Telco Network Inventory Validation with NoHR

Vedran Kasalica[1], Ioannis Gerochristos[2], José Júlio Alferes[2], Ana Sofia Gomes[2], Matthias Knorr[2], and João Leite[2]

[1] Department of Information and Computing Sciences, Utrecht University, 3584 CC Utrecht, The Netherlands
[2] NOVA LINCS, Departamento de Informática, FCT-NOVA Lisboa, 2829-516 Caparica, Portugal

Abstract. Network database inventory is a critical tool for the operations of any telecommunication company, by supporting network configuration and maintenance, as well as troubleshooting of network incidents. Whereas an incorrect inventory can often lead to severe implications and financial losses, the sheer size of a telecommunication network, the number of equipment involved, and other operational constraints, often lead to outdated inconsistent inventories, which are usually validated and updated *by hand*, during change management processes – a time-consuming task highly prone to human error. In this paper, we describe a solution to automate the validation of network inventories within the context of a multinational telecommunication company, with operations in several different countries, using NoHR, a reasoner that allows the user to query (hybrid) knowledge bases composed of ontologies and non-monotonic rules, both of which are necessary to perform the kind of reasoning required by this task. In addition, to address severe performance issues – essentially in terms of memory – resulting from NoHR v3.0's need to pre-process the entire database into OWL assertions or rule facts, in this paper, we also present v4.0 of NoHR, which extends NoHR v3.0 with native support for Databases, solving not only the memory consumption problems, but also improving the average reasoning times.

1 Introduction

Network database inventory is a critical tool for any telecommunication company, maintaining information on what network nodes exist and their characteristics (model, band, frequency, etc.); how nodes connect with each other, their physical and logical paths; and general topology configuration. Network inventory supports many different parts of a telecommunication organization. In particular, it provides important inputs for planning and provisioning, but it is absolutely critical to network operations, by supporting network configuration and maintenance, as well as troubleshooting of network incidents. However, given the sheer size of a telecommunication network, and the number of equipment involved, network inventories are often outdated, providing a misleading image of the network configuration. On a network operation service, an incorrect inventory can lead, for instance, to a wrong assessment of the problem root-cause,

evaluating an incident with the incorrect priority, or setting up an insufficient work-around to restore operations. In any case, an incorrect inventory can often lead to financial losses and implications.

Normally, on any telecommunication company, network inventories are stored in the back-end of some specialized enterprise system, supported by some relational database. While some tools and processes may exist to automate inventory validation and correctness, these are normally very limited and incomplete.

Most enterprise inventory solutions incorporate some constraints over the data being inserted and updated, e.g., that a Radio Base Station (RBS) cannot be inserted without a related Location, or that a Cell can only be created for a given parent RBS. However, these validations are very limited and rely on simple database constraints defined by the vendor. Additionally, since engineers interact with the inventory at a user interface level without the context of a transaction, they often may need to leave the database in an inconsistent state during a planned work. As such, these solutions cannot implement very strong constraints over what is being inserted, and one may, e.g., find a 3G RBS station (aka NodeB) with a defined connection to a 2G Radio Controller (aka BSC), although that simply cannot be the case.

Other more sophisticated solutions exist, which connect to the live network and then compare it with the network inventory. However, these can overload the live network with requests and impact network service, which requires that they be planned with care, managed as low priority requests, and validation can take several hours/days depending on the network size. Additionally, these solutions rely on having connectors to each of the different telco hardware vendors and technologies, which make the solution considerably expensive. In fact, all vendors have closed and proprietary protocols to connect to each of their network nodes, which may change (and be charged differently) depending on the technology or even the equipment's firmware version. Moreover, most telecommunication companies have at least three generations of equipment and have contracts with several different vendors. As a consequence, normally only some part of the network is scanned with these tools, due to prohibitive integration costs. Finally, these solutions do not perform any validation per se. They just update the inventory with the current image of that day, and cannot validate if network nodes are implemented according to the organization's best-practices and rules.

Due to all these reasons, often these network inventories are simply validated and updated *by hand*, during change management processes, which is a time-consuming task highly prone to human error.

In this paper, we describe an implemented solution to automate the validation of network inventories within the context of a multinational telecommunication company, with operations in several different countries, based on NoHR (Nova Hybrid Reasoner) [11, 6, 17]. NoHR v3.0 [17] is a reasoner theoretically founded on the formalism of Hybrid MKNF under the well-founded semantics [15], with support for paraconsistent reasoning [13], which allows the user to query (hybrid) knowledge bases composed of both ontologies and non-monotonic rules. Using a top-down reasoning approach, which means that only the part of the

ontology and rules that is relevant for the query is actually evaluated, NoHR is implemented in a way that combines the capabilities of the DL reasoners ELK [14], HermiT [9], and Konclude [21] with the rule engine XSB Prolog,[3] to deliver very fast interactive response times.

Turning back to the network inventory validation problem, on closer inspection of the existing inventory data, we observe that even though operations in different countries rely on data managed with very different levels of maturity – from those with state-of-the-art enterprise inventory systems to those where most up-to-date data is stored in very basic excel spreadsheets – ultimately, all is accessible through a standard database connector such as ODBC. Also, although the enterprise systems implemented in each of the countries are very different, network inventories are, in their essence, all the same for any telecommunication company. Namely, a Mobile Network is normally split into three domains and networks: radio access network (RAN) which comprises the necessary equipment for the interaction with a Client User Equipment; the CORE network that comprises all the equipment and technology to provide Voice, Data and any other service; and the Transmission (Tx) Network that deals with transport between the nodes and between the RAN and CORE. As a result, such generic knowledge and concepts of a Mobile Network that are required to make sense and validate the network inventories are naturally modelled through an ontology. Additionally, over the years, the telecommunication company has maintained a body of knowledge gathered from experts, encoding their reasoning when validating the network inventory. This knowledge mostly encodes possible (and somewhat typical) problems usually found in inventories. For example, the experts know that, except from those using Long Term Evolution (LTE) technology, every active station must be connected to exactly one active controller, which means that any non-LTE active station, in the inventory, with no connection to one active controller, or with more than one active controller connected to it, must represent an error in the inventory. What this body of knowledge reveals is that the kind of reasoning carried out by these experts – for example, referring to defaults and exceptions – is the same kind of reasoning that requires and can be expressed as non-monotonic rules.

Whereas, in principle, NoHR v3.0 would be sufficient to deal with all the data and knowledge used in the validation of network inventories problem – besides directly using the ontology and encoding the experts' reasoning rules, we would extract the data from the databases, convert it to a format that is acceptable by the reasoner and then load it into the memory – this would be complicated, time-consuming, and, depending on the data size, sometimes not even feasible. Additionally, it would likely be far from optimal in the sense that the working memory would likely be loaded with huge amounts of facts from the database not required for the reasoning process. Finally, the static interaction with database management systems would mean that any change in the database would yield an inconsistency with the working memory, which would require the data extraction process to restart. It turns out that this is not a limitation

[3] http://xsb.sourceforge.net

specific to the validation of network inventories problem: most of the data used in modern industry is stored in some type of database management system, and its use within NoHR would always result in a similar problem.

Addressing this issue, in this paper, we also describe v4.0 of NoHR, which extends NoHR v3.0 with native support for Databases. From a theoretical standpoint this support is encoded through the concept of *mappings* between predicates in the hybrid knowledge base (both in the rules and the ontology) and SQL query results from the corresponding database systems. In particular, the mappings allow to transparently and simultaneously integrate data stored in different databases, which is of crucial importance to deal with the validation of network inventories. From a practical perspective, the mappings are implemented using ODBC drivers, thus allowing the integration of NoHR with all major database management systems, together with a user interface.

The evaluation has shown that this implementation of NoHR v4.0 not only solves the problems of memory consumption and preprocessing in case of loading large amounts of data, but also managed to considerably improve the average reasoning time over this data when making use of query optimizations based on state-of-the-art database technology.

2 Background on NoHR

We start by providing some background information on the hybrid knowledge bases considered within NoHR.

2.1 Description Logics

Description logics (DLs)[4] are commonly decidable fragments of first-order logic, defined over disjoint countably infinite sets of *concept names* $\mathsf{N_C}$, *role names* $\mathsf{N_R}$, and *individual names* $\mathsf{N_I}$, matching unary and binary predicates, and constants, respectively. *Complex concepts* (and *complex roles*) can be defined based on these sets and the logical constructors a concrete DL admits to be used. An *ontology* $\mathcal{O}$ is a finite set of *inclusion axioms* of the form $C \sqsubseteq D$ where C and D are both (complex) concepts (or roles) and *assertions* of the form $C(a)$ or $R(a, b)$ for concepts C, roles R, and individuals a, b. The semantics of such ontologies is defined in a standard way for first-order logic.

The DL $\mathcal{SROIQ}$ [10] underlying the W3C standard OWL 2 is very general and highly expressive, but reasoning with it is highly complex, which is why the profiles OWL 2 EL, OWL 2 QL and OWL 2 RL have been defined [18], for which reasoning is tractable. NoHR supports all three profiles, in fact, even a combination of the constructors provided by them [17].[5]

For the use case, we require $\mathcal{EL}^+_\bot$, a large fragment of the DL underlying OWL 2 EL, which only allows conjunction of concepts, existential restriction of concepts, hierarchies of roles, and disjoint concepts, and, in addition, symmetric roles. Hence, we use a combination of constructors from different OWL 2 profiles.

[4] We refer to [2] for a more general and thorough introduction to DLs.
[5] We refer to [11, 6, 17] and pointers therein for all constructors supported by NoHR.

2.2 Hybrid Knowledge Bases

The hybrid knowledge bases we consider here are MKNF knowledge bases (KBs), which build on the logic of minimal knowledge and negation as failure (MKNF) [16]. Among the two different semantics defined for these [19, 15], we focus on the well-founded one [15], due to its lower computational complexity and amenability to top-down querying without computing the entire model. Again, we only point out important notions, and refer to [15, 1] for the details.

A *rule* r is of the form $H \leftarrow A_1, \ldots, A_n, \mathbf{not}\ B_1, \ldots, \mathbf{not}\ B_m$ where the *head* of r, H, and all A_i with $1 \leq i \leq n$ and B_j with $1 \leq j \leq m$ in the *body* of r are atoms, possibly built from the unary and binary predicates occurring in the ontology.[6] A *program* $\mathcal{P}$ is a finite set of rules, $\mathcal{O}$ is an ontology, and an *MKNF knowledge base* $\mathcal{K}$ is a pair $(\mathcal{O}, \mathcal{P})$. A rule r is *safe* if all its variables occur in at least one A_i with $1 \leq i \leq n$, and $\mathcal{K}$ is *safe* if all its rules are safe.[7]

The semantics of MKNF knowledge bases $\mathcal{K}$ is given by a translation π into an MKNF formula $\pi(\mathcal{K})$, i.e., a formula over first-order logic extended with two modal operators **K** and **not**. The well-founded MKNF model can be computed efficiently [15] in a bottom-up fashion, and queried based on $\mathbf{SLG}(\mathcal{O})$, as defined in [1]. This procedure extends SLG resolution with tabling [4] with an *oracle* to $\mathcal{O}$ that handles ground queries to the DL-part of $\mathcal{K}$ by returning (possibly empty) sets of atoms that, together with $\mathcal{O}$ and information already proven true, allows us to derive the queried atom. We refer to [1] for the full account of $\mathbf{SLG}(\mathcal{O})$.

3 Validating Telco Network Inventory Data

In this section, we illustrate how the knowledge relevant for the validation of telco network inventory data can be expressed in an MKNF knowledge base so that the reasoner NoHR can be applied to solve this problem. Following Sec. 1, we thus have to represent the ontology on network inventories, the expert knowledge on validating network inventories, and the data itself within an MKNF KB.

As outlined in the introduction, the ontology describes generic knowledge of the network inventory, common to the inventories in all countries of the multinational telco company. E.g., the knowledge that RAN, CORE, and Tx are all (disjoint) Mobile Nodes can be expressed as follows.

$$\begin{array}{lll} Core \sqsubseteq MobileNode & RAN \sqsubseteq MobileNode & Tx \sqsubseteq MobileNode \\ Core \sqsubseteq \neg RAN & Core \sqsubseteq \neg Tx & RAN \sqsubseteq \neg Tx \end{array}$$

A further part of this hierarchical knowledge as shown in Fig. 1 can be represented similarly. Moreover, there are several axioms in the ontology encoding specific knowledge about the telecom network topology. For example:

[6] Conceptually, this allows to simultaneously view certain predicates under the closed world semantics in rules and under the open world semantics in the ontology, and admits the bidirectional flow of information between both the rules and the ontology.

[7] In general, the notion of DL-safety is used in this context which requires that these variables occur in atoms that do themselves not occur in the ontology, but due to the reasoning method employed in NoHR, we can relax that restriction.

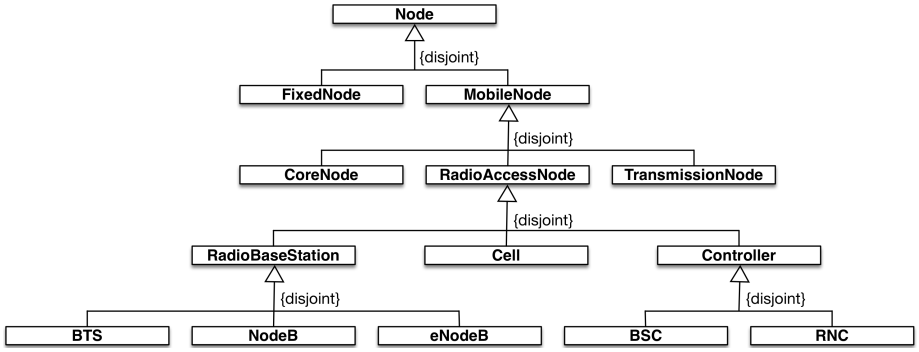


Fig. 1. Part of the inventories ontology

– Every BTS station is always connected to a BSC controller:

$$BTS \sqsubseteq \exists isConnected.BSC \tag{1}$$

– If a Radio Base Station is directly connected to a core node, then it is a 4G station (aka eNodeB):

$$\exists isConnected.CoreNode \sqcap RBS \sqsubseteq eNodeB \tag{2}$$

Note that these axioms do not represent integrity constraints. Rather, they allow us to infer knowledge that is not explicitly present in the knowledge base, i.e., to compensate for some incompleteness in the inventory. This is especially important because not all inventories in the different countries have a detailed specification of all kinds of nodes, and sometimes leave the specification at higher levels in the taxonomy.

The expert knowledge about possible problems in the inventories can be encoded using non-monotonic rules. For example, the experts' knowledge that, except from those using Long Term Evolution (LTE) technology, every active station must be connected to exactly one active controller, means that any non-LTE active station in the inventory with no connection to one active controller, or more than one connection to an active controller, must represent an error in the inventory. Using rules, this can be encoded as follows.

$$\begin{aligned}
badnLTEConn(X) \leftarrow & RBS(X), \textbf{not } lteNode(X), active(X), \\
& \textbf{not } controllerConnected(X) \\
lteNode(X) \leftarrow & eNodeB(X) \\
controllerConnected(X) \leftarrow & isConnected(X,Y), active(Y), Controller(Y), \\
& \textbf{not } duplicateController(X,Y) \\
duplicateController(X,Y) \leftarrow & isConnected(X,Z), active(Z), Z \neq Y, \\
& Controller(Z), Controller(Y)
\end{aligned}$$

Note that the information on $active(X)$ can be found in the database, and that isConnected is a symmetric role in the ontology based on content in the database.

We emphasize that encoding this problem requires the usage of default negation, e.g., because we want to determine nodes that are not known to be connected, i.e., connections not present in the database, as well as ontological inference, since the inventory usually does not explicitly store information on controllers, but rather on more specific types of equipment that, according to the ontology, can be inferred to be controllers, and since LTE nodes can be inferred from the ontology given its connections (cf. axiom (2)). Further cases of such expert knowledge can also be encoded with rules and will be discussed in Sec. 5.

Finally, regarding the data,[8] it can be included in the reasoning process by transforming it into rule facts or ontology assertions. As mentioned, in principle, this readily allows the usage of NoHR v3.0 for validation of network inventory data. However, this transformation of the data would be complicated, time consuming, and, depending on the data size, sometimes not even feasible. Moreover, we would possibly load huge amounts of facts from the database not required for the reasoning process,[9] and any change in the database would require the data extraction process to restart. This is why we next present the new version of NoHR that overcomes these problems by providing native support for databases.

4 NoHR: Database Integration

In this section, we describe the new version of NoHR, NoHR v4.0, and discuss several features of its implementation, with a particular focus on the novel native support for databases including new functionalities and the associated benefits.

4.1 A Third Component for Hybrid KBs

In order to support the integration with external datasets, we have to extend MKNF KBs. Such an extension could be realized by the addition of a database component, effectively turning MKNF KBs into a triple comprising an ontology, a program (of non-monotonic rules), and a database to which we wish to connect. However, in the context of validating data of network inventory, in particular on a multi-national scale, it is clearly preferable to admit several databases to be integrated. Arguably, one could join several databases into one for the sake of the formalism, but this is not necessarily easy as it would require, e.g., to handle (partially) repeated columns with potentially contradictory data, and, from a practical point of view, we would like to simply consult data in tables of different databases, and MKNF KBs should be conceptually close to this idea.

To tackle the integration of several databases within an MKNF KB, we introduce the concepts of mappings and of a mapping knowledge base.[10]

[8] As the actual database schemas are confidential, we cannot disclose them here.

[9] E.g., there are around 200K of facts for one of the countries involved.

[10] Similar concepts have been used before for adding database support to rule systems, such as DLV^{DB} [22], and in ontology based data access, such as in ontop [3].

Definition 1. *Let p be a predicate, db a database and q a query defined over db, where p and the tuples returned by q have the same arity. A triple $\langle p, db, q\rangle$ is called a* mapping for *p, where the result set from db for the query q is mapped to the predicate p. A set of mappings $\mathcal{M}$ is called a* mapping knowledge base.

Essentially, mappings are used to create predicates that are populated with the result set obtained from queries to external databases. Based on this we can extend MKNF KBs as follows.

Definition 2. *An MKNF knowledge base $\mathcal{K}$ is a triple $\mathcal{K} = (\mathcal{O}, \mathcal{P}, \mathcal{M})$, where $\mathcal{O}$ is an ontology, $\mathcal{P}$ is a finite set of rules, and $\mathcal{M}$ is a mapping knowledge base.*

For the semantics of such MKNF knowledge bases, we can extend the translation function to mappings and the mapping knowledge base by turning, for each triple $\langle p, db, q\rangle$ in $\mathcal{M}$, all tuples for p into rule facts. Based on this, it can be shown that all technical results for the well-founded MKNF semantics [15], as well as for top-down querying in **SLG($\mathcal{O}$)** [1] hold.

For reasons of space, we omit the details here, and proceed by showing how this extension is reflected in the architecture of NoHR.

4.2 Architecture of NoHR

NoHR is available as a plugin for Protégé[11], a well-known and widely used ontology editor, and we describe the system architecture of this plugin NoHR v4.0 as shown in Fig. 2. When compared with NoHR v3.0, the architecture has been extended with external databases, corresponding ODBC drivers and the integration of the mapping knowledge base (all labelled with a green background).

The input for the plugin consists of an OWL file, a rule file and a mappings file. All three components can be edited in Protégé, using the built-in interface for the ontology and the custom "NoHR Rules" and "NoHR Mappings" tabs, provided by the plugin, for the rule and mapping components. After the inputs (which can be empty) and the first query are provided, the ontology is translated into a set of rules, using one of the provided reasoners, ELK [14], HermiT [9] or Konclude [21], depending on the DL in which the ontology is written. This resulting set of rules is not equivalent to the ontology in general, but it yields exactly the same answers for ground queries (for more details cf. [11, 6, 17]). The resulting set is then combined with the rules and mappings provided by the input. This joined result serves as input to XSB Prolog via InterProlog[12], which is an open-source Java front-end, allowing the communication between Java and a Prolog engine, and the query is sent via the same interface to XSB to be executed. During the execution, mappings are providing facts from the external databases as they are requested in the reasoning process. This procedure is supported by the installed ODBC connections and handled within XSB, thus providing full control over the database access during querying and taking advantage of the

[11] https://protege.stanford.edu/
[12] http://interprolog.com/java-bridge/

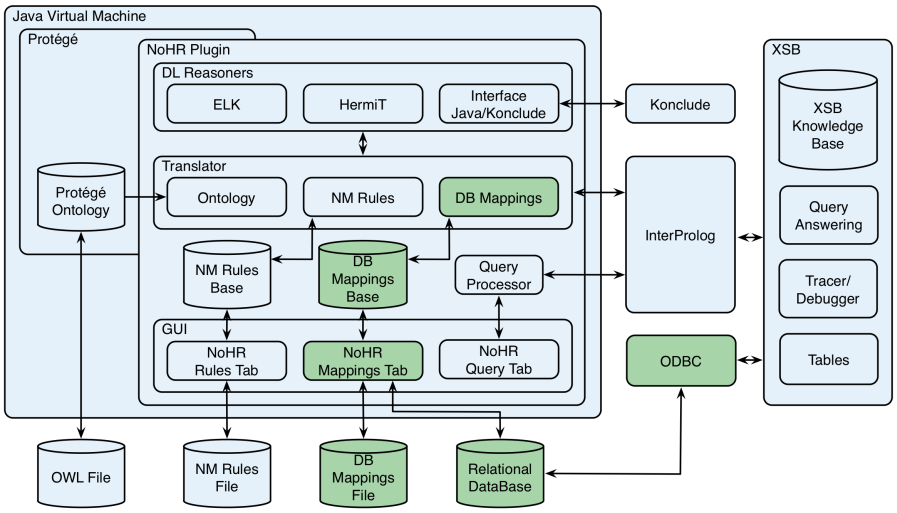


Fig. 2. System architecture of NoHR v4.0 with native database support

built-in optimization to access only the relevant part of the database. Answers are returned to the query processor, which displays them to the user in a table (in the Query Tab). The user may pose further queries, and the system will simply send them directly to XSB, without any repeated preprocessing. If the knowledge base is edited, the system recompiles only the part that was changed.

4.3 Implementing Database Support

We now discuss several of the design decisions on implementing database support within NoHR 4.0 and the benefits we were able to leverage.

First, the connection to various databases is realized via the *XSB - ODBC Interface*, because it is part of the query engine XSB used in NoHR, it supports all major database management systems (DBMSs), and it provides an easy-to-use and well-known connection driver independently of the operating system.

Among the three levels of interaction with the database the *XSB - ODBC Interface* provides, *SQL level, relation level* and *view level*, we chose the SQL level, because, unlike the other two, it allows the usage of standard SQL syntax and provides the flexibility to map an arbitrary SQL query to the predicate, which also provides considerable performance gains compared to the other two.

To allow the user to create the necessary mappings, i.e., combinations of a predicate, an SQL query and a database connection, the "Mappings Tab" has been introduced to the Protégé plugin version of NoHR. It contains a parametrizable mapping form, which offers two different approaches to create mappings, namely mapping with the SQL Designer and Manual SQL mappings.

The *SQL Designer* allows the creation of mappings based on the user's specification of what columns from which tables of which database should be com-

bined, where the underlying SQL queries are dynamically generated, based on the structure of the schema. As this interface has full control over the structure of the SQL query, several optimizations are applied, including improved handling of floating point number unification and of bounded variables. For example, the WHERE clause of the SQL query is dynamically adjusted to fetch only the relevant tuples, depending on the bounded variables in the predicate.

In order to generalize the DBMS integration, we also provide support for *Manual SQL mappings*, i.e., arbitrary SQL queries, to take advantage of the capabilities of the specific DBMS at hand. This allows, e.g., the usage of nested queries and benefiting from the associated performance gains when querying.

5 Evaluation

Previous tests of NoHR have shown that different ontologies can be pre-processed into rules in reasonably short amounts of time (around one minute for Snomed CT with over 300,000 concepts), loading of rules is only linearly dependent on the size of the rule file, and querying can often be done with interactive response times (cf. [11, 12, 6, 5, 17]). Here, we evaluate three measures in the use case of validating the network inventory that show that the native support of databases comes with considerable performance benefits for the reasoning process of NoHR.

We compare NoHR 4.0 with its predecessor in terms of preprocessing time and memory usage. To replicate the tests in the telco company, we generated sets of facts corresponding to database instances of increasing size, closely resembling the data used in the network inventory validation use case, and tested the impact of loading these files in NoHR. All tests were performed on an i5-2.4GHz processor with 8 GB under win64. The results are shown in Fig. 3.

We first note that, since NoHR v4.0 does not require preloading of facts corresponding to the data stored in a database, the observed values for NoHR v4.0 can serve as a (constant) baseline in both cases. In terms of memory for NoHR v3.0, we observe a steady increase of used memory until the limit of the free available RAM is reached (around 20K of facts with many arguments). From there on, memory usage does not increase any further. Rather virtual memory is used increasing the size of the virtual address space beyond the available amount of RAM using paging and/or swapping to secondary storage. We can observe that this has a considerable impact on the loading time, e.g., loading 100K of facts takes around 38 minutes since swapping/secondary storage is in general considerably slower. In fact, we tried loading 200K (roughly the amount of data corresponding to one of the countries in which validation of network inventory is applied) and it failed to upload within the set time-out of one hour. We note that the given times only consider the loading, and do not even include the time necessary to transform the database content into rule facts in the right format. Again, for NoHR v4.0, the problem ceases to exist, as there is no need to load large amounts of facts corresponding to the database content into memory, which makes the NoHR v4.0 usable also for applications with larger amounts of data.

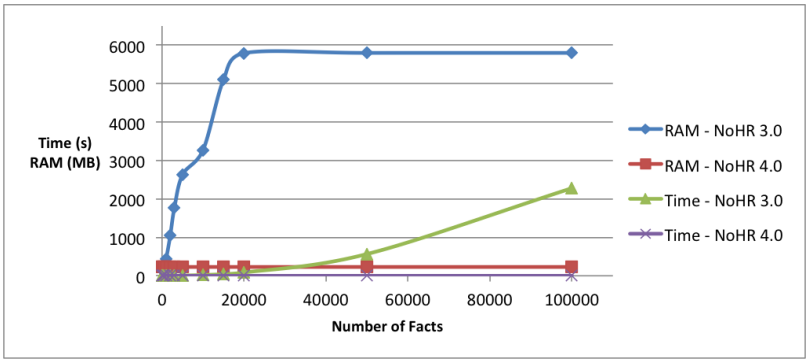


Fig. 3. Memory usage and time of data preloading for NoHR v3.0 and v4.0.

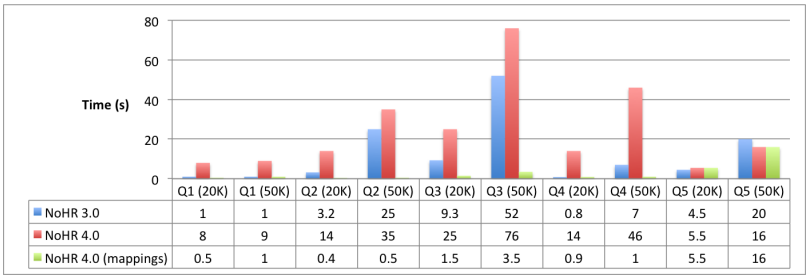
In order to evaluate the effect of using external databases on querying, we compare the time for answering several queries in the network inventory validation use case for NoHR v3.0 and NoHR v4.0. In fact, we consider two cases for NoHR v4.0: one with (simple) direct mappings from predicates to corresponding database columns, and one with advanced mappings using sophisticated queries to make use of optimizations in efficient state-of-the-art DBMS where possible. We use two sets of generated data instances of size 20K and 50K, resembling the actual data used, and five queries inspired by the real use case, namely: (1) find all active nodes that are located at a location that is marked as out of order; (2) find all nodes (equipments) manufactured by Ericsson before 1995 that are connected to Huawei equipment manufactured after 2010 (because they are incompatible); (3) find non-LTE active stations that are not connected to exactly one active controller (cf. Sec. 3); (4) find two locations that share the same coordinates and are both active; and (5) Find active nodes that are not connected to any other node. Among the two general purpose DL reasoners available in NoHR (given that the ontology does not fit one profile), we used HermiT, as it has been shown to be superior for all but the really large ontologies [17].

The results are shown in Fig. 4. As expected, NoHR v4.0 is slightly slower, on average, when querying, as the connection via ODBC adds an overhead to the query process. However, if we use advanced mappings, which allow to outsource certain joins over data from XSB to the DBMS, then NoHR v4.0 outperforms NoHR v3.0 by a considerable margin, in particular when advanced database joins reduce the amount of data that needs to be sent to XSB for reasoning.

Overall, we observe that NoHR v4.0 is competitive with NoHR v3.0 in terms of querying, and superior when part of the query can be processed by the DBMS directly, while eliminating the memory usage and preprocessing time problems.

6 Conclusions

We have presented an implemented solution to automate the validation of network inventories within the context of a multinational telecommunication com-



	Q1 (20K)	Q1 (50K)	Q2 (20K)	Q2 (50K)	Q3 (20K)	Q3 (50K)	Q4 (20K)	Q4 (50K)	Q5 (20K)	Q5 (50K)
NoHR 3.0	1	1	3.2	25	9.3	52	0.8	7	4.5	20
NoHR 4.0	8	9	14	35	25	76	14	46	5.5	16
NoHR 4.0 (mappings)	0.5	1	0.4	0.5	1.5	3.5	0.9	1	5.5	16

Fig. 4. Time of query answering in NoHR v3.0 and NoHR v4.0

pany, with operations in several different countries, based on the reasoner NoHR. Since using NoHR v3.0 for this solution would require to load all the data into memory, which is problematic given the amount of data in the context of the multinational telco company, we also introduced NoHR v4.0, which extends NoHR v3.0 with native database support. We have described this database support in NoHR extending the underlying formalism by mappings and its integration in NoHR's architecture, and we have discussed important features such as the flexible XSB-ODBC interface, general support for SQL queries, and interfaces for the creation of optimized SQL queries. The evaluation confirms that using the new version is highly beneficial, in particular for use cases with large amounts of data, such as the validation of network inventories, because it avoids the overhead of transforming the database into facts in NoHR, reducing time and memory usage considerably, as well as during querying where we can make use of query optimizations that are based on state-of-the-art database technology.

In terms of future work, currently MySQL and Oracle DBMSs are fully supported, which sufficed for the validation use case. The XSB-OBDC interface is, however, flexible, and making the necessary adjustments so that all major DBMSs are supported is important, to admit the usage of NoHR in other use cases (with different DBMSs). Adapting ideas on dynamic hybrid KB's [20] and semi-automatic mapping creation [22] is also promising to further improve usability, and a comparison with the integration of databases with ontologies and rules in the HEX formalism [7], based on dl-programs [8], is of interest, even if arguably less general than hybrid MKNF [19]. A more ambitious objective is the integration of data on the Semantic Web, i.e., Linked Open Data. While, conceptually, the idea corresponds to database integration, the technical solution will certainly differ, due to different standards and formats employed. Given the wide-spread availability of Linked Open Data sets nowadays, such addition would provide a valuable extension to NoHR for knowledge integration.

Acknowledgments. We would like to acknowledge the helpful comments by the anonymous reviewers, the valuable contribution of N. Costa, V. Ivanov, and C. Lopes to the development of NoHR, and partial support by FCT projects RIVER (PTDC/CCI-COM/30952/2017) and NOVA LINCS (UID/CEC/04516/2013).

References

1. Alferes, J.J., Knorr, M., Swift, T.: Query-driven procedures for hybrid MKNF knowledge bases. ACM Trans. Comput. Log. 14(2), 1–43 (2013)
2. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, 3rd edn. (2010)
3. Calvanese, D., Cogrel, B., Komla-Ebri, S., Kontchakov, R., Lanti, D., Rezk, M., Rodriguez-Muro, M., Xiao, G.: Ontop: Answering SPARQL queries over relational databases. Semantic Web 8(3), 471–487 (2017)
4. Chen, W., Warren, D.S.: Tabled Evaluation with Delaying for General Logic Programs. J. ACM 43(1), 20–74 (1996)
5. Costa, N., Knorr, M., Leite, J.: Querying LUBM with non-monotonic features in protege using nohr. In: Procs. of ISWC Demos. CEUR Procs., vol. 1486 (2015)
6. Costa, N., Knorr, M., Leite, J.: Next step for NoHR: OWL 2 QL. In: Procs. of ISWC. LNCS, vol. 9366 (2015)
7. Eiter, T., Fink, M., Ianni, G., Krennwallner, T., Redl, C., Schüller, P.: A model building framework for answer set programming with external computations. TPLP 16(4), 418–464 (2016)
8. Eiter, T., Ianni, G., Lukasiewicz, T., Schindlauer, R., Tompits, H.: Combining answer set programming with description logics for the semantic web. Artif. Intell. 172(12-13), 1495–1539 (2008)
9. Glimm, B., Horrocks, I., Motik, B., Stoilos, G., Wang, Z.: Hermit: An OWL 2 reasoner. J. Autom. Reasoning 53(3), 245–269 (2014)
10. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible $\mathcal{SROIQ}$. In: Procs. of KR. AAAI Press (2006)
11. Ivanov, V., Knorr, M., Leite, J.: A query tool for $\mathcal{EL}$ with non-monotonic rules. In: Procs. of ISWC. LNCS, vol. 8218 (2013)
12. Ivanov, V., Knorr, M., Leite, J.: Reasoning over ontologies and non-monotonic rules. In: Procs. of EPIA. LNCS, vol. 9273. Springer (2015)
13. Kaminski, T., Knorr, M., Leite, J.: Efficient paraconsistent reasoning with ontologies and rules. In: Procs. of IJCAI. AAAI Press (2015)
14. Kazakov, Y., Krötzsch, M., Simančík, F.: The incredible ELK: From polynomial procedures to efficient reasoning with $\mathcal{EL}$ ontologies. Journal of Automated Reasoning 53, 1–61 (2013)
15. Knorr, M., Alferes, J.J., Hitzler, P.: Local closed world reasoning with description logics under the well-founded semantics. Artif. Intell. 175(9–10), 1528–1554 (2011)
16. Lifschitz, V.: Nonmonotonic databases and epistemic queries. In: Mylopoulos, J., Reiter, R. (eds.) Procs. of IJCAI. Morgan Kaufmann (1991)
17. Lopes, C., Knorr, M., Leite, J.: Nohr: Integrating XSB prolog with the OWL 2 profiles and beyond. In: Procs. of LPNMR. LNCS, vol. 10377. Springer (2017)
18. Motik, B., Cuenca Grau, B., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C. (eds.): OWL 2 Web Ontology Language: Profiles (Second Edition). W3C (2012)
19. Motik, B., Rosati, R.: Reconciling description logics and rules. J. ACM 57(5), 93–154 (2010)
20. Slota, M., Leite, J., Swift, T.: On updates of hybrid knowledge bases composed of ontologies and rules. Artif. Intell. 229, 33–104 (2015)
21. Steigmiller, A., Liebig, T., Glimm, B.: Konclude: System description. J. Web Sem. 27, 78–85 (2014)
22. Terracina, G., Leone, N., Lio, V., Panetta, C.: Experimenting with recursive queries in database and logic programming systems. TPLP 8(2), 129–165 (2008)