

# Advanced Query Functionalities in Natural Logic Knowledge Bases

**Andreasen, Troels; Bulskov, Henrik; Nilsson, Jørgen Fischer**

*Published in:*
International Journal of Computational Intelligence Systems

*Link to article, DOI:*
10.1007/s44196-024-00484-x

*Publication date:*
2024

*Document Version*
Publisher's PDF, also known as Version of record

International Journal of Computational Intelligence Systems (2024) 17:96
https://doi.org/10.1007/s44196-024-00484-x

RESEARCH ARTICLE

# Advanced Query Functionalities in Natural Logic Knowledge Bases

**Troels Andreasen**[1] · **Henrik Bulskov**[1] · **Jørgen Fischer Nilsson**[2]

Received: 5 December 2022 / Accepted: 26 March 2024


**Abstract**
Natural logics are formal logics characterized by expressions that bear a resemblance to simplified natural language sentences. The inherent readability of natural logics renders them suitable also for large knowledge bases. Within the realm of natural logics, the rules of logical proof are applicable directly to the sentences constructed using this framework. This direct application ensures explainability of query inferences. In the paper we describe a specific natural logic named NATURALOG, suited for ontology-structured knowledge bases. We outline how NATURALOG can be effectively implemented into a database system to facilitate deductive querying. The primary focus of this paper lies in exploring various query functionalities and elucidating methods to achieve these capabilities, also when dealing with large-scale knowledge bases.

**Keywords** Natural logics · Deductive querying · Explainability · Knowledge bases · Formal ontology

## 1 Introduction

In a number of papers we have advanced a form of natural logic, termed NATURALOG. NATURALOG is meant for qualitative domain modeling and deductive querying of logic-based knowledge bases. The key idea of this natural logic is to provide a formal logic for knowledge base sentences coming close to formulations within a useful fragment of natural language. Furthermore, the natural logic enables establishment of versatile query facilities by means of computational deductive inference. These features facilitate the much sought explainability of results computed from a knowledge base.

In [1] the basic ideas of the considered form of natural logic were put forward in [1] and in [2] they were consolidated. The recent [3] offers a comprehensive account of the proposed syntax and semantics of NATURALOG and elaborate on the devised inference rules. We describe a proposal for realizing NATURALOG as a database application in [4]. In a forthcoming paper we focus on the relation to natural language formulations. These various papers contain background references to the literature concerning natural logic. Some important key references to natural logic are [5–7].

In this paper our objective is to provide an easily accessible survey that discusses the applied concepts and principles, while directing readers to the referenced papers for detailed technical information. The proposed systems design leverages advanced retrieval algorithms and a mechanism for efficiently combining data by bulk processing using equi-joins in contemporary relational database systems. We envision in this approach to enable realization of large NATURALOG knowledge bases, e.g., in the life science domain, for provision of deductive query facilities on top of database systems.

The paper is organized as follows. Section 2 introduces NATURALOG, its graph representation and its predicate logical construal. Furthermore, various distinguished relations and compound concepts are described. Section 3 describes the logical inference rules for NATURALOG realized in a metalogic for the purpose of deductive querying. In Sect. 4 we describe how to encode NATURALOG in a database system and in Sect. 5 we explain how deductive querying of a NATURALOG knowledge base can be realized as conventional database querying. We round off the paper with a brief

Troels Andreasen, Henrik Bulskov and Jørgen Fischer Nilsson contributed equally to this work.

✉ Troels Andreasen
troels@ruc.dk

Henrik Bulskov
bulskov@ruc.dk

Jørgen Fischer Nilsson
jfni@dtu.dk

[1] Computer Science, Roskilde University, Roskilde, Denmark

[2] Department of Mathematics and Computer Science, Technical University of Denmark, Lyngby, Denmark

Published online: 18 April 2024

description of related work in Sect. 6 before the conclusion in Sect. 8.

## 2 The Natural Logic NATURALOG

A NATURALOG knowledge base simply comprises a collection of NATURALOG sentences. Let us begin the exploration by examining a pair of illustrative NATURALOG sentences:

every betacell produce insulin
insulin isa hormone

This corresponds to the natural language sentences "every betacell produces insulin" and "insulin is a hormone".

These sentences instantiate and display the general form of NATURALOG sentences, albeit simplified for the moment, as being rather straightforwardly:

*Det Cterm Relation Cterm*

Here *Det* is one of the determiners every and some. The full form is:

*Det Cterm Relation Det Cterm*

An example sentence following this form is every insulin isa some hormone, which is the full form of the above insulin isa hormone. This is to be understood as 'every amount of insulin is (also) some amount of hormone'. Thereby is stated that the class of insulin is a subclass of the class of hormones.

- *Cterm* stands for a class term. In the simplest case it is a common noun. More generally it is a nominal phrase consisting of a common noun with restrictive attributions. They may consist of relative clauses or prepositional phrases. In the latter case we also refer to *Cterm* as a concept term.
- *Relation* is a linguistically transitive verb, possibly attributed with adverbial restrictions.
- The determiners *Det* are optional. The first determiner is every by default whereas the second is some by default. Therefore we can write simply betacell produce insulin for every betacell produce some insulin.

The NATURALOG sentence form directly reflects the common and basic form of natural language declarative sentences, namely as being *subject verb object*. Accordingly, within the provided pair of example sentences, the subject and object are composed of an optional determiner followed by a common noun. The relationship is indicated through the presence of a transitive verb.

From the above two sentences we logically deduce that:

every betacell produce hormone

Such a simple deduction enables computation of answers to queries stated to the knowledge base. For instance, one could ask "which kind of cell produces hormone?" expecting betacell to be in the answer set.

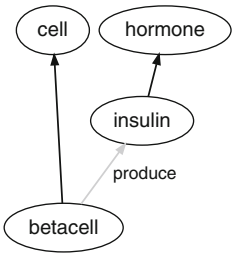



**Fig. 1** A graph representing {betacell isa cell, betacell produce insulin, insulin isa hormone}

Strictly, there is a morphologic error in the first sentence, witness 'produce' instead of the correct verb form 'produces' to unify inflections. Moreover, we use 'isa' for the verb 'is' in the second sentence following the conventions in formal ontologies. These simplifying aberrations from correct usage of language remind us that the sentences belong to a formal logic, although they look like and can be read as if they were natural language sentences.

In the present version of NATURALOG affirmative sentences, only, are admitted in the knowledge base. Negative sentences come about in answers by appeal to negation as non-provability with the CWA as well known from logic programming and database query languages.

### 2.1 Graph Representation

We have devised a convenient and instructive graph visualization for the NATURALOG sentences in a knowledge base in our above-mentioned papers. Figure 1 displays the concept graph for the sample NATURALOG sentences above. Notice that unlabelled arrows in the graph notation represent the common isa inclusion relationship by convention.

A knowledge base forms one, usually coherent, graph. In such graphs a given concept term is represented by a single node being shared by diverse sentences throughout the knowledge base. Knowledge bases are exemplified in Figs. 7 and 8 in Sect. 5. Thus, the concept graph shows how the knowledge base sentences are interrelated. The directed edges representing the relations come with quantifier symbols, although in most cases due to the mentioned default conventions they are omitted.

Every directed edge has a dual directed edge in the opposite direction due to the mathematical existence of inverse relations. They are often left implicit in graph figures. For instance the sentence betacell isa cell comes with its dual some cell isa betacell. See further Sect. 2.4.

The applied deductions in the system are topic independent and, for the basic part, purely logical. All the available knowledge is to be represented as NATURALOG sentences on equal terms. Accordingly, the system does not apply information about what 'betacell' and 'insulin' etc. "really is" except for what is given by the sentences. Consequently, to make clear that the class term 'betacell' denotes a cell, we should add to the knowledge base the sentence:

betacell isa cell

In addition, the graph form is useful for illustrating the function of the inference rules as demonstrated with many figures in [3, 4]. However, the graph form is meant solely as an explanatory device: The knowledge base consists simply of NATURALOG sentences.

The graphs appear as generalized formal ontologies, where the isa relationships make up the ontology. The ontology is adorned with the remainder relationships recorded in the knowledge base.

### 2.2 Predicate-Logical Construal of NATURALOG

NATURALOG can be reconstructed in predicate logic. The predicate logical construal serves to make precise the semantics of NATURALOG sentences. However, predicate logic is not used for representation and reasoning in the system, as to be explained.

As an example, consider again the NATURALOG sentence every betacell produce some insulin and insulin isa hormone. In predicate logic they become what we call a ∀∃ form, cf. [8], due to the order of the quantifiers:

$$\forall x(\mathsf{betacell}(x) \rightarrow \exists y(\mathsf{insulin}(y) \wedge \mathsf{produce}(x, y)))$$
$$\forall x(\mathsf{insulin}(x) \rightarrow \mathsf{hormone}(x))$$

From these two sentences follow logically every betacell produce some hormone, that is, in predicate logic:

$$\forall x(\mathsf{betacell}(x) \rightarrow \exists y(\mathsf{hormone}(y) \wedge \mathsf{produce}(x, y)))$$

Such logical deductions form the basis for query functionalities, as to be elaborated in Sect. 5. Further explanation of the relationship to predicate logic is provided in [3].

However, according to the doctrines of natural logic the computational reasoning is conducted directly at the surface forms, cf. [9]. Thus we stress that we do not translate NATURALOG sentences to predicate logic to compute inferences.

In NATURALOG the computational reasoning is achieved through a metalogical embedding. We adopt DATALOG as metalogic language, cf. Sect. 3, subsequently to be realized as a database application, cf. Sect. 4.

### 2.3 The Distinguished Class Inclusion Relation and Ontologies

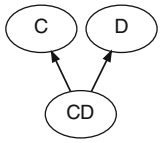
The relation isa signifies the distinguished, well-known and important class-class relation, as exemplified in the sentence form [ every ] $C$ isa $D$, stating that $C$ is a subclass of $D$.

In predicate logic this is expressed as:

$$\forall x(C(x) \rightarrow D(x))$$

This is obtainable from the ∀∃ form by $\forall x(C(x) \rightarrow \exists y(D(y) \wedge x = y))$, cf. [1, 3]. The isa relation forms a partial order with accompanying inference rules used for building formal ontologies.

**Fig. 2** Overlap of $C$ and $D$




The variant sentence form some $C$ isa $D$ is omitted in NATURALOG for explicit inclusion in a knowledge base. However, it emerges as a derived sentence as explained below.

In many cases ontologies form hierarchies. This implies that two concepts, say $C$ and $D$, overlap only if one is a subclass of the other. A deviant declaration of an overlap of otherwise disjoint classes $C$ and $D$ is obtained by stating:

[ every ] $CD$ isa $C$
[ every ] $CD$ isa $D$

Here $CD$ is a freely named intersection class, as shown in Fig. 2. It is assumed that all NATURALOG classes mentioned in the knowledge base are non-empty.
The introduced auxiliary class $CD$ guarantees overlap of $C$ and $D$ by its being non-empty by its very mentioning. The content of the class may remain unknown.

### 2.4 Inverse Relationships and Active–Passive Voice

We appeal throughout to the principle of existential import, cf. [8]. According to this principle each introduced concept is assumed to contain at least one individual. Consider the sentence:

every $C$ $R$ $D$

The inverse weaker sentence, supported by the principle of existential import, follows logically:

some $D$ $R^{inv}$ $C$

Here $R^{inv}$ is the inverse relation of $R$, bound to exist mathematically, and coming about by swapping of its two arguments. In natural language this conforms with active to passive voice switching. Accordingly, the ∀∃ form in every betacell produce insulin is accompanied by the NATURALOG ∃∃ sentence some insulin is produced by betacell. The latter sentence is endorsed by the NATURALOG syntax specification available in [3]. In the graph picture every ∀∃ or ∃∃ arc comes implicitly with a derived opposite directed ∃∃ arc, cf. Fig. 9.

### 2.5 Compound Concepts and Proxies

The exemplified concept terms in NATURALOG sentences have so far simply been common nouns. However, NATURALOG further features recursively structured compound terms consisting of a core common noun with attributed restrictions. We may assume here that they take the form of restrictive relative clauses and prepositional phrases.

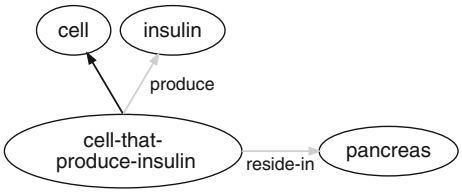



**Fig. 3** Compound concept from [3]

As an example, consider the compound concept cell that produce insulin within a statement like cell that produce insulin reside in pancreas. The NATURALOG approach to the handling of compound terms involves breaking them down internally into ∀∃ sentences. These sentences would refer to a newly introduced auxiliary concept cell-that-produce-insulin, representing the compound concept. The definition of the new auxiliary concept is introduced to the knowledge base by the pair of sentences:

cell-that-produce-insulin isa cell
cell-that-produce-insulin produce insulin

with the original sentence being replaced by:

cell-that-produce-insulin reside in pancreas.

The resulting graph is shown in Fig. 3. Notice that the arcs for a pair of defining sentences extend from the same point in the graph. The introduced concept term is specified by the above pair of defining sentences. The defined compound concept cell-that-produce-insulin becomes a proxy. The replacement of compound terms with proxies throughout simplifies the deductive query computation conducted in DATALOG.

We wish to obtain definitional contributions of proxy concepts functioning as "if-and-only-if". For the definition in the example in Fig. 3 the corresponding predicate logical expression would be:

$$\forall x(\text{cell-that-produce-insulin}(x) \leftrightarrow \text{cell}(x) \wedge \exists y(\text{produce(x, y)} \wedge \text{insulin(y)}))$$

To obtain such "if-and-only-if" definitions of proxy concepts all isa relationships, except possibly for those that follows by transitivity, are to be made explicitly present in the knowledge base. This is achieved by a so-called subsumption procedure. It traverses the knowledge base and inserts the isa relationships required for achieving the "only-if" part of the definition. cf. [3, 4]. Figure 4 illustrates addition of a subsumption isa relationship as a dashed line.

The treatment of natural language compound phrases such as noun-noun compounds and adjectival restrictions are projected along the same line. However, they call for more elaborate semantical procedures as touched in Sect. 3.2.

The verb in a sentence may also be subjected to restrictive attributions. In [3] we devised a method for handling adverbial prepositional phrases such as for instance in the verb form $X$ produce in pancreas $Y$. The proposal, which is not considered further here, relies on a metalogical predicate that relate verbs with their nominalized counterpart common noun.

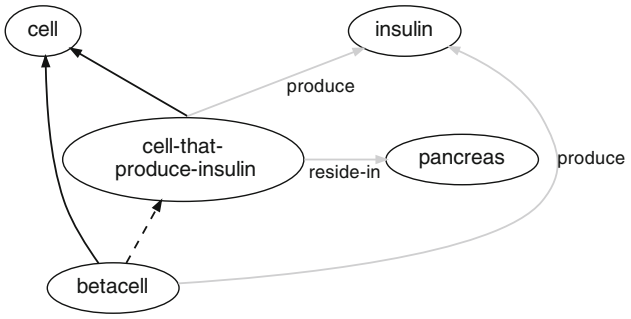



**Fig. 4** The concept betacell is now subsumed by cell-that-produce-insulin as required

## 3 Logical Inference Rules in the Metalogic

We encode NATURALOG sentences as terms in another logic used as metalogic. As metalogic we choose DATALOG, which is a sublanguage of predicate logic without function symbols and compound terms. The role of DATALOG is to serve as an intermediate explanatory form on the way to an implementation in a relational database system.

DATALOG sentences take the form of so-called definite clauses where all variables are implicitly universally quantified. In particular a clause may simply consist of an atomic sentence with a predicate symbol and argument terms.

For the DATALOG encoding of NATURALOG we introduce the predicate *kb* for storing NATURALOG sentences. As an example we have the following two factual atomic sentences in the knowledge base:

$kb$(every, betacell, produce, insulin)
$kb$(every, insulin, isa, hormone)

Notice that there is no argument position for the second determiner, which in the present simplified context is restricted to some.

Let us consider some sample inference rules. The so-called monotonicity rules, generalization and inheritance, are crucial to deductive inference and henceforth to deductive querying. The generalization rules become in DATALOG the following two clauses:

$$kb(\text{every}, C, R, Dsup) \leftarrow kb(\text{every}, C, R, D) \wedge kb(\text{every}, D, \text{isa}, Dsup)$$
$$kb(\text{some}, C, R, Dsup) \leftarrow kb(\text{some}, C, R, D) \wedge kb(\text{every}, D, \text{isa}, Dsup)$$

Similarly the inheritance rule becomes:

$$kb(\text{every}, Csub, R, D) \leftarrow kb(\text{every}, C, R, D) \wedge kb(\text{every}, Csub, \text{isa}, C)$$

The terms with a front uppercase letter are universally quantified metalogic variables. For instance $C$ and $Csub$ are variables supposed to range over class names introduced in the knowledge base.

As an example, the generalization rule being applied to the two facts above yields $kb$(every, betacell, produce, hormone) by way of:

$$kb(\text{every}, \text{betacell}, \text{produce}, \text{hormone}) \leftarrow kb(\text{every}, \text{betacell}, \text{produce}, \text{insulin}) \wedge kb(\text{every}, \text{insulin}, \text{isa}, \text{hormone})$$

Additionally, let us mention here a weakening rule and an inverse rule. The weakening rule becomes:

$$kb(\text{some}, C, R, D) \leftarrow kb(\text{every}, C, R, D)$$

while the inverse rule, as explained in 2.4, is:

$$kb(\text{some}, D, Rinv, C) \leftarrow kb(Q, C, R, D) \wedge inverse(R, Rinv)$$

These rules rely throughout on the above mentioned principle of existential import. According to this principle, as mentioned, all concept terms that appear in some sentence in the knowledge base are assumed to denote a non-empty set of individuals, cf. e.g., [1, 3, 8]. These rules are to be realized in database system implementation as explained in Sect. 4. We refer to [2–4] for a comprehensive presentation of the applied inference rules.

### 3.1 Deductive Querying of NaturaLog Knowledge Bases

As already hinted at in the introduction, query sentences are obtained by replacing a term in a NaturaLog sentence with a variable and posing the sentence as a query. Query answer terms are then computed as those term instantiations of the variables that make the instantiated sentence follow from the knowledge base using the inference rules. This reminds of answer computation in logic programming. However, whereas the answers obtained in logic programming are individual constants, the answers obtained in NaturaLog are concept terms. Moreover, answers in NaturaLog are obtained algorithmically in quite another way as explained in Sect. 4 below.

As an example consider again this tiny knowledge base where internally the NaturaLog sentences are encoded in Datalog:

$$kb(\text{every}, \text{betacell}, \text{produce}, \text{insulin})$$
$$kb(\text{every}, \text{insulin}, \text{isa}, \text{hormone})$$

The knowledge base can be queried with a clause containing variables as:

$$kb(\text{every}, X, \text{produce}, \text{insulin})$$

providing an answer in the form of an instantiation of $X$ to betacell. As a second example consider the query:

$$kb(\text{every}, X, \text{produce}, \text{hormone})$$

in this case there is no matching knowledge base sentence. However, the answer betacell is derived appealing to the above generalization rule for the linguistic object of a sentence.

As already mentioned, the Datalog explication is meant as an intermediate step towards a database realization. The encoding of NaturaLog sentences into the database representation opens for a range of versatile query- and constraint checking functionalities. In Sect. 4 we explain how these functionalities can be realized through database querying.

### 3.2 Extra-Logical Rules

The above ordinary logical inference rules enable computation of logical consequences. Let us consider some additional rules intended to enhance the versatility of the system.

The ordinary logical inference rules compute relationships not present in the given knowledge base, as we have seen above. We propose so-called materialization as rules that generate new concept terms and relate these properly to already existing terms, as cell-that-produce-hormone in Fig. 5. They serve to make sure that concept terms that potentially candidate for participation in a query answer are made present, being "materialized" as it were, in knowledge base sentences. This is detailed in [3, 4].

Materialization is meant as an internal systems feature. By contrast, the following open-ended proposals for extra-logical rules are intended to be known and under control of the domain expert building and applying the knowledge base.

1. Common sense rules. They are for instance rules that endow selected relations expressed by linguistically transitive verbs with special properties. As an example, one may choose to make the relation 'cause' logically transitive. This can be done by introducing an inference rule similarly to the above stated rules.
   Partonomic (part-whole) relationships generally call for an inference rule stating that when objects of a class $C$ are present in an area $A1$ in an application domain being part of a larger area $A2$, then the members of $C$ are said to be present in $A2$ as well:
   $$kb(\text{every}, C, \text{residein}, A2) \leftarrow kb(\text{every}, A1, \text{ispartof}, A2) \wedge kb(\text{every}, C, \text{residein}, A1)$$
   For instance given that betacells reside in the islands-of-Langerhans which are parts of pancreas it follows that betacells reside in pancreas according to this rule. Furthermore, the relation 'ispartof' may straightforwardly be made transitive with:

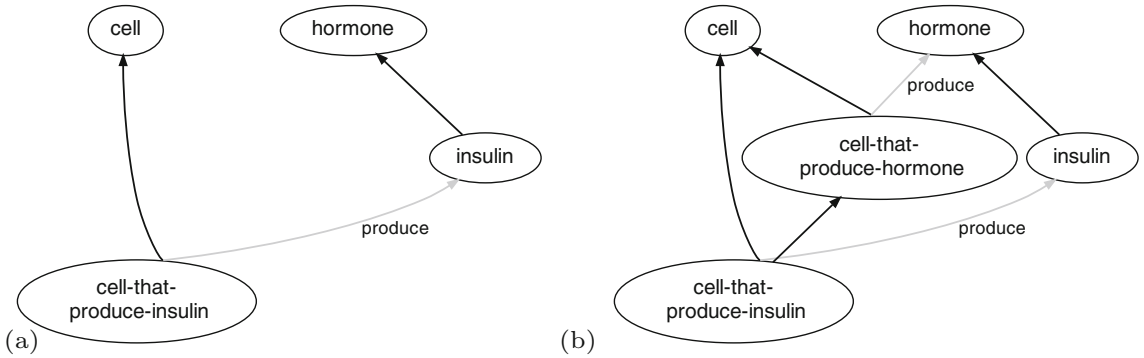



**Fig. 5** Materialization of the concept term cell-that-produce-hormone

$$kb(\text{every}, A1, \text{ispartof}, A2) \leftarrow kb(\text{every}, A1, \text{ispartof}, A12) \wedge kb(\text{every}, A12, \text{ispartof}, A2)$$

However, this mathematical partonomic principle may be constrained in practical applications.

2. Application of specific rules that introduce ad hoc properties. As an example one might state that properties of cells expressed by the verb 'produce' are "exherited" (that is, inherited upwards), as it were, to organs to which the cell belongs.
3. Linguistic rules are rules that serve to decode certain compound concept terms such as noun-noun compounds. These require abduction of the unstated relation between the two involved nouns. Given an abduced relation, the restriction on the head noun then logically resembles restrictive relative clauses. For instance 'lung disease' by rule application thereby may be resolved as disease that residein lung.

## 4 Realization of NATURALOG in Databases

Above we explained how NATURALOG sentences can readily be represented as logical atomic facts by their encoding in DATALOG. Moreover, we explained how inference rules can be expressed as definite DATALOG clauses known from logic programming.

It is well-known that DATALOG clauses in turn can be realized by the relational database operations of projection, selection, equi-join and difference, see for instance [10]. These latter operations are also indirectly available in contemporary database query languages. This opens for implementation of the various NATURALOG inference rules using standard database query expressions.

We propose and describe an implementation model in [4] applying an iterative bottom-up scheme. This differs from a query evaluation involving reasoning through a top-down goal-directed computation. This iterative bottom-up computation using the inference rules leads to partial formation of the deductive closure. This closure is calculated in a pre-processing stage, where the computed NATURALOG sentences are added to the knowledge base. This is exemplified below in Figs. 6, 7 and 8. Thus in the trade-off between space (adding closure to the knowledge base) and computation (reasoning) we accept space consumption to reduce re-computation.

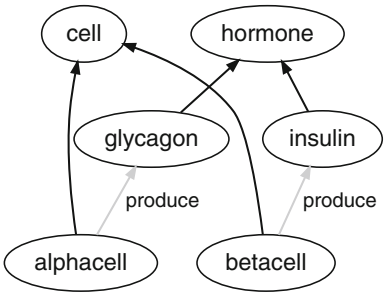



**Fig. 6** Knowledge base with given propositions

### 4.1 Encoding of Natural Logic in Database Relations

The NATURALOG sentences, as mentioned above, are meant to be stored within a database relation kb constituting the knowledge base. The knowledge base shown in Fig. 6 corresponds to that of Fig. 1 with a few added concepts. It can be represented by the following tuples:

kb(every, alphacell, isa, cell),
kb(every, alphacell, produce, glucagon),
kb(every, betacell, isa, cell),
kb(every, betacell, produce, insulin),
kb(every, glucagon, isa, hormone),
kb(every, insulin, isa, hormone)

The attributes of the kb relation, are named quant, sub, rel, obj, and represent the quantifier, subject, relation and object terms.

The pre-processing of the knowledge base iteratively builds the deductive closure applying the inference rules. New tuples that can be inferred by the rules using the present tuples are added. A first iteration of the knowledge base from Fig. 6 would add tuples such as:

kb(every, alphacell, isa, cell-that-produce-glucagon),
kb(every, alphacell, produce, hormone), ...

The closure of the graph from Fig. 6 is shown in Fig. 8. Figure 7 illustrates an intermediate stage in the iterative closure formation. In this simple example only the monotonicity, materialization and dual proposition rules contribute with new edges. In this case, three iterations was needed to provide the closure and the materialized node cell-that-produce-hormone. This latter node subsumes the materialized compounds cell-that-produce-glucagon and cell-that-produce-insulin, which were introduced in the previous iteration. The dual proposition rule would introduce inversions of all edges with appropriate quantifiers such as

kb(some, glucagon, produced_by, alphacell),
kb(some, hormone, isa, insulin)

as also shown in Fig. 9.

## 5 Query Processing of a Sample Knowledge Base

We now introduce various types of queries and describe how these can be evaluated by accessing of the kb relation. The pre-computation of the closure obtained by applying the inference rules, as explained above, implies that computation of query answers then reduces to mere selection without appeal to inference rules. The preceding steps in Sects. 3 and 4 has prepared for obtaining query answers simply by means of SQL query expressions.

### 5.1 Concept and Relation Querying

A basic query form is an open NaturaLog sentence, that is a sentence with one or more free query variables. As an example, to formulate the query "what produce insulin" the following parameterized NaturaLog sentence can be used:

X produce insulin

For the knowledge base shown in Fig. 8, this query would yield {betacell, cell-that-produce-insulin} as possible instantiations for the variable X. The query

X produce hormone

would lead to the answer {alphacell, cell-that-produce-glucagon, betacell, cell-that-produce-insulin, cell-that-produce-hormone}. The query betacell produce Y would yield the answer {insulin, hormone}.

Query expressions in SQL for such concept queries with one or more free variables are straightforwardly derived from the NaturaLog sentence form. Recall that the attributes of the kb relation, are named quant, sub, rel, obj, and represent the quantifier, subject, relation and object terms.

The first query stated above can be expressed in SQL as

```
SELECT sub FROM kb
 WHERE rel = 'produce' and obj = 'insulin';
```

and the second query becomes

```
SELECT sub, rel FROM kb
 WHERE obj = 'hormone';
```

Observe that the latter query exploits that the queried closure comprises sentences formed by means of the monotonicity rules as elaborated in Sect. 2.

Also, queries involving compound logical expressions are supported along this line. An example of a conjunctive concept query is:

X produce glucagon AND X produce insulin

This query would yield {cell-that-produce-hormone} as the only possible instantiation for the variable X.

Using a variable in the position of the relation provides possible instantiations of the relation. For instance, the query betacell R hormone yields {produce}. The sample query

X R hormone

leads to the answer {(glucagon, isa), (cell-that-produce-hormone, produce), (cell-that-produce-glucagon, produce), (cell-that-produce-insulin, produce), (insulin, isa), (alphacell, produce), (betacell, produce)}.

In the query examples above, we evaluate the default NaturaLog sentence form every $C$ $R$ $D$. As noticed, the indicated closures shown in Figs. 6 and 8 do not include sentences derived from the dual relationship rule that take the form some $D$ $R^{-1}$ $C$. This is to avoid cluttering and maintain readability of the graph rendition, especially in the last of these figures. It should be emphasized that due to the dual rule, relationships can always be read in two directions. The opposite of the given direction, however, should be read by the inverted relation, such as produce_by as dual to produce. For instance the query X R insulin to the knowledge base in Fig. 8 would yield, explicating also the quantifier: {(every, betacell, produce), (some, hormone, isa)}. The SQL expression for obtaining this would simply be:

```
SELECT quant, sub, rel FROM kb
 WHERE obj = 'hormone';
```

In the same vein, consider the query X isa cell that produce insulin. The answer is to be obtained from the sentence

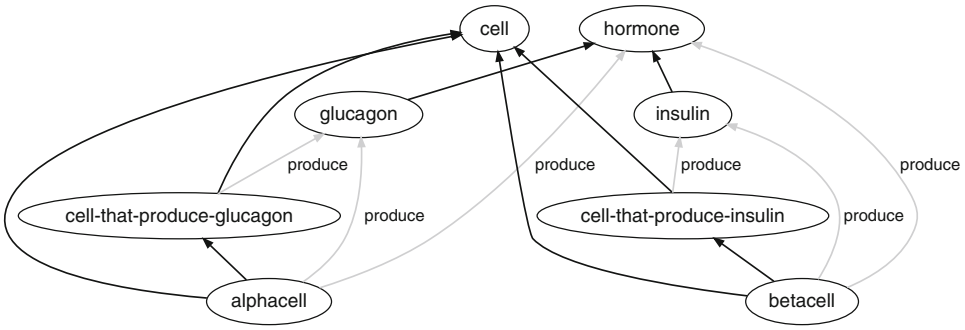



**Fig. 7** Partial closure of the knowledge base in Fig. 6 (dual propositions omitted) from [4]

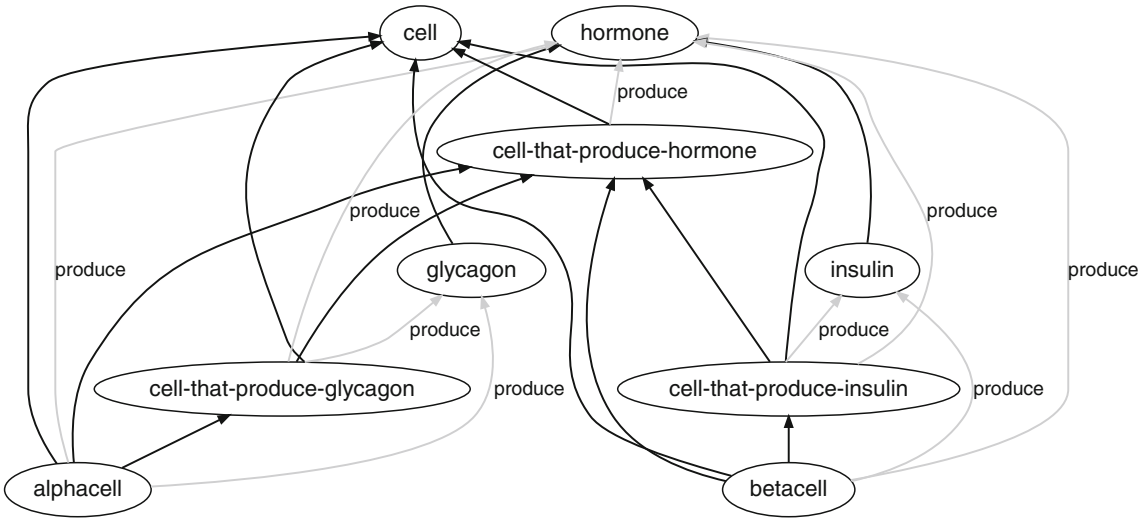



**Fig. 8** Closure of the knowledge base in Fig. 6 (dual propositions omitted) from [4]

betacell isa cell that produce insulin. However, this sentence is absent in the given knowledge base, but present in the closure, cf. Fig. 4. Accordingly, the answer is retrieved with:

```
SELECT sub FROM kb
 WHERE rel = 'isa' AND obj
= 'cell-that-produce-insulin'
```

## 5.2 Commonality Querying

NATURALOG and the supporting graph view of the knowledge base inspire to various more sophisticated query forms that afford browsing at a conceptual level. One of these is commonality querying. The commonality for a given pair of stated concepts C and D encompasses all the properties they have in common. Considering for instance alphacell and betacell in Fig. 8, the commonality would be {(produce, hormone), (isa, cell), (isa, cell-that-produce-hormone)}. This can be retrieved by the simple SQL expression:

```
SELECT rel, obj FROM kb
 WHERE quant = 'every' AND sub = 'alphacell';
 INTERSECT
 SELECT rel, obj FROM kb
 WHERE quant = 'every' AND obj = 'betacell';
```

It appears that the most interesting contribution to the answer in this case would be the most specific part, that is, {(isa, cell-that-produce-hormone)}. This can also be obtained in SQL in a straightforward manner as shown in [4].

## 5.3 Analogy Querying

The metalogic level for the encoded NATURALOGsentences in the knowledge base offers a variety of advanced retrieval opportunities going beyond mere deductive querying. As yet an example let us consider analogy computation.

In the simplest case, analogies express that in a knowledge base sentence X is R-related to U in analogy to a sentence Y being R-related to V. In this "analogy square" a corner may be unknown and therefore made subject to deductive querying. For instance, given that betacells produce insulin, what about alphacells? In DATALOG such analogies may be formalized with the clause:

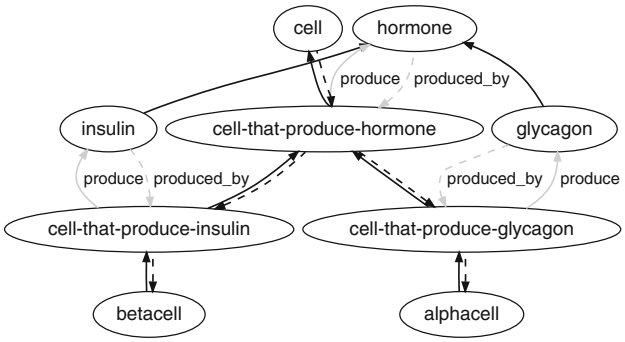



**Fig. 9** A transitively reduced version of the knowledge base in Fig. 8 (dual propositions included)

$$analogy(X, U, Y, V) \leftarrow kb(every, X, R, U) \wedge kb(every, Y, R, V)$$

In the query example this clause is to be invoked with $analogy(betacell, insulin, alphacell, V)$ to give the answer glucagon. The reformulation to SQL is straightforward. Observe that more implicit analogies may also be achieved drawing on the monotonicity rules according to the general deductive querying principles.

### 5.4 Pathway Querying

The entire knowledge base graph virtually forms a road map between all the applied concepts in a knowledge base. Appropriate ontologic structure provided by the isa inclusion relation with a universal concept at the top of the ontology ensures that all concepts are connected. This concept map can be queried by means of dedicated rules that search pathways in the graph between two stated concepts in the knowledge base. The pathway querying applies the given sentences supplemented with their duals, while other derived propositions can be ignored, if needed, to provide more intuitive answers. The closure computed by the inference rules, as shown in Fig. 8, introduces transitive edges connecting concepts which do not contribute to interesting pathways. The pathway querying is to be done on a transitively reduced knowledge base including dual propositions adopting an appropriate algorithm for selecting among the shortest paths. Pathway querying is particularly relevant in life science applications with causal relations in connection, say, with partonomy and inclusion relations.

As an example, referring to the knowledge base shown in Fig. 9, take a pathway query involving the concepts alphacell and hormone:

path(alphacell, hormone)

The answer to this query includes the pathways:

(alphacell isa cell-that-produce-glucagon, cell-that-produce-glucagon produce glucagon, glucagon isa hormone)

and

(alphacell isa cell-that-produce-glucagon, cell-that-produce-glucagon isa cell-that-produce-hormone, cell-that-produce-hormone produce hormone)

The example query path(alphacell, betacell) would involve derived dual sentences to find the pathways:

(alphacell isa cell-that-produce-glucagon, cell-that-produce-glucagon isa cell-that-produce-hormone, some cell-that-produce-hormone isa cell-that-produce-insulin, some cell-that-produce-insulin isa betacell)

and

(alphacell isa cell-that-produce-glucagon, cell-that-produce-glucagon produce glucagon, glucagon isa hormone, some hormone isa insulin, some insulin produce_by cell-that-produce-insulin, some cell-that-produce-insulin isa betacell)

The small knowledge base in Fig. 9 has only the above pathways between alphacell and betacell, excluding cycles. But for larger graphs many paths may connect the concepts in question, and simple shortest path would not be enough to select only interesting pathways. This invites further rules and heuristics to be taken into consideration, such as weighting of edges.

## 6 Related Knowledge Representation Systems

The knowledge base logics outside natural logic coming closest to NATURALOG seem to be descriptions logics and Sowa's conceptual graphs [11]. Let us first briefly compare NATURALOG with description logics [12, 13]. The fundamental difference is that NATURALOG applies the sentence form subject-verb-object (known in linguistics as SVO). This is in a more semantic view quantified triples *[quantifier] concept relation concept*. By contrast, description logics at the level of concepts is restricted to the copula form every *concept* is *concept*. The copula form is afforded as a special case of the general form in NATURALOG cf. Sect. 2.3.

As an example, the straightforward NATURALOG sentence [every] betacell produces insulin in description logic would require a rewriting effectively becoming betacell is thing that produces insulin. This limitation to copula sentences imposed by the logic tends to become awkward when one takes as departure for a logical formalization task a text in natural language. Moreover, what we consider crucial, the SVO accepted in NATURALOG aligns with the supporting

graph forms described above, which go beyond the copula forms of pure ontologies.

As a further point, description logics apparently do not support the some quantifier in front. As it appears in Sect. 2.4 this quantifier is needed in active to passive voice conversion, say, as in some insulin is-produced-by betacell from betacell produces insulin thanks to existential import. This is bound up with description logics neglecting existential import, whereas we consider it as underlying common sense in natural language. The active to passive conversion ensures that every arc comes implicitly with an oppositely directed arc in the graph rendition. In particular for copula forms as an example insulin isa hormone comes, due to inference rules, with the opposite some hormone isa insulin.

Other knowledge representation proposals, e.g., conceptual graphs [11], offer graph representations of individual sentences. However, as a point to notice, in the NATURALOG graph rendition all concepts (including syntactical subconcepts of compound concepts, cf. Sect. 2.5) in the collection of knowledge base sentences are uniquely represented as a node across the sentences in which it appears. Thus the graph forms a roadmap, as it were, for exploring paths through the knowledge base. A theorem prover for a natural logic similar to NATURALOG is described in [14] and an online prototype is provided at [15]. In comparison, our NATURALOG setup is distinguished by offering deductive querying by means of parameterized sentences yielding concept terms as answers.

## 7 The Case of Negation

Besides the differences to description logic mentioned in the previous section, there is also a fundamental distinction in the handling of negation. Whereas description logics apply classical negation with the open-world assumption (OWA), as mentioned, NATURALOG applies the closed-world assumption (CWA). This latter seems in better accord with the common assumption in textual descriptions. For instance, the negative fact that betacells do not produce glucagon may be left implicit in descriptions.

The CWA is the key principle for handling of negation in database systems and logic programming. Negative information in CWA is achieved by means of the failure-to-prove principle. Whenever a sentence does not hold (whether being present or derivable) in a knowledge base its negation is supposed to hold. Accordingly, CWA is a common-sense principle that one often relies on in daily life information handling. The non-monotonicity of CWA means that additional information may cause retraction of previously confirmed sentences.

The OWA rejects the failure-to-prove principle. This means, informally, that a considered sentence can hold, or its negation can hold, or the situation may be open. As an illustration of the difference between the two principles consider as a tiny example that there is a class cell with the two subordinate classes alphacell and betacell. Accordingly, in the knowledge base there are just the two sentences alphacell isa cell and betacell isa cell. Now, in the OWA, if we wish to state that the twin subordinate classes alphacell and betacell are disjoint, we have to make this explicit in a sentence. And this becomes tedious for large knowledge bases where typically the numerous classes at the bottom of the ontological or taxonomical hierarchies are all to be mutually disjoint. Indeed, this is assumed implicitly as a common sense principle in scientific expositions.

By contrast, in NATURALOG with CWA we simply state alphacell isa cell, betacell isa cell. Then a query: does there exist an X such that X isa alphacell and X isa betacell is answered in the negative, meaning that the two classes are disjoint. Recall here that in NATURALOG we adopt the principle of existential import, implying that all mentioned classes are non-empty by supposed presence of an anonymous individual.

Thus, CWA aligns well with taxonomies and ontology-structured knowledge bases. And if it happens to be so that alphacell and betacell are not meant as disjoint classes, one simply introduces an extra class, say alphabetacell, and posit the two sentences alphabetacell isa alphacell and alphabetacell isa betacell, as a possible exception from a hierarchical structure.

In the described version of NATURALOG all sentences in a knowledge base are affirmative ones. Negative ones come about in query answers by negation-as-failure prove.

## 8 Conclusion

We have outlined a natural logic system affording deductive querying of knowledge bases consisting of sentences in natural logic. Explainability is promoted in that the knowledge base sentences, the deduction steps and the query answers thanks to the readability of NATURALOG can be understood by a domain expert. We have explained how the NATURALOG natural logic system can be realized as a database application.

The paper proposes that deductive querying of NATURALOG knowledge base is achieved by encoding of the sentences in a metalogic. Compound NATURALOG sentences in the knowledge base are decomposed into simpler sentences in the metalogic in a way so that they are regainable to their original form. The paper focusses on deductive querying by way of inference rules in the metalogic DATALOG. The metalogic setup further offers addition of extra-logical rules such as common-sense reasoning rules. This approach in turn is implemented using conventional relational database query languages. Such an implementation enables the exploitation of available efficient query algorithms, with inference steps

reducing to bulk equijoin operations. As for negative information we rely on the CWA wellknown from databases and logic programming.

We explain how a NATURALOG knowledge base may conveniently be visualized as a labeled graph. In the graph view the class inclusion isa sentences form a formal ontology admitting compound concepts. The ontology part is extended with the remainder knowledge base sentences connecting the knowledge base concepts across the ontology. This is done in a way such that each concept in the knowledge base is uniquely represented as a node in the graph view. The graph view intimately supplements the strict logical understanding of NATURALOG. In particular the graph view exposes pathway querying, that is querying which seeks paths in the form of connecting NATURALOG sentences between two stated concepts.

We have proposed to try to face complexity problems by suggesting a pre-computation of the relevant parts of the deductive closure to avoid excessive re-computation in the inference engine during query answer computation. It is our expectation that the pre-computation in the relevant application cases is manageable with respect to complexity due to the overall ontological structure of the knowledge bases. We are conducting experiments with a knowledge base in the life-science area to validate the viability of our proposed approach, in particular with respect to computational complexity in "real application" knowledge bases. The result is to be reported in a coming paper.

**Author Contributions** The authors have contributed equally to this work. All data generated or analyzed during this study are included in this published article.

**Funding** No funding received.

**Availability of Data and Materials** Data used in examples are included in the text.

## Declarations

**Conflict of Interest** No conflict of interest relate to this work.

**Ethics Approval and Consent to Participate** No aspects that relate to ethics approval and consent to participate are compromised.

**Consent for Publication** No parts of this paper call for consent for publication.



## References

1. Nilsson, J.F.: In pursuit of natural logics for ontology-structured knowledge bases. In: The Seventh International Conference on Advanced Cognitive Technologies and Applications (2015). https://www.thinkmind.org/index.php?view=article&articleid=cognitive_2015_3_20_40044
2. Andreasen, T., Bulskov, H., Jensen, P.A., Nilsson, J.F.: Deductive querying of natural logic bases. In: Cuzzocrea, A., Greco, S., Larsen, H.L., Saccà, D., Andreasen, T., Christiansen, H. (eds.) Flexible Query Answering Systems, pp. 231–241. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-27629-4_22
3. Andreasen, T., Bulskov, H., Jensen, P.A., Nilsson, J.F.: Natural logic knowledge bases and their graph form. Data Knowl. Eng. (2020). https://doi.org/10.1016/j.datak.2020.101848
4. Andreasen, T., Bulskov, H., Nilsson, J.F.: A natural logic system for large knowledge bases. In: Tropmann-Frick, M., Thalheim, B., Jaakkola, H., Kiyoki, Y., Yoshida, N. (eds.) Information Modelling and Knowledge Bases, vol. 333, pp. 119–133. IOS Press, Amsterdam (2021). https://ebooks.iospress.nl/volumearticle/56439
5. Moss, L.S.: Syllogistic logics with verbs. J. Log. Comput. **20**(4), 947–967 (2010). https://doi.org/10.1093/logcom/exn086
6. Pratt-Hartmann, I., Moss, L.S.: Logics for the relational syllogistic. Rev. Symb. Log. **2**(4), 647–683 (2009). https://doi.org/10.1017/S1755020309990086
7. van Benthem, J.: Essays in Logical Semantics. Studies in Linguistics and Philosophy, vol. 29. D. Reidel, Dordrecht (1986). https://doi.org/10.2307/2274593
8. Nilsson, J.: A cube of opposition for predicate logic. Log. Universalis **14**(1), 103–114 (2020). https://doi.org/10.1007/s11787-020-00244-3
9. Karttunen, L.: From natural logic to natural reasoning. In: Gelbukh, A.F. (ed.) Computational Linguistics and Intelligent Text Processing—16th International Conference, CICLing 2015, Cairo, Egypt, April 14–20, 2015, Proceedings, Part I. Lecture Notes in Computer Science, vol. 9041, pp. 295–309. Springer, New York (2015). https://doi.org/10.1007/978-3-319-18111-0_23
10. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases: The Logical Level, 1st edn. Addison-Wesley Longman Publishing Co., Inc., Boston (1995). http://webdam.inria.fr/Alice/pdfs/all.pdf
11. Sowa, J.F.: Knowledge Representation: Logical, Philosophical and Computational Foundations. Brooks/Cole Publishing Co., Pacific Grove (2000). https://www.jfsowa.com/krbook/
12. Krötzsch, M., Simančík, F., Horrocks, I.: A description logic primer (2012). https://doi.org/10.48550/arXiv.1201.4089. CoRR. arXiv:1201.4089
13. Grosof, B.N., Horrocks, I., Volz, R., Decker, S.: Description logic programs: combining logic programs with description logic. In: Proceedings of the 12th International Conference on World Wide Web. WWW '03, pp. 48–57. ACM, New York (2003). https://doi.org/10.1145/775152.775160
14. Abzianidze, L.: LangPro: natural language theorem prover. In: Conference on Empirical Methods in Natural Language Processing (2017). https://doi.org/10.18653/v1/D17-2020
15. Abzianidze, L.: LangPro: Natural Language Theorem Prover (2021). https://naturallogic.pro/LangPro/. [Online; Accessed 31 May 2021]