Undefined 1 (2009) 1–5
IOS Press

FrameBase: Enabling Integration of Heterogeneous Knowledge

Editor(s): Name Surname, University, Country
Solicited review(s): Name Surname, University, Country
Open review(s): Name Surname, University, Country

Jacobo Rouces [a,*], Gerard de Melo [b] and Katja Hose [c]
[a] *Department of Electronic Systems, Aalborg University, Niels Bohr Vej 8, 6700 Esbjerg, Denmark*
E-mail: jrg@es.aau.dk
[b] *Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing, 100084, China*
E-mail: gdm@demelo.org
[c] *Department of Computer Science, Aalborg University, Selma Lagerlöfs Vej 300, 9220 Aalborg, Denmark*
E-mail: khose@cs.aau.dk

Abstract Large-scale knowledge graphs such as those in the Linked Data cloud are typically represented as subject-predicate-object triples. However, many facts about the world involve more than two entities. While n-ary relations can be converted to triples in a number of ways, unfortunately, the structurally different choices made in different knowledge sources significantly impede our ability to connect them. They also increase semantic heterogeneity, making it impossible to query the data concisely and without prior knowledge of each individual source. This article presents FrameBase, a wide-coverage knowledge-base schema that uses linguistic frames to represent and query n-ary relations from other knowledge bases, providing multiple levels of granularity connected via logical entailment. Overall, this provides a means for semantic integration from heterogeneous sources under a single schema and opens up possibilities to draw on natural language processing techniques for querying and data mining.

Keywords: knowledge representation, semantic web, n-ary relations, frames, reification, semantic integration

1. Introduction

Over the past few years, large-scale Knowledge Bases (KBs) have grown to play an important role on the Web. Increasing numbers of institutions publish their data using Semantic Web standards [4] and Linked Open Data (LOD) principles, contributing to the global LOD cloud. These KBs are mostly based on simple statements expressed as subject-predicate-object (SPO) triples, as defined by the RDF model [27]. Such triples are convenient to process and can be visualized as entity networks with labeled edges.

This data can be used for a variety of purposes. For instance, commercial search engines exploit these KBs to provide direct answers to user queries, while IBM's Watson question answering system [20, 31], which defeated human champions of the Jeopardy! quiz show, used them to find or to rule out answer candidates.

Whereas triple representations work straightforwardly for relations involving two entities, many interesting facts relate more than just two participants – a problem that has gained renewed attention in several recent papers [24, 39] as well as in the current W3C proposal to add roles to schema.org [1]. For a birth event, for instance, one may wish to capture not just the time but also the location and parents. For an actress starring in a

*Corresponding author. E-mail: jrg@es.aau.dk

movie, the name of the portrayed character may be relevant. Such facts naturally correspond to n-ary relations. In order to capture them as triples, several different representation schemes have been proposed.

Figure 1 shows some possibilities of expressing that two entities John and Mary married in 1964. These different modeling patterns are used across different KBs in the LOD cloud, which will be discussed in more detail later in Section 2.

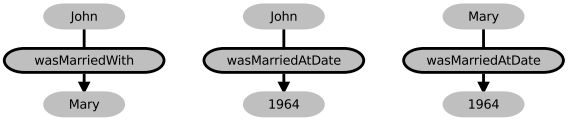
- The basic-triple pattern in Figure 1a is very basic and just establishes pair-wise connections between the arguments of the n-ary relation. If one regards every triple as representing an underlying n-ary relation with only two arguments filled, it could be said that this pattern occurs in every KB in the LOD. It lacks the expressive power to connect more than two arguments of the same n-ary relation.
- The triple-reification[1] pattern in Figure 1b is used in the YAGO ontology [28] and attempts to solve the above problem by creating an entity representing a triple, but it incurs a significant overhead that is superlinear to the number of elements in the relation, and its semantics are also problematic.
- The singleton-property pattern in Figure 1c improves the pattern above [39], but still carries some of the same problems.
- The pattern in Figure 1d is an event-centric pattern used frequently in specific parts of many KBs (e.g. Freebase [6]), usually to represent public events by means of a reduced ad-hoc vocabulary. It uses specific properties connected to an event class, which may be specific but sometimes may also be general (as the specific class can explicitly or implicitly be inferred from the specific roles).
- The pattern in Figure 1e is similar to the previous one but uses a reduced set of generic roles, and it is found in some KBs and schemas such as the Simple Event Model (SEM) Ontology [63] and LODE (Linking Open Descriptions of Events) [55].
- The pattern in Figure 1f uses a "role class" that substitutes the object of a triple and to which additional properties can be appended.
- Other ad-hoc solutions can be found[2], for instance encoding the value of the third, fourth, etc. argument in the IRI of a property connecting the first two, for instance `John marriesMaryAtDate 1964`. This reduces the overhead of the patterns from Figures 1b and 1c but at the cost of breaking the RDF standard by creating ad-hoc semantics encoded within the IRIs, which would require extra processing and in the long run produce incompatibilities and defeat the purpose of RDF of serving as a simple, homogeneous standard.

As the examples show, this sort of semantic heterogeneity leads to significant data integration challenges. One KB might use a simple binary property between two entities, whereas another may instead choose a more complex representation that accommodates additional arguments. The representations can easily be so at odds with each other that no particular mapping between entities could bridge the differences. There are entities at each side that have no counterpart at the other. This leads to several challenging problems:

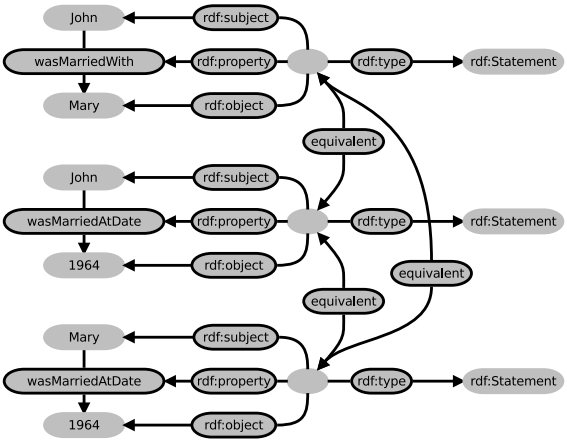
1. When **linking data**, there are currently no mechanisms to connect KBs with different modeling choices. Predicates exist to link equivalent classes, instances, or properties, but not for connecting the different patterns arising from the different modelling choices, as the ones introduced above. For instance, the entities of type `rdf:Statement` in Figure 1b cannot be linked with `owl:sameAs` to the entities of type `rdf:Property` in Figure 1c or to the entity of type `:Marriage` in Figure 1d. Existing work on ontology and KB alignment [5] is limited to finding aliases.
2. When **using structured queries**, the query must be built in a way that fits the particular modeling choices made for the respective KB. Otherwise, the recall may be as low as zero [46]. Even worse, for the case of a set of

[1] This kind of reification is different from the other kind of reification that is discussed in this paper, although both kinds of reification have in common that they consist of creating an entity for something that was not represented explicitly by a single entity before. To avoid confusion, we will refer to this kind of reification as triple-reification, while the other kind – more related to the field of linguistics – will be referred to as "reification" without any qualifier.

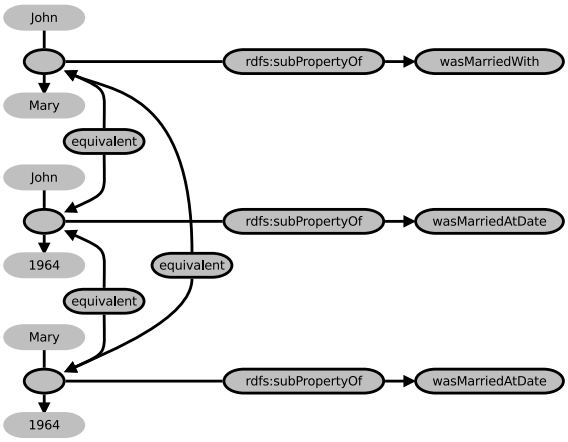
[2] `http://www.w3.org/wiki/TaskForces/CommunityProjects/LinkingOpenData/DataSets/CKANmetainformation`



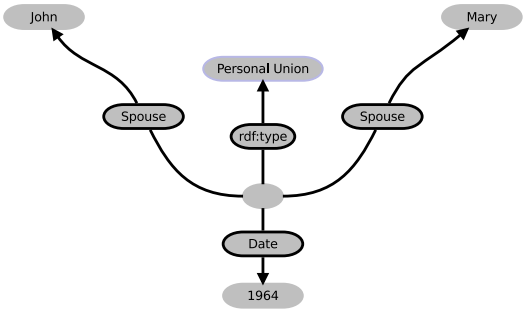
(a) Basic-triple pattern



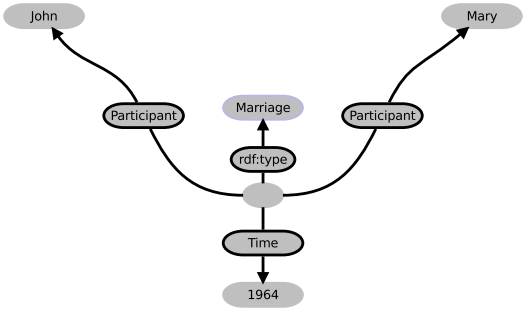
(b) Triple-reification pattern



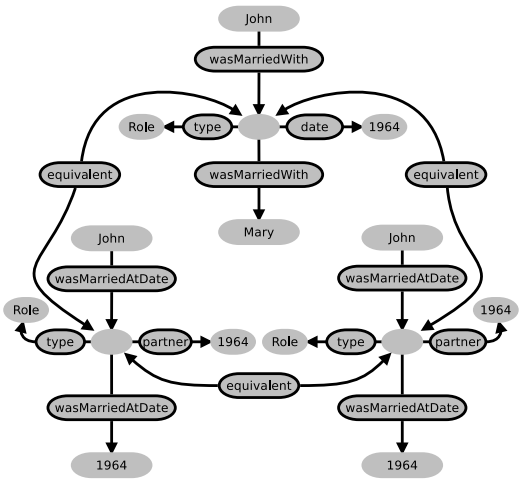
(c) Singleton-property pattern



(d) Specific-Role-Neo-Davidsonian pattern



(e) General-Role-NeoDavidsonian pattern



(f) Role-class pattern

Figure 1. The same information represented using different modelling patterns used in different KBs in the LOD.

different KBs instead of a single coherent KB, there is no simple query (as could be formulated on a single given schema) that can have a high recall across all KBs.
3. Similarly to the previous point, when **natural language interfaces** to KBs are queried, state-of-the-art systems typically attempt to map verbs and predicate phrases to RDF predicates [64]. This approach, however, cannot be applied when the KB fails to provide a compatible binary relation.

FrameBase. In this article, we describe how these problems are addressed by FrameBase [47, 49], a broad-coverage multi-layered schema that can represent a wide range of knowledge and therefore is a suitable candidate to homogeneously integrate other KBs. Figure 2 shows an example of how FrameBase represents knowledge. It combines the ability to express n-ary relations unambiguously and efficiently from the "neo-Davidsonian with specific roles" pattern in Figure 1d with the abstraction of the "neo-Davidsonian with general roles" pattern in Figure 1e, by connecting roles, together with a wide-coverage vocabulary of events, in a rich hierarchy, and it also provides the conciseness from the "Basic-triple" pattern in Figure 1a.

The latter is achieved by offering a two-layered structure with a mechanism to convert back and forth between the neo-Davidsonian representation and one based on direct binary predicates, using a vocabulary of binary properties automatically generated exploiting the ties to resources in linguistics. These are more concise and can be used when only two arguments are relevant, either in the KB or in a query.

This paper is structured as follows. Section 2 reviews related work and conducts a thorough analysis of existing approaches for modeling n-ary relations and their space efficiency. Then, an overview of FrameBase is given in Section 3. Section 4 explains how the FrameBase schema is constructed, including rules to convert between different levels of granularity and expressiveness. Section 5 provides an evaluation of the quality of the FrameBase schema. Section 6 presents examples and typology of integration rules used to capture knowledge from external KBs into the FrameBase schema, and existing methods to automatically create the simplest kinds of rules. Section 7 discusses challenges regarding the creation of more complex integration rules, and possible ways to address them. Section 8 provides a conclusion and outlines other potential lines of future work.

2. State of the Art

In this section, we review prior work in this area. In particular, Section 2.1 provides a deeper analysis of the patterns introduced in Figure 1. Section 2.2 discusses previous work on integrating knowledge. Section 2.3 introduces FrameNet, which serves as the backbone of our schema, as well as other related work based on it.

2.1. Modeling Patterns for N-ary Relations

Different approaches for modeling n-ary relations exist, which are summarized in Figure 1. Table 1 provides a novel analysis of their general space efficiency, which has consequences with regards to their applicability for large-scale KBs. Each row represents a modeling approach, for representing an event with n participants, where $k \leq \frac{n(n-1)}{2}$ is the number of pairs that are relevant to be linked by direct binary relations (we do not count inverse properties because these can be accounted for by using `owl:inverseOf`). Columns represent complexity functions for the modeling approaches.

- "All triples" indicates the total number of triples that can be materialized.
- "Core" excludes the k direct binary relations, which can always be retrieved with some sort of inference.
- "Linking event" indicates the number of triples needed to connect entities that represent the same event (aliases), which is something that is not required with a Neo-Davidsonian representation, because it can use a single one.
- "Reification Reasoning" indicates the inference system required to obtain the representation in "All triples" or "Core" from the k direct binary relations.
- "Dereification Reasoning" indicates the inference system required to obtain the k direct binary relations or the representation in "All triples" from the representation in "Core". Definite clauses are a kind of rules that can be expressed as a disjunction of logical atoms with only one negated, which is the consequent

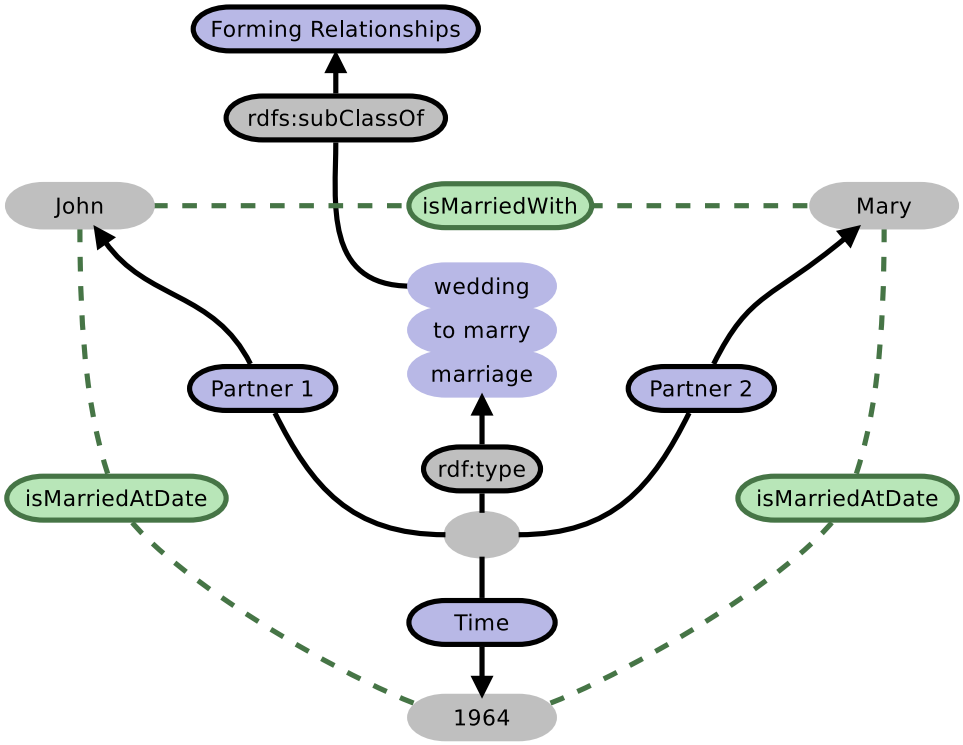


Figure 2. Knowledge represented the examples in Figure 1, represented under the FrameBase model, which combines expressiveness with conciseness by combining different representation layers (reified in blue, dereified in green).

	All triples	Core	Linking event	Reif. Reasoning	Dereif. Reasoning
Triple-Reification	$(n+4)k$	$(n+3)k$	$k(k-1)$	$4k$ Def. clauses	k Def. clauses
Singleton Property	$(n+2)k$	$(n+1)k$	$k(k-1)$	$2k$ Def. clauses	1 Def. clause / RDFS
Schema.org Roles	$(n+3)k$	$(n+2)k$	$k(k-1)$	$3k$ Def. clauses	k Def. clauses
Neo-Davidsonian	$1+n+k$	$1+n$	0	$3k$ Def. clauses	k Def. clauses

Table 1
Complexity associated to different approaches for modeling n-ary relations.

when it is written as an implication (rule). In this context, the atoms are of the form triple(subject,predicate,object). In Section 4.3, we will describe in more detail these rules for the case of FrameBase.

Figure 1 can be regarded as a specific case of Table 1 with $n = k = 3$. Each modeling approach will be discussed in detail in the following subsections.

2.1.1. Basic-Triple Pattern

A common way to represent n-ary facts is to simply decompose them directly into binary relations between two participants [14]. However, in doing so, important information may be lost. For instance, given a triple with property `wasMarriedOnDate` and two triples with `gotMarriedTo`, we cannot be sure to which marriage the given time span applies. This is shown in the example in Figure 1a.

2.1.2. Triple-Reification Pattern

The RDF standard includes a method for performing triple reification [27], which introduces a new Internationalized Resource Identifier (IRI) for a statement and then describes the original RDF triple using three new triples with `subject`, `predicate`, and `object` properties. Subsequently, arbitrary properties of the statement can be captured by adding further triples about it.

Triple-reification is used in the different versions of YAGO [28, 58, 59] to attach additional information to the event represented by the original RDF triple (evoked by its property). It has also been proposed in the W3C WebSchema drafts [1]. This modelling pattern is exemplified in Figure 1b (in YAGO, `marry` is labelled as `isMarriedTo`, but this does not change the semantics). Using the triple-reification pattern in this manner has the advantage that both the original triple as well as the triple-reified triple can be present in the KB and queries that do not require the additional information can still use the original binary relation directly. However, this also has several drawbacks:

- Formally, the event represented by a triple and the triple as a statement are different entities with different properties. For instance, an institution may endorse the triple as a statement without endorsing the marriage. Using triple-reification, both are represented by the same RDF resource identifier, which conceptually is meant to be unambiguous. This is a potential source of confusion and inconsistency.
- The number of triples increases by a factor of 4. For each triple `S P O`, one has to add `T a rdf:Statement`, `T rdf:subject S`, `T rdf:predicate P`, and `T rdf:object O`. These do not add any new information themselves but are merely a prerequisite for then being able to extend the original binary relation to an n-ary relation by subsequently adding more triples with `T` as subject.
- The advantage of being able to include the original non-triple-reified triple only applies for the primary binary relation, and not for the other $\frac{n(n-1)}{2} - 1$ ones that can be formed (not counting inverses). Some of these may be rare or irrelevant, but others may be important and are indeed used in YAGO (e.g. `yago:bornAtPlace`, `yago:bornOnDate`).
- The choice of the primary pair of entities and their binary relation (John and Mary in Figure 1b) is arbitrary, and a third party willing to query the KB cannot replicate the choice independently. If their choice is different, they will not obtain any results. A possible solution, which is actually implemented in YAGO, is to include the triples for the other pairs and reify them, too, but this adds yet another factor of overhead, besides data redundancy that would complicate updates.
- When two or more different events share the same values for the primary pair of arguments, they also share the same triple, but require separate triple reifications, producing non-unique triple identifiers. For example, if there are two flight connections between Paris and London with different airlines, the triple `:Paris yago:isConnectedTo :London` will be triple-reified twice in YAGO, with two different triple identifiers. Or, related to the example in Figure 1b: John and Mary could have divorced and married again, in which case the triple `:John yago:isMarriedTo :Mary` would also have to be triple-reified twice.

If the triplestore implementation makes use of quads[3], the 4-fold overhead can be avoided (though the underlying storage needs a new column), but the other disadvantages still remain. Quad-based singleton named graphs [27] could be used instead of triple-reification, the problems being the same.

2.1.3. Singleton-Property Pattern

The "singleton property" approach [39] aims to solve some of the issues with triple-reification by instead declaring a subproperty of the original property in the primary pair, and using this subproperty as the subject for the other arguments of the n-ary relation. This is shown in Figure 1c.

While the approach enables us to use RDFS reasoning to obtain the triple with the parent property that relates two of the participants, and also reduces the overhead of triple-reification, it still suffers from the problems mentioned above related to the existence of a primary pair. For one, the non-triple-reified binary relationships for the other pairs cannot be inferred from that subproperty.

[3] `http://www.w3.org/TR/n-quads/`

2.1.4. Role-Class Pattern

Schema.org is an effort sponsored by Google, Yahoo, and Microsoft to establish common standards for semantic markup in Web pages. It offers a method to qualify additional information to a binary predicate [2], which in practice is equivalent to representing the n-ary relation arising from adding arguments to the binary relation underlying the binary predicate. This works by substituting the object of the binary predicate with a fresh instance of a class *Role* (or a subclass thereof with its own properties), and appending to this role instance the original object by means of the same binary predicate, alongside other properties such as time, instrument, etc. In order to avoid confusion, it is relevant to note that Schema.org's use of the term "role" differs from its standard use in linguistics, which are qualifying properties such as agent and patient [22]. This definition has also been adopted in ontologies, for instance `CaseRole` in the SUMO ontology [53]. An example of the role-class pattern is shown in Figure 1f. Another example, originally used by Schema.prg contributors, uses the triple `:SanFrancisco49ers schemaorg:athlete :JoeMontana`, which would be converted to:

```
:SanFrancisco49ers schemaorg:athlete _:x
_:x a schemaorg:Role .
_:x schemaorg:athlete :JoeMontana .
_:x schemaorg:startDate "1979" .
```

This transformation offers a certain level of compatibility between the simple pattern with the direct binary predicate and the complex pattern, because the binary predicate is preserved in the complex pattern, with the same subject. However, the object changes, and therefore the simple pattern as such is not truly preserved after the transformation. Besides, the definition or original contract of the direct binary predicate is broken in the complex pattern. For example, `schemaorg:athlete` has SportsTeam and Person as domain and range respectively, and the semantics is that the object is a person that plays in the team denoted by the subject. However, none of the two usages in the complex pattern follow this: one has SportsTeam and Role as domain and range, and the other has Role and Person. Using RDFS-like inference one would infer the role instance is also a participant, and other participants would be attached.

An example of how this conflation can lead to problems can be fully appreciated with non-transitive predicates. In case the predicate was `somekb:fatherOf`, people's children would become their grandchildren after the transformation.

Furthermore, the complex pattern produced by this method, given a direct binary predicate between two entities and a further qualifying value (like time in the example), is not equivalent to the one produced by another binary predicate between one of these entities and the qualifying value. This produces a similar effect of redundancy as in the method using triple-reification.

2.1.5. Neo-Davidsonian Pattern

Another approach, and the one that FrameBase adopts, is to make use of neo-Davidsonian representations [30, p. 600f.]. This means that we first define an entity that represents the event or situation (also referred to as a *frame*) underlying the n-ary relation. Then, this entity is connected to each of the entities filling the n arguments by means of a property describing the *semantic role* [24, 40] associated with each argument position.

The process of converting from the binary representation to the neo-Davidsonian one is called reification, but this is different from *triple-reification* as discussed earlier. In triple-reification, an entity is defined that stands for a whole triple so that additional triples can be used to describe the reified triple as a unit that represents a statement. However, in the context of event semantics, reification is used to denote the process by which an entity is defined that refers to the event, process, situation, or more generally, frame, evoked by a property or binary relation. Having done this, additional information about it can then easily be added. Both kinds have in common that a new entity is defined to refer to something that before was not explicitly represented by an entity in the KB, but in one case it is an RDF statement, while in the other it is an event.

Advantages. Table 1 compares the neo-Davidsonian approach to the alternatives. These require a lot more triples when several direct binary relations need to be included. In the worst case, $k = \frac{n(n-1)}{2}$ despite discounting reciprocal relations, but even if not all of these relations are relevant, connecting all agents and possibly patients to all other elements would be relevant, which would easily satisfy $k > n$.

Semantic Heterogeneity. Even when using the neo-Davisonian approach, there are different ways to do so, corresponding to different levels of granularity for the events and the semantic roles: from a very small set of abstract generic ones [55] to more specific ones [6].

The Simple Event Model (SEM) Ontology [63] uses the general-role neo-Davidsonian pattern in Figure 1e. It defines four very general entities, *Event*, *Actor*, *Place*, and *Time*. It also establishes a framework for creating more specific ones by extending these, but it does not provide these extensions, nor ways to integrate existing KBs in a way that would solve the problem of semantic heterogeneity. Similarly, LODE (Linking Open Descriptions of Events) [55] specifies only very general concepts such as the four just mentioned.

Freebase [6] was built both by tapping on existing structured sources and via collaborative editing. Although it uses its own formalisms, there are official and third-party translations to RDF. Freebase makes use of *mediators* (also called *compound value types*, CVTs) as a way to merge multiple values into a single value, similar to a `struct` datatype in C. An example of a CVT is `/people/marriage`, which has outgoing properties such as `/people/marriage/spouse`, `/people/marriage/from`, `/people/marriage/to`, and `/people/marriage/type_of_union`. There are around 1,870 CVTs in Freebase (1,036 with more than one instance) and around 14 million composite value instances. These CVTs do not represent frames or events per se, but are regarded as complex data types. Some CVTs, for instance, connect a number and a unit. Still, in terms of their structure, they correspond to the specific-role neo-Davidsonian pattern in Figure 1d. However, Freebase places a number of restrictions on CVTs. For instance, CVTs cannot be nested, and thus, if a CVT involves a monetary value, it cannot re-use the existing Dated Money Value CVT, but needs to include separate entries for the amount and currency. Also, there is no hierarchy or network-like organization, and thus Freebase does not capture any particular relationship between similar CVTs such as the film performance and TV guest role CVTs.

2.2. Knowledge Integration

Connecting and integrating different knowledge sources is a long-standing problem. For KBs, there has been substantial work on ontology alignment [16] to identify matching classes from different sources, and in some cases also instances and properties [35, 38, 57].

However, relatively little work has considered scenarios in which the same type of ontological knowledge is modeled in different ways, as in the different modeling patterns illustrated in Figure 1 and explained in Section 2.1. In these cases, alignment by means of binary properties such as equivalence or subsumption is no longer sufficient, because an entity in a KB may not have a direct counterpart in another KB. For instance, neither any of the properties in Figure 1a, nor the statement instance in Figure 1b, the subproperty in Figure 1c, or the event instance in Figure 1d can be connected by `owl:sameAs`, `owl:equivalentClass`, `rdfs:subClassOf`, `rdfs:subPropertyOf`, `owl:equivalentProperty`, `skos:exactMatch`, `skos:closeMatch`, or any other binary relation.

The EDOAL (Expressive and Declarative Ontology Alignment Language) format [11] has been proposed to express complex relationships between properties. It defines a way to describe complex correspondences but it does not address how to create them. Similarly, complex correspondence patterns between ontologies have been described and classified in an ontology [54]. However, this approach does not provide any method to create the correspondence patterns, neither fully nor semi-automatically. The iMAP tool [15] searches a space of possible complex relationships between the values of entries in two KBs, e.g., `room-price = room-rate * (1 + tax-rate)`, but these are limited to specific types of attribute combinations. The S-Match tool [26] makes use of formal reasoning to prove possible matches between ontology classes, involving union and intersection operators, but it does not address complex matching of properties beyond this. The work from Ritze et al. [45] uses a rule-based approach to detect specific kinds of complex alignment patterns between entries in small ontologies.

Unlike previous work, the approach presented in this paper does not focus on matching pairs of entities but provides techniques to match knowledge

that can also be expressed with complex patterns involving multiple entities at one side. However, these techniques can be combined with the existing work on creating the one-to-one mappings.

2.3. FrameNet

FrameNet [21, 52] is a well-known resource in natural language processing (NLP) that defines over 1,000 *frames*, which represent abstract concepts that encompass situations, events, or processes. These are evoked by certain words, called *Lexical Units* (LUs), which can be any part of speech: nouns, verbs, adjectives, etc. For example, the verb *to buy* and the noun *acquisition* can evoke (depending on the intended sense) a "commercial transaction" frame. Frames have associated participants (called *Frame Elements* or FEs, for short), For instance, the "commercial transaction" frame has FEs for the seller, the buyer, the goods, and so on.

FrameNet includes a corpus of text that has been annotated with frames and FEs. Each annotation consists of a frame and an LU that appears (possibly inflected) in a piece of text, and some FEs whose values also appear in the text. This information can be used for training semantic role labelling (SRL) systems, also known as semantic parsers, to extract semantics or meaning from arbitrary text.

There has been previous work on producing conversions of FrameNet to RDF as a resource [41] instead of a schema. Also, previous work [23] has proposed a framework, in the form of a meta-schema, for using frames as units of meaning to address the semantic heterogeneity problem. This work proposes a model for generating schemas from FrameNet, but does not produce a specific one.

FRED [43] builds semantic representations of text, based on Discourse Representation Theory, VerbNet, and its linking to FrameNet. Unlike FrameBase, it does not integrate existing knowledge bases.

3. System Overview

As seen in the previous section, there are a number of different patterns used to represent n-ary relations in KBs.

This paper describes the construction of FrameBase, an extensible KB schema that allows for representing a wide range of knowledge, aiming at an optimal balance between the existing modelling patterns. The paper also discusses methods to integrate knowledge from external KBs.

FrameBase consists of two layers. The more expressive but also more verbose layer of the FrameBase schema is referred to as the *reified layer*. It consists of classes, representing frames, which can be events, situations, processes of a very general kind. It also contains frame-element properties that specify qualities about frame instances: agents participating in different ways, time, place, cause, consequence, instrument, etc. The frames are organized in a rich hierarchy of macroframes, clustermicroframes, and synset- and LU-microframes, ordered here from more general to more specific kinds of frames. Synsets and LUs (Lexical Units) are concepts imported from WordNet [17] and FrameNet [3], respectively, which are both resources from computational linguistics. FrameNet constitutes the backbone of FrameBase and is a compilation of such frames and FEs to annotate the semantics of natural language. WordNet is a computational lexicon that includes word senses grouped by synonymy and other semantic relations. Both synsets and LUs are closely related to sense-disambiguated words and therefore they are used to produce the most specific frames, whereas clustermicroframes and macroframes represent groups of near-synonymous or related concepts.

The less verbose but also less expressive layer of the FrameBase schema is the *reified layer*, which consists of direct binary predicates (DBPs). These are properties for simple binary relationships between elements of a given frame. Rather than having to query such relationships via a common frame instance, this layer enables direct querying of these binary relationships.

Data from external KBs in the LOD cloud can be imported using *integration rules*, which can create FrameBase instance data from the instance data of the external KBs. This work also describes the creation of these rules in manual, semi-automatic and automatic ways, exploiting the linguistic aspects of FrameBase inherited from FrameNet. The results for automatic and semi-automatic methods are evaluated. Examples are also provided of how the resulting FrameBase instance data can be queried. Figure 3 provides a general overview of the dataflow in the FrameBase system.

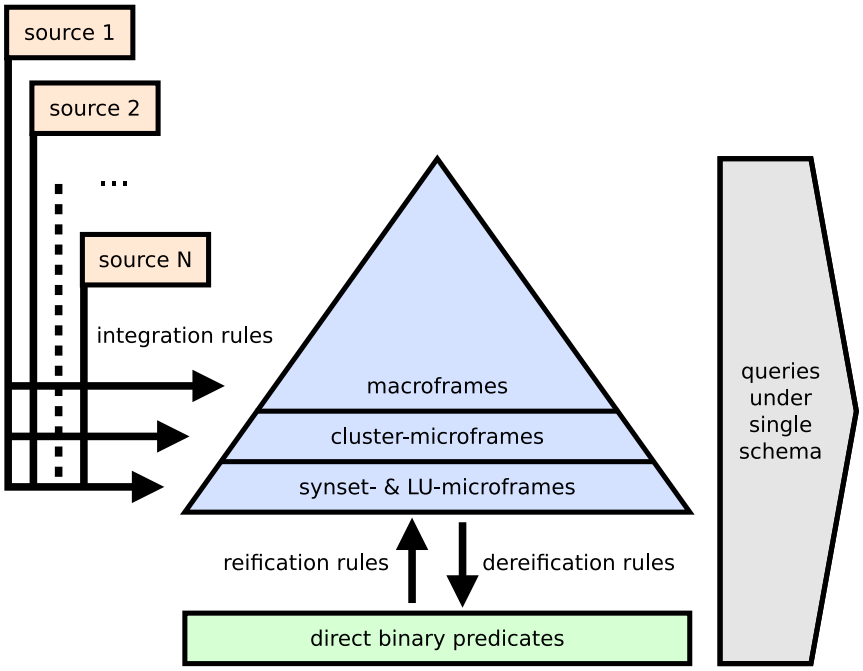


Figure 3. Overview of the structure of the FrameBase system.

3.1. FrameNet-based Representation

The use of FrameNet as the backbone of FrameBase is motivated by the following considerations.

- FrameNet has long been used to describe the semantics of general natural language. It thus provides a relatively large and growing inventory of frames and roles, with a coverage of different domains. The average number of FEs per frame is 9.45.
- FrameNet comes with a large collection of English sentences annotated with frame and frame element labels. This data led to the task of automatic *semantic role labeling* (SRL) [25] of text, now one of the standard tasks in NLP. This strong connection to natural language facilitates question answering and related tasks.
- While FrameNet's lexicon and annotations cover the English language, its frame inventory is abstract enough to be adopted for languages as different as Spanish and Japanese [56]. This also makes it more suitable as a basis for language-independent knowledge representation than more language-specific syntax-oriented SRL resources such as PropBank [32], although being more abstract can make the SRL task more challenging.
- In terms of what is expressed as a frame and what is expressed as a role or frame element, FrameNet provides a reasonable level of granularity for the phenomena that humans care to describe. From a theoretical perspective, there is no universally appropriate single level of reification. Any frame element might itself be reified, and any two elements of a frame could be connected directly by a predicate. Using FrameNet strikes a well-motivated balance, at a point that is granular enough to constitute a model for natural language semantics. However, as Section 4.3 will explain, a second level of representation is provided in FrameBase, which is based on the direct binary predicates between frame elements, and therefore less expressive but more concise.

4. FrameBase Schema Creation

The FrameBase schema consists of a reified layer and a dereified layer, connected by inference rules. The reified layer provides a rich hierarchy of frames

and FEs, with English lexical labels. Te dereified layer provides direct binary predicates that can be used between the values of the FEs. The creation of the schema is carried out in the following steps.

a) **FrameNet–WordNet Mapping.** First, a high-precision mapping is created between FrameNet and another well-known lexical resource called WordNet [17], which will be used to enrich the lexical coverage and relations of the FrameBase schema. This is described in Section 4.1.
b) **Hierarchy Construction.** FrameNet, WordNet, and their mapping are used to create a rich hierarchy of frames and FEs that has very wide coverage and is also extensible. This involves creating general *macroframes*, extracted from FrameNet, as well as specific *LU-microframes* and *synset-microframes* extracted from FrameNet and WordNet, respectively. However, these microframes are too fine-grained, with separate entries for synonyms and near-synonyms. For instance, there are distinct LUs for *get* vs. *obtain*. This is a problem for knowledge representation because it increases the sparsity of data. At the same time, some macroframes are very coarse-grained, as mentioned above, so direct inheritance from a common macroframe cannot be used as a criterion for considering LU-microframes semantically equivalent. For instance, various kinship relationships such as *mother*, *sister-in-law*, etc. are lumped together under the same macroframe. This wide range of LUs may stand in various lexical-semantic relationships without these being indicated, including synonymy, antonymy, or nominalization. The only characteristic they have in common is that, by definition, they evoke a similar kind of situation. Therefore, neither the fine-grained nor the coarse-grained levels are ideal for knowledge representation purposes. In FrameBase, this is addressed by providing a novel intermediate level composed of *cluster-microframes* that group together LU-microframes and synset-microframes that have equivalent or near-equivalent meanings, solving the problem described above. The children of each cluster-microframe are conntected in a clique with the property `:isSimilarTo`. The creation of the hierarchy is described in Section 4.2. An example of two resulting sibling cluster-microframes with all their members can be appreciated in Figure 4. Without the extended hierarchy, it would not be possible to determine that two instances of `:frame-Quitting_a_place-withdraw.v` and `:frame-Quitting_a_place-withdrawal.n` are equivalent (and optionally, they can be converted to the same type `:frame-Quitting_a_place-cluster-retreat.v` if desired, with external logic or by adding the triple `:isSimilarTo rdfs:subPropertyOf owl:sameClassAs` in an OWL-enabled triple-store).
c) **Automatic Reification–Dereification Mechanism.** Reification–dereification (ReDer) rules are created, in the form of definite clauses that allow a KB to be stored or queried independently of whether reified frames or dereified direct binary predicates are used. This mechanism may also be used to reduce overhead in the KB. The structure, implementation, and creation of ReDer rules is described in Section 4.3.

4.1. FrameNet–WordNet Mapping

While FrameNet [21, 52] is the largest high-quality inventory of semantic frame descriptions and their participants, WordNet [17] is the most well-known resource capturing meanings of words in a lexical network, covering for example nouns and named entities missing in FrameNet. WordNet, for instance, serves as the backbone of YAGO's ontology. This section proposes a novel way of mapping the two resources, which later enables us to integrate both of them into FrameBase's schema.

WordNet contains synsets, which are sets of sense-disambiguated synonymous words with a given part of speech (POS), such as noun or verb. FrameNet contains lexical units (LUs), which are also POS-annotated words associated with frames. Because of the semantics of the containing frame, LUs are also disambiguated to a certain extent, though not with the same granularity as in WordNet (for instance, WordNet has different senses for the verb *to assert* corresponding to stating something categorically and to declaring or affirming something solemnly as true; this is a nuanced difference that is conflated under a single LU in the frame *Statement*). The objective at hand is to produce an alignment of synsets and LUs with the same meaning, which can be later used to enrich

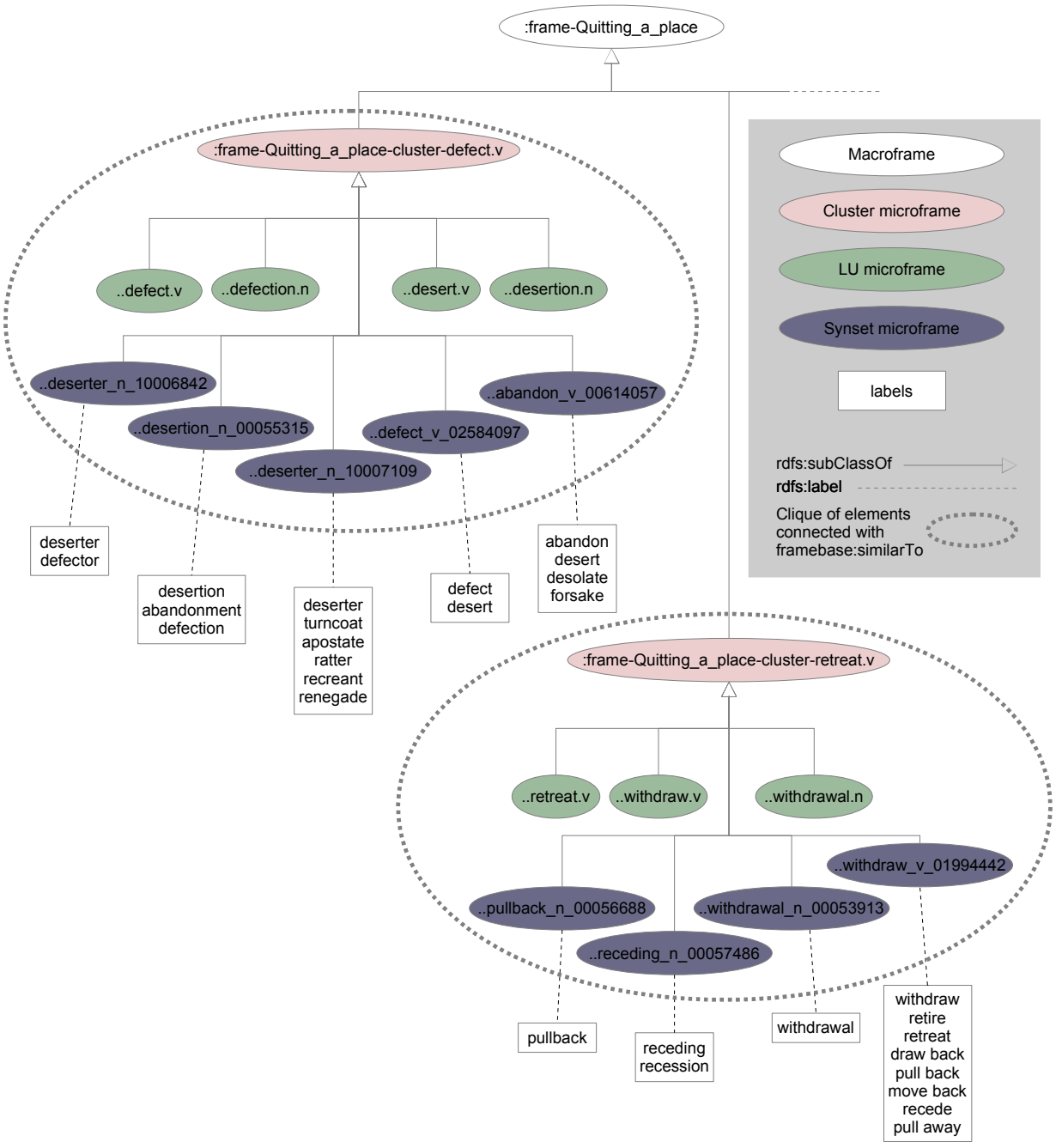


Figure 4. Example of some microframes and labels under the general frame class `:frame-Quitting_a_place`. The initial part of the names of classes is common and has been ommitted.

:frame-Personal_relationship-marital.a	...-wn_cohabit_v_02651193
:frame-Personal_relationship-married.a	...-cohabitation.n
:frame-Personal_relationship-wn_marital_a_02852920	...-wn_cohabitation_n_01054876
:frame-Personal_relationship-marriage.n	...-cohabit.v
:frame-Personal_relationship-wn_marriage_n_13963970	...-buddy.n
:frame-Personal_relationship-wn_date_n_08385009	...-wn_buddy_n_09877951
:frame-Personal_relationship-engagement.n	...-chum.n
:frame-Personal_relationship-date.v	...-pal.n
:frame-Personal_relationship-wn_go_steady_v_02486232	...-wn_chummy_a_00452114
:frame-Personal_relationship-wn_widow_v_00360337	...-wn_pal_v_02588871
:frame-Personal_relationship-widow.v	...-wn_chummy_a_01075524
:frame-Personal_relationship-wn_widow_n_10780284	...-wn_friend_n_10112591
:frame-Personal_relationship-widow.n	...-wn_friendship_n_13931145
	...-friend.n
	...-friendship.n

Figure 5. Example of six clusters of LU- and synset-microframes under the macroframe `:frame-Personal_relationship` (which is also how the IRI of the children microframes start, but this is omitted in the second column). The connection of derivationally related words evoking nearly equivalent situations or relations (in general, frames) can be appreciated in all of them. For instance, LU-microframes *friend* and *friendship* are connected to the homonymous synset-microframes by the FrameNet-WordNet mapping, and the two synset-microframes are connected by the WordNet relation "derivationally related from".

FrameBase's FrameNet-based schema with relations and annotations from WordNet.

More specifically, the objective is to map each LU to exactly one synset. While there are some LUs that could be mapped to more than one synset, as a general rule the restriction to a single one favors precision, which is desirable for the purpose of obtaining a clean knowledge base. The only cases where this model would be detrimental to precision are those for which LUs do not have any associated synset, but these are few and most can easily be avoided by omitting LUs with parts of speech not covered in WordNet, such as prepositions.

This choice allows for modeling the mapping as a function $S(l|a,b)$ from LUs to synsets as in Eq. (1). In this definition, S_l stands for the set of synsets with the same lexical label and part-of-speech tag as the LU l, μ_{L} and μ_{G} are the lexical and gloss (definition) overlap, respectively, f yields the corpus frequency of the synset, and a and b are parameters for a linear combination (the third parameter can be omitted because of the argmax function).

$$S(l|a,b) = \underset{s \in S_l}{\operatorname{argmax}}\ \mu_{\mathrm{L}}(l,s) + a \cdot \mu_{\mathrm{G}}(l,s) + b \cdot f(s) \quad (1)$$

The lexical overlap μ_{L} of a LU l and a synset s is the size of the intersection between the POS-annotated words from the LUs in the same frame as l and the POS-annotated words in s and its neighborhood. The neighborhood is defined as the synsets interconnected by a selection of lexical and semantic relations (called "semantic pointers" in WordNet) such as "See also", "Similar to", "Antonym", "Attribute" and "Derivationally related". This expansion is useful to reduce sparsity and to better match the sets with those generated for the LUs, which, due to the different semantics of frames and synsets, may already include these related words.

The gloss overlap μ_{G} is the size of the intersection between the set of words in the definition of the LU and the gloss of the synset. The Stanford CoreNLP library [61] is used to clean XML tags, tokenize, POS-label, and lemmatize the text, and all words except nouns and verbs are filtered out.

Parameters a and b are trained with a greedy search starting at several randomized seeds, obtaining optimal values $a = 5$, $b = 0.13$.

4.2. Hierarchy Construction

In FrameBase, frames are modeled as classes whose instances are specific events or situations. The frame elements of each frame are properties whose domain is that frame. The class hierarchy of frames is created as follows.

1. **General Frames:** These frames are obtained from the original FrameNet frames and are referred to in FrameBase as *macroframes*, because they correspond to broad general concepts.

They are connected to each other via the relations `:inheritsFrom` and `:isPerspectiveOf`[4], which are obtained from FrameNet's frame inheritance and perspectivization relations between frames. Both relations are made subproperties of `rdfs:subClassOf`, because every subframe or perspectivized frame is also an instance of the parent or general frame, and inheritance between frame element properties (belonging to frames connected by inheritance) is modeled with `rdfs:subPropertyOf`. Perspectivization is similar to but still somewhat different from inheritance. It is a sort of specialization relation that captures a particular perspective of a situation or event associated with a frame: for instance, in the resulting FrameBase schema, the frame `:frame-Transfer` is perspectivized by `:frame-Giving`, which reflects a perspective centered around the frame element *Donor* being the agent. This is further reflected by the fact that `:frame-Giving` inherits from `:frame-Intentionally_act` and the FE property `:fe-Transfer-Donor`, belonging to the former frame, is a subproperty of the FE property `:fe-Intentionally_act-Agent`, belonging to the latter frame. On the contrary, the frame element property `:fe-Transfer-Donor` from the frame `:frame-Transfer` does not inherit from any agentive frame element property. Additionally, a top frame is declared for the hierarchy. Semantic types are sometimes provided as ranges in FrameNet, but their current coverage is limited, and therefore have been left out of FrameBase.

Another example covering both inheritance and perspectivization is the following. Using RDFS inference, an instance of `:frame-Commerce_sell` with a certain property`:fe-Commerce_sell-Buyer` B, is also an instance of `:frame-Giving`, and B is its `:fe-Giving-Recipient`, because the former frame inherits from the latter. Likewise, it is also an instance of `:frame-Transfer` and B is the `:fe-Transfer-Recipient`, because `:frame-Giving` is a perspective on `:frame-Transfer`.

2. **Leaf Nodes:** Since FrameNet's original frame inventory is coarse-grained and different LUs like *construction* and *to glue* evoke the same frame, more specific frames associated with particular LUs are employed. In other words, every LU is treated as evoking its own separate fine-grained frame, an *LU-microframe*, which is made a subclass of the more coarse-grained original FrameNet frame. In addition, another type of microframes, denoted as *synset-microframes*, are created from the synsets in WordNet 3.0. The IRIs for the microframes are coined by appending the more specific identifier (LU or synset) to the IRI of the parent macroframe. For instance, macroframe class `frame-Personal_relationship` have, among others, two subclass microframes: `frame-Personal_relationship-partner.n` and `frame-Personal_relationship-wn_spouse_n_10640620`. The former is obtained from an LU in FrameNet, and the "n" suffix indicates that it is a noun concept. The latter is obtained from a synset in WordNet, including the number (synset ID).
3. **Intermediate Nodes:** As mentioned earlier in this section, macroframes are sometimes too general, while LU-microframes and synset-microframes are too fine-grained, sometimes leading to multiple aliases for near-identical concepts. This is addressed by providing a novel intermediate level composed of *cluster-microframes* that group together LU-microframes and synset-microframes that have equivalent or near-equivalent meanings.

   The clusters are generated in the following way. First, for each LU-microframe l, the corresponding set s_l of synsets equivalent to l is retrieved from the FrameNet–WordNet mapping. In the case of the mapping in Section 4.1, $|s_l| = 1$, but in general it could have more than one element. Then, s_l is expanded by adding all other synsets related by lexical relations reflecting cross-POS morphological transformations: "Derivationally related", "Derived from Adjective", "Participle" and "Pertainym". The lexical relation "Derivationally related" connects word senses that share the root (normally from different POS, e.g., the verb *visualize* and the noun *visualizer*, but can also have the same POS like *author* and *authorship*). "Pertainym" and "Derived from Adjective" are more specific and overlapping, connecting nouns and adverbs (respectively) with adjectives, but cover some cases not covered by "Derivation-

[4] We use `http://framebase.org/ns` as default prefix.

ally related" (e.g., *textile* as an adjective and as a noun). "Participle" connect verbs with their participle form (e.g., *stack* with *stacked*).
In general, these lexical relations do not necessarily imply any close semantics (e.g., the verb *create* and the noun *creature*), but when restricted to synsets all tied to the same FrameNet frame, such cases are normally factored out. Therefore, as a further step, s_l is restricted to those synsets that also belong to another set $s_{l'}$ produced from a sibling LU l' from the same macroframe. The goal of using the lexical relations is linking cross-POS LU-microframes that evoke the same specific situation with a different syntactic form, such as nominalizations (*produce–production*), non-finite verb forms (*produce–produced*), adjectivization, or adverbization. Next, the LU-microframe is connected with the synset-microframes from the set of synsets, using the property `framebase:isSimilarTo`, which is declared to be transitive and symmetric in OWL (although the sets of triples produced materialize the transitive and symmetric closure, so in practice this is not needed).
Figure 5 presents examples of clusters under a single macroframe.
After this process has concluded for all LU-microframes and the transitive closure of `framebase:isSimilarTo` has been materialized, each cluster of near-synonym microframes is represented by a clique of `framebase:isSimilarTo`. Finally, for each cluster, intermediate cluster-microframes are reified[5] and declared superframes of the members of the cluster, and at the same time subframes of their previously immediate superframe. The cluster-microframe is also connected by `framebase:isSimilarTo` to the subframes. An example of the result can be appreciated in Figure 4.
The use of the property `framebase:isSimilarTo` yields direct connections between members of the cluster. It may also be convenient in contexts when users wish to reduce sparsity by completely merging all members of each cluster. In this case, they can achieve this simply by declaring `framebase:isSimilarTo` a subproperty of `rdfs:subClassOf` and enabling RDFS inference. By virtue of the already materialized inverses of `framebase:isSimilarTo`, every instance of a member of the cluster, including the cluster-microframe, becomes an instance of the others. Alternatively, `owl:equivalentClass` can be used.

Names, definitions, and glosses in FrameNet and WordNet are also used to create text annotations for our schema. Lexical forms are attached with `rdfs:label` and definitions and glosses from FrameNet and WordNet are attached with `rdfs:comment`. Additional linguistically rich annotations are added using Lemon [37]. An example annotation is provided in Figure 6.

```
@prefix lemon: <http://lemon-model.net/lemon#> .
@prefix lexinfo: \
  <http://www.lexinfo.net/ontology/2.0/lexinfo#> .

:lexicon a lemon:Lexicon ;
  lemon:entry :frame--Self_motion--fly.v-le .

:frame--Self_motion--fly.v-le a lemon:LexicaEntry;
  lemon:canonicalForm [
    lemon:writtenRep "fly"@en ] ;
  lemon:sense [
    lemon:reference :frame-Self_motion-fly.v ] ;
  lemon:synBehavior [
     a lemon:Frame ;
     lexinfo:partOfSpeech lexinfo:verb ] .
```

Figure 6. Example of Lemon annotation for LU-microframe.

Following the best practices in the Linked Open Data community, we link synset-microframes to URIs in the canonical RDF translation of WordNet [36]. We also provide links to word-sense URIs in lexvo.org, a KB that connects information about languages, words, characters, and other human language-related entities [13]. This allows FrameBase to be transitively connected to other KBs in the Linked Open Data web, as well as provide multilingual support.

In general, the schema does depend on OWL inference, albeit of a lighter kind, consisting merely of RDFS inference plus support for `owl:TransitiveProperty`, `owl:SymmetricProperty` and `owl:equivalentClass`. However, the use of

[5] This is yet another different but related use of the term *reification*. In general, reification means the process of making something real, and in the context of knowledge bases, can be used whenever a new entity is created for something that was only implicitly represented before, generally as a function of pre-existing entities.

the transitive and symmetric closure (which is manageable for the size of the schema) and the inverted `rdf:subClassOf` properties makes it also possible to rely only on RDFS inference, which is more widely implemented and usually more efficient than OWL inference.

4.3. Automatic Reification–Dereification Mechanism

While frames are convenient for representational purposes, users wishing to query the knowledge base benefit from direct binary predicates between pairs of frame elements. For example, for a birth event, binary predicates like `bornInPlace` and `bornOnDate` can facilitate querying by offering a more compact and simple representation.

Thus, FrameBase presents a novel mechanism to convert between frame representations and direct binary predicates. This mechanism can also allow us to avoid materializing frame instances when only two frame elements are needed.

4.3.1. Structure of ReDer rules

The *dereification rules* have the form expressed in Figure 7. Additionally, for each dereification rule there is a converse reification rule so that one can go back from binary predicates to the frame representation. Each Direct Binary Predicate (DBP) has only one set of possible frame and frame elements associated, and therefore chaining reification and dereification rules is an idempotent operation. We call the pair of a reification rule and its converse dereification rule a ReDer (reification-dereification) rule. An example of a ReDer rule is provided in Figure 8.

```
?s  <DIRECT_BINARY_PREDICATE>  ?o
↕
<FRAME_INSTANCE>  a  <FRAME_CLASS> ,
<FRAME_INSTANCE>  <FRAME_ELEMENT-S>  ?s ,
<FRAME_INSTANCE>  <FRAME_ELEMENT-O>  ?o .
```

Figure 7. The general pattern of a ReDer rule. The conjunction of the three triples below is semantically equivalent to the triple above.

The ReDer rules can be implemented in different ways.

- As SPARQL CONSTRUCT queries, due to SPARQL's prominence as a standard query language for KBs. These can be used to materialize the DBPs into the KB.

```
?s  :dbp-Statement-writesAboutTopic  ?o
↕
?F  a  :frame-Statement-write.v ,
?F  :FE-Statement-Speaker  ?s ,
?F  :FE-Statement-Topic  ?o .
```

Figure 8. A particular example of a ReDer rule. The direct binary predicate `:dbp-Statement-writesAboutTopic` has the lexical label "writes about topic", and connects the values of `:FE-Statement-Speaker` and `:FE-Statement-Topic` when they are connected to a common frame of type `:frame-Statement-write.v`, which is associated with the verb "to write" when it evokes a "Statement" frame.

- As clauses, with triples as atoms, to be fed to general-purpose inference engines, with or without materialization. For example, ReDer rules have also been implemented as rules for the Rubrik reasoner in Jena [7].

Given an instance set (ABox), the reified and dereified layers can be stored using different strategies.

1. Materializing both the reified and dereified layers. This is the simplest but less space-efficient approach. Ensuring consistency between both layers after updates to a single one requires some bookkeeping.
2. Materializing the reified layer and virtualizing the dereified layer. This offers moderate space efficiency. Only dereification rules are used. Ensuring consistency after updates is trivial if only the materialized layer is updated.
3. Materializing frame instances with two FEs in the dereified layer and the rest in the reified layer. This offers the highest space efficiency. Ensuring consistency after updates is the most complex of the three cases, because knowledge has to be moved between the reified and dereified layers when triples with FE predicates are added or deleted.

This choice of the storage strategy is in theory orthogonal to the implementation of the ReDer rules. In practice, however, storage strategy 1 is relatively trivial to implement using SPARQL CONSTRUCT implementations of the ReDer rules, while storage strategy 2 is trivial to implement using dereification rules in Jena format. Storage strategy 3 would require internal logic (which has not been implemented so far), making the choice of the format a design choice.

Besides the plain `rdfs:label` and `rdfs:comment` annotations, we annotate the DBPs using Lemon [37]. This provides syntactically rich annotations that describe the internal structure and external syntactic frame of their labels. Instead of using the automatic generator, that uses automatic tokenization, parsing, etc., we use our knowledge of the synthetic structure of the different possible labels for DBPs to create annotations with human-level precision. Similarly, we also use Lemon for annotating microframes.

4.3.2. Creation of ReDer rules

The ReDer rules are automatically built using the annotations of English sentences given for different LUs in FrameNet, like the grammatical function (GFs) and phrase types (PTs) [52]. Each instance of an example sentence annotated by a frame is accompanied by the GF and PT associated with each of the FEs of that frame filled in that sentence.

FrameNet provides three kinds of GF labels that we shall use.

- External Argument (Ext). In the case of verb LUs, it represents the subject of the LU ("[The physician] *performed* the surgery" [52]), any constituent that controls the subject of the LU ("[The doctor] *tried* to cure me"), or a dependent of a governing noun ("We are glad for the [American] decision to *provide* relief"). In the case of adjective LUs, it is the subject of a copular verb ("[The chair] is *red*"), or other semantically similar constructions ("We consider [Pat] very *intelligent*"). In the case of noun LUs, the external argument can be interpreted as the subject of a semantically related verb in a periphrasis ("[He] made a *statement* to the press").
- Object (Obj). The syntactic object of a verb LU ("Voters *approved* [the stadium measure]").
- Dependent (Dep). This is the general grammatical function assigned to adverbs, Prepositional Phrases (PPs), and some other attached constituents, but in our case only PPs are used. In these cases, the PP annotation is attached (between square brackets) to the preposition forming the PP. It can be used for verb LUs ("*Give* the gun [to the officer]"; PP[to]), adjective LUs ("Lee is *certain* [of his innocence]" PP[to]) or noun LUs ("The *letter* was [to the President]" PP[to]).

Some of the PT labels that can be found are N (noun), NP (noun phrase), Obj (object) and PPinterrog (PP interrogative).

ReDer rules and new DBPs are created using ReDer rule constructors. Each constructor specifies certain conditions on the annotations associated with a pair of FEs in an example sentence. When the conditions are met, a new DBP is generated and a ReDer rule containing the pair of FEs is created.

The constructors are shown in Figures **??**. As in the general reification-dereification rule pattern in Figure 7, the postfixes "-S" and "-O" in the constructors indicate the data associated with the FEs that fill the first and second arguments of the DBP, respectively, or equivalently, the respective subject and object of the resulting RDF triple. The creation of the DBP implies the creation of a dereification rule following the pattern in Figure 7, with `<FRAME_CLASS>` defined by the LU, and `<FRAME_CLASS>` left as a free variable. The corresponding reification rule is built similarly, but assigning an anonymous node or a skolem constant to `<FRAME_CLASS>`.

The Agent-Verb-Patient constructor in Figure 9 creates DBPs whose lexical head are verbs, whose subject in the KB is an agent, and whose object is a patient, thus having a lexical representation in the form of a linguistic predicate in active voice. The constructor inverts example sentences that are deemed to be in passive form.

There is no explicit syntactic annotation in FrameNet to indicate if the verb LUs are evoked in passive form. Therefore, two different heuristics are used for detecting this. One (IsPassivePosHeuristic(LU)) draws on the POS annotations available in FrameNet, and decides that the target (LU) verb is in passive if and only if it appears as a past participle, and the verb *to be*, in any form, is in a prior position, without another verb in between. The other heuristic (IsPassiveDepHeuristic(LU)) uses the Stanford dependency parser [33], determining that the target (LU) verb is in passive if and only if it is the source of any of the dependencies NSUBJPASS, CSUBJPASS or AUXPASS. Both heuristics make type I and II mistakes differently, so the cases where they disagree were discarded, and in the ones where they agree that they there is passive form, the rules are created inverting the Ext and Obj GFs.

Agent-Verb-Patient constructor

Create ReDer rule with DBP whose name is:
"ConjugateThirdPersonSingular(LU)"
when an annotated sentence satisfies:

IsVerb(LU) and PT-O in {N, NP, Obj, PPinterrog, Sinterrog, QUO, Sfin, Sub, VPing} and
(
(GF-S==Ext and GF-O==Obj
and not IsPassivePosHeuristic(LU)
and not IsPassiveDepHeuristic(LU))
or
(GF-S==Obj and GF-O==Ext
and IsPassivePosHeuristic(LU)
and IsPassiveDepHeuristic(LU))
)

Examples of created ReDer rules:

```
?S :dbp-Forming_relationships-divorces ?O
↕
?R a :frame-Forming_relationships-divorce.v ,
?R :fe-Forming_relationships-Partner_1 ?S ,
?R :fe-Forming_relationships-Partner_2 ?O .

?S :dbp-Win_prize-wins ?O
↕
?R a :frame-Win_prize-win.v ,
?R :fe-Win_prize-Competitor ?S ,
?R :fe-Win_prize-Prize ?O .
```

Figure 9. Agent-Verb-Patient ReDer rule constructor and some examples of ReDer rules created.

Patient-Verb-Agent constructor

Create ReDer rule with DBP whose name is:
"is ConjugatePastParticiple(LU) by"
when an annotated sentence satisfies:

IsVerb(LU) and PT-O in {N, NP, Obj, PPinterrog, Sinterrog, QUO, Sfin, Sub, VPing} and
(
(GF-S==Obj and GF-O==Ext
and not IsPassivePosHeuristic(LU)
and not IsPassiveDepHeuristic(LU))
or
(GF-S==Ext and GF-O==Obj
and IsPassivePosHeuristic(LU)
and IsPassiveDepHeuristic(LU))
)

Examples of created ReDer rules:

```
?S :dbp-Filling-isLoadedBy ?O
↕
?R a :frame-Filling-load.v ,
?R :fe-Filling-Goal ?S ,
?R :fe-Filling-Agent ?O .

?S :dbp-Kidnapping-isKidnapedBy ?O
↕
?R a :frame-Kidnapping-kidnap.v ,
?R :fe-Kidnapping-Victim ?S ,
?R :fe-Kidnapping-Perpetrator ?O .
```

Figure 10. Patient-Verb-Agent ReDer rule constructor and some examples of ReDer rules created.

The Patient-Verb-Agent constructor in Figure 10 is the converse of the Agent-Verb-Patient constructor: it also creates DBPs whose lexical head are verbs, but whose subject in the KB is a patient, and whose object is an agent, thus having a lexical representation using the passive voice. Every time the Agent-Verb-Patient constructor is invoked on an example sentence and a pair of FEs, the Patient-Verb-Agent constructor is invoked as well, creating the converse DBP.

The Agent-Verb-Complement constructor in Figure 11 creates DBPs whose lexical heads are verbs, whose subjects in the KB are agents, and whose objects are complements that are contained in a PP in the example sentence. In the DBP label, a new PP is included using the name of the FE-O, following the convention used to name predicates in many LOD KBs (e.g., *diedOnDate*, *isWrittenByAuthor*, etc.). However, the proposition in the PP in the example sentence is not always the most appropriate to insert in the DBP label. Therefore, Algorithm 1 is used, where different options are tried in order, with more precise but narrow-scoped ones first.

The Patient-Verb-Complement constructor (Figure 12) changes agent with patient with respect to the constructor Agent-Verb-Complement, in the same way Patient-Verb-Agent does with respect to the constructor Agent-Verb-Patient. It creates verb-based DBPs whose subjects in the KB are patients instead of agents, and the DBP has a lexical representation using passive voice.

Using only agent and patient as subject of the triple prevents the constructors from forming DBPs that would rarely be useful, like those connecting the time and place, or the place and the cause.

The Agent-Verb-Noun constructor (Figure 13)

Agent-Verb-Complement constructor

Create ReDer rule with DBP whose name is:
"ConjugateThirdPersonSingular(LU) Prep FrameElement-O"
when an annotated sentence satisfies:

IsVerb(LU) and PT-O==PP[Prep] and (
(GF-S==Ext and GF-O==Dep
and not IsPassivePosHeuristic(LU)
and not IsPassiveDepHeuristic(LU))
or
(GF-S==Obj and GF-O==Dep
and IsPassivePosHeuristic(LU)
and IsPassiveDepHeuristic(LU))
)

Examples of created ReDer rules:

```
?S :dbp-Creating-createsFromComponents ?O
↕
?R a :frame-Creating-create.v ,
?R :fe-Creating-Creator ?S ,
?R :fe-Creating-Components ?O .
```

```
?S :dbp-Win_prize-winsAtVenue ?O
↕
?R a :frame-Win_prize-win.v ,
?R :fe-Win_prize-Competitor ?S ,
?R :fe-Win_prize-Venue ?O .
```

Figure 11. Agent-Verb-Complement ReDer rule constructor and some examples of ReDer rules created.

Patient-Verb-Complement constructor

Create ReDer rule with DBP whose name is:
"is ConjugatePastParticiple(LU) Prep FrameElement-O"
when an annotated sentence satisfies:

IsVerb(LU) and PT-O==PP[Prep] and (
(GF-S==Obj and GF-O==Dep
and not IsPassivePosHeuristic(LU)
and not IsPassiveDepHeuristic(LU))
or
(GF-S==Ext and GF-O==Dep
and IsPassivePosHeuristic(LU)
and IsPassiveDepHeuristic(LU))
)

Examples of created ReDer rules:

```
?S :dbp-Destroying-isDestroyedByMeans ?O
↕
?R a :frame-Destroying-destroy.v ,
?R :fe-Destroying-Undergoer ?S ,
?R :fe-Destroying-Means ?O .
```

```
?S :dbp-Beat_opponent-isDefeatedByWinner ?O
↕
?R a :frame-Beat_opponent-defeat.v ,
?R :fe-Beat_opponent-Loser ?S ,
?R :fe-Beat_opponent-Winner ?O .
```

Figure 12. Patient-Verb-Complement ReDer rule constructor and some examples of ReDer rules created.

and the Agent-Verb-Particle-Noun constructor (Figure 14) create ReDer rules with DBPs whose heads are nouns, based on noun LU-microframes. In these cases, a verb is needed that takes the noun as an argument, normally as a direct object. Across RDF vocabularies and ontologies, this verb is sometimes made implicit in human-readable IRIs. For example `skos:hasTopConcept` includes "has" explicitly, while `skos:topConceptOf` includes "is" implicitly. In FrameBase, the modeling choice has been to always make them explicit both in the IRI and in the lexical annotations, in order to avoid ambiguity and prevent incorrect use. The verbs have been conjugated in third person singular form.

The difference between these two constructors is that in the Agent-Verb-Noun constructor (Figure 13), the noun is part of the object of the verb, while in the Agent-Verb-Particle-Noun constructor (Figure 14) it is part of a PP with its own proposition.

In both cases, the verb governing the noun is obtained using the same method. For each noun LU in an annotation, the head verb is extracted by parsing the example annotated sentences with the Stanford dependency parser and searching the paths of dependencies indicated in the constructors Agent-Verb-Noun and Agent-Verb-Particle-Noun[6]. For brevity, the paths are annotated with the notation of SPARQL property paths, but this is not part of any query.

The Agent-Verb-Noun constructor contains several possible dependency paths using dependencies of type "dobj" (direct object), "cop" (copula), "nsubj" (nominal subject), and "prep" (preposition).

- (LU `^dobj` HeadVerb) matches HeadVerb="*make*" and LU="*comment*" for the

[6]We use collapsed CC-processed dependencies, version 3.2.0.

Algorithm 1 Algorithm used to select the preposition. $c(e, p, s)$ is 1 if frame element e is annotating a PP with preposition p in example sentence s, and 0 otherwise. $h(e)$ is an function that maps the 40 most common frame elements to a manually selected preposition.

Input:
s_0 ▷ Annotated sentence
e_0 ▷ FE-O in annotated sentence

Output:
p ▷ Proposition

$p' \leftarrow \arg\max_p \sum_{s \in S} c(e_0, p, s)$
if $\sum_{s \in S} c(e_0, p', s) / \sum_{s \in S, p \in P} c(e_0, p, s) \geq 0.5$ **then**
  return p'
end if
if $p \in \text{domain}(h)$ **then**
  return $h(e_0)$
end if
if $\max(\{c(e_0, p, s_0) | p \in P\}) = 1$ **then**
  return $\arg\max_p c(e_0, p, s_0)$
end if
if $\max(\{\sum_{s \in S} c(e_0, p, s) | p \in P\}) > 0$ **then**
  return p'
end if
return "with"

sentence "*I have decided not to make any further comment concerning the change of ball during the lunch interval at Lord 's on Sunday*".

- (LU `cop` HeadVerb) matches HeadVerb="*is*" and LU="*maiden name*" for the sentence "*The maiden name of one of his wives (probably the second) was Watt*".
- (LU `^nsubj/cop` HeadVerb) matches HeadVerb="*is*" and LU="*cause*" for the sentence "*The short-term cause of overriding local significance were the droughts and crop failures in 1920 and 1921*".
- (LU `^prep_*/cop` HeadVerb) matches HeadVerb="*is*" and LU="*cause*" for the sentence "'Well-meaning ignorance is one of the biggest causes of animal suffering in this country (...)'.
- (LU `^prep_*/^dobj` HeadVerb) matches HeadVerb="*give*" and LU="*thought*" for the sentence "*I have given a great deal of thought as to how much I should actually tell you about this period and what just to leave to your imagination*".

The Agent-Verb-Particle-Noun constructor fires in cases of phrasal verbs, where the head verb must be extracted with a particle.

- (LU `^prep_`VerbParticle HeadVerb) matches HeadVerb="*go*", VerbParticle="*on*" and LU="*tour*" for the sentence "*Something else I shall miss by going on this dratted tour with Gwen!*".

Agent-Verb-Noun constructor
Create ReDer rule with DBP whose name is: "ConjugateThirdPerson-Singular(HeadVerb) LU Prep Frame-Element-O" *when an annotated sentence satisfies:*
IsNoun(LU) and PT-O==PP[Prep] and GF-S==Ext and GF-O==Dep and (LU `^dobj` HeadVerb or LU `cop` HeadVerb or LU `^nsubj/cop` HeadVerb or LU `^prep_*/cop` HeadVerb or LU `^prep_*/^dobj` HeadVerb)

Examples of created ReDer rules:

```
?S :dbp-Coming_to_believe-\
makesInferenceFromEvidence ?O
```
↕
```
?R a :frame-Coming_to_believe-inference.n ,
?R :fe-Coming_to_believe-Cognizer ?S ,
?R :fe-Coming_to_believe-Evidence ?O .
```

```
?S :dbp-Arriving-makesEntranceByMeans ?O
```
↕
```
?R a :frame-Arriving-entrance.n ,
?R :fe-Arriving-Theme ?S ,
?R :fe-Arriving-Means ?O .
```

Figure 13. Agent-Verb-Noun ReDer rule constructor and some examples of created ReDer rules.

The Copula-Adjective-Complement constructor in Figure 15 creates adjective-based DBPs using the copular verb "to be".

With the rules obtained with the process above, the same DBP can be associated with different reified patterns (i.e., pairs of frame elements in a given LU-microframe), owing to different senses or syntactic frames for a given verb – for example the

Agent-Verb-Particle-Noun constructor

Create ReDer rule with DBP whose name is:
"ConjugateThirdPerson-Singular(HeadVerb) VerbParticle LU Prep Frame-Element-O"
when an annotated sentence satisfies:

IsNoun(LU) and PT-O==PP[Prep] and GF-S==Ext and GF-O==Dep and (
  LU ^prep_VerbParticle HeadVerb
)

Examples of created ReDer rules:

```
:dbp-Awareness-\
worksTowardsUnderstandingAboutTopic ?O
```
↕
```
?R a :frame-Awareness-understanding.n ,
?R :fe-Awareness-Cognizer ?S ,
?R :fe-Awareness-Topic ?O .
```

```
 :dbp-Discussion-\
 -goesIntoDiscussionWithInterlocutor2 ?O
```
↕
```
?R a :frame-Discussion-discussion.n ,
?R :fe-Discussion-Interlocutor_1 ?S ,
?R :fe-Discussion-Interlocutor_2 ?O .
```

Figure 14. Agent-Verb-Particle-Noun ReDer rule constructor and some examples of created ReDer rules.

transitive and intransitive frames for *smuggle*. This would conflate different senses, and if the reification and the dereification directions of the rules were chained, it would logically entail different pairs of frame elements, which would not be sound. Furthermore, a given reified pattern can also produce different DBPs, which would lead to redundancy. To achieve the idempotency mentioned earlier, a DBP should not be connected to more than one reified pattern (i.e. not present in more than one ReDer rule). To avoid redundancy, a reified pattern should not be connected to more than one DBP (ditto). Therefore, it is necessary to find an $\{0,1\}$-to-$\{0,1\}$ assignment between DBPs and reified patterns. To obtain the most correct and intuitive of such possible assignments, we optimize the number of example sentences on which the ReDer rules in the one-to-one assignment are based. This can be seen as an instance of the assignment problem. We build a bipartite graph with the set of DBPs and the set of reified patterns as the two sets of vertices, and with pairs of DBPs and reified patterns connected by edges weighted with the additive inverse of the

Copula-Adjective-Complement constructor

Create ReDer rule with DBP whose name is:
"is LU Prep FE-o"
when an annotated sentence satisfies:

IsAdjective(LU) and phrase-type-o==PP[Prep]
and grammatical-function-s==Ext
and grammatical-function-o==Dep

Example of created ReDer rule:

```
?s dbp-Sound_level-isLoudToDegree ?o
```
↕
```
f type frame-Sound_level-loud.a ,
f fe-Sound_level-Entity ?s ,
f fe-Sound_level-Degree ?o .
```

Figure 15. Copula-Adjective-Complement ReDer rule constructor and some examples of created ReDer rules.

number of annotated example sentences creating a ReDer rule that connects that DBP with that reified pattern (positive infinite is used as weight for the pairs that do not have any associated ReDer rule created from examples). The Kuhn-Munkres algorithm [42] algorithm can be applied over this graph to find a maximal subset of the ReDer rules that satisfies the $\{0,1\}$-to-$\{0,1\}$ condition between DBPs and reification patterns and maximizes the number of example sentences on which they are based. The Kuhn-Munkres algorithm is chosen for its polynomial (cubic) complexity. Although this could still be a problem for the total number of original ReDer rules, it is averted by creating an independent instance of the problem for the rules created from each frame. This does not change the results because ReDer rules are not created connecting DBPs and reified patterns from different frames.

5. Evaluation of the Schema

In this section, we evaluate the results of the methods used to create the FrameBase schema (Section 4) as well as some practical examples resulting from the integration of knowledge (Section 6).

First, Section 5.1 presents the evaluation of the FrameNet-WordNet mapping described in Section 4.1, and we discuss why this is used in the next steps.

Then, Sections 5.2 and 5.3 present the results for the methods described for the construction of the schema hierarchy (Section 4.2) and the construction of the ReDer rules (Section 4.3), respectively. These two sets of results (summarized in Table 2) cover all the parts of the schema that are created automatically, and since the original resources (FrameNet and WordNet) are created manually, these results provide a complete evaluation of the quality of the FrameBase schema with respect to the standard of human-level annotations.

	Correctness	**Nuanced correct.**
Cluster pairs	$87.55\% \pm 6.18\%$	$31.15\% \pm 9.38\%$
V-ReDer rules	$96.22\% \pm 3.22\%$	$80.43\% \pm 7.61\%$
N-ReDer rules	$87.50\% \pm 6.41\%$	$91.91\% \pm 6.28\%$

Table 2

Quality measures for the FrameBase schema for intra-cluster pairs of microframes, verb-based ReDer rules and noun-based ReDer rules. *Nuanced correctness* is a variable collected over correct elements, that reflects how perfectly accurate the element is (perfect synoynmy for pairs of microframes, readability for rules).

5.1. FrameNet–WordNet Alignment

To evaluate the created schema, the created FrameNet–WordNet mapping has been compared to the MapNet gold standard [60]. MapNet uses older versions of FrameNet and WordNet, so mappings from WordNet 1.6 to 3.0 [10] had to be applied, removing those with a confidence lower than one, and the few LUs of FrameNet 1.3 that are not contained in FrameNet 1.5 were discarded. Table 3 compares the results against state-of-the-art approaches and the scores that they report on the MapNet gold standard. As stated as goal when setting the cardinality restrictions in Section [**?**], the approach described in section 4 achieves higher precision (albeit for a very narrow margin), while still maintaining good recall. For this reason, we consider it more appropriate than the previously existing ones to be used in the following steps, because high precision is usually prioritized for tasks related to knowledge representation. 5-fold cross-validation was used for obtaining the results.

It may be relevant to note that there is in practice an upper bound to precision scores in tasks like this, because of the subjective component of any gold standard. The creators of the gold standard [60] report "0.90 as Cohen's Kappa computed over 192 LU-synset pairs for the same mapping task" by [12]. More generally, [18] maintains that "both people and automatic systems, when asked to assign tokens in a text to the appropriate senses in dictionaries, find the task difficult and do not agree among themselves".

5.2. Creation of the Hierarchy

The frame hierarchy in the FrameBase schema is based on FrameNet and WordNet and the mapping created between the two resources. It provides 19,376 frames, including 11,939 LU-microframes and 6,418 synset-microframes, all with lexical labels. A total of 18,357 microframes are clustered into 8,145 logical clusters, which are the sets of microframes whose elements are linked by a logical equivalence relation. The size of the schema is 250,407 triples.

The quality of the microframe clusters has been evaluated by asking two independent reviewers to evaluate a random sample of 100 intra-cluster pairs of LU-microframes. Each pair has been annotated with two variables: correctness (1 if the pair is correct, 0 otherwise) and similarity (only applying when the pair is correct; acquiring value 1 if they they are totally equivalent, 0 if there is a change of nuance). An resulting average correctness is $87.55\% \pm 6.18\%$ with a 95% Wilson confidence interval has been obtained. The evaluation showed a small change of nuance (similarity=0) for $31.15\% \pm 9.38\%$ of the correct pairs – most of these are caused by the choice to use semantic pointers such as "Similar to", which could be removed if very fine-grained distinctions of microframes were desired. The linear weighted Cohen's Kappa (inter-annotator agreement) over the three-valued combination of the two variables with which are annotated for each cluster pair, was 0.23 (over a maximum of 0.87).

5.3. Reification–Dereification Rules

Additionally, reification-dereification rules are provided, with the same number of direct binary predicates, with both human-readable IRIs and lexical labels. 83,790 are verb-based, 3,190 are noun-based and 7,248 are adjective-based. For evaluating them, the same methodology was used, with two independent human annotators. Two different

	Prec	Rec	F1	Acc
SVM Polynomial kernel 1 [60]	0.761	0.613	0.679	—
SVM Polynomial kernel 2 [60]	0.794	0.569	0.663	—
SSI-Dijkstra [34]	0.78	0.63	0.69	—
SSI-Dijkstra+ [34]	0.76	0.74	0.75	—
Neighborhoods [19]	—	—	—	0.772
FrameBase's mapping	0.789	0.709	0.746	0.864

Table 3
Comparison of FrameBase's FrameNet–WordNet mapping to state-of-the-art approaches in terms of precision, recall, F1, and accuracy.

variables were used for each rule: correctness and readability. A rule is considered to be not easily readable if the name of the direct binary predicate contains a frame element whose meaning is not obvious for a layman reader, or if it contains a preposition that is appropriate for some but not all possible objects, or it is not appropriate for the frame element in the name. The obtained average correctness for verb-based rules is $96.22\% \pm 3.22\%$, whereas $80.43\% \pm 7.61\%$ of the correct rules were found easily readable. For noun-based rules, the scores are $87.5\% \pm 6.41\%$ and $91.91\% \pm 6.28\%$. The Cohen's kappa for the two annotations was 0.39 over a maximum of 0.54.

6. Integration

Knowledge from other KBs such as Freebase can be integrated using *integration rules*. In practice, these result in a graph transformation from the source KB to FrameBase. Formally, these are rules whose antecedent and consequent are graph patterns sharing some variables. Whenever there is an instantiation of variables that, applied to the antecedent, returns a subset of the source KB, then the consequent, after being applied the same so instantiation of variables, can be added to the FrameBase instance data (the ABox in the jargon of description logics).

When the sources are in RDF, the most obvious choice for implementing integration rules is using SPARQL CONSTRUCT queries with the WHERE clause containing the antecedent and the CONSTRUCT clause containing the consequent. Additionally, SPARQL CONSTRUCT queries support predicates and logical operators that allow for imposing additional logical conditions on the WHERE clause to match the original KB (i.e., for the rule to be fired). For non-RDF sources, a simple choice would be applying an off-the-shelf RDF converter[7] to pre-process the source, after which SPARQL CONSTRUCT queries can still be used.

The SPARQL examples in this and the next sections use the following prefixes.

```
PREFIX :      <http://framebase.org/ns/>
PREFIX freeb: <http://rdf.freebase.com/ns/>
PREFIX dbr:   <http://dbpedia.org/resource/>
PREFIX sch:   <http://schema.org/>
```

In Section 6.1, some examples of manually built integration rules are presented for integrating events from two different sources: DBpedia and schema.org. Besides showing concrete examples of rules, the section provides an assessment for the expressiveness of the FrameBase schema in its current state, by reviewing to which extent external knowledge can be integrated when using manually built rules. It also introduces a basic typology of integration rules. These are important steps before reviewing the task of integrating knowledge automatically.

Subsequently, Section 6.2 discusses the creation of ReDer rules based on existing work.

Finally, in Section 6.3, we provide examples of queries that make use of the schema.

6.1. Manually Built Integration Rules

We will first show two simple examples of integration rules integrating knowledge from Freebase. They belong to two basic rule types that we label

[7] http://www.w3.org/wiki/ConverterToRdf

Class-Frame and *Property-Frame*, which will later serve as the basis for constructing more complex rules.

Class-Frame integration rules integrate a class from the source KB into a frame in FrameBase, and the outgoing properties from the external class into FE properties. The following example integrates a class `organization.leadership` into the frame `:frame-Leadership-leader.n`.

```
CONSTRUCT {
  _:f a :frame-Leadership-leader.n .
  _:f :fe-Leadership-Leader ?o1 .
  _:f :fe-Leadership-Governed ?o2 .
  _:f :fe-Leadership-Role ?o3 .
  _:f :fe-Leadership-Type ?o4 .
  _:timePeriod a :frame-Timespan-period.n .
  _:timePeriod :fe-Timespan-Start ?o5 .
  _:timePeriod :fe-Timespan-End ?o6 .
} WHERE {
  ?cvti a freeb:organization.leadership .
  OPTIONAL { ?cvti
    freeb:organization.leadership.person ?o1 .}
  OPTIONAL { ?cvti
    ...organization.leadership.organization ?o2 .}
  OPTIONAL { ?cvti
    freeb:organization.leadership.role ?o3 .}
  OPTIONAL { ?cvti
    freeb:organization.leadership.title ?o4 .}
  OPTIONAL { ?cvti
    freeb:organization.leadership.from ?o5 .}
  OPTIONAL { ?cvti
    freeb:organization.leadership.to ?o6 .}
}
```

Property-Frame integration rules translate a property from the source KB into a frame and two FEs in FrameBase. The structure is similar to that of ReDer rules, but the property in the antecedent is not a FrameBase DBP, although the similarity of the structure will be exploited later to automatically produce integration rules of this type from existing ReDer rules. The following example integrates a property `freeb:people.person.nationality` into the frame `frame-People_by_jurisdiction-citizen.n`.

```
CONSTRUCT {
  _:f a :frame-People_by_jurisdiction-citizen.n .
  _:f :fe-People_by_jurisdiction-Person ?person .
  _:f :fe-People_by_jurisdiction-Jurisdiction ?country .
} WHERE {
  ?person freeb:people.person.nationality ?country .
}
```

The next example pertains to the `Event` class in DBpedia. It is a Class-Frame rule with extensions. From the nine properties of the `Event` class, `numberOfPeopleAttending` was omitted because the `Event` class is too general for it, as it has subclasses such as `PersonalEvent` (`Birth`, etc.) and `SocietalEvent`, that appear more appropriate for this. The remaining eight properties were integrated, but although the example shares the same basic structure as the Class–Frame rule provided for Freebase, it includes additional complex patterns in the consequent.

```
CONSTRUCT {
  ?f a :frame-Event-event.n .
  #
  ?f :fe-Event-Time _:timePeriod .
    _:timePeriod a :frame-Timespan-period.n ;
      fbe:fe-Timespan-Start ?o1 ;
      fbe:fe-Timespan-End ?o2 .
  #
  _:af2 a :frame-Relative_time-preceding.a ;
    :fe-Relative_time-Landmark_occasion ?f ;
    :fe-Relative_time-Focal_occasion ?o3 .
  #
  _:af3 a :frame-Relative_time-following.a ;
    :fe-Relative_time-Landmark_occasion ?o3 ;
    :fe-Relative_time-Focal_occasion ?f .
  #
  _:af4 a :frame-Relative_time-following.a ;
    :fe-Relative_time-Landmark_occasion ?f ;
    :fe-Relative_time-Focal_occasion ?o4 .
  #
  _:af5 a :frame-Relative_time-preceding.a ;
    :fe-Relative_time-Landmark_occasion ?o4 ;
    :fe-Relative_time-Focal_occasion ?f .
  #
  _:af6 a :frame-Relative_time-following.a ;
    :fe-Relative_time-Landmark_occasion ?f ;
    :fe-Relative_time-Focal_occasion ?o5 .
  #
  _:af7 a :frame-Relative_time-preceding.a ;
    :fe-Relative_time-Landmark_occasion ?o5 ;
    :fe-Relative_time-Focal_occasion ?f .
  #
  ?f :fe-Event-Reason ?o6 .
  #
  _:af8 a :frame-Dimension-length.n ;
    :fe-Dimension-Object ?f ;
    :fe-Dimension-Measurement ?o7 .
  #
  ?f a :frame-Social_event-meeting.n ;
    :fe-Social_event-Attendee ?o9 ;
    :fe-Social_event-Duration ?o7 .
  #
} WHERE {
  ?f a dbr:Event .
  OPTIONAL{?f dbr:startDate ?o1}
  OPTIONAL{?f dbr:endDate ?o2}
```

```
  OPTIONAL{?f dbr:previousEvent ?o3}
  OPTIONAL{?f dbr:followingEvent ?o4}
  OPTIONAL{?f dbr:nextEvent ?o5}
  OPTIONAL{?f dbr:causedBy ?o6}
  OPTIONAL{?f dbr:duration ?o7}
  OPTIONAL{ #Omitted
    ?f dbr:numberOfPeopleAttending ?o8}
  OPTIONAL{?f dbr:participant ?o9}
}
```

The `dbr:Event` class has several subclasses that can also be translated. However, the hierarchy in the original ontology is not necessarily consistent with the hierarchy in FrameBase. Only in certain cases does a subsumption relationship between two entities of the source also exist between the two entities' respective translations to FrameBase. Therefore, for each translation of an element in the source KB, the translations of more general elements can be added, and this provides additional knowledge that would not always be inferred by the FrameBase schema alone.

For example, using RDFS inference, the substitutions for `?f` that fire the rule below ("`?f a dbr:SocietalEvent`"), do also fire the one for `dbr:Event`, because `dbr:SocietalEvent` is a subclass of `dbr:Event`. This rule is very short because in DBpedia, all of the outgoing properties belong to the parent `Event` class itself.

```
CONSTRUCT {
  ?f a :frame-Social_event-meeting.n .
} WHERE {
  ?f a dbr:SocietalEvent
}
```

Similarly, the substitutions for `?f` that fire the following five examples from DBpedia (`dbr:SpaceMission`, `dbr:Convention`, `dbr:Election`, `dbr:FilmFestival`, `dbr:MilitaryConflict`), do also fire the ones for `dbr:SocietalEvent` and `dbr:Event`, because the classes captured in the antecedent are subclasses of `dbr:SocietalEvent`.

In the rule for `dbr:SpaceMission`, we minimize the need for declaring new frames and frame elements for specialized domains by making use of the compositionality of most specialized terms, creating complex structures that combine the semantics of simpler, basic elements. For instance, the translation for the type `dbr:SpaceMission` declares a frame of type `Project-project.n`, and specifies that it is about space exploration by assigning `dbrl:SpaceMission` as the value for the `Project-Activity` FE.

```
CONSTRUCT {
  ?f a :frame-Project-project.n .
  ?f :fe-Project-Activity dbr:Space_exploration .
} WHERE {
  ?f a dbr:SpaceMission
}
```

```
CONSTRUCT {
  ?f a fbe:frame-Social_event-convention.n .
} WHERE {
  ?f a dbr:Convention
}
```

```
CONSTRUCT {
  ?f a :frame-Change_of_leadership-election.n .
} WHERE {
  ?f a dbr:Election .
}
```

```
CONSTRUCT {
  ?f a :frame-Social_event-festival.n .
  ?f :fe-Social_event-Attendee ?o3 .
  ?f :fe-Social_event-Descriptor dbr:Film .
  ?f a :frame-Competition-competition.n .
  ?f :fe-Competition-Participant_1 ?o3 .
  ?f :fe-Competition-Competition dbr:Film .
  _:af1 a :frame-Ordinal_numbers-first.a .
  _:af1 :fe-Ordinal_numbers-Item ?o1 .
  _:af1 :fe-Ordinal_numbers-Comparison_set ?f .
  _:af1 :fe-Ordinal_numbers-Comparison_set dbr:Film .
  _:af2 a :frame-Ordinal_numbers-last.a .
  _:af2 :fe-Ordinal_numbers-Item ?o2 .
  _:af2 :fe-Ordinal_numbers-Comparison_set ?f .
  _:af2 :fe-Ordinal_numbers-Comparison_set dbr:Film .
} WHERE {
  ?f a dbr:FilmFestival .
  OPTIONAL{?f dbr:closingFilm ?o1}
  OPTIONAL{?f dbr:openingFilm ?o2}
  OPTIONAL{?f dbr:film ?o3}
}
```

```
CONSTRUCT {
  ?f a :frame-Hostile_encounter-hostility.n .
  _:af1 a :frame-Death-die.v .
  _:af1 :fe-Death-Sub_event ?f .
  _:af1 :fe-Death-Protagonist ?o1 .
  ?f :fe-Hostile_encounter-Side_1 ?o2 .
  _:af3 a :frame-Part_whole-part.n .
  _:af3 :fe-Part_whole-Part ?f .
  _:af3 :fe-Part_whole-Whole ?o3 .
  ?f :fe-Hostile_encounter-Place ?o4 .
  ?f :fe-Hostile_encounter-Result ?o5 .
  ?f :fe-Hostile_encounter-Depictive ?o6 .
```

```
  ?f :fe-Hostile_encounter-Side_2 ?o7 .
} WHERE {
  ?f a dbr:MilitaryConflict .
  OPTIONAL{?f dbr:casualties ?o1}
  OPTIONAL{?f dbr:combatant ?o2}
  OPTIONAL{?f dbr:isPartOfMilitaryConflict ?o3}
  OPTIONAL{?f dbr:place ?o4}
  OPTIONAL{?f dbr:result ?o5}
  OPTIONAL{?f dbr:strength ?o6}
  OPTIONAL{?f dbr:opponents ?o7}
}
```

Below, we also present the translation of the class `Event` in schema.org.

Due to space restrictions, we omit the subclasses here, but these have very few genuine properties, and therefore the specialization is relatively simple. Besides, the taxonomy of schema.org events has some inconsistency issues that makes its use complex: the Event class is defined as capturing events such as concerts, lectures, and festivals, with properties such as "typical age range", but there are sub-events such as `UserInteraction` and `UserPlusOnes` that actually represent a more general kind of events.

```
CONSTRUCT {
  ?f a :frame-Social_event-meeting.n .
  ?f a :frame-Event-event.n .
  #
  ?f :fe-Social_event-Time _:timePeriod .
    _:timePeriod a fbe:frame-Timespan-period.n ;
      fbe:fe-Timespan-Start ?Osta ;
      fbe:fe-Timespan-End ?Oend .
  ?f :fe-Event-Time _:timePeriod .
  #
  ?f :fe-Social_event-Duration ?Odur .
  ?f :fe-Event-Duration ?Odur .
  #
  ?f :fe-Social_event-Place ?Oloc .
  ?f :fe-Event-Place ?Oloc .
  #
  ?f :fe-Social_event-Attendee ?Oatt .
  ?f :fe-Social_event-Host ?Oorg .
  #
  ?f :fe-Social_event-Occasion ?Osup .
  ?Osub :fe-Social_event-Occasion ?f .
  #
  ?Ooff a :frame-Offering-offer.v ;
    :fe-Offering-Theme ?f .
  #
  ?f a :frame-Performing_arts-performance.n ;
    :fe-Performing_arts-Performer ?Oper ;
    :fe-Performing_arts-Performance ?Owor .
  #
  _:af1 a :frame-Recording-record.v ;
    :fe-Recording-Phenomenon ?f ;
    :fe-Recording-Medium ?Orec .
  #
  ?f :fe-Social_event-Descriptor ?Oeve .
  #
  _:af2 a Change_event_time-postpone.v ;
    Change_event_time-Event ?f;
    Change_event_time-Landmark_time ?Opre.
  #
  _:af a :frame-Typicality-normal.a .
  _:af :fe-Typicality-Entity _:af2 .
  _:af2 :frame-Age-age.n .
  _:af2 :fe-Age-Age ?Otyp .
} WHERE {
  ?f a sch:Event .
  OPTIONAL{?f sch:startDate ?Osta}
  OPTIONAL{?f sch:endDate ?Oend}
  OPTIONAL{?f sch:duration ?Odur}
  OPTIONAL{?f sch:location ?Oloc}
  OPTIONAL{?f sch:attendee ?Oatt}
  OPTIONAL{?f sch:organizer ?Oorg}
  OPTIONAL{?f sch:superEvent ?Osup}
  OPTIONAL{?f sch:subEvent ?Osub}
  OPTIONAL{?f sch:offers ?Ooff}
  OPTIONAL{?f sch:performer ?Oper}
  OPTIONAL{?f sch:workPerformed ?Owor}
  OPTIONAL{?f sch:recordedIn ?Orec}
  OPTIONAL{?f sch:eventStatus ?Oeve}
  OPTIONAL{?f sch:previousStartDate ?Opre}
  OPTIONAL{?f sch:typicalAgeRange ?Otyp}
  # No translation
  OPTIONAL{?f sch:doorTime ?Odoo}
}
```

The only extension of the FrameBase schema used for these examples was the frame `:frame-Timespan-period.n` with the start and end frame elements, used to denote periods of time. This, however, is not an ad-hoc extension motivated by a particular need of only one source, but a very general one. Of the 16 properties of the `Event` class, only one (`sch:doorTime`, with an official gloss "The time admission will commence"), was not integrated. The remaining 15 were integrated.

Property-Frame rules, as well as complex Class-Frame rules that declare new frame instances that do not correspond to entities existing in the source KB (either by means of anonymous nodes, like in the examples, or coining new, essentially skolemized IRIs) do not merge frame instances that correspond to the same n-ary relation but are created by different rules or different instances of the same rule. This is a later step for which an out-of-the-box entity de-duplicator [35, 38, 57] can be applied. What the integration rules allow is to provide entities of the same type and that actually

represent the same thing (event, situation, process, i.e. frame), so that the entities can actually be linked – and if the de-duplication process has high enough precision, it is recommended that they are merged, so the full efficiency indicated in Table 1 is achieved. This would not be possible with the heterogeneous models in Figure 1.

6.2. Automatically Built Integration Rules

There has been recent work on automatically creating basic Property-Frame and Class-Frame integration rules using automatic methods guided with KB specific heuristics, and these have been tested for Freebase and Yago [48]. Table 4 shows the number of triples and frame types integrated in FrameBase under this method.

Table 4

Number of statements and distinct frame types in the integrated data, from YAGO2s and from Freebase. The numbers in parentheses include the equivalent microframes that can be obtained with RDFS inference.

	YAGO2s	Freebase
Reified		
Number of triples	32,927,963	7,483,430
Instantiated frame types	186 (1634)	29 (130)
Dereified		
Number of triples	3,933,207	6,120,201

More specific work has been performed on the creation of Property-Frames by matching canonicalized properties from external KBs with FrameBase DBPs, and substituting the DBP in the ReDer rule with the external property that creates a good match [50]. This is similar to how Legalo [44] works for extracting relations from hyperlinks surrounded by text. In general, the ability to create Property-Frame integration rules towards FrameBase exploiting its linguistic nature and its corpus of annotations is one of the main values of this work. First, because traditional ontology alignment systems cannot produce such complex mappings, as was discussed in Section 2, and therefore their recall will be effectively equal to zero in this task. Second, because the same ontology alignment systems can be re-used to create Class-Frame rules (mapping classes with classes and properties with properties, if the ontology alignment system allows declaring constraints related to the properties' domain). The creation of complex Property-Frame rules, which will be discussed in Section 7, is also explored in this work.

Additionally, a demo system has been developed that allows us to re-use these methods as search and suggestion engines behind a intuitive GUI, enabling human-level accuracy while minimizing the effort for the user [51].

6.3. Querying

FrameBase facilitates novel forms of queries. The query in Figure 16, for instance, uses reified patterns to find the heads of the World Bank.

```
SELECT DISTINCT
?leader ?leaderLabel ?role ?roleLabel
WHERE {
  ?lumfi a :frame-Leadership-leader.n .
  ?lumfi :fe-Leadership-Governed ?worldBank.
  ?lumfi :fe-Leadership-Leader ?leader .
  ?leader rdfs:label ?leaderLabel .
  VALUES ?worldBank {
    yago:World_Bank freeb:m.02vk52z
  }
  OPTIONAL{
    ?lumfi :fe-Leadership-Role ?role .
    ?role rdfs:label roleLabel .
  }
}
```

Figure 16. Example query using reified pattern.

The results in Table 5 show example instances integrated into the FrameBase schema from both Freebase (rows 1–3, extracted from the second example integration rule above) and YAGO2s (rows 4–5, extracted with a similar integration rule made for YAGO2s) [48].

Alternatively, if the triple `:isSimilarTo rdfs:subPropertyOf rdfs:subClassOf` is added with RDFS inference activated, then the microframe `:frame-Leadership-leader.n` in Figure 16 can be substituted with any of the microframes in the cluster, shown in Figure 17. This effectively helps increasing recall.

A DBP from the dereification rules can also be used to obtain the same non-optional results, as illustrated in the query in Figure 18. Either of the verb-based DBPs *leads* or *heads* can be used because the LU-microframes for these verbs are in the same cluster as the nouns *leader* and *head*, and there is a dereification rule between the *Leader* and *Governed* FEs for both.

?leader	?leaderLabel	?role	?roleLabel
freeb:m.0h_ds2s	'Caroline Anstey'	freeb:m.04t64n	'Managing Director'
freeb:m.0d_dq5	'Mahmoud Mohieldin'	freeb:m.04t64n	'Managing Director'
freeb:m.047cdkk	'Sri Mulyani Indrawati'	freeb:m.01yc02	'Chief Operating Officer'
yago:Jim_Yong_Kim	'Ji, Yong Kim'	-	-
yago:Robert_Zoellick	'Rober Zoellick'	-	-

Table 5
Results from the query

```
:frame-Leadership-wn_lead_n_01256743
:frame-Leadership-lead.v
:frame-Leadership-wn_head_v_02729023
:frame-Leadership-chief.n
:frame-Leadership-wn_head_n_10162991
:frame-Leadership-principal.n
:frame-Leadership-leadership.n
:frame-Leadership-wn_headship_n_00593108
:frame-Leadership-wn_leader_n_09623038
:frame-Leadership-wn_leadership_n_05617310
:frame-Leadership-head.v
:frame-Leadership-leader.n
:frame-Leadership-wn_head_v_02440244
:frame-Leadership-head.n
```

Figure 17. Microframes from the cluster where `:frame-Leadership-leader.n` belongs.

```
SELECT DISTINCT ?leader WHERE {
  ?leader :dbp-Leadership-heads ?worldBank .
  VALUES ?worldBank {
    yago:World_Bank freeb:m.02vk52z
  }
}
```

Figure 18. Example query using a dereified pattern.

7. Towards Complex Integration

The methods described in Section 6.2 create basic Class-Frame and Property-Frame integration rules. However, as some examples in Section 6.1 illustrate, integration rules can become very complex.

7.1. Complex Property-Frame Integration Rules

- **FrameBase driven.** This involves extending an approach already explored in existing FrameBase integration work [50], creating very complex ReDer rules whose DBPs could also be matched with external properties. These DBPs could have for instance a "(VP <VBZ> (NP <NP_1> (PP <IN> <NP_2>)))" structure, as e.g. "developsUnderstandingOfContent" (see Figure 19) or "startsDemolitionOfBuilding", but other more complex structures would be possible, too. This involves two frame instances (one evoked by VBZ and the other by NP_1), and some challenges:
  - Syntactically correct but semantically nonsensical combinations should be filtered out (e.g. "procrastinationDrunkByQuadruplicity"). This could be done based on example sentences in FrameNet.
  - If the frames evoked by the VBZ and NP_1 are not annotated in the same sentence, the correct pair of frames should be chosen from the pair of lexical units (VBZ,NP_1), and the correct FE connecting both should be chosen too.

  An advantage of this approach is that it provides richer ReDer rules in FrameBase, but the disadvantage is that being driven by FrameBase, it may have poor recall for real-life datasets, both because of its reliance on FrameNet example sentences and FrameNet's non-specialized vocabulary. The latter problem could be significantly reduced by also updating the similarity function between DBPs and source properties, to account for hypernymy and synonymy relations that would allow capturing very specific concepts in source KBs for which hypernyms can be found in FrameBase (for instance: "increasesSpeedOfProcess" and "catalyzesChemicalReaction").
- **Source-data driven.** This would involve parsing predicate names with a semantic role labelling system, similar to Legalo [44]. However, such SRL systems are also constrained by their reliance on annotated example sentences for training. In any case, an advantage of this approach is that if FrameBase is extended with PropBank, SRL systems for this could be used as well.

```
?s dbp-Awareness-developsUnderstandingOfContent ?o
↕
f type frame-Progress-develop.v
f fe-Progress-Entity ?s
f fe-Progress-Post_state f'
f' type frame-Awareness-understanding.n
f' fe-Awareness-Cognizer ?s
f' fe-Awareness-Content ?o
```

Figure 19. Example of a very complex noun-based ReDer rule

7.2. Complex Class-Frame Integration Rules

There are multiple ways in which Class-Frame rules can differ from their basic pattern. We will use the examples in Section 6.1 to illustrate this.

1. Sometimes, a class integration rule may need to instantiate multiple frames rather than just a single one. We distinguish two main types of this phenomenon.
   a) The instantiated frame instances may be connected by FEs. Examples of this include the frame `:frame-Timespan-period.n`, created to represent time periods, and the subframes of `Relative_time` to express precedence between events (all in the example for `dbr:Event`). The same applies when a FE is used to specify a frame beyond the lexical unit (see the rule for `dbr:Space_exploration`).
   b) Several frames can also be evoked separately, without the instances being directly connected by any FE. When these frames describe different perspectives of the same event, there is the possibility that FrameNet links them by means of *perspectivization*, and therefore FrameBase can infer one from another. In this case, inference is possible because RDFS subclass and subproperty properties are used in FrameBase to reflect the perspectivization relation between frame classes and FEs respectively. Another example are `:frame-Receive_visitor_scenario` and `:frame-Visit_host`, which are perspectives of `:frame-Visitor_and_host`. However, in other cases one cannot rely on existing inference. For instance, on can observe that the rule to translate `Event` from schema.org, besides the frames `Event-event.n` and `Timespan-period.n`, also instantiates `Performing_arts-performance.n`, `Recording-record.v`, and `Offering-offer.v`, when certain properties are present.
2. Another possible source of complexity is that FEs can be inverted. In this case, the integration rules need to invert the order of the arguments, as in the second occurrence of `:fe-Social_event-Occasion` in the integration rule for the class `Event` in schema.org.

Arbitrary combinations of these phenomena are possible (e.g., the rule integrating the Event class from schema.org).

A possible way to address this problem may be defining a reduced alphabet of transformations over a basic Class-Frame rule, similar to our list above, so that a complex Class-Frame rule can be represented as a basic initial one followed by a sequence of transformations, and this representation can be fed into a supervised learning algorithm.

However, the high number of variables involved would mean that any attempt to train a system would face extreme sparsity. Inter-annotator agreement, which is already low for simple integration rules [48], would probably be even lower. Investigating how to produce such genuinely complex rules entirely automatically thus remains an important research challenge.

In the short term, we believe that a combination of automated assistance and user feedback, as provided by user interfaces such as Klint [51], may be the best strategy whenever high-quality integration is desired.

8. Conclusion

FrameBase is a novel approach for integrating knowledge from different heterogeneous sources and connecting it to decades of work from the NLP community. It provides a flexible and homogeneous model to describe n-ary relations, which combines efficiency, expressiveness and is based on a linguistically sound foundation. The ties with natural language can be exploited to automatically integrate knowledge from external sources. FrameBase

opens up several new research directions, which we enumerate next.

Integrating additional sources. Either using a unified approach [48] or focusing on Property-Frame rules and combining them with existing ontology alignment systems [50], additional sources could be integrated. Both generic KBs such as WikiData and domain-specific ones such as from the biomedical domain could be incorporated.

Interfacing from natural language. Due to its use of linguistic resources for ontological purposes, FrameBase has significant potential for text mining and other natural language related tasks. Both pure Semantic Role Labelling (SRL) systems for FrameNet such as SEMAFOR [9] as well as text-to-ontology systems such as FRED [43] and Pikes [8] could be adapted to produce FrameBase data from natural language text. Similar methods could also enable question answering. For example, for the example in Figure 16 in Section 6.3, given the question "Who has been the head of the World_Bank?", the SRL tool SEMAFOR [9] successfully extracts the frame *Leadership* with lexical unit *head.noun* and frame elements *Governed* and *Leader*. Based on this, and after a named entity disambiguator such as AIDA [29] matches World_Bank to the entities in the KBs, a structured query can easily be built. Although accurate semantic role labeling is still very challenging, semantics has become one of the largest research areas in natural language processing and thus FrameBase can easily benefit from progress made in this area in the future.

Natural language generation. FrameBase also offers opportunities for natural language generation from KBs. Dereification rules can be interpreted as syntactic templates [62] for simple English sentences without subordinate clauses. For instance, `:dbp-Statement-writesAboutTopic` from Figure 8 could be used to produce a natural language representation "X writes about Y". It would be trivial to produce similar 3-ary syntactic templates for "X writes about Y in Z" (for Z being a Time or a Date) and "X writes to Z about Y" (for Z being the Addressee).

Implementing virtual querying. Currently, the integration rules for integrating source KBs into FrameBase have been implemented as SPARQL CONSTRUCT queries applied over the sources' data, which can be used to materialize the integrated knowledge. An alternative implementation would involve virtual querying: using the integration rules to provide FrameBase-adapted virtual views of the source KBs. This would allow re-using existing SPARQL endpoints from the different sources and enable access to the most recent version of the source data.

Further information. Details and more information about FrameBase are available at `http://framebase.org`. The FrameBase data is freely available under a Creative Commons Attribution 4.0 International license (CC-BY 4.0).

Acknowledgments

This research was partially funded by the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement No. FP7-SEC-2012-312651 (ePOOLICE project). Additional funding was provided by China 973 Program Grants 2011CBA00300, 2011CBA00301, and NSFC Grants 61033001, 61361136003, 20141330245.

References

[1] Roles in Schema.org. Technical report, W3C, 2014. http://schema.org/Role.

[2] Roles in W3c WebSchemas. Technical report, 2014. http://blog.schema.org/2014/06/introducing-role.html.

[3] C. F. Baker, C. J. Fillmore, and J. B. Lowe. The Berkeley FrameNet Project. ICCL '98, pages 86–90, 1998.

[4] C. Bizer, T. Heath, and T. Berners-Lee. Linked data–the story so far. *IJSWIS*, 5(3):1–22, 2009.

[5] C. Böhm, G. de Melo, F. Naumann, and G. Weikum. LINDA: Distributed Web-of-data-scale Entity Matching. CIKM '12, pages 2104–2108, 2012.

[6] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. SIGDATA, pages 1247–1250, 2008.

[7] J. J. Carroll, I. Dickinson, C. Dollin, D. Reynolds, A. Seaborne, and K. Wilkinson. Jena: Implementing the semantic web recommendations. WWW '04, 2004.

[8] F. Corcoglioniti, M. Rospocher, and A. P. Aprosio. A 2-phase frame-based knowledge extraction framework. In *Proc. of ACM Symposium on Applied Computing (SAC'16)*, pages 354–361, 2016.

[9] D. Das, D. Chen, A. F. T. Martins, N. Schneider, and N. A. Smith. Frame-semantic parsing. *Computational Linguistics*, 40(1):9–56, Mar. 2014.

[10] J. Daudé, L. Padró, and G. Rigau. Mapping wordnets using structural information. ACL, 2000.

[11] J. David, J. Euzenat, F. Scharffe, and C. Trojahn dos Santos. The Alignment API 4.0. *Semantic Web Journal*, 2(1):3–10, 2011.

[12] D. De Cao, D. Croce, and R. Basili. Extensive Evaluation of a FrameNet-WordNet mapping resource. LREC, 2010.

[13] G. de Melo and G. Weikum. Language as a foundation of the Semantic Web. In C. Bizer and A. Joshi, editors, *Proceedings of the Poster and Demonstration Session at the 7th International Semantic Web Conference (ISWC 2008)*, volume 401 of *CEUR WS*, Karlsruhe, Germany, 2008. CEUR.

[14] L. Del Corro and R. Gemulla. Clausie: Clause-based open information extraction. WWW '13, 2013.

[15] R. Dhamankar, Y. Lee, A. Doan, A. Halevy, and P. Domingos. iMAP: Discovering Complex Semantic Matches Between Database Schemas. In *SIGMOD 2004*, pages 383–394, 2004.

[16] J. Euzenat and P. Shvaiko. *Ontology Matching*. Springer, 2007.

[17] C. Fellbaum, editor. *WordNet: An Electronic Lexical Database*. The MIT Press, 1998.

[18] C. Fellbaum and C. F. Baker. Can WordNet and FrameNet be Made "Interoperable"? ICGL '08, 2008.

[19] O. Ferrández, M. Ellsworth, R. Munoz, and C. F. Baker. Aligning FrameNet and WordNet based on Semantic Neighborhoods. LREC '10, 2010.

[20] D. A. Ferrucci, E. W. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A. Kalyanpur, A. Lally, J. W. Murdock, E. Nyberg, J. M. Prager, N. Schlaefer, and C. A. Welty. Building Watson: An Overview of the DeepQA Project. *AI Magazine*, 31(3), 2010.

[21] C. J. Fillmore, C. R. Johnson, and M. R. Petruck. Background to Framenet. *International journal of lexicography*, 16(3):235–250, 2003.

[22] W. Frawley. *Linguistic semantics*. Hillsdale, 1992.

[23] A. Gangemi and V. Presutti. Towards a pattern science for the semantic web. *Semantic Web*, 1(1):61–68, 2010.

[24] A. Gangemi and V. Presutti. A Multi-dimensional Comparison of Ontology Design Patterns for Representing n-ary Relations. In *SOFSEM '13*, pages 86–105, 2013.

[25] D. Gildea and D. Jurafsky. Automatic labeling of semantic roles. *Computational linguistics*, 28(3):245–288, 2002.

[26] F. Giunchiglia, P. Shvaiko, and M. Yatskevich. S-Match: an Algorithm and an Implementation of Semantic Matching. In *The Semantic Web: Research and Applications*, pages 61–75. Springer, 2004.

[27] P. Hayes and P. Patel-Schneider. RDF 1.1 semantics. Technical report, W3C, 2014. http://www.w3.org/TR/rdf11-mt/.

[28] J. Hoffart, F. M. Suchanek, K. Berberich, and G. Weikum. YAGO2: A spatially and temporally enhanced knowledge base from Wikipedia. *Artificial Intelligence* , 194(0):28–61, 2013.

[29] J. Hoffart, M. A. Yosef, I. Bordino, H. Fürstenau, M. Pinkal, M. Spaniol, B. Taneva, S. Thater, and G. Weikum. Robust Disambiguation of Named Entities in Text. EMNLP '11, pages 782–792, 2011.

[30] D. Jurafsky and J. H. Martin. *Speech and Language Processing*. Pearson Prentice Hall, 2nd edition, 2009.

[31] A. Kalyanpur, B. K. Boguraev, S. Patwardhan, J. W. Murdock, A. Lally, C. Welty, J. M. Prager, B. Coppola, A. Fokoue-Nkoutche, L. Zhang, et al. Structured data and inference in deepqa. *IBM Journal of Research and Development*, 56(3.4):10–1, 2012.

[32] P. Kingsbury and M. Palmer. From TreeBank to PropBank. LREC '02, 2002.

[33] D. Klein and C. D. Manning. Accurate unlexicalized parsing. ACL '03, pages 423–430, 2003.

[34] E. Laparra, G. Rigau, and M. Cuadros. Exploring the integration of WordNet and FrameNet. GWC '10, 2010.

[35] J. Li, J. Tang, Y. Li, and Q. Luo. RiMOM: A Dynamic Multistrategy Ontology Alignment Framework. *IEEE TKDE*, 21(8):1218–1232, 2009.

[36] J. McCrae, C. Fellbaum, and P. Cimiano. Publishing and Linking WordNet using lemon and RDF. In *Proceedings of the 3rd Workshop on Linked Data in Linguistics*, 2014.

[37] J. McCrae, D. Spohr, and P. Cimiano. Linking lexical resources and ontologies on the semantic web with lemon. In *The semantic web: research and applications*, pages 245–259. Springer Berlin Heidelberg, 2011.

[38] A.-C. N. Ngomo and S. Auer. Limes: A time-efficient approach for large-scale link discovery on the web of data. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Three*, IJCAI'11, pages 2312–2317. AAAI Press, 2011.

[39] V. Nguyen, O. Bodenreider, and A. Sheth. Don't Like RDF Reification?: Making Statements About Statements Using Singleton Property. WWW '14, 2014.

[40] N. Noy and A. Rector. Defining N-ary Relations on the Semantic Web. W3C Working Group Note, W3C Consortium, April 2006. http://www.w3.org/TR/swbp-n-aryRelations/.

[41] A. G. Nuzzolese, A. Gangemi, and V. Presutti. Gathering lexical linked data and knowledge patterns from FrameNet. K-CAP '11, pages 41–48, 2011.

[42] D. T. Phillips and A. Garcia-Diaz. *Fundamentals of network analysis*. Prentice Hall, 1981.

[43] V. Presutti, F. Draicchio, and A. Gangemi. Knowledge Extraction Based on Discourse Representation Theory and Linguistic Frames. In *EKAW'12*, pages 114–129. 2012.

[44] V. Presutti, A. G. Nuzzolese, S. Consoli, D. R. Recupero, and A. Gangemi. From hyperlinks to semantic web properties using open knowledge extraction. *Submitted to Semantic Web Journal*, 2014.

[45] D. Ritze, C. Meilicke, O. SvÃąb-Zamazal, and H. Stuckenschmidt. A Pattern-based Ontology Matching Approach for Detecting Complex Correspondences. In *OM'10*, 2008.

[46] J. Rouces. Enhancing Recall in Semantic Querying. volume 257 of *SCAI '13*, page 291, 2013.

[47] J. Rouces, G. De Melo, and K. Hose. FrameBase: Representing N-ary Relations using Semantic Frames. In *Proceedings of the 12th Extended Semantic Web Conference, ESWC*, 2015.

[48] J. Rouces, G. de Melo, and K. Hose. Heuristics for Connecting Heterogeneous Knowledge via FrameBase.

In *ESWC'16*, 2015.

[49] J. Rouces, G. De Melo, and K. Hose. Representing Specialized Events with FrameBase. In *DeRiVE '15*, 2015.

[50] J. Rouces, G. de Melo, and K. Hose. Complex Schema Mapping and Linking Data: Beyond Binary Predicates. In *LDOW'16*, 2016.

[51] J. Rouces, G. de Melo, and K. Hose. Klint: Assisting Integration of Heterogeneous Knowledge. IJCAI'16, 2016.

[52] J. Ruppenhofer, M. Ellsworth, M. R. Petruck, C. R. Johnson, and J. Scheffczyk. *FrameNet II: Extended Theory and Practice*. ICSI, 2006.

[53] A. C. Schalley and D. Zaefferer. *Ontolinguistics: How Ontological Status Shapes the Linguistic Coding of Concepts*, volume 176. Walter de Gruyter, 2007.

[54] F. Scharffe and D. Fensel. Correspondence patterns for ontology alignment. In *Proceedings of the 16th International Conference on Knowledge Engineering: Practice and Patterns*, EKAW '08, pages 83–92, Berlin, Heidelberg, 2008. Springer-Verlag.

[55] R. Shaw, R. Troncy, and L. Hardman. LODE: Linking Open Descriptions of Events. In *ASWC '09*, Lecture Notes in Computer Science, pages 153–167, 2009.

[56] C. Subirats. Spanish FrameNet: A frame-semantic analysis of the Spanish lexicon. In *Multilingual FrameNets in Computational Lexicography: Methods and Applications*. Mouton de Gruyter, 2009.

[57] F. M. Suchanek, S. Abiteboul, and P. Senellart. PARIS: Probabilistic Alignment of Relations, Instances, and Schema. *Proc. VLDB Endow.*, 5(3):157–168, Nov. 2011.

[58] F. M. Suchanek, J. Hoffart, E. Kuzey, and E. Lewis-Kelham. YAGO2s: Modular High-Quality Information Extraction with an Application to Flight Planning. In *BTW*, pages 515–518, 2013.

[59] F. M. Suchanek, G. Kasneci, and G. Weikum. YAGO: A Large Ontology from Wikipedia and WordNet. *Web Semantics: Science, Services and Agents on the World Wide Web* , 6(3):203 – 217, 2008. <ce:title>World Wide Web Conference 2007Semantic Web Track</ce:title>.

[60] S. Tonelli and D. Pighin. New Features for FrameNet: WordNet Mapping. CoNLL '09, pages 219–227, 2009.

[61] K. Toutanova, D. Klein, C. D. Manning, and Y. Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. HTL-NAACL '03, 2003.

[62] K. Van Deemter, E. Krahmer, and M. Theune. Real versus template-based natural language generation: A false opposition? *Comput. Linguist.*, 31(1):15–24, Mar. 2005.

[63] W. R. Van Hage, V. Malaisé, R. Segers, L. Hollink, and G. Schreiber. Design and use of the Simple Event Model (SEM). *Web Semantics: Science, Services and Agents on the World Wide Web*, 9(2):128–136, 2011.

[64] M. Yahya, K. Berberich, S. Elbassuoni, M. Ramanath, V. Tresp, and G. Weikum. Natural language questions for the web of data. EMNLP-CoNLL '12, 2012.