717 of November 27, 1998. In Article 1, the law establishes specific social security rules (RPPS) for public employees of the Union, the States, the Federal District, and the municipalities. $\vdash_{iALC} (w : (\forall tipoFunc.SERVIDOR \sqcap \neg(\forall temVinculo.EST)) \longrightarrow\!\!\!\!\!\!\dashv (\exists temVinculo.CEL \sqcap (\exists temPrev.RGPS)))$

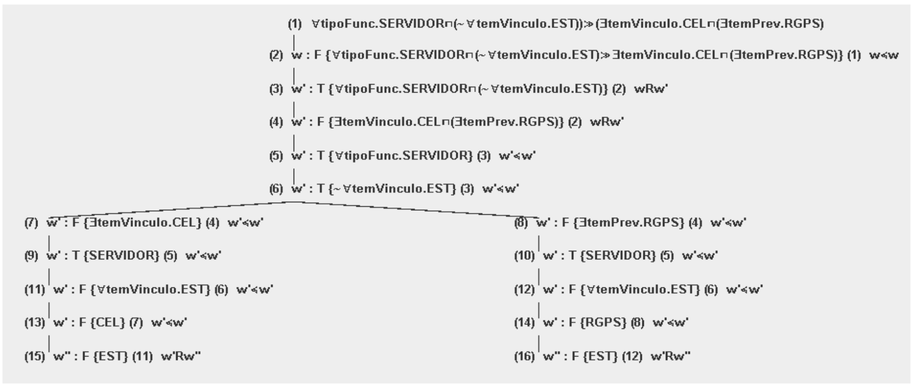

(1) ∀tipoFunc.SERVIDOR⊓(~∀temVinculo.EST))⪢(∃temVinculo.CEL⊓(∃temPrev.RGPS)
(2) w : F {∀tipoFunc.SERVIDOR⊓(~∀temVinculo.EST)⪢∃temVinculo.CEL⊓(∃temPrev.RGPS)} (1) w⋖w
(3) w' : T {∀tipoFunc.SERVIDOR⊓(~∀temVinculo.EST)} (2) wRw'
(4) w' : F {∃temVinculo.CEL⊓(∃temPrev.RGPS)} (2) wRw'
(5) w' : T {∀tipoFunc.SERVIDOR} (3) w'⋖w'
(6) w' : T {~∀temVinculo.EST} (3) w'⋖w'
(7) w' : F {∃temVinculo.CEL} (4) w'⋖w'
(8) w' : F {∃temPrev.RGPS} (4) w'⋖w'
(9) w' : T {SERVIDOR} (5) w'⋖w'
(10) w' : T {SERVIDOR} (5) w'⋖w'
(11) w' : F {∀temVinculo.EST} (6) w'⋖w'
(12) w' : F {∀temVinculo.EST} (6) w'⋖w'
(13) w' : F {CEL} (7) w'⋖w'
(14) w' : F {RGPS} (8) w'⋖w'
(15) w'' : F {EST} (11) w'Rw''
(16) w'' : F {EST} (12) w'Rw''

Figure 1. An open tableau.

The open tableau 1 has a counter-model with two counter-examples, represented by two branches. The left branch contains the counter-example of nodes 15, 13, and 9 IDs. The right counter-example of nodes 16, 14, and 10 IDs. The framework generates one query for each branch in cases with more than one counter-example. Given page limitations, we show one branch as see Listing 1 (right branch). When the counter-example contains the `F` (false signal) of a nominal (nodes 15, 13, 16, and 14), the framework decides a template with a non-existence filter. The `FILTER NOT EXISTS` allows filtering results based on whether certain triple patterns do not exist in an RDF graph.

[6]A term describing the conversion of code from one language to another [Keith D. Cooper 2011].

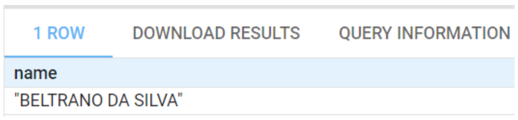
```
PREFIX foaf:  <http://xmlns.com/foaf/0.1/>
SELECT ?name
WHERE { {
SELECT ?name
WHERE {
?x foaf:tipoFunc  'SERVIDOR'  .
FILTER NOT EXISTS {?x foaf:temVinculo  'EST'}
?x foaf:name ?name  .
      } } {
SELECT ?name
WHERE {
?x foaf:tipoFunc  'SERVIDOR'  .
FILTER NOT EXISTS {?x foaf:temPrev  'RGPS'}
?x foaf:name ?name  .
      } } }
```

Code Listing 1. SPARQL query generated (right branch) by framework.

```
<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF
  xmlns:foaf="http://xmlns.com/foaf/0.1/"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <foaf:Vinculos rdf:nodeID="Nf1002b3c57c3f3225c88f6XX">
        <foaf:name>FULANO DA SILVA</foaf:name>
        <foaf:tipoFunc>SERVIDOR</foaf:tipoFunc>
        <foaf:temVinculo>EST</foaf:temVinculo>
        <foaf:temPrev>RGPS</foaf:temPrev>
    </foaf:Vinculos>
   <foaf:Vinculos rdf:nodeID="N22e140cbdf80cb49b537834YY">
        <foaf:name>SICRANO DA SILVA</foaf:name>
        <foaf:tipoFunc>SERVIDOR</foaf:tipoFunc>
        <foaf:temVinculo>EST</foaf:temVinculo>
        <foaf:temPrev>RPPS</foaf:temPrev>
    </foaf:Vinculos>
    <foaf:Vinculos rdf:nodeID="N6388d7508deb49e585c9b699d9ad03DD">
        <foaf:name>BELTRANO DA SILVA</foaf:name>
        <foaf:tipoFunc>SERVIDOR</foaf:tipoFunc>
        <foaf:temVinculo>CEL</foaf:temVinculo>
        <foaf:temPrev>RPPS</foaf:temPrev>
    </foaf:Vinculos>
</rdf:RDF>
```

Code Listing 2. XML/RDF file.

Listing 2 shows an XML/RDF instance of the hypothetical functional relationships. The cases of non-compliance are `FULANO DA SILVA` and `BELTRANO DA SILVA` where the values of the predicates for `<foaf:temVinculo>` and `<foaf:temPrev>` are not compatible. Furthermore, the formula looks for cases where employee not is `EST` (Estatutário) and `RGPS`. Therefore, only `BELTRANO DA SILVA` (Figure 2) is retrieved by the query. In summary, the



```
name
"BELTRANO DA SILVA"
```

Figure 2. Result query generated by AllegroGraph.

execution of the case study presented 100% in *accuracy* and 100% in *query coverage* of the framework.

5. Conclusion and Future Works

Logical formalisms, like first-order and description logic, express structured knowledge and relationships within a KB. By encoding knowledge in a logical form, systems can perform automated reasoning and inference to derive assertions valid from existing data. This is the task of deductive databases that experienced a more challenging escape for validating within lower complexity KB with theoretical and model-based assertions. We proposed a formalization for reasoning and querying on a KB, integrating the conceptual part of the KB with its model or data-driven aspects. Formalism helps to clarify the risk of producing an incomplete, rather inconsistent KB, which is a first contribution. We also argue that this two-tiered KB is more efficient than one-tier, such as usual KGs and DLs. Further research includes (i) instantiating this approach to other data models and (ii) considering other theories and KB that could enable other interesting inferences.

References

Alkmim, B., Haeusler, E. H., and Nalon, C. (2022). A labelled natural deduction system for an intuitionistic description logic with nominals. In Arieli, O., Homola, M., Jung, J. C., and Mugnier, M., editors, Proceedings of the 35th International Workshop on Description Logics (DL 2022) co-located with Federated Logic Conference (FLoC 2022), Haifa, Israel, August 7th to 10th, 2022, volume 3263 of CEUR Workshop Proceedings. CEUR-WS.org.

Baader, F., Calvanese, D., Mcguinness, D., Nardi, D., and Patel-Schneider, P. (2007). The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press; 2nd Edition.

Bezerra, C., Freitas, F., and Santana, F. (2013). Evaluating ontologies with competency questions. In 2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT), volume 3, pages 284–285.

Craig, W. (1957). Three uses of the herbrand-gentzen theorem in relating model theory and proof theory. Journal of Symbolic Logic, 22(3):269285.

Fitting, M. C. (1969). Intuitionistic logic model theory and forcing. North-Holland Publishing, Amsterdam, Netherlands.

Haeusler, E. H., d. Paiva, V., and Rademaker, A. (2010). Intuitionistic logic and legal ontologies. Conference: Legal Knowledge and Information Systems - JURIX 2010.

Keith D. Cooper, L. T. (2011). Engineering a Compiler . Elsevier, 2nd edition.