Undefined 1 (2009) 1–5
IOS Press

# Integrating Heterogeneous Knowledge with FrameBase

**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, Esbjerg 6700, 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, Aalborg Ø 9220, 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 also different levels of granularity connected by logical entailment. This altogether provides for flexible and expressive seamless semantic integration from heterogeneous sources. It also opens 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. Many institutions rely on Linked Data principles to publish their data using Semantic Web standards [2]. These KBs are mostly based on simple subject-predicate-object (SPO) triples, as defined by the RDF model [20]. Such triples are convenient to process and can be visualized as entity networks with labeled edges.

Commercial search engines exploit them to provide direct answers to user queries, and IBM's Watson question answering system [14], which defeated human champions of the Jeopardy! quiz show, used them to find and 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 [18, 30] 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 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, sev-

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

Table 1

Triple Representations of n-ary Relations

| | | |
|---|---|---|
| **Direct Binary Relation** | | |
| `John` | `wasMarriedOnDate` | `1964 .` |
| **RDF Reification** | | |
| `John` | `marries` | `Mary .` |
| `s` | `type` | `Statement .` |
| `s` | `subject` | `John .` |
| `s` | `property` | `marries .` |
| `s` | `object` | `Mary .` |
| `s` | `time` | `1964 .` |
| **Subproperties** | | |
| `p` | `subPropertyOf` | `Marriage .` |
| `John` | `p` | `Mary .` |
| `p` | `time` | `1964 .` |
| **Neo-Davidsonian (Specific Roles)** | | |
| `e` | `type` | `Marriage .` |
| `e` | `groom` | `John .` |
| `e` | `bride` | `Mary .` |
| `e` | `time` | `1964 .` |
| **Neo-Davidsonian (General Roles)** | | |
| `e` | `type` | `Marriage .` |
| `e` | `agent` | `John .` |
| `e` | `agent` | `Mary .` |
| `e` | `time` | `1964 .` |

eral different representation schemes have been proposed. Table 1 shows some possibilities of expressing that an entity John was married in 1964, some of which also include additional information such as the name of the bride. These representations will me discussed in more detail later in section 2.

As the example shows, 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, as explained above. Existing work on ontology and KB alignment [3] is limited to finding aliases.
2. When **querying**, 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 [34]. Even worse, for the case of a set of different KBs instead of a single coherent KB, there is no simple query (as could be formulated on a single given schema) that will have a high recall across all KBs.
3. When **natural language interfaces** to KBs are queried, state-of-the-art systems typically attempt to map verbs and predicate phrases to RDF predicates [44]. This approach, however, cannot be applied when the KB fails to provide a compatible binary relation.

**FrameBase.** These problems are addressed by FrameBase, a broad-coverage schema that can homogeneously integrate other KBs and has strong connections to natural language. It overcomes the above-mentioned forms of heterogeneity – by sticking to a specific modeling choice general enough to subsume the others (neo-Davidsonian representation) – together with a large vocabulary for events and roles. This vocabulary is reusable and based on an extensible hierarchy. FrameBase also provides a mechanism to convert back and forth between the new representation and direct binary relations, using a vocabulary of binary relations automatically generated from linguistic annotations. These are more concise and can be used when only two arguments are relevant.

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 reification. Section 6 presents methods to integrate knowledge from external KBs into FrameBase. Section 7 provides a qualitative evaluation, and section 8 concludes the paper with an outlook to future work.

## 2. State of the Art

Different approaches for modeling n-ary relations exist, which are summarized in Table 1. Table 2,

| | All triples | Core | Linking event | Reif. Reasoning | Dereif. Reasoning |
|---|---|---|---|---|---|
| RDF Reification | $(n+4)k$ | $(n+3)k$ | $k(k-1)$ | $4k$ Def. clauses | $k$ Def. clauses |
| Subproperties | $(n+2)k$ | $(n+1)k$ | $k(k-1)$ | $2k$ 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 2

Triple Overhead. *$n$ is the number of participants in an event, and $k <= \frac{n(n-1)}{2}$ the number of pairs that are relevant to be linked by direct binary relations. "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 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 which can be expressed as a disjunction of logical atoms with only one negated, which is the consequent when it is written as an implication (rule). In this context, the atoms are of the form triple(subject,predicate,object). In section 5, we will describe more in detail these rules for the case of FrameBase.*

provides a novel comparison of their space efficiency, which has consequences with regards to their applicability for large-scale KBs. Each approach will be discussed in detail in the following subsections.

## 2.1. Direct Binary Relations

A common way to represent n-ary facts is to simply decompose them directly into binary relations between two participants [10]. But 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.

## 2.2. RDF Reification

The RDF standard proposes RDF reification [20], which introduces a new identifier (IRI) for a statement and then describes the original RDF statement 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.

In the different versions of YAGO [22, 39, 40], RDF-reification[1] is used to attach additional information to the event represented by the original RDF triple (evoked by its property) – as in the *RDF-Reification* example in Table 1. This has the advantage that both the original triple as well as the RDF-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 RDF-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.

[1]We will use the term *RDF-reification* because the term reification has other meanings, one of which will be heavily used later in the paper.

- 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-RDF-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. `bornAtPlace`, `bornOnDate`).
- The choice of the primary pair of entities and their binary relation (John and Mary in Table 1) 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 will share the same triple, but require separate RDF-reifications, producing non-unique triple identifiers. For example, if there are two flight connections between Paris and London with different airlines, the triple `Paris isConnectedTo London` will be RDF-reified twice, with two different triple identifiers.

If the triplestore implementation makes use of quads (*http://www.w3.org/TR/n-quads/*), 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 [20] could be used instead of RDF-reification, the problems being the same.

### 2.3. Subproperties

A recent proposal [30] aims to solve some of the issues with RDF-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 the *Subproperties* example in Table 1.

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 RDF-reification, it still suffers from the problems mentioned above related to the existence of a primary pair. For one, the non-RDF-reified binary relationships for the other pairs cannot be inferred from that subproperty.

### 2.4. Schema.org's "Roles"

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 [1], which in practice is equivalent to representing the n-ary relation arising from adding arguments to the binary relation underlying the binary predicate. It works by substituting the object of the binary predicate with a fresh instance of a class *Role*[2] (or a more specific sub-class with its own properties), and appending to this role instance the old object by means of the same binary predicate, alongside other properties such as time, instrument, etc.

For example `:SanFrancisco49ers schema:athlete :JoeMontana` would be converted to:

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

This transformation offers 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 "contract" of the direct binary predicate is broken in the complex pattern. For example, `schemaorg:athlete` has a domain SportsTeam and Person as domain and range respectively, and the semantics is that the

[2]Schema.org's use of the term "role" differs from its standard use in linguistics, which are qualifying properties such as agent and patient [16]. This definition has also been adopted in ontologies, for instance `CaseRole` in the SUMO ontology [36].

object is a person that plays in the team denoted by the object. 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.

An example of how this conflation can lead to problems can be fully appreciated with non-transitive predicates. In the case the predicate was `somekb:fatherOf`, someone's children would become his 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 than in the method using RDF-reification.

## 2.5. Neo-Davidsonian Representations

Another approach, and the one that FrameBase will adapt, is to make use of so-called neo-Davidsonian representations [24, 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 $n$ arguments by means of a property describing the *semantic role* [18, 31].

The process of converting from the binary representation to the neo-Davidsonian one is called reification, but this is different from *RDF-reification* as discussed earlier. In RDF-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 a RDF statement while in the other it is an event.

**Advantages.** Table 2 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.** Unfortunately, there are different ways of using the neo-Davidsonian approach, with different levels of granularity for the events and the semantic roles, from a very small set of abstract generic ones [37] to more specific ones [4].

The Simple Event Model (SEM) Ontology [43] falls within the category of neo-Davidsonian representation with general roles (see Table 1). 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) [37] specifies only very general concepts such as the four just mentioned.

Freebase [4] is a KB built both from 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 so-called *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. There are around 1,870 composite value types in Freebase (1,036 with more than one instance) and around 14 million composite value instances. While CVTs do not represent frames or events per se, from a structural perspective, they can be regarded as isomorphic to a neo-Davidsonian representation with specific roles (see Table 1). However, Freebase places a number of restrictions on CVTs. For instance, they cannot be nested, and there is no hierarchy or network of them that would for example relate a purchasing event to a getting event.

FrameNet [15,35] is a well-known resource in natural language processing (NLP) that defines over 1,000 *frames* with participants (so-called *frame elements*). For example, the verb *to buy* and the noun *acquisition* are assumed to evoke a commercial transaction frame, with frame elements for the seller, the buyer, the goods, and so on.

Previous work has proposed general patterns for using FrameNet in knowledge representation [17] and converted FrameNet to RDF [32], proposing a way to generate schemas from FrameNet. Similarly, the FRED system [33] for building semantic representations from natural language can be configured to use FrameNet.

## 3. System Overview

As seen in the previous section, there are a number of different representations used in KBs. FrameBase will use the linguistic resources FrameNet [15] and WordNet [11] to fully develop an extensive schema for large-scale knowledge representation and integration. The schema is composed of an expressive neo-Davidsonian level that draws on a large common inventory of frames, together with a more concise level of direct binary relations, which is connected to the former by means of inference rules.

### 3.1. FrameNet-based Representation

The use of FrameNet is motivated by the following considerations.

- FrameNet has a long history and aims at descriptions of arbitrary natural language. It thus provides a relatively large and growing inventory of frames and roles, with a broad coverage of numerous different domains.
- 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) [19] 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 [38]. This also makes it much more suitable as a basis for knowledge representation than language-specific syntax-oriented SRL resources such as PropBank [25].
- FrameNet provides an 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 be reified on its own, 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. As section 5 will explain, a second level of representation will be provided as well, which will be based on the direct binary predicates between frame elements, and therefore less expressive but more concise.

### 3.2. Overview

For creating the FrameBase schema using FrameNet, the following steps have been taken, which will be further explained in section 4.

a) **FrameNet–WordNet Mapping.** First, a high-precision mapping is created between FrameNet and another well-known lexical resource called WordNet [11], which will be used to enrich the lexical coverage and relations of the FrameBase schema.
b) **Schema Induction.** FrameNet, WordNet, and the mapping are used to create an RDFS schema for FrameBase that has very wide coverage and is extensible. The schema exploits semantic relations from these components (e.g., synonymy, hyponymy, and perspectivization) to transform the original resources for into FrameBase's lightweight RDFS model.
c) **Automatic Reification–Dereification Mechanism.** Reification–dereification rules are created, in the form of definite clauses that allow the KB to be queried independently either using reified frames or dereified direct binary predicates, and that may also be used to reduce overhead in the KB.

## 4. FrameBase Schema Creation

Before external KBs can be integrated, the FrameBase schema must be created. This process involves creating an initial mapping between FrameNet and WordNet (section 4.1), the use of these resources and mappings to create the FrameBase core schema (Section 4.2). It also involves the creation of reification–dereification rules to enable the use of direct binary predicates (section 5).

### 4.1. FrameNet–WordNet Mapping

While FrameNet [15, 35] is the largest high-quality inventory of semantic frame descriptions and their participants, WordNet [11] 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 to 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. The objective at hand is to map synsets and LUs with the same meaning, so this can be later used to enrich FrameBase's FrameNet-based schema with relations and annotations from WordNet.

More specifically, the objective is to map each LU to one and only one synset. While there are some LUs that could be mapped to more than one synset, this will favor 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 where 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 to model the mapping as a function $S(l|a,b)$ from LUs to synsets as in (1). $S_l$ stands for the synsets that have the same lexical label and POS as the LU $l$, $\mu_{\mathrm{L}}$ and $\mu_{\mathrm{G}}$ 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).

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

The lexical overlap $\mu_{\mathrm{L}}$ 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 connected by a selection of lexical and semantic pointers such as "See also", "Similar to", "Antonym", "Attribute" and "Derivationally related". This expansion is useful to reduce sparsity and 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 $\mu_{\mathrm{G}}$ is the size of the intersection between the set of words in the definition of the LU and the gloss of the synset. CoreNLP library [42] 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 over several randomized seeds, obtaining optimal values $a = 5, b = 0.13$.

## 4.2. Schema Induction

In FrameBase, frames are modeled as classes whose instances are the particular events. 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:** FrameNet's frame inheritance and perspectivization relations are modeled as class subsumption between frames, by means of two specific properties that inherit from `rdfs:subClassOf`, so that both remain distinguishable but contribute to the hierarchy and allow RDFS inference. Additionally, a top frame is declared for the hierarchy. Inheritance between frame element properties is modeled with a direct subproperty relation. Semantic types are sometimes provided as ranges in FrameNet, but their current coverage is limited, and therefore have been left out of FrameBase.
   Under this model, an instance of the *Commerce_sell* frame with a certain *Commerce_sell-Buyer x*, is also an instance of the *Giving* frame and $x$ is the *Giving-Recipient*, because the first frame inherits from the latter. Likewise, it is also an instance of *Transfer* and $x$ is the *Transfer-Recipient*, because *Giving* is a perspective on *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 to each LU are employed. In other words, every LU is treated as evoking its own separate fine-grained frame, denoted as *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.

3. **Intermediate Nodes:** The LU-microframes resulting from the process above are very fine-grained. There are distinct LUs for *buy* from *acquire*. This is a problem for knowledge representation because it increases sparsity. At the same time, some original frames from FrameNet are very coarse-grained, as mentioned above, so they cannot be used. For instance, various kinship relationships such as *mother*, *sister-in-law*, etc. are lumped together. 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. Overall, neither the fine-grained nor the coarse-grained levels are ideal for knowledge representation purposes.

   This is addressed by providing a novel intermediate level composed of *cluster-microframes* that group equivalent LU-microframes and synset-microframes together, solving the problem described above, and integrating synset-microframes into a single backbone.

   The clusters are generated in the following way. First, for each LU-microframe, the corresponding synsets from the FrameNet–WordNet mapping are retrieved. In the case of the mapping in section 4.1, the set has no more than one element, but in the general case it could have more. Then, that set is expanded by adding all other synsets related by lexical relations reflecting cross-POS morphological transformations: “Derivationally related”, “Derived from Adjective”, “Participle” and “Pertainym”. In general, these lexical relations do not necessarily imply any close semantics (e.g., *create/make – creature/animal*), but when restricted to synsets all tied to the same FrameNet frame, such cases are normally factored out. Therefore, the set is reduced to those synsets that also belong to another set produced from a sibling LU from the same frame. 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.

   After the process is run for all LU-microframes and the transitive closure of `framebase:isSimilarTo` is materialized, each cluster is represented by a clique of `framebase:isSimilarTo`. Finally, the intermediate cluster-microframes are reified[3] and declared superframes of the members of the cluster, and subframes of their previously immediate superframe. The cluster-microframe is also connected by `framebase:isSimilarTo` to the subframes. An example of two sibling cluster-microframes with all their members can be appreciated in Figure 1.

   The use of the property `framebase:isSimilarTo` allows to have a direct connection between members of the cluster. It may also be convenient in contexts where a user wants to reduce sparsity by completely merging all members of each cluster. In this case, he can do it as simply as declaring `framebase:isSimilarTo` to be a subproperty of `rdfs:subClassOf` and enable RDFS inference. By virtue of the already materialized inverses of `framebase:isSimilarTo`, every instance of a member of the cluster, including the cluster-microframe, will become 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 [29].

Following the best practices in the Linked Open Data community, we link synset-microframes to URIs in the canonical RDF translation of WordNet [28]. 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 [9]. This allows Frame-

[3]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 if 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.

Base to be transitively connected to other KBs in the Linked Open Data web, as well as provide multilingual support.

## 5. Automatic Reification–Dereification Mechanism

While frames are convenient for representational purposes, users wishing to query the knowledge base benefit from 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 seamlessly convert between frame representations and DBPs. This mechanism can also allow us to avoid materializing frame instances when only two frame elements are needed.

### 5.1. Structure of ReDer rules

The *dereification rules* have the form expressed in Figure 2. 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 DBP (direct binary predicate) has only one set of possible frame and frame elements associated, and therefore chaining reification and dereification rules is an idempotent operation. We term the set of a reification rule and its converse dereification rule as a ReDer (reification-dereification) rule. An example of a ReDer rule is provided in Figure 3.

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

Figure 2. The general pattern of a dereification rule.

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

Figure 3. The general pattern of a dereification rule.

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.
– As clauses with triples as atoms to be fed in general-purpose inference engines, with or without materialization. For example, ReDer rules have also been implemented as rules for the Rubrik reasoner in Jena [5].

Besides the plain `rdfs:label` and `rdfs:comment` annotations, we annotate the DBPs using Lemon [29], which allows 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 structure of the different possible labels for DBPs to create perfect annotations. Similarly, we also use Lemon for annotating microframes.

### 5.2. Creation of ReDer rules

The ReDer rules are automatically built using the annotations of English sentences given for different LUs in FrameNet, namely the grammatical function (GFs) and phrase types (PTs) [35]. Each instance of an example sentence annotated by a frame is accompanied by the GF and PT associated to each of the FEs of that frame filled in that sentence.

For verb-based LUs, FrameNet provides three kinds of GF labels: External Argument (Ext), Object (Obj), and Dependent (Dep). Some of the PT labels that can be found are N, NP, Obj, PPinterrog [35]. Dereified binary predicates and reification-dereification rules are created for the pairs of frame elements whose syntactic annotations for some sentence satisfy the creation rules below, using the GF and PT labels.

As for the general reification-dereification rule pattern in Figure 2, the postfixes "-s" and "-o" indicate the data associated to the FEs that fill the first and second arguments of the DBP, or equivalently, the subject and the object of the resulting RDF triple. The creation of the DBP implies a creation of a dereification rule following the pattern in Figure 2, 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>`.

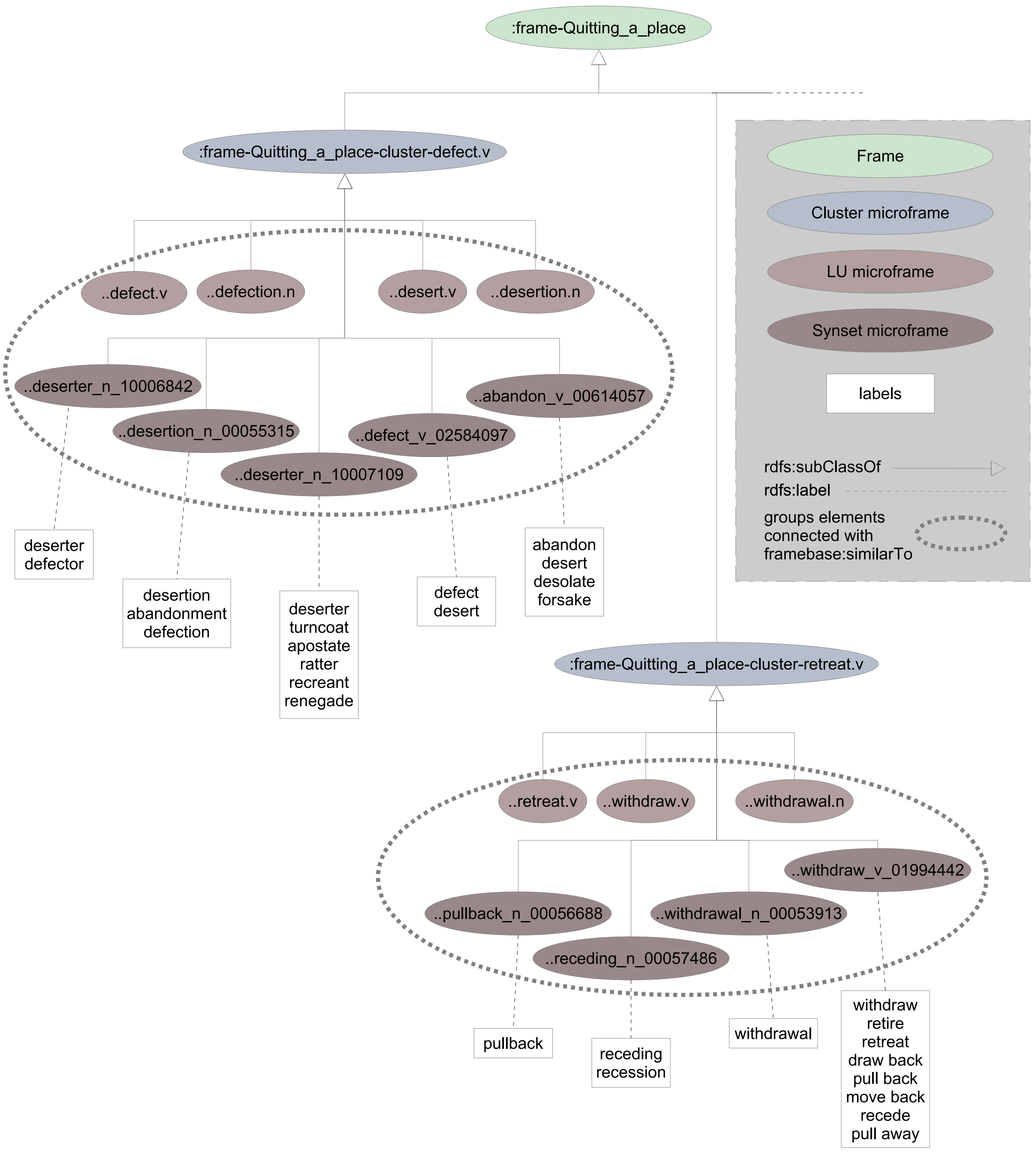

Figure 1. Example of some microframes and labels under the general frame class `:frame-Quitting_a_place`. The initial part of the names of classes is common and has been ommitted.

**Creation Rule 1: Verb Agent-Patient**

*Create DBP with name*
"CONJUGATETHIRDPERSONSINGULAR(LU)"
*if*

```
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 obtained DBPs and reification-dereification rules:*

```
?S :dbp-Forming_relationships-divorces ?O
```
↕
```
?R a :frame-Forming_(...)-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 .
```

**Creation Rule 2: Verb Patient-Agent**

*Create DBP with name*
"is CONJUGATEPASTPARTICIPLE(LU) by"
*if*

```
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 obtained DBPs and reification-dereification 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 .
```

**Creation Rule 3: Verb Agent-Complement**

*Create DBP with name*
"CONJUGATETHIRDPERSONSINGULAR(LU) PREP FRAMEELEMENT-O"
*if*

```
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 obtained DBPs and reification-dereification 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 .
```

*For some FEs in this and the next rule, we assign a specific preposition, like "at" for* `Time` *and "in" for* `Place`*. For example:*

```
?S :dbp-Destroying-destroysAtTime ?O
↕
?R a :frame-Destroying-destroy.v ,
?R :fe-Destroying-Cause ?S ,
?R :fe-Destroying-Time ?O .
```

```
?S :dbp-Intent(...)-establishesInPlace ?O
↕
?R a :frame-Intent(...)-establish.v ,
?R :fe-Intentionally_create-Creator ?S ,
?R :fe-Intentionally_create-Place ?O .
```

**Creation Rule 4: Verb Patient-Complement**

```
    Create DBP with name
"is ConjugatePastParticiple(LU)
Prep FrameElement-o"
    if
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 obtained DBPs and reification-dereification 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 .
```

Using only agent and patient as subject of the triple avoids rules defining certain kinds of DBPs that would be rarely useful, like those connecting the time and place, or the place and the cause.

There is no explicit syntactic annotation in FrameNet to indicate if the verb LUs are evoked in passive form. Therefore, two different heuristics for detecting this. One (IsPassivePosHeuristic(LU)) draws on the POS annotations available in FrameNet, and decides that the target (LU) verb is in passive iff 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 [26], determining that the target (LU) verb is in passive iff 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.

For noun-based LU-microframes, a verb is needed that takes the noun as argument, normally as 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 it explicit both in the IRI and the lexical annotations, in order to avoid ambiguity and prevent incorrect use. The verbs have been conjugated in third person of singular.

For each noun LU in an annotation, the head verb has been extracted by parsing the example annotated sentences with the Stanford dependency parser and searching the paths of dependencies indicated in the creation rules 5 and 6[4]. For brevity, the paths are annotated with the notation of SPARQL property paths, but this is not part of any query.

Creation rule 5 contains several possible dependency paths.

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

[4]We use collapsed CC-processed dependencies, version 3.2.0)

*"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*".

Creation rule 6 fires cases with 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 !*".

```
Creation Rule 5: Verb Noun
    Create DBP with name
"CONJUGATETHIRDPERSON-
SINGULAR(HEADVERB)
LU PREP  FRAME-ELEMENT-O"
    if
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 obtained DBPs and reification-dereification rules:*

```
(...)-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 .
```

```
Creation Rule 6: Verb Particle Noun
    Create DBP with name
"CONJUGATETHIRDPERSON-
SINGULAR(HEADVERB)  VERBPARTICLE
LU PREP  FRAME-ELEMENT-O"
    if
ISNOUN(LU) AND PT-O==PP[PREP]
AND GF-S==Ext AND GF-O==Dep AND (
  LU ^prep_VERBPARTICLE HeadVerb
)
```

*Examples of obtained DBPs and reification-dereification rules:*

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

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

In the cases where the FE-O is included in the DBP but neither the FrameNet annotations or the dependency parsing can provide a suitable preposition come before, we use statistics from the cases where such preposition can be obtained, and we choose, if available, the most common preposition associated to the name of that FE across all frames.

With the rules obtained with the process above, the same DBP can be associated to different pairs of frame elements in a given LU-microframe, owing to different senses or syntactic frames for a given verb (for example the 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 pair of frame elements can also produce different DBPs. To achieve the idempotency mentioned earlier, the Kuhn–Munkres algorithm is used in order to obtain a one-to-one assignment, using as weights the number of annotated example sentences for a DBP and a pair of frame elements, because the patterns with more example sentences are usually more intuitive. The cubic complexity of the algorithm is not a concern because each frame leads to a separate graph which can be handled independently.

## 6. Integration

Knowledge from other KBs such as Freebase can be integrated *integration rules* with two graph patterns as antecedent and consequent sharing some variables. When there is a variable substitution that, applied to the antecedent, makes it a subset of the source KB, then the consequent after the same transformation can be added to the FrameBase instance data (A-Box in the jargon of description logics). When the sources are in RDF, the integration rules can be implemented as SPARQL CONSTRUCT queries. Otherwise, an off-the-shelf RDF converter[5] can be applied to pre-process the source.

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 the first subsection of this section, some example integration rules are presented for integrating events from different sources into FrameBase. Later, a discussion about the complexity of integration rules in general and the challenges they present is added.

### 6.1. Example Integration Rules

The two example integration rules above integrate knowledge from Freebase. They follow a relatively simple pattern, the first reifying a property from the source KB into a frame in FrameBase (Property–Frame), and the second translating a class from the source KB into a frame in FrameBase, and the outgoing properties into FE properties (Class–Frame).

```
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 .
}
```

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

```
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 .} }
```

The next example pertains the `Event` class in DBpedia.

```
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}
}
```

From the 9 properties of the class `Event`, `numberOfPeopleAttending` was omitted because the class `Event` is too general for it, as it has subclasses such as `PersonalEvent` (`Birth`, etc.) and `SocietalEvent`, that appear more appropriate for this. The remaining 8 properties were integrated, but even though the example shares the same basic structure as the Class–Frame rule provided for Freebase, it includes additional complex patterns in the consequent.

The `dbr:Event` class has several subclasses which 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 will provide 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 will also fire the one for `dbr:Event`, because `dbr:SocietalEvent` is a subclass of `dbr:Event`. This rule is very short because 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 rest of the examples from DBpedia below, will also fire the ones for `dbr:SocietalEvent` and `dbr:Event`, because the classes captured in the antecedent are subclasses of `dbr:SocietalEvent`.

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

In the rule above, 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 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}
}
```

We also present the translation of the class `Event` in schema.org. This provides an example of integration. 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.

### 6.2. Complex Transformations

Most of the integration rules we have described follow a pattern which involves an event *class* in the source being translated as a frame class, and each of their outgoing properties being mapped to individual frame elements. However, there are multiple ways in which the rules can differ from this basic pattern.

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 frame elements. 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 frame element 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 frame element. 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. For example, classes `:frame-Commerce_buy-buy.v` and `:frame-Commerce_sell-sell.v`, which are used for classes `Buy` and `Sell` in the organized crime taxonomy, are both perspectivizations of `:frame--Commerce_goods-transfer`. 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 frame elements 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, see how the rule to translate `Event` from schema.org, besides 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 frame elements can be inverted. In this case, the integration rules need to invert the order of the arguments, like in the second appearance of `:fe-Social_event-Occasion` in the integration rule for the class `Event` in schema.org.
3. Oftentimes, a *property* (rather than a class) in the source can be translated as evoking a frame on its own. In this case, the two involved entities become connected to the new frame by means of frame elements. This phenomenon can also appear on its own: an example of this is the first integration rule example, for `freeb:people.person.nationality`.

Arbitrary combinations of these phenomena are possible (e.g. the rule integrating the Event class from schema.org). Overall, this makes automatic generation of the integration rules a very hard task, because it generates so many free variables that any attempt to train a system would face extreme sparsity. In some cases, it may thus make sense to sacrifice some recall, developing a system that only covers simpler transformations.

### 6.3. Representational Flexibility

Finally, another potential challenge for data integration is that even when a homogeneous schema such as FrameBase is used, certain kinds of knowledge can still be expressed in multiple possible ways.

– One example is that there are several ways of narrowing down the meaning of a frame instance. One is creating a new sub-microframe associated with a new lexical unit. Another one is assigning a value to a frame element (see example for `SpaceMission`), as mentioned above. This may lead to divergent choices of representation even within the core part of the schema that comes from FrameNet.

– Another example of this is when a frame element needs to be reified, i.e. represented as a frame instance, to express something additional about it (as would be the case of the property `previousStartDate` in schema.org), or when there is no direct frame element available and creating it would lead to a combinatorial explosion in the size of the schema. An example of the latter is the difference between our proposal for using the frame `Part_whole` for expressing sub-event relations, and how we used the frame element `Occasion` for the frame `Social_event`, but this is a particularity of that frame. Again, this may lead to an incoherent representations in the knowledge base. One potential way of addressing this would be extending the reification–dereification mechanism of FrameBase.

## 7. Evaluation

This section evaluates the quality of the results and show some example queries.

### 7.1. FrameNet–WordNet Alignment

To evaluate the created schema, the created FrameNet–WordNet mapping has been compared to the MapNet gold standard [41]. MapNet uses older versions of FrameNet and WordNet, so mappings from WordNet 1.6 to 3.0 [7] 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 desired, the approach described in section 4 achieves high precision, while still maintaining good recall. 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 [41] report "0.90 as Cohen's Kappa computed over 192 LU-synset pairs for the same mapping task" by [8]. More generally, [12] 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".

### 7.2. Schema Induction

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.

An average precision of 87.55% ± 6.18% with a 95% Wilson confidence interval has been obtained. The evaluation showed a small change of nuance for 31.15% ± 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 precision has been calculated from a random sample of 100 intra-cluster pairs that have been independently annotated by two of the authors. The linear weighted Cohen's Kappa over the three-valued combination of the two variables with which are annotated for each cluster pair, has a value of 0.23 over a maximum of 0.87. The scores were obtained with a random annotator.

In addition to the number of frames, the FrameBase schema provides a vocabulary of frame elements that goes well beyond the knowledge currently included in most KBs, in particular beyond time and location. This additional knowledge is routinely conveyed in natural language, and it seems likely that using a schema that provides for it paves the way to include it in KBs, either manually or automatically.

### 7.3. Reification–Derefication Rules

Additionally, reification–dereification rules are provided, with the same number of direct binary predicates, with both human-readable IRIs and lexical labels. 14,930 are verb-based and 10,270 are noun-based. The obtained average precision for verb-based rules is 96.22% ± 3.22%, and 80.43% ± 7.61% of the correct rules were found easily readable. For noun-based rules, the scores are 87.5% ± 6.41% and 91.91% ± 6.28%. 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. For this evaluation, the same annotation methodology as for the intra-cluster pairs was followed, obtaining a Cohen's kappa of 0.39 over a maximum of 0.54.

| | Prec | Rec | F1 | Acc |
|---|---|---|---|---|
| SVM Polynomial kernel 1 [41] | 0.761 | 0.613 | 0.679 | — |
| SVM Polynomial kernel 2 [41] | 0.794 | 0.569 | 0.663 | — |
| SSI-Dijkstra [27] | 0.78 | 0.63 | 0.69 | — |
| SSI-Dijkstra+ [27] | 0.76 | 0.74 | 0.75 | — |
| Neighborhoods [13] | — | — | — | 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.

### 7.4. Querying

FrameBase facilitates novel forms of queries. The following query, for instance, uses reified patterns to find the heads of the World Bank. Note that the clusters implemented in RDFS allow searching for the noun *head* (from the leadership frame), although the integration rule above only produced an instance of `fmbs:frame-Leadership-leader.n`. The results in Table 4 show example instances seamlessly 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).

```
SELECT DISTINCT
?leader ?leaderLabel ?role ?roleLabel
WHERE {
  ?lumfi a :frame-Leadership-head.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 .
  }
}
```

Alternatively, a direct binary predicate from the dereification rules can be used to obtain the same non-optional results, as illustrated in the query below. Either *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* frame elements for both.

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

FrameBase can also be applied with natural language processing tools for question answering and data mining. For example, given the question "Who has been the head of the World_Bank", the SRL tool SEMAFOR [6] 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 like AIDA [23] matches World_Bank to the entities in the KBs, the structured query can easily be built. Moreover, the same procedure can also be used to integrate new knowledge from a text into the KB, like FRED [33] does.

## 8. Conclusion

FrameBase is a novel approach for connecting knowledge from different heterogeneous sources to decades of work from the NLP community. Events can be described in very different ways across different knowledge bases. Our framework not only provides an efficient model to describe n-ary relations, but also integrates and transforms FrameNet and WordNet to yield a broad-coverage inventory of frames. Additionally, linguistic annotations in FrameNet such as the ones used to create the reification–dereification rules can also be used to generate natural language, for instance, for summarizing a portion of a KB for non-technical users.

In our future work we will continue our efforts to integrate arbitrary knowledge with frame structures by automatically generating integration rules such as the examples in section 6, for arbitrary

| ?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 4
Results from the query

knowledge bases. Given FrameBase’s close connection to natural language, we also intend to study methods for better adapting semantic role labeling tools to question answering [6].

The state-of-the art FrameNet SRL system is Google-internal [21], but the CMU system is close [6]. Using FrameBase, would automatically benefit from the rapid advances in NLP.

Details and more information about FrameBase are available at `http://framebase.org`. 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). as well as China 973 Program Grants 2011CBA00300, 2011CBA00301, and NSFC Grants 61033001, 61361136003, 20141330245.

## References

[1] Roles in Schema.org. Technical report, W3C, 2014. http://www.w3.org/wiki/WebSchemas/RolesPattern.
[2] C. Bizer, T. Heath, and T. Berners-Lee. Linked data–the story so far. *IJSWIS*, 5(3):1–22, 2009.
[3] C. Böhm, G. de Melo, F. Naumann, and G. Weikum. LINDA: Distributed Web-of-data-scale Entity Matching. CIKM ’12, pages 2104–2108, 2012.
[4] 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.
[5] J. J. Carroll, I. Dickinson, C. Dollin, D. Reynolds, A. Seaborne, and K. Wilkinson. Jena: Implementing the semantic web recommendations. WWW ’04, 2004.
[6] 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.
[7] J. Daudé, L. Padró, and G. Rigau. Mapping wordnets using structural information. ACL, 2000.
[8] D. De Cao, D. Croce, and R. Basili. Extensive Evaluation of a FrameNet-WordNet mapping resource. LREC, 2010.
[9] 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.
[10] L. Del Corro and R. Gemulla. Clausie: Clause-based open information extraction. WWW ’13, 2013.
[11] C. Fellbaum, editor. *WordNet: An Electronic Lexical Database*. The MIT Press, 1998.
[12] C. Fellbaum and C. F. Baker. Can WordNet and FrameNet be Made “Interoperable”? ICGL ’08, 2008.
[13] O. Ferrández, M. Ellsworth, R. Munoz, and C. F. Baker. Aligning FrameNet and WordNet based on Semantic Neighborhoods. LREC ’10, 2010.
[14] 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.
[15] C. J. Fillmore, C. R. Johnson, and M. R. Petruck. Background to Framenet. *International journal of lexicography*, 16(3):235–250, 2003.
[16] W. Frawley. *Linguistic semantics*. Hillsdale, 1992.
[17] A. Gangemi and V. Presutti. Towards a pattern science for the semantic web. *Semantic Web*, 1(1):61–68, 2010.
[18] 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.
[19] D. Gildea and D. Jurafsky. Automatic labeling of semantic roles. *Computational linguistics*, 28(3):245–288, 2002.
[20] P. Hayes and P. Patel-Schneider. RDF 1.1 semantics. Technical report, W3C, 2014. http://www.w3.org/TR/rdf11-mt/.
[21] K. M. Hermann, D. Das, J. Weston, and K. Ganchev. Semantic Frame Identification with Distributed Word Representations. In *Proceedings of ACL*, June 2014.
[22] 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.
[23] 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.

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

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

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

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

[28] 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.

[29] 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.

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

[31] 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/.

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

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

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

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

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

[37] 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.

[38] 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.

[39] 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.

[40] 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>.

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

[42] 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.

[43] 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.

[44] 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.