Existential N3: From N3 to Existential Rules

Dörthe Arndt[1] and Stephan Mennicke[2]

[1] Computational Logic Group, Technische Universität Dresden, Germany
[2] Knowledge-Based Systems Group, Technische Universität Dresden, Germany
`{firstname.lastname}@tu-dresden.de`

Abstract. Notation 3 Logic (N3) extends the Resource Description Framework (RDF) by the opportunity to directly state rules on Semantic Web data. Although N3 is already used in practice (e.g., by the reasoners EYE, Cwm, and Data-Fu), the formal semantics of N3 is currently under development. For so-called *simple formulas*, a subset of N3 which is interpreted equally by all reasoners, the semantics is already defined. In this paper we introduce existential N3 ($N3^\exists$), a subset of *simple N3 formulas* being directly mappable to existential rules, an important paradigm in knowledge representation and reasoning. We define such a mapping and show that it preserves and reflects equivalence of N3 formulas: two semantically equivalent N3 formulas (i.e., they have the same models) are translated to semantically equivalent rule sets and conversely, whenever we translate two formulas into equivalent rule sets, the formulas have already been equivalent. In order to also show the practical relevance of our work, we provide an implementation which translates $N3^\exists$ rules to existential rules. We compare the performance of EYE and Cwm to VLog, a reasoner for existential rules. By doing so, we provide a bridge from N3 to existential rules. We strongly believe that our work enables the communities around both frameworks to benefit from each other's findings.

Keywords: Notation3 · Datalog · Existential rules.

1 Introduction

Reasoning in the Semantic Web is mostly done using Description Logics (DL). By applying OWL DL and its profiles on facts and axioms, reasoners like Pellet [18] or ELK [16] produce new knowledge. Rule-based reasoning is mostly either only applied on top of DL reasoning – using SWRL – or it is limited to SPARQL querying. Despite the fact that there are frameworks and standards for rule-based reasoning in the Semantic Web – like the rule-interchange format (RIF) or Notation3 Logic (N3) [1, 8] – rules are not as established in the Web as they could be. This is even more surprising considering the fact that the logic-programming community offers powerful systems (e.g., Vlog [11] or Vadalog [4]). The latter act on rules and facts but the format they use, existential rules, is not native to the Web. This hinders interested Web practitioners from using these technologies, making it difficult for developers of Semantic Web rule engines to compare their results with these implementations and it stops the existential rule community from focusing on the interesting problems with distributed and heterogeneous data, as commonly present in the Web.

In this paper, we aim at providing a bridge between the different approaches and technologies: we introduce *existential Notation3 Logic* (N3$^\exists$), a subset of N3 supporting existential quantification. We define the semantics of N3$^\exists$ in a way which is compliant with N3 semantics[3] and provide a mapping between N3$^\exists$ and existential rules. We show that this mapping preserves equivalence in the sense that the images of two equivalent N3$^\exists$ formulas are equivalent and that the equivalence of two images implies the equivalence of their pre-images. This enables us to solve N3$^\exists$ reasoning problems with existential rule technologies. In order to also show the practical applicability of our approach, we provide an implementation of our mapping in Python which allows us to compare the reasoning performance of the existential rule reasoner Vlog and the N3 reasoners EYE and Cwm for different benchmarks. Our tests show that the selected reasoners are comparable in their performance, but also that they have different strengths. We see that as a motivation for the communities around N3 and existential rules to learn from one another, and hope that our work provides a first step towards this direction.

The remainder of the paper is structured as follows: In Sect. 2 we motivate our approach by providing examples of N3 and existential rule formulas, and discuss how these are connected. In Sect. 3 we provide a more formal definition of N3$^\exists$, introduce its semantics and prove some of its properties. In Sect. 4 we formally introduce existential rules, provide the mapping from N3$^\exists$ into this logic, and prove its truth-preserving properties. Sect. 5 discusses our implementation and provides an evaluation of the different reasoners. In Sect. 6 we discuss the related work to then conclusde our paper with Sect. 7. Due to space restrictions, we had to shorten some of the proofs by leaving out details. There is, however, an extended version of the paper that, together with the scripts we used in our experiments, are publicly available[4].

2 Motivation

Notation3 Logic has been inroduced as a rule-based extension of the Resource Description Framework (RDF). As in RDF, N3 knowledge is stated in triples consisting of *subject*, *predicate* and *object*. In ground triples these can either be Internationalized Resource Identifiers (IRI) or literals. The expression

```
:lucy :knows :tom.                    (1)
```

means[5] that *"lucy knows tom"*. Conjunctions of triples are simply expressed by stating them after each other. As in RDF, N3 triples can contain blank nodes, usually starting with `_:`, which stand for (implicitly) existentially quantified variables. The statement

```
:lucy :knows _:x.                     (2)
```

[3] The semantics of N3 is currently under development, but the semantics of its subset *simple formulae* is already defined [2]. N3$^\exists$ is a subset of *simple formulas*.

[4] Our GitHub repository: https://github.com/smennicke/n32rules

[5] Here and for the remainder of this paper, the empty prefix denotes the example namespace <http://www.example.org#>.

means *"there exists someone who is known by lucy"*. N3 furthermore supports implicitly universally quantified variables, indicated by a leading question mark `?`, and implications which are stated with curly braces `{}` around premise and conclusion connected via an arrow `=>`. The formula

```
{:lucy :knows ?x}=>{?x :knows :lucy}.
```
(3)

means that *"for everybody holds that if lucy knows him, then he also knows lucy"*. Premises and conclusions of rules can also be conjunctions. Furthermore, N3 allows the use of blank nodes in these rules. The latter are not quantified on top level, but in the part of the rule they occur in, that is either in its premise or its conclusion.

```
{?x :knows :tom}=>{?x :knows _:y. _:y :name "Tom"}.
```
(4)

means *"for everybody holds that if he knows tom then* there exists *someone he knows whose name is* Tom*"*.

This last example shows, that N3 supports rules yieding statements about the *existence* of certain instances which makes it easy to express them as *existential rules*. To see this connection, we first introduce the notion of an *instance* on an intuitive level, which is just a (finite or infinite) set of triples like `:lucy :knows :tom.` or `:tom,:name "Tom".` We express such triples using the ternary predicate *tr*. Furthermore, we use the same notation for constants as in N3 to make the connection more explicit. For instance, $tr(\texttt{:lucy}, \texttt{:knows}, \texttt{:tom})$ is a fact we would like to see in instances satisfying the rule representation of(1):

$$\to tr(\texttt{:lucy}, \texttt{:knows}, \texttt{:tom}) \tag{5}$$

An instance satisfies the rule (5) if it includes the fact $tr(\texttt{:lucy}, \texttt{:knows}, \texttt{:tom})$ and is, thus, recognized as a *model* of the rule. As such, existential rules are first-order sentences that are implications allowing for existentially quantified variables in the head (formula on the right-hand side of the implication arrow). The N3 formula in (2), encoded as an existential rule, uses this feature:

$$\to \exists x.\ tr(\texttt{:lucy}, \texttt{:knows}, x) \tag{6}$$

Such as rule (5), rule (6) has an empty body (the left-hand side of the implication arrow), which means the head holds unconditionally in both cases. Models of (5) are also models of (6), but not vice versa. If instead of the fact $tr(\texttt{:lucy}, \texttt{:knows}, \texttt{:tom})$ an instance contains the fact $tr(\texttt{:lucy}, \texttt{:knows}, \texttt{:tim})$, rule (6) is equally satisfied. An important concept to facilitate all the different models is the notion of *universal models* that may incorporate a separate set of terms, called *labeled nulls*. Such nulls may be used as arguments of predicates where a concrete constant is (currently) unknown. By this intuition, labeled nulls allow us to implement blank nodes on an instance level. The universal model of the rule (6) may be described by $\{tr(\texttt{:lucy}, \texttt{:knows}, n)\}$ where n is a labeled null. Universality means that these models admit homomorphisms to each and every model of the rule (or rule set), where mapping nulls to other nulls or even constants is allowed.

The implication of (3) has

$$\forall x.\ tr(\texttt{:lucy},\texttt{:knows},x) \rightarrow tr(x,\texttt{:knows},\texttt{:lucy}) \tag{7}$$

as its rule counterpart. Every instance that contains the fact $tr(\texttt{:lucy},\texttt{:knows},c)$ for some constant (or null) c must also contain the fact $tr(c,\texttt{:knows},\texttt{:lucy})$ in order to satisfy rule (7).

Ultimately, the implication (4) with an existential statement in its consequence may be transferred to a rule with an existential quantifier in the head:

$$\forall x.\ tr(x,\texttt{:knows},\texttt{:tom}) \rightarrow \exists y.\ tr(x,\texttt{:knows},y) \wedge tr(y,\texttt{:name},\texttt{"Tom"}). \tag{8}$$

It is clear that the instance $\mathcal{I} = \{tr(\texttt{:lucy},\texttt{:knows},\texttt{:tom}), tr(\texttt{:tom},\texttt{:name},\texttt{"Tom"})\}$ satisfies the rule. However, instance $\mathcal{J} = \{tr(\texttt{:lucy},\texttt{:knows},\texttt{:tom})\}$ does not satisfy rule (8). Again, to accomodate for all models of the rule, a universal model including $\mathcal{J}$ is $\mathcal{U} = \mathcal{J} \cup \{tr(\texttt{:lucy},\texttt{:knows},n), tr(n,\texttt{:name},\texttt{"Tom"})\}$ (where n is a labeled null) takes arbitrary replacements of y into account. There is, in fact, a homomorphism from $\mathcal{U}$ to $\mathcal{I}$, mapping null n to `:tom`.

In the remainder of this paper, we further analyse the relation between N3 and existential rules.

3 Existential N3

In the previous section we introduced essential elements of N3, namely triples and rules N3 also supports more complex constructs like lists, quoted graphs, nested rules, and various built-ins operating on top of all these. In the past, these complex constructs have been a source for ambiguity [3] and the N3 community group is currently working on a semantics which resolves the uncertainties. As this work is not published yet, we base our research on so-called *simple N3 formulae* [2, Chapter 7]. These are N3 formulae which do not allow for nesting, which is known to be the main source for disagreements when it comes to the meaning of a formula. We further restrict *simple formulae* by also excluding lists and quotations. This allows us to define a simple yet powerful subset of N3 which we call *existential N3* (N3$^\exists$). As we show in the subsequent section, N3$^\exists$ can be mapped directly to existential rules in a way that preserves equivalence and non-equivalence.

3.1 Syntax

We start by introducing the syntax of N3$^\exists$. Here, we reuse the alphabet of RDF. Let C be a set of constants, U a set of universal variables and E a set of existential variables. Sets C, U, and E are mutually disjoint and disjoint with $\{\texttt{\{},\texttt{\}},\texttt{=>},\texttt{.}\}$. Then we call $\mathfrak{A} := C \cup U \cup E \cup \{\texttt{\{},\texttt{\}},\texttt{=>},\texttt{.}\}$ an *N3 alphabet*.

As the distinction is not relevant for our purposes, we treat IRI and literals as introduced in Section 2, as a joint set of constants. As indicated before, universal variables start with the symbol ? and existential variables with `_:` in concrete representations. In Fig. 1 we display the syntax of N3$^\exists$.

`f ::=`		formulas:	`t ::=`		terms:
	`t t t.`	atomic formula		`ex`	existential variables
	`{e}=>{e}.`	implication		`c`	constants
	`f f`	conjunction			
`n ::=`		N3 terms:	`e ::=`		expressions:
	`uv`	universal variables		`n n n.`	triple expression
	`t`	terms		`e e`	conjunction expression

Fig. 1. Syntax of N3$^\exists$

N3$^\exists$ fully covers RDF – RDF formulas are conjunctions of atomic triples – but allows literals and blank nodes to occur in subject, predicate and object position. On top of the triples, it supports rules which can contain existential and universal variables. Note, that the syntax allows rules having new universal variables in their conclusion like for example

`{:lucy :knows :tom}=>{?x :is :happy}.` (9)

which results in a rule expressing *"if lucy knows tom, everyone is happy"* This implication is problematic: Applied on Triple (5) it yields `?x :is :happy.` which is a triple containing a universal variable. Such triples are not covered in our restricted version of N3. In order to avoid such situations and to keep the logic closed, we further restict our rules making use of the notion of *components*.

Here and below, we always assume, that $\mathfrak{A}$ is an N3$^\exists$ alphabet, $\mathfrak{F}$ the set of N3$^\exists$ formulas, $\mathfrak{N}$ the set of N3 terms and $\mathfrak{E}$ the set of expressions over $\mathfrak{A}$.

Components of a formula. Let $f \in \mathfrak{F}$ be a formula over $\mathfrak{A}$. The set $comp(f) \subset \mathfrak{N} \cup \mathfrak{E}$ of *direct components* of f is defined as follows:

- If f is an atomic formula of the form $t_1\ t_2\ t_3.$, $comp(f) = \{t_1, t_2, t_3\}$.
- If f is an implication of the form `{`e_1`}=>{`e_2`}.`, then $comp(f) = \{$`{`e_1`}`$,$`{`e_2`}`$\}$.
- If f is a conjunction of the form $f_1 f_2$, then $comp(f) = comp(f_1) \cup comp(f_2)$.

Likewise, we define the set of *components* $comp(e) \subset \mathfrak{N}$ of an expression e as:

- If e is triple expression of the form $n_1\ n_2\ n_3.$, $comp(e) = \{n_1, n_2, n_3\}$.
- If e is a conjunction of the form $e_1 e_2$, then $comp(e) = comp(e_1) \cup comp(e_2)$.

We define the set of *nested components* of f as

$$comp^n(f) := \mathrm{comp}(f) \cup \{t \in \mathfrak{N} | t \in comp(e) \wedge \texttt{\{}e\texttt{\}} \in comp(f)\}$$

With that we can introduce *well-formed implications*: We call an N3$^\exists$ implication `{`$\mathtt{e}_1$`}=>{`$\mathtt{e}_2$`}` *well-formed* if $U \cap (\mathrm{comp}(\mathtt{e}_2) \setminus \mathrm{comp}(\mathtt{e}_1)) = \emptyset$. That means that the consequence of the implication does not contain "new" universal variables. For the remainder of this paper we assume all implications to be well-formed.

3.2 Semantics

In order to define the semantics of N3$^{\exists}$ we first note, that in our fragment of N3 all quantification of variables is only defined implicitely. The blank node in Triple 2 is understood as an existentially quantified variable, the univeral in Formula 3 as universally quantified. Universal quantification spans over the whole formula - the variable `?x` occurring in premise and conclusion of Rule 3 is univearsally quantified for the whole implication - while existential quantification is local - the conjunction in the consequence of Rule 4 is existentailly quantified there. Adding new triples as conjuncts to Formula 4 like for example

```
:lucy :knows _:y. _:y :likes :cake.                (10)
```

leads to the new statement that *"lucy knows someone who likes cake"* but even though we are using the same blank node identifier `_:y` in both formulas, the quantification of the variables in these formulas is totally seperated and the person named "Tom" is not necessarily related to the cake-liker. To handle these different kinds of scoping behavior we introduce two ways to apply a substitution.

Substitution A *substitution* is a mapping from a set of variables $X \subset U \cup E$ to the set of N3-terms $\mathfrak{T}$.

For a substitution $\sigma : X \to \mathfrak{T}$ and a formula, N3 term or expression x we define the *component application* $x\sigma^c$ as follows:

- $x\sigma^c = \sigma(x)$ if $x \in X$,
- $(s\ p\ o)\sigma^c = (s\sigma^c)(p\sigma^c)(o\sigma^c)$ if $x = s\ p\ o$ is an atomic formula or a triple expression,
- $(f_1 f_2)\sigma^c = (f_1\sigma^c)(f_2\sigma^c)$ if $x = f_1 f_2$ is a conjunction,
- $x\sigma^c = x$ else.

For a substitution $\sigma : X \to \mathfrak{T}$ and a formula, N3 term or expression x we define the *total application* $x\sigma^t$ as follows:

- $(\texttt{\{e}_1\texttt{\}=>\{e}_2\texttt{\}})\sigma^t\texttt{=\{e}_1\sigma^c\texttt{\}=>\{e}_2\sigma^c\texttt{\}}$, if $x = \texttt{\{e}_1\texttt{\}=>\{e}_2\texttt{\}}$ is an implication,
- $x\sigma^t = x\sigma^c$ else.

The component application of a substitution only replaces the variables occuring directly in triples, for the formula f = `x :p :o. {x :b :c}=>{x :d :e}`, with `x` a variable, we get $f\sigma^c$ = (σ`(x) :p :o. {x :b :c}=>{x :d :e}`). The total application replaces all occurrences of `x`, we get $f\sigma^t$ = (σ`(x) :p :o. {`σ`(x) :b :c}=>{`σ`(x) :d :e}`). The total application of a substitution is normally used to handle universal quantification,[6] with component applications we treat existential variables. With this idea we can define the meaning of formulas. We start by introducing the interpretation.

[6] Note that in full N3 the universals can also occur deeply nested, which makes this way of applying substitutions even more suitable for implicit universal quantification.

N3 Interpretation An N3 interpretation $\mathfrak{I}$ of an N3 alphabet $\mathfrak{A}$ consists of:

1. A set $\mathfrak{D}$ called the domain of $\mathfrak{I}$.
2. A function $\mathfrak{a} : C \to \mathfrak{D}$ called the object function.
3. A function $\mathfrak{p} : \mathfrak{D} \to 2^{\mathfrak{D}\times\mathfrak{D}}$ called the predicate function.

Note that the interpretation function for predicates starts from the domain of discourse and not as first-order logic directly on the relation-symbol. That is to stay compatible with RDF semantics and to be able to handle the quantification over predicates without leaving first-order logic.

Semantics of N3$^{\exists}$ Let $\mathfrak{I} = (\mathfrak{D}, \mathfrak{a}, \mathfrak{p})$ be an interpretation of an N3$^{\exists}$ alphabet $\mathfrak{A}$, let $\mathfrak{T}$ be the set of terms over $\mathfrak{A}$, U the set of universal and E the set of existential variables. For an N3$^{\exists}$ formula f over $\mathfrak{A}$ we define:

1. If $\text{comp}^n(f) \cap U \neq \emptyset$, then $\mathfrak{I} \models f$ iff $\mathfrak{I} \models f\sigma^t$ for all substitutions $\sigma : U \to C$.
2. If $\text{comp}^n(f) \cap U = \emptyset$ and $W = \text{comp}(f) \cap E \neq \emptyset$, then $\mathfrak{I} \models f$ iff $\mathfrak{I} \models f\mu^c$ for some substitution $\mu : W \to C$.
3. If $\text{comp}^n(f) \cap U = \emptyset$ and $\text{comp}(f) \cap E = \emptyset$:
   (a) If f is an atomic formula $t_1\, t_2\, t_3$, then $\mathfrak{I} \models t_1\, t_2\, t_3.$ iff $(\mathfrak{a}(t_1), \mathfrak{a}(t_3)) \in \mathfrak{p}(\mathfrak{a}(t_2))$.
   (b) If f is a conjunction $f_1 f_2$, then $\mathfrak{I} \models f_1 f_2$ iff $\mathfrak{I} \models f_1$ and $\mathfrak{I} \models f_2$.
   (c) If f is an implication, then $\mathfrak{I} \models$ `{`e_1`}=>{`e_2`}` iff $\mathfrak{I} \models e_2$ if $\mathfrak{I} \models e_1$.

Note that the semantics as defined above uses a substitution into the set of constants instead of a direct assignment to the domain of discourse in order to interpret quantified variables. This is because N3$^{\exists}$ needs to be compatible with its superset N3 where this approach was chosen to support referential opacity of quoted graphs.

Model und equivalence Let Φ and Ψ be sets of N3$^{\exists}$ formulae. We say that an Interpretation $\mathfrak{M}$ is a *model* of Φ, written as $\mathfrak{M} \models \Phi$, if $\mathfrak{M} \models f$ for each formula $f \in \Phi$. We say that Φ and Ψ are *equivalent*, written as $\Phi \equiv \Psi$, if for all models $\mathfrak{M}$: $\mathfrak{M} \models \Phi$ iff $\mathfrak{M} \models \Psi$. If $\Phi = \{\phi\}$ and $\Psi = \{\psi\}$ are singleton sets, we write $\phi \equiv \psi$ omitting the brackets.

In oder to be able to handle subformulas seperately if needed, we would like to transform formulas in equivalent sets of sub-formulas. The implicit quantification present in N3$^{\exists}$ forms an obstacle here: If we for example go back to Formula 10, we know that every model of that formula is also model of its conjuncts but the opposite direction is not true since the triples share blank nodes which need to be assigned equally by an interpretation of the conjunction. The separation of sub-formulas is still possible, if we keep the blank nodes together.

Lemma 1 (Pieces). *Let $f = f_1 f_2$ be an N3$^{\exists}$ conjunction and let $\text{comp}(f_1) \cap \text{comp}(f_2) \cap E = \emptyset$, then for each interpretation $\mathfrak{I}$ the following holds:*

$$\mathfrak{I} \models f \textit{ iff } \mathfrak{I} \models f_1 \textit{ and } \mathfrak{I} \models f_2.$$

Proof. 1. If f does not contain variables the claim follows from the *Semantics of N3$^{\exists}$* Definition (point 3b).

2. If $\mathrm{comp}^n(f) \cap U = \emptyset$, but $\mathrm{comp}(f) \cap E \neq \emptyset$:
   ($\Rightarrow$) If $\mathfrak{I} \models f$ then there exists a substitution $\mu : \mathrm{comp}(f) \cap E \to C$ such that $\mathfrak{I} \models f\mu^c$, that is $\mathfrak{I} \models (f_1\mu^c)\ (f_2\mu^c)$. According to Step 1 that implies $\mathfrak{I} \models f_1\mu^c$ and $\mathfrak{I} \models f_2\mu^c$ and thus $\mathfrak{I} \models f_1$ and $\mathfrak{I} \models f_2$.
   ($\Leftarrow$) If $\mathfrak{I} \models f_1$ and $\mathfrak{I} \models f_2$, then there exist two substitutions $\mu_1 : \mathrm{comp}(f_1) \cap E \to C$ and $\mu_2 : \mathrm{comp}(f_2) \cap E \to C$ such that $\mathfrak{I} \models f_1\mu_1^c$ and $\mathfrak{I} \models f_2\mu_2^c$. As the domains of the two substitutions are disjoint (by assumption), we can define the substitution $\mu : \mathrm{comp}(f) \cap E \to \mathfrak{T}$ as follows:
   $$\mu(v) = \begin{cases} \mu_1(v) & \text{if } v \in \mathrm{comp}(f_1) \\ \mu_2(v) & \text{else} \end{cases}$$
   Then $\mathfrak{I} \models f\mu^c$ and therefore $\mathfrak{I} \models f$.
3. If $\mathrm{comp}^n(f) \cap U \neq \emptyset$:
   ($\Rightarrow$) Let $\mathfrak{I} \models f$. Then $\mathfrak{I} \models f\sigma^t$ for each substitution $\sigma : U \to C$. Then $\mathfrak{I} \models (f_1f_2)\sigma^t$, that is $\mathfrak{I} \models (f_1\sigma^t)(f_2\sigma^t)$. Since $\mathrm{comp}^n((f_1\sigma^t)(f_2\sigma^t)) \cap U = \emptyset$ we get, according to Step 2, that $\mathfrak{I} \models f_1\sigma^t$ and $\mathfrak{I} \models f_2\sigma^t$. This holds for all substitutions $\sigma : U \to C$. We therefore get $\mathfrak{I} \models f_1$ and $\mathfrak{I} \models f_2$.
   ($\Leftarrow$) Let $\mathfrak{I} \models f_1$ and $\mathfrak{I} \models f_2$. Then, for all substitutions σ_1 and σ_2 from universals to constants: $\mathfrak{I} \models f_1\sigma_1^t$ and $\mathfrak{I} \models f_2\sigma_2^t$. Thus $\mathfrak{I} \models f_1\sigma^t$ and $\mathfrak{I} \models f_2\sigma^t$ for all $\sigma : U \to C$. As $f_1\sigma^t$ and $f_2\sigma^t$ do not contain universal variables we can apply Step 2 and get $\mathfrak{I} \models (f_1f_2)\sigma^t$ and thus $\mathfrak{I} \models f\sigma^t$. □

Below, we often only consider rules without existentials in their premise. This is no strong restiction since such existentials can always be replaced by "fresh" universals without changing the meaning of a rule. The formulas `{_:x :likes :cake}=>{:cake :is :good}.` and `{?y :likes :cake}=>{:cake :is :good}.` have the same meaning namely *"if there is someone who likes cake, then cake is good."*. We prove that below:

Lemma 2 (Existential-elemination). *Let $f =$ `{`e_1`}=>{`e_2`}` and $g =$ `{`e_1'`}=>{`e_2`}` be N3$^\exists$ implications such that $e_1' = e_1\sigma^c$ for some injective substitution $\sigma : \mathrm{comp}(e_1) \cap E \to U \setminus \mathrm{comp}^n(f)$ of the existential variables of e_1 by universals. Then: $f \equiv g$*

Proof. ($\Rightarrow$) We assume that $\mathfrak{M} \models f$ but $\mathfrak{M} \not\models g$ for some model $\mathfrak{M}$. Let $\nu : U \to C$ be a substitution such that $\mathfrak{M} \models f\nu^t$ but $\mathfrak{M} \not\models g\nu^t$. If $\mathfrak{M} \models e_2\nu^c$, then we can immediately conclude, that $\mathfrak{M} \models g\nu^t$ since both implications share the same conclusion. Let's therefore assume that $\mathfrak{M} \not\models e_2\nu^c$ and $\mathfrak{M} \not\models e_1\nu^c$. Since $\mathfrak{M} \not\models g\nu^t$, we get $\mathfrak{M} \models e_1'\nu^c$. But if that is the case, we can define a substitution $\mu : \mathrm{comp}(e_1) \cap E \to C$ as: $\mu(x) := \nu(\sigma(x))$. But then $\mathfrak{M} \models e_1\nu^c\mu^c$ and thus $\mathfrak{M} \not\models f\nu^t$. Contradiction.

($\Leftarrow$) We assume that $\mathfrak{M} \models g$ but $\mathfrak{M} \not\models f$ for some model $\mathfrak{M}$. As $\mathfrak{M} \not\models f$, there exists a substitution $\nu : U \to C$ such that $\mathfrak{M} \models e_1\nu^c$ but $\mathfrak{M} \not\models e_2\nu^c$. But if $\mathfrak{M} \models e_1\nu^c$, there exists a substitution $\mu : comp(e_1) \cap E \to C$ such that $\mathfrak{M} \models e_1\nu^c\mu^c$. We define a substitution $\nu' : U \to C$ as follows:
$$\nu'(v) = \begin{cases} \mu(\sigma^{-1}(v)) & \text{if } v \in range(\sigma) \\ \nu(v) & \text{else} \end{cases}$$
With that ν' we get $\mathfrak{M} \not\models g\nu'^t$ and thus $\mathfrak{M} \not\models g$. Contradiction. □

Normalized rules: For a rule f we call the formula f' in which all existentials occurring in its premise are replaced by universals following the strategy of Lemma 2 the *normalized* version of the rule. For all conjunctions of formulas, we call f *normalized*, if all rules occurring in it as conjuncts are normalized. As a consequence of the lemmas above, we get a special noemal form which we are going to use throughout this paper.

Corollary 1 (Piece Normal Form). *For every well-formed N3$^{\exists}$ formula f, there exists a set $F = \{f_1, f_2, \ldots, f_k\}$ of N3$^{\exists}$ formulas such that $F \equiv \{f\}$ and F is in* piece normal form *(PNF). That is, all $f_i \in F$ are normalized formulae, and $k \in \mathbb{N}$ is the maximal number such that for $1 \leq i, j \leq k$, $comp(f_i) \cap comp(f_j) \cap E \neq \emptyset$ implies $i = j$. If f_i $(1 \leq i \leq k)$ is a conjunction of atomic formulas, we call f_i an* atomic piece.

4 From N3 to Existential Rules

Without loss of generality, we translate sets F of N3$^{\exists}$ formulae in PNF (cf. Corollary 1) to sets of existential rules $\mathcal{T}(F)$. As a preliminary step, we introduce the language of existential rules formally. Later on, we explain and define the translation function. The section closes with a correctness argument, establishing a strong relationship between existential rules and N3$^{\exists}$.

4.1 Foundations of Existential Rule Reasoning

As required in the semantic framework of N3, we also consider a first-order vocabulary, consisting of constants ($\mathbf{C}$) and variables ($\mathbf{V}$), and additionally so-called (labeled) nulls ($\mathbf{N}$). As already mentioned in Sect. 2, we use the same set of constants as N3 formulae, meaning $\mathbf{C} = C$. Furthermore, let $\mathbf{P}$ be a set of *relation names*, where each $p \in \mathbf{P}$ comes with an arity $ar(p) \in \mathbb{N}$. As usual, $\mathbf{C}$, $\mathbf{V}$, $\mathbf{N}$, and $\mathbf{P}$ are countably infinite and pair-wise disjoint. Recall that we used the ternary relation name $tr \in \mathbf{P}$ to encode N3 triples. If $t_1, t_2, \ldots, t_{ar(p)}$ is a list of terms (i.e., $t_i \in \mathbf{C} \cup \mathbf{N} \cup \mathbf{V}$), $p(t_1, t_2, \ldots, t_{ar(p)})$ is called an *atom*. We often use $\mathbf{t}$ as an abbreviation for the term list $t_1, \ldots, t_n$, and treat it as a set whenever order is irrelevant. An atom $p(\mathbf{t})$ is *ground* if $\mathbf{t} \subseteq \mathbf{C}$. An *instance* is a set $\mathcal{I}$ of variable-free atoms and a finite set of ground atoms $\mathcal{D}$ is called a *database*.

For a set of atoms $\mathcal{A}$ and an instance $\mathcal{I}$, we call a function h from the terms occurring in $\mathcal{A}$ to the terms in $\mathcal{I}$ a *homomorphism from $\mathcal{A}$ to $\mathcal{I}$*, denoted by $h : \mathcal{A} \to \mathcal{I}$, if (1) $h(c) = c$ for all constants c (occurring in $\mathcal{A}$), and (2) $p(\mathbf{t}) \in \mathcal{A}$ implies $p(h(\mathbf{t})) \in \mathcal{I}$. If any homomorphism from $\mathcal{A}$ to $\mathcal{I}$ exists, we write $\mathcal{A} \to \mathcal{I}$. Please note that if n is a null occurring $\mathcal{A}$, then $h(n)$ may be a constant or null.

An *(existential) rule* is a first-order sentence of the following form:

$$r\colon \forall \mathbf{x}, \mathbf{y}.\ \varphi[\mathbf{x}, \mathbf{y}] \to \exists \mathbf{z}.\ \psi[\mathbf{y}, \mathbf{z}], \tag{11}$$

where $\mathbf{x}, \mathbf{y}, \mathbf{z}$ are mutually disjoint sets of variables, and φ and ψ are conjunctions of atoms, referred to as body ($\mathsf{body}(r)$) and head ($\mathsf{head}(r)$) of r, respectively. Rule bodies and heads will also be considered as sets of atoms for a more compact representation of the semantics. The notation $\varphi[\mathbf{x}, \mathbf{y}]$ ($\psi[\mathbf{y}, \mathbf{z}]$, resp.) indicates that the only variables occurring in φ (ψ, resp.) are $\mathbf{x} \cup \mathbf{y}$ ($\mathbf{y} \cup \mathbf{z}$, resp.).

Let r be a rule and $\mathcal{I}$ be an instance. We call a homomorphism $h : \mathsf{body}(r) \to \mathcal{I}$ a *match for r in $\mathcal{I}$*. A match h is *satisfied for r and $\mathcal{I}$* if there is an extension $h^\star$ of h (i.e., $h \subseteq h^\star$) such that $h^\star(\mathsf{head}(r)) \subseteq \mathcal{I}$. If all matches of r are satisfied in $\mathcal{I}$, we say that r is satisfied in $\mathcal{I}$, denoted by $\mathcal{I} \models r$. For a rule set Σ and database $\mathcal{D}$, we call an instance $\mathcal{I}$ a *model of Σ and $\mathcal{D}$*, denoted by $\mathcal{I} \models \Sigma, \mathcal{D}$, if $\mathcal{D} \subseteq \mathcal{I}$ and $\mathcal{I} \models r$ for each $r \in \Sigma$. We call a model $\mathcal{U} \models \Sigma, \mathcal{D}$ *universal* if $\mathcal{U} \to \mathcal{M}$ for all models $\mathcal{M} \models \Sigma, \mathcal{D}$. Note that databases $\mathcal{D}$ can be internalized as facts (rules with empty bodies) to rule set, which makes mentioning them separately obsolete. We say that two rule sets Σ_1 and Σ_2 are *equivalent*, denoted $\Sigma_1 \leftrightarrows \Sigma_2$, iff for all instances $\mathcal{I}$, $\mathcal{I} \models \Sigma_1$ iff $\mathcal{I} \models \Sigma_2$.

The new aspect introduced by rule sets, as compared to models of N3, is the incorporation of nulls. They play the role of fresh constants without further specification. The chase is a family of algorithms that soundly produces models of rule sets by continuously applying rules for unsatisfied matches. If some rule head is instantiated, existential variables may be replaced by fresh nulls in order to facilitate arbitrary constants. Although the chase is not guaranteed to terminate, it always produces a (possibly infinite) universal model [13]. For an alternative equivalence relation between rule sets, we could have equally considered equality of ground models (i.e., null-free ones). Let us define $\leftrightarrows_g$ as follows: $\Sigma_1 \leftrightarrows_g \Sigma_2$ if, and only if, for each ground instance $\mathcal{I}$, $\mathcal{I} \models \Sigma_1$ iff $\mathcal{I} \models \Sigma_2$. The following lemma, showing that $\leftrightarrows = \leftrightarrows_g$, helps simplifying the proofs concerning the correctness of our translation function later on.

Lemma 3. *$\leftrightarrows$ and $\leftrightarrows_g$ coincide.*

Proof. Of course, $\leftrightarrows \subseteq \leftrightarrows_g$ holds since since the set of all ground models of a rule set is a subset of all models of a rule set. For the converse direction, let Σ_1 and Σ_2 be rule sets, such that $\Sigma_1 \leftrightarrows_g \Sigma_2$. Towards a contradiction, assume $\Sigma_1 \not\leftrightarrows \Sigma_2$. Then there is a model $\mathcal{M}$ of Σ_1, such that $\mathcal{M} \not\models \Sigma_2$, implying that for some rule $r \in \Sigma_2$ there is a match h in $\mathcal{M}$ but for no extension $h^\star$, we get $h^\star(\mathsf{head}(r)) \subseteq \mathcal{M}$. As $\Sigma_1 \leftrightarrows_g \Sigma_2$, $\mathcal{M}$ cannot be a ground instance and, thus, contains at least one null. **Claim:** Then there is a ground instance $\mathcal{M}_g$, such that $\mathcal{M}_g \models \Sigma_1$ and $\mathcal{M}_2 \not\models \Sigma_2$. But then $\mathcal{M}_g$ constitutes a counterexample to the assumption that $\Sigma_1 \leftrightarrows_g \Sigma_2$.

It remains to be shown that the claim actually holds. Our plan is to construct $\mathcal{M}_g$ from $\mathcal{M}$ by replacing every null n in $\mathcal{M}$ by a fresh constant c_n. Unfortunately, there might be not enough constants since $\mathcal{M}$ may already use all countably infinite constant $c \in \mathbf{C}$. Therefore, we take a little detour: the set of used constants might be infinite in $\mathcal{M}$, but the constants used in the rule sets Σ_1 and Σ_2 is finite. Therefore, we will create an instance $\mathcal{M}''$ from $\mathcal{M}$ by replacing all constants c not part of Σ_1 or Σ_2 by fresh nulls n_c. Unfortunately, once again, $\mathcal{M}$ may already use all nulls $n \in \mathbf{N}$. So we have to take another detour from $\mathcal{M}$ to $\mathcal{M}'$ as follows: Let $\gamma : \mathbf{N} \to \mathbb{N}$ be a (necessarily injective) enumeration of $\mathbf{N}$. Define $\eta : \mathbf{C} \cup \mathbf{N} \to \mathbf{C} \cup \mathbf{N}$ by (1) $\eta(c) := c$ for all $c \in \mathbf{C}$ and (2) $\eta(n) := \eta^{-1}(2 \cdot \eta(n))$. Then apply η to $\mathcal{M}$ to obtain $\mathcal{M}'$. Note, for each number $k \in \mathbb{N}$, $\eta^{-1}(2k+1)$ is not a null in $\mathcal{M}'$. It holds that $\mathcal{M} \models \Sigma$ iff $\mathcal{M}' \models \Sigma$ since η is an isomorphism between $\mathcal{M}$ and $\mathcal{M}'$. Recall that isomorphic models preserve all first-order sentences [14]. Hence, $\mathcal{M}' \models \Sigma_1$ and $\mathcal{M}' \not\models \Sigma_2$.

Now we construct $\mathcal{M}''$ from $\mathcal{M}'$ by function ω mapping the terms occurring in $\mathcal{M}'$ to $\mathbf{C} \cup \mathbf{N}$, such that (1) $\omega(c) = c$ if c is a constant occurring in $\Sigma_1 \cup \Sigma_2$, (2) $\omega(d)$ is a fresh null n_d if d is a constant not occurring in $\Sigma_1 \cup \Sigma_2$, and $\omega(n) = n$ if n otherwise. ω

exists because there are countably infinitely many nulls not used by $\mathcal{M}'$. Note that ω is injective and $\omega(\mathcal{M}') = \mathcal{M}''$ uses only finitely many constants. Once again we show that $\mathcal{M}' \models \Sigma$ iff $\mathcal{M}'' \models \Sigma$ for arbitrary rule sets Σ, implying that $\mathcal{M}'' \models \Sigma_1$ and $\mathcal{M}'' \not\models \Sigma_2$. Let $r \in \Sigma$ with match h in $\mathcal{M}'$. If h is satisfied in $\mathcal{M}'$, then there is an extension $h^\star$, such that $h^\star(\mathsf{head}(r)) \subseteq \mathcal{M}'$. By definition of ω and, thus, the construction of $\mathcal{M}''$, $\omega \circ h$ is a match for r in $\mathcal{M}''$ and $\omega \circ h^\star$ its extension with $\omega \circ h^\star(\mathsf{head}(r)) \subseteq \mathcal{M}''$. The converse direction uses the the same argumentation, now from $\mathcal{M}''$ to $\mathcal{M}'$, using the fact that ω is injective.

From $\mathcal{M}''$ we can finally construct our ground instance $\mathcal{M}_g$ by ν mapping all (finitely many) constants c in $\mathcal{M}''$ to themselves and every null n in $\mathcal{M}''$ to a fresh constant c_n. It holds that $\mathcal{M}'' \models \Sigma$ iff $\nu(\mathcal{M}'') = \mathcal{M}_g \models \Sigma$ (for all rule sets Σ) by a similar argumentation as given in the step from $\mathcal{M}'$ to $\mathcal{M}''$ above. Thus, $\mathcal{M}_g \models \Sigma_1$ and $\mathcal{M}_g \not\models \Sigma_2$, which completes proof. □

4.2 The Translation Function from N3 to Rules

The translation function $\mathcal{T}$ we are going to define maps sets F of N3$^\exists$ formulae in PNF to sets of rules Σ. Before we go into details of the translation for every type of piece, we consider an auxiliary function $\mathbb{T} : C \cup E \cup U \to \mathbf{C} \cup \mathbf{V}$ mapping N3 terms to terms in our rule language (cf. previous subsection):

$$\mathbb{T}(t) := \begin{cases} v^\forall_{\mathtt{x}} & \text{if } t = \mathtt{?x} \in U \\ v^\exists_{\mathtt{y}} & \text{if } t = \mathtt{\_:y} \in E \\ t & \text{if } t \in C. \end{cases}$$

While variables in N3 belong to either E or U, this separation is only preserved in the identity of the variable, which enhances readability of subsequent examples. For the rest of this section, we assume a set of N3$^\exists$ formulae $F = \{f_1, \ldots, f_k\}$ in PNF, for which we give a translation for every piece f_i $(1 \leq i \leq k)$.

Translating Atomic Pieces. If f_i is an atomic piece, it has the form $f_i = g_1\ \ g_2\ \ldots\ g_l$ for some $l \geq 1$ and each g_j $(1 \leq j \leq l)$ is an atomic formula. The translation of f_i is the singleton set $\mathcal{T}(f_i) = \{\to \exists \mathbf{z}.\ tr(\mathbb{T}(g_1)) \wedge tr(\mathbb{T}(g_2)) \wedge \ldots \wedge tr(\mathbb{T}(g_l))\}$, where $\mathbb{T}(g_j) = \mathbb{T}(t^1_j), \mathbb{T}(t^2_j), \mathbb{T}(t^3_j)$ if $g_j = t^1_j\ \ t^2_j\ \ t^3_j$ and $\mathbf{z}$ is the list of translated existential variables (via $\mathbb{T}$) from existentials occurring in f. For example, the formula in (10) constitutes a single piece $f_{(10)}$ which translates to a set containing the rule

$$\to \exists v^\exists_{\mathtt{y}}.\ tr(\mathtt{:lucy}, \mathtt{:knows}, v^\exists_{\mathtt{y}}) \wedge tr(v^\exists_{\mathtt{y}}, \mathtt{:likes}, \mathtt{:cake}) \tag{12}$$

Translating Rules. For f_i being a rule `{`e_1`}=>{`e_2`}` we also obtain a single rule. Recall that the PNF ensures all variables of e_1 to be universals and all universal variables of e_2 to also occur in e_1. If $e_1 = g^1_1\ \ g^2_1\ \cdots\ g^m_1$ and $e_2 = g^1_2\ \ g^2_2\ \cdots\ g^n_2$, $\mathcal{T}(f_i) = \{\forall \mathbf{x}.\ \bigwedge_{j=1}^m tr(\mathbb{T}(g^j_1)) \to \exists \mathbf{z}.\ \bigwedge_{j=1}^n tr(\mathbb{T}(g^j_2))\}$ where $\mathbf{x}$ and $\mathbf{z}$ are the lists of translated universals and existentials, respectively. Applying the translation to the N3 formula in (4), which is a piece according to our definitions, we obtain again a singleton set, now containing the rule

$$\forall v^\forall_{\mathtt{x}}.\ tr(v^\forall_{\mathtt{x}}, \mathtt{:knows}, \mathtt{:tom}) \to \exists v^\exists_{\mathtt{y}}.\ tr(v^\forall_{\mathtt{x}}, \mathtt{:knows}, v^\exists_{\mathtt{y}}) \wedge tr(v^\exists_{\mathtt{y}}, \mathtt{:name}, \mathtt{"Tom"}), \tag{13}$$

which is the same rule as the one given in (8) up to α-conversion.

Translating Sets. For a set $F = \{f_1, f_2, \cdots, f_k\}$ of N3$^\exists$ formulae in PNF, $\mathcal{T}(F)$ is the union of all translated constituents (i.e., $\mathcal{T}(F) = \bigcup_{i=1}^{k} \mathcal{T}(f_i)$). The rest of this section is concerned with proving the correctness of $\mathcal{T}$, which splits into soundness — whenever we translate two equivalent N3$^\exists$ formulae, their translated rules turn out to be equivalent as well — and completeness — formulae that are not equivalent are translated to rule sets that are not equivalent. Although the different formalisms have quite different notions of models, models of a translated rule sets $\mathcal{M}$ can be converted into models of the original N3 formula by using a Herbrand argument. This turns out to be a useful property when subsequently soundness of $\mathcal{T}$ shall be proven.

Lemma 4. *Let F be a set of N3$^\exists$ formulae in PNF and $\mathcal{M}$ be a ground instance. Define* the canonical interpretation of $\mathcal{M}$ *by* $\mathfrak{I}(\mathcal{M}) = (C, \mathfrak{a}, \mathfrak{p})$ *such that*

- $\mathfrak{a}(t) := t$ *for all* $t \in C$ *and*
- $\mathfrak{p}(p) := \{(s,o) \mid tr(s,p,o) \in \mathcal{M}\}$ *for all* $p \in C$.

$\mathcal{M}$ *is a model of* $\mathcal{T}(F)$ *if, and only if,* $\mathfrak{I}(\mathcal{M})$ *is a model of* F.

Proof. By induction on the number k of pieces in $F = \{f_1, f_2, \ldots, f_k\}$:

Base: For $k = 1$, $F = \{f\}$ and f is either (a) an atomic piece or (b) a rule. In case (a), $\mathcal{T}(F) = \mathcal{T}(f) = \{\to \exists \mathbf{z}.\ \bigwedge_{i=1}^{n} tr(s_i, p_i, o_i)\}$. Every model of $\mathcal{T}(F)$ satisfies its single rule, meaning that if $\mathcal{M}$ is a model, there is a homomorphism $h^\star$ from $\mathcal{A} = \{tr(s_i, p_i, o_i) \mid 1 \le i \le n\}$ to $\mathcal{M}$. Then $\mathfrak{I}(\mathcal{M}) = (C, \mathfrak{a}, \mathfrak{p})$ with $(s_i, o_i) \in \mathfrak{p}(p_i)$ for all $i \in \{1, \ldots, n\}$. In case $\mathrm{comp}(f) \cap E = W$ is nonempty, define $\mu : W \to C$ alongside $h^\star$ (i.e., $\mu(\_:\mathtt{y}) = h^\star(v_{\mathtt{y}}^{\exists})$ for each $\_:\mathtt{y} \in W$). For each atomic formula $g_j = s_j\ \ p_j\ \ o_j$ of f, we get $\mathfrak{M} \models g_j \mu^c$ since $tr(h^\star(s_j), h^\star(p_j), h^\star(o_j)) \in \mathcal{M}$ implies $(h^\star(s_j), h^\star(o_j)) \in \mathfrak{p}(h^\star(p_j))$ and, thus, $(\mathfrak{a}(s_j\mu^c), \mathfrak{a}(o_j\mu^c)) \in \mathfrak{p}(\mathfrak{a}(p_j\mu^c))$. This argument holds for every atomic formula g_j of f, implying $\mathfrak{M} \models F$. The converse direction uses the same argumentation backwards, constructing $h^\star$ from μ. In case (b), we have $F = \{f\}$ with $f = \texttt{\{}e_1\texttt{\}=>\{}e_2\texttt{\}}$ and $\mathcal{T}(F) = \{\forall \mathbf{x}.\ \varphi \to \exists \mathbf{z}.\ \psi\}$ where φ and ψ are translated conjunctions from e_1 and e_2. Let $\mathfrak{I}(\mathcal{M})$ be a model of F. To show that $\mathcal{M}$ is a model of $\mathcal{T}(F)$, it suffices to prove, for each match h of the rule, the existence of an extension $h^\star$ (of h), such that $h^\star(\psi) \subseteq \mathcal{M}$. Let h be a match for the body of the rule and the body of the rule is a conjunction of atoms. Then σ with $\sigma(\mathtt{?x}) = h(v_{\mathtt{x}}^{\forall})$ for each universal variable in e_1 is a substitution, such that $\mathfrak{I}(\mathcal{M}) \models e_1\sigma^c$. In order to prove this claim, let $s\ \ p\ \ o$ be a triple in e_1. Hence, $tr(s,p,o) \in \varphi$ and, by the choice of h, $tr(h(s), h(p), h(o)) \in \mathcal{M}$. This implies that $(h(s), h(o)) \in \mathfrak{p}(h(p))$, which also implies $(s\sigma^c, o\sigma^c) \in \mathfrak{p}(o\sigma^c)$. As this argument holds for all triples in e_1, the claim follows. Please note that, as in case (a), this reasoning can be converted to construct a match h from a substitution σ. As $\mathfrak{I}(\mathcal{M})$ is a model of f, there is a substitution $\mu : \mathrm{comp}(e_2) \cap E \to C$, such that $\mathfrak{I}(\mathcal{M}) \models e_2\sigma^c\mu^c$. Define $h^\star := h \cup \{w \mapsto \mu(w) \mid w \in \mathrm{comp}(e_2) \cap E\}$. It holds that $h^\star$ satisfies match h since for each atomic formula $s_i\ \ p_i\ \ o_i$ of e_2, we get $\mathfrak{a}(\mu(\sigma(s_i)), \mu(\sigma(o_i))) \in \mathfrak{p}(\mathfrak{a}(\mu(\sigma(p_i))))$ implying $tr(\mu(\sigma(s_i)), \mu(\sigma(p_i), \mu(\sigma(o_i)))) \in \mathcal{M}$ and $h^\star(\mathbb{T}(x)) = \mu(\sigma(x))$ $(x \in \{s_i, p_i, o_i\})$

providing a match for $tr(\mathbb{T}(s_i), \mathbb{T}(p_i), \mathbb{T}(o_i))$ (part of the head ψ). As this argument holds for all atomic formulae of e_2, h is a satisfied match via $h^\star$. As before, the construction can be inverted, obtaining μ from $h^\star$ and σ from h, which completes the proof for this case.

Step: Let $F = \{f_1, f_2, \ldots, f_k, f_{k+1}\}$ be a set of N3$^\exists$ formulae in PNF. By induction hypothesis, $\mathcal{M}$ is a model of $\mathcal{T}(\{f_1, f_2, \ldots, f_k\})$ iff $\mathfrak{I}(\mathcal{M})$ is a model of $\{f_1, f_2, \cdots, f_k\}$. Also by induction hypothesis, $\mathcal{M}$ is a model of $\mathcal{T}(\{f_{k+1}\})$ iff $\mathfrak{I}(\mathcal{M})$ is a model of $\{f_{k+1}\}$. Thus, $\mathcal{M}$ is a model of $\mathcal{T}(F)$ iff it is a model of $\mathcal{T}(\{f_1 f_2 \cdots f_k\})$ and of $\mathcal{T}(\{f_{k+1}\})$ iff $\mathfrak{I}(\mathcal{M})$ is a model of $\{f_1 f_2 \cdots f_k\}$ and of $\{f_{k+1}\}$ iff $\mathfrak{I}(\mathcal{M})$ is a model of F. □

The previous lemma allows us to strongly relate models we obtain for translated N3$^\exists$ formulae to models of the original N3$^\exists$ formulae. In particular, it provides a soundness argument for our translation. Our correctness proof also considers completeness since, otherwise, a more trivial translation function would have been sufficient: Let $\mathcal{T}_0$ be a function mapping all N3$^\exists$ formulae to the empty rule set: All equivalent N3$^\exists$ formulae are mapped to equivalent rule sets (always $\emptyset$), but also formulae that are not equivalent yield equivalent rule sets under $\mathcal{T}_0$.

Theorem 1. *For sets F and G of N3$^\exists$ formulae in PNF, $F \equiv G$ if, and only if, $\mathcal{T}(F) \leftrightarrows \mathcal{T}(G)$.*

Proof. We prove soundness and completeness separately.

Soundness: If $F \equiv G$, then $\mathcal{M}$ is a ground model of $\mathcal{T}(F)$ iff $\mathfrak{I}(\mathcal{M})$ is a model of F (by Lemma 4) iff $\mathfrak{I}(\mathcal{M})$ is a model of G (by assumption) iff $\mathcal{M}$ is a ground model of $\mathcal{T}(G)$ (again by Lemma 4). Hence, $\mathcal{T}(F) \leftrightarrows_g \mathcal{T}(G)$ which implies $\mathcal{T}(F) \leftrightarrows \mathcal{T}(G)$ (by Lemma 3).

Completeness: If $F \not\equiv G$, then there is an interpretation $\mathfrak{M} = (\mathfrak{D}, \mathfrak{a}, \mathfrak{p})$, such that (w.l.o.g.) $\mathfrak{M} \models F$ and $\mathfrak{M} \not\models G$. **Claim:** By using a Herbrand argument once more, there is an interpretation $\mathfrak{M}_g = (C, \mathfrak{b}, \mathfrak{q})$ such that (1) for each set of N3$^\exists$ formulae H in PNF, $\mathfrak{M}_g \models H$ if, and only if, $\mathfrak{M} \models H$ and (2) there is an instance $\mathcal{M}_g$ such that $\mathfrak{I}(\mathcal{M}_g) = \mathfrak{M}_g$. If the claim holds, we obtain that $\mathfrak{M} \models F$ iff $\mathfrak{M}_g \models F$ (by (1)) iff $\mathcal{M}_g \models \mathcal{T}(G)$ for ground $\mathcal{M}_g$ with $\mathfrak{I}(\mathcal{M}_g) = \mathfrak{M}_g$ (by Lemma 4). $\mathcal{M}_g \not\models \mathcal{T}(G)$ since, otherwise, $\mathfrak{M} \models G$ (using the previous chain of arguments backwards) which contradicts our choice of $\mathfrak{M}$. Thus, $\mathcal{T}(F) \not\leftrightarrows_g \mathcal{T}(G)$ implying $\mathcal{T}(F) \not\leftrightarrows \mathcal{T}(G)$ (by Lemma 3). The claim remains to be shown: For $\mathfrak{M} = (\mathfrak{D}, \mathfrak{a}, \mathfrak{p})$, define $\mathfrak{M}_g = (C, \mathfrak{b}, \mathfrak{q})$ by (a) $\mathfrak{b}(c)$ is the identity on C and (b) $\mathfrak{q}(p) := \{(s, o) \mid (\mathfrak{a}(s), \mathfrak{a}(o)) \in \mathfrak{p}(\mathfrak{a}(p))\}$ for all $p \in C$. Instance $\mathcal{M}_g := \{tr(s, p, o) \mid (s, o) \in \mathfrak{q}(p)\}$ has property (2) (i.e., $\mathfrak{I}(\mathcal{M}_g) = \mathfrak{M}_g$). We show that $\mathfrak{M} \models H$ iff $\mathfrak{M}_g \models H$ for arbitrary sets H of N3$^\exists$ formulae in PNF by induction on the number of pieces $|H| = k$.

Base: There are two cases to consider for $k = 1$ and $H = \{f\}$: f is an atomic piece $g_1 \ \cdots \ g_l$ and f is an N3 rule `{`e_1`}=>{`e_2`}`. In case f is an atomic piece, then $\mathfrak{M} \models f$ iff $\mathfrak{M} \models f\mu^c$ for some $\mu : \mathrm{comp}(f) \cap E \to C$ iff for each atomic formula $g = s \ \ p \ \ o$ in f, $(\mathfrak{a}(\mu(s)), \mathfrak{a}(\mu(o))) \in \mathfrak{p}(\mathfrak{a}(\mu(p)))$ (all by the semantics of N3$^\exists$) iff $(\mathfrak{b}(\mu(s)), \mathfrak{b}(\mu(o))) \in \mathfrak{q}(\mathfrak{b}(\mu(p)))$ for each atomic formula g of f (by construction of $\mathfrak{M}_g$) iff $\mathfrak{M}_g \models f\mu^c$ iff $\mathfrak{M}_g \models f$ (by the semantics of N3$^\exists$).

In case f is an N3 rule {e_1}=>{e_2}, $\mathfrak{M} \models f$ iff for each substitution $\sigma : U \to C$ with $\mathfrak{M} \models e_1\sigma^c$, there is a substitution $\mu : \text{comp}(e_2) \cap E \to C$ such that $\mathfrak{M} \models e_2\sigma^c\mu^c$. Let $\sigma : U \to C$ and $\mu : \text{comp}(e_2) \cap E \to C$ be substitutions. $\mathfrak{M} \models e_1\sigma^c$ iff $(\mathfrak{a}(\sigma(s)), \mathfrak{a}(\sigma(o))) \in \mathfrak{p}(\mathfrak{a}(\sigma(p)))$ for each atomic formula $s\ \ p\ \ o$ in e_1 (by the semantics of N3$^\exists$) iff $(\mathfrak{b}(\sigma(s)), \mathfrak{b}(\sigma(o))) \in \mathfrak{q}(\mathfrak{b}(\sigma(q)))$ for each atomic formula $s\ \ p\ \ o$ of e_1 iff $\mathfrak{M}_g \models e_1\sigma^c$. The same argumentation can be used to argue for $\mathfrak{M} \models e_2\sigma^t\mu^c$ iff $\mathfrak{M}_g \models e_2\sigma^c\mu^c$. Thus, for each $\sigma : U \to C$ for which $\mathfrak{M} \models e_1\sigma^c$ implying a substitution $\mu : \text{comp}(e_2) \cap E \to C$ such that $\mathfrak{M} \models e_2\sigma^c\mu^c$, we get that $\mathfrak{M}_g \models e_1\sigma^c$ and $\mathfrak{M}_g \models e_2\sigma^c\mu^c$, and vice versa.

Step: For $H = \{f_1, \dots, f_k, f_{k+1}\}$, the induction hypothesis applies to $H' = \{f_1, \dots, f_k\}$ and $H'' = \{f_{k+1}\}$, meaning that $\mathfrak{M} \models H$ iff $\mathfrak{M} \models H'$ and $\mathfrak{M} \models H''$ (by Lemma 1) iff $\mathfrak{M}_g \models H'$ and $\mathfrak{M}_g \models H''$ (by induction hypothesis) iff $\mathfrak{M}_g \models H$ (by Lemma 1). □

5 Implementation

We have implemented our translation function $\mathcal{T}$ as a python script that takes an arbitrary N3$^\exists$ formula f, constructs its set representation F in PNF, and produces the set of rules $\mathcal{T}(F)$. The script and some additional scripts to translate existential rules (with at most binary predicates) to N3$^\exists$ formulae are available at our GitHub repository. Our implementation allows us to compare N3 reasoners with existential rule reasoners, performance-wise. As rule reasoner we chose VLog [11], a state-of-the-art reasoning engine designed for working with large piles of input data. As N3 reasoners we chose cwm [6] and EYE [19] which – due to their good coverage of N3 features – are most commonly used.

We used the Deep Taxonomy benchmark (DT)[7] and Lubm from the *Chasebench* [5]. DT was developed for the *WellnessRules* Project[8]. The benchmark simulates deeply nested subclass reasoning in varying sizes[9]. We translated the subclass declarations in forward-rules. Even though EYE also supports backward reasoning, we chose forward rules to keep the comparison fair. The *Chasebench* is a benchmarking suite for existential rule reasoning. Among the different scenaria the Chasebench provides, we picked Lubm for its direct compatibility with N3: all predicates in Lubm have at most arity 2. Additionally to the script that performs the translation, we created a script that transforms the input dataset and rules into files containing N3 facts and rules. By this script, we prepared the inputs for the N3 reasoners. To prepare the input data for VLog, we used the script implementing $\mathcal{T}$. All scripts and a brief description that allow the reader to reproduce the results we obtained on own hardware is provided on our GitHub repository. All subsequent times exclude the time for the translations, meaning they are solely related to the reasoning time. All experiments have been performed on a laptop with 11th Gen Intel Core i7-1165G7 CPU, 32GB of RAM, and 1TB disk capacity, running a Ubuntu 22.04 LTS.

[7] http://responder.ruleml.org/WellnessRules/files/WellnessRulesN3-2009-11-10.pdf
[8] http://responder.ruleml.org/WellnessRules/documentation.html
[9] N3 available at: http://eulersharp.sourceforge.net/2009/12dtb/.

Table 1. Experimental Results

Dataset	# facts	# rules	cwm	EYE	VLog
DT 1000	3004	3001	180 s	0.1 s	1.6 s
LUBM 001	100,543	136	117.4 s	3.4 s	0.2 s
LUBM 010	1,272,575	136	—	44.8 s	4.3 s
LUBM 100	13,405,381	136	—	—	47.3 s

Table 1 not only presents the runtimes of the three reasoners but also gives statistics about the sizes of the given knowledge base (# facts) and the rule set (# rules). The experiment with DT suggests that EYE is more optimized towards this dataset. However, the reasoning time of VLog is off by one order of magnitude. Conversely, VLog shows low reasoning times for all the LUBM datasets while EYE has thrown an exception after having read the input facts. The reasoner cwm is consistently slower than the other two and from LUBM 010 on, we had to break the runs of cwm up after several hours without result.

6 Related work

When originally proposed as a W3C member submission [7], the formal semantics of N3 was only introduced in an informal way. As a consequence, different systems, using N3, interpreted concepts like nested formulae differently [3]. Since then, the relation of N3 to other Web standards has been studied [2] and a W3C Community group has been formed [1], which is going to publish the semantics of N3 without functions (expected in 2022). But even with these definitions, the relation of the logic to other standards, especially outside the Semantics Web, has not been studied thouroghly.

For RDF, a subset of N3, de Bruijn and Heymans [10] provide a translation to first-order logic and F-Logic which uses very similar embeddings (e.g., a tenary predicate to represent triples) as we do in this paper, but do not cover rules. Boley [9] covers N3 in his RuleML Knowledge-Interoperation Hub by providing a translation of N3 to PSOA RuleML. This can then be used to translate to other Web-logics. But this work rather focuses on common representations of rules and leaves their semantics open.

In the realm of Description Logics (DL), rewritings in rule-based languages has its own tradition, especially when it comes to handling large amounts of data (see, e.g., [12] for a good overview of existing rewritings and their complexity, as well as more references). The goal there is to (1) make state-of-the-art rule reasoners available for DLs and, thereby, (2) use a fragment of a rule language that reflects on the data complexity of the given DL fragment. Also practical tools have been designed to capture certain profiles of the Web Ontology Language (OWL), like the Orel system [17] and, more recently, DaRLing [15]. To the best of our knowledge, such a rewriting does not exist for N3 before. Also, existential rule reasoning engines have not been compared to the existing N3 reasoners.

7 Conclusion

In this paper we defined Existential N3, a sublanguage of N3 which can be mapped to existential rules in a straight-forward manner. We provided such a mapping and showed that it preseves equivalence and non-equivalence. This result enables us to use existential rule reasoners to solve $N3^{\exists}$ problems and, thereby, connect these two rule frameworks. In order to illustrate this idea, we implemented that mapping and compared the performance of common N3 reasoners to Vlog, a state-of-the-art existential rule reasoner. We saw that the performance of the reasoners was in general comparable, but that each of them performed better on the benchmark which was more common in their own community (DT is often used for N3, LUBM for existential rules). We see that as a strong indication that it is interesting for both communities to broaden their horizon and consider problems of different fields. We hope that we provided the tools towards this direction. Future work includes comparing the reasoners at hand on a wider scale to thoroughly analyse where the implementors could learn from each other. Furthermore, the translation could be extended to cover a larger part of N3. This will give valuable insights for the definition of N3's semantics and it will make it possible to directly use existential rule reasoner for N3 and, thereby, to connect existential rules directly to the Semantic Web.

References

1. Arndt, D., Van Woensel, W., Tomaszuk, D.: Notation3: Draft Community Group Report (2021), https://w3c.github.io/N3/spec/
2. Arndt, Dörthe: Notation3 as the unifying logic for the semantic web. Ph.D. thesis, Ghent University (2019)
3. Arndt, Dörthe and Schrijvers, Tom and De Roo, Jos and Verborgh, Ruben: Implicit quantification made explicit : how to interpret blank nodes and universal variables in Notation3 Logic. JOURNAL OF WEB SEMANTICS **58**, 100501:1–100501:25 (2019), http://dx.doi.org/10.1016/j.websem.2019.04.001
4. Bellomarini, L., Sallinger, E., Gottlob, G.: The vadalog system: datalog-based reasoning for knowledge graphs. vol. 11, pp. 975–987. VLDB Endowment (2018)
5. Benedikt, M., Konstantinidis, G., Mecca, G., Motik, B., Papotti, P., Santoro, D., Tsamoura, E.: Benchmarking the chase. In: Sallinger, E., den Bussche, J.V., Geerts, F. (eds.) Proc. 36th Symposium on Principles of Database Systems (PODS'17). pp. 37–52. ACM (2017)
6. Berners-Lee, T.: cwm (2000–2009), http://www.w3.org/2000/10/swap/doc/cwm.html
7. Berners-Lee, T., Connolly, D.: Notation3 (N3): A readable RDF syntax. W3C Team Submission (Mar 2011), http://www.w3.org/TeamSubmission/n3/
8. Berners-Lee, T., Connolly, D., Kagal, L., Scharf, Y., Hendler, J.: N3Logic: A logical framework for the World Wide Web. Theory and Practice of Logic Programming (3), 249–269 (2008). https://doi.org/10.1017/S1471068407003213
9. Boley, H.: The ruleml knowledge-interoperation hub. In: Alferes, J.J., Bertossi, L., Governatori, G., Fodor, P., Roman, D. (eds.) Rule Technologies. Research, Tools, and Applications. pp. 19–33. Springer International Publishing, Cham (2016)
10. de Bruijn, J., Heymans, S.: Logical foundations of (e)rdf(s): Complexity and reasoning. In: Aberer, K., Choi, K.S., Noy, N., Allemang, D., Lee, K.I., Nixon, L., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) The Semantic Web. pp. 86–99. Springer Berlin Heidelberg, Berlin, Heidelberg (2007)

11. Carral, D., Dragoste, I., González, L., Jacobs, C., Krötzsch, M., Urbani, J.: Vlog: A rule engine for knowledge graphs. In: International Semantic Web Conference. pp. 19–35. Springer (2019)
12. Carral, D., Krötzsch, M.: Rewriting the description logic ALCHIQ to disjunctive existential rules. In: Bessiere, C. (ed.) Proceedings of the 29th International Joint Conference on Artificial Intelligence, IJCAI 2020. pp. 1777–1783. ijcai.org (2020)
13. Deutsch, A., Nash, A., Remmel, J.B.: The chase revisited. In: Lenzerini, M., Lembo, D. (eds.) Proc. 27th Symposium on Principles of Database Systems (PODS'08). pp. 149–158. ACM (2008)
14. Ebbinghaus, H.D., Flum, J., Thomas, W.: Semantics of First-Order Languages, pp. 27–57. Springer New York, New York, NY (1994)
15. Fiorentino, A., Zangari, J., Manna, M.: DaRLing: A Datalog rewriter for OWL 2 RL ontological reasoning under SPARQL queries. Theory and Practice of Logic Programming **20**(6), 958–973 (2020)
16. Kazakov, Y., Krötzsch, M., Simančík, F.: The incredible elk. Journal of automated reasoning **53**(1), 1–61 (2014)
17. Krötzsch, M., Mehdi, A., Rudolph, S.: Orel: Database-driven reasoning for OWL 2 profiles. In: Haarslev, V., Toman, D., Weddell, G. (eds.) Proc. 23rd Int. Workshop on Description Logics (DL'10). CEUR Workshop Proceedings, vol. 573, pp. 114–124. CEUR-WS.org (2010)
18. Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: A practical owl-dl reasoner. Journal of Web Semantics **5**(2), 51–53 (2007)
19. Verborgh, R., De Roo, J.: Drawing Conclusions from Linked Data on the Web: The EYE Reasoner. IEEE Software (5), 23–27 (2015). https://doi.org/10.1109/MS.2015.63