FernUniversität in Hagen
Fakultät für Mathematik und Informatik
Lehrgebiet Wissensbasierte Systeme

Between Facts and Knowledge

Issues of Representation on the Semantic Web

Thomas Lörtsch

Master Thesis in Applied Computer Science (M.Sc.)

Version 1.0.2
incorporating corrections and suggestions
by the supervisors

Supervisor: Prof. Dr. Christoph Beierle
Co-Supervisor: Prof. Dr. Aidan Hogan

Abstract

The aim of the Semantic Web is to foster meaningful exchange and integration of data in a completely decentralized fashion and on a global scale, enabled by a small set of conventions and shared vocabularies — extending the human-oriented document space of the World Wide Web to a shared data space that can be navigated, used and built upon by applications. To foster such unprecedented interoperability the Semantic Web is based on a very simple data model of directed graphs and an Open World semantics.

This formalism however runs into the limits of its expressivity quite quickly when not only simple facts but more complex knowledge structures need to be represented, and what first seems straightforward can turn tedious, verbose and hard to understand and manage. As a consequence an immense range of works and proposals has been brought forward to ameliorate the situation, tackling many expressive needs and following manifold varied approaches. However, none of those contributions has been able to meet the demands and desires of users to a degree that the issue could be considered resolved. The underlying problems of knowledge representation, some of which have plagued philosophers for millennia, run deep and often interact and interfere in subtle and hard to manage ways.

This thesis first provides an inventory of what the Semantic Web's stack of specifications supplies and which demands were articulated in research and materialised in real world application. On that background it presents a comprehensive account of the abundance of approaches that have been brought forward during the last two decades to improve on or resolve the described problems. It then develops criteria to systematically characterize and compare the features and strategies of those approaches. Further analysis however suggests that under the surface those works often follow implicit intuitions that aren't necessarily well understood, let alone consciously decided on. It seems that "the Gestalt of it all" is indeed quite complex, contradictory even. A consistent solution may therefore need to be composed of a well-designed combination of multiple approaches rather than striving for a single magic bullet. In any case it seems that a better understanding of the conventions and constraints faced by any attempt on knowledge representation and a thorough discussion of the semantics of the Semantic Web are an indispensable prerequisite to resolving its issues of representation.

Contents

Chapter 1

Introduction

People think RDF is a pain because it is complicated. The truth is even worse. RDF is painfully simplistic, but it allows you to work with real-world data and problems that are horribly complicated.

Dan Brickley and Libby Miller[1]

When the World Wide Web began to gain traction and grow at an exponential rate in the early 1990s, Tim Berners-Lee — who is rightfully credited as being its inventor, although he didn't do it all by himself — was already advancing an even further reaching vision: the Semantic Web. The idea was that all that information that was published on the web in human-readable form, and more, should also be accessible to machines, in a way so that applications could understand the meaning of that data and make it usable by automated agents. "Understanding" and "meaning" are big words in the context of computers and may allude to grand visions of Artificial Intelligence that had gloriously failed in the decades before. The approach that Tim Berners-Lee advocated was much more down to earth, and very consciously so: his vision of the Semantic Web was not some kind of AI singularity but focused on scalability and low-hanging fruit, making data that already loiters in closed databases broadly accessible and re-usable.

The World Wide Web became a tremendous success although in terms of theory and practice of hypertext systems at the time it had been woefully behind the state-of-the-art. So why not do the same in the field of AI, riddled of a history of several generations — and hype cycles — of AI systems that didn't scale well, couldn't interoperate, required expert logicians to operate and still were not nearly as powerful as initially hoped for. In stark contrast to that tradition the design idea behind the Semantic Web was rather profanely akin to a very big decentralized database that spans the whole World Wide Web. Data would be published like web pages, encoded in commonly agreed on and shared vocabularies and following certain rules that allowed applications to read

[1]From the foreword to "Validating RDF Data", [Gay+17]

that data and work on it without any further prior knowledge of or agreement between the participants — much like any web browser can display any web page to any human reader because the format in which web pages are encoded is fixed and the meaning of tags like 'heading', 'quote' and 'list' is shared and agreed upon among all readers. As most web pages were already generated by applications from structured data sources the extra step required to build the Semantic Web didn't seem prohibitive, but the added benefit enticing.

To make this idea work on a global scale — and to not fall into the well-known AI trap of over-ambitious lofty goals — it had to be kept simple enough to be easy to understand, straightforward to implement and performant to work with. A very basic data model of a directed graph built from binary relations was chosen as it provided maximum flexibility and required only minimal upfront agreement. A small vocabulary was designed to express the most basic constructs that a data model needs like types, relations, lists, a few basic datatypes and some more. This provided the ground for more extensive vocabularies, covering all kinds of specific application domains, to build on. More advanced logical formalisms were envisioned to facilitate more expressive ontological constructs that would help to expose implicit knowledge hidden in application specific formalizations, furthering the exploitability of data to other uses than those intended by its original publishers. Complexity was delegated to conceptualizations encoded in shared ontologies, conventions negotiated out-of-band and more involved processes encoded in application logic. Like on the web essential parts of the design are driven by liberal rather than rigid formalisms. A late binding style of handling semantics tries to avoid common pitfalls that plague knowledge representation by leaving a lot of ambiguity in the way IRIs identify the subjects of discourse. This lowers the barriers of entry and provides flexibility in the near term.

However, a few issues quite quickly became apparent. Most obvious was the need for a meta-modelling mechanism that could apply the Semantic Web machinery on itself: in an open web where anybody can publish anything about anything it is indispensable to be able to manage data by e.g. source or date or viewpoint. Applications need to be able to record and act on such metadata. This has proven to be a formidable problem that, although many have tried, has not been resolved to general satisfaction until this day. Other problems proved equally obstinate. The most fundamental building block of the Semantic Web, identification of subjects of discourse through IRIs, has ambiguous semantics — quite a bummer in theory, thankfully mostly less problematic in practice, nonetheless a worrisome potential breaking point. Formal model-theoretic semantics needed to be introduced to define the meaning of the basic language constructs in unambiguous terms, but such rigour and the implementation effort it requires often doesn't sit well with practitioners concentrating on specific tasks where semantics are sufficiently clear-cut by the immediate application context. Perhaps most importantly the focus of the Semantic Web on global integration of shared data routinely collides with the intuitions and requirements of local applications to control and act on that data, but no mechanism is provided to manage and reconcile these two realms. The list of unintuitive, underspecified or plainly unavailable features is long: relations with more than two participants, basic data structures like lists, qualification of topics or relations, self-reflection required to manage data integration or competing viewpoints, complex objects, grouping of statements, the subtle semantics of blank nodes etc. The few facilities that RDF provides in attempts to reconcile those competing demands and meet the more advanced needs lack either a solid semantics or a broadly accepted syntax. They rather have themselves been the source of much debate, misconceptions and frustration.

Human communication is immensely rich and varied in form, often intricate, highly faceted, most of the time contextual, sometimes suggestive, and on closer inspection even outright evasive. The formalizations and vocabularies on which the Semantic Web is based try to capture at least the most essential parts of it in ways easy to use and performant to compute. However, moderating the tension between simplicity and expressivity is incredibly hard to get right. It requires designing a language that is not overwhelmingly complex but still usefully expressive and operating on correct assumptions in most cases, sensibly capturing intuitions that are all too often dependent on context and application, avoiding verbosity but also hidden implicit fixings that might later bite the user, providing more expressive constructs when the need arises without invalidating any prior work.

Over the years a lot of effort has been spent on research and development of approaches to extend the basic binary formalism of RDF with means to express complex knowledge structures without jeopardizing its essential features of scalability and simplicity. Standardizing triple-based modelling in common design patterns is touted as a way to ease use and reuse of complex structures while retaining reasoning support. Attributed relations extend the basic triple with properties on relations and sometimes even nodes, facilitating layered representation of primary facts and secondary detail. Grouping triples in annotated graphs is a means to support integration through annotations on provenance and such but can just as well be used to model any other aspect. Non-standard extensions on the storage level through triple identifiers are another popular means to annotate triples, bypassing the verbose reification mechanism RDF provides. Fluents encapsulate additional detail in the nodes of a relation, effectively splitting an identifier in many facets. Sometimes the data model is extended with more primitives and additional positions or vocabularies encode application specific semantics. Syntactic sugar can provide a user-friendly interface over involved triple-based structures.

All those angles have been investigated, worked on, modified, tested, dismissed and scrutinized again, creating an immense corpus not only of published work but also of passionate mailing list threads, highly instructive blog posts and sprawling working group archives. The basic dilemma however remains the same for all those works. They may preserve the basic triple but lose generality in nodes and properties by cutting the world into smaller and smaller pieces, tessellating the shared vocabulary for expressing relations and thus compromising basic pillars of intuitiveness and shared understanding. They may extend the triple to n-tuples and lose simplicity and reasoning support. Or otherwise they may combine triples in complex constructs in which the simple link between pairs of entities gets hidden, and ultimately lost, dissolving the basic fabric of the Semantic Web in a noisy, unwieldy mess. Any successful solution to this fundamental problem will have to walk a very thin line between expressivity and simplicity as anything too malleable will lead to Fragmentation, anything too involved won't gather wide acceptance, anything too subtle will be ignored or misused.

To make matters worse this is not just a problem of a missing syntactic feature or modelling primitive. The underlying problems of knowledge representation run very deep — some have plagued philosophers for millennia — and they often combine and interfere in subtle, not easy to understand ways. As a matter of fact after 25 years of Semantic Web development there still isn't even a clear and broadly agreed upon understanding of what exactly the problem is ("and if so, how many?"). This might help explain why it has proven so hard to come up with a consistent proposal.

1.1 Objective

The RDF triple provides the Semantic Web with a solid foundation with great expressivity. It is extremely flexible, easily approachable, intuitive to navigate and straightforward to query and reason about in low-dimensional scenarios. It is simplistic, but powerful. It does however not scale well to more complex use cases. The Semantic Web could greatly profit from a mechanism to model complex facts, rich on facets and perspectives, while still remaining true to its straightforward and simplistic formalism of binary relations. The question is how to best extend this basic idiom to better capture the more involved scenarios, the outliers, the special cases that in true 80/20 fashion any solution has to cover if it aims for general usefulness.

This thesis first examines what we already have, how it makes sense and where it falls short of expectations. It aims to provide a solid account of both the expressive means and the structural problems embodied in the formalisms and specifications that drive the Semantic Web and an understanding of the state of the art after almost three decades of bootstrapping, standardization and application.

Part I provides an *Inventory* of the state of the art. It first analyses the basic Semantic Web specifications *Supply* w.r.t. model, syntax and semantics in Chapter 2. In that context it also tries to free the issue of 'semantics' from its stigma of being a rabbit hole best avoided for the sake of one's mental well-being by discussing what some constructs actually *mean* and how that corresponds to the Semantic Web's primary focus: the integration of data from different sources. Chapter 3 gives an overall perspective on *Demand*, the range of information modelling needs and desires that arise in use as witnessed and documented in research and application. It is important to recognise the width and depth of this problem to be able to correctly assess the merits of concrete proposals.

Part II documents the *Approaches* brought forward since the first round of Semantic Web standardization efforts culminated in the 2004 specifications of RDF and its accompanying ontology languages RDFS and OWL. This is a rich and varied field with an abundance of innovative approaches, clever combinations, ingenious hacks and inspiring extravaganzas. Those works touch on most aspects of Semantic Web engineering: syntax and model, vocabulary and formalism, operational and model-theoretic semantics, querying and reasoning.

Above all this thesis provides a comprehensive overview of research and proposals targeting *Representation* in Chapter 4 as the most immediate aspect of complex modelling and meta-modelling: works that focus on syntactic means to facilitate complex structures and meta-modelling, either by introducing new constructs or by re-defining or extending existing features, all for the sake of more intuitive and expressive facilities to

– add administrative metadata to statements,
– refine and qualify statements,
– form complex objects from simple statements,
– explicate and integrate application contexts and
– provide more elegance, but also more expressivity.

It also touches on works in the closely related areas of *Identification* in Chapter 5, *Serialisations* in Chapter 6, *Ontologies* in Chapter 7, *Querying* in Chapter 8, *Reasoning* in Chapter 9, *Implementation* in Chapter 10, *Graph Models* in Chapter 11 and *Example Use Cases* in Chapter 12. These treatments however are less detailed. They rather aim at giving an overview and establishing some relations and dependencies.

Part III on *Semantic Graphs* analyses what has been laid out so far — supply, demand and all kinds of approaches to make them meet more tightly — and makes an effort to put all those findings in perspective.

Chapter 13 on *Schrödinger's Triple* rubs some salt into the wound of the Semantic Web: there is indeed often little understanding of the informal semantics involved in the ways facts are woven into knowledge. This chapter makes the case for the extra effort required to develop sound but also intuitive semantics, navigating between over-zealous attempts to formalize the tiniest nuances of natural language and sloppy nonchalance that takes local context for shared understanding.

Chapter 14 on *The Gestalt Of It All* tries to develop a deeper understanding of the problem space and its many dimensions beyond technicalities, measures and syntactic considerations. This is the original territory of semantics, not understood as formal model-theory, but as the search of the actual meaning of both the constructs we use and the expressive goals we try to achieve.

Chapter 15 focuses on *Perspectives*: what possible conclusions might be drawn from all these analyses? It recalls important principles to guide the design of a successful meta-modelling approach, points out some low-hanging fruit that an agile standardization process might quickly pick to ease at least some of the most pressing pains, and in Section 15.5 sketches an experimental proposal, named *Silhouettes*, to test the waters and see how far a clever combination of approaches might carry.

In Appendix A a range of *Abstractions* is established, based on an analysis of the commonalities, differentiations and motivations that seem to drive the works introduced in Part II. It derives a scheme of categorizations to compare and relate the different approaches based on *Primitives* in Section A.1, *Content* in Section A.2, *Form* in Section A.3 and *Function* in Section A.4.

The Semantic Web is an effort targeted not at logicians and AI specialists, but at all kinds of knowledge workers, both trained and untrained in the field of knowledge representation: from web designers to librarians, from database administrators to business strategists, from domain experts to system integrators, from lay authors to data scientists. It therefore needs to stay with simple, even simplistic tools and instruments. However, it may well fail if those tools can only accomplish tasks that are too simple to express the complexities of practice in straightforward fashion. The right balance still has to be found.

1.2 Methodology

This thesis aims for a comprehensive coverage of its subject. To constrain the area it largely ignores the initial bootstrapping period beginning in the mid-1990s and leading to the second round of Semantic Web standardization in 2004, when version 1.0 of the RDF specification was published, most notably providing RDF with a solid model-theoretic foundation. That point in time is also of significance because in the same year two important ontology languages were standardized: the very basic RDFS and the much more elaborate OWL. Sometimes older sources are referred to but only to help explain what RDF provides now.

A second constraint is thematic: while extending the expressive powers of the Semantic Web can be achieved by many means and needs to consider many angles, this work concentrates on representational approaches. Prominent representational topics include reification, named graphs, meta-modelling in general and n-ary relations, but also treatments of other controversial issues like lists and irregular complex objects are examined. Related areas like reasoning, querying, serialisations, implementations and ontologies are discussed too as they obviously are part of the bigger picture, but not in the same depth as approaches focusing on issues of representation. Still it seems essential to cover the whole terrain at least in passing and all relevant questions and aspects should at least be introduced and helpful entry points for more thorough exploration given.

The description and analysis of the expressive powers of RDF is first based on the relevant specifications published by the W3C. Not only the most current versions were consulted as especially the 2004 versions of RDF specifications sometimes provided more explanations into the semantic considerations driving some design choices. Further very valuable resources were Semantic Web related mailinglists and archives of W3C working groups. Academic papers discussing the syntax and semantics of RDF are rather rare, at least in the time frame studied.

With respect to representational approaches this investigation has cast a net as wide as possible. Databases of scholarly work have been searched for a plethora of keywords like 'reification', 'named graph', 'annotation' and 'contextualization', and many more. The proceedings of leading conferences in the field like ISWC and ESWC as well as relevant journals were consulted. The bibliographies of works gathered provided further valuable pointers to other works, and so on. The aim was to make this study as comprehensive as possible. While it is practically guaranteed that not *all* works form this area are covered in this thesis, this author is rather confident that not many are missing either (but nonetheless very sorry for those that are!).

The attempt to abstract away from concrete proposals and their often quite specific use cases to underlying principles and more general aches is mostly based on extracting and gathering in one place what is only implicit in the different proposals and efforts. It did profit from a few works that developed a wider perspective but in general it seems that attempts to analyse and generalize the various and recurring design issues of the Semantic Web have not attracted much attention within the community recently.

1.3 Outlook

The amount of work and the wide spectrum of proposals that has been brought forward is indeed astounding. It seems like no stone has been left unturned in the search for better meta-modelling facilities and in general for modelling primitives that ease expression and management of complex knowledge structures beyond simple facts. Yet no proposal so far can satisfy all usability requirements and more than a limited array of use cases.

The representation facilities of the Semantic Web in many ways resemble the World Wide Web in its early days when layout was largely constrained to headings, bold text and bulleted lists. Tables were used as a hack to structure pages into regions just like RDF Datasets use named graphs to partition the data space, albeit without any shared semantics. HTML's infamous `spacer.gif` has many equivalents in the proposals brought forward to add meta-modelling and application style semantics to the Semantic Web. Such rather unprincipled attempts to solve all problems in one genius stroke however are bound to fail. What is really needed is an equivalent to what Cascading Style Sheets brought to HTML: a categorical step forward in terms of expressivity, backwards compatible to all the data and machinery already out there, based on a solid understanding of both the users' needs and the fundamental properties of knowledge representation.

The most surprising insight from this thesis might be how different the representational needs are and how intricate the semantics of what is to be expressed can be. What is obvious from context to a human speaker can be very tedious to formalize in representation and the variations can be overwhelmingly hard to understand, differentiate and pin down. Knowledge representation is a field with a long tradition, leading back to questions discussed by Greek philosophers millennia ago and with no straightforward answer up to today.[2] In fact, it shouldn't come as a surprise that the Semantic Web hasn't been able to provide a simple and convincing answer to all of them. Yet simply ignoring those questions is not an option either as the widespread dissatisfaction with what RDF provides vividly illustrates. This area probably needs a lot more work and discussions to ensure a better shared understanding of the gestalt of the problem space and the intricacies that knowledge representation has to master.

This thesis' combination of anthology and abstraction hopefully will enable a fresh look, maybe facilitate some new, or very old, ideas or a hitherto untried combination thereof. Some immediate results are given but for sure more work is necessary to develop a broadly satisfying solution for representing knowledge structures that transcend the realm of simple facts. That is a bold endeavour requiring effort and courage, but without such determination the Semantic Web effort will just continue to plaster Band-Aid on Band-Aid, and risks getting stale over time. Sometimes the Semantic Web seems in need of a new influx of enthusiasm: many obvious shortcomings are lingering and well versed bricolage is often considered the most sensible approach to seemingly unsolvable problems. Semantics itself has become a bad word. Understanding semantics is indeed not always easy, but indispensable. The necessity to get them *right* has to be understood, and the impossibility to get them *tight* has to be understood as well. Then it should be possible to develop a solution that strikes a better balance between simplicity and expressivity.

[2]See [Sow00] for an equally thorough and inspiring introduction

Part I

INVENTORY

Use and envisioned use of the Semantic Web has gone through some iterations. The initial vision that personal agents would roam the Semantic Web autonomously and use the published data to deal with everyday tasks on behalf of the user did never materialise. There was for quite some time a chicken-and-egg problem that not enough data was published to drive interesting applications that in turn would need data to work on. The Linked Data initiative tried to bootstrap the data side of things but the resulting Linked Open Data (LOD) cloud consisted mostly of silos of data that weren't interconnected much. As search engines began to use metadata to enrich search results the incentive to publish metadata RDF (or other formats) greatly improved, albeit concentrating on search engine optimization (SEO) purposes. Another important search engine related impulse was Google introducing a so-called *Knowledge Graph* from which it generated enriched search results in the form of info boxes. Knowledge Graphs became the latest buzzword for integrating data in a way that the result is more than the sum of the individual sources. Enterprises connect their data silos into Knowledge Graphs to discover, explore and manage dependencies. Data scientists and researchers collect data from different sources to derive new insights on social public issues, climate change or Covid-19. In these environments the value proposition of the Semantic Web shines: labelled directed graphs can easily integrate any existing data schema and well-defined semantics and shared ontologies enable self-describing data that doesn't need a domain expert (or a database administrator intimate with the data) to understand. Such data can be published, used and reused freely to unforeseen applications and benefits. It seems that the Semantic Web finally has reached the point where it doesn't need academia and public funding any more to make its way into practical application: there are now enough chickens as well as eggs.

On the other hand however the situation is not all rosy at all: the Semantic Web is an endeavour of unprecedented size and breadth and its design couldn't build on many successful predecessors or time-tested approaches. This shows in more than one place, resulting in some rather unsatisfactory or even frustrating issues and shortcomings. The ideal of a decentralized but fully integrated and shared global knowledge space is still a distant goal. Despite the impressive results so far a lot of bits and pieces are missing, and it is not yet entirely clear if the aim is indeed achievable with reasonable effort. The problem is that representing knowledge that surpasses simple "one-dimensional" facts is indeed a very complex endeavour and knowledge that extends beyond narrowly-specified domains, not to mention on a global scale, is all but one-dimensional. The Semantic Web however has so far not been specified to deal with such complexities. It provides some stubs and some merely syntactic features that applications can re-purpose to their liking but nothing that would allow capturing complex knowledge beyond flat world excerpts of reality.

Users do navigate those shallows by re-purposing and overloading constructs, trying to game RDF and bend it to their needs, and most of the time not even consciously so. Logic however is a rather rigid matter — it doesn't bend, it breaks — and the results is only acceptable as long as it is constrained by local application semantics. The first victim is interoperability.

First, Chapter 2 on *Supply* will investigate in detail what RDF provides and discuss the rationale behind its design. Then Chapter 3 on *Demand* will give an overview of use cases and motivations to overcome its limitations. This provides the background for a detailed presentation of *Approaches* in Part II.

Chapter 2

Supply

This chapter will introduce the purposefully simplistic design of RDF — focusing on the very basic tools and methods that RDF provides to model complex subjects — to establish the necessary background for the discussion of a plethora of efforts and approaches brought forward to extend RDF's expressive powers in later chapters. It starts with the fundamental issue of how to identify subjects of discourse, then introduces the basic primitives of RDF, followed by the vocabulary that RDF provides to support not only strictly binary relations and meta-modelling. It then investigates some of RDF's syntactic facilities, especially the infamous construct known as 'blank node', followed by a deep dive into the much-discussed formal model-theoretic semantics of RDF. This order reflects the historical development of RDF: the first RDF specification from 1999 lacked a formally defined semantics — the model-theoretic semantics was only added in the second round of standardization in 2004 for reasons that will be discussed in the respective section.

After the expressive means of RDF have been introduced this chapter will conclude with a discussion of how well they support advanced representational strategies, here subsumed under the label of meta-modelling: qualification, annotation, contextualization and structuring of information into entities both more elaborate and more elegant than the basic triple.

2.1 Identification

The Semantic Web is built on shifting grounds. Nothing about it is as solid and well-rounded as one would like to think. Its design is full of compromises between the desire to make things simply work and reap the benefits of all that data readily available out there, and on the other hand the complexities and intricacies of human knowledge and communication that over the millennia have evaded any and all attempts to formalize and operationalize them in some clear-cut, easy to use and all-encompassing scheme of things.

And the troubles begin with identification itself.

At the very heart of the Semantic Web lurks a nasty little problem: IRIs are the central mechanism to identify the subjects of discourse but depending on application it can be quite unclear what exactly they identify. The simple and elegant naming mechanism that the World Wide Web relies on to access web pages, services, videos and many other types of resources on the web gets into trouble when its reach is extended to not just address information resources on the web but to reference anything whatsoever, especially resources external to the web like people, things, locations, events, concepts, theories etc.

Identification may therefore be the topic where the tensions between application-driven pragmatic intuitions and the endless variations and subtleties inherent in human communication become the most apparent. This has been dubbed the 'Identity Crisis of the Semantic Web' and was the subject of many heated debates in the early days of the Semantic Web.

Nomenclature: Reference, Denotation, Identification, Indication The usage of terms in this area can be confusing and a short list of vague definitions may be helpful to navigate the space and for reference, although some of it will probably only become clear later in the course of this chapter.

identify, identification These words are often used as the overarching terms that can mean anything from what a resource is about to its textual representation itself. The RDF 1.1 specifications however use these terms to specifically describe a situation where the meaning of a term is defined by widely accepted norms and common practice established outside the RDF specifications.

access, address, indicate These words describe the relation between an IRI and the resource that can be retrieved when accessing the IRI via the protocol in which it is encoded. Accessing the IRI `<https://www.paris.fr>` via a web browser will return a web page, encoded as HTML, and that is what the IRI addresses.

refer to, referent, reference These words describe the relation between an IRI and the thing it is about. The referent of 'Paris' in natural language is most probably the city Paris, the referent of `<https://www.paris.fr>` may be the website so addressed, but it might also be the city Paris or something related to this city. This will be discussed below. It is good practice to use the words 'access' etc when speaking about a network retrievable resource, aka a web resource, but use words like 'refer' etc when discussing what that resource describes or is about.

denote, denotation The terms 'refer to' and 'denote' and their cognates are used interchangeably in the specifications and in this work.

Philosophical Problems [HH10] provide a brief introduction into issues of identification in philosophy and Knowledge Representation efforts preceding the Semantic Web. [Raa+19] summarize that from a philosophical standpoint there are two mayor problems concerning identity: temporality and contextuality.

Identity has a temporal component: a car that has all its parts exchanged over the course of its lifespan may be considered the same car still, or not, depending on viewpoint. Likewise, a person as an infant is and is not the same person as an adult, a city may change considerably in size, status, even change its name over the centuries etc. Also, identity is often contextual. Two drugs may be considered identical because they contain the same ingredients but may be considered different because they carry different names, coming from different producers. Two cars of the same type may be considered identical to a car insurers classification but different in actual insurance contracts. While temporal aspects have mostly been treated as part of modelling and annotation strategies, covered below in Sections 4 and 9, the latter aspect of identity 'flux' has been given quite some attention w.r.t. to identification disambiguation as will be discussed in Section 5. But before such approaches can be investigated in detail, the peculiarities of identification on the Semantic Web have to be explored. We cover this topic in some breadth as it nicely illustrates the kinds of competing goals, even opposing interests that a design has to take into account and reconcile to be sound in theory but also successfully usable in practice.

2.1.1 Access & Reference

> *The HTTP spec did not even specify whether the URI denoted a person or a document about them, it just explained that the thing returned [was a] representation of the resource. [...] for one reason or another people demanded the right to be able to use `http://example.net/people/Pat` to denote Pat rather than a web page about Pat. [...]*
> *It is also very expensive to go on debating it as though it is an open issue.*
>
> Tim Berners-Lee[1]

As Berners-Lee recounts the first designs of the Semantic Web already aimed at employing the identification mechanism of the World Wide Web to identify anything whatsoever, not just web accessible resources.[2] Many applications in the early days like e.g. the Dublin Core initiative focused on so-called *metadata* about web resources (e.g. author, date of creation etc.) but the design already

[1]In [Ber09], a post to his blog 'Design Issues'

[2][HP09] provides a compact overview on the evolution of the definition of a 'resource' in IETF RFC's, from Roy Fielding's first HTTP RFC over RFC 2396 to RFC 3986 which states that "This specification does not limit the scope of what might be a resource; rather, the term 'resource' is used in a general sense for whatever might be identified by a URI. Familiar examples include an electronic document, an image, a source of information with a consistent purpose (e.g., 'today's weather report for Los Angeles'), a service (e.g., an HTTP-to-SMS gateway), and a collection of other resources. A resource is not necessarily accessible via the Internet; e.g., human beings, corporations, and bound books in a library can also be resources. Likewise, abstract concepts can be resources, such as the operators and operands of a mathematical equation, the types of a relationship (e.g., 'parent' or 'employee'), or numeric values (e.g., zero, one, and infinity)."

had a much broader goal. The IRI based identification mechanism however ran into a problem: as per the Axioms of Web Architecture[3] an IRI can refer to one resource only. On the web this makes perfect sense and will never cause any problems, quite to the contrary. On the Semantic Web however IRIs are used to refer to web pages but also to things that those web pages may be about. In the latter case the web pages are used to refer to their meaning. Those are two very different things to refer to — the web page itself, and what it is about — but in this design they have the same IRI. This obviously not only breaks an axiom of web architecture but can lead to quite unexpected results. In the words of [Ber02]: "Before I recommend the Vietnam War page, I have to be careful I am not recommending the Vietnam War."

[HH08] characterizes this basic dichotomy as one between *access* and *reference*: using an IRI to access an information resource on the web or to refer to a non-information resource that is described by the web resource retrievable from that IRI. On closer inspection there a few more facets to take into account. An IRI addressing a web resource doesn't always return the same set of bits. When a user client like a web browser contacts a web server, behind the scenes a process called content negotiation is established in which client and server exchange details like e.g. available and desired language versions, graphics options, screen reading preferences etc. Different clients may thus get different representations from the same IRI, one an English language version with high-res PNG graphics, another one a mobile-friendly Spanish version. The representations delivered may even change over time, but on the bottom line all users get roughly the same content — or otherwise a basic expectation by the user would have been thwarted and the web would not have worked as intended. So while addressing a web resource via an IRI is not as simple as it may seem, it is still a relatively straightforward process with little risk of misunderstanding, and it works very well and reliably on the web. Using an IRI to refer to something else on the other hand is a much riskier endeavour. As [Ber02] puts it: "You can argue that people say 'I work for w3.org' or '`http://www.amazon.com/shrdlu?asin=314159265359` is a great book', just as they happily say 'Moby Dick weighs over three thousand tons', 'Moby Dick was finished over a century ago' and 'I left Moby Dick on the beach' without expecting to be misunderstood." As another example, references to Paris as a European city or a political symbol in May 1968, as the capital of France or a certain administrative body, as a moniker for the Paris Climate Accord or the 2024 Olympic Games may all be using an IRI like `<https://www.paris.fr/>`. Obviously this side of the access/reference dichotomy is considerably more messy and under-defined than the information resource side that underlies the World Wide Web.

[Ber02] again argues that "we won't use human language as a guide when defining unambiguously the question of what a URI identifies. If we want to do that on the Semantic Web, we will say 'I work for the organization whose home page is `http://www.ww3.org`'." Writing this out in Turtle reveals that this would

[3][Ber96b]

actually boil down to minting a new blank node for a non-information resource:[4]

```
:TimBL :workingFor  [
    a :Organization ;
:homepage <https://www.w3.org/>
] .
```

To a certain degree this would indeed solve the problem, but in a rather non-obvious way and only at the expense of adding one more layer of indirection to the already complex and contentious topic of blank nodes (discussed in detail below in Section 2.5.2). It especially doesn't provide an easily shareable identifier for the organization itself, falling short on an important requirement when integrating information on the Semantic Web. Other proposals have been put forward trying to square the circle, and distinguish the use of an IRI as addressing a web resource vs. referring to a resource external to the web, without breaking the axiom of web architecture that an IRI can only refer to one and only one resource. [Ber09] mentions the first convention put in place to this effect: adding a hash sign to an IRI results in the IRI not referring to the web resource itself but to the thing described by that web resource. This convention reused the fragment identifier from the IRI specification in a hacky, but not inelegant way. However, for various reasons — people didn't get the significance of the problem, insisted on using IRIs without the 'ugly' hash, claimed that the Semantic Web was about things and not web pages, used hashes for other (and arguably prior) purposes etc. — it wasn't followed with enough consequence in practice to be a reliable solution: people started to omit the hash from an IRI but still intended to refer to a thing itself, not the web resource so addressed. Since the definition of what an IRI refers to wasn't very precise in the HTTP specification itself the RDF community resorted to the Technical Architecture Group (TAG) of the W3C for a resolution to the problem. The TAG after much debate proposed a solution that in effect solved little but created practical problems of its own.[5] In the words of [Ber09]: "The spec wasn't changed. The spec editors were not brought on board to the new model. The spec was interpreted. The TAG negotiated in a way a truce between the existing HTTP spec, RDF systems, and people who wanted to use HTTP URIs without '#' to identify people. That truce was HTTPRange-14, which said that you don't a priori know that a hash-less HTTP URI denoted a document, but if the server responded with a 200 then you did, and you had a representation of the document. If you did a get on one of these new URIs which identified things were not documents (people, RDF properties, classes, etc) them the server must not return 200, it can return 303 pointing to a document which explains more." That solution has a few problems: it is opaque to a user that doesn't check HTTP response codes (which would be quite many), it is not

[4]Examples in this work will always be encoded as Turtle (`https://www.w3.org/TR/turtle/`) or, when graphs are used, TriG (`https://www.w3.org/TR/trig/`) — however in a slightly simplified form: namespace declarations will generally be omitted if not absolutely necessary as the readers are assumed to be knowledgeable about the usual meaning of prefixes like `rdf:`, `ex:` and `:`.

[5]See `https://lists.w3.org/Archives/Public/www-tag/2005Jun/0039.html` for the full TAG resolution on [httpRange-14], as the problem became known following the TAG nomenclature

reflected in any RDF serialisation but hidden in server configurations, server configurations are not easily accessible to lay users of semantic web technologies and, last not least, it still doesn't provide a way to definitely express that an IRI is meant to refer to a thing outside the web. According to the TAG resolution a 303 response merely indicates that 'the resource identified by that URI could be any resource'. The W3C published a Note on 'Cool URIs', [SC08], propagating both the hash and the 303 approaches whereas [HH08] and also [HP09] provide in-depth critiques of their practical shortcomings.

[Ber02] discusses more positions on what an IRI may mean, who is to decide on that meaning and how it might be expressed. They range from suggestions to the effect that the difference could be safely ignored[6] over impositions that every IRI on the Semantic Web is referring to things outside the web (and never to web resources themselves) and on to different proposals how other means like the publisher of the resource, the property used, the context of use or some external heuristic mechanisms would or might resolve the ambiguities of identification.[7]

Some of these proposals do approach naming and identification as inherently ambiguous processes in natural language and argue in favour of equally powerful identification mechanisms in RDF. [Ber02] however points out how such mechanisms, if designed too simplistically, are unsuited to less straightforward cases like e.g. a web page referring to another web page that is the subject of a third web page. In the end they will require a redesign that makes them obsolete in the first place, exhibiting the so-called *2-level syndrome*. [Ber02] also discusses an approach that delegates identification to context: "A catalogue card will talk about a document. A car catalogue will talk about a car." This is indeed the way that Semantic Web applications work in practice and the reason that this works might well be the reason why the whole endeavour still hasn't collapsed into a pitiful puff of smoke. It does however break the axiom that an IRI must identify *one* thing and, in practice, it doesn't allow for applications to speak about things *and* the web resources about those things, or about Moby Dick the whale *and* the novel *and* some hard copy *and* the first edition etc. within the same application context. This is rather unsatisfactory for an information system that claims to enable integration of any statements made by anybody about any subject.

Popular Semantic Web-based Knowledge Graphs seem to have settled for an approach that bites the bullet of having to provide different identifiers for things and the web pages that represent those things. As [Hog+20] notes, "in the RDF representation of the Wikidata [. . .] while the URL `https://www.wikidata.org/wiki/Q2887` refers to a web page that can be loaded in a browser providing human-readable meta-data about Santiago, the IRI `http://www.wikidata.org/`

[6]But, as [Ber02] comments, "now that we – for the first time – have not only internet protocols which can talk about document but also those which talk about real world things, we must either distinguish or be hopelessly fuzzy."

[7]Tennison for example discusses a 'punning' approach in which properties determine the identification semantics of subject and object, see `https://web.archive.org/web/20130130000809/http://www.jenitennison.com/blog/node/170`

`entity/Q2887` refers to the city itself. Distinguishing the identifiers for both resources (the web page and the city itself) avoids naming clashes." A third identifier, `https://www.wikidata.org/wiki/Special:EntityData/Q2887.rdf`, returns an RDF graph. DBpedia[8] follows a similar pattern as witnessed by the IRIs `https://dbpedia.org/page/Santiago`, `http://dbpedia.org/resource/Santiago` and `http://dbpedia.org/data/Santiago`. However, questions like what a reference to an entity/resource like 'Paris' or 'Moby Dick' refer to exactly remain unaddressed by this approach.

2.1.2 Unique Identification & Referential Ambiguity

> *[W]e are not analyzing a world, we are building it. We are not experimental philosophers, we are philosophical engineers. We declare 'this is the protocol'. When people break the protocol, we lament, sue, and so on. But they tend to stick to it because we show that the system has very interesting and useful properties.*
>
> Tim Berners-Lee[9]

> *What might indeed be true is that in many circumstances, a URI somehow provides access to information which is sufficient to enable someone or something to uniquely identify a particular thing (that the representation accessed via that URI is in some sense about), but even there the thing identified might vary between contexts (such as when we use someones email address to refer to the person) without harm. This kind of ambiguity resolved by context is at the very basis of human communication: it works in human life, it works on the Web, it will work on the semantic Web. [. . .]*
> *I'm not saying that the 'unique identification' condition is an unattainable ideal: I'm saying that it doesn't make sense, that it isn't true, and that it could not possibly be true. I'm saying that it is* crazy.
>
> Pat Hayes[10]

The introducing quotes stem from an eMail thread on the W3C Technical Architecture Group (TAG) mailinglist in which Tim Berners-Lee and Pat Hayes discuss an essential conflict on how identification and naming on the Semantic Web should be designed.

Berners-Lee insists on the axiom of web architecture that an IRI identifies exactly one thing — a principle that makes complete sense and works well on the web where identification and access fall into one. The Semantic Web however is concerned with any resources whatsoever, not just those that can be accessed on

[8][Leh+15]

[9]In a post to the Semantic Web mailing list, `https://lists.w3.org/Archives/Public/www-tag/2003Jul/0158.html`, in conversation with Pat Hayes

[10]In a post to the Semantic Web mailing list, urlhttps://lists.w3.org/Archives/Public/www-tag/2003Jul/0198.html, in reply to Tim Berners-Lee's post above

the web and Hayes elaborates that for resources in this more general sense this is not how identification in natural language works. The impossibility to look into each other's head makes identification of resources — connecting names with meaning — a process of communication, approximation and ultimately always ambiguous no matter how hard we try (and sometimes even getting ever more ambiguous the more detail we add). Hayes also points out the common practice of overloading terms with more than one meaning as a way to economize on communication by contextualization.

Berners-Lee counters with the argument that the Semantic Web is something new, and we can build it the way we want, especially that we are not bound by prior conventions and don't have to mimic natural language: 'We are building a new system. We can design it differently from existing linguistic systems. Toto, we are not in Kansas any more.'[11] He is convinced that the usefulness of the system will be pervasive enough for people to accept and play by its naming conventions. Hayes casts reservations that human language is stronger than specifications. "Reference is by nature ambiguous in any language. So any attempts by Web architecture to make reference completely unambiguous will fail on the Web."[12] He concedes that within usual (application) contexts commonly shared understanding is at least most of the time strong enough to overcome ambiguity in language. However, a formalism for naming on the Semantic Web has to take them into account, or it will fail to live up to practice.

At the heart of this conflict is the question of how much precision the Semantic Web needs and how much it can handle. Berners-Lee wants to keep the design simple and is prepared to trade expressive precision for immediate usability. What doesn't fit will be made fit by the sheer dynamics of usefulness. Hayes on the other hand argues that the concept of unique identifiers is too detached from theoretical and practical foundations of naming to provide a solid enough underpinning for the Semantic Web as a whole. The question might be: will the Semantic Web strive from simplicity and let a thousand applications bloom, or will it sink into a morass of 'garbage in, garbage out' when applications connect, and the hard cover edition of the whale that Alice bought at the antiquarian and took with her to the beach weighs 3000 tons and although it was created in 1851 it smells only just a bit, rather a result of the previous owner's smoking habits than of inappropriate cooling as one might expect? Some will say that the question has already been answered as almost two decades have gone by since that debate and the Semantic Web is alive and kicking. Others will counter that it still resembles an archipelago of sparsely connected data islands and in practice fails precisely on its central promise of universal integration and connectivity.[13]

[HH08] elaborate on the position that identification on the Semantic Web functions as either access or reference. They argue in great detail how reference is an inherently ambiguous process in natural language and that the Semantic Web can't avoid such ambiguity — not by decree and not even by some kind of

[11] `https://lists.w3.org/Archives/Public/www-tag/2003Jul/0158.html`
[12] [HH08]
[13] See e.g. [Asp+19]

lookup service or naming authority — as the meaning of terms is all too often dependent on context, on the sentences in which they are used. They derive their approach to identification from a reference to Russell's discrimination of reference 'by acquaintance or by description', arguing that reference by description as e.g. defined in Kripke's 'causal theory of naming' won't work on the Semantic Web as the participants are not linked closely enough to establish a chain of communication that would not rely on ... descriptions, which again are always and unavoidably ambiguous. To strengthen their point [HH08] refer to Frege's distinction of 'sense' and 'reference' which aims to capture the difference between what a term refers to in the intention of its user (as e.g. used in a sentence) and what it refers to in some wider setting. Frege's example is that of Phosphorus, the Morning Star and Hesperus, the Evening Star, that to ancient Greeks had different meanings while to modern speakers they both refer to the planet Venus.[14] As they put it, "names simply make no sense by themselves."

[HH08] then reflect on Davidson's insight that successful communication in any case has enormous prerequisite, "that disagreement and agreement alike are intelligible only against a background of massive agreement." That would be what Berners-Lee takes for granted or at least as desirable enough for people to incentivize them to make the necessary effort to establish such understanding by buying in conventions, shared vocabularies and generally a best-effort approach. Instead, they suggest that a more principled approach is needed to overcome in a sound and reliable way the impasse that the unique identification approach in their view represents. They argue for embracing ambiguity and overloading of terms as a natural and indeed very efficient feature of language instead of trying (or maybe, in fact, merely proclaiming) to eliminate it from the Semantic Web. To that end they propose the introduction of a small vocabulary to disambiguate the intended meaning of a term on demand, allowing to specify the intended identification semantics in sufficient detail when the need arises.[15]

[HP09] portrays the discussion as one between at least two distinct positions: the *direct reference* position and the *logicist* position. In the direct reference viewpoint, as championed by e.g. Berners-Lee, the referent of an IRI is some unique thing like the Eiffel Tower or the concept of unicorns, and the owner of the IRI establishes that referent by fiat. In the logicist position an IRI refers "to whatever model(s) - including actual things - that satisfy the formal semantics of the Semantic Web." In this view the meaning of an IRI is almost always ambiguous as no description is so precise that it can't be satisfied by many interpretations, like e.g. an IRI referring to Paris might be used without specifying if it means the metropolitan region or the city centre or even something slightly different like the 'atmosphere', and still make sense in the context of its usage. The Tarski-style formal semantics of RDF covers this ambiguity but collides with the direct reference position.

[14]Another example and for some reason more popular in Semantic Web circles is that of Superman and Clark Kent being different entities to Lois Lane while 'in reality' both referring to the extraterrestrial being Kal-El, see also Section 2.6.2.8

[15]Described in Section 7.2

Expanding on the philosophical backgrounds introduced by [HP09], [Hal12] adds considerable depth to those discussions, analysing how the arguments and intuitions brought forward by Berners-Lee and Hayes reflected well-known positions established in debates in the fields of philosophy and linguistics in the twentieth century, namely "the 'descriptivist theory of reference' attributed to early Wittgenstein, Carnap, Russell, and turned into its pure logical form by Tarski" driving Hayes' position, and "the 'causal theory of reference' championed by Kripke and extended by Putnam" reflecting Berners-Lee's direct reference position. After discussing how both positions ultimately fall short of providing a solid theoretical foundation for naming and identification on the Semantic Web he identifies Frege's idea of 'sense' — the "contention that the meaning of any representational term in a language is determined by what Frege calls the 'sense' of the sentences that use the term, rather than any direct reference of the term" — as a possible foundation for a proper solution to the problem of identification. [HHT15] sketch an approach along these lines using the concept of pollarding.[16]

2.1.3 Not Assuming Anything

In consequence the discussion didn't lead to a definitive resolution and the RDF Working Group punted on the issue: the RDF 1.0 Semantics "does not assume any particular relationship between the denotation of a URI reference and a document or Web resource which can be retrieved by using that URI reference in an HTTP transfer protocol, or any entity which is considered to be the source of such documents. Such a requirement could be added as a semantic extension, but the formal semantics described here makes no assumptions about any connection between the denotations of URI references and the uses of those URI references in other protocols."[17] [Hay12a] describes a semantic extension in which "[a] given IRI may have a different meaning when it occurs in a graph which inherits an extension, so an interpretation mapping has to be considered now as defined on occurrences of IRIs rather than on IRIs themselves." This was proposed for standardization to the RDF 1.1 Working Group as a semantics for named graphs — see Section 2.7.3 below — but not approved.

So nothing is decided — an IRI may be used to access or to refer — and no instruments are provided by RDF to disambiguate between the two options. Section 5 will discuss various studies and proposals to better understand and solve this problem at the very heart of the Semantic Web.

2.1.4 Web Names

The RDF 1.1 Semantics from 2014 made a change to the way the meaning of datatypes is defined. Whereas in RDF 1.0 the semantics of datatypes are specified as a 'datatype mapping', the later RDF 1.1 specification acknowledges that "IRI meanings may also be determined by other constraints external to the

[16] See below in Section 5.2.1
[17] [Hay04b], Section 1.2

RDF semantics [...] For example, the fact that the IRI `http://www.w3.org/2001/XMLSchema#decimal` is widely used as the name of a datatype described in the XML Schema document might be described by saying that the IRI identifies that datatype."[18] This change was made in recognition of the fact that "some datatype IRIs, like the XML schema datatypes, had achieved the status of universally recognized Web names, with fixed meanings ultimately specified socially, by their universal use, rather than by any RDF semantic rule."[19]

Identifying RDF in RDF The problem of proper identification in RDF is not constrained to references to the outside world but just as much affects the syntactic constructs of RDF themselves. Section 2.7 will discuss this w.r.t. to individual statements as well as graphs, and the topic will re-occur various times when discussing representational proposals in Section 4.

2.2 Data Model

Indeed, the data model behind RDF was significantly motivated by the fact that directed labelled graphs provided a simple, but effective model for aggregating data from different sources.

R.V. Guha[20]

The basic building blocks of RDF are triples, connected triples and graphs. Triples are well formalized in model and semantics. Graphs, although playing a veritable part in the model, are treated rather passingly in RDF 1.0. They get a good deal of discussion in RDF 1.1, but are still not properly formalized. The entities established by triples that are connected by sharing nodes, although ubiquitous in examples and any application of RDF, are not discussed at all in the standards specifications.

2.2.1 Triples

Triples are the central concept of RDF. Each triple consists of two nodes, subject and object, and a predicate (also called a property) that expresses a relation from subject to object. Predicates are always IRIs, ideally pointing to a web resource that describes their intended meaning and purpose. Subjects are either IRIs or so-called *blank nodes*, that serve mostly structural purposes and will be discussed later. The object can be either, as before, an IRI or a blank node or, exclusively in the object position, a literal value.[21] For literal values more

[18][HP14]

[19]Pat Hayes in private conversation with this author

[20][GMF04]

[21]The restriction that literals can only be used in object position stems from a technicality with the first serialisation format for RDF, RDF/XML (see `https://lists.w3.org/Archives/Public/semantic-web/2009Nov/0040.html`). At the time — the heyday of XML — RDF/XML was the predominant and even defining serialisation of RDF, so putting that restriction in place seemed justified. See also Section 2.4.1

facilities like language tags and several pre-defined datatypes are available.

Triples are the sole harbingers of truth in RDF. Asserting a triple is the same as making the claim that what it states is true. No triple is allowed to alter or even retract what another triple claims. Without getting too deep into semantics right away it should be clarified what it does mean to 'assert' something: asserting a triple amounts to making a statement that the relationship indicated by the predicate holds between (or: is true for) the things that the subject and object node refer to. 'To refer to' can also be understood as 'to mean': an IRI addresses a web resource that describes something. The essence of that description is the meaning of that web resource. In other words a triple states that what the subject node means is related to what the object node means by a relation that is identified by the predicate IRI (and probably, ideally, described by a web resource retrievable from that IRI). Actually it's even more complicated: IRIs in the subject and object position may either point to a web resource that expresses their meaning, as described before, or they are meant to identify that web resource itself. This subtle distinction will be further explored later.

This description is a bit wordy because it also introduces some aspects of the semantics that will be fully explained only later below but the gist of it is very straightforward: the relation between two nodes establishes a so-called *triple*, which simply states a fact and claims it to be true.

2.2.2 Graphs

The basic data model of RDF is that of a directed graph. An RDF graph is composed of a set of triples. Asserting a graph amounts to asserting all the triples it contains, e.g. claiming that all the statements expressed by those triples are in fact true.

The rather abstract, set-based semantics of graphs will be discussed in more detail in Section 2.6.2.5. RDF 1.1 introduced the notion of Datasets that may contain so-called *Named Graphs*: graphs associated with a name (a blank node or IRI). Those will be discussed in Section 2.7.3. The original proposal by [Car+05a] that coined the term Named Graphs, and many related works, will be discussed in Section 4.2.5. Section 2.4.4.3 disambiguates graphs from their sources and from named graphs.

2.3 Patterns

The expressiveness and scalability of the simple triple formalism is enormous. It requires very little upfront agreement and coordination, thereby "allowing anyone to make statements about any resource"[22] and integrate all those statements into one giant global graph. That at least is the idea.

A formalism this simple is bound to get into trouble when things get more complicated, and they do indeed in foremost two ways: integration of data from

[22][KC04]

disparate sources with no prior knowledge of each other will inevitably run into slight deviations and outright incompatibilities that require some construct to manage and resolve them. Those issues will be discussed later below but the most basic mechanism that RDF provides is grouping triples by source or any other criterion in graphs. Grouping triples in a graph is a very explicit way to construct an entity composed of multiple triples. Those triples don't need to share any nodes and in graph parlance establish a forest.

Before one might get into such fine troubles of integrating data, a much more immediate problem is that many subjects of interest are not represented by some single, easy to address node and most topics of interest are impossible to cover exhaustively by one relation from A to B. Instead such subjects and topics are themselves composed of a collection of facts that have to be expressed via multiple triples. Such collections will most probably share one or more nodes, establishing a more tightly connected group than a mere set — akin rather to a tree than a forest.[23] They form a network of triples, an interconnected graph of facts that are all related to each other by the nodes they share. As already mentioned such entities are not discussed in the core RDF specifications as an issue of some importance but rather taken as an obvious artefact. They do get a name in the SPARQL specification as Basic Graph Patterns (BGP).[24] The literature[25] distinguishes four types of BGPs: star, snowflake, linear and complex.[26]

2.3.1 Star Pattern

Star shaped BGPs are simply a node with some attributes, as in the following example:

```
:Alice :likes [
    a :Car ;
    :color :Red
] .
```

2.3.2 Snowflake Pattern

In their simpler form snowflake BGPs resemble star BGPs that spread out more than one level deep, as in the following example:

[23]Not necessarily a tree in graph parlance as they may have multiple root nodes

[24]`https://www.w3.org/TR/sparql11-query/#BasicGraphPatterns`

[25]E.g. [Khe+19], [Sch+16]

[26]These terms are reused from other database related fields and have slightly different meanings there, see e.g. [KR02]. This classification of modelling primitives is not the only one used in the literature: other works for example distinguish multipart objects, n-ary relations and arrays, but re-using established terms from database research seems more appropriate and, despite some slight variations in meaning, less ambiguous.

```
:Alice :likes [
    a :Car ;
    :color :Red ;
    :condition [
        :mileage 20000 ;
        :tires :Good
    ]
] .
```

Things can get more complicated as not every branching node is required to be a blank node (although that makes algorithmic detection of snowflakes much easier) and the shape is not required to be a tree but may have multiple root nodes:

```
:Alice :likes :Item_123 .
:Item_123 a :Car ;
    :color :Red ;
    :condition [
        :mileage 20000 ;
        :tires :Good
    ] .
:Bob :likes :Item_123 .
```

2.3.3 Linear Pattern

Linear BGPs are usually interpreted like linked lists in programming languages, e.g.:

```
:Alice :knows :Bob .
:Bob :knows :Carol .
:Carol :watches :TV .
```

They may also be constrained to using only one property, as property paths. Some conceptualizations favour to interpret them even more strictly as ordered lists or, less strict but still more constrained, as just lists, ordered or not.

2.3.4 Complex Pattern

A complex BGP is very much the same as all the above BGPs combined in a graph. It is not always regarded as a BGP itself but completes the abstraction in a useful way.

2.3.5 Existential Pattern

It is worth noting that at the core of most of these patterns sits a blank node. The Turtle syntax can elegantly hide this node with square brackets as long as the structure is tree-like:

```
[
    a :Car;
    :color :Red
] .
```

This modelling style favours integration of data as long as one object's properties are a subset of the other object's properties (and automatic merging of objects with overlapping sets of properties would be too risky anyway). It also is logically sound: the existential in question is defined by its properties only and therefore no issues w.r.t. non-monotonic qualifications can arise. However, as will be discussed in Section 2.5.2, this elegance comes at some cost.

2.4 Vocabulary

RDF enthusiasts like to think of graphs as the structure to structure them all because they provide unparalleled flexibility. That flexibility however comes at a price: the pre-structuring inherent to lists, trees and tables is not only a constraint but in practice often a feature. Those more rigid structures convey a lot of semantics implicitly, thereby making the life of authors as well as readers easier. RDF on the other hand has a hard time making such semantics explicit when it makes sense without getting overly verbose when it doesn't.

RDF provides special vocabulary for n-ary relations, lists and reification. However, these vocabularies all come with less expressive power than one might expect and hope for, making them the source of many frustrations especially to novice users. They also come without formal semantics, restricting their usefulness for advanced applications.

N-ary relations get a little help from `rdf:value` but not as much as the importance and ubiquity of n-ary relations would suggest. The RDF standard reification vocabulary is exceedingly verbose and its semantics is neither normative nor complete. The constraints imposed on lists by the open world semantics of RDF don't allow them to meet the expectations of developers accustomed to the closed world semantics of stand-alone applications and relational databases — or just about any programming language, for that matter.

2.4.1 rdf:value

RDF introduces the `rdf:value` property to emphasize the main component of a structured value. The RDFS specification gives as example a composite measurement value where the `rdf:value` property has as object the actual value, an integer, and is accompanied by a further statement explicating the unit of measurement — a somehow additional but still essential part of the information.[27] However, needing a blank node and two triples just to encode a measurement and its unit may seem like quite an effort for a rather mundane task. A proposal to implement structured values instead as multi-value datatypes is discussed below in Section 2.5.1.

Measurements are not a niche application. As will be discussed in Section 4.1.1 many complex n-ary relations are composed of IRIs as main values and additional,

[27] RDF Schema 1.1, [BGM14], Section 5.4.3 'rdf:value', `https://www.w3.org/TR/2014/REC-rdf-schema-20140225/#ch_value`

secondary attributes. The range of `rdf:value` is defined as `rdfs:Resource`, so it can well be used in such cases. In practice however the usage of `rdf:value` has mostly been limited to literal values. It became a rather marginal term in the RDF vocabulary, with no formal semantics and only rudimentary documentation in an obscure corner of the RDF 1.1 specifications.[28] Even the W3C Working Group Note on Defining N-ary Relations on the Semantic Web [Noy+06] only mentions it in passing in a footnote. Some vocabularies provide alternatives as [Lin22] discusses, but none of those captures the semantics of a primary value in an n-ary relation quite exactly or has become a de facto standard in this context — see Section 7.3 for further discussion.

Another application of `rdf:value` is making assertions about literal values 'themselves', e.g. if they are resources on their own right like an address, a poem, a citation etc. As RDF doesn't allow literal values in subject position this requires an indirection: introducing a blank node to stand in for the literal, adding to that blank node the literal via a `rdf:value` property and then adding assertions with the blank node as subject.

2.4.2 Lists

It is very hard to query RDF lists, using standard SPARQL, while returning item ordering. This inability to conveniently handle such a basic data construct seems brain-dead to developers who have grown to take lists for granted.

David Booth[29]

RDF does not have ANY data structures in it (except maybe triples). It describes data structures, just like it describes everything else. It does not PROVIDE data structures. Maybe it should - make the case! - but then it will need to change rather drastically in its very foundation. Triples describing lists are not the same as triples-plus-lists.

Pat Hayes[30]

The fact that SHACL is seen as a syntactic constraint, rather than just another description, is touted as a big advantage of SHACL over OWL. Sounds like just what we need here.[. . .]
OWL/RDF parsers have been in this position, doing OWL syntax checks on chunks of RDF, for over a decade. And the RDF spec

[28]It was covered in much greater detail in the RDF 1.0 Primer. Dan Brickley in 2010 provided a summary of the history of `rdf:value`, see `https://lists.w3.org/Archives/Public/semantic-web/2010Jul/0252.html`

[29]From his post "Towards easier RDF: a proposal", starting an epic thread on the Semantic Web mailing list, November 2018 `https://lists.w3.org/Archives/Public/semantic-web/2018Nov/0036.html`

[30]In a post to the Semantic Web mailing list, June 2010, `https://lists.w3.org/Archives/Public/semantic-web/2010Jun/0192.html`

does say explicitly that a semantic extension can impose syntactic conditions on RDF graphs (and keeping list descriptions well-formed was exactly what I had in mind).

Pat Hayes[31]

Lists are easily the most ubiquitous data structure in computing. Churning relentlessly through lists of data is what computers excel at. Lists are also very versatile and can be combined to the most relevant data structures: nested lists form trees, lists of lists produce tables and even graphs are commonly implemented and serialized as lists of nodes and their relations. Given their central importance most programming languages offer a range of different types of lists optimized for different application behaviours like e.g. first-in-first-out (FIFO) lists or lists optimized for random access to any member.

RDF however is not a programming language. Besides basic datatypes it provides no complex data type primitives and does little to accommodate expectations of programmers. RDF can only *describe* data structures. It does indeed offer several ways to describe lists but because of its specific, web-oriented semantics it can't enforce any syntactic constraints on those constructs. In particular, it can't give any guarantees on well-formedness.

A description of a list is necessarily composed of multiple statements. Such statements have to obey certain rules to form a syntactically valid list. Usual constraints are that they can have only one start and one end, come one after the other, are not allowed to branch etc. On the open web there is no way to enforce such rules. Anybody can publish an ill-formed list, anybody can add a statement to a well-formed list that would break well-formedness requirements. In the strictly monotonic logic of RDF however no statement is allowed to rule into another one (statements may of course happen to contradict each other, but that is then a conflict that RDF has no means to resolve). Consequently, RDF has no way to guarantee that a list follows all rules laid out in the list vocabulary definition. Applications however usually expect and would like to rely on such guarantees. The unusual behaviour of RDF, although well-founded in its web based approach to logic, can deeply frustrate such expectation, leading to outcries like "LISTS for fuck's sake!."[32]

2.4.2.1 Containers

The original list vocabulary provided by RDF is the `rdfs:Container` vocabulary. Three types of containers are defined to express common list semantics: `rdf:Seq` for ordered sequences, `rdf:Bag` for unordered sequences with possibly multiple instances of the same type (which is not possible inside the basic data structure of RDF, sets) and `rdf:Alt` for alternative entries like e.g. synonyms. Some

[31]In a post to the Semantic Web mailing list, July 2020, `https://lists.w3.org/Archives/Public/semantic-web/2020Jul/0138.html`

[32]Manu Sporny's rant on the unpleasant surprises that RDF has in stock for the unaware developer is definitely worth a read, at `http://manu.sporny.org/2014/json-ld-origins-2/`

more guidelines are given like that the first entry in a `rdf:Alt` list should be the preferred one. However, all those semantics carry no formal weight whatsoever and no inferences can be drawn from them. Very much in the logic of RDF there is also no way to express that a list is closed and complete as under the open world semantics anybody can add anything to it anytime.

2.4.2.2 Collections

A second list vocabulary was defined to meet the requirements of OWL for a list that can be closed: the Collection vocabulary with `rdf:List` as the central class. This vocabulary is based on first-rest-ladders well known from the Lisp programming language and describes a list as an interconnected sequence of elements, explicitly closed by a `rdf:nil` element. This modelling style is very clear about the structure of a list but also very verbose and hard to write, read and query. Like containers it has no normative semantics and can't preclude any malformed deviations from the intended syntax. Even its closedness, although explicitly indicated by the `rdf:nil` element, is only informal as any application can change the list element and add new elements before it.

2.4.2.3 Comparison

Both list styles have their pros and cons — some of them real, some perceived. Collections are said to "have more semantics", a perception that is grounded in their explicit end element but not so much in the actual semantics as specified. More importantly collections are well-supported through syntactic sugar in popular and rather user-friendly serialisations like N3, Turtle, TriG and JSON-LD. In those serialisations they easily beat the already concise Container vocabulary w.r.t. usability despite their verboseness under the hood. Using `rdf:Lists` instead of `rdfs:Containers` because of their easier syntax and on the way — informally — closing every list is however not really in the spirit of the open world semantics of the Semantic Web. When querying in SPARQL `rdfs:Containers` have the usability edge as SPARQL requires some quite involved query constructs to retrieve `rdf:Lists` in the original order.[33] Containers do fare better in that respect but have some usability issues either. [DMM19] compare both vocabularies in more detail and find that `rdfs:Containers` provide much better SPARQL performance than `rdf:List` collections in retrieval heavy scenarios. They also suggest that it should be relatively easy to considerably improve syntactic support for `rdfs:Container` in SPARQL.

[33] For example a pure SPARQL 1.1 query to retrieve the n-th element in an ordered list, while indeed possible though not exactly easy to formulate, will become computationally quite expensive even for medium-sized lists already. A programmatic solution will generally fare much better. See `https://stackoverflow.com/questions/17523804/is-it-possible-to-get-the-position-of-an-element-in-an-rdf-collection-in-sparql/17530689#17530689` for details

2.4.2.4 Semantic Extensions, Reasoning, Design Patterns

All remarks above about the semantics of lists in RDF are of course only valid in RDF on the open Semantic Web. RDF does provide a mechanism called Semantic Extension to implement tighter definitions that e.g. demand and enforce certain well-formedness constraints. Such constraints however require closing the world and therefore are only applicable in local, mostly application centric, scenarios.

OWL DL for example uses `rdf:List` collections in axiomatic statements to encode syntactic constructs like e.g. an intersection of classes. To that end it imposes certain well-formedness constraints like single first elements, single rest elements and termination. However, for technical reasons it can't allow the use of `rdf:List`s for any other application than describing an OWL ontology. For e.g. representing user space assertional data like a list of instances the use of `rdf:List` is off limits when OWL DL compatibility is desired. The `rdfs:Container` vocabulary on the other hand is allowed, but, as [Dru+06] puts it, "depends on lexical ordering and has no logical semantics accessible to a DL classifier."

Several design patterns have been introduced to model lists in OWL DL, focusing on different aspects and features. [Dru+06] discusses such an approach in detail; other examples include the Ordered List Ontology[34] and, from the Ontology Design Patterns portal, the List[35] and Sequence[36] patterns. One such pattern is also described in the Semantic Web Best Practice Working Group's note on n-ary relations by [Noy+06]. [CP14] explores possible improvements to list ontologies offered by new features introduced in OWL 2 DL. [Ebe+21] explore the reasoning properties of trees constructed from lists in OWL DL.

2.4.2.5 Lists as Native DataType

The discrepancy between developer expectations for efficient data structures and the hurdles put up by the strictly descriptive nature of RDF has led to proposals for the integration of a native list datatype in RDF, making lists a first class citizen on the Semantic Web. [LW10] present a proposal to make lists another node type in RDF alongside IRIs, blank nodes and literals, possibly accompanied by their own set of operators. [Las+21] discusses a literal value type for lists, accompanied by matching folding/unfolding-operators for processing such lists e.g. in SPARQL. N3 develops a mixed approach where the syntactic sugar known from Turtle is implemented as a proper datatype with its own methods (built-ins in N3 speak) but also mirrored in the `rdf:List` vocabulary.[37] In discussions on the next iteration of SPARQL, version 1.2, the topic has come up as well.[38] All

[34]`http://smiy.sourceforge.net/olo/spec/orderedlistontology.html`
[35]`http://ontologydesignpatterns.org/wiki/Submissions:List`
[36]`http://ontologydesignpatterns.org/wiki/Submissions:Sequence`
[37]`https://www.w3.org/TeamSubmission/n3/#lists`, `https://lists.w3.org/Archives/Public/semantic-web/2020May/0065.html`
[38]`https://github.com/w3c/sparql-12/issues/46`

proposals require resolving some semantic questions like the relation between a native list term and its mirroring representation as RDF triples or that between terms occurring within a list term and in the containing graph.[39] Last not least introducing a new complex datatype like lists requires updates across the whole RDF toolchain of specifications, syntaxes, parsers, databases, editors etc. Any older tools will not 'understand' the new datatype and treat it as as mere literal rather than a list. So, although RDF supports the creation of custom datatypes, their introduction is not a simple undertaking in practice.

For further reading on this topic see a thread on the Semantic Web mailinglist discussing the Ordered List Ontology and lists in RDF in general[40], a discussion about extending the Container vocabulary[41] or a recent thread about lists in OWL and as a native datatype.[42]

2.4.3 RDF Standard Reification

RDF provides a small vocabulary to reify statements by describing them with the help of other statements. It consists of three properties — `rdf:subject`, `rdf:predicate` and `rdf:object` — to describe the terms a statement is composed of. An additional class, `rdf:Statement`, serves to type the resource so described, being the fourth element in the so-called *reification quad*.

This vocabulary has been introduced with the first RDF standardization effort from 1999[43], but in 2004, when RDF 1.0 was standardized, its deprecation was already discussed as nobody was particularly happy with the verbosity of the approach. Although it is arguably the only sensible way to represent triples as triples, it is tedious to write and query and adds considerable bloat to a triple store if used extensively.

The subtleties of the model-theoretic semantics of RDF standard reification as specified in 2004 will be discussed in Section 2.7.2. For a review of the applications of the RDF reification vocabulary in practice see Section 4.2.4.1. Section 6 presents alternative serialisations and syntactic sugar to ease the pain induced by the RDF standard reification quad.

2.4.4 Nomenclature

This section briefly mentions some peculiarities about the RDF vocabulary that are helpful to be aware of.

2.4.4.1 RDF & RDFS Namespaces

The RDF core vocabulary is defined in two different namespaces, RDF and RDFS. This has historical reasons: both standards were created at about the same time

[39] `https://lists.w3.org/Archives/Public/semantic-web/2020May/0069.html`
[40] `https://lists.w3.org/Archives/Public/semantic-web/2010Jun/0168.html`
[41] `https://lists.w3.org/Archives/Public/semantic-web/2020Jun/0071.html`
[42] `https://lists.w3.org/Archives/Public/semantic-web/2022Aug/0011.html`
[43] `https://www.w3.org/TR/PR-rdf-syntax/#higherorder`

in the early 2000s, but by different working groups at the W3C. Those working groups had overlapping agendas but tackled them from different perspectives. When things started to settle down and coalesce it became apparent that it followed largely arbitrary lines if classes and properties had been defined in RDF or RDFS. For example the class of all containers, `rdfs:Container`, is defined in the RDFS namespace. Its subclasses for bags, sequences and alternatives are defined in the RDF namespace. Membership in such containers is declared via the member property, again defined in the RDFS namespace. It seems that there's only one thing that can be done about this mess: memorizing! It might be comforting to call to mind that some natural languages behave even less predictably.

2.4.4.2 Resources

On the web a resource is something that is addressed by an IRI, e.g. the document that is returned by a web server when requested by a web browser. On the Semantic Web however the term resource doesn't necessarily refer to the document itself but may also refer to what that document is about, what it *means* in the "universe of discourse." In other words: resources on the web are documents like HTML-files, or, more strictly speaking, syntactic artefacts. When speaking about the Semantic Web however, and specifically its semantics, the term resource is used to refer to the meaning of an IRI, its denotation in the realm of interpretation, e.g. the real world entity or concept that is described in a document retrievable from some IRI. [Ber09] gives a thorough account of how this confusing choice of terminology came into being. It may be unfortunate, but the Semantic Web is now stuck with it.[44]

2.4.4.3 Graphs, Sources & Named Graphs

RDF defines graphs in abstract mathematical terms: a graph is defined by the triples it contains. Adding, altering or taking away a triple results in a different graph no matter if the IRI at which it can be accessed remains unchanged. Likewise, just changing the name doesn't change the identity of the graph because its meaning is defined by the triples it contains (and those are unmodified by renaming). This concept is consistent but rather unintuitive in practice where graphs are often identified by and accessed through the name of a document or a named graph in a triple store that contains them. On the formal level however that name has no effect on the meaning of the graph.

RDF 1.1 added the notion of a 'source' to account for this mismatch between abstract definition and practical usage. A source is described as 'a persistent yet mutable source or container of RDF graphs'[45]. Sources are to be understood as resources — in the Semantic Web sense: belonging to the realm of interpretation,

[44][Ber09] reflects on replacing the term 'resource' by 'document' and 'thing' respectively, but one shouldn't hold one's breath for that to happen.

[45]RDF 1.1 Concepts and Abstract Syntax, [Cyg+14], Section 1.5 'RDF and Change over Time', `https://www.w3.org/TR/rdf11-concepts/#change-over-time`

the universe of discourse — and like all resources they can be named and referred to by an IRI. However, the term is little known and the rift between definition and intuitions largely remains. Both terms are not incorporated in the RDF vocabulary[46] and are only informally defined in the specification.

So called named graphs on the other hand as defined in SPARQL and RDF 1.1 may better reflect common intuitions but are purely syntactic constructs without a formal model-theoretic semantics. RDF doesn't provide a term to refer to them either; the RDF 1.1 Working Group Note 'On Semantics of RDF Datasets' [Zim14] instead points at some other vocabularies to describe named graphs and the datasets that contain them. That Note provides a detailed account of the difficulties the WG faced when trying to define precisely what a graph means — hence maybe also explaining why no proper vocabulary term was defined — and is the subject of Section 2.7.3 below.

2.5 Syntax

RDF provides some syntactic means to facilitate the modelling of structures that go beyond strictly binary node-to-node relations. There's the notorious blank node, a very versatile instrument, but also prone to misinterpretations, subtle semantic variations and irritating manifestations of logic-induced rigour. At the other end of the syntactic spectrum datatypes provide a little used extension mechanism to encode structured values more efficiently.

2.5.1 Datatypes

Datatypes are another vector to reduce complexity in RDF. Literal values in RDF can be tagged with a datatype. As the name suggests they specify the datatype of values, like if some number is an integer or if the string 'true' represents a boolean, plus a set of date and time related datatypes. RDF reuses the XML Schema built-in datatypes, and defines two additional datatypes, `rdf:HTML` and `rdf:XMLLiteral`.

RDF datatypes do not, however, support complex datatypes composed of value and unit, as is very common in measurement data. Some measurements consist of even more than two components — an example given by [Noy+06] is a fever reading composed of value and unit, but also incorporating a tendency of falling or rising temperature. Geographical data likewise regularly consists of multiple values: latitude, longitude and sometimes also elevation.

RDF offers the `rdf:value` property, discussed above in Section 2.4.1, to model such complex measurements. The `rdf:value` property has no normative semantics, but the advised usage is to introduce a blank node representing the measurement, attach the measurement value as e.g. an integer per `rdf:value` and add a second triple representing the measurement unit, e.g. centimetre or

[46]Providing such a term was considered by the RDF 1.1 Working Group but dropped eventually, see `https://www.w3.org/2011/rdf-wg/track/issues/38`

degree Fahrenheit. This approach is flexible and extensible, but it's also tedious to author and query and adds to triple proliferation.

Users may define their own datatypes but by consequence venture somewhat outside the area of RDF as standardized and commonly understood and will miss tool support e.g. in SPARQL. See Section 4.2.1 for proposals on how to extend the RDF datatype mechanism. For a discussion of the virtues of a native List datatype see Section 2.4.2.5 above.

2.5.2 Blank Nodes

They are an important convenience for RDF authors, but they cause insidious downstream complications. They have subtle, confusing semantics. (As Nathan Rixham once aptly put it, a blank node is 'a name that is not a name'.) Blank nodes are special second-class citizens in RDF. They cannot be used as predicates, and they are not stable identifiers. A blank node label cannot be used in a follow-up SPARQL query to refer to the same node, which is justifiably viewed as completely broken by RDF newbies.

David Booth[47]

This is just a bit of nested structure in the language, which is valuable, understandable and no cause for alarm. [. . .] it is a thing which has no URI. A little less hysteria over blank nodes may be in order.

Tim Berners-Lee[48]

Blank nodes — colloquially also known as 'bnodes' — seem to be a very simple concept: a node that allows a binary statement to branch out into a subtree structure but that doesn't have, and doesn't need to have, a proper name — a little drop of glue to hold stuff together, a transparent plastic bag, a connector of some sort but without any identity of its own. Such a device should come in handy, even prove indispensable, for a minimalistic design like the RDF triple. And indeed, as [Hog+14] found, blank nodes are used in about 25% of RDF statements on the Semantic Web. But despite their ubiquity and straightforward purpose they have spurred quite some misconceptions, frustrations and serious calls for outright abolishment. They represent one of the issues of the Semantic Web that has always been, and still is, good for a heated debate on the communities' mailing lists — see for example the epic thread on the Semantic Web's main mailing list in late 2018 from which the introducing quotes stem.

[47]In "Towards easier RDF: a proposal", a post to the Semantic Web mailing list `https://lists.w3.org/Archives/Public/semantic-web/2018Nov/0036.html`

[48]In reply to David Booth's mail above, `https://lists.w3.org/Archives/Public/semantic-web/2018Nov/0053.html`

2.5.2.1 Lists & Logic

> *1) bnodes are a trick to avoid thinking about useful names in situations you do not really care about them and used f.e. in implementing lists in RDF. Obviously they were not really needed but make life easier.*
> *2) Logicians entered the place and started to interpret them as existential quantified variables. This is not wrong (since they are statements about something that exists and has a certain property), however, it is a somehow heavy way to interpret a simple syntactical short-cut.*
>
> Dieter Fensel[49]

Blank nodes in RDF are used for two very different purposes: they serve to encode structural detail of complex objects like lists or n-ary relations, but they also stand in for subjects of discourse which don't have a name. Put another way: blank nodes are used to represent either anonymous structural individuals or concept-level existentials.

Anonymous structural individuals represent things that we'd rather not be bothered to name explicitly — structural elements whose name is not unknown but rather not deemed important or significant enough to make explicit. Blank nodes abound in structures like lists, composite value types — composed of measurement value and measurement unit — or n-ary relations and tree-shaped nested objects.

Concept-level existentials on the other hand represent something that doesn't have a name but is of a certain importance. Maybe the name isn't known just yet, like with a so far unidentified person. Maybe the thing usually has no name on its own, like a post address. Maybe the object is of rather transient nature, like an event in a stream. It may even be that the description fits multiple objects — the precise meaning of a blank node is not "(exactly) one" but "at least one".

In the following, blank nodes as originally designed and intended — a means to add structure to a graph — will be introduced first, turning then to their formalization in the model-theoretic semantics of RDF. After thus setting the stage the focus will be on how their intended purpose and their formal specification interact, creating both benefits and tensions in practice, and what proposals have been made to ease the tension between formalism and pragmatics.

Blank nodes have been investigated in quite some breadth and detail. [Mal+11] provide a comprehensive overview over blank nodes in theory and practice, presenting their foundation in logic and implementation in various relevant standards, collecting results for computational complexity, analysing their use in published data, polling developers and exploring alternative treatments of blank nodes. That work is extended by [Hog+14] with an even broader scope,

[49]In a post to the Semantic Web mailing list, `https://lists.w3.org/Archives/Public/semantic-web/2011Mar/0299.html`

e.g. investigating the relation between blank nodes and incomplete information in database theory, and by giving an in almost every respect more detailed account of their treatment in W3C standards and their usage in a large sample of data on the web. They also analyse in depth the occurrence of non-lean blank nodes in real world data and their implication w.r.t. to data merging. [Che+12] investigate the ways in which blank nodes enable various modelling primitives in RDF and propose steps to reduce the semantic mismatch between blank nodes in RDF and SPARQL.

As witnessed by the introducing citations, as of today the topic is not quite settled and some loose ends that these authors analyse still plague common understanding and usage of blank nodes in practice.

2.5.2.2 Structural Stopgaps

The naturalness of lists and tables that is usually taken for granted is not available in the graph structure that RDF is based on. Every list entry needs a slot in a list and the list itself and every such slot has to be explicitly created in a graph. Complex objects are often best encoded in tree-like structures of nested lists that require even more effort. Representing tables in graphs again follows the same principle. It is quite tedious and counter-intuitive to define a proper name for every such entry in a list or cell in a table. Blank nodes can spare one that effort and the verbosity that comes with it. In an appropriate surface syntax like Turtle those blank nodes can be replaced by brackets that elegantly convey the structural information without needing explicit identifiers of any kind — at least as long as the underlying object is of tree-like shape and not a true graph, but that's quite often the case.

Another popular application of blank nodes are entities that have no proper natural identifier. A postal address for example has a range of properties like street and number, town, postcode etc. However, in natural language it usually has no name of its own; it is just the address of some person or institution and is defined by that relation. So in RDF it is quite intuitive to model the identifier of the address itself as a blank node connected via some `:hasAddress` property to the addressee and some other relations describing its actual properties like street, town etc. It would feel unnatural and tedious to mint a proper name for the address itself; a blank node is good enough.

Last, not least, important modelling primitives like the snowflake pattern 2.3.2 in practice heavily rely on blank nodes. Here however they not only provide some structural convenience: without their existential semantics it would be much harder to form complex propositions from monotonic statements — a topic that we'll investigate in more detail in Section 2.6 below.

2.5.2.3 Existential Quantifiers

The RDF specifications cover blank nodes in various places, although nowhere in a quite exhaustive fashion. Some possible pitfalls are mentioned, some are hinted at, but crucial details are easy to miss if one is not expecting them anyway.

> *It is sometimes handy to be able to talk about resources without bothering to use a global identifier. For example, we might want to state that the Mona Lisa painting has in its background an unidentified tree which we know to be a cypress tree. A resource without a global identifier, such as the painting's cypress tree, can be represented in RDF by a blank node. Blank nodes are like simple variables in algebra: they represent some thing without saying what their value is. [...] They can be used to denote resources without explicitly naming them with an IRI.*
>
> RDF 1.1 Primer[50]

> *Blank nodes do not have identifiers in the RDF abstract syntax. The blank node identifiers introduced by some concrete syntaxes have only local scope and are purely an artefact of the serialisation.*
> *In situations where stronger identification is needed, systems MAY systematically replace some or all of the blank nodes in an RDF graph with IRIs. [...] This transformation does not appreciably change the meaning of an RDF graph [...].*
>
> RDF 1.1 Concepts and Abstract Syntax[51]

> *Blank nodes are treated as simply indicating the existence of a thing, without using an IRI to identify any particular thing. This is not the same as assuming that the blank node indicates an 'unknown' IRI. [...]*
> *Blank nodes themselves differ from other nodes in not being assigned a denotation by a simple interpretation, reflecting the intuition that they have no 'global' meaning. [...]*
> *RDF graphs can be viewed as conjunctions of simple atomic sentences in first-order logic, where blank nodes are free variables which are understood to be existential. [...] RDF syntax has no explicit variable-binding quantifiers, so the truth conditions for any RDF graph treat the free variables in that graph as existentially quantified in that graph.*
>
> RDF 1.1 Semantics[52]

The essential message is that a blank node represents '(at least one) something' without giving it a name: 'something exists which has such and such properties' etc. and blank node identifiers in concrete syntaxes are ephemeral. [53] As the

[50][SR14], Section 3.4 'Blank nodes', `https://www.w3.org/TR/rdf11-primer/#section-blank-node`

[51][Cyg+14], Section 3.5 'Replacing Blank Nodes with IRIs', `https://www.w3.org/TR/rdf11-concepts/#section-skolemization`

[52][HP14], Section 5.1 'Blank nodes', `https://www.w3.org/TR/rdf11-mt/#blank-nodes`

[53]The tension between formal and real-world semantics shines through in the claim that systematically replacing blank nodes by IRIs 'does not appreciably change the meaning of an RDF graph' — that issue will be re-visited when discussing Skolemisation below.

quote from the RDF 1.1 Semantics document clarifies the RDF semantics is based on first-order logic (FOL) and blank nodes are defined analogously to the FOL construct of existential quantifiers. According to Wikipedia in FOL "an existential quantification is a type of quantifier, a logical constant which is interpreted as 'there exists', 'there is at least one', or 'for some'."[54]

What may throw off the non-logician, rather web-developing user is the fact that this semantics doesn't fix how many somethings there exist, except that it's at least one. This deviates from the concept of 'something' in natural language which may commonly be interpreted as one and only one specific entity, whereas multiple somethings are rather called 'some things', 'something' and 'something else', 'anything' etc. In the intuition of a not logically trained human author blank nodes may just as well represent existential objects in the operational sense, claiming the existence of exactly one such thing and the blank node standing in for it. One might think of them not as of mere existentials but as of ground individuals, albeit lacking a proper name. As will be discussed below such intuitions can painfully clash with the formal semantics of blank nodes in RDF.

In the opposite direction however the RDF semantics is very open: two different blank nodes — different empty circles in the graphical representations used in the RDF specifications or different individuals like _:b1, _:b2... in the popular Turtle serialisation — may very well refer to the same 'something' out there in the world so described.[55] RDF guarantees that each blank node refers to not more than one real world entity, but it doesn't guarantee that each blank node refers to a different entity. The latter interpretation may be drawn from the properties assigned to a blank node — the blank node itself is agnostic to such differentiations and doesn't convey them.

Leaning in Theory If multiple blank nodes have the same properties, the RDF semantics treats them as referring to the same entity and making the same assertion about it. Consequently, they can be unified into a single blank node. For example, stating that 'I have something and something.' may be reduced to just 'I have something.', stripping the graph of one blank node in the process. This process in RDF is called `leaning`[56], also known as unification in logic. Some examples may illustrate the concept more clearly. Take the following two statements:

```
:Bob :has _:b1 ,
          _:b2 .
```

Those two statements are not lean as they are semantically equivalent to:

```
:Bob :has _:b3 .
```

[54] `https://en.wikipedia.org/wiki/Existential_quantification`

[55] This follows from RDF's so-called *No Unique Name Assumption (NUNA)* that will be discussed in a later section.

[56] RDF 1.1 Semantics, [HP14], Section 4 'Notation and Terminology', `https://www.w3.org/TR/rdf11-mt/#notation-and-terminology`

The meaning of those blank nodes however changes when further information is added that differentiates them from one another:

```
:Bob :has _:b1 ,
          _:b2 .
_:b1 :is :Red .
_:b2 :is :Round .
```

That graph is lean as the two blank nodes can not be unified without changing their meaning. Likewise, the following graph:

```
:Bob :has _:b1 .
_:b1 :is :Red ,
         :Round .
:Bob :has _:b2 .
_:b2 :is :Red ,
         :Risky .
```

is lean. As a counter example, unifying _:b1 and _:b2 to _:b3:

```
:Bob :has _:b3 .
_:b3 :is :Red ,
         :Round ,
         :Risky .
```

would ignore that they have different sets of properties and would lead to a graph with a different meaning — it is not a meaning-preserving operation and therefore is not leaning.

Given however the following set of statements:

```
:Bob :has _:b1 .
_:b1 :is :Blue ,
         :Big .
:Bob :has _:b2 .
_:b2 :is :Blue .
```

leaning to

```
:Bob :has _:b3 .
_:b3 :is :Blue ,
         :Big .
```

is possible as all information that _:b2 provides was already contained in the description of _:b1 and therefore _:b2 added nothing new by itself.

2.5.2.4 Theory & Practice

[Hog+14] derive some usage statistics on blank nodes from a large corpus of linked data. They observe that the vast majority of blank nodes form tree structures. Only about 2.3% of blank nodes appeared just as subjects but not as objects of some statement. Those might be candidates for minting of proper IRIs. An even more marginal group of 0.005% appeared only in object position, hinting at a possible use as operational pseudo-ground individuals. It seems that blank nodes are predominantly used not to stand in for subjects of discourse but to encode data structures — in line with their original purpose as a syntactic feature to facilitate the implementation of slightly more involved structures like lists and trees in the graph structure of RDF.

Results from a questionnaire published in the same work however suggest that the use of blank nodes as ground existentials could be much more prevalent than those numbers suggest. About 30% of the participants declare that they would be willing to encode a pair of unknown telephone numbers as two blank nodes[57]. Consequently, there seems to be little awareness that this would violate their existential semantics with at least a third of Semantic Web developers. As already mentioned above this intuition of blank nodes as ground individuals is indeed implemented in most parts of a typical RDF tool chain, probably most importantly in SPARQL engines.

Leaning in practice The RDF specification doesn't mandate leaning. Although fundamental to the concept of existential quantifiers, graph leaning is optional in RDF and in practice it is rather the exception than the norm. RDFS and OWL may unify blank nodes, but instead ground semantics — which treat each blank node as an individual — are often applied. SPARQL, even under the SPARQL 1.1 RDFS entailment regime, by default doesn't lean graphs. Inserting two triples that differ only by a blank node will result in the addition of two different triples. Likewise, counting those triples regularly relies on treating them as two distinct triples. [58]

As a rule of thumb applications generally treat blank nodes like nominals where different blank nodes are assumed to point to different entities whereas reasoning engines must follow the FOL semantics and unify bnodes as existentials.

This may seem irritating, inconsequential or even broken. Why is leaning not enforced in every RDF system on every step of the toolchain, despite the very clearly formalized semantics of blank nodes? One reason might be that people use RDF not predominantly for exchange of data over the Semantic Web but for authoring and in-house applications where more 'grounded' intuitions predominate. Another reason is probably that reasoning is less customary on the Semantic Web than its design suggests and even when exchanging data with unknown counterparts assuming grounded blank node semantics is a safer bet to ensure interoperability. A deep dive into the pre-2004 mailinglist archives of the W3C working groups on RDF and RDFS will probably unearth some enlightening discussions. An at least very plausible explanation seems to be that no one was convinced that a stronger commitment could be expected, enforced even, but on the other hand there was also no proposal on the table for a more intuitive formalization — at least nothing as well proven as FOL existentials.

Notably this divergence between theory and practice is nowhere formally

[57][Hog+14] Section 6.3, Table 4, Questions 2d and 2e

[58]The application of counting semantics in SPARQL is not specific to blank nodes. Whereas a graph in RDF is defined as a mathematical set and as such cannot contain multiple occurrences of the same triple, SPARQL implements an operational semantics of multisets (see [Ang+17]), triple stores usually eschew duplicate removal as computationally too expensive, and syntaxes are of course the most malleable link in this chain.

recognised and clearly documented in the RDF standards or even expressible in RDF itself. One does not only have to be aware of this issue but also resort to out-of-band means to communicate if ground or existential semantics are applied. Consequently, it takes a good understanding of the topic plus knowledge of the local leaning policy to successfully use blank nodes. Granted, as common usage of grounded 'counting' semantics is prevalent and awareness of the issue not well-developed one can most of the time assume that existential semantics are not followed. This however neither bodes well for exploiting the benefits of reasoning on the semantic web nor does it ease frustrations among the unassuming when the unthinkable happens and blank nodes are indeed treated as specified.

2.5.2.5 Dis/Integration

Integration Integrating data from different sources is one of the central value propositions of the Semantic Web. Blank nodes do not only ease authoring by providing a sort of self-identifying type of nodes, they can also facilitate aggregation of data from different sources. Unifying structural elements (represented by blank nodes) with the same properties is a good way to avoid clutter and verbosity. Ingesting the same list twice shouldn't result in a second list just because the blank nodes have different "names" the second time around. Likewise, versioning a list shouldn't result in a completely new list after editing one entry just because all the identifiers of all (structural) blank nodes got redefined in the process. Blank nodes elegantly enable these very desirable and practical features.

Merging data from different sources however is a notoriously difficult task and blank nodes provide no panacea. For example splitting a graph that contains some blank nodes into two and then merging the two resulting graphs into one again will not necessarily lead to the initial graph. One may e.g. split the graph:

```
:Bob :has [ :is :Small ,
            :is :Round ]
```

into the graphs

```
:Bob :has [ :is :Small ]
```

and

```
:Bob :has [ :is :Round ]
```

When merging those two graphs again one gets:

```
:Bob :has [ :is :Small ]
:Bob :has [ :is :Round ]
```

losing the information that Bob has something that is *both* small and round.

Note that RDF distinguishes `merge` and `union` operations over graphs with blank nodes. In a union operation the same blank node identifier in two different graphs is considered to refer to the same entity. This is useful when blank nodes are shared between graphs e.g. because they belong to the same application context and are stored in the same RDF dataset. The merge operation on the other hand would rename those blank nodes before smushing two sets of triples into one — honouring the fact that their name is purely a technical artefact, carries no meaning whatsoever and is not governed by any rule other than that renaming a blank node requires renaming all occurrences of that blank node in a graph. In general the merge operation is the safer way to go.

On a related note SPARQL engines that skolemise blank nodes internally to enable round-tripping (described below) may fail to lean data that is loaded multiple times.

Linking When data sources are not merged but linked to from an external source, blank nodes become problematic as they have only local scope and can't be referenced from outside a graph. For this reason their use has been discouraged in the context of Linked Data. [HB11] advie that "[t]he scope of blank nodes is limited to the document in which they appear, [...] reducing the potential for interlinking between different Linked Data sources. [...] it becomes much more difficult to merge data from different sources when blank nodes are used [...] Therefore, all resources in a data set should be named using URI references."

Encapsulation When striving for data encapsulation the locality of blank nodes can become a welcome feature. In OWL blank nodes are used to 'close' descriptions within a local document. An OWL axiomatic list describing e.g. a union of classes cannot be extended by another document on the open web as the blank nodes that form its list structure are not referenceable from the outside. In a related approach to encapsulation an object is identified by a blank node and described through statement with that blank node as subject. References to that object will have to copy the full description, ensuring that the object is identified by its properties instead of by a name.

Boundaries Some algorithms aiming to consolidate complex information objects from an indiscriminate sea of triples follow all links from a node until they reach an IRI or literal value. In other words: as long as they reach a blank node they follow all links from that blank node. When used with such algorithms in mind — Section 4.3 provides an overview — modelling with blank nodes can help to bring structure to RDF data.

2.5.2.6 Querying

SPARQL treats blank nodes as existentials in queries. However, in query results it most of the time doesn't perform leaning, but instead treats blank nodes not as existentials but as ground individuals — reflecting the intuition that two blank nodes with the same properties and property values represent not the same thing but 'something' and 'something else'.

Counting Semantics Obviously the triple count of a graph may change as a direct consequence of a leaning operation and two graphs that the RDF semantics considers to be equivalent (under simple entailment) may give different results for COUNT. For that reason not leaning a graph and treating blank nodes as ground individuals is sometimes called 'counting semantics'. Other features of SPARQL like NOT EXISTS and FILTER exhibit the same behaviour.

Round-Tripping The SPARQL specification doesn't demand that blank nodes returned in a query result can be reused in a follow-up query. This can lead to a quite frustrating developer experience as it makes it e.g. hard to incrementally refine queries or delete nodes fetched in previous queries. Many SPARQL implementations have therefore decided to enable this so-called *round-tripping* by internally skolemising blank nodes. [Mal+11] mention some SPARQL engines providing special blank node syntaxes to improve support for roundtripping. Another approach would be to more universally represent query results in a syntax like Turtle which hides blank nodes in braces[59].

SPARQL ASK However, SPARQL doesn't always eschew existential blank node semantics in query results. In answers to ASK queries it is obliged to match exactly what the semantics of the employed entailment regime dictates.[60]

2.5.2.7 Reasoning

Blank nodes are the reason why simple entailment in RDF is NP-complete. [Hog+14] however argue that although this is quite bad in theory, the amounts of blank nodes that occur in real-life data can easily be handled in practice and even nefarious attacks aren't hard to catch.

2.5.2.8 Patches, Evasions, Alternatives

Different approaches to limit the irritating effects of blank nodes have been proposed: from augmenting them with a more intuitive type of existential to restricting their scope and usage and up to replacing them by ground individuals altogether.

Syntactic Sugar One might wish that blank nodes would become completely invisible behind surface syntax like Turtle's [] and () braces, being 'just there' when no proper name is needed to address subjects of discourse. With the ubiquitous line-based serialisations this is however only possible as long as the serialized structures are at most tree-shaped. As soon as real graphs are concerned, where connections may go back and forth in non-hierarchical ways,

[59]"If the data is serialized as turtle, typically the blank nodes all appear as [ ] square brackets, so there is no blank node identifier which would cause a newbie to think they could query it.", Tim Berners-Lee, `https://lists.w3.org/Archives/Public/semantic-web/2018Nov/0053.html`

[60][Hog+14] Section 4.2 explain how to check lean-ness with SPARQL ASK and SELECT commands

some kind of explicit identifier becomes indispensable. This perception also oversees the importance of blank nodes not only for structural purposes but for modelling entities that have no proper identifier of their own.

Alternative Semantics The semantics of blank nodes in RDF is very conservative: it assumes less than one might expect. One might consider that 'weak' and call for stronger semantics like a differentiation between 'something' and 'something else'. On the plus side this semantics precludes little and gives implementors a lot of freedom in how they handle the specifics. On a highly decentralized information system like the Semantic Web this approach certainly has merit but some discussions have focused on possibilities to find a better balance between existential openness and intuitive ground semantics.

Hybrid Semantics As indicated above the ubiquitous SPARQL query engine and often also local applications employ ground semantics that treat each blank node as an individual constant. On the other hand managing versions of data either from different sources or from updates or versioning systems can profit heavily from existential semantics. Hybrid semantics strive to establish ways to consider blank nodes as ground within a local scope but existential across scopes. Merging does then require to distinguish non-lean local nodes from non-lean merged nodes, preserving the former but merging the latter. [Hog+14] argue that "such a semantics would be even more complicated and far less intuitive than the existing semantics."

Nominals Instead of changing the semantics of blank nodes they could also be augmented by a second type of existential that provides stronger naming semantics as has been proposed with nominal quantifiers[61]. Blank nodes here would continue to serve structural purposes. Nominals however would act like concept-level, grounded individuals that are not subject to unification, providing a more intuitive semantics to logically untrained users.

Scoping A graph can seem chaotic and prove hard to navigate when it contains lots of structural information woven into the fabric of data. A graph that models complex irregular objects or an involved structure like a relational table is bound to contain many blank nodes that only serve structural purposes. Those structures may also interlock, making it hard to realize where e.g. a n-ary relation ends and a list begins. Binding blank nodes more strongly to the structure they are part of might be a way to counter those effects.

Blank nodes according to the RDF specification are scoped to graphs, or even across graphs depending on which type of merging operation [62] is employed. Some proposals aim at restricting the scope of blank nodes to smaller structures, possibly grounded in some root node that is a proper IRI. Such structures

[61] `https://lists.w3.org/Archives/Public/semantic-web/2020Jun/0116.html`

[62] See Section 2.5.2.5

could be lists[63], structured values and n-ary relations, but eventually also more involved structures defined by SHACL or ShEx shapes, or even independently scoped sections in RDF graphs[64].

In an invited talk to ISWC 2009 Pat Hayes proposed to scope blank nodes strictly per graph, making the distinction between union and merge disappear[65]. The accompanying notion of a graph however differs from the set-theoretic definition of graphs in RDF as will be discussed in Section 2.6 below.

Skolemisation The idea of skolemisation is to get rid of blank nodes by replacing them with artificially generated IRIs, 'grounding' the existential quantifiers. In practice skolemisation can help locally e.g. with round-tripping in SPARQL engines. Linked Data proponents argue that blank nodes can't be referenced outside a data source whereas skolem nodes can, improving linkage in the Linked Open Data cloud. An attempt to establish skolemisation globally as a replacement for blank nodes or even to retro-fit to existing data however seems improbable to succeed. Skolem nodes don't provide the syntactic sugar that makes blank nodes so practical for structural purposes. Keeping blank nodes as syntactic sugar in local environments, but skolemizing them in data intended for exchange might be a workable compromise.

The semantics of skolemisation is not without pitfalls either as blank nodes themselves are really just anchor points for the properties attached to them and in total defined by those properties. Skolem identifiers on the other hand naturally convey a little more identity. Although the RDF Concepts and Abstract Syntax document clearly states that their meaning should 'not appreciably' differ, the intuitive semantics of an IRI — of skolemised origin or not — is naturally more definitive than a blank node.[66]

Duplication of data is a more practical problem associated with skolemisation: a non-deterministic skolemisation algorithm might skolemise blank nodes from the same graph to different skolem nodes on different runs, e.g. when the same graph is ingested multiple times from different sources, slightly changed versions or consecutive parser runs. Duplicate triples can be leaned when blank nodes are involved. Skolem nodes however are IRIs and normally can't be traced back to the blank nodes they were generated to replace.

[63]On the Semantic Web mailing list Pat Hayes argues that that would leave OWL's usage of lists alone and restrict blank nodes to an RDF-specific kind of Lisp S-expressions. It would however not resolve the issue of blank nodes in n-ary relations and other star-formed structures — presumably making up a sizeable portion of blank node usage, and much harder to identify than lists, https://lists.w3.org/Archives/Public/semantic-web/2011Mar/0323.html

[64]Again Pat Hayes on the Semantic Web mailing list: "A better system, which would allow for more elaborate structures, would be to have convention of labelled scope brackets of the form [ID]", https://lists.w3.org/Archives/Public/semantic-web/2018Dec/att-0018/00-part

[65]http://videolectures.net/iswc09_hayes_blogic/, https://www.slideshare.net/PatHayes/blogic-iswc-2009-invited-talk

[66]"[S]kolem 'names' aren't names in the proper, identifying, rigid identifier sense. They are just brand-new logical constants, whose very newness guarantees that they shall be unencumbered by any prior burden of referring to anything in particular.", Pat Hayes, https://lists.w3.org/Archives/Public/semantic-web/2011Mar/0346.html

Somewhat ironically a similar argument was also made to come to the opposite conclusion: a second import of the same graph with blank nodes as blank nodes will most probably generate new blank nodes, leading to a second copy of the same graph modulo blank node identifiers. The argument is somewhat stale as this time around the two graph copies that differ only by their blank node identifiers can be leaned, unifying the duplicate blank nodes into one. Not too long ago however this was only a possibility in theory as no efficient leaning mechanism was available, but this changed with the recent work on canonicalization.

Canonicalisation To reap the benefits of ground skolem nodes while avoiding unwanted duplications it would be the most helpful if skolem nodes could be generated in a deterministic, reproducible fashion, probably taking into account the other, ground nodes in the statements in which these blank node occur. [Hog15] presents such an algorithm, guaranteeing that blank nodes from structurally equivalent, isomorphic graphs are converted to skolem nodes in a deterministic fashion: blank nodes from the same graph ingested twice, e.g. from different sources, will be skolemised to the same skolem node. [Hog17] extends that work to graphs that, while not being isomorphic, mean the same according to RDF semantics by first leaning those graphs and then applying the skolemisation algorithm for isomorphic graphs. The problem of graph-isomorphism with blank nodes is in theory GI-complete[67], but for graphs without blank nodes it is only polynomial. In fact as almost all except some rather construed or even adversarial graphs contain a healthy dose of ground nodes both algorithms perform quite well in practice.[68] This makes them suitable to tackle real-world use cases like re-loading blank-node heavy data in a deterministic manner, de-duplication in large graphs and computing security-relevant artefacts like signatures.[69]

2.5.2.9 Muddling Through

Rallying calls that "Blank nodes must DIE"[70] notwithstanding it's safe to assume that blank nodes are here to stay, as in general they are just too comfortable and practical to be given up. The sheer size of RDF data already out there and the difficulty of coming up with a convincingly superior alternative to blank nodes makes it rather improbable that radical change will come to them anytime soon.

They are however so overloaded and burdened with orthogonal purposes and subtle semantic differentiations that it might take still some more experience and practice to establish modelling strategies and patterns that manage to reap their structural and operational benefits while avoiding their pitfalls. Smaller

[67]GI stands for Graph Isomorphism and is known to be in the NP complexity class, but not known to be NP-complete

[68]A proof-of-concept implementation can be found at `https://github.com/iherman/canonical_rdf`

[69]For a slightly more technical but still compact discussion of this topic see `https://lists.w3.org/Archives/Public/semantic-web/2018Nov/0212.html`

[70]See for example the mega-thread on the Semantic Web mailing list by that subject, starting with `https://lists.w3.org/Archives/Public/semantic-web/2020Jun/0123.html`

changes like standardizing round-tripping in SPARQL may smoothen some of the most annoying edges. A better portrayal of the tensions between existential semantics and ground intuitions in the specifications could be beneficial as well. Adding a small mechanism to explicitly state if existential or ground semantics are applied might help to avoid unpleasant surprises — maybe hybrid semantics or nominals can formalize ground intuitions without jeopardizing the benefits of existentials.

Still the mismatch between intuitions when scribbling some triples on one hand and the existential semantics optimized for parsimonious exchange of the finished representation on the other can be hard to bridge. The most secure way forward might be a general awareness of the underlying logic principles and taking to heart some basic modelling advice:

- give proper IRIs to anything that is of some importance and that should be referenceable from outside the local graph
- try to use blank nodes for structure only (lists, branches in nested structures etc)
- in a local system make use of intuitive ground semantics, but when publishing encapsulate those 'something' and 'something else' pseudo-constants in lists to protect them from leaning[71]
- publish and ask for only leaned data on the open web
- make sure the SPARQL engine used supports round tripping
- use surface syntaxes like Turtle that go a long way in hiding structural blank nodes

Something more fundamental can probably be learned from all this confusion and frustration around blank nodes: nothing is really simple when it comes to intuitions, meaning and semantics. Even the most innocent and straightforward looking construct will be interpreted differently under different circumstances and by different people. It will be used, reused and misused for diverging tasks and with incompatible connotations. Human language is extremely malleable and versatile, and untrained users tend to expect the same of knowledge representation formalisms. The more useful formalisms will be those that are aware of their inherent limited-ness, designed with this problem in mind and prepared for it by clever handling of the not so straightforward uses — at least by educating their users, even better by providing some explicit extension point to channel the unavoidable. Much more of this in the upcoming Section 2.6 on semantics.

2.5.3 Named Graphs

Named graphs is a term coined by [Car+05a] and in that work equipped with a very specific model-theoretic semantics. Its syntax is implemented in most RDF triple stores and standardized in SPARQL and subsequently RDF 1.1, albeit without respecting the accompanying model-theoretic semantics proposed by

[71] This can make querying much harder, however.

[Car+05a]. In the context of this section it's sufficient to say that named graphs as standardized in SPARQL and RDF 1.1 are a syntactic construct without a proper model-theoretic formalization. As such they provide a means to attach a name, IRI or blank node[72], to an RDF graph — a set of triples, syntactically e.g. in Turtle enclosed by curly braces — but the exact meaning of the relationship between the name and the graph is undefined. The details and consequences will be discussed in Section 2.7.3 below. As it stands, this syntactic feature is very powerful and can be used in a lot of ways. Sadly however in the context of meta-modelling providing an exact semantics is quite essential as there regularly are many possible connotations that allow intuitions to diverge in wildly different directions, and named graphs as standardized in RDF 1.1 Datasets fail in that respect.

2.6 Semantics

> *Contrary to popular belief in some circles, formal semantics are not a silver bullet. Just because a construct in a knowledge representation language is prescribed a behavior using formal semantics does not necessarily mean that people will follow those semantics when actually using that language 'in the wild.' This can be laid down to a wide variety of reasons. In particular, the language may not provide the facilities needed by people as they actually try to encode knowledge, so they may use a construct that seems close enough to their desired one. A combination of not reading specifications — especially formal semantics, which even most software developers and engineers lack training in — and the labeling of constructs with 'English-like' mnemonics naturally will lead to the use of a knowledge representation language by actual users that varies from what its designers intended. In decentralized systems like the Semantic Web, this problem is naturally exacerbated.*
>
> Harry Halpin and Pat Hayes[73]

The semantics of RDF is formalized using a technique called model-theory and designed for the open environment of a decentralized global information system. This part of the architecture of the Semantic Web is probably the most contested, challenging to grok and hard to abide by in practice — but as the name suggests it is also very close to its heart.

After introducing model theory in theory and covering the discussions about its usefulness for the Semantic Web in practice this section will investigate some specific design decisions for the model-theoretic semantics of RDF that were informed by the very nature of the Semantic Web as a mechanism to integrate

[72]SPARQL only allows IRIs, but RDF 1.1 datasets also allow blank nodes
[73][HH10]

data from anywhere, by anybody, about anything. It will conclude with an outlook into possible future directions for the semantics of RDF.

2.6.1 On Model Theory

The most glorious wars of course were fought over model theory until my dear friend Pat Hayes came in. Most of the people were like "What's thaaat?" and after you explained model theory to them they said "Who cares!". And then Pat Hayes said "We need one!". It doesn't really need one, but: "We need one!".

R.V.Guha[74]

This treatment will follow the path outlined in R.V.Guha's account of the early days of RDF standardization, leading to the RDF 1.0 recommendation in 2004: first introducing the theoretical foundations and then exploring the pro's and con's of basing the semantics of RDF on such a formalism.

2.6.1.1 "What's thaaat?"

Classic logic distinguishes between the realms of syntax and of interpretation. The names that are used to refer to anything — be it real world objects or abstract concepts — have no meaning on their own. Only through a step of interpretation are those names related to something in the 'real world', namely said objects or ideas or whatever one desires to make a statement about. Those names in RDF are IRIs or literal values, their interpretation is what they 'mean'.

A model theory is a logical formalism that captures the relation between a name and its meaning and in extension also of constructs that can be formed by combining those names in ways facilitated by the symbols and rules the logic provides. RDF for example combines names into binary relations, called triples, to formulate statements and the model theory formalizes the ways in which the names can be combined into triples and what those combinations mean.

The way that a model theory formalizes interpretation — what a name or a statement 'means' — is necessarily indirect. There is no way in which one can precisely describe the relation between a conceptualization in one's mind and the real world entity, material or otherwise, that it refers to. In human communication we use words and names as intermediaries, but only through convention and sometimes even only after repeated attempts can we be quite sure that we do indeed mean the same thing when we use the same word. A model theory essentially takes the same approach: names (or, in RDF, IRIs) and their relations (or, in RDF, triples) are mapped to real world entities. The names are in theory regarded as totally void of meaning and therefore many possible mappings, also called interpretations, exist. If such an interpretation establishes a possible 'world' that makes the statements expressed in such triples,

[74]"Light at the End of the Tunnel", keynote at ISWC 2013, http://videolectures.net/iswc2013_guha_tunnel/ at 07:37

by such names, true, it is called a model. The more statements contribute to the model, the more precise the description gets and the fewer worlds exist that can satisfy it. So we get an increasingly good but never perfect account of what the names and the statements that use them really mean. In practice on the Semantic Web the names are IRIs and in general those IRIs, especially if they belong to standard vocabularies, are not void of meaning but address web pages that return human-readable descriptions of their meaning. Often those IRIs are even self-descriptive by using words from natural language that describe their subject, helping humans to quickly understand what the data is about. To machines however such dimensions of meaning are largely inaccessible.

Obviously this account of meaning is rather limited. It doesn't capture the full extent of 'meaning' that a word or phrase can have to a human being. It is restricted to a very limited subset which "could be characterized as the part that is common to all other accounts of meaning, and can be captured in mechanical inference rules", as the RDF Semantics specification [Hay04b] puts it. However, although limited this is still quite useful because, being a mathematical formalism, it is also extremely precise and this can be an astonishingly valuable feature.

Given the ambiguities and ambivalence of natural language, establishing a common account of meaning can be very difficult and time-consuming. This is an inherently unsolvable problem and no amount of rigour can guarantee a common understanding, especially the absence of undetected misunderstandings. A mathematically sound description however, while requiring some prior training to properly understand, is unambiguous and ultimately makes establishing common understanding and, based on that, establishing interoperability between decentralised systems much easier.

The second important feature of a model-theoretic semantics is that such a logically sound formalism can be operationalized and extended with further functionality through deductive reasoning. Information that is logically implicit in some snippet of RDF but not expressly stated can be made explicit through inference and thereby accessible to further computation and retrieval. Such derivations are not necessarily helpful to fulfil rather varied, even chaotic day-to-day information desires, but they can be very useful in highly formalized knowledge environments like medicine, technical disciplines or business processes.

For a more thorough treatment of the RDF model theory the reader is directed to the first version of the RDF Semantics specification [Hay04b] itself which introduces it in substantial detail and is quite accessible also for not logically trained readers.[75] A quite compact account is given by [PH07]. Another very readable and also thorough introduction into model-theoretic semantics with a focus on its foundations in twentieth century philosophy is provided by [Hal12].

[75]The later RDF 1.1 Semantics [HP14] omits most detailed explanations but doesn't introduce any substantial changes

2.6.1.2 "Who cares?!"

Ironically, when RDF was standardized by W3C over three years ago, it came without a semantics. There is now growing understanding that a Semantic Web language without a semantics is an oxymoron, and a number of efforts are directed towards giving RDF a precise semantics.

Guizhen Yang and Michael Kifer[76]

Practitioners in the realm of knowledge representation, programmers just as well as ontologists, are accustomed to creating applications in which they rule supreme, that are tailored to fit a specific purpose and in which sensible defaults and clever guardrails guarantee that users get work done efficiently and with ease — that at least is the idea. One could call those applications little worlds and each application constructs its own world according to its users' needs and desires. This can probably be said for any work, or theory: one makes some implicit assumptions, employs a few fixings and limits to increase efficiency w.r.t. the main task at hand.

The Semantic Web can be understood as one such application, but that doesn't do it justice as it is also, and arguably even more so, a meta application: a framework designed to facilitate the implementation of all kinds of specific applications and services. That quite profoundly changes what base assumptions have to be put in place as will be discussed in Section 2.6.2 below. Yet, as integrating data is an application in itself and usually done not for the sake of it but with specific data to achieve a specific goal, the tensions between a (necessarily, as will be argued below) open paradigm and a (necessarily, by its very nature) closed application scenario can become rather straining. So, unsurprisingly, practitioners tend to call for more 'practical' solutions, with defaults that mimic local application scenarios and assuming a level of control over data customary to closed world environments.

Local environments don't profit much from the semantics that RDF provides, quite to the contrary. The intricacies of meaning that are encapsulated in application logic and interface and shared with users through the application context — the task at hand — itself usually provide much richer semantics than logic based formalisms can encode without getting ridiculously convoluted and cumbersome. Logic is also a two-edged sword: it can be very powerful, but has to be used with care so not to hurt oneself. The rigour and precision, and training, required to successfully apply logical mechanisms is not too common among coders and users alike. To the uninitiated or unaware user they can lead to unpleasantly surprising and unintuitive results, consequently hindering adoption more than providing utility. As [Sow07] notes "even advocates of logic have disagreed among themselves about the role of logic, the subset appropriate to any particular problem, and the trade-offs of ease of use, expressive power, and computational complexity."

[76] [YK03]

For all those reasons formal model-theoretic semantics are often met with a good deal of scepticism. [Bak+13] make the case for a more lightweight semantics when developing the Simple Knowledge Organization System (SKOS) as existing thesauri and other ontologies commonly "cannot typically be translated into the language of RDFS and OWL properties and classes, with their formal-logical implications, without introducing potentially false or misleading logical precision [. . .] Hierarchical relationships, for example, must be disambiguated into relationships of class instantiation, class subsumption, part-whole, or other types—a process that cannot usually be automated." They point to examples where well-established thesauri where first converted to OWL with the aim to increase ontological precision over time, but eventually that project was given up and the bulk of the thesaurus was converted to SKOS. Only those parts that could profit from reasoning in specific implementations where then upgraded to class-based ontologies. This experience is reflected in the account of [Sow07] that "logic is the basis for precise reasoning in every natural language; but without vagueness in the early stages of a project, it would be impossible to explore all the design options.."

Even leading semanticists sometimes admit, or accuse, that "nobody follows the semantics anyway." [Pat10] argues that RDF would be just as useful, and easier to extend and adapt to new requirements, if it was defined as it is mostly understood and used: as merely a data structuring language. Semantics could be defined for higher level Semantic Web languages such as OWL or even extensions to full First Order Logic, that would in turn be free to accept only the parts of RDF they can handle or interpret them in ways they see fit.

2.6.1.3 "We need one!"

Those that do not remember the history of artificial intelligence are bound to repeat it, and the process of specifying inference in RDF led to an almost complete repeat of the 'procedural versus declarative' semantics debate. The original RDF specification defined its inference procedure by natural language and examples. Yet differing interpretations of the original RDF specification led to decidedly different inference results, and so incompatible RDF processors. This being unacceptable for a Web standards organization, the original defender of formal semantics in artificial intelligence, Pat Hayes, oversaw the creation of a declarative, formal semantics for RDF and RDF(S) in order to give them a principled inference mechanism.

Harry Halpin[77]

It is important to differentiate between critiques that question the usefulness of a semantics founded in logic altogether and those that specifically argue against

[77][Hal12] For those interested in better understanding the history of this discussion [Hay77] provides a very readable account by a (or maybe even *the*) proponent of the model-theoretic approach.

basing it on a declarative model-theoretic formalism but would favour a more database-centric approach that feels more like the usual application environment.

It was indeed largely undisputed that the Semantic Web languages would need a logically sound base to enable

- decentralized sharing of information on a global scale based on common vocabularies and exchange syntaxes
- reasoning services that are able to combine, complete, refine and reshape such data as required by local applications.

If one just wanted to share data it would be sufficient to send database dumps around. To enable real interoperability — exchanging and reusing data in a shared information space where sender and receiver can be reasonably sure that they both interpret the data in the same way — some common base of understanding has to be established. The most obvious way in which the Semantic Web approaches this problem is through development and propagation of shared vocabularies for all areas of life. But RDF itself has a much more complicated task to fulfil: it has to provide means to describe and relate the entities described and also the vocabulary terms and properties used to describe them in unambiguous ways. That is much harder than describing some specialized and by and large well-understood domain as it has to wrangle with the many ways in which natural language employs contextualization, is ambiguous and ambivalent and often just plain imprecise. That is where formal semantics come in. Without a formalized semantics interpretation is always in danger to fall back to “Let’s pretend it’s english” [Hay77] and the effects have been studied decades earlier in Artificial Intelligence. [Woo75] in the seminal article ‘What’s in a link?’ analyses how the seemingly straightforward “IS-A” relation has quite different meanings in various projects and formalizations proposed at the time, from subclassing to instantiation to close similarity, and more [Hal12].

However, given the precedent of the largely failed efforts at general artificial intelligence, the design of the Semantic Web aimed to not make the same mistakes again but go for a more pragmatic approach, much in the same way as the design of the World Wide Web ignored a lot of the requirements that at the time were supposed indispensable for a hypertext system to meet.[78] So one was looking for a logically sound solution, but with a pragmatic touch.

The first attempt to standardize RDF by [LS99] was not based on a formal model theory but on descriptions in natural language and test cases as it is customary in application development. That specification however wasn’t able to guide implementations accurately enough to facilitate interoperability but “led to disagreements about the meaning of parts of the language, e.g., whether multiple range and domain constraints on a single property should be interpreted conjunctively or disjunctively” [HP03]. So [LS99] could seem like an attempt to standardize the Semantic Web without a formal semantics the irony of which

[78] See e.g. Tim Berners-Lee’s design note on ‘What the Semantic Web can represent’, `https://www.w3.org/DesignIssues/RDFnot.html`

that wasn't lost on the logicians around. Several proposals were brought forward to fill the obvious gap.

[PH07] categorize that first standardization effort as being in the tradition of what they call the Datalog paradigm — "based on notions from object-oriented databases and rule languages" — , as opposed to the Classical paradigm — "based on notions from standard logics, such as propositional logic, first-order logic, and Description Logics" — but both paradigms can be given formal definitions.

Some proposals are based on the Datalog approach e.g. [SD02], [YK03] and [Bru+05]. Especially [Bru+05] provide a detailed discussion of the merits of a Datalog oriented approach. They describe their approach as "a variant of OWL which is based on Logic Programming rather than Description Logics", taking "Description Logic Programs (the intersection of Description Logics and Logic Programming) as a basis" and extending it among other things with meta-modelling support. [PH07] compare the applicability and utility of the Datalog approach to the classical First Order Logic based approach that drives model-theoretic formalizations and argue that the web as an open environment is quite different in nature from an application centric environment that databases pre-suppose, and much better served by a model-theoretic formalism. Those arguments will be discussed in more detail next in Section 2.6.2 when the model theory that RDF adopted is introduced. [Bru+05] as well as [PH07] stress that both paradigms have strengths and merit in either application oriented or open integration environments respectively and discuss approaches to improve interoperability and complementing application. [PH07] see promise in the addition of epistemic constructs to OWL whereas [Bru+05] envision interoperation between Description Logic-based and Logic Programming-based languages through a common superset and extensions in each direction.

The model-theoretic approach itself can take many forms. The model theory of RDF provides unusually extensive meta-modelling facilities, allowing to reason over the classes and properties that the language is composed of themselves. Normally this would immediately move the logic into Higher Order territory, with very unwelcome consequences w.r.t. tractability and decidability, but the RDF semantics avoids this through a trick that distinguishes classes and properties from their extension. That cleverness however comes at a cost when more expressive languages — especially well-understood dialects and sub-languages of First Order Logic — are layered on top of RDF(S). The entire architecture of the Semantic Web is based on the idea of layering more and more expressive languages on top of each other — from the basic syntax layer in RDF/XML to the fundamental knowledge representation formalism RDF itself to the very basic ontology metalanguage RDFS and on to more expressive languages like rule languages, description logics based ontology languages, more specialized applications like trust management etc. — with each layer building on and extending the lower layers. It turned out that this is easier said than done.

Logical paradigms can diverge in quite fundamental ways and unifying them is challenging not only in terms of computational complexity but also in terms of use and usability because their application follows different intuitions, styles and approaches. Seemingly similar constructs can mean very different things in different logics. [PF02] report on problems encountered when developing more expressive logic languages like OWL as extensions layered on top of RDF(S) model and syntax. Syntactic constructs that are required to encode OWL formalisms in RDF triple syntax have different meanings in RDF and OWL. Paradoxes through meta-modelling, while avoided in RDF and RDFS through its extensional definition, surface in more expressive languages like OWL. All this makes the desired approach of a straightforward same-syntax extensions with extended semantics impossible. Some workarounds are presented for further discussion. They remark however that layering as intended is not even realized between XML/RDF (the then standard RDF encoding) and RDF but only between RDF and RDFS. [HP03] discuss three alternative formalizations of an RDF model theory — one as specified in RDF, one closely following First Order Logic and a one, L_{base}[79], that tries to establish a middle ground — and come to the conclusion that a closer alignment with classical First Order Logic would lead to vastly improved compatibility with suggested upper layers of more expressive reasoning formalisms and stay in well-known logical territory, suggesting a more prudent approach in those early days of the Semantic Web, albeit at the expense of meta-modelling capabilities such as reasoning over classes and properties themselves and "the unrestricted mixing of classes and instances." [Pat05] extends the discussion of the merits of the RDF(S) model theory and comes to the conclusion that "a paradox-free same-syntax extension of RDF to first-order logic is not possible."

However, not all semantics on the Semantic Web is based on a model-theoretic formalization. Most notably SPARQL[80], the predominant query language, is defined by an operational semantics — which is not surprising given the application focus of a query language — and has the same expressive power as non-recursive safe Datalog with negation [AG08b]. This divergence causes some irritating effects e.g. with respect to blank nodes which in SPARQL are considered as constants scoped to the graph they appear in.[81]

2.6.2 A Model Theory for the Open World

There are several aspects of meaning in RDF which are ignored by this semantics; in particular, it treats URI references as simple names, ignoring aspects of meaning encoded in particular URI forms [...] Some parts of the RDF and RDFS vocabularies are not assigned any formal meaning, and in some cases, notably the reification and container vocabularies, it assigns less meaning than one might expect. [...] RDF is an assertional logic, in which each triple expresses a simple proposition. This imposes a fairly strict monotonic discipline

[79] `https://www.w3.org/TR/lbase/`

[80] `https://www.w3.org/TR/sparql11-query/`

[81] [Mal+11], for more detail see Section 2.5.2.6 above

> *on the language, so that it cannot express closed-world assumptions, local default preferences, and several other commonly used non-monotonic constructs.*
>
> RDF 1.0 Semantics[82]

The purpose of the Semantic Web is to integrate data from different sources all over the world and then let that data drive applications. On the logical level these are two very different tasks, suggesting quite contrary strategies and based on opposing intuitions. The semantics of RDF first and foremost serves the purpose of integrating data. By consequence it can be quite irritating and counter-intuitive when putting that data to use in RDF-based applications.

The logic underlying RDF is "based on notions from standard logics, such as propositional logic, first-order logic, and Description Logics"[83]. The RDF Semantics specification from 2004[84] gives a thorough — and also to non-logicians very accessible — introduction into the design of this semantics. [PH07] provide a detailed discussion of the pros and cons of the RDF semantics compared to alternative designs that favour the application aspect — calling the paradigm that RDF is based on the "Classical paradigm" and the other, application focused approach the "Datalog paradigm." As [PH07] discuss, a semantics based on the application-centric Datalog paradigm cannot be extended to the requirements of an integration-centric open world environment. They do however present several approaches that take the opposite direction and offer ways to close down the open world for local applications, thereby creating an environment that provides optimizations in terms of performance and parsimony of expression that are customary in database-driven applications.

2.6.2.1 Open World Assumption (OWA)

A key consideration in the design of the RDF semantics is the fact that the Semantic Web is designed as a decentralized knowledge framework where there is no limit to what can be known and where local access to information is necessarily always incomplete. This is not only a very natural assumption in an open environment, it also enables data modelling tasks that a closed-world formalism can't as easily provide. One may e.g. know that Alice has bought a car but not the exact brand. In RDF the known fact can be recorded and the brand simply left undefined. Another statement may assert that every car has a brand and yet there is no conflict in the data, just some missing information. A database-driven application however might require either complete information or otherwise reject the record about Alice's new car as incomplete, and a logic following the Datalog paradigm would indeed require complete information.

Such constraints can prove very practical in applications that 'know their world': if in a closed-world setting a database contains no record about a car

[82][Hay04b], Section 0.1 'Specifying a formal semantics: scope and limitations', `https://www.w3.org/TR/rdf-mt/#intro`

[83][PH07]

[84][Hay04b], `https://www.w3.org/TR/rdf-mt/`

owned by Alice one may safely assume that Alice does indeed own no car. In the open world of the Semantic Web however drawing such a conclusion would be ill-advised. On the other hand in the open world expressing that Alice has exactly five books requires listing those books e.g. by their title and then also expressing that this list is complete and exhaustive and does not contain duplicates. This can get verbose and cumbersome quite quickly and although tool support can help it is a far from ideal situation. In a database setting however adding five records for Alice's five books implicitly states that those are *all* her books and no further 'closing' is required. While there may be reasonable doubt that a list of Alice's books is indeed complete such considerations become less of an issue in tightly controlled settings like e.g. medical records or critical administrative data. Here a closed world paradigm clearly provides a lot of practical benefit.

As it happens such closed world environments are the norm for programmers and database practitioners and guide the predominant intuitions. Decentralized data integration on the other hand is a rather new field and its terms and conditions neither have "sunk in" in the same way nor are they particularly helpful when the collected data is considered complete enough for the task at hand and is going to be put to good use.

2.6.2.2 No Unique Name Assumption (NUNA)

A related design decision concerns the naming of resources. On the web each IRI is required to point to exactly one resource, but several IRIs may point to the *same* resource. The same naming paradigm is adopted on the Semantic Web because in a decentralized information system it is only natural to assume, and practically unavoidable, that different participants will refer to the same entity by different names. Again there are ramifications and complications. In RDF a set of triples stating that Alice knows both Bob and John doesn't justify the conclusion that Alice knows at least two people as the names John and Bob could both refer to the same person. Such a conclusion would require further statements to the effect that John and Bob do indeed and explicitly not co-refer to the same being. On the other hand if one later learns that a third acquaintance of Alice called Robert is indeed identical to Bob the two references can be safely unified, including any further information that might have been acquired about them — illustrating the benefits of the NUNA for data integration purposes. Despite the downsides it therefore seems that the decision to go without a unique name assumption was quite inevitable. Yet, as will also be discussed below w.r.t. the OWA, a mechanism to declare a unique name assumption locally would be beneficial for applications that can guarantee tight control over their data.

2.6.2.3 Monotonicity

Closely related to the Open World paradigm is the topic of monotonicity. Entailments in RDF are strictly monotonic: all entailments that hold before the addition of some information — via new statements, new graphs or even a semantic extension — must also hold thereafter. In other words: no statement can be made to the effect that some entailments derived from other statements before do no longer hold. As [Hay04b] states "[a]ll logics based on a conventional model theory and a standard notion of entailment are monotonic. Monotonic logics have the property that entailments remain valid outside of the context in which they were generated."[85] This obviously is a very good fit for an open and decentralized environment like the Semantic Web. However it also "imposes a fairly strict monotonic discipline on the language, so that it cannot express closed-world assumptions, local default preferences, and several other commonly used non-monotonic constructs."[86] That is unfortunate as such constructs reflect the intuitions and implicit semantics of local applications like "database applications, where one concludes from a lack of information about an entity in some corpus that the information is false (e.g. that if someone is not listed in an employee database, that he or she is not an employee.)"[87]

2.6.2.4 Monotonicity & Statement Annotation

Because of the monotonicity of RDF the semantics of statement annotations is a delicate matter. "RDF graphs can be viewed as conjunctions of simple atomic sentences in first-order logic"[88], those 'simple atomic sentences' being RDF statements. Individual terms in those statements denote real-world entities in interpretations that in turn fix the truth-values (true or false) of statements that contain those terms. No statement is allowed to change the truth value of another statement, i.e. no statement is allowed to claim that another statement is false, invalid, only true under certain conditions etc. In the case that e.g. statements are found to contradict each other the conflict has to be resolved with means outside RDF. Under that perspective annotating a statement by e.g. constraining its validity to a certain time span may be understood as limiting its truth value and therefore be illegal. If on the other hand the annotation merely notes the source of the statement it might be considered okay. Another perspective however argues that e.g. for certain use cases the source might be the decisive indicator of the validity of a statement, and that in general there exists no universally valid distinction between (harmless) annotations with metadata and (dangerous) qualifications with additional data. This discussion will be covered in more detail in Section 14.2 below.

2.6.2.5 Sets & Types

RDF graphs are defined as sets of statements as is customary in applications of model theory in logic. A set is a mathematical abstraction: it is defined by the elements it contains and it cannot contain the same element multiple times. By consequence an RDF graph can't contain the same triple twice. From a logical

[85][Hay04b], Glossary "Monotonic"
[86][Hay04b] Section 0.1
[87][Hay04b], Glossary "Nonmonotonic"
[88]RDF 1.1 Semantics, Section 5.1.1, `https://www.w3.org/TR/rdf11-mt/#blank-nodes`

perspective that would make no sense: a triple stands for an assertion of some fact and that fact is and remains the same fact no matter how often, by whom, for what purpose etc. it is asserted. Instead RDF understands each statement as an abstract type. Likewise a graph in RDF is not understood as a concrete realization as it may occur in an RDF document or some other self-contained snippet of RDF data. A graph in RDF is defined by the set of triples it contains — add a triple, remove a triple, or replace it by another triple and in each case the result is a different graph, no matter what name it may be given in an RDF dataset or under which IRI it is dereferencable.

This choice of semantics is not only customary in applications of model theory, it also fits well the integration purposes of the Semantic Web to which the facts themselves are much more important than who stated them, when, how often etc. In use however there are many cases when this semantics runs counter to practical requirements: in a globally shared information space as envisioned for the Semantic Web, practice demands to manage data depending on source, purpose and application. An RDF graph in this practical sense may be a document or a named graph with an IRI and further attributes that all influence its meaning within an application just as much as the statements it contains (which too may vary over time). Statements, just like graphs, are not independent of such aspects of provenance and context.

This tension between integration semantics and application intuitions has some tricky consequences. Annotating a graph or statement with e.g. some provenance information can be interpreted as referring not to the abstract type but to an occurrence. One might argue that such an annotation constitutes the description of an event — the occurrence of the statement — instead of the statement itself. On the other hand, depending on how one reads the statements one might as well understand the annotation as referring to the statement itself, treating its occurrence(s) as attributes to the statement. In natural language the pronunciation would make the difference, and it would be a gradual difference as there are infinitely many ways to pronounce the sentence "Carol said that Alice likes Bob": emphasizing the terms "Carol" or "said" or "Alice likes Bob" or any other combination, specific intonation etc. In essence the perceived dichotomy between type and occurrence is actually gradually differentiated and depends on contextual aspects like usage and intention.

If a statement or graph is (to be) understood as a type or an occurrence doesn't depend on attributes it is annotated with but on use and intention. Every statement is inevitably associated with the act of stating it (ignoring the infinitely many possible statements that have not been stated, at least not yet). Some even consider any RDF statement to constitute a speech act, rejecting applicability of the notion of an abstract statement type altogether. One might e.g. ask in what sense a statement coming from two different sources, one very trustworthy but the other known to disseminate fake news, is one and the same statement. The statement — and all that comes with it, like e.g. qualifying context — and the act of stating it — and what comes with it, like e.g. source credibility — can hardly be separated. Representational techniques have to take that dichotomy and the gradual shift between its extremes into account. .

Unsurprisingly such subtleties tend to get lost in a formalism as basic as RDF and indeed the distinction between type and occurrence doesn't seem to require much attention in the representation of annotated statements in practice. However, they come back with a vengeance when multiple occurrences *and* multiple attributes are involved. Catering for those more involved cases from the get-go requires some upfront design decisions that may seem tedious and counter-intuitive in the standard case, forcing the user to provide detail that is not needed most of the time. The other possible design choice is to tackle the problem only when it actually occurs. That however leads to the need to re-model existing data or at least to introduce a second modelling primitive. In any case it makes the data less predictable and complicates queries and navigation. Similar issues arise with (named) graphs. [Hof+13] present a design diverging from RDF where every statement has an identifier and multiple occurrences of the same statement have different identifiers. [Las+21] describes a quad based approach and discusses ensuing questions like how to update or delete annotated triples with multiple occurrences. Section 2.7 on meta-modelling facilities provided by RDF and Section 4 on proposals to overcome their deficits will discuss the representational aspects of those issues in more depth.

2.6.2.6 Axioms & Constraints

A topic of much debate has been the focus of RDF on entailments rather than constraints — the latter again a domain of application driven intuitions. Especially the confusingly named RDF Schema specification[89], designed as an ontology language providing vocabulary to describe some domain, is often misunderstood and misused as a constraint language to define restrictions and illegal constructs — something that RDF can describe, but not enforce — resulting in undesired entailments instead of cleaned data. So if for example the `rdfs:domain` of a property is declared, the correct interpretation in RDF is that every subject of that property is of the type given by the domain declaration. This just adds information about subjects in relations of the given property. The 'conventional' reading of a constraint would be that in the domain of that property only values of the type given in the domain declaration are allowed.

The concepts of axioms and constraints can spur similar discussions about half-empty or half-full glasses as seen above. Axioms are logical instructions that allow us to entail new statements from existing ones by well-defined rules, making knowledge explicit that was already present, although hidden, in existing facts. The facts generated by those entailments are not really new but rather have already been implicitly contained in the knowledge from which they are derived. Entailments just add them to the knowledge base explicitly and thereby make them readily available for further processing, querying etc. Constraints on the other hand are rather a means to discriminate valid from invalid constructs,

[89]`https://www.w3.org/TR/rdf-schema/`

to check facts for validity and to enforce data validation. However, constraints are expressed via axioms just as well, and axioms can describe conditions that make the data invalid. In that sense it depends on the application if an axiom acts as a constraint or a description.

In the open and strictly monotonic world of RDF semantics axioms entailing new facts are the way to go whereas constraints are problematic. It is possible, and valid, to describe an expected construct, but it is not possible within RDF to enforce such expectations — that requires out-of-band means. However, a constraint that can't be enforced but merely describes a desired outcome is rather an axiom than a constraint.

2.6.2.7 Restrictions vs. Constraints

[PH07] compare RDF, which they describe as following a 'Classical' paradigm, and particularly its approach to restrictions and constraints to that of application oriented logics that follow the Datalog paradigm. Whereas RDF describes a possible state of a world, and new statements narrow down the number of possible worlds that fit that description, an approach following the Datalog paradigm defines exactly one world and restrictions on that world can only be introduced by making certain inputs illegal. Both paradigms might be characterized as approaching the problem from opposite directions, but they don't always meet in the middle. The work discusses various aspects of this issue in considerable depth.

2.6.2.8 Referential Transparency

As discussed above in Section 2.1 identification is a topic of much hidden complexity. One of the core design principles of the Semantic Web is that, being a decentralized information system, it cannot rely on some central naming authority. It does demand from a name that it uniquely identifies one resource, but not that it is the unique identifier for that resource. Every identifier is required to refer to precisely one resource — be it a person, a thing, an idea, an event, whatever — but one such resource can be identified by many names. For example a person may be known by multiple names and can be referred to by any of those names, by its homepage on the web, its Wikipedia-entry, one of its email-addresses, its physical address, its social security number or many other means. Likewise, some thing or idea or event can be referred to by different names, different web pages or through different naming schemes.

To guarantee interoperability between different references to the same resource the semantics distinguishes between a reference — e.g. a person's name — and its interpretation — the real life person in flesh and blood. This aspect is called referential transparency: an identifier is referentially transparent as it can at any time be replaced by another identifier that refers to the same resource in the realm of interpretation without changing the interpretation, and thereby the meaning, of what was said. And preserving the meaning irrespective of the name that was used to express it is what RDF is most concerned about. This is called co-denotation and to foster interoperability the Semantic Web provides the `owl:sameAs` property to expressly declare that two identifiers refer to the same entity.

This semantics however clashes with certain use cases that do indeed care about the specifics of the term used to refer to some resource. An exemplary use case in the RDF world is known as the 'Superman problem':[90] the comic character Lois Lane knows that Superman can fly, but she doesn't know that the reporter Clark Kent is a persona of Superman. So she thinks that Clark Kent can not fly. Modelling this kind of incomplete information from the higher perspective of the all-knowing comic reader is problematic in RDF because Clark Kent and Superman both refer to the same extraterrestrial being Kal-El. A slightly more down-to-earth example is that of Morning Star and Evening Star which in minds of contemporary observers identified different stars although they both refer to the planet Venus. It is of course possible to describe all this in RDF, but it is quite difficult to honour the respective viewpoints.

Besides these rather exotic examples there are also real world applications that require to suppress co-reference and to preserve the exact terminology used, much like quotation in natural language. They are often operating very close to the metal like in application specific scenarios such as versioning in code repositories or Explainable AI, but also in security sensitive or legal applications or whenever it is important to convey a message exactly as it was formulated, possibly including verification thereof.

[CS04] propose an XML-based alternative to the RDF/XML syntax that among other features offers referentially opaque graphs. They give an informal account of the intended semantics and refer to [Car+05b] which present a formal semantics for graphs as referentially opaque speech acts, including an elaborate motivating example. That work will be discussed below in Section 4.2.5 in more detail. [Ber+07] proposes Notation 3 (N3) of which the popular Turtle syntax was derived as a subset and in which graphs — here called formulae — are a prominent feature, again with referentially opaque semantics. Lately [Har+21] introduced quoted triples in RDF-star as entities whose IRIs refer to resources in the realm of interpretation but only when using the exact same syntactic representation — a very specific take on referential opacity and quotation.

2.6.2.9 Non-Normative Parts

Some parts of the RDF specification have no normative semantics which is, however, not to say that they have no semantics at all. The reification vocabulary is very precisely defined. `rdf:value` is provided with a somewhat looser but intuitive semantics. Lists and collections do have clear semantics, but fall short of programmers' expectations because of limitations imposed by the open world assumption. If anybody is helped much by a precise but not normative

[90] `https://www.w3.org/2001/12/attributions/#superman`

specification of a semantics — maybe because it can't satisfy certain expectations — is a question that will not be further discussed here.

2.6.2.10 Semantic Extensions

RDF provides an extension mechanism that may define syntactic restrictions and "treat RDF graphs which do not conform to the required syntactic restrictions as syntax errors"[91]. OWL[92] defines such restrictions for example for lists, requiring them to follow certain well-formedness restrictions that one would usually expect from a list — in this case for example that the list doesn't fork. Otherwise the mechanism is rarely used, although it would allow for implementing very application specific behaviours. [Hay12a] describes the design of a semantic extension to provide named graphs with contextualization semantics.

2.6.2.11 Social Meaning

As [PP04] note, early versions of what would eventually become the 2004 RDF Concepts and Abstract Syntax specification did contain a more nuanced definition of meaning, differentiating between formal and social meaning of a graph. Formal meaning would encompass all that was defined logically by the model theory of RDF whereas social meaning would capture those aspects of meaning that the formal system of RDF is too weak to express. The January 23rd 2003 draft of RDF Concepts and Abstract Syntax[93] states that "[a]n RDF graph may contain 'defining information' that is opaque to logical reasoners. This information may be used by human interpreters of RDF information, or programmers writing software to perform specialized forms of deduction in the Semantic Web." As an example qualified assertions are given. As discussed above due to its monotonic semantics RDF can't handle qualifications well. The social semantics however would still honour such meaning that is out of reach to the formal semantics: "[. . .] publishing the same statements with a qualification, such as 'here are some common myths', or as part of a rebuttal, would likely not be construed as an assertion of the truth of those statements. Similar considerations apply to the publication of assertions expressed in RDF." This approach could however not garner consensus as, in the words of Bijan Parsia, "specifying the interaction between 'social' and formal meaning, heck, just specifying much of anything about social meaning is an INCREDIBLY hard task. So either this section is vacuous (i.e., it doesn't really specify anything and thus can be ignored) or it's dangerously underthought and underspecified."[94] It would, as [Hal12] puts it, by consequence have introduced "a special type of *non-logical* reasoning." It therefore was dropped from the final specification, deleting all references to

[91] [Hay04b]
[92] [MPP12]
[93] https://www.w3.org/TR/2003/WD-rdf-concepts-20030123/#section-authority
[94] In a message to www-rdf-comments@w3.org, https://lists.w3.org/Archives/Public/www-rdf-comments/2003JanMar/0366.html

'Social Meaning' and "leaving RDF with only the sparse meaning provided by its model-theoretic semantics."[95]

2.6.3 There be Dragons

Meta-modelling, when not designed properly, enables the introduction of paradoxes which can have devastating effects on the underlying logic. Excluding paradoxes is possible but in general requires formalisms that are undecidable or at least very involved. RDF however is designed with purposefully low expressivity and a strong focus on ease of use and decidability. Therefore, there exists a clear conflict of interest between much desired meta-modelling facilities and on the other hand straightforward syntax and effective reasoning based on sound semantics.

2.6.3.1 Paradoxes

The model theory of RDF is couched in set theory. Set theory was developed in the late 19th century and "is commonly employed as a foundational system for the whole of mathematics."[96] It was however soon discovered that naïve implementations of set theory as formalized by Frege at the time[97] enable paradoxes. Such implementations contain a very large collection of built-in sets: for any well-defined property there exists a set that contains the objects that have that property. From one such property the set of all sets that do not contain themselves could be constructed. If that set is a member of itself, then it does contain itself. So consequently it is not a member of itself. If however it is not a member of itself then it doesn't contain all sets that don't contain themselves. In other words: there is no interpretation that makes the sentence either true or false. It is a paradox, and in this form known as Russel's Paradox.[98] A simpler and well known example is the Liar's Paradox: a Cretan says that all Cretans are liars. It can be put even more succinctly as a statement that states that it itself is false.

2.6.3.2 Inconsistency

A paradox must not be confused with a contradiction: if a contradiction occurs it hints at a problem in the data, e.g. it may have been derived from a false assumption. That is useful to know and can indeed be employed as a way of checking a hypothesis. If however a paradox is included in the data then a contradiction can be inferred from nothing, and from nothing everything can be inferred. Consequently, the logic is inconsistent, and the whole formalization collapses. As the above set of all sets that don't contain themselves is a well-defined set according to the naïve formalization of sets and thus is contained in

[95] [PP04]
[96] https://en.wikipedia.org/wiki/Set_theory
[97] https://en.wikipedia.org/wiki/Naive_set_theory
[98] See https://plato.stanford.edu/entries/russell-paradox/

all models based on that formalization, that is exactly what happened to Frege's formalization of set theory.

2.6.3.3 Self Reference

There are many more ways how a paradox can appear and no automatable way to catch them is known. Russell's paradox is based on self-reference, as is the Liar's paradox and many other popular examples. In general, being able to refer to a statement in the same language — no matter if it is the statement itself or another statement — opens the door to paradoxes. As [YK03] sum up, "[t]he basic reason for the existence of this paradox in Frege's logic is the ability to reify logical sentences and make statements about these sentences. Ever since this discovery logics that permit reification were subject to strict and just scrutiny." However, purposefully impoverished expressivity poses a serious problem to meta-modelling.

2.6.3.4 Getting By

For an introductory treatment of the Liar's paradox and ways to deal with it in logic, classic or otherwise, see `https://plato.stanford.edu/entries/liar-paradox/`. One customary way to avoid paradoxes is to put self-references on a higher level. Such logics are therefore also called Higher Order Logics, in contrast to the First Order Logic that underlies the semantics of RDF. That approach however requires one new level for each increment of self-reference and the logic quickly becomes too complex to be computable at a scale required by the Semantic Web. Other approaches avoid layering but instead introduce mechanisms to quote and unquote statements, or, as [Per85] puts it: "The connection between self-reference and truth is simply that to do self-reference we need names for expressions, hence quotation (of some sort) and a way to relate the names to what is named, hence unquotation (i.e., a truth predicate)." [Hay09] shows how e.g. the IKL project treats references to statements as descriptions that are not evaluated. Only on evaluation an eventual paradox would return. Unevaluated it just describes a contradiction.

[PF02] explain how layering OWL on top of RDF and RDFS as a same-syntax extension with extended semantics — prima facie the most desirable design — falls prey to the same problem as Frege's formalization of sets: it would have to include a large set of classes, among them the class that is defined as those resources that do not belong to the class. They then discuss several alternative designs.

2.6.4 Back to the Future

> *Few weeks ago in conversation with TimBL and @ImageSnippets. Me: giving RDF a formal semantics was a mistake. TimBL: it was needed to suppress endless 'philosophical' debate. True. [...] I now think that the mistake was making it a NORMATIVE part of the RDF spec. See what inferences people expect to be able to make, and what kinds of contradictions they find important. Then is a good time to try to make the intuitions formal; not before anything has been done.*
>
> Pat Hayes, 2019[99]

One might claim that the bread and butter of the Semantic Web are closed applications, with the semantics encapsulated in code, and all that is really needed to successfully exchange information is a suitable syntax while everything more far-fetched is pipe dreams and castles in the air. If that was true however it would have serious repercussions on the vision of a decentralized Semantic Web, where unforeseen benefits can grow from open exchange of data. The jury is still out on a set of modelling primitives and a semantics that are general enough to be easy to use for the lay practitioner but at the same time powerful enough to successfully communicate more involved, complex or strictly local issues.

In 2010 the W3C held a workshop called "RDF Next Steps" to discuss further standardization efforts for the Semantic Web. Proposals ranged from "Back to the Graph"[100], discarding the model-theoretic semantics, over "Keep the core and pave the cow paths"[101] to extending RDF towards contextualization[102] and full-on First Order Logic[103].

2.6.4.1 Back To The Graph

When trying to establish a broadly used and semantically sound knowledge representation environment one can take a laissez-faire approach that starts from a shared syntax and enforces specific semantics for well-defined application environments only or one starts with a shared but very basic semantics that can be made more expressive and powerful through additional layers. The Semantic Web was first designed with a syntax and some overly optimistic assumptions w.r.t. the semantics by [LS99]. As that didn't work out satisfactory the 2004 standardization effort endorsed the second approach and designed what was envisioned to be the foundation of a layered architecture of increasingly expressive formalisms. However, [Pat05] shows that "a puff of philosophical wind is sufficient to knock over a Semantic Web tower built solely from RDF. A

[99]On Twitter, `https://twitter.com/ThePatHayes/status/1192869509188722689` — retrieved from Twitter in 2020, however not available there anymore as Hayes seems to have deleted his Twitter account

[100][Pat10]

[101][Cyg10]

[102][JM10]

[103]`https://www.slideshare.net/PatHayes/blogic-iswc-2009-invited-talk`

stronger tower needs better building material." and Hayes' 'BLogic' contribution below proposes just that.

[Pat10] however argues for the first approach[104] and proposes to treat RDF as merely a data structuring language without any semantics attached, "going 'back to the graph', i.e., treating RDF simply as a data structuring language, much as was done in the initial specification of RDF [LS99]." He concedes that "[i]t may seem that this is a step backward, as the respecification of RDF was done precisely to firm up just what should be done with RDF graphs. The problem is that it is not really possible to treat RDF graphs as both a syntactic and semantic foundation for the Semantic Web. Even at the RDFS level, there is a desire to ignore parts of the RDF semantics. At the OWL level, effective reasoning requires a different semantic treatment. At higher levels, such as a Semantic Web language that covers all of first-order logic, a different semantic treatment is required to avoid paradoxes [Pat05]. Therefore something needs to be done, and it seems to me that the simplest thing to do is to turn RDF graphs into a neutral data structure so they can be used as the higher levels of the stack desires."

2.6.4.2 BLogic & Some More

In 2009 Pat Hayes gave an invited talk at the International Semantic Web Conference[105] in which he discussed shortcomings of the current specification and sketched a way forward towards a model theory for RDF with contexts, delivering on the promise of a layered architecture that the current semantics fails to achieve, and a more expressive and adaptable design altogether. The RDF 1.1 Working Group discussed this further[106] but eventually decided that the topic was not ripe for standardization and took a rather minimalistic stance, changing the specifications as little as possible.[107]

In an email from around that time[108] Hayes proposes to act on a few more issues like:

– simplify RDF abstract syntax, "literals can be subjects, properties can be blank nodes, etc"
– add a scope mechanism to the syntax and on that foundation implement named graphs, negated graphs, application-oriented closed world semantics, reification, unasserted assertions etc
– recommend *one* way to encode n-ary relations, "explain it fully, and deprecate all the others"

[104]And Hayes in his 2019 tweet cited above seems to agree

[105]See the video-recording at `http://videolectures.net/iswc09_hayes_blogic/`, slides at `https://www.slideshare.net/PatHayes/blogic-iswc-2009-invited-talk`

[106]See `https://www.w3.org/2011/rdf-wg/wiki/RDFwithContexts` and `https://www.w3.org/2011/rdf-wg/wiki/AnotherSpin`

[107]See e.g. [Cyg10]arguing for such a restrained approach. Hayes' talk however still inspires further investigation as manifested e.g. by Jos de Roo's post "Implementing Pat's Blogic", `https://lists.w3.org/Archives/Public/semantic-web/2021Sep/0038.html`.

[108]`https://lists.w3.org/Archives/Public/semantic-web/2009Nov/0040.html`

- replace RDFS by SKOS, effectively taking a more restrained approach to reasoning and not generating entailments that are not explicitly asked for
- develop a more nuanced approach to ontology imports than owl:imports
- develop a more nuanced approach to identity reconciliation than owl:sameAs
- add some support for versioning, deprecation and similar.

Most of these issues are still unresolved.

For Tim Berners-Lee's list of topics for the upcoming RDF 1.1 Working Group — that has some overlap with this one, plus nested graphs, collections as native objects and a focus on N3 — see `https://www.w3.org/DesignIssues/RDF-Future.html`.

2.6.4.3 The Problem Of Scope

In a discussion on the Semantic Web mailinglist about blank nodes and more specifically the scoping of blank nodes which only serve structural purposes to those specific structures via some syntactic device like e.g. a special pair of brackets, Antoine Zimmermann argued for applications of a scoping mechanism that go way beyond mere structural issues:[109]

> *It would be good to be able to say (in a standard way), for instance:*
>
> 1. *within my scope, this graph has to be interpreted with a unique name assumption (I know the things in my local system, I do not identify them twice)*
> 2. *within my scope, this graph has to be interpreted with a closed world assumption (I know everything about my things in my local system, if you can't conclude a statement is true from my scoped graph, you can conclude it is false)*
> 3. *within my scope, bnodes are interpreted as constants (so, the graph 'ex:s ex:p _:bn1, _:bn2' is effectively not equivalent to 'ex:s ex:p _:bn3')*
> 4. *within my scope, I abide by the RDFS entailment regime (or OWL 2 RL; or OWL RDF-based semantics; or RDF recognising xsd:duration, geo:wktLiterals; etc.) - which means I may not agree with other semantic restrictions made by other standard regimes*
> 5. *within my scope, this specific set of inference rules hold (such as, that owl:sameAs is reflective and symmetric, but not necessarily transitive) - if you consume my data, you can assume what you want, but you've been warned! and possibly more:*
> 6. *within this scope, the triples are assumed to be truthfully describing the world as it was at this specific time point / time interval*

[109]`https://lists.w3.org/Archives/Public/semantic-web/2018Dec/0124.html`

> 7. *within this scope, the triples are not trustworthy (or only trusted to a certain degree)*
> 8. *within this scope, the graph describes Mr X's opinion*
>
> *etc.*
> *Any real life application that consumes an open set of data from the Web has, in any case, to define its scoping mechanism. There's much research work showing that RDF is used all over the web with different assumptions: assumptions regarding bnodes, regarding unique names, regarding closed world, even regarding the meaning of normative terms like rdfs:domain, rdfs:range, owl:sameAs; regarding literals as well. If one ever wants to build an application that consumes Web data at large, one has to put barriers to the normative semantics of Semantic Web standards. Unfortunately, the standards do not make this explicit, and people have to experience the big slap in the face of Web inference before working out yet another scoping mechanism (or just give up and do GraphQL...).*

2.6.4.4 The Baby & the Bathwater

[VS20] address "the community's unconscious bias toward addressing the Paretonian 80% of problems through research — handwavingly assuming that trivial engineering can solve the remaining 20%. In reality, that overlooked 20% could actually require 80% of the total effort and involve significantly more research than we are inclined to think, because our theoretical experimentation environments are vastly different from the open Web. [. . .] If we are hesitant to step up, more pragmatic minds will gladly reinvent technology for the real world, only covering a fraction of the opportunities we dream of."

They see a "desire for more simple choices with less flexibility [as] illustrated by the backing of Schema.org by the major search engines and the increasing popularity of the shape languages SHACL and ShEx. They cover an important gap between data in the wild and applications that need to know what kind of data to expect [...] The disconnect between the need of semantics and the effort to provide it, has cultivated a heterogeneous and underspecified Web of Data." In particular, they warn that "enterprises and developers start to give up on the formal semantics, and we risk the baby being thrown out with the bath water."

2.6.4.5 (No) Locally Closed Worlds (Yet)

As discussed above one way to take the edge off these problems is to start from an open-world monotonic semantics and provide means to 'close' that world on demand. As e.g. [PH07] show different possible approaches exist to close the open world of RDF for the benefit of local applications. The RDF Semantics specification itself points out that "[t]he relationship between monotonic and nonmonotonic inferences is often subtle. For example, if a closed-world assumption is made explicit, e.g. by asserting explicitly that the corpus is complete and providing explicit provenance information in the conclusion, then closed-world reasoning is monotonic; it is the implicitness that makes the reasoning nonmonotonic. Nonmonotonic conclusions can be said to be valid only in some kind of 'context', and are liable to be incorrect or misleading when used outside that context. Making the context explicit in the reasoning and visible in the conclusion is a way to map them into a monotonic framework."[110]

There always seems to have existed broad consensus that some device to locally close worlds would be very beneficial. As Hayes argues early on, "[t]he global advantages of monotonicity should not be casually tossed aside, but at the same time the computational advantages of nonmonotonic reasoning modes is hard to deny, and they are widely used in the current state of the art. We need ways for them to co-exist smoothly."[111]

[PH07] point out that "[g]ood modelling requires considerable (fore)thought in any context. The open and distributed nature of the Semantic Web does not make modelling any easier; on the contrary, modelling in the Semantic Web is more difficult than modelling in, for example, a database setting. Pushing the Semantic Web back towards a closed paradigm, however, would negate much of the power of the Semantic Web." And they continue with a cautiously optimistic outlook, saying that a "promising direction for the future is to add epistemic constructs to OWL. An epistemic Description Logic provides a formalism that is mostly open, but that can close certain areas of information as desired. Much work remains to be done here, particularly to identify useful subsets in which reasoning can be performed effectively (much as regular Description Logics identify such subsets of first-order logic), but the result could be a logic that combines most of the advantages of the Classical and Datalog paradigms."

Hayes sketches a very simple solution in the email already cited above: "There is a way to combine the global security of monotonic reasoning with the local advantages of nonmonotonic reasoning (eg when working with hospital records on a website, say), which is to provide a way to state the closed world assumptions explicitly. Take the hospital-records example again, where you fail to find any record of a patient and conclude that the person never was a patient. That is a non-monotonic inference from just the patient records, but it is monotonic from the patient records PLUS an explicit statement of the closed-world assumption, ie the statement that the set of records is exhaustive. So if we have a way to refer to a set of assertions - say, all those at a certain URL, or all those which use a certain namespace, or by some other means - and a way to state the closed-world assumptions that are being made about this set of assertions - say, they are exhaustive with respect to all queries of a certain form - then the overall reasoning can be considered monotonic, even though it proceeds locally by using efficient nonmonotonic methods."[112]

However, no such mechanism has been standardized yet. Attempts based on named graphs have been unsuccessful as will be discussed below in Section 2.7.3.

[110][Hay04b], Glossary "Nonmonotonic"
[111]Pat Hayes in `https://lists.w3.org/Archives/Public/www-rdf-logic/2001Jul/0067.html`
[112]Pat Hayes in `https://lists.w3.org/Archives/Public/www-rdf-logic/2001Jul/0067.html`

This forces practitioners to rely on out-of-band means which is disadvantageous not only w.r.t. data sharing but also because the availability of an explicit mechanism would help raise awareness for the sometimes subtle and irritating issues spurred by open world semantics.

2.6.4.6 Personalized Semantics

In a keynote[113] to the International Semantic Web Conference 2022, [22], Markus Krötzsch argues for the need to move away from huge monolithic semantics standards and develop local semantics that capture application-specific aspects and needs. He gives examples from Wikidata where queries are collaboratively developed and shared in the community. Those queries can be as expressive as OWL QL. With a rule language based on terminating existential rules even an expressiveness equal to OWL 2 DL can be achieved.

In the Notation3 community[114] work is ongoing to implement configurable semantics for individual graphs, see e.g. [AW19]. It will be interesting to follow how much such initiatives can contribute to the promise of a semantics that is more welcoming and adaptable to use cases and user needs.

2.7 Complex Relations

The distinction between "data" and "metadata" is not an absolute one; it is a distinction created primarily by a particular application. Many times the same resource will be interpreted in both ways simultaneously.

Ora Lassila and Ralph Swick[115]

[S]tatements are part of the very world they purport to describe.

Donald Perlis[116]

This section will finally discuss how RDF can be used to model more than straightforward relations between two entities. This concern meta-modelling, but also qualification of terms and composition of complex objects. Support for such advanced modelling tasks in RDF is limited as RDF is a purposefully simplistic language, providing only basic building blocks. While those can cover a very wide range of tasks, the focus on simplicity and flexibility comes at a cost: if the tasks are less than simple, the resulting constructs suffer from increasing verbosity and a lack of guidance of how to model things, endangering interoperability.

Meta-modelling, making statements about other statements, is the area where the representational facilities of RDF probably leave the most to be desired. This

[113]video `https://www.youtube.com/watch?v=ryxusv0s604`, slides `https://iccl.inf.tu-dresden.de/w/images/1/1b/Iswc-2022-keynote-slides-kroetzsch.pdf`

[114]`https://www.w3.org/community/n3-dev/`

[115][LS99], Resource Description Framework (RDF) Model and Syntax, `https://www.w3.org/TR/WD-rdf-syntax-971002/`, the first RDF specification

[116][Per85]

however is not a regrettable oversight and also not a shortcoming specific to RDF. Formally sound meta-modelling is rather a topic that has logicians kept busy since the beginning of the twentieth century. It is a problem that is hard in theory as well as in practice — quite finicky to pin down precisely and explosive w.r.t. computational demands — and no straightforward general solution has been established yet. The success of Labelled Property Graphs (LPG)[117] however is a clear indication that users find meta-modelling very intuitive. Those solutions tackle only specific dimensions of expressivity, lack clear semantics and tend to work only in local, application-driven environments governed by shared assumptions and out-of-band conventions. Still their popularity underscores the relevance and urgency of the problem.

Other areas like qualification and complex objects are less actively discussed and researched but seem to pose significant problems in practice nonetheless. The rudimentary support for lists, the uncertainty of how to best express n-ary relations, the widely varying uses of blank nodes frustrate newcomers and seasoned developers alike.

2.7.1 Node Qualification

RDF does provide no direct means to qualify IRIs as it would be required to express e.g. specific identification semantics — e.g. if an IRI is used to denote a document or what the document refers to, as discussed in Section 5 — or to denote an entity at a specific time or place. Such qualifications require minting a new, more specialized identifier. For example one way to identify Berlin in the 1920ies would be to mint a new IRI for that purpose, like `https://example.net/resource/Berlin#1920ies`, and ideally let it point to a web resource that explains its meaning to a human user.[118]

To make that meaning more accessible to Semantic Web agents one could also add some explaining statement, e.g.:

```
https://example.net/Berlin#1920ies
    rdf:value https://en.wikipedia.org/wiki/Berlin ;
    :temporalConstraint "1920ies" ;
    :identificationSemantics :Reference .
```

[117]Discussed in Section 4.2.4.6 further below

[118]The example IRI is mimicking DBpedia's `http://dbpedia.org/resource/Berlin` instead of re-using it because adding a temporal fragment identifier to the DBpedia IRI with would seem inappropriate, and certainly wouldn't lead to a page that explains the IRI's meaning to a human reader.

Another option would of course be to omit minting a proper IRI and just introduce a blank node, like so:

```
:Alice :likes [
    rdf:value https://en.wikipedia.org/wiki/Berlin ;
    :temporalConstraint "1920ies" ;
    :identificationSemantics :Reference .
]
```

However, as already discussed in Section 2.4.1 above, the `rdf:value` property has no formal semantics. Its meaning is defined only informally as pointing to the main component of a structured value. While that might be considered semantically close enough to fit the use case of qualification it doesn't provide the solid basis required for an adequate support of term qualification in RDF.

The example also shows that opportunities for qualifications can be numerous and use-case specific approaches like e.g. DBpedia's `resource` and `page` paths to discriminate access from reference can only ever cover just one of them.

Section 4.2.2 below presents proposals to fill this gap and Section 7.3 discusses alternatives to `rdf:value`.

2.7.2 Reification

Reification is broken, because it has never been given a satisfactory semantics. (I would bet a good beer that there isn't a single deployed use of RDF reification that strictly conforms to what the spec says about it, normatively.)

Pat Hayes[119]

The basic issue is the "abstract syntax" in which an RDF graph is defined to be a SET of triples. [...] the intention is that the subject node of an RDF reification always refers to a token — a piece of concrete syntax in some surface form of RDF, whether actually physically realized or not — and not to a set-theoretic or mathematical abstraction. [...] The RDF spec is almost entirely concerned with [abstract triples], and deliberately avoids the topic of [triple tokens], but reification has to be understood to refer to [triple tokens] [...].

Pat Hayes[120]

The semantics of RDF standard reification, although not normative, is well specified in [Hay04b]. It also is well integrated into the model-theoretic semantics and, arguably, quite sensibly adapted to the Semantic Web. However, it doesn't meet common demands, it doesn't fit well into popular intuitions and its subtleties

[119]In a post to the Semantic Web mailing list, `https://lists.w3.org/Archives/Public/semantic-web/2010Jan/0181.html`

[120]In a post to the Semantic Web mailing list, `https://lists.w3.org/Archives/Public/semantic-web/2018Jul/0025.html`

can be rather surprising, adding only to the dissatisfaction spurred by the syntactic verbosity of the reification quad. In practice the formal semantics is often misused, modified or outright ignored. The predominant motivation for introducing a reification mechanism in RDF was to record provenance of statements retrieved from the web. This seemed like an indispensable facility when integrating data from near and far, especially from unknown or not necessarily trustworthy sources.

2.7.2.1 Token, not Type

To this end the reification quad describes a triple. However, it doesn't identify the triple so described as an 'abstract' type considered as a grammatical form. Instead it identifies one such token of the triple, an occurrence — as warranted by the provenance use case.

If the reification describes a statement and a statement fitting that description occurs in the same graph one might assume that the reification describes that occurrence. Also, if the subject of the reification quad is an IRI addressing a document or named graph that contains a triple as described one might assume that the reification quad identifies that token. This is however only a possible interpretation but not prescribed by the RDF semantics. Fixing this interpretation would require a semantic extension.

2.7.2.2 Speech Act, not Artefact

This semantics is not well suited to describe a statement token understood as an artefact. Although the fact that the reification quad identifies a token, not the abstract type of a statement, would fulfil one important prerequisite for annotating a specific statement token, the most important link is missing: no reference to the syntactic representation, the triple as written out in a serialisation, is provided. RDF can't address that syntactic layer as its model theory is only concerned with the realm of interpretation. What reification refers to is the act of making an assertion, or, in other words, the event of stating rather than the statement itself.

2.7.2.3 Set, not Multiset

In fact the model-theoretic semantics of RDF doesn't support a notion of a specific triple as it becomes visible in a serialisation. Triples in the RDF sense are abstract, just as the graphs containing them are themselves defined as a mathematical abstraction: sets. This quite naturally follows from the main purpose of the Semantic Web: integrating data. To that end the meaning of a statement is the most important aspect that there is to it, not the who/where/when and whatever else there might be to add. The meaning of a statement doesn't change, no matter how often or by whom it is stated.

The reification vocabulary allows for describing multiple occurrences of the same statement by giving them different identifiers. It might seem like this would circumvent the set-based semantics of RDF, but that would be a misconception. Those identifiers don't refer to the triple they describe as an abstract triple but rather to occasions that the statement represented by that triple was stated. Each such occasion, or token, or occurrence, does indeed represent a distinct resource. That's why there can be multiple occurrences, but only one triple of that type.

2.7.2.4 Mention, not Fact

Since RDF standard reification doesn't refer to the triple itself it stays clear of some dangerous semantic territory. In particular, it enables neither paradoxes nor endless recursion through self-reference — the pitfalls of meta-modelling discussed above in Section 2.6.3. It also can't be used to qualify statements — again, because it can't refer to them — which would put the strictly monotonic nature of RDF in jeopardy.

A further property of the reification vocabulary is that it doesn't assert the triple it describes. Describing a triple occurrence and actually asserting a triple (as a type) are two completely distinct activities. Consequently, there exist no entailment relationships between a triple and a reification of it.

This characteristic can be used to create so-called *unasserted assertions*, a somewhat abusive application of RDF standard reification. It comes in handy when one needs to document or comment on a statement without endorsing it. Discussions on RDF-star revealed that it is a quite popular technique in practice.[121]

2.7.2.5 Provenance, not Application Control

The semantics of RDF standard reification is well suited to document occasions that a triple was stated, e.g. that Alice stated it on Monday and/or that Bob stated it under the influence. It can also express that some document or named graph contains that statement as a token. But, again, this shouldn't lead to the assumption that it can refer to the triple itself. In the following example both reification quads describe the same triple, and also in the same graph, but they refer to different speech acts, reifying different occurrences:

```
_:x rdf:subject :Alice ; rdf:predicate :knows ; rdf:object :Bob ;
    :inGraph :G1 ;
    :source :Carol ;
    :trust :High .
_:y rdf:subject :Alice ; rdf:predicate :knows ; rdf:object :Bob ;
    :inGraph :G1 ;
    :source :Doug .
```

One could now add further assertions to record the when, where etc of these occurrences. It is just important to understand that such provenance doesn't refer to the statement as a type or even the syntactic entity — although it might

[121]See e.g. https://lists.w3.org/Archives/Public/public-rdf-star/2019Sep/0082.html and https://www.ontotext.com/blog/rdf-star-implemented-in-graphdb-and-graphites-access-control/

look that way if the statement in question was authored only once or if one is inspired by intuitions, circumstances or an operational mindset to jump to such a conclusion. The latter has probably been driving the design of the 'id' attribute in RDF/XML.[122]

This semantics does not support working at the syntactic level, close to the metal of an application. As its main author commented, "the problem with a semantics which *would* support [syntax level] provenance is that it would have needed to introduce a semantic notion of a single published instance of a triple, and no such notion had ever been considered. It would have introduced a large and not well-understood extension to the basic semantic framework of RDF, one that was rejected by the RDF WG as being too large a change to RDF. One can get a sense of what is needed by the subsequent work by [Car+05a] on the semantics of named graphs. (If anyone decides to try to read this, notice that the first thing it does is to distinguish between a named graph and an RDF graph.)."[123]

2.7.2.6 Reference, not Quotation

Correspondingly RDF standard reification is not a form of quotation: it is not concerned with the precise wording used. Instead, it refers to what the statement means in the realm of interpretation.[124] This is equivalent to indirect speech, like e.g. "Alice said that she'll go to the movies later." and is different from precise quotation like "Alice said: 'I plan to go see a movie tonight.'." Or, as the RDF recommendations puts it: if the original statement used some IRI to refer to Mount Everest then the reified statement also refers to Mount Everest, not to said IRI. The reification quad can be read intuitively as saying "This piece of RDF talks about these things" rather than "This piece of RDF has this form".

2.7.2.7 Nomenclature: Reifications, Tokens and Occurrences

The RDF Semantics recommendation[125] calls the reification quad a 'reification' and its subject a 'reified triple'. That wording might be a bit confusing as intuitions would rather suggest that the triple that is reified by the quad would be called the reified triple, so it's good to be aware of this. Also, the recommendation uses the term 'token' to refer to instances of a triple in e.g. a concrete syntax document. It might be more appropriate to speak of 'occurrences' instead of 'tokens', as the Semantics specification does, as in the context of the Semantic Web the focus of interest doesn't lie so much on the rendering of triples on a screen or printout but rather on their occurrence in an RDF resource itself.[126]

---

[122]See Section 2.4.3

[123]`https://lists.w3.org/Archives/Public/public-rdf-star/2020Jun/0010.html`

[124]In the words of the RDF 1.1 Semantics specification, [HP14], it "describes the relationship between a token of a triple and the resources that the triple refers to", `https://www.w3.org/TR/rdf11-mt/#reification`

[125]`https://www.w3.org/TR/rdf11-mt/#reification`

[126]See the article about "Types and Tokens" in the Stanford Encyclopedia of Philosophy, `https://plato.stanford.edu/entries/types-tokens/`, for a more complete discussion of the

It might seem sensible to stay with the wording used in the RDF specifications as much as possible, as this may help to avoid confusion, but using the term 'occurrence' has become quite common as well. This text will use the two terms interchangeably, with a slight preference towards 'occurrence'.

2.7.3 Named Graphs

Named graphs are discussed in multiple sections of this work: Section 2.5.3 introduces the syntactic device, Section 2.4.4.3 disambiguates some graph-related terminology and Section 4.2.5 discusses other proposals to make use of this syntactic instrument. This section however will elaborate on the status quo of named graphs in RDF. It centres around the work of the RDF 1.1 Working Group (WG) which did not only define the current status but did discuss the issue at great length.

The standardization of means to group statements into graphs for various purposes like administration of sources, contextualization of information and separating structure from content was already discussed in the RDF 1.0 Working Group pre-2004. However, no clear consensus could be reached and for lack of practical experience the issue was eventually postponed as a possible work item for a next round of standardization.

Subsequently, [Car+05a] published a proposal that coined the label Named Graph, defining syntax and model-theoretic semantics of a mechanism to group sets of RDF statements and refer to them in unambiguous ways. This proposal and its semantics is discussed — along with other approaches — in more detail in Section 4.2.5 below but the gist of it is that a graph and its name are tightly connected. A graph name exclusively names a graph, nothing else, and two graphs composed of an identical set of statements but with different names represent two distinct named graphs. The statements in those graphs are referentially opaque and do not support any entailments, e.g. IRIs do not co-denote. In effect this semantics formalizes speech acts, differentiating between different occurrences of a graph and accounting for the exact 'wording'. Its envisioned use cases were e.g. provenance control, trust management and other applications that would benefit from a more Closed World oriented approach than what the integration-focused RDF semantics provides.

Named graphs were implemented in numerous systems and standardized, as part of an enclosing concept of datasets, in RDF's query language SPARQL [PS08]. SPARQL however is not based on a model theory but employs a much looser, operational semantics. Implementations and applications mostly picked up the named graph syntax but didn't implement the semantics as defined by [Car+05a], instead following their own and often application-specific intuitions.

Specifying named graphs was again on the agenda for RDF 1.1, and with a very high priority at that, and the RDF 1.1 WG formed in 2011 did spend a great amount of time and energy to achieve a consensus among the various stakeholders.

difference between tokens and occurrences

Yet it proved, again, impossible to standardize a semantics of named graphs — this time around not because there wasn't yet enough practical experience, as the RDF 1.0 WG had cautioned, but because existing implementations had diverged too far from each other and from the initial proposal by [Car+05a] and vendors weren't willing to give up on their specific approaches. As a result the RDF 1.1 specifications do little more than acknowledge what SPARQL 1.0 specified as named graph syntax years earlier, leading to the status quo that RDF also in its third and current round of standardization[127] provides no semantically sound meta-modelling mechanism.

The RDF 1.1 WG however did publish a Note [Zim14] that documents its discussions and explains the different positions it wasn't able to consolidate. To illustrate the spectrum of approaches and existing implementations the Note describes various possible interpretations of what the pair `(name,graph)` denotes, with sometimes subtle, sometimes far-reaching consequences:

The pair (name,graph) itself This is the definition proposed by [Car+05a]. It ensures that the graph is understood as an occurrence. The name is not merely a label but part of the identity of the named graph.

The RDF graph in the (name,graph) pair Here two named graph pairs with different names but composed of the same set of triples can be considered equal. The graph itself determines the identity and is understood as a type.

A container for a graph In this interpretation the graph is a mutable element and the name is just an address from where it can be retrieved. This corresponds to the notion of 'source' as discussed above in Section 2.4.4.3

A resource on the web This overloads the graph name to refer to multiple things. Used in SPARQL's `FROM` or `FROM NAMED` clauses it addresses a graph, but at the same time it addresses an information resource on the web. It is undefined if the graph describes the web resource or represents the RDF contained in that resource (e.g. some RDFa content), etc.

An arbitrary resource The graph name might be an IRI referring to a thing outside the web, e.g. a person, a monument or a theory. In this case the graph might be in some special relationship to that thing like describing it, belonging to it, recording its temperature etc.

The deductive closure of the graph inside the pair This interpretation alludes to the fact that RDF has a formal semantics and defines various reasoning regimes whose purpose is among other things to make explicit information that is already contained in the data implicitly.

Etc. The Note lists more variants, ending with 'etc'.

From this mix of perspectives and concerns the Note identifies two main topics about which no consensus could be achieved:

- how the graph name relates to the graph it names, and
- how the statements that the graph contains are to be interpreted.

[127]Counting also the 1999 version [LS99], although the 2004 specification is called "RDF 1.0"

2.7.3.1 Naming the Graph

It is one of the foundational axioms of web architecture that an identifier e.g. an IRI should identify exactly one thing. This makes immediate sense on the web: how else could the addressing and retrieval of a resource be reliably implemented? It also makes a lot of sense on the Semantic Web since how would one be able to make statements about something if the identifier for that thing points to something else as well. Graph store implementers and users however often repurpose the syntactic sugar that graph naming provides as a shortcut with implicit, application-specific semantics. This introduces IRI collisions for the sake of saving one more triple that would declare explicitly the nature of the connection between the graph and the thing the name refers to.

[Hay12b] gives an account of the different ways in which applications overload the graph name with other purposes than naming. In his words:

> *a datastore can be:*
>
> - *a way to name some graphs*
> - *a way to keep track of versions of a graph*
> - *a way to keep track of time-varying data (the graph 'name' encodes times)*
> - *a way to distinguish data from meta-data (in the default graph)*
> - *a way to distinguish data depending upon its provenance or source (the graph name denotes the source)*
> - *a way to distinguish data depending upon its topic (the graph name denotes the topic)*
> - *a way to keep data sorted into groups which share a common meaning for the URIs in the graph (an 'island')*
> - *any combination of the above; or sometimes one of the above, sometimes another*
>
> *and various other things*

— the main distinction being that in the first case the graph name actually and exclusively *names* the graph whereas in all other cases the name rather *labels* the graph but refers to something else. In those cases the nature of the relation between label and graph is not made explicit in the data. It might be encoded in application logic, documented out-of-band or 'intuitively' intelligible to a human reader, but it is not explicitly asserted, as it arguably should be.

As [Hay12b] notes "This is a problem." It makes it impossible to soundly refer to the graph itself: e.g. labelling a graph `<paris.com>` because it contains statements about a trip to Paris may seem perfectly fine. It does indeed work alright for querying as the syntactic constraints of SPARQL, specifically the necessity to use its `FROM` or `FROM NAMED` clause to address a graph, reliably disambiguate querying for a graph of that name from querying for statements about Paris. It breaks however when one desires to attribute a graph. It might be obvious to human readers that a statement like `<paris.com> :created`

`:lastSummer` is referring to the graph, not the city nor the website. To a machine however the referent of this annotation is ambiguous. Named graphs as standardized in SPARQL and RDF don't allow to reliably state assertions about graphs as they provide no way to disambiguate naming from labeling semantics.

The WG developed a small vocabulary, documented in the WG wiki[128], to express explicitly if a graph name is indeed a name, not merely a label, and can safely be used as reference to the graph:

```
{ :Graph_1 a rdfx:Graph . }  # the default graph of a dataset
:Graph_1 { :a :b :c . }      # :Graph_1 names this graph
:Graph_2 { :d :e :f . }      # :Graph_2 merely labels this graph
```

Here, stating (in the default graph of a dataset) that the graph named `:Graph_1` is of type `rdfx:Graph` pins down its naming semantics. A more comprehensive solution could introduce a graph naming vocabulary to declare per dataset if graph names are indeed proper names or, in case they are labels, how the relationship between graph and label is defined. Assuming that in most datasets all graph names are governed by the same semantics solving this problem could in practice often be as simple as adding a single statement to the default graph expressing the graph naming semantics for all graphs in the dataset.

Many proposals have been presented that operate under the assumption that graph naming semantics can be standardized to one specific meaning, e.g. provenance. The many different semantics found in use 'in the wild', e.g. as presented above by [Hay12b], should illustrate convincingly that the hope for such a common agreement is not realistic. Being able to group statements into graphs is a very basic information modelling need. As long as RDF only provides one syntactic mechanism to implement such needs, that mechanism will be used in widely diverging ways.

2.7.3.2 Interpreting the Graph

SPARQL organizes named graphs in so-called *datasets*. A dataset contains a collection of graphs: a default graph with no name at the root of the dataset, and any number of named graphs. This brings up questions about the relation between default graph and named graphs. The Note discusses different ways in which named graphs may contribute to the truth of the containing dataset: they might all contribute equally, or they might not contribute at all, with all but the default graph being mere comments. Between these extremes many different arrangements can be defined. This can also be understood as the issue if or under what conditions a graph is asserted or not. Some possible semantic arrangements are:

Universal The default graph and named graphs all contribute to the meaning of the dataset in the same way. The named graphs mechanism is merely an administrative instrument with no meaning.

[128]`https://www.w3.org/2011/rdf-wg/wiki/Graphs_Design_6.1`

Contextualized Named graphs represent different 'contexts' or 'possible worlds' and the truth of the statements they contain depends on how those contexts are defined and evaluated. The default graph may constitute a 'global context' and/or hold metadata about the named graphs defining those contexts.

Hypothetical The named graphs are not only contextualized but also not automatically part of the dataset. This allows the graphs to be contradictory without implicating the whole dataset.

Speech Act The named graphs are interpreted in exactly the way they are stated but support no further inferences.

Quoted The named graphs are treated as literals. Their meaning is opaque to RDF and therefore has no effect on the truth of the dataset.

These different semantics can also be described by different choices along the following three axes:

Type or Occurrence Does the graph name refer to a particular occurrence of the graph or does it refer to all graphs of that type (all graphs containing that same set of statements)?

Interpretion or Quote Are the graph's statements interpreted like 'normal' RDF statements, enabling inferences like subtype relationship, co-denotation etc, or are they quoted for purposes like documentation of sources and processing? In more technical terms: is the graph referentially transparent or opaque?

Global or Local Do statements — or rather, more precisely, the IRIs they are composed of — mean the same everywhere, inside a graph as well as on the 'global' outside, or are they contextualized and qualified by the graph's attributes and is their interpretation local to the graph?

Some of these topics have already been discussed in Sections 2.6.2.5, 2.6.2.8 and 2.7.1 above but a second look seems appropriate given the relevance of the named graph approach to meta-modelling in RDF.

Type or Occurrence Referring to graphs as types means differentiating them from their source, context etc. — treating them exclusively for what they express. Referring to them as occurrences takes the act of expressing them into account: who, where, when, why, etc. A graph understood as a type is the same everywhere it occurs. Understood as an occurrence it is a different graph each time it occurs. Both approaches have strengths and weaknesses. Obviously the same set of triples itself is the same set of triples everywhere, period. But just as obviously it always matters who published it, when, under which circumstances etc. But the distinction is — as is rather the norm than the exception in knowledge representation — not absolute: both approaches can be transformed into each other. Annotating a graph type with provenance information creates a new graph type: the type of a specific occurrence of the original graph. Creating a reference to a graph occurrence as type creates a graph type, as occurrence. And not to forget: a name, if it's not strictly functional, always has some meaning, subtly mutating a type into an occurrence.

Interpretation or Quotation RDF statements normally are treated as syntactic representations that are interpreted as referring to things in the universe of discourse. E.g. the terms `ex:car` and `yz:auto` could both be interpreted as referring to the same concept that we know from the real world: a 4-wheeler, often with 4 or 5 passenger seats and either combustion engine or electric drive train. Referential transparency refers to this ability to communicate about meanings without being forced to use the exact same terms. It is essential to building robust data integration mechanisms, RDF's primary purpose. Quoting those terms as strings "ex:car" and "yz:auto" however preserves the exact details of a statement: the vocabularies used, capitalization, etc. — no inferencing applied. This allows tight control over data integration, capturing precisely what a graph conveyed in isolation, outside the ever-volatile NUNA, OWA, entailment-enabling space of interpretation that the semantic web is. However to an RDF processor it is then just a string and its meaning is totally inaccessible and opaque.

Global or Local A graph can be understood as a part of the giant global graph composed of *all* RDF statements, without any identity of its own. In that case the named graph has no influence on the meaning of the statements it contains. A more pedantic way to put it is that the IRIs used in the graph have the same meaning as in any other statement or graph, everywhere else on the Semantic Web. If however the named graph is understood as an entity in its own right, possibly equipped with attributes of its own, then it has itself a meaning. That meaning reflects on the meaning of the statements it contains, and the IRIs that those statements are composed of. If the graphs's attributes are strictly administrative they may not change its meaning very much. If however they state that e.g. the graph is describing a certain time period, then the statements and in consequence the IRIs they are composed of are also describing just a certain time period. That differentiates the meaning of those IRIs in the local named graph from what they mean in the global graph.[129]

Possible Combinations The Note presents different combinations of naming and interpretation semantics in more detail, discussing overloaded naming practices, issues specific to the interactions of default graph and named graphs in a dataset, named graphs as occurrences or types, interpreting or quoting of graphs and correspondences to established best practices in SPARQL.

RDF 1.1 WG Graph Task Force The WG's Graph Task Force (Graph TF) analysed use case and developed various proposals to cover all needs and

[129][Hay12c] and [Hay12a], presented below, elaborate on this idea.

semantics found in already existing implementations. They are documented in the Graph TF's wiki[130] and illustrate not only to what length the WG went to find a solution but also how intricate the problem is.

[Zim12b] proposes a contextualized approach with graph-local semantics in line with the way SPARQL ASK queries interpret graphs[131], and [Zim12a] adds an optional quoting semantics for individual graphs. [Hay12c] and [Hay12a] extend that proposal towards a more general, configurable contextualization extension to RDF in which the approach outlined by [Zim12b] and the original RDF 1.0 semantics would constitute the local and global extremes.[132] [Haw12] and [Haw+12] concentrate on closing the gap between the rather abstract set-theoretic model of RDF and the practical requirements of dealing with mutable data sources. [Her10] explores the idea of graph literals as a means to express graphs without actually asserting them.[133]

More insight into the WG's discussions can be gleaned from its meeting minutes and eMail discussions which are archived online as well and provide some valuable backgrounds. If one just wants to tip a toe into that pond, maybe try `https://lists.w3.org/Archives/Public/public-rdf-wg/2012May/0065.html` or `https://www.w3.org/2011/rdf-wg/meeting/2011-10-12#Named_Graphs`.

A 2×2 Matrix Hayes, the author of the original RDF 1.0 semantics from 2004 and of the original named graph semantics by [Car+05a], and Zimmermann, the author of the RDF 1.1 Note on Named Graph semantics from 2014, in 2020 had a brief exchange on the Semantic Web mailinglist about the essence of the issue.[134] They favour distinctly different choices for certain aspects of a named graph semantics as discussed by [Zim14], especially in Example 3.3. Hayes advocates a named graphs semantics of quoted occurrences whereas Zimmermann prefers to define them as interpreted types.

Hayes envisions use cases like warrants and versioning that require representation of statements faithful to their original form, without any reasoning applied, especially without co-references to syntactically different representations of equivalent meaning. This semantics takes great care to treat named graphs as occurrences: important is not only what they say but also the fact that they say it. Two graphs containing the same set of triples but with different names are considered different utterances, different speech acts. This is a very practical approach to an obvious problem: RDF is from the ground up optimized to integrate triples but it provides no semantically sound means to partition the

[130] `https://www.w3.org/2011/rdf-wg/wiki/TF-Graphs`

[131] See [Zim14], Section 3.8

[132] See also an accompanying slide deck at `https://www.slideshare.net/PatHayes/rdf-with-contexts`

[133] A related approach is documented at `https://www.w3.org/2011/rdf-wg/wiki/TF-Graphs/RDF-Quadless-Proposal`

[134] See Zimmermann's mail, `https://lists.w3.org/Archives/Public/semantic-web/2020Jul/0232.html`, and Hayes' response `https://lists.w3.org/Archives/Public/semantic-web/2020Jul/0233.html`

knowledge space and control integration. Such tasks are left to applications and their semantics defined out of band. But the approach favoured by Hayes is rather strict: while the use cases brought forward are well justified, there are many more reasons to group triples and few of them require such syntactic fidelity. Yet the proposed semantics requires to explicitly 'accept' a graph before standard semantics and the usual entailment services like co-denotation are available. In practice not many applications of named graphs found it worth the effort to follow this aspect central to Hayes' semantics.

Zimmermann's take is more in line with the integration focus of RDF, emphasizing what a graph means. He envisions the use of graphs to e.g. control reasoning over different contexts or to manage inferences from eventually contradicting or incomplete graphs. This asks for a semantics where graphs are not only referentially transparent and readily available for entailment procedures, but also are treated as types, so that what they mean is not influenced by their name or other attributes but only be the set of statements they contain. To realize the full potential of reasoning on the Semantic Web it is necessary to separate and manage statements, and interpreted graphs can provide the necessary tooling. The name or attributes of those graphs however would only complicate the matter. This makes it desirable to define graphs as types, not occurrences, and treat their names as labels without meaning. Representing the act of assertion would require an approach similar to RDF standard reification, minting a new identifier for each occurrence, each speech act. That would address the type/occurrence distinction but still not support faithful reproduction of syntactic, un-interpreted state.

Obviously two more combinations are possible: interpreted occurrences and quoted types. Quoted types are what Zimmermann and others propose to implement with a new construct: an RDF literal datatype as outlined by [Her10].[135] Interpreted occurrences is what could be understood as underlying the "intuitive" semantics of named graphs in SPARQL.

SPARQL's Social Semantics SPARQL was the first specification to standardize named graphs and it defines them according to the needs of database administrators. As SPARQL is defined not by a model theory, but with operational semantics, aspects of meaning that are critical to RDF semantics are left undefined. It does however make sense to investigate what model-theoretic semantics could be gleaned from the usage of SPARQL ìn practice.

Along the three axes defined above — type or occurrence, interpretation or quotation, global or local — SPARQL doesn't take a definitive stance. It therefore seems plausible to assume that SPARQL's semantics should be understood to be closely aligned to that of RDF and only carry as much semantics as a database administration would demand.

Statements in RDF are referentially transparent. Absent any indication to

[135]And also what RDF-star proposes as semantics for quoted triples - more on that below in Section 4.2.4.5

the contrary it seems only natural to assume the same for statements in named graphs.

Likewise there is no reason to assume that IRIs in a named graph should refer to something different than outside of it, settling the question of contextuality to the globalist default.

Trickier to answer is the question if intuitions default to graphs as types or occurrences. In general this distinction tends to be quite fluid in practice. However, two graphs containing the same set of triples but carrying different names are by common intuition considered different graphs. This reflects one of the main motivations to introduce named graphs — keep sources separate — as well as the importance and weight that names generally have. The distinction between type and occurrence becomes irrelevant when there can be only one occurrence per type.

The naming problem could be argued in a similar way: as graphs serve to keep statements apart, the dominant purpose of a graph name is to identify a certain set of statements. In that sense it clearly is a name. The `FROM` and `FROM NAMED` syntax not only disambiguates if a name is used to address a graph but also makes this the most prominent use of the name in SPARQL. Consequently the most basic and intuitive relation between graph name and graph in SPARQL could be interpreted as one of naming, irrespective of further information the name may encode and other resources it may refer to. So while confusion introduced by overloading the name is of course a problem and proper disambiguation of different uses of the name needs further attention, the social semantics seems clear enough.[136]

The results of this analysis is a sort of de-facto semantics that understands named graphs in practice as predominantly referentially transparent occurrences with global meaning and names that refer to the graph (but possibly also to something else).

SPARQL Graph Identification The SPARQL 1.1 Graph Store HTTP Protocol[137] provides "a non-normative suggestion for HTTP operations on RDF graphs which are managed outside of a SPARQL 1.1 graph store", i.e. when accessing an RDF database over the internet. It defines ways to address named graphs and differentiates between direct and indirect identification. In direct identification the IRI of the database's HTTP interface is combined with the path that leads to the named graph in question, e.g. `http://example.com/rdf-graphs/employees`. The specification notes that "in using an IRI in this way, we are not *directly* identifying an RDF graph but rather the RDF graph content that is represented by an RDF document, which is a serialization of that graph." Annotating a graph via direct access would therefore not annotate the graph itself but the

[136]Different datasets using the same IRI to name different graphs may be seen as a problem. However, SPARQL provides means to disambiguate those named graphs per dataset. The full name of a named graph in such a scenario has to be understood as a combination of the dataset IRI and the graph name IRI.

[137]`http://www.w3.org/TR/sparql11-http-rdf-update/`

statements it contains. No reasons are given for this arrangement, which seems to deviate from the usual practice of annotating named graphs in the default graph of a dataset with administrative metadata like provenance. On the other hand the RDF 1.1 specification requires graph names to be either IRIs or blank nodes. SPARQL indirect graph identification offers support for query identifiers to encode such IRI graph names in the path of a database interface, e.g. `http://example.com/rdf-graph-store?graph=http%3A//www.ex.org/graph`. Graphs that are named with blank nodes are of course inaccessible from the outside.

SPARQL Self Reference A pair of angular brackets, `<>`, in the default graph or a named graph refers to the enclosing dataset. However, a similar device to refer to the enclosing named graph from inside a named graph is not provided.

2.7.3.3 Configurable Semantics

The failure of the WG to standardize named graph semantics was met with some frustration.[138] There is probably not only one single reason why it failed. Differing implementations and stakeholders unwilling to compromise might have played a role but it was also apparent from the start of the Working Group that some wanted the phase of bootstrapping of the Semantic Web via standardization at breakneck speed to end, and consolidate the standardization process in order to encourage actual adoption by industry. Named graphs and the underlying problem of partitioning the knowledge space was unfinished business from RDF 1.0 standardization and in the meantime no consensus had presented itself. Rather the space was cluttered with incompatible approaches. That could be interpreted both ways: as a sign that a specification has to go ahead of implementors, or that the issue was not ripe for standardization.

As named graphs are, apart from plain old documents, the only mechanism that RDF provides to group triples into sets and impose some order on the giant global graph, it might be considered precarious to equip them with more than the most standard semantics. Leaving the semantics of graph naming undefined has its advantages too as it leaves practitioners a certain amount of wiggle room where they can manage data without having to care about subtle differentiations of meaning. Sometimes it's better to say nothing than to say the wrong thing because one is required to say something. And given the installed base of RDF systems and data it may be too late anyway to introduce any changes to named graph semantics.

According to [Zim14] the RDF 1.1 WG considered "to define several semantics, among which an implementation could choose, and provide the means to declare which semantics is adopted. This was not retained eventually, because of the

[138]See e.g. Jeremy Carroll's formal objection, `https://lists.w3.org/Archives/Public/www-archive/2013Oct/att-0035/fo.html`, comment, `https://lists.w3.org/Archives/Public/public-rdf-comments/2013Jul/0021.html`, and replies e.g. by Pat Hayes, `https://lists.w3.org/Archives/Public/public-rdf-comments/2013Sep/0008.html`

lack of experience, so there is no definite option for this." Section 7 will discuss such vocabularies.

Even considering all the difficulties the WG faced it still seems disappointing that not even a small vocabulary was standardized — leaving the basic naming syntax and informal semantics of named graphs untouched but adding a layer of expressivity available on demand that would enable the prudent user to make explicit which semantics is employed.

2.7.4 N-ary Relations

> *N-ary relations are everywhere in human interaction. Underlying natural language verbal structures, in relational database tables, when recognizing or talking about events, when interpreting news, when making medical diagnoses, interpreting legal norms, controlling situations, or simply preparing a coffee. Most of the time, more than two entities are bound together, and we cannot easily represent that boundary with just unary and binary predicates.*
>
> Aldo Gangemi and Valentina Presutti[139]

Many relations do not fit well the binary model because they relate more than two actors, are composed of a primary value and multiple secondary attributes, require contextual detail and background or come with administrative metadata. N-ary relations are the only semantically sound instrument that RDF provides to model anything more complex than a simple relation between two nodes. They do this of course in the RDF-way, by means of the blank node. Given the premise that there is no clear-cut distinction between data and metadata, meta-modelling also has to resort to blank node heavy constructs if it wants to stay within the confines of the basic, binary-relation based RDF formalism.

The basic structural elements of n-ary relations have been discussed in Section 2.3 above. How to effectively use and compose them is a different story though. More involved n-ary relations quickly turn into unwieldy agglomerations of triples with lots of blank nodes that only serve structural purposes. The resulting structures have no clear beginning or end, no well-defined entry points and boundaries. There exist no standardized rules for which links have to be followed, nor which attributes to include when gathering information about a subject of interest. This makes n-ary relations hard to comprehend and navigate for users unfamiliar with the data. Since RDF doesn't allow blank nodes in the predicate position it is impossible to qualify properties, which can lead to even more long-winded and ambiguous constructs.

From a user perspective what really sets apart meta-modelling techniques as discussed above — i.e. reification and named graphs — and n-ary relations is that the former introduce some kind of fix point like a statement identifier or graph name. That fix point is then the nucleus around which the complex object

[139][GP13]

is built, and from which it can be recognised, explored and comprehended. An n-ary relation doesn't provide an equally robust sense of object-like appearance and identity.

2.7.4.1 Types of N-ary Relations

[T]he potential application scope of the logical n-ary pattern is very broad. The n-ary pattern is often considered as an ad-hoc workaround and thus it is frequently used in an arbitrary fashion, lacking any design rationale. Various modelling solutions are possible, especially since RDF(S), as an epistemological KR language, should be neutral regarding metaphysical and ontological assumptions. One may, for example, treat roles as binary properties or as instances of classes, making ontological commitments that remain implicit. Without any explicit design rationale, reaching even a partial agreement on the conceptual level is unlikely.

Johannes Trame, Carsten Keßler and Werner Kuhn[140]

My other major pain is N-ary predicates. Sure, we have the W3C note on them: `http://www.w3.org/TR/swbp-n-aryRelations/` *However, this is more a discussion of the benefits of several different approaches. It doesn't define a standard, and doesn't provide any semantics. Personally, I'd be happy to see a standard which told everyone to build n-ary predicates some particular way (even if it's not ideal for a given scenario), and provided semantics that everyone could rely on.*

Paul Gearon[141]

Numerous works provide examples and possible usage patterns for n-ary relations and try to develop categories and systematic treatments of the topic. Yet, as [TKK13] express in the above quoted article, application areas and usage patterns vary widely.

[Noy+06] in a note for the W3C propose best practices for n-ary relations on the Semantic Web on the basis of four different use cases, [Zar09] discusses the design patterns proposed by [Noy+06] and relates them to earlier works on n-ary relations predating the Semantic Web. [GP13] compare seven different n-ary patterns along several axes. [TKK13] categorize n-ary relations into two types: Davidsonian and Neo-Davidsonian n-ary relations. Davidsonian n-ary relations are so named after works by Donald Davidson, namely the 1967 essay "The logical form of action sentences"[142], whereas Neo-Davidsonian n-ary relations are a generalization of the Davidsonian approach proposed by Terence Parsons in 1990[143]. This categorization informs the following treatment below.

[140][TKK13], references omitted
[141]In a post to the Semantic Web mailing list, `https://lists.w3.org/Archives/Public/semantic-web/2010Jan/0161.html`
[142][Dav01]
[143][Par90]

But even identifying and clearly describing those patterns is not necessarily enough. RDF doesn't provide many instruments to express which form an n-ary relation is intended to have. Such rules, or even just well-established conventions, might at first put a burden on some practical applications, but in the longer run would benefit everybody as they could make it much easier to standardize retrieval operations, navigate unknown data sources, and even support the process of authoring itself by providing proven templates. It seems that work on such guidelines was planned by the W3C but not concluded.[144] Design patterns and shape languages, discussed below in Section 4.1, aim at this aspect, although with varying success.

Davidsonian - Annotated Relation The Davidsonian pattern interprets n-ary relations as a primary relation between a subject and an object and secondary attributes on the object or the relation as a whole. In the W3C note on n-ary relations [Noy+06] demonstrate three possible sub-categories. They all have a primary subject but differ on the way the object is structured:

one primary object value and secondary attributes Those secondary attributes might either
- refer to the main object value (the note gives as an example a primary medical diagnosis with an accompanying secondary probability value) or
- add metadata like provenance or date of ingestion (such uses however are discouraged by the note as they would more soundly be expressed through RDF standard reification[145])

all equal objects The value space is composed of multiple equally important and self-contained values (in the note's example a patient's temperature reading, composed of the description 'high' and the tendency 'falling').

a list of objects The object value consists of several components of the same kind. This may constitute an unordered collection, like multiple people sitting in the same room, or an ordered list as in the note's example of United Airlines flight 1377 starting from the airport LAX, then stopping at DFW, and finally JFK.

Davidsonian n-ary relations and rdf:value RDF provides the `rdf:value` property to support the first case above: one primary object value and secondary attributes. The semantics of the `rdf:value` property is designed to support such a sense of object-like belonging when a multipart value has a clear primary component and secondary attributes, as discussed above in Section 2.4.1.

[144][Noy+06] states that "The task force plans to produce a suggested vocabulary for describing that a class represents an n-ary relation and for defining mappings between n-ary relations in RDF and OWL and other languages. A note on this vocabulary is forthcoming.", but no such work was published.

[145][Noy+06] does differentiate reification from n-ary relations as RDF standard reification is used to "put additional information about" some triple, e.g. its source, while n-ary relations "do not usually characterize the statement but rather provide additional information about the relation instance itself."

The RDF 1.0 Primer[146] gives the following example:

```
exproduct:item10245  exterms:weight _:weight10245 .
_:weight10245        rdf:value      "2.4"^^xsd:decimal .
_:weight10245        exterms:units  exunits:kilograms .
```

The semantics is however not normative and the property isn't used much in practice. In the note by [Noy+06] it is mentioned only rather passingly. Established algorithms to automatically gather n-ary relations into objects, discussed below in Section 4.3, don't assign it any special significance. This seems like a missed opportunity to add more rigour and predictability to n-ary relations in practice, especially when their semantics is close enough to the binary relations that RDF is optimized for.

Davidsonian n-ary relations share a certain brittleness concerning change. As they are rather closely woven into the basic triple structure, adding an attribute later on may result in a change of topology that has the potential to invalidate existing queries, or worse. A well-known issue are statements that mean to refer to an occurrence but attribute the type because they don't expect more occurrences or because they are just too focused on the specific task at hand. Take for example Alice buying a car, suitably modelled as:

```
:Alice :buys :Car .
```

What if other information like the price or the car's colour has to be added? This requires re-modelling the data:

```
:Alice :buys [
    rdf:value   :Car ;
    :color      :DarkGray .
]
```

What if she buys a second car? this again leads to a change in triple topology, possibly invalidating existing queries or shape constraints.

Neo-Davidsonian - Objectified Relation The Neo-Davidsonian pattern generalizes the Davidsonian approach to n-ary relations that are composed of multiple nodes of equal significance, possibly without even clear subject and object roles. Here the n-ary relation itself is turned into the subject, sometimes also called the class, of the n-ary relation, to which all actors, values and other components are attributed. This approach has minimal semantics and can express just about anything, much like a relational table in a RDBMS.

[Noy+06] give as example a purchase — John buys a 'Lenny the Lion' book from books.example.com for $15 as a birthday gift — modelled as an eCommerce process:

```
:Purchase_1  a  :Purchase ;
    :has_buyer  :John ;
    :has_object :Lenny_The_Lion ;
    :has_purpose :Birthday_Gift ;
    :has_amount  15 ;
    :has_seller :books.example.com .
```

[146] https://www.w3.org/TR/rdf-primer/#rdfvalue

The RDF 1.0 Primer[147] provides the example of an address:

```
exstaff:85740   exterms:address     _:johnaddress .
_:johnaddress   exterms:street      "1501 Grant Avenue" .
_:johnaddress   exterms:city        "Bedford" .
_:johnaddress   exterms:state       "Massachusetts" .
_:johnaddress   exterms:postalCode  "01730" .
```

In the ‘Linked Data Patterns’ book [DD11] define a pattern for n-ary relations[148] and a subtype for ‘qualified relations’[149], explaining that “[i]n the Qualified Relation pattern the desire is to annotate a relationship between two resources, whereas in the N-ary Relation pattern the goal is to represent a complex relation between several resources.” Both patterns however implement the Neo-Davidsonian approach.

Symmetric relations can be modelled rather efficiently using the Neo-Davidsonian pattern as the following example[150] shows:

```
:Marriage1 rdf:type :Marriage ;
    :partner1 :man ;
    :partner2 :woman ;
    :startDate "2020-02-11"^^xsd:date .
```

This is a much clearer solution than the idiomatic RDF approach that would put one of the two partners in the subject position, calling for a second relation documenting the opposite direction and still not providing a clear way to annotate the relation with a start date. However, in contrast to the idiomatic approach, it doesn’t provide a direct relation between `:man` and `:woman` — something that the uninitiated user might, with some justification, expect.

Hayes puts the dilemma as follows:[151]

> *How to encode n-ary relationships in what is essentially a binary language (RDF properties) is an issue that goes back at least to the 1960s and arguably to Pierce’s work in around 1885, so there are plenty of ideas out there. If, as here, we want the n in ‘n-ary’ to be extendable, there is really only one good way to do it, which is to reify the ‘fact’ of the relation’s holding and connect it to its relational arguments (in this case, JohnDoe, Bar, 1996, 2002 and the rank number) by suitable binary relations, RDF triples in this case. In plain RDF that would look something like*

```
_:x rdf:type PresidencyOf .
_:x subject JoeDoe .
_:x object Bar .
_:x startDate 1996 .
_:x endDate 2002 .
_:x rankOrdinal 2 .
```

[147] https://www.w3.org/TR/rdf-primer/#structuredproperties
[148] https://patterns.dataincubator.org/book/nary-relation.html
[149] https://patterns.dataincubator.org/book/qualified-relation.html
[150] http://graphdb.ontotext.com/documentation/9.2/free/devhub/rdf-sparql-star.html
[151] https://lists.w3.org/Archives/Public/public-rdf-star/2020Sep/0018.html

> *And this `_:x` thing is variously called a situation, a state of affairs (SOA), an event, a history, an occurrence, a circumstance, a happening and probably a lot of other things depending on which background literature you find the idea described in. In defense intelligence work it is often called the 5-W because information is organized around these and the basic questions are What happened, Who was involved, Why it happened, Where it happened and When it happened.*
> *The trouble with this is not that it doesn't work (it's about the only way that does work) but that it requires simple binary facts to also be written in this peculiar way, so instead of the single triple*
>
> ```
> JoeDoe presidentOf Bar .
> ```
>
> *we would have to write the first three triples from the larger set. And nobody wants to have to do this for every simple fact.*

. . . which is exactly the problem. And as there is no definitive way to disambiguate unary, binary and n-ary relations, authors as well as users have to decide on a case by case basis which route to take, which pattern to deploy or query for. Especially when working with data from unknown sources this inherent vagueness can be very tedious and error-prone.

2.7.4.2 OWL and N-ary Relations

If that was a thing in Computer Science one could describe the relationship between OWL and n-ary relations as love and hate: from the perspective of OWL there are as many reasons to prefer n-ary relations over other approaches at meta-modelling and complex relations in RDF as there are against them.

N-ary relations are true to the binary formalism of RDF and as such are the only approach to complex modelling that is compatible with OWL. RDF standard reification as well as its OWL equivalent, OWL statement annotations, have no consequences on entailment and are semantically equivalent to mere comments on the statement. Named graphs as specified in RDF 1.1 have no model-theoretic semantics and are outside the reach of OWL. N-ary relations profit from the fact that all their expressive powers are based on blank nodes[152]. Meta-modelling in RDF is extremely constrained by the Open World Assumption and the consequential axiom that no statement may change the truth value of another one. N-ary relations in that respect are safe precisely because of their reliance on blank nodes: the mere assertion that "something exists" is extremely forgiving as it asserts not an iota more than the absolute minimum. One simply can't go wrong with stating that there is something, period. That semantic parsimony and elegance shines in comparison to the troubles incurred by reifying statements and naming graphs as discussed above. This blissful ignorance is the beauty and the horror of the blank node.

[152]Or specifically minted throw-away IRIs, but that doesn't make a difference in practice

On the other hand important constructs from RDF and OWL, like domain and range axioms and inverse property definitions, cannot be applied easily to n-ary relations, limiting their attractiveness for more than simplistic use cases.[153] Usability challenges have been mentioned already but specifically OWL-oriented n-ary design patterns like e.g. proposed by [Hoe09] and popularized by the Ontology Design Patterns initiative[154] tend to be very involved and quite hard to author, read and query alike.

2.7.5 Between a Rock and a Hard Place

Most things in life have more than two components, participants or vectors. Which of these constituents is primary, and which are secondary, is often not unambiguously distinguishable. Technical or administrative details may be easy to disambiguate from primary facts, but contexts, perspectives and multiple participants very often are not. Such differentiations are notoriously fluid: they may seem rather obvious in a given application domain, but are impossible to pin down in general and on a scale as wide as the Semantic Web.

RDF however is a binary formalism, and very strongly so: no qualification of statements is allowed, no standard way to extend binary to n-ary relations is provided, no semantically sound way to address, relate or even annotate an RDF statement is available.

In other words: RDF's basic formalism lacks expressiveness as well as extensibility. The reasoning apparatus of RDF(S) and OWL suffers from the same focus on binary relations and makes the problem even more profound.

The only extensibility mechanism that RDF has to offer are blank nodes: an essentially structural device with minimalistic semantics. The resulting constructs are rather free wheeling in nature. They have no boundaries, no identity, they just meander, and although built from binary statements they behave very differently from those, breaking with the idiomatic style that RDF is based on and that its reasoning extensions rely on.

The next chapter will give an overview of the problems this causes in practice.

[153]There are ways, involving property chain axioms, existential and universal restrictions, but they are not for the faint of heart.

[154]See e.g. an n-ary pattern at `http://ontologydesignpatterns.org/wiki/Submissions:N-Ary_Relation_Pattern_(OWL_2)`

Chapter 3

Demand

We need a away of distinguishing between things we said and we stand by, and statements we wish to discuss. This is going to be of primary importance on the Web in which information from many sources is combined. It is a fundamental part of language design.

Tim Berners-Lee[1]

This chapter addresses the complexities that arise when a simplistic formalism is put to use. It gives an overview of the problem space that has been covered in research, in prototypes and real world applications and into the range of shapes that complexity can take on the Semantic Web. This treatment is certainly not meant to be exhaustive — its intent is rather to illustrate the wide range of issues that have come up and the enormous breadth of approaches and proposals that have been brought forward to resolve them. One purpose of this section is to stress the complexity of the problem space and by consequence the importance of a principled and well-balanced approach that doesn't content itself with only a part of the picture or some low hanging fruit.

Over time the main focus of attention seems to have shifted back and forth: from complex integration of different sources on the web through contextualization to the more modest goal of enhancing data with metadata annotations as suggested by the rather pragmatic Linked Data initiative and then to contextualized integration again but this time in localized large repositories also known as Knowledge Graphs. Instead of trying to retrace these broader historic lines — which tend to dissolve the closer one looks anyway — this section puts spotlights on what seems to be the main aspects that drive research interests and application needs.

Investigations started either big, outlining all-encompassing solutions of various degrees of depth, or small, concentrating on pragmatic concerns and low-hanging fruit. Works on *annotation* often concentrate on rather one-dimensional, regular and ubiquitous dimensions like source or temporal and spatial properties.

[1]In a post to his blog 'Design Issues', `https://www.w3.org/DesignIssues/Toolbox.html`

Many other dimensions are explored as well, and also combinations of different dimensions are investigated but almost always the expressivity aspired to is put under some pragmatic limitation. On the other hand *contextualization* approaches tend to cover a broader array of perspectives including very irregular ones like viewpoints or compositions of contexts that describe situations, complex states and so-called *microtheories*. Of course, because those approaches put less limits on their expressivity upfront, they tend to more often run into problems in practice.

Applications of the Semantic Web effort have two prominent crystallization points as well: Linked Data and Knowledge Graphs. *Linked Data* represents a pragmatic approach to put into practice what more lofty and AI-inspired initial endeavours on the Semantic Web couldn't achieve at the required scale. The Linked Data effort however ironically largely fails in the 'Linked' department: it created a huge cloud of data silos that all follow the same principle and in theory are ready to be integrated, but in reality exist largely isolated from each other. *Knowledge Graphs* on the other hand tend to focus on the integration of data from different sources and can be quite successful at reaping the knowledge benefit from clever combinations thereof. They are however often designed as inhouse applications and mostly tailored to specific applications and use cases. Their architectures still lack the generality and robustness required to scale to the decentralized Semantic Web in general.

An issue not covered in this overview is that of native complex objects that surpass the barrier of individual triples, e.g. native list objects, complex OWL constructs as discussed by [PF02] or the plea by [AF19] for a shift towards n-ary relations. Those are discussed in more depth in Sections 2.4.2, 2.7.4 and of course in Part II.

It's not too often that the architecture of the Semantic Web as a whole has been discussed after the dust from the first round of standardization efforts settled in the mid-noughties. One such occasion was when an RDF 2 standardization effort was discussed[2] — which then turned into a decidedly more modest RDF 1.1 "Keep the core, pave the cow paths" exercise. The issue popped up again with surprising intensity when David Booth posted a proposal "Towards easier RDF" on the Semantic Web mailinglist in late 2018[3]. The results of that discussion are collected in a wiki on Github.[4] More recently [TH20] provided an overview of RDF as used in practice and some suggestions for an eventual effort towards RDF 2 and [Suc20] discussed the limited expressivity of RDF and explores related fields for possible future directions.

[2]https://www.w3.org/2009/12/rdf-ws/
[3]https://lists.w3.org/Archives/Public/semantic-web/2018Nov/0036.html
[4]https://github.com/w3c/EasierRDF

3.1 Investigations

The examples and subjects of study that researchers choose don't necessarily express practical needs but may just have been considered most suitable to make a specific point. Still they can give an idea of the problems and roadblocks that Semantic Web practitioners are facing. From the range of topics covered two stand out specifically: provenance and time. This may well reflect the needs of applications that do what a Semantic Web application is supposed to do: fetch and integrate data from sources near and far. On the other hand these are also rather ubiquitous dimensions that don't need any extra introduction and therefore make for good example scenarios. The labels are at times misleading and there is less differentiation between approaches to contextualization, annotation or any specific type of topic than one might expect: most works on contexts list the very regular and ubiquitous dimensions of time, space and provenance as examples while there are approaches that sail under the flag of annotations but cover very irregular perspectives like argumentative viewpoints. Last not least approaches that cover only one dimension explicitly are often applicable to other dimensions and to multiple dimensions simultaneously just as well. Yet, this is not always the case and that is a problem: solving the somewhat popular provenance use case doesn't mean that all is fine — e.g. the topic of handling differing viewpoints that recently became of increasing practical significance might require quite different mechanisms.

3.1.1 Annotation

Works concerned with annotations tend to make a sharp difference between knowledge and meta knowledge. They mostly concentrate on pragmatic issues and either deal with administrative tasks like provenance or tackle the ubiquitous aspects of time and space. Less regular, more extrinsic dimensions like secondary detail to primary relations or viewpoints and argumentative stance are mostly ignored and application-centric, one-dimensional perspectives dominate.

3.1.1.1 Space & Time

Some approaches emphasize a distinction between thematic, spatial and temporal relationships between entities and claim that RDF caters predominantly for thematic relationships while neglecting temporal and spatial relationships despite their ubiquity in the real world. Part of the problem is that spatial and temporal relationships are themselves rather complex. The technical and semantic details of locations, regions, time intervals or eras and their corresponding attributes and relationships like `near`, `during`, `open ended` etc. can be intricate and overwhelming when not handled pragmatically. [KK10] argue that for Internet of Things (IoT) applications like the Semantic Sensor Web RDF's thematic metadata facilities need to be augmented by spatial and temporal components. They propose to express spatial positions and geometries with constraints, encoded in a new literal datatype analogously to the datatypes that RDF provides for time. Temporal validity constraints are to be expressed by a forth element in stRDF and need to distinguish user-defined, valid and transaction time types. They also define a corresponding extension to the SPARQL query language, called stSPARQL. [BSK13] elaborate on the temporal dimension of stSPARQL, with special datatypes like periods and intersections of periods and functions like `before`, `during` etc. In stRDF those temporal triples are stored as named graphs where the graph names are IRIs constructed from the valid time annotation. [Hof+13] expect "a new level of usefulness" from consistently integrating "the spatial and the temporal dimension into today's knowledge bases." They represent both time and space as annotations to triples. To make formulating queries syntactically easier SPARQL Basic Graph Patterns can be extended with temporal and spatial constraints on each triple, thus reducing the number of joins that have to be formulated explicitly in the query.

3.1.1.2 Space

[Wan+13] aim at closer integration between semantic and spatial attributes by integrating them in a hybrid index and supporting more generic spatial attributions as well as range queries and joins over spatial data. Spatial data is encoded as longitude/latitude pair in the forth position of a quad statement.

3.1.1.3 Time

[WF06] examine which representation of assertions with valid times best supports reasoning in OWL. Reification and n-ary relations deconstruct the original binary relation and remove it from OWL's reach. Their solution is based on fluents and represents the actors in a relation as time slices. [LT07] model events in time by way of standard RDF n-ary relations. They improve on descriptions of static situations at a certain time by developing reasoning rules to infer assessments over unfolding developments from series of events. [GHV05] model the validity time of a statement as a label on the RDF triple. They also present a rule based deductive system and a query language. [GHV07] present in passing a representation based on RDF standard reification. [TB09], building on the work of [GHV05] and [GHV07], propose a syntax where triples that share the same validity period are grouped in the respective named graph. Time dependent queries can then be formulated in standard SPARQL. [Kri12] extends the ABox triples of an ontology to quintuples, incorporating the start and end time of the period in which a statement is valid. In subsequent works by [Kri14] and [KD15] this approach compares favourably to other representations of time in RDF, especially in terms of query and reasoning performance but also w.r.t. modelling complexity. [Tae+16] investigate dynamic dataspaces that are continuously updated with streams of data. To reduce server load from high-frequency queries for the latest data or for previous versions they annotate either groups of triples in named graphs or, if the datastore supports only triples, through singleton properties and serve them as Triple Pattern Fragments 4.3.3. [Sch+13] undertake the laudable endeavour of actually conducting empirical tests on the usability of popular representation techniques for time annotated data. [Rul+12] investigate to what extent and in which representations temporal data is indeed present on the Semantic Web. [TKK13] point out that the various representational techniques discussed in the context of temporal metadata like temporal validity are inadequate when it comes to representing events in time as primary subjects of discourse. They argue that events, like objects and concepts, need to be conceptualized properly in their own right and provide an example based on a pattern from the DOLCE+DnS Upper Ontology.

3.1.1.4 Provenance

[Din+05] argue that tracking provenance of documents as named graphs is too coarse but fails on triples when they contain blank nodes. As a solution they present an algorithm to decompose a graph into its smallest self-contained subsets, called molecules. [Flo+09] record provenance of aggregated and inferred triples in a forth column, obtaining an RDF quadruple, but technically implemented as named graphs. This allows for realizing update operations with coherence or foundational semantics. Provenance information is organized in a semigroup structure, encoded in graph names. That and an extension to RDFS allows to determine and reason over the provenance of source and inferred triples. [GGV10] introduce the concept of nanopublications as core scientific statements with associated context, stressing the importance in scientific work of backing up claims with proof and provenance in reproducible ways. Nanopublications use named graphs to address single statements just as well as sets of statements. They provide sound reification semantics by taking account of the concrete occurrence of an assertion. Further works extend the concept by disambiguating metadata about the assertion versus the Nanopublication itself [Kuh+16] and add provisions for cryptographically sound identification [KD14] and versioning [Kuh+17]. [Eck13] proposes statement identifiers that encode the whole triple in an IRI and are tied to the named graph or document in which they occur. Those identifiers are self-contained and like with RDF standard reification do not rely on the statement being actually stated in the source graph. However, unlike RDF standard reification the approach makes no provision for multiple occurrences of the same statement. [Fu+15] compare representations according to how they deal with redundancy in metadata. Verbose and redundant metadata can become a performance bottleneck when very fine grained provenance data is required. [Mee+17] argue that capturing provenance has to start when raw data is ingested and processed, explicating what is otherwise only documented in software or notes. To that end they track provenance at the term level to e.g. document parsing of complex value types. [Wyl+17] are concerned with fine-grained multilevel provenance with lineage tracing in query execution. They implement several different storage models and query execution strategies into a high-performance triple store. The representation is based on named graphs that group statements with the same subject, including property values related via blank nodes.

3.1.1.5 Fuzzyness & Uncertainty

[MD06] uses RDF standard reification to attach fuzzy values to statements. [Str09] advances reasoning and querying capabilities but doesn't address representation in RDF. [DTG19] develop an ontology, algorithms and an extension to SPARQL to contextualize and integrate uncertain data on the Semantic Web, using named graphs as representation mechanism but providing only a SPARQL-based operational semantics.

3.1.1.6 Identity

General solutions to the mapping of related concepts from different data sources will be presented below in Section 3.1.2, but even the seemingly easy task of establishing if two entities are the same can be surprisingly hard in practice as discussed in Section 2.1 above. The rdfs:seeAlso property provided by RDF has only very weak and informal semantics. OWL has introduced the infamous owl:sameAs property which, because of its strict semantics, often incurs unwanted and unwarranted entailments. SKOS offers properties like skos:exactMatch, skos:closeMatch and skos:relatedMatch with deliberately restrained formal semantics. [Bat+14] present a concept called 'scientific lens' through which they define contexts in which a link holds and implement it in the biomedical domain, integrating data from different sources. A scientific lens is a set of rules that modifies the links between datasets according to some notion of operational equivalence. Two lenses are introduced, one applying some standard criteria for equivalence and another one where equality is much more liberally defined, dropping most constraints. The authors argue that while technically they could easily define more lenses, users would need training to successfully employ them. [Idr+17] propose contextualized mappings to overcome the very generic nature and often unintended consequences of owl:sameAs. They use a combination of singleton properties and named graphs to express metadata at the triple and at the graph level. [KD14] present a solution to a very technical aspect of the problem space: cryptographically sound identifiers for statements and graphs that even allow self-reference.

3.1.1.7 Multilinguality

[Ehr+14] represent multilingual resources as freshly generated subject identifiers to which properties are attached in usual RDF fashion. As their domain is well structured this works out without problems. Navigating from a BabelNet entry to corresponding entries in other sources like Wikipedia however takes a few hops.

3.1.1.8 Versioning

[Vol+05] use RDF standard reification to address individual triples and `rdf:Bag`s of reification identifiers to represent sets of statements. A full RDF diff contains a set of added and a set of removed statements. [HC14] present a semantic framework that "enables the semantics of SPARQL Update to be used as the basis for a 'cut-and-paste' provenance model", targeting use cases such as recording derivation processes (aka Explainable AI) and in general revision history of the data over time, focusing on dynamic provenance of processes rather than the more static provenance approach of the W3C PROV data model and vocabulary. [Fro+16], [Tae+17] and [Fer+19] all use named graphs to store triples in versions. Again one graph per version holds triples that are newly inserted while a second one records deletes. Metadata about those versions is either stored in further graphs or in the default graph. The focus of these works is mostly on versioning triples rather than whole graphs or even datasets. [And19] presents a quin store that annotates statements in datasets with timestamps, allowing versioning, retaining of previous states and conflict-free replication in a distributed environment.

3.1.1.9 Viewpoints

[Nic07] investigates ways to deal with contradictory assertions and communications and to represent propositional attitudes. The goal is not primarily to map amd lift assertions into an agreed upon realm of 'truth' but to develop a formalism and semantics suited for dealing with heterogeneous knowledge as prevalent on the web. [Bat+14] concentrate on individual entities instead of whole statements and develop criteria and mechanisms dubbed 'scientific lenses' to control equivalence relations. They observe that even "scientific data is messy" and different "datasets frequently capture alternative views of the world at different levels of granularity." On the example of chemical data published in the Linked Data Cloud they show how different sets of data can be automatically linked "according to different equivalence criteria [. . .] To cater for different use cases, the platform allows the application of different lenses which vary the equivalence rules to be applied based on the context and interpretation of the links." [Tch+19] introduce ClaimsKG, a knowledge graph of fact-checked claims, that is constructed from fact-checked data harvested from different fact-checking sites on the web. Integration is based on a specific data model to describe a claim and the fact-checking procedure, entities extracted from the textual representation of the claim, a lifting procedure that normalizes ratings from different fact-checking sources and a simple claims co-reference resolution strategy. The claims themselves are however represented textually only, not as RDF statements. [Suc20] in a vision paper makes the case that much more powerful knowledge representation and reasoning techniques are needed to represent and reason on controversies as covered in ordinary Wikipedia entries, to analyse fake news, detect contradictory claims etc. He observes a dramatic increase in automatically extracted data but with very little semantics and on the other hand a lack of formalisms that could express and reason about more complex statements at scale and in the vast and unruly environment of the web. Instead the paper proposes to aim for a trade-off between expressivity and tractability that is informed by the type of data and the type of uses that can be observed in today's massive, web-scale systems. A concrete vision is drawn of scaled-back description logic expressivity within a context combined with more expressive reasoning about contexts which in the author's opinion might provide a tractable approach better suited to the tasks described at the outset. [EB21] argue that the need for a "polyvocal and contextualised Semantic Web" is e.g. reinforced by current machine learning trends in AI that favour popular or majority vote to the disadvantage of minority perspectives, "resulting in simplified representations of the world in which diverse perspectives are underrepresented." They present a variety of techniques from ontological approaches to data modelling strategies that different projects have employed to express and disambiguate biases, viewpoints and other propositional attitudes.

3.1.1.10 Annotation Frameworks

Obviously all the approaches introduced above cover important annotational dimensions and some of them can be used for different dimensions. In practice however it will often be the case that more than one dimension needs to be recorded per entity. Some approaches explicitly tackle combinations of multiple annotation dimensions.

[URS06] present a framework called aRDF that can handle different annotational dimensions as long as the annotation values are members of a partially ordered set with a bottom element. Consistency for annotated triples can be guaranteed if the partial order also has a top element. [URS10] bring the query apparatus more in line with SPARQL and address performance issues, specifically caching. [Div+09] offer a generic extension of RDF called RDF$^+$ to represent metadata of any kind. They decide against the use of reification because that would "decompose existing triples and fully change the representational model". Instead they use named graphs and internally add a fifth element that reifies a statement in a specific graph. When exporting to standard RDF however statements have to be grouped per graph again and occurrence specificity of metadata is lost. [Zim+12] not only cover annotations of various domains but also their combination into one compound domain. Prerequisite is that the annotation domains follow a certain mathematical structure, an idempotent commutative semi-ring, that guarantees a natural partial order. A proposed extension to SPARQL enables conjunctive queries over annotated statements selected by annotation values, even by combinations of values from different annotation domains. [Hog14] focuses on performance issues when reasoning over combinations of annotational dimensions and offers algorithms optimized for scalability.

3.1.2 Contextualization

A context mechanism for the Web would be a major step forward for a number of information-related uses. Current datamining and knowledge discovery techniques often require domain specific knowledge to function properly, but this knowledge can bias the results. In many systems, however, this knowledge is embedded in procedural code or is implicit in the problem encodings or fitness functions used. Making the different ontological commitments of competing interpretations explicit, and linked together, can permit different views of data to be simultaneously developed and explored. [...] contexts will also be critical in providing social machines with the ability to manage simultaneous, but conflicting, views of data. It is the capability for different scientists to interpret the same data in different ways that provides for the argumentation and testing so crucial to scientific discourse.

Jim Hendler and Tim Berners-Lee[5]

Nobody ever thought that collecting all kinds of information in a decentralized system on a world wide scale would lead to one huge homogeneous and contradiction-free graph. Mechanisms to contextualize and, if needs be, also isolate data from each other were envisioned early on in the development of the Semantic Web. The main problem was and is that such mechanisms are not only computationally expensive but also not straightforward to design. The notion of context can take many different forms and be interpreted quite differently depending on the envisioned application. Consequently proposals vary widely in expressivity and premises.

[GMF04] are concerned with problems that arise when aggregating data from different sources. Slight or hidden variations in the underlying semantics like technical details, units used, differing viewpoints, implicit assumptions etc. can produce unintended and damaging results when merging data. Therefore they propose to extend the RDF(S) model theory with a notion of context and define operations like so-called *lifting rules* to manage and control such differences during data integration. While their work is rooted in classic AI work on context pioneered by McCarthy, they omit advanced features like context nesting and transcending in favour of scalability and ease of use. [Bou+04] base their work on a more general interpretation of context by differentiating between globally shared and locally closed ontologies. In their intuition local ontologies have some practical advantages like requiring less coordination upfront and enabling the encapsulation of incompatible views. To communicate across contexts local ontologies need to be 'contextualized' via explicit mapping rules. They propose a formalism called C-OWL by modifying the OWL semantics and adding some syntactic extensions to define bridge rules. A context itself however and its eventual properties is not an object in C-OWL. [Bao+10a] refine the McCarthy

[5][HB10]

style context mechanism by differentiating between a statement belonging to a context and being true in a context. A fictional character from a cartoon for example might have no truth value whatsoever in a social security context. They do not provide, however, a concrete axiomatisation. [Bao+10b] add to that a proposal for a semantics and syntax based on named graphs. Similar to C-OWL, interpretations are local to a context and mappings between contexts have to be specified explicitly[6]. [HTS10] start from the intuition that "most of the knowledge available in the Semantic Web is context dependent." Consequently contextualization cannot be limited to a mapping phase. To that end they treat contexts as first class objects that can have properties and be reasoned upon. Dimensions of context include time, space, topic etc. Relations between contexts do not have to be captured explicitly by lifting axioms, instead knowledge 'propagates' in declarative fashion based on specific properties. Their approach doesn't require any semantic extension and is implemented on top of an off-the-shelf RDF triple store. The architecture is called Contextualized Knowledge Repository (CKR) and has since been repeatedly refined and extended. [BHS12] take a step back and present for discussion a set of criteria they see as essential for any contextualization, without going into technical detail. They stress the importance of flexible contexts that can be related to, lifted to and mapped, overlapping and locally closed, explicitly described and reasoned upon. These criteria resemble the sort of powerful AI system that [GMF04] rules out as too involved for the Semantic Web w.r.t. to usability as well as tractability. [Kla13] presents a theoretical framework to compare the computational properties of approaches to contextual reasoning, based on McCarthy's concept of contexts as formal objects that have properties, can be reasoned upon and can be organized in hierarchical structures. This framework allows for studying the interactions of different content and context logics as a system of two-dimensional, two-sorted combinations of DL-based representations. He applies this framework to various formalisms and application scenarios with varying complexities.

3.2 Applications

Applications of Semantic Web technologies in Linked Data and Knowledge Graphs in a way mirror the dichotomy between annotation and contextualization.

3.2.1 Linked Data

Linked Data (LD) or, with an emphasis on reuse-friendly licenses, Linked Open Data (LOD) was the banner under which the first wave of large scale applications of Semantic Web technology on 'real' data was developed.[7] It aimed and to a good part succeeded at solving the chicken-and-egg problem from which the

[6]Pat Hayes sketched a similar context mechanism, complete with ideas on model theory and syntax, while participating in the RDF 1.1 Working Group, at `https://www.slideshare.net/PatHayes/rdf-with-contexts`

[7]Notwithstanding earlier deployments of the FOAF vocabulary with social networks.

Semantic Web suffered in the beginning, making it look like a solution in search of a problem: attractive Semantic Web applications depend on the availability of data, but without attractive applications there is little incentive to publish data in the first place. A blog post[8] by Tim Berners-Lee gives a short introduction into the concepts behind Linked Data, while [HB11] provide a more detailed guide to developing Linked Data applications. A visualization of the so-called *Linked Open Data cloud*[9] couldn't be missing from any Semantic Web presentation for many years, just like the layer cake[10] in the preceding decade.

Not much research has however been conducted on modelling styles applied in Linked Data apart from investigations into problems related to entity mappings via `owl:sameAs`[11] and a treatment of blank node usage in Linked Data by [Mal+11] (ignoring the advice given by [HB11] to avoid them). [Asp+19] is a notable and rather recent exception, studying link structures and interlinkage in the LOD cloud.

3.2.2 Knowledge Graphs

The term Knowledge Graph became a buzzword since Google claimed in 2012 that such a thing drives its search service to new levels of accuracy and usefulness.[12] Since then a lot of knowledge bases have been re-labelled as Knowledge Graphs although a concise and commonly agreed upon definition still has to emerge. The following definition, quoted from [Krö17], captures the aspects essential to this work:

1. Normalisation: Information is decomposed into small units of information, interpreted as edges of some form of graph.
2. Connectivity: Knowledge is represented by the relationships between these units.
3. Context: Data is enriched with contextual information to record aspects such as temporal validity, provenance, trustworthiness, or other side conditions and details.

This area of work is probably the closest to the at times quite theoretical and far fetching research on context. It has developed some interesting approaches and implementations for integrating rather diverse knowledge structures and sources.

3.2.2.1 YAGO

YAGO is one of the earliest research prototypes of a Knowledge Graph on the Semantic Web, predating even the term itself by a few years. It has been redesigned and reimplemented several times, to put different intuitions and

[8]`http://www.w3.org/DesignIssues/LinkedData.html`
[9]`http://richard.cyganiak.de/2007/10/lod/`
[10]`http://www.w3.org/2001/09/06-ecdl/slide17-0.html`
[11]See Section 5 below
[12]`https://googleblog.blogspot.com/2012/05/introducing-knowledge-graph-things-not.html`

designs to the test [Reb+16]. The basic approach is to extract facts from the Wikipedia encyclopedia and create a YAGO ontology by structuring those facts according to a combination of the WordNet taxonomy and the Wikipedia thematic classification.

The YAGO model originally introduced by [SKW07] is a slight extension of RDFS. It relies heavily on reification and follows a concept of primary relations with secondary attributes. The storage schema in a relational database consists of the standard RDF components subject, predicate and object plus a statement identifier and a confidence value. The statement identifier serves as subject or object in further statements that add attributes to a primary fact. The confidence value is rather application specific, motivated by YAGO's use of automated extraction and classification techniques. [Hof+13] in the YAGO 2 project extend the RDF triple in a different way: with context, time, space and a statement identifier. The context component has no specific semantics, akin to a rather associative 'catch all' property. Time and space are considered as ubiquitous parameters that are always useful to have at hand. The statement identifier is as before used in further triples to add information that's not covered by the other dimensions, but also to encode those in a more RDF compliant way. [TWS20] with YAGO 4 again pursue a complete rewrite of YAGO, aiming at a more solid reasoning environment. It switches data source and taxonomy — from Wikipedia infoboxes to Wikidata and from WordNet to schema.org — and adds constraints formulated in SHACL to filter out inconsistencies in the source data. Statement annotations are now encoded in RDF-star in 'separate assertions mode' which allows for annotating a fact without actually asserting it (e.g. to annotate facts from the past that are not valid any more).

3.2.2.2 DBpedia

[Leh+15] present DBpedia, a project similar to YAGO, that uses the standard RDF reification mechanism to capture provenance and other metadata. It implements some neat tricks like encoding the subject, predicate and object of a triple in the statement identifier and organizing statements, reification statements and statements on reified statements in different graphs, as described in detail by [Ism+18].

3.2.2.3 Wikidata

Whereas YAGO and DBpedia automatically extract facts from Wikipedia and other sources and build their own knowledge graph from them, Wikidata, the central Wikipedia data management platform, takes a different route. Wikidata is a system edited by humans and its contents can be exported to RDF, among other formats. It is however based on statements with annotations and references, a structure that doesn't translate easily into RDF. [Erx+14] design an export that converts the basic relations of Wikidata to n-ary relations in RDF, with a newly generated statement identifier as the main anchor point and relations typed according to node types. Annotations on the basic relation are linked to the statement identifier. This way to model relations is rather unnatural to RDF but provides sound semantics to support reasoning with OWL DL. The authors therefore also offer simpler RDF exports that represent statements by single RDF triples but omit qualifiers. Simple and involved exports can be combined and used together. [HHK15] compare alternative modelling approaches for Wikidata, in particular RDF standard reification, n-ary relations, singleton properties and named graphs. They export Wikidata to each format and examine triple count, query patterns and database performance on five popular triple stores. In a follow up [Her+16] focus on comparing different database system paradigms: a graph database, a relational database and two triple stores whereas [Fre+19b] uses the Wikidata test data from that work to compare again more ways to model reification in RDF. [Krö17] analyse with the example of Wikidata how description logics and rule languages struggle with attributed statements, stressing how unfortunate that is and pointing to ongoing research and future directions to address this problem. [Car+19] report that on request of their users they model the "same concept both with n-ary relation patterns that enable high-level modelling needs such as temporal indexing, state changes, model evolution, meta-classes etc., and shortcut binary relations, which support lightweight modelling and intuitive navigation." The shortcut relations seem not to be explicitly linked back to the n-ary relations that they are derived from.

3.2.2.4 CommonSense Knowledge Graph (CSKG)

[ISZ21] integrate several key sources like ConceptNet, FrameNet, WikiData and WordNet into a "CommonSense Knowledge Graph." Problems they have to overcome include different foci, diverging modelling approaches, sparse overlap and sparse interlinking. To that end they develop a set of core design principles such as embracing the heterogeneity of basic node types, minimizing the set of edge types, leveraging external links to establish node equivalence, improving interlinkage through heuristically generated embeddings and manual mappings and providing human-readable labels to facilitate retrieval and usage. Their implementation is based on the Knowledge Graph Toolkit (KGTK)[13] and uses 10 columns to store the relation and its most frequently used properties. An `id` attribute allows to add more properties as desired.

[Noy+19] in a follow-up article to a panel discussion at ISWC 2018[14] provide a rare view into the workings of and challenges faced by huge corporate Knowledge Graphs like those developed at Google, Microsoft, Facebook, IBM and eBay. A recurring design strategy those KGs employ is to ingest and store information as detailed as possible and deal with the ensuing complexity later on e.g. through case-based reduction, "scalable levels of abstraction", "push[ing] entity resolution to runtime through context" etc. The generation of views tailored to specific

[13] [Ili+20], see Section 10

[14] International Semantic Web Conference, panel introduction and slides at `http://iswc2018.semanticweb.org/panel-enterprise-scale-knowledge-graphs/`

applications from very involved knowledge bases in the background is described as a standard technique to deal with complex knowledge structures in leading industry-scale Knowledge Graphs. Fine-grained , even statement level provenance plays an important role in conflict resolution and confidence evaluation.

3.2.2.5 Open Knowledge Network (OKN)

Whereas the aforementioned projects are all implemented in isolation — building on and integrating data from but not interacting with each other — the Open Knowledge Network (OKN) introduced by [GM16] aims at bootstrapping a decentralized Knowledge Graph, thus more in line with the original vision of the Semantic Web. The project seems to struggle to gain momentum but has recently seen renewed activity.[15]

3.2.2.6 Mainstreaming

The question of if and when all the data hidden and trapped in closed Knowledge Graph silos will be “freed” and integrated — just like a large part of code bases became Open Source in the last two decades — waits for a satisfying answer. In the meantime Knowledge Graphs of all kinds and purpose flourish: from personal to corporate, from topical to general. In a quickly growing corpus of literature on what Knowledge Graphs are and how to build them [Hog+21] and [SL21] provide useful starting points for further reading.

[15]See `https://beta.nsf.gov/tip/updates/nsf-releases-open-knowledge-network-roadmap-report?s=09` and there especially `https://nsf-gov-resources.nsf.gov/2022-09/OKN%20Roadmap%20-%20Report_v03.pdf`

Part II

APPROACHES

This part aims to present all the little ideas, grand designs, ingenious hacks and daring advances that have been proposed in pursuit of a more expressive, more integrative, more complexity-savvy Semantic Web. The main focus is on representational approaches, Chapter 4, a vast and varied field on its own.

But many parts have to mesh, many components may be tuned, many seemingly small details can make an important difference when trying to make a project as complex as the Semantic Web work as desired as a whole. Therefore other topics and aspects besides the issue of representation are investigated.

The problem of identification, Chapter 5, is at the very heart of the Semantic Web, although sometimes hiding in plain sight as it might seem.

Serialisations, Chapter 6, play an important role in usability, at least as long as no other, more abstract advance at a surface layer for RDF has materialised — those are called for, and attempts have been made e.g. in visualizations and automatic form generation, but that field is regrettably outside the scope of this work.

Some topics however — especially ontologies, Chapter 7, querying, Chapter 8, reasoning, Chapter 9, implementation, Chapter 10, alternative graph models, Chapter 11 and example use cases, Chapter 12 — are too complex and substantial to cover them in the same detail as representational aspects. Still, an attempt is made to discuss at least the most important aspects, present influential works and provide some useful pointers for further exploration.

Chapter 4

Representation

The most obvious place to start when looking for ways to extend the expressivity of RDF are of course its syntactic facilities, the representation of facts as triples, composed of IRIs, blank nodes and literals. Many works focus on representational approaches: ways to compose triples into more complex constructs, ways to extend them to quads, quins and beyond, ways to increase the expressiveness of the basic building blocks — IRIs, literals, bnodes, or certain positions in the triple like the predicates — themselves.

This is a tricky terrain for several reasons.

As was the topic of Section 3, many different applications demand more expressive constructs in RDF. They do however have to compete for a limited supply of plausible ways the triple syntax can be extended. This can easily lead to proposals that rather convincingly tackle a common use case, but lose their elegance when applied to the whole spectrum of problems and demands. Designing an extension mechanism that is simple *and* flexible *and* extensible *and* unambiguous *and* expressive *and* follows the semantics as specified *and* seamlessly fits into the installed base *and* . . . is, obviously, a really hard task.

Syntactic primitives tend to carry with them semantic intuitions that do not always correspond to what their definition says, if a precisely specified semantics is even provided to begin with. Sometimes, to make matters worse, they carry multiple plausible — but mutually exclusive — semantic intuitions.

Another problem is that new syntactic features need to be supported by what in the last twenty-five years has become a huge stack of standards and systems. They have to play nice and seamlessly integrate with the installed base and the data that is already out there. Both installed machinery and the data provided represent huge investments in effort, time and and money. The Semantic Web community is very conservative in that respect: nothing is allowed to break what already exists. That makes the design space even more tricky than it already is.

There are other problems like the difficulty to understand and always be aware of the many degrees of semantic freedom that natural language provides. The expressive richness of natural language guides our intuitions and from there finds its way into use cases and syntaxes, but time and again the resulting proposals only manage to grasp a part of reality and sometimes break spectacularly when confronted with the many different aspects and desires present in the Semantic Web as a whole. For now that should just serve as a warning and reminder, but it will be the topic of some discussions below.

This section focuses on the representational aspects, syntactic or otherwise, and aims to collect everything that has been proposed in the last two decades in one place. Maybe there is a hidden gem that was overlooked at the time and could provide the answer to all problems (w.r.t. RDF's need for more expressive means that is), maybe some new combination of what has been tried and proposed will spring to mind, maybe at least some repetition of past mistakes can be prevented. In any case, the field is so vast that it is hard to have an overview of. Establishing a good overall picture is a good enough goal for this section.

4.1 Composition

Composing knowledge from sets of simple facts is the way RDF is designed to work. The least intrusive approach to extend its expressivity is therefore, quite naturally, to search for ways to make those compositions themselves more expressive, more intricate than simply piling up more and more triples. On the syntactic level this often involves blank nodes as the main structural tool RDF provides. Out-of-band means external to the RDF formalism like modelling conventions encoded in so-called *design patterns* can play a role. Shapes have been introduced to enforce constraints on data structures but can also be employed to integrate design patterns more closely into the RDF machinery.

4.1.1 N-ary Relations

> *There is a long history of solutions to deal with relations that range over more than two arguments, or have polymorphic arities. (Generic) use cases and solution proposals come from AI (e.g. situation calculus, relation reification, Minsky's frames, description logics, etc.), from philosophy (e.g. Gestalt theory, Davidson's reified events, and the debate on the nature of events vs. propositions), from linguistics (e.g. Bach's eventualities, Discourse Representation Theory, FrameNet frames, cognitive linguistics schemata), from conceptual modeling, from the Semantic Web (e.g. n-ary relation reification, fluents, named graphs, linked data integration, RDF data structure extensions), etc.*
>
> Aldo Gangemi and Valentina Presutti[1]

Works discussing n-ary relations in general and comparing different approaches have been introduced in Section 2.7.4. This section presents specific approaches

[1][GP13], references omitted

to employ them in place of more expressive formalisms or to take off the edge of their rank growth. It also reminds of a once close cousin that had proposed a much more expressive, but also less flexible alternative to RDF.

4.1.1.1 N-ary Relations as Idiomatic Constructs

[Erx+14] describe the complex data model of Wikidata where statements are often enriched with qualifiers that provide additional context like technical detail, validity time etc, but also more arguments to relations that are essentially n-ary. A second form of enrichment is the addition of references that support a claim. This data model cannot easily be reproduced in RDF. Instead of following the same modelling style as Wikidata — main relations, refined with properties — [Erx+14] choose a suitably involved neo-Davidsonian n-ary pattern to faithfully represent both the main facts and their qualifications and references in RDF. A new individual is introduced to represent the statement, and linked to subject, object and qualifiers via relation-specific properties. Those properties are derived from the original property, but specialized w.r.t. their target, thus omitting the need to explicitly reify the property itself but adding another level of complexity to the model. Qualifiers and references are modelled with similarly relabelled properties, but without a reifying identifier as they themselves can't be annotated in Wikidata. All major structures of the Wikidata data model are reproduced as resources and no objects of the data model are encoded as blank nodes. The resulting RDF data is OWL compatible and properties are typed with domain and range. The authors argue that although their approach doesn't create idiomatic RDF it is a faithful reproduction of the relations in Wikidata because it doesn't lose any detail, as "Wikidata statements can even be argued to be the primary subjects that Wikidata collects information about. In this sense, representing Wikidata objects by RDF individuals is not a technical workaround but a modelling goal." This claim seems far fetched given that what a Wikidata user *sees* is a relation with attributes, not some reproduction thereof in a quite orthogonal model. Neither a user accustomed to Wikidata nor one familiar with idiomatic RDF will find the export particularly intuitive or easy to use.

[FDG18] prefer n-ary relations to reification or named graphs for "applying the Frame Semantics linguistic theory, as opposed to binary techniques." N-ary relations are a popular base construct in many KR formalisms and matching them to n-ary relations in RDF is much more straightforward than using the immature techniques that RDF provides to extend binary relations — even if it leads to non-idiomatic RDF constructs.

4.1.1.2 N-ary Relations with View Shortcuts

Some works tackle the usability problems of n-ary relations by accompanying them with 'standard triple' binary relations that repeat the main information of the n-ary relation in a more accessible manner.[2]

[2]See Section 7.4 for practical approaches to create such shortcuts

[SKW07] in the Yago project[3] argues that in each n-ary relation a primary binary relation can be identified to which the other relations add secondary attributes. This interpretation is implemented via RDF standard reification.[4] As YAGO uses the RDF/XML serialisation, the verbose reification quad disappears behind a concise statement identifier.[5]

[Erx+14] model Wikidata statements in RDF as n-ary relations. Although the details are different, the approach is similar to the way RDF standard reification models statements: a node represents the statement and specialized properties connect it to subject, object and further qualifying values (as Wikidata implements an attributed graph model). While this faithfully reproduces the information in Wikidata as RDF data it is not idiomatic RDF anymore and neither easy to query nor navigate. To mitigate this problem they also provide alternative conversions to RDF which convert only unqualified statements to idiomatic RDF (arguing that just omitting qualifying information could lead to statements of questionable accuracy, e.g. if temporal limitations on the validity of a fact are missing from the converted statement). Faithful and simple exports can be combined and used side by side.

[Car+19] report on the introduction of shortcut relations in addition to n-ary relations. Their base modelling constructs are expansive n-ary patterns that enable "high-level modeling needs such as temporal indexing, state changes, model evolution, meta-classes, etc." Non-expert users however were frustrated by the complexity of the resulting constructs. To enhance usability the authors added "shortcut binary relations, which support lightweight modeling and intuitive navigation" to express the most essential facts and concepts in a more accessible way. Those shortcut relations however are neither systematically generated nor connected via some sort of backlink to the n-ary structure they are derived from.

[RMH17] aim at integrating knowledge from different sources. To ensure the necessary expressivity they base their data model on a flexible and expressive but not very intuitive n-ary relation scheme. This primary model is complemented by an easier to use binary representation, which captures the basic facts but not necessarily every detail. The conversion between n-ary and binary representation can be run back and forth and may be implemented e.g. with SPARQL `CONSTRUCT` queries.[6]

[Gan+16] develop Framester, "a frame-based ontological resource acting as a hub between linguistic resources such as FrameNet, WordNet, VerbNet, BabelNet, DBpedia, Yago, DOLCE-Zero, and leveraging this wealth of links to create an interoperable predicate space formalized according to frame semantics." From n-ary relations following the frame paradigm they derive 'frame projections' as typically unary or binary aspects, not un-similar to the shortcut relations

[3] See Section 3.2.2.1

[4] [RMH17] reports that when no obvious primary relation exists, multiple instances of statement and annotated reification are generated

[5] See Section 6.1

[6] It is however not entirely clear to this author how the conversion from a binary relation back to the originating n-ary relation would work.

discussed above. “Due to the expressivity limitations of OWL, some refactoring is needed to represent frame semantics: frames are represented as both classes and individuals, semantic roles and co-participation relations as both (object or datatype) properties and individuals, selectional restrictions and semantic types as both classes and individuals, situations and their entities as individuals. Frames and other predicates are represented as individuals when a schema-level relation is needed (e.g. between a frame and its roles, or between two frames), which cannot be represented by means of an OWL schema axiom (e.g. subclass, subproperty, domain, range, etc.).”

4.1.1.3 Topic Maps

Topic Maps[7], very readably presented by [Pep00], are a formalism related to RDF but based on quite specific n-ary relations. Topic Maps were conceived around the same time as the Semantic Web but not with exactly the same goal. Their prototypical use case of integrating documentations from a merger of two companies is much more specific than the more general vision driving the Semantic Web.

They addressed some issues from the start that RDF still struggles with and that this survey is all about: they incorporate a notion of scope in their basic modelling primitive, they explicitly tackle the problem of identity by distinguishing a resource itself from what the resource is about and they quite naturally model n-ary relations. Reification is deeply embedded in the formalism without sacrificing parsimony, as [New02] nicely illustrates and contrasts to RDF’s approach. On the other hand this expressiveness comes at a cost and the basic Topic Maps primitives are nowhere as simple and flexible as the RDF triple.

Some attempts were made to align the two standards and improve interoperability, e.g. [Gar03], the RDF/Topic Maps Interoperability Task Force[8] and “Guidelines for RDF/Topic Maps Interoperability”.[9] Also a W3C Working Group Note titled “A Survey of RDF/Topic Maps Interoperability Proposals”[10] was published on the subject. [Gar02] specifically investigates the use of quads and quintuples to efficiently translate from Topic Maps to RDF and vice versa.

The Topic Maps technology eventually went stale although some RDF practitioners today still have fond memories of it. Broader interest shifted to a new, less strictly defined formalism: Labelled Property Graphs (LPG). Like Topic Maps LPGs can be understood as annotated and objectified relations and thus as specializations of general n-ary relations. They will be discussed in more detail below in Section 4.2.4.6.

[7]`https://www.isotopicmaps.org/`

[8]`https://www.w3.org/2001/sw/BestPractices/RDFTM/`

[9]`http://www.w3.org/2001/sw/BestPractices/RDFTM/guidelines-20060630.html`

[10]`https://www.w3.org/TR/rdftm-survey/`

4.1.2 Design Patterns

Design patterns, also often referred to as Ontology Design Patterns (ODP), are one possible approach to bring some structure and predictability to data modelling in RDF. Design patterns have come to some fame through the work of Christopher Alexander in architecture. The idea has then been applied with considerable success in computer science to programming, helping developers to refine their architectural abilities, develop styles and reuse code patterns.

The approach is essentially the same in any domain: search for recurring themes and practices in an area of activity, abstract them away from concrete applications, name and group them into recognisable units that help both the author as well as the reader to develop a better understanding and familiarity with the essence of what's going on. [Gan05] and [Hoe09] provide introductions into theory and practice of ontology design patterns for Semantic Web content, including example patterns and backgrounds in established Knowledge Representation formalisms like Frames and Semantic Networks.

N-ary Design Pattern [Hoe09] presents among others a pattern to model n-ary relations[11], and discusses in detail ontological commitments, practical implications and consequences for OWL reasoning.

[GP13][12] explore seven different ways to model n-ary relations in RDF and compare them along several axes like usability, suitability to reasoning in OWL, triple count and expressivity. The patterns they present are sometimes a bit surprising but the work is remarkable not only for the breadth of aspects it addresses, but especially for tackling the intricate questions of usability and expressivity in a principled and rigorous manner.

[SPH19] discuss a set of recurring representation problems involving different kinds of context-sensitive knowledge and introduce n-ary patterns to represent them in an OWL-compatible way. Their approach deploys the OTTR framework (discussed below) and builds on earlier work on aspect-oriented ontology development in [SP14] and [SP18].

4.1.2.1 Usability

To counter the well-known difficulties of non-expert users with involved n-ary patterns, [Pre+16] presents a technique to represent ontology design patterns as a network of UML components — a formalism popular among software developers — claiming that the "notation hides the complexity of OWL representations while still conveying the main semantics to domain experts."

[Hit+17] propose to use OWL annotation properties to indicate patterns, components of patterns (and e.g. their relevance for a given pattern) and relationships between different patterns and between patterns and ontology modules. A representational ontology called OPLa[13] is presented to apply the

[11] `http://ontologydesignpatterns.org/wiki/Submissions:N-Ary_Relation_Pattern_(OWL_2)`

[12] [GP16] is a slightly updated version

[13] `http://ontologydesignpatterns.org/opla/`

proposal in practice.

[KPS19] investigate reuse in ontology design patterns. As the research topic is rather new they focus in a first step on repetitive structures in content-oriented and logic-oriented patterns.

4.1.2.2 Repositories

Central hubs promoting design patterns are e.g. the Manchester Ontology Design Pattern Public Catalogue[14], the community driven Ontology Design Pattern wiki[15], the Linked Data Patterns website[16] and the Modular Ontology Design Library (MODL) catalogue[17]. The Ontology Design Pattern wiki provides a platform for proposing new patterns and open discussion and exchange. Linked Data Patterns, also available as a book [DD11], provide a "pattern catalogue for modelling, publishing, and consuming Linked Data" while the other repositories have a stronger focus on formally sound and OWL-compatible patterns. [SHH19] present a more recent addition to the space, focusing on findability, accessibility and modularity of resources and suggesting a "curated collection of well-documented ontology design patterns." All these projects offer a range of patterns for tasks relevant to this work, e.g. patterns to model n-ary relations, lists, reification, annotation, graph naming etc.

4.1.2.3 Frameworks

[ESA08] present the Ontology Pre-Processor Language (OPPL) to automate the manipulation of OWL ontologies. It is intended for use cases where ontologies contain a lot of entities, have repetitive or complex axiomatic structures or have repetitive annotation values. The popular Protégé ontology editor is supported by an OPPL plugin extension[18].

[Skj+18] also aim to support users at developing and using design patterns. Reasonable Ontology Templates (OTTR)[19] provides tools to implement patterns rather than the patterns themselves, acting like a macro language to RDF. Modelling patterns are represented by templates that can be build from simpler or combined to more complex templates, establishing a higher level of abstraction then RDF and OWL.

4.1.3 Shape Languages

Whereas vocabularies provide a list of possible attributes, shapes mandate a specific structure for data, combining attributes from vocabularies in a certain way.

[14] `https://odps.sourceforge.net/odp/html/index.html`
[15] `http://ontologydesignpatterns.org/wiki/`
[16] `https://patterns.dataincubator.org/`
[17] `https://github.com/Data-Semantics-Laboratory/modular-ontology-design-library`, `https://dase.cs.wright.edu/content/modl-modular-ontology-design-library`
[18] `http://oppl2.sourceforge.net/index.html`
[19] `https://ottr.xyz/`

Ruben Verborgh [20]

The whole initial set of Semantic Web languages, RDF at the core and the ontology languages RDFS and OWL stacked directly on top, is optimized to find hidden connections, to unearth information implicit in the data, to extend the known world, to spread out. Practitioners and application developers however often work from very different premises: they need and employ tools to guarantee operational safety, to constrain data to the forms and formats their applications can handle and to guard their code from unruly data sources. RDFS with its `domain` and `range` properties, unfortunately even called a 'schema' language and thereby invoking all the wrong expectations, and OWL with its equally evocative vocabulary are often misunderstood as providing the tools for such data management demands, quite contrary to their actual semantics. This mismatch of intuitions caused a lot of confusion and frustration among newcomers to the Semantic Web.

OWL can be used to define constraints but not in the way that users expect, not as easy as hoped for and not for data that doesn't adhere to OWL's requirements. The problem is grounded in OWL's adherence to the Open World Assumption (OWA) and the No Unique Name Assumption (NUNA). These assumptions make it at least very cumbersome to define constraints for data validation in applications where complete knowledge can be presupposed. [Tao+10] and [Pat15] discuss ways to *close the world* for the purpose of constraint checking and propose formalizations that can be implemented in SPARQL.

Other approaches, entirely sidestepping OWL in favour of SPARQL, led to the quite successful development of so-called *shape* languages. Community processes w.r.t. shape languages have been a bit bumpy and two competing standards were established:[21] the Shapes Constraint Language (SHACL)[22], a W3C Recommendation, and Shape Expression (ShEx)[23], specified by a W3C Community Group.[24]. SHACL is a "SPARQL-based rule and constraint language [...] Similar to SHACL, ShEx is a constraint language built on regular bag expressions inspired by schema languages for XML."[25] A comprehensive introduction into both languages is given by [Gay+17].[26]

4.1.3.1 Usage

The differences between SHACL and ShEx are most visible at the syntactic level, but they both allow "to specify a set of characterizing properties and attributes for classes of entities" and can be used to check for type compliance

[20] [Ver19]

[21] They are not as different as so they couldn't be unified into one, but that would of course require the willingness to undertake the effort.

[22] `https://www.w3.org/TR/shacl/`

[23] `http://shex.io/`

[24] `https://www.w3.org/community/shex/`

[25] [RLH22]

[26] See also the recording of a tutorial at ISWC 2020, `http://www.validatingrdf.com/tutorial/iswc2020/`

to optimize query processing, for automatic generation of form controls, for automatic detection of metadata errors etc.[27] [RLH22] in a community survey found that usage in practice is quite heterogeneous, using a wide range of tools and often involving either manual generation of shapes or time-consuming quality control of automatically generated shapes. [LDV20] present some preliminary statistics on the use of SHACL constraints in data shapes found on GitHub, suggesting a focus on datatype and cardinality constraints. [Gay22] propose WShEx, "a language inspired by ShEx that directly supports the Wikibase data model and can be used to describe and validate Wikibase entities." Most notably it extends ShEx with the ability to validate Wikidata's qualified relations.

4.1.3.2 Shapes for Navigation and Exploration

In the context of this work an interesting feature of Shapes is their potential to structure and group statements, to describe n-ary relations algorithmically — not unlike the approaches discussed in Section 4.3 but in more sophisticated ways. The opposite direction is the focus of a suite of tools that aim at generating shapes from data sources algorithmically, based on techniques employing data profiling, predictive modelling, Inverse Open Path rules, ontology-based approaches, RML mappings, etc.[28] These approaches differ most in whether or not they work mainly on ontologies (TBox) or instance data (ABox). Using shapes as a means of navigation however seems less common. In a blog post [Ver19] discusses the usage of shapes as definitions of the "fields and structure that client and apps can expect to find in a view over a piece of data", together with definitions of interface elements and storage guidelines as the three elements that are required to enable true interoperability of decentralized applications in the Solid ecosystem.[29] The author argues: "Personally, I consider validation to be one of the least interesting uses for shapes; I see so much more potential in shapes as points of convergence between apps and decentralized data sources. I envision apps stating what shape they expect, and pods declaring what shapes they have or can make available." That is also the perspective of this work on shapes.

4.2 Attribution

Composing involved data structures from simple triples is the way RDF is designed to work, but as just discussed it has its limits: the resulting structures quickly become unwieldy, detail becomes overwhelming and the main message is in risk of getting lost in the noise. An alternative approach is to try to either extend the few primitives that RDF provides and attribute them with further detail, secondary administrative metadata or annotations to specific actors, or add new primitives that group sets of triples in annotated graphs.

[27] [RLH22]
[28] [RLH22]
[29] `https://solidproject.org/`

This section advances from the smallest parts to the largest: it starts with datatypes and continues to subject and object terms, the properties that relate them, whole statements, groups of statements, statements with predefined additional attributes and finally n-tuples as prevalent in relational databases. Each approach has its pros and cons, weighing expressiveness versus conciseness in different ways. All these approaches have their merits and while some may well be combined, others are perceived as mutually exclusive.

To be sure: this area is a minefield. Some approaches have spurred long and heated debates before everybody gave up on them. Some have been hyped as finally *the* solution but then fell from grace for no obvious reason. It seems that there's always a new kid on the block — reification, statement identifiers, named graphs, singleton properties, RDF* — just to be out-shined by the next contender who promises to deliver on every meta-modelling need ever felt, and practically for free. The underlying semantic questions are often neither well understood nor thoroughly discussed. An approach may shine for perceived simplicity because it handles a common use case with ease — but it may well fail on many others. This may help explain the ups and downs in public opinion and should be kept in mind when weighing elegance versus utility.

Notably, few of the works presented in the following have tried to understand the problem space first and then investigate if a combination of some approaches might lead to a reasonably well-balanced design. Most works concentrate on either one use case or one possible technique. The hope to find some magic key to all problems of expressivity is still strong in the Semantic Web community. On the other hand hope that anything more involved than a single straightforward primitive will be accepted is equivalently limited. It often seems that such sentiments determine the design space more than the actual complexity of the problem.

Trickle Down Semantics Before presenting these proposals one by one it is important to understand that from a logical standpoint it doesn't make such a big difference if an approach is put on individual terms or their relation or the whole statement or even groups of statements. [GP13] pass on a remark by Pat Hayes from a mailinglist discussion about temporal annotations that restrict the validity of a statement to a specified time.[30] Hayes explains how such an annotation can be equally well attached to the whole statement *or* the property of the statement *or* the individual terms that the property relates, all without changing the meaning of the annotation. Hayes calls this 3D for annotations on the whole statement, 3D+1 for annotations on the property and 4D for annotations on the terms. The temporal restriction 'trickles down' from the whole, through the relation into the parts. The transformation between these different ways to represent the same meaning is purely syntactical.

The 3D approach is often associated with contextualization and possible

[30]http://ontolog.cim3.net/forum/ontolog-forum/2011-02/msg00009.html

world semantics, e.g.:[31]

```
(NodeA relatedTo NodeB)@timeX .
```

The statement itself doesn't make any reference to the annotation, which is externally attached to it.

The 3D+1 approach resembles the classical AI/KR approach, used to be called *situation calculus*, e.g.:

```
NodeA relatedTo@timeX NodeB .
```

In Hayes' experience it is still mostly perceived as applying to the whole statement, like the 3D approach, although in this case it is attached to an element of the statement itself, therefore rather internal to the statement.

The 4D approach is based on a different intuition. Here the terms represent things in time, e.g.:

```
NodeA@timeX relatedTo NodeB@timeX .
```

establishing rather 'worm-like' entities of which the temporally annotated parts represent slices. Those temporal parts are often called fluents.[32] Hayes notes that many people find this approach unintuitive, and works by [WF06] and [Sch+13] confirm that concern w.r.t. to newcomers, but find that it tends to grow on users and more experienced users find it rather expedient.

Hayes' remark however doesn't get into more fine-grained annotations that don't target the statement as a whole but only specific terms in it. In that case some subtle distinctions can be observed, e.g. the annotation `@1920s` attached to Berlin in the following example:

```
Alice likes Berlin@1920s .
```

shouldn't propagate up to the whole statement, and neither should `@18` as attached to Alice:

```
Alice@age18 likes Berlin@1920ies .
```

nor as attached to the property:

```
Alice likes@age18 Berlin@1920ies .
```

The latter however might seem like a somewhat questionable modelling decision. Attaching a proper date to the property instead, e.g.:

```
Alice likes@2022 Berlin@1920ies .
```

would seem to propagate just fine.

[31]To illustrate this point the following examples introduce an annotation syntax using an `@` sign to add a temporal qualifier to terms (and other constructs). That syntax is made up as RDF does not provide any means to qualify IRIs any further.

[32]The term applies also to 3D+1 annotated properties but is more often used for nodes in the 4D approach. The term is also used for other types of annotations besides temporality.

Likewise the discussion doesn't differentiate between annotations on a statement itself and *all* the nodes it is comprised of — a question that becomes even more relevant when the annotation is attached not to a statement but to a graph.

The decision on which approach to pursue has consequences on different aspects. As mentioned already not every modelling style is equally intuitive. But also not every modelling style is equally easy to reason with, as will be discussed below in Section 9. Also not every modelling style lends itself equally well to more differentiated scenarios where nodes, statements and graphs or some combination of them all need to be annotated.

Hayes' categorization of 3D, 3D+1 and 4D approaches may seem to map quite naturally to the well-known concepts of contextualization, annotation and qualification. However, as will be discussed in Section 14, the interrelations are more complex than that.

4.2.1 Datatypes

> *Literals are extremely important since they are, after all, the carriers of the data that is eventually processed. In fact, we argue that IRIs are only crucial insofar as they offer a way of traversing linked data towards the discovery of literal values.*
>
> Maxime Lefrançois and Antoine Zimmermann[33]

As discussed in Section 2.5.1 above, RDF reuses datatypes as defined in XML Schema (XSD)[34] — a comprehensive but far from exhaustive set of basic datatypes — and adds two additional datatypes, `rdf:HTML` and `rdf:XMLLiteral`. Missing however are RDF specific datatypes for constructs like triples or graphs, and also more complex datatypes encoding multipart values like geographical coordinates or values and units. Beyond some isolated areas datatypes haven't received a lot of attention although they do provide an interesting starting point. [CBR18] extend the OWL DatatypeProperty by a new domain type RoleDomain to capture one or more roles a property plays. As a downside this approach requires renaming properties to address individual statements.

4.2.1.1 Complex Datatypes

There are pros and cons to representing complex data structures either as compounds of RDF statements or as datatyped literals. Encoding a complex value in a datatype makes its inner parts unreachable to RDF and OWL native operations, but it enables more efficient encoding and tailored operations in specialized applications. Geo coordinates for example are sufficiently predictable in structure and sufficiently complex to be worth the effort, operations on them are geometric/algebraic rather than logical and they use specialized engines.[35].

[33] [LZ16]
[34] [Pet+12]
[35] `https://lists.w3.org/Archives/Public/semantic-web/2020Jul/0120.html`

Consequently the Geographic Information System (GIS) community has standardized structured literals for coordinates in GeoSPARQL many years ago [36]. GeoJSON is a related effort that's seeing increased adoption in the RDF world alongside JSON-LD.

[LZ16] investigate ways to facilitate the support of custom datatypes in RDF. To that end they discuss requirements and means to define custom datatypes and make such custom datatype definitions discoverable on the web. They develop a minimal set of functions to process such datatypes by query engines and reasoners on the fly. Those functions are made available as JavaScript code to be downloaded by an RDF engine on encountering the associated datatype. The proposal is implemented for the Jena triple store and ARQ query engine, and evaluated on a real world DBpedia use case.

[LZ18] focus on support for the Unified Code for Units of Measure (UCUM), "a code system intended to include *all* units of measures being contemporarily used in international sciences, engineering, and business." By means of a single datatype `cdt:ucum`[37] their implementation supports lightweight descriptions and querying of physical quantities in practically unlimited variations. More specific datatypes such as `cdt:length` and `cdt:speed` are provided to more readily express the kind of value being encoded. A plugin for the Jena triple store provides specific support for these datatypes, especially comparisons of and arithmetic operations on values.[38]

4.2.1.2 Graph Literals

[Her10] outlines a proposal for adding an RDF graph literal datatype to RDF. The goal of the proposal is to "explore what can be achieved with a *minimal* departure from the current RDF model, keeping an ease of implementation and transition from current systems in mind." The proposal doesn't require any change to the semantics of RDF nor the introduction of a new primitive or other extensions to the RDF model and syntax. [39] Besides defining syntax and semantics of graph literals it expands on further details like the conversion of a graph literal into a proper and asserted graph, functions to determine if graph literals are equivalent or subgraphs of each other, some syntactic sugar etc.

As already mentioned in Section 2.7.3.2 the semantics of graph literals can be understood as quoting a graph, i.e. as a reference to a referentially opaque occurrence. It is similar to RDF standard reification in that is doesn't assert the graph it quotes but it is different in that it quotes the graph, i.e. unlike RDF standard reification it doesn't refer to the meaning of the graph (e.g. IRIs in the quoted graph don't co-denote).

[36]`https://www.ogc.org/standards/geosparql`

[37]`cdt:` referring to `http://w3id.org/lindt/custom_datatypes#`

[38]A public playground is available at `https://ci.mines-stetienne.fr/lindt/playground.html`

[39]A. Zimmermann posted a similar proposal, or rather a reminder on its virtues, along with a short treatment of its entailment properties to the Semantic Web mailing list, `https://lists.w3.org/Archives/Public/semantic-web/2021May/0052.html`

This semantics is useful for application scenarios that require syntactically faithful representations of RDF graphs. It also exhibits good usability: the intuition evoked by a string enclosed in parentheses immediately matches the formal semantics very well. It is however not very suitable for contextualization or qualification purposes as the graph literal doesn't assert the graph, i.e. like an RDF standard reification quad it only describes some occurrence but doesn't refer to one specific occurrence directly.

Implementing graph literals would require only a minor change to existing RDF engines as the datatype mechanism is already in place: the only change required is adding a new datatype for RDF literals. Being able to query such a literal graph in SPARQL would require an extension, albeit not a major one. Blank nodes however are local the graph literal and cannot — or at least not easily[40] — bridge the gap between graphs and graph literals.

4.2.2 Terms

The terms that a statement is composed of — subject, predicate and object — are the most immediate handles for attribution. Adding to the attractiveness of this approach is that it largely avoids conflicts with the triple-based semantics and reasoning regime. However, shared identifiers with an agreed upon meaning are crucial for the functioning of the whole Semantic Web. The danger that every approach targeting the terms directly has to confront is that meddling with these terms may jeopardize interoperability.

[Sah+10] propose to annotate the nodes in a statement directly — either only the subject or the subject and predicate or the subject, predicate and object — with provenance information, creating so-called *provenance contexts*. They thus eschew any need to reify a statement or employ other means like blank node indirections, named graphs etc. A provenance context itself may be described by a single node or a more elaborate set of statements. Annotating a subject term alone puts all statements in which the term occurs in the subject position into the same provenance context. Likewise annotating a subject and a predicate puts all statements with values for that pair in a provenance context. Disambiguating different provenance contexts in which a term occurs will necessitate more fine-grained annotations. Data structures that are more tree-like than graph-like profit from succinct annotations on the subject alone. The proposal essentially reverses the intuition that many other approaches follow: it doesn't create more fine-grained entities but piggy-backs on existing data. The proposal is idiomatic, straightforward to query, doesn't interfere with OWL reasoning and is rather intuitive w.r.t. the provenance use case that it aims for. Qualifying annotations on the other hand often are associated with opposite intuitions, calling for sub-divided representations of entities as they are discussed next in Section 4.2.2. If, however, those intuitions are well-founded is discussed in Section 14.2.

[Liv+13] develop a vocabulary similar to RDF standard reification[41] to

[40]The RDF-star semantics, see below, makes an attempt to that effect.

[41]See Section 4.2.4.1 below

annotate statements with provenance not only of the whole statement, but as detailed as down to the level of individual nodes “to document the source of individual statement elements that are used to construct triples.”

[Mee+17] capture changes through data processing, e.g. parsing of complex value types, at the term level. Processed terms are represented via n-ary relations: in the output graph the respective term is replaced by an intermediate node to which the actual processed term is linked via `rdf:value`. Any documentation of provenance, processing instructions etc. may then be attached to the same intermediary. Statement level provenance however is not discussed.

RDF provides the property `rdf:value` to describe a primary value, discussed in Section 2.4.1 above. Other vocabularies to qualify terms, e.g. as playing ‘roles’, are discussed in Section 7.3 below.

Fluents The term fluent was introduced into Computer Science by [MH69] and there defined as a function that relates objects with situations. A common application is that of time and an object is then understood as a sort of worm-like entity, of which each slice corresponds to its temporal representation at a certain point in time. [WF06] target the representation of time as a contextual aspect of relations. As their project heavily makes use of OWL based inferencing they aim for a solution that conveys the essential facts in idiomatic binary relations but enriches them with temporal information. To that end they propose to represent the participants in relations as time-sliced instances of the original entities. The example from Section 4.2 would then be expanded as follows:

```
:NodeA@timeX :relatedTo :NodeB@timeX .
:NodeA@timeX :temporalPartOf :NodeA ;
             :temporalExtent :timeX .
:NodeB@timeX :temporalPartOf :NodeB ;
             :temporalExtent :timeX .
:timeX a :PointInTime .
```

The examples by [WF06] go into more detail about the participants in the relation but don’t discuss the semantics of time slices on only one participant or the property alone, or how the relation itself would be annotated, e.g. for provenance recording purposes.

Regarding usability the authors note that the approach “appeals to users with a scientific background, and is confusing to others.”

[Bat+17] compare the modelling of time slices via reification, n-ary relations or as fluents. They discuss specifically if and how property characteristics such as symmetry and inversion can be preserved by these modelling styles and find that fluents perform best in that respect.

[ZG17] present an approach similar to [WF06] but with a much more sophisticated concept of contextualization. This and other works on annotating and contextualizing terms will be discussed in Section 9 on reasoning below, as they touch representational aspects only in passing, if at all.

4.2.3 Properties

Properties present an attractive opportunity to get a grip on RDF triples without extending the RDF model. However, properties are also the area where Semantic Web standardization is most valuable and most important: they represent the central part of the shared vocabularies that drive interoperability on the Semantic Web. Developing, establishing agreement and fostering reuse of ontologies takes a lot of effort and makes them a precious resource. Meddling with commonly agreed upon properties jeopardizes interoperability and should be avoided if not essential, and very straightforward.

4.2.3.1 Name Nesting

[GP13] mentions the 'Name Nesting' logical pattern as a common Semantic Web practice. Here the names of one or more of the arguments to a multi-valued relation are nested in the name of the property. Their running example is Garibaldi's *Expedition of the 1000*, where Garibaldi in 1860 landed on Sicily with an army of 1000 soldiers. Encoding that event using name nesting results in the statement:

```
:Garibaldi :landsOnSicilyIn1860With :1000Soldiers .
```

Location and date of the landing of Garibaldi's army are nested into the 'lands' property. This technique "disrupts the relation footprint", as the connection to the property 'lands', which may stem from a well-known shared vocabulary, is lost.

4.2.3.2 Annotated Property

[URS06] present an annotation formalisms that combines annotations from any desired dimension as long as its members form a partially ordered set, like e.g. time, time-intervals, certainty etc., complete with model-theoretic semantics and algorithms for consistency checking and query answering. Syntactically the annotations are attached to the property, e.g.:

```
:Adam rdf:type:(0.85,1999) :AcademicResearcher .
```

is meant to state that with 85% certainty Adam was an academic researcher until 1999. However, the paper is sketchy on syntactic details.

[Gan11] discusses a modelling approach in which, following the "singleton property logical pattern with punning", a subproperty is created and annotated via `rdfs:domain` and `rdfs:range` declarations. Using again his running example of Garibaldi's *Expedition of the 1000*, the event could be encoded as follows:

```
:landsOn rdf:type owl:ObjectProperty .
:with1000SoldiersIn1860landsOn rdfs:subPropertyOf :landsOn ;
    rdfs:domain :Garibaldi ;
    rdfs:range :Sicily ;
    :with "1000 soldiers" ;
    :holdsAt 1860 .
:Garibaldi :with1000SoldiersIn1860landsOn :Sicily
```

His example actually omits the last statement, probably relying on the domain and range declarations alone. That might seem a bit daring, but domain and range declarations could be used to annotate the nodes separately and independent from the whole relation, e.g.:

```
:landsOn rdf:type owl:ObjectProperty .
:with1000SoldiersIn1860landsOn rdfs:subPropertyOf :landsOn ;
    rdfs:domain [ :with "1000 soldiers" ] ;
    rdfs:range [ :holdsAt 1860 ] ;
    rdfs:label "Expedition of the 1000".
:Garibaldi :with1000SoldiersIn1860landsOn :Sicily
```

4.2.3.3 Singleton Property

Singleton properties became more prominent with a proposal by [NBS14] to create contextualized properties. For each context that a property occurs in they create a new *Singleton Property*. That context specific property is related to the original property via a `rdf:singletonPropertyOf` relation, e.g.:

```
:Alice :buys#1 :Car .
:buys#1 rdf:singletonPropertyOf :buys ;
    :date 2012-06-18 .
:Alice :buys#2 :Car .
:buys#2 rdf:singletonPropertyOf :buys ;
    :date 2021-12-13 ;
    :source :Bob .
```

describing that Alice has two times bought a car, and the second purchase was reported by Bob.

The authors argue that defining a new property type `rdf:singletonPropertyOf` is necessary because re-using either `rdf:type` or `rdfs:subPropertyOf` would lead to undesirable interactions with established uses of those properties (stemming predominantly from the fact that they envision the singleton property mechanism to be used in *every* statement and thus overwhelming other intended uses of `rdf:type` and `rdfs:subPropertyOf`).

Use Cases and Usage Patterns [NBS14] exclusively discuss administrative metadata dimensions like provenance, but not qualification of the statement as a whole nor its nodes or the property itself. They define the singleton property as a subproperty of `rdf:type` instead of `rdfs:subPropertyOf`, which might rule out any qualifications.

[NS17] drop the notion of a singleton property being an instance and loosen the perspective on use cases. In their running example a marriage relation between two persons is qualified as taking place at a certain location. The annotation concerning the relation in the example applies to the fact described by the statement. No other example is given, but it is easy to see that annotating a singleton property with information concerning both the statement itself — as suggested in [NBS14] — and the fact that the statement describes can lead to confusion. Nothing is said about annotations on individual terms.

Taking inspiration from [Gan11] above, one could think of ways to annotate individual nodes in a singleton property relation. The following example uses domain and range declarations on the singleton property to qualify the nodes, stating that another car bought by Alice was a vintage model and that at the time she was 32 years old:

```
:Alice :buys#3 :Car .
:buys#3 rdf:singletonPropertyOf :buys ;
    rdfs:domain [ :age 32 ] ;
    rdfs:range :Classic .
```

[Pat18] adds another perspective, arguing that since the singleton property "can only have one triple in its extension [...] [it] can then be used as the subject of other triples that are considered not to be about the singleton property itself but instead are about its single triple." He illustrates that approach with the following example:

```
:castMember_1 rdf:singletonPropertyOf :castMember .
:GWTW :castMember_1 :ClarkGable .
:castMember_1 :characterRole :RhettButler .
```

Performance Problems The fact that each annotated statement has a unique singleton property can lead to problems with standard Semantic Web code bases. On the Semantic Web properties are highly standardized, as they are the central part of vocabularies, and shared vocabularies are the corner stone on which integration of data from different sources relies. In practice there are at most a few thousand properties in common use and database index structures and query optimization algorithms are tuned for such relatively low numbers of predicates (low compared to the number of different subject and object nodes that is). Singleton properties break that assumption and establish "a very uncommon uniform distribution of properties"[42], making some databases struggle as observed by [Her+16][43]. [NBS14] mention explicitly with respect to their experience with the Virtuoso RDBMS that "since the meta query has certain singleton graph patterns, creating indexes or views of those patterns might help."[44]

[Fre+19b] in a proposal called 'Companion Property' groups singleton properties by subject to ease indexing. However, the main downside of this approach is that it adds one more indirection to all operations on the data, leading to an overall increase of complexity. With more involved queries it increases the number of join operations such that query optimization becomes unreliable and initial performance gains dwindle.

Instance, Subproperty, 'Having' [NBS14] declare that `rdf:singletonPropertyOf` is a subproperty of `rdf:type`, making a singleton property an instance

[42] [Fre+19b]
[43] However, 3 years later [Fre+19b] report better results
[44] For an overview of indexing patterns employed in popular RDF databases see e.g. [Big+20]

of the property it is derived from.[45] However, the model-theoretic formalization just states that "$x_s \in IP_s$ if $\langle x_s, x^I\rangle \in I_{EXT}$(rdf:singletonPropertyOf^I), x $\in$ IP", without making any reference to a subproperty relationship to `rdf:type` although the accompanying text repeats the claim: "A singleton property x_s is an instance of a generic property x if they are interconnected by the property rdf:singletonPropertyOf, where x is called a *generic property.*"

[Ngu+15] on the contrary argue that `rdf:singletonPropertyOf` is a subproperty of `rdfs:subPropertyOf` because the singleton property is less generic than the property is is derived from. They do however not mention `rdf:type`, nor the difference to the earlier work, nor do they explain their change of mind.

[NS17] do not address the issue at all, instead giving the interpretation that the singleton property is neither an instance nor a subtype of the general property it is derived from, but rather an entity of its own kind. In their words a generic property *'has'* singleton properties.[46] However, an entailment relation is defined to the same effect as a `rdfs:subPropertyOf` relation.

Entailments [NS17] provides a detailed account of reasoning with singleton properties in RDFS and OWL, specifically on how contextual annotations interact with standard RDFS and OWL features like inverse, functional, transitive, sub-, etc. properties and how annotated and non-annotated statements successfully interact and support each other.

One issue of particular interest is how a singleton property relates to the generic property it is derived from in practice, i.e. if:

```
:Alice :buys#1 :Car .
:buys#1 rdf:singletonPropertyOf :buys .
```

entails the regular statement `<:Alice :buys :Car>`. RDFS axiomatisation supports this entailment in case the singleton property is defined to be a `rdfs:subPropertyOf` of the generic property it is derived from, as claimed by [Ngu+15]. According to the semantics defined in [NBS14] and [NS17] however this entailment needs to be triggered explicitly and the latter work provides the necessary axiomatisation. Therefore, unlike RDF standard reification the singleton property approach does actually assert the statement it annotates. However, to express the unannotated statement it needs to carry out an extra step of — according to the latest version non-standard — reasoning.

Multiple Singletons The use cases and examples that [NBS14] bring forward concentrate on metadata dimensions like provenance, time, location and validity of statements where many singleton properties would indeed occur only once. The authors stress that their "approach is based on the intuition that the nature of every relationship is universally unique. A singleton property is a property

[45]The paper states that "we define the `singletonPropertyOf` property as a sub property of `rdf:type`."

[46][NS17], Section 3.1: "For example, `isMarriedTo` is a generic property and it has two singleton properties".

instance representing one specific relationship between two particular entities under one specific context.[. . .] While creating a new statement, what we actually do is connect two existing entities and establish a new relationship between them." Following publications by [Ngu+15] and [NS17] only strengthen that argument.

The authors do explicitly enforce the singleton design in the model-theoretic semantics and discuss means to ensure uniqueness of singleton properties in practice e.g. by employing Universally Unique Identifiers (UUID). They note that blank nodes would provide similar semantics but are forbidden in predicate position of standard RDF.[47]

However, this semantics is not always followed in practice. [Tae+16] uses the same singleton property in multiple relations to group triples under the same time label if a datastore doesn't support quads. Other scenarios in which singleton properties can be used multiple times would be plausible as well, e.g. establishing the singleton properties `:buys#Cash` and `:buys#CreditCard` to record Alice's purchases by payment type. However, they might collide with use cases that do indeed require to annotate individual statements, calling for undesirably involved measures like composition of annotations, inheritance etc.

Singleton Properties and Named Graphs [NS17] discuss the transformation of existing meta-modelling approaches into singleton properties. For named graphs they advise to add the graph name as an argument to the singleton property: each statement contained in a named graph, otherwise annotated or not, is to be replaced by three statements, e.g.:

```
:Graph_1 { :Alice :buys :Car . }
```

is to be converted[48] to:

```
:Alice :buys#1 :Car .
:buys#1 :singletonPropertyOf :buys ;
    prov:wasDerivedFrom :Graph_1 .
```

Given the popularity of named graphs among real world deployed RDF data sources, this is a rather daunting proposition. Of course the increase in overhead is less dramatic in case a statement also gets annotated in any other way.

[Idr+17] take a different route and try to combine the strengths of both approaches. Their proposal to solve the sameAs-problem[49] contextualizes equality relations in 'correspondences', organized as 'linksets'. Singleton properties are used to annotate statements about individual correspondences with metadata. Additionally, the singleton properties are organized in a property hierarchy to "capture context-specific notions of equality at different levels", and linksets collect statements describing individual correspondences in named graphs to which then metadata about the whole linkset is attached.

[47]Generalized RDF doesn't impose that restriction, see https://www.w3.org/TR/rdf11-concepts/#section-generalized-rdf

[48]Using the well-known Prov-o provenance ontology https://www.w3.org/TR/prov-o/

[49]See Section 5.2.1

Related Approaches [Ngu+19] explore the application of singleton properties to the problem of integrating the Semantic Web and Labelled Property Graphs and are discussed below in Section 4.2.4.6.

[GZ18] take inspiration from [NBS14] to represent statements by their predicate, but also from [GZM17] to separate contexts. Their results, which focus on contextualized reasoning, are discussed in Section 9 below.

[Ngu+16] stress that standard RDF favours nodes to edges. While it allows to make properties the subjects of statements, those properties will never show up in queries employing the idiomatic "follow your nose" procedures and navigation style. To remedy this problem they present a new graph model for RDF that provides better support for a property-centric approach like singleton properties — see Section 11 for details.

4.2.4 Statements

Statements are not the most important building blocks of RDF — that honour can't go to one single element of the formalism alone — but they are easily the most recognisable. However, statements, or triples as they are more colloquially called, are the most important *structure*, as they are the only harbinger of truth in RDF. Statements are absolute, as no statement is allowed to change the truth value of another statement. Statements also are, despite the standardization of named graphs in SPARQL and RDF 1.1, the only structure with formal, model-theoretic semantics in RDF. Yet, or rather because of their paramount importance, RDF provides no way to address statements in other statements. They are untouchable and that is not an oversight, but a precaution. For one thing it guarantees monotonicity, see Section 2.6.2.3, an indispensable feature to establish an open world semantics. Secondly, it prevents the introduction of paradoxes, as discussed in Section 2.6.3 above.

RDF provides the standard reification vocabulary that as discussed above doesn't refer to the statement itself — which is a type, an abstraction — but to an occurrence, a speech act. This allows for recording of provenance, a popular use case, but with a catch: an RDF statement that it might seem to describe is still categorically different from the occurrence that is actually described.

To give an example how this can be problematic in practice: one might add a statement to a graph, describe it by RDF standard reification, note time of assertion, and then remove the statement but not the reification. The reification can now be understood as describing an unasserted statement: a not uncommon interpretation, although not covered by the RDF specification. Let's now add the statement a second time, and one arrives at a graph that obviously is syntactically correct. However, is it also factually correct? The statement was indeed added to the graph (the first time) at the date described by the reification. Still in the current graph the statement was added later (a second time) and not at the time that the reification states (referring to the first occurrence which has then been deleted). Therefore, the facts are correct, but incomplete in a non-obvious way, resulting in a misleading description.

Annotating the triple *itself* seems like a very attractive proposition, but this is tricky territory. The semantics of statement types, reification, qualification etc. has been discussed at length above in the first Part of this work and shall not be repeated. Some approaches presented in the following try to bridge the gap between type and occurrence in one way or another, but most of them do not provide a precise and sound semantics. If they do provide a semantics, that semantics doesn't necessarily reflect the intuitions of users — as just demonstrated in the lengthy example above — leading to usability problems and unsound triples in practice.

4.2.4.1 Standard Reification

As mentioned already in Section 2.4.3 the RDF standard reification vocabulary is not exactly well-liked among Semantic Web practitioners and its deprecation has been discussed as early as during standardization of RDF 1.0 in 2004. [Sah+10] for example argue that using RDF standard reification to track provenance incurs a number of problems, namely "limited formal semantics", ensuing "application-dependent interpretation of reified RDF triples", the use of blank nodes and the negative impact on scalability through reification triples. [TKK13] give a number of reasons to not use RDF standard reification: well-known syntactic verbosity and cumbersome querying, ambiguous model-theoretic implications like possible violations of the Open World Assumption, mismatches with established meanings in conceptual modelling and linguistics, best practices in Linked Data publishing.

Nonetheless the reification vocabulary is part of the RDF recommendation and for that reason is readily available and supported in all major code bases. This brings with it some advantages as e.g. described by [Ism+18] concerning the integration of Wikidata in DBpedia. In contrast to [Erx+14] they use RDF standard reification to represent qualified statements, arguing for this to "lead to a simpler design and further reuse of the DBpedia properties." As DBpedia already uses RDF standard reification, this also helps to provide a unified query interface over the combined data. The mapped data is split in different datasets, separating statements from their reifications and thus providing chances to improve performance. [HHK15] compare the performance of encoding Wikidata qualified statements with RDF standard reification, n-ary relations, named graphs and singleton properties. There is no clear overall winner and despite its bad reputation reification does fare quite well in comparisons. An RDF export of Wikidata introduced by [Erx+14] settles for another, home-grown schema.

Some projects using RDF standard reification today still stick with RDF/XML as their preferred serialisation, despite the advantages that Turtle or JSON-LD would provide, because in RDF/XML the ID attribute[50] keeps the reification quad out of sight, making their lives considerably easier.[51]

[50]See Section 6.1

[51]As reported in an RDF-star use case, `https://w3c.github.io/rdf-star/UCR/rdf-star-ucr.html#uniprot-attributed-evidenced-triples`

[MK03] provides some historic background on similar reification vocabularies in Knowledge Representation systems in the field of Artificial Intelligence, predating the Semantic Web.

[GHV07] present a framework to incorporate temporal reasoning into RDF, based on annotating single statements as opposed to graphs as they believe that this better "preserves the spirit of the distributed and extensible nature of RDF." To that end they define a model and syntax very similar but not identical to RDF reification, "in order to stress the fact that the notions presented in this paper are independent of any view one may have about the concept of reification in RDF." This approach is picked up by [Zim+12].

[Liv+13] also develop a vocabulary similar to RDF standard reification, here aimed at annotating statements as detailed as down to the level of individual nodes. The resulting graph however is even more verbose as each node in a statement requires two reifying statements to be annotatable, similar to approaches discussed in Section 4.2.2.

[HHK15] sketch a syntactic alternative that shares the semantics with RDF standard reification, but cuts its triple count in half:

```
:tripleID rdf:subject :S .
:tripleID :p :O .
```

However, what is won in succinctness is lost in clarity — or how would one distinguish an annotation on the triple from its original property and value?

4.2.4.2 IRI-encoded Triples

[Eck13] develops a twofold extension to RDF standard reification. First he introduces the notion of a *provenance context*, defined as a cascade of either document URL, DataSet containment or explicit denotation. The cascading definition ensures that each statement is contained in no more than one context. He then defines a syntax for addressing a statement in such a context: a statement is addressed through its context IRI, extended by a query parameter that contains the percent encoded subject, predicate and object of the statement in question. Such an IRI is always dereferencable to its constituents, e.g.:

```
<http://example.org/provcontext1?spo=%3Chttp%3A%2F%2Fexample.org%2Fdata%
 2Fdoc1%3E,%3Chttp%3A%2F%2Fpurl.org%2Fdc%2Fterms%2Fcreator%3E,%3Chttp%3A
 %2F%2Fexample.org%2Fpersons%2Fkai%3E>
    a rdf:Statement ;
    rdf:subject <http://example.org/data/doc1> ;
    rdf:predicate <http://purl.org/dc/terms/creator> ;
    rdf:object <http://example.org/persons/kai> ;
    dm2e:context <http://example.org/provcontext1> .
```

Obviously the resulting IRI is rather hard to read due to the unavailability of namespacing. Diverging from RDF standard reification this approach assumes that there is only one occurrence per type and per context. It can't address multiple occurrences of the same triple. The construction of the statement IRI makes little use of the fragment identifier — "We use the fragment identifier in the query part as it allows us to dereference the URI as usual and to provide explaining statements to clients that do not directly support our fragment identifiers" — leaving room for further refinements.

[Knu21] reports on an approach implemented in the TopBraid IDE[52] that encodes triples as URNs of the form `<urn:triple:${enc(S)}:$enc(P):${enc(O)}>`, where `enc(N)` is the URI-encoded Turtle serialisation of a given node N, optionally using provided prefix mappings. A short example is also given, showing the annotation of a statement:

```
ex:Bob ex:age 23 .
<urn:triple:ex%3ABob:ex%3Age:23>
    ex:date "2019-12-05"^^xsd:date ;
    ex:author ex:Claire .
```

Compared to [Eck13] above this approach lacks the definition of a statement's context and doesn't generate standard IRIs but defines its own URN scheme. That allows it to make good use of Turtle's prefix mapping mechanism. [Bol19] elaborates on the URN approach in the context of the UniProt project.[53]

[Ism+18] uses the standard RDF reification mechanism to capture provenance and other metadata from an import of Wikidata into DBpedia, but it takes advantage of Wikidata's concise identifiers and encodes the subject, predicate and object of a triple in the statement identifier.

[Tae+16] construct triple identifiers as IRIs, interpreting triples as singleton Triple Pattern Fragments (TPF), also referred to as 'implicit graphs'. For example "the URI for the triple `<s> <p> <o>` on the TPF interface located at `http://example.org/dataset/` is `http://example.org/dataset?subject=s&predicate=p&object=o`."

4.2.4.3 Triple + id

There are many solutions that 'add a fourth field' to a triple, to use popular database terminology. There do however exist important differences in how the fourth element is used:

Triple + id The fourth element acts as a statement identifier, making the individual statement addressable in further statements

(Named) Graph Here the fourth element is a grouping device that adds the statement to a graph, often called the named graph (with many possible semantics as discussed in Section 2.7.3)

Quad The fourth element can also act as a regular data field to express additional, application specific information e.g. date of ingestion, provenance, credibility etc. Here the transition to quintuples and beyond is fluid.

This section discusses the first variant, whereas graphs and quads are discussed below in Section 4.2.5.

[MBS15] present YAGO3, a knowledge base from multilingual Wikipedias. In an online FAQ the data format of YAGO3 is described as an extension of Turtle

[52] `https://archive.topquadrant.com/products/topbraid-composer/`
[53] `https://www.uniprot.org/`

to an ‘N4’ format in which every triple can have an identifier. This so-called *fact identifier* is “specified as a comment in the line before the triple. As a result, all N4 files are fully backwards compatible with standard Turtle and N3.”[54] Fact identifiers can be used in further statements to annotate statements with qualifying detail about temporal and spatial validity as well as provenance information.

[HHK15] explore different options to represent qualified statements from Wikidata in RDF, namely RDF standard reification, named graphs, n-ary relations, singleton properties and RDF*. For their internal representation they settle for a combination of triples and triples+ID: statements that are further annotated are stored together with an identifier. Statement annotations that themselves are not attributed are stored as triples. Differing from RDF’s set based semantics the same statement type can occur multiple times, each with different identifiers, to represent the statement in different contexts. Re-using the example of [Hog18] below, a chilean president, Sebastián Piñera, who served two non-consecutive terms, and using a made up syntax of Turtle plus triple identifiers this could be encoded as follows:

```
:Pinera :president :Chile   t:1 .
:Pinera :president :Chile   t:2 .
t:1 :start 2010 ;
    :end 2014 ;
    :replacedBy :Bachelet .
t:2 :start 2018 ;
    :replaces :Bachelet.
```

[Hog18] adds another variant to possible representations of Wikidata: ‘Hoganification’. This approach is not meant to be a serious proposal but rather to illustrate that there are a plethora of options to serialize an annotated relation. It describes statements in a way similar to RDF standard reification. From this description it derives actual occurrences that are then subject to further annotations, e.g.:

```
_:b :subject :Pinera ;
    :predicate :president ;
    :object :Chile ;
    :statement s:1 ;
    :statement s:2 .
s:1 :start 2010 ;
    :end 2014 ;
    :replacedBy :Bachelet .
s:2 :start 2018 ;
    :replaces :Bachelet.
```

[Vrg+21] present a graph database targeted at both RDF and Property Graphs. To support the two approaches equally well they employ a data model based on triples complemented by an identifier. That identifier is not accessible in the RDF data model but can be used with RDF models to implement RDF* or named graphs and with Property Graph models to annotate edges.

[54] https://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago/faq/

[Fre+19b] report on statement identifiers in the Stardog triple store[55] which supports statement annotations and is queryable via a proprietary SPARQL extension. The statement identifiers also support Stardog's implementation of Property Graphs. Similar arrangements are known for other codebases, see Section 10 for more details.

[Ang+22], in a position paper representing several graph database projects and products, argue "that there is a need for a unifying data model that can support popular graph formats such as RDF, RDF* and property graphs, while at the same time being powerful enough to naturally store information from complex knowledge graphs, such as Wikidata, without the need for a complex reification scheme." They advocate a data model based on triples with edge identifiers, to account for the "need to be able to reference an edge as it if were a node (or possibly multiple nodes)."

[Las+] discusses the differences between RDF and Labelled Property Graphs with the help of statements and statement identifiers. The article stresses that the choice of this notation just serves demonstrative purposes and "does not imply a necessity (neither does it suggest a preference) for a triples-based or a quads-based physical indexing scheme for graph data", but anecdotal evidence suggests that this is also the basic data structure driving the Amazon Neptune[56] graph store.[57]

4.2.4.4 RDF*

[HT14] propose to extend the RDF model and syntax by a new element: the embedded triple.[58] Syntactically the embedded triple is an RDF triple that is enclosed in double angular brackets.[59] It can occur as a node in subject and object position of other triples, but, as it is only a node, it cannot stand alone. [HT14] give the following example, which is also the running example in following publications:

```
:bob foaf:name "Bob" .
<<:bob foaf:age 23>> dct:creator <http://example.com/crawlers#c1> ;
                     dct:source <http://example.net/homepage-listing.html>
                     .
```

Embedded triples can be nested in other embedded triples, recursively enabling annotations on annotations (although then any syntactic elegance tends to get lost in the ensuing verbosity of the representation). Embedded triples are also asserted, making the above example syntactically equivalent to the following eight triples if RDF standard reification was used:

[55] https://www.stardog.com/

[56] https://aws.amazon.com/neptune/

[57] Brad Bebee of Amazon Web Services in a talk at Stanford university shows how they put every triple in its own graph because they want to be able to annotate them one by one: https://web.stanford.edu/class/cs520/abstracts/bebee.html, https://youtu.be/4agR9qoS73o.

[58] The article uses the term "embedded" whereas later articles call the triple "nested". More recently with the advent of RDF-star, see below,"quoted" has become the preferred term.

[59] In the prototypal Turtle* and SPARQL* serialisations. Other serialisations may use other syntaxes.

```
:bob foaf:name "Bob" ;
     foaf:age 23 .
_:s a rdf:Statement ;
    rdf:subject :bob ;
    rdf:predicate: foaf:age ;
    rdf:object 23 ;
    dct:creator <http://example.com/crawlers#c1> ;
    dct:source <http://example.net/homepage-listing.html> .
```

The proposal includes an extension to SPARQL, named SPARQL*, that defines syntax and semantics for querying the embedded triple and to Turtle, named Turtle*, for serialisation.

[HT14] tout RDF* as essentially just syntactic sugar for RDF standard reification, its main advantages being a much more compact representation in serialisations and storage as well as a much more compact syntax when querying RDF data. However, the claim that RDF* embedded triples are syntactic sugar for RDF standard reification proved unfounded. To ground the semantics of the embedded triple in standard RDF, [HT14] define a transformation that maps each embedded triple to a newly minted blank node that is the subject of an RDF standard reification quad, as seen in the above example. But there is a catch: RDF standard reification separates the definition of an occurrence from an RDF triple itself which represents a type. Because of that, multiple different occurrences of the same triple can be defined, and annotated separately. RDF* embedded triples on the other hand don't refer to occurrences, but to the triple itself, the *type*, no matter how often the embedded triple occurs in the data.

The approach of RDF standard reification is useful to e.g. describe that some triple has been stated by different sources at different times. Ironically, although this provenance use case is employed also in the RDF* standard example cited above, an RDF* based solution needs to introduce an intermediary node to record two different pairs of creator and source. Otherwise multiple pairs of annotations will be smushed together.[60] For [HHK15] this design decision is reason enough to not consider RDF* as a possible basis to represent Wikidata in RDF. Another aspect to take into account: the fact that the RDF standard reification quad doesn't assert the statement it describes is sometimes used to make assertions about statements that one doesn't want to endorse. However, although RDF* doesn't support this use case it doesn't prevent it either, if one is prepared to use the RDF standard reification vocabulary for that task. It should also be noted that no mapping from RDF standard reification to RDF* is provided. It is therefore not clear how to integrate the two approaches, to e.g. convert legacy data to RDF*,

[Har17] extends the work of [HT14] and provides a more detailed formalization of the approach, complete with necessary extensions to the RDF data model and the SPARQL query language, mappings to convert RDF* and SPARQL* back to RDF and SPARQL with information preservation and query result equivalence. However, like [HT14] it does not provide a model-theoretic semantics, but defines

[60]See Section 13 for a more detailed discussion of the issue.

an operational semantics of RDF* in terms of SPARQL* query results and then a mapping from RDF* to RDF that preserves those query results.[61] Also, the paper gives two semantics, a direct and an indirect one, but unintentionally they diverge and implementations appear to implement the direct semantics, not the indirect one.[62] [OGO20] examine how RDF* is implemented in 3 popular RDF databases and find some discrepancies and divergent implementations.

4.2.4.5 RDF-star

The RDF* approach found some support in the Semantic Web community, being perceived as an elegant and straightforward solution to a nagging problem. It was presented and commended at the 2019 W3C Graph Workshop[63] as a fitting approach to help bridge the gap between Property Graphs (see Section 4.2.4.6 below) and RDF[64] — despite the fact that [HHK15] had in 2015 already dismissed RDF* in a similar use case, because it is not able to handle multi-edges.

Subsequently an RDF* Community Group (CG) was formed under the auspices of the W3C to prepare an eventual standardization effort based on [HT14] and [Har17]. [Har+21], the report published by the RDF-star CG in 2021 after about two years of work, changed RDF* in a few ways. Besides the obvious name change to RDF-star, some profound changes have been made under the hood:

- RDF-star *quoted* triples, as they are now called, are not asserted.
- RDF-star is now equipped with a model theoretic semantics, a very peculiar one however.
- Contrary to RDF* it is now clarified that RDF-star is *not* syntactic sugar for RDF standard reification.
- The seminal example on provenance annotations featured above is now considered "to have set wrong expectations."
- Multi-edges are now at least acknowledged, but still require additional constructs.

These changes have both obvious and subtle repercussions. To ease discussion, from now an the terms RDF* and embedded or nested triples will be used to refer to the pre-CG report version of RDF*, whereas the terms RDF-star and quoted triple will denote their variant as defined in the report published by the RDF*–turned–RDF-star CG.

Unasserted Quoted Triples The way RDF* is defined, embedded/nested triples are considered asserted. This design provides optimal compactness and

[61] `https://lists.w3.org/Archives/Public/public-rdf-star/2020Jun/0017.html`

[62] Peter F. Patel-Schneider, `https://w3c.github.io/rdf-star/Minutes/2021-02-12.html`

[63] W3C Workshop on Web Standardization for Graph Data - Creating Bridges: RDF, Property Graph and SQL, `https://www.w3.org/Data/events/data-ws-2019/`

[64] See Hartig's position statement and presentation at `https://www.w3.org/Data/events/data-ws-2019/assets/position/Olaf%20Hartig.pdf` and `W3CWorkshop2019RDFStarAndSPARQLStar.pdf`

tightly couples an assertion with its annotation. Implementations however took diverging paths on the issue. An informal survey[65] taken by the RDF-star CG in 2019 revealed that of the two handful of implementations then known roughly half implemented the RDF* semantics (at the time dubbed *Property Graph (PG) mode*), whereas the other half implemented an RDF standard reification-like semantics of unasserted triples (dubbed *Separate Assertion (SA) mode*), with two libraries implementing both semantics as options.[66] [TWS20] discuss the advantage of quoted triples with un-asserted semantics to encode temporal constraints. The following example:

```
<< :DouglasAdams schema:homeLocation :SantaBarbara >> schema:endDate 2001 .
```

according to their interpretation does not assert that Douglas Adams lives in Santa Barbara, as that would not be true *now*. This very strict interpretation of qualification has of course its own problems, as will be discussed in Section 14.2 later below. Also other use cases were brought forward, like the documentation of statements that one doesn't want to endorse, versioning of data etc. The CG eventually decided that the demand for unasserted assertions shouldn't be ignored and made it the default semantics. To regain the conciseness of the original syntax, a shorthand syntax was introduced that adds annotations to a standard asserted triple, e.g. applied to the original example:

```
:bob foaf:name "Bob" .
:bob foaf:age 23 {|
        dct:creator <http://example.com/crawlers#c1> ;
        dct:source <http://example.net/homepage-listing.html>
    |} .
```

[Del+21] develop a mapping from RDBMS to RDF-star, based on R2RML ([Har+21]) and the RDF-star CG report. The mapping distinguishes between asserted and unasserted triples, introducing special *Non-Asserted Triples Maps* to ensure that data from untrusted sources is mapped to nested triples only but not to standard RDF triples.

Extending the RDF Model with a new Node Type RDF-star quoted triples are (as in RDF*) defined as a new node type, extending the RDF model. Part of the justification for this has disappeared as they are not considered asserted anymore: IRI-encoded triples as discussed above or RDF literals, which will be discussed below, could serve the purpose equally well. Another justification for the expenditure to define a new node type is the chosen semantics for blank nodes, discussed next.

Model-Theoretic Semantics Until the RDF-star Community Group started working on it, RDF* came without a model-theoretic semantics. This was

[65]`https://lists.w3.org/Archives/Public/public-rdf-star/2020Aug/0026.html`

[66]Adding two modes to distinguish between PG and SA semantics was a proposal discussed at some point, see `https://lists.w3.org/Archives/Public/public-rdf-star/2019Sep/0051.html`, but dropped later.

a shortcoming that needed to be fixed to fully integrate RDF* into RDF.[67] Members of the Notation3 community joined the RDF-star CG and proposed a rather peculiar model-theoretic semantics, similar to the semantics of formulas in Notation3:[68] quoted triples are defined as referentially opaque, except blank nodes which are referentially transparent. This semantics makes sense when one aims for syntactic fidelity, operating close to the metal of the Semantic Web. Use cases are versioning or all the use cases that [Car+05a] list[69], but also explainable AI and reasoning with configurable semantics as e.g. [AW19] explore in the context of Notation3.

On the downside the proposed semantics neglects the needs of the overwhelming majority of use cases documented in the CG wiki, which unsurprisingly operate on the assumption of referentially transparent triples, just as the RDF semantics presets. This is problematic as the quoted triple will most of the times not serve as an information resource in its own right — a step in a reasoning process, a version to keep track of, a syntactically reliable representation of a document — but merely as a hinge between a (referentially transparent) triple and its (referentially transparent) annotation. It does not only make no sense to define that hinge with a different semantics, it may cause subtle shifts in meaning and ensuing uncertainties: either referential opacity of the quoted triple might be ignored, rendering the semantics useless, or the annotated and annotating triples might be understood to as referentially opaque, letting this semantic arrangement creep into statements for which it wasn't intended.

To achieve a conventional semantics the CG suggests a proposal called Transparency-Enabling Properties (TEP), that requires users to explicitly assert that a property is meant to operate on referentially transparent quoted triples, if referential transparency is desired. This mechanism would have to be applied per property per graph. An application or even mention of this irritatingly tedious proposal outside the CG report has never been observed. Other, less involved approaches would be imaginable. However, given that the proposed semantics is widely out of touch with the requirements and the general awareness of regular Semantic Web practitioners towards such rather subtle semantic delicacies, it seems quite unreasonable to assume that even the slightest extra effort would be accepted. A reasonable design should take care to satisfy the overwhelming need first and require special needs to go the extra mile, not the other way round.

The proposed semantics so far it is all but ignored in practice. It remains to be seen how the recently instantiated RDF-star Working Group will proceed with it.

[67]RDF* was extolled at the above mentioned 2019 W3C Graph Workshop not only as a perfect match for representing LPG in RDF but also as 'having a semantics'. This however was only partly true as RDF* had an operational semantics, defined in terms of SPARQL query results, but not a model-theoretic semantics — which is the customary meaning of 'having a semantics' when used as a shorthand in Semantic Web circles.

[68]See Section 4.2.5.3 below.

[69]See Section 4.2.5.1 below

Blank Nodes Another curious aspect about the proposed semantics is its treatment of blank nodes, which are defined as referentially transparent. This arrangement might serve as a kludge to introduce Notation3 formulas as lists of quoted triples,[70] but otherwise its usefulness is questionable as the meaning of a bnode is determined by all the statements in which it takes part. Annotating a syntactically defined representation of just one of statements while the others are left undocumented seems to introduce at least as much uncertainties as it removes.

Property Graph Compatability The suggestion that RDF* could improve the compatibility between RDF and LPG was an important driver of its popularity e.g. at the Berlin 2019 W3C Graph Workshop,[71] but it was not the main focus of the ensuing RDF-star CG report. Especially the decision to let RDF-star refer to triple types instead of occurrences creates interoperability problems. Some recent works investigate possible approaches, and hurdles, when transforming LPG to RDF-star — see Section 4.2.4.6 below, especially the discussion of [Abu+22] and [KFH22]

4.2.4.6 Property Graphs

It wouldn't be totally correct to call Property Graphs — or, more precisely, Labelled Property Graphs (LPG) — a fork of the Semantic Web, but they were developed by and for people that found the Semantic Web technology stack intimidatingly complex, yet inept to meet common modelling needs.[72] At the same time when the RDF 1.1 Working Group struggled and ultimately failed to standardize a model-theoretic semantics for named graphs as a meta-modelling technique, the LPG community came up with a simple model of annotated relations that adequately covered the most basic use case of meta-modelling: a primary relation with secondary attributes. This approach was much welcomed by people who wanted to take advantage of the intuitiveness of knowledge representation with graphs, but without all the troubles that the formally more sound, but also more unyielding RDF stack incurs.

The basic modelling primitive of LPGs is a directed edge between two nodes, with the possibility to annotate both nodes as well as the edge itself. Notwithstanding all the features of RDF that LPGs don't bother to care about this basic primitive is not much different from, but rather an extension of the RDF triple. With respect to the different types of n-ary relations discussed above LPGs resemble Davidsonian n-ary relations — see Section 2.7.4.1 — as they cleanly differentiate between nodes and their attributes, binary relations between those nodes and attributes on those relations. [SLS21] provide a tabular comparison of feature support for common property graph models and RDF.

[70]Indeed, the two instances of a correct application of the proposed semantics that this author is aware of both emulate formulas as lists of quoted triples

[71]W3C Workshop on Web Standardization for Graph Data - Creating Bridges: RDF, Property Graph and SQL, `https://www.w3.org/Data/events/data-ws-2019/`

[72]`https://www.w3.org/Data/events/data-ws-2019/assets/position/Joshua%20Shinavier.pdf`

Another clear advantage of LPGs, besides their perceived intuitiveness for meta-modelling, is that their query languages Cypher[73] and Gremlin[74] provide much better support for path queries than SPARQL. LPG databases are optimized for local searches based on graph-traversal algorithms. This gives LPGs an edge in graph analytics settings.

LPG databases soon enough rivalled the market shares of established RDF triple stores.[75] The success of LPGs has been understood by some as a call for action to RDF as it markedly confirms the point that the simplistic RDF triple as a modelling primitive can't satisfy common knowledge representation problems in practice.

However, there are downsides to Property Graphs as well. First of all LPGs are not a standard but rather a community with shared concepts but many slightly different conceptualizations and not even a standard data format to enable exchange between different implementations.[76] There is no formalized semantics and no support for reasoning. What's more, the basic modelling primitive of LPGs, the annotated relation, has it's limitations too. Nodes and relations can be annotated, but those annotations can't be annotated themselves and are merely literal values. RDF in this respect is much more expressive and versatile. Globally valid identifiers, which are ubiquitous in RDF and drive the vision of a globally distributed and shared information space, are not common in the world of LPGs. Likewise there are no shared ontologies, a further testament to the application centric and local world view of LPGs. In the words of [BSH18] "many use cases are focused on providing stores for single applications and single organisations, with less commitment to, and support for standardization, interoperability and sharing. This contrasts with the priorities in the Semantic Web World, where sharing and interoperability are primary motivations."

So although LPGs are often perceived as more flexible and intuitive in the beginning and much easier to get started with than RDF, they can become problematic and run into impasses when use cases get more involved or even include exchange of data across application boundaries. This is e.g. reported by [Las+21] who present the Amazon Neptune cloud-based graph database offering, which supports both RDF and LPG. They describe requirements and assumptions that lead customers to choose either RDF or LPGs — not always to their later benefit — and discuss possible approaches to bridge the gap between the two. They find that "information architects prefer the features of the RDF model because of a good fit with use cases for data alignment, master data management, and data exchange" whereas "software developers often choose an LPG language because they find it more natural and more 'compatible' with their programming paradigm" and "developers coming from the SQL world often like that a vertex in an LPG is much like a row in a relational database." However,

[73] `https://neo4j.com/docs/getting-started/current/cypher-intro/#_cypher_query_language`
[74] `https://tinkerpop.apache.org/gremlin.html`
[75] `https://db-engines.com/en/ranking/graph+dbms`
[76] [TAT20] provide a tabular overview of features supported by the most popular property graph database systems.

when applications get more complex or need to integrate outside sources LPG based solution start to struggle, but customers then have no easy way to switch from one paradigm to the other.

RDF & Labelled Property Graph Interoperability Investigations and efforts into RDF & LPG interoperability started quite soon after the introduction of LPGs, and haven't come to a conclusion yet. Different aspects prove hard to find an answer to. RDF's lack of the notion of edge properties is the most obvious barrier to RDF-LPG interoperability and from RDF standard reification to named graphs to quoted triples every popular approach has been tried, to varying success. Less prominent is an idiomatic aspect: RDF knows only one kind of relation, the `rdf:Property`, of which there are of course many different types. LPGs on the other hand differentiate relations between nodes and attributes to nodes. Nodes and their attributes form objects and the relations between nodes relate those objects, not their central node alone. RDF does not offer a native equivalent to the notion of an object, although named graphs and other approaches to group statements, like those discussed in Sections 4.1 and 4.3, come to mind. However even the original author of some RDF data will not always find it easy to distinguish node attributes from relations to other nodes, and much less can an automated mapping be expected to always generate 'correct' results. A third difference concerns a limitation of LPGs: attributes of nodes can't be attributed themselves. In RDF it is fairly common that attribute values form tree-like structures with sub- and sub-properties. What's more, attribute values in RDF often are IRIs with attributes and relations (to use the terminology in the LPG sense) of their own. All this is hard to represent in LPGs, and hardly 'natural'. A fourth difference is another restriction of LPGs, namely that node attributes can only be literal values. This requires a transformation to either treat IRI values as different nodes or encode them as literals, or choose between both options on a case-by-case basis depending on intended object semantics.

[TAT20] provide a tabular overview of transformation approaches, including supported formats and degree of automation.

From LPG to RDF [Das+14] examine different ways to model LPGs in RDF, namely RDF standard reification, singleton properties and named graphs. Their focus is not on precise semantics but on triple count, ease of modelling and query performance. The main problem in their view is how to encode edge properties in RDF and they find that singleton properties and named graphs are both well suited to the task. However, using named graphs to model annotated relations — the graph name equals the edge identifier, edge annotations are stored in the graph to keep the total number of named graphs at bay — may collide with other uses of named graphs. singleton properties on the other hand have a higher triple count and perform worse for queries on edge annotations, as joins get more expensive.

[Har14] presents two transformations from RDF* to LPGs and one in the opposite direction.[77] The two transformations from RDF* to LPG differ in that one is lossless, but doesn't generate idiomatic LPG because it straightforwardly translates every RDF* triple into two LPG nodes, connected by an — optionally attributed — relation. A second transformation converts RDF* relations with a literal value as object to node attributes and RDF* relations between two IRIs to LPG nodes related by the property. This creates a topology in the LPG graph that may or may not resemble the intent of the original RDF* graph. Metadata on relations that are converted to node attributes is lost in the transformation. In effect, both mappings can't properly handle the case where an attribute to a node is itself attributed. The first, lossless transformation may not loose the piece of information itself, but it will loose or distort some of the context encoded in the original structure.

A mapping in the opposite direction is mostly unproblematic: Every LPG node is mapped to a blank node, with its LPG label attributed as the value of an `rdfs:label` property. Every LPG node property is mapped to an RDF statement with the respective blank node as subject, appropriate RDF property types and the respective literal values. Every LPG edge that is not further attributed is mapped into an RDF triple with the label converted to a property type and the blank nodes representing the LPG nodes it connects in subject and object position . Every attributed LPG edge is mapped to a statement that contains the RDF representation of the corresponding edge as quoted triple in the subject position, and attribution property and value in the predicate and object position.

However, this mapping suffers from one peculiarity inherent to RDF*: the RDF* quoted triple doesn't denote an occurrence of the triple, as RDF standard reification does,[78] but the abstract triple as a type. Consequently, and in contrast to RDF standard reification, it can't differentiate between multiple occurrences with different attributes. LPGs on the other hand support multiple edges without problem, which is not surprising given the application-driven intuitions informing the approach.

[Har19] present a mapping from LPG to RDF* similar to the one in [Har14], but with one important difference: a quoted triple is not considered asserted, as was the case in the earlier work. Instead asserting and annotating a fact requires two separate statements: one of the statement itself, and one expressing the annotation on the statement, which itself is represented by the quoted triple. This change reflects discussions in the RDF-star Community GRoup at that time, see Section 4.2.4.5 above.

[AP18] identify LPG as the successor to RDF, providing more guidance in modelling, but lacking some of RDF's underpinnings. To that end they "propose to give property graphs a formal semantic grounding based on RDF/RDFS/OWL,

[77]The paper states that "RDF* is simply a syntactic extension of RDF", a claim that doesn't hold as discussed in Section 4.2.4.5. Consequently the emphasized claim that it "*formalizes*" the transformations doesn't have much weight either.

[78]The paper states that "RDF* is simply a syntactic extension of RDF", but that claim doesn't hold as discussed in Section 4.2.4.5

with blank nodes reification, geared to JSON-LD serialization." The LPG encoding in JSON-LD employs the Davidsonian n-ary pattern, with a property akin to `rdf:value` used to encode the main value. It profits from the fact that JSON-LD, like Turtle, hides blank nodes rather well, e.g.:

```
{ "@id": "StreetA",
  "hasState":{ "hasValue": "30% busy",
               "reliability": "90%" }}
```

[TAT20] present an ontology-based approach to transform LPGs into RDF. They define a set of terms to describe the elements of an LPG like node, edges, attributes and datatypes. Provided an ontology describing an LPG, a corresponding algorithm automatically transforms an LPG into RDF according to this description. However, the resulting RDF graph is not idiomatic, but much rather akin to RDF standard reification and accordingly verbose and unintuitive.

[Bru+21] propose a user-configurable transformation from LPG to RDF. So-called *contexts*, inspired by JSON-LD, serve to describe a) a "mapping between the terms used in the PG and IRIs" in RDF and b) "templates to represent the different properties and edges in the resulting RDF graph". An LPG is transformed in a two-step process. First a reified representation in RDF (actually using the RDF standard reification vocabulary) is generated which is rather unwieldy but complete. This step is lossless and reversible. In a second step a PREC-C context is applied, resulting in an idiomatic and intuitive representation of the LPG in RDF-star. While the result is certainly convincing, the PREC context seems too require quite specific instructions — an effort that might not be viable for large and heterogeneous LPGs. See also [CYM20] below for a similar approach in the opposite direction.

From RDF to LPG [MTS15] combine RDF and LPG to make use of their respective strengths. Initially an RDF knowledge graph is built to provide a complete and detailed representation of available data. From this graph key entities, relationships and events are extracted via SPARQL queries. The condensed version is made available through an attributed graph API that connects to popular LPG implementations, allowing to reap their benefits w.r.t. graph traversal and analytics.

[Tom16] propose a formalization of the LPG model and a serialisation format for RDF data, named Yet Another RDF Serialization (YARS), both with the aim to facilitate processing of RDF data in LPG databases. The LPG formalization concentrates on the rather uncontroversial core features of LPGs. The YARS serialisation is modelled after the syntax of the popular LPG query language Cypher, but adds support for namespacing IRIs — a rather indispensable feature when working with RDF data.

[BSH18] introduce a tool to populate Neo4j databases with RDF data. They use configurable SPARQL queries to identify central nodes and relations, extract their labels and other attributes, and map the results to the LPG data model.

[ATT20] distinguish three categories of interoperability: syntactic, semantic and query interoperability. Their work concentrates on semantic interoperability and investigates several mappings, allowing automated translation between the two at different levels of schema support. They find that any RDF database can be transformed into a LPG database and inverse mappings can restore the original database without loss of information, concluding that "the PG data model subsumes the information capacity of the RDF data model." An efficient example implementation to process large datasets is demonstrated and provided. Some features such as reification and RDFS-based inheritance relations have been excluded, but are planned to be studied in future work.

[CYM20] similarly to [MTS15] above try to combine RDF and LPG systems to benefit from their respective strengths: standardized data integration in RDF, better support for graph traversal and analytics in LPG. To that end they develop a mapping framework based on the Graph to Graph Mapping Language (G2GML), propose a standardized common model and serialisation for LPGs to improve interoperability on the LPG side, and present a prototype implementation. G2GML is a declarative language which maps from RDF patterns described in SPARQL to LPG patterns described in a syntax derived from openCypher.[79] An example provided shows mappings requiring rather detailed descriptions. The mappings are designed to be intuitive but may loose detail in some cases, e.g. smushing multi-edges with different properties in RDF into a single edge in the LPG representation.

[Abu+22] extend the scope of mappings on the RDF side from standard RDF to RDF-star. They identify two approaches: "RDF-topology Preserving Transformation (RPT) and Property Graph Transformation (PGT). RPT tries to preserve the RDF-star graph structure by transforming each RDF statement into an edge in the PG. PGT, on the other hand, ensures that datatype property statements are mapped to node properties in the PG." This properly frames the discussion of how to represent node annotations: as proper relations like in RDF or as mere attributes to nodes like in LPG (and in consequence also on how to represent annotations on such attributes in LPG). They find that non of the tested approaches support all test cases and identify different use cases and usage scenarios in which users might prefer PGT over RPT approaches or vice versa.

Mixed RDF/LPG Systems [Ngu+19], aim at a unifying approach to represent both LPG and RDF. Following their proposal from [Ngu+16] for a new graph model for RDF, they replace edges by nodes which may then be attributed and are connected to the nodes in the original statement by `in` and `out` relations. This approach doubles the triple count in RDF. It circumvents a performance problem with querying singleton properties that could however also be solved with one more join on the `:singletonPropertyOf` relation.

[Vrg+21] present MillenniumDB, a graph database engine based on a graph data model that provides"a simple abstraction upon which a variety of popular

[79] `https://www.opencypher.org/`

graph models can be supported." The model is defined as a relation `(source, type, target, id)`, akin to the Triple+ID pattern seen above. Abstract graph- and concrete data model, query semantics- and syntax, as well as storage, indexing, query planning and query evaluation techniques are discussed. An accompanying evaluation is based on Wikidata, which employs a data model that has similarities to both RDF and LPG.

The system described by [Las+21], already mentioned above, is based on a quad store as well, with an identifier appended to each triple. Annotations and even graph membership declarations are attributed to those identifiers. They discuss some difficulties in mapping RDF to LPG, from representational issues to fundamental differences in the underlying graph models. They also describe the challenges that customers encounter when confronted with the choice of either RDF or LPG. The choice, once made, cannot easily be reversed and can lead to unfortunate consequences later on with customers having to re-implement what the database already would have provided if they hadn't made the wrong choice.

[Tha+18] pursue another path towards RDF-LPG-interoperability: rather than mapping data structures they map query languages. They present *Gremlinator*, a tool that translates SPARQL queries to the popular LPG query language Gremlin. The query can then be run on most LPG databases. This can be useful for RDF practitioners that are proficient in SPARQL but would like to take of advantage LPG's efficient graph traversal implementations without having to learn a new query language.

Standardization A"W3C Workshop on Web Standardization for Graph Data - Creating Bridges: RDF, Property Graph and SQL"[80] in 2019 discussed possible ways to tackle some of these issues and to improve interoperability between the two technologies. Community efforts of vendors and users of LPG database systems are working on a common model and syntax to achieve both better interoperability and a more solid formal foundation.[81] Related work on standardizing a property graph query language, called GQL, is under way as well.[82] Efforts to implement LPG style statement annotations in RDF have lately gained momentum in a new W3C Working Group, see Section 4.2.4.5 above on RDF-star.

Graphs, defined as sets of triples, will be the topic of the following section. There is however a subcategory of graphs, singleton graphs, that could just as well belong into this section here. There is some confusion if a graph consisting of only one triple should be regarded as a graph, or if a a singleton graph should rather be understood as a triple, just represented in another way, like reified triples, RDF-star quoted triples, etc., all represent triples. While some works

[80] `https://www.w3.org/Data/events/data-ws-2019/index.html`

[81] PGSWG: Property Graph Schema Working Group, https://ldbcouncil.org/gql-community/pgswg/, also see a recent presentation at https://mosaicrown.github.io/scg2021/#mu-schedule, slides at https://mosaicrown.github.io/scg2021/slides/keynote.pdf, video recording at https://www.youtube.com/watch?v=efIRizFchdE&t=583s

[82] See `https://www.gqlstandards.org/` and `https://www.iso.org/standard/76120.html`

argue that there is a categorical difference between a set of triples and a single triple, others find that a singleton set is a set just as well. As a practical effect this allows to address both singleton and multiple statements with the same mechanism, which may lead to more predictable data structures. This will be discussed in more detail below in Section 4.2.5.5.

4.2.5 Graphs

Graphs are for various reasons considered an attractive toehold for attribution. Most obviously they provide effective means to annotate multiple statements at once, as it might occur when ingesting data from external sources, keeping versions separated or managing access rights. The challenge to handle data at web scale, and in a highly normalized format at that, makes any means to streamline and optimize processes very attractive. When the focus is not on fine-grained meta-modelling, qualification and contextualization even, but on moving around and putting to good use large quantities of data, then a grouping device like graphs — as RDF documents in a file system or as named graphs in a triple store[83] — is a very natural choice, especially compared to the very verbose RDF standard reification quad, but quite necessarily also to any other approach to statement attribution, even if it's much more succinct.

Graphs also provide means to put boundaries around sets of statements and provide some structure - something that RDF in general is quite bad at. One might expect additional means to connect, nest or otherwise put graphs in relation to each other. Those however are rare: some proposals provide support for nested graphs in one way or the other, Notation3 being the most consequent among them, but by and large support begins and ends with named graphs as implemented in SPARQL and RDF 1.1 datasets, queryable through `FROM` clauses strictly separate from the main query, without stable or even standardized semantics and not supported by reasoning in RDFS or OWL.

4.2.5.1 Named Graphs (Carroll et al. 2005)

[Car+05a] were the first to formally introduce the concept of named graphs. They argue that the "Semantic Web consists of many RDF graphs nameable by URIs" and their proposal "extends the syntax and semantics of RDF to cover such named graphs." The concept in principle, and the moniker, have become very successful, the semantics they proposed however was largely ignored.

[Sto+06] notes that the work of the W3C Named Graph Interest Group "has mainly been driven by the need for developing a trust model in RDF", whereas [Car+05a] name as intended purposes a far more elaborate set of scenarios, here quoted in full:

Data syndication systems need to keep track of provenance information, and provenance chains.

[83]In that case rather a quad store, but both monikers are used.

Restricting information usage Information providers might want to attach information about intellectual property rights or their privacy preferences to graphs in order to restrict the usage of published information

Access control A triple store may wish to allow fine-grained access control, which appears as metadata concerning the graphs in the store

Signing RDF graphs [. . .] it is necessary to keep the graph that has been signed distinct from the signature, and other metadata concerning the signing, which may be kept in a second graph.

Stating propositional attitudes such as modalities and beliefs

Scoping assertions and logic where logical relationships between graphs have to be captured

Ontology versioning and evolution OWL provides various properties to express metadata about ontologies. In OWL Full, these ontologies are RDF graphs. [. . .]

Both accounts can motivate the very specific semantics that [Car+05a] propose. [CS04] describes the "intended informal semantics" as follows: "the URI used for naming a graph is interpreted as the RDF graph specified within the <graph> element. Thus, statements about the URI are statements about the graph." [Car+05a] very specifically defines the relation between graph name and graph such that the name refers not to the graph itself but to the pair of name and graph. Therefore a graph of the same type, i.e. composed of the same set of triples, but with a different name, is a different named graph. This ensures that different sources of the same graph can be disambiguated by the name they give that graph. Hayes calls this a 'speech act semantics'. This definition avoids the ambiguity in graph naming that then ensued in practice because implementations chose their own semantics, using the graph name to denote the graph itself or the thing the graph is about or its date of ingestion or all graphs of that type (also when they have another name) and many other variations.[84]

[CS04] continues: "More strictly such a URI denotes an equivalence class of RDF graphs. RDF graph equivalence, as defined by RDF Concepts permits reordering of the triples, and relabelling of the blank nodes. This differs from merely extending RDF triples to RDF quads, in that the full extent of the graph is known, and is not treated with the open world assumption. Unlike a subject resource, which may have additional properties not mentioned in a document, the assertion of a named graph asserts that this graph is exactly the triples given, and there are not any others that have been omitted. Significantly, this intended semantics is a quoting mechanism and does not suffer the 'two-stage interpretation process' discussed for RDF reification in RDF Semantics. A naive extension of the RDF model theory to cover quads rather than triples would replicate this defect in the reification semantics."

In other words: named graphs per this definition are not referentially transparent, co-denotation — one of the corner stones of RDF semantics, enabling decentralisation through the use of different vocabularies to express the same

[84]See Section 2.7.3

meaning — is suppressed, and the integration-focused semantics of standard RDF graphs is suspended for the sake of applications that require a faithful record of the exact syntactic nature of a statement (or a set thereof). The 'two-stage interpretation process' mentioned, the process of interpretation that distinguishes between a string and what it means, is here described almost as a nuisance whereas it actually is a process at the very core of a logic: it establishes the difference between the words we use and the message we want to convey, and an integration focused platform such as the Semantic Web is well served with this differentiation — and its focus on the latter, the meaning, rather than the former, the vocabulary used. Nonetheless this purposeful wobbly-ness of RDF w.r.t. syntactic representations stands in the way of application requirements as those named above, and for those the proposed semantics makes perfect sense.

The elaborate set of use cases given by [Car+05a] as well as [Sto+06]'s account suggest that such applications presented an urgent need. 'Trust' certainly was an important part of the famous 'Semantic Web layer cake' in it's various incarnations. Also, [Car+05a] weren't late to the party: they named the concept and provided the semantics when the Semantic Web was still in its infancy and triple stores had just begun the add a fourth column. Yet the named graphs semantics as defined didn't succeed in practice, in fact it is probably very nïche on the real existant Semantic Web, and not even the RDF 1.1 WG was able to remedy the Babylonian variety of named graph semantics that developed within a few years, despite huge expectations and enormous effort.

4.2.5.2 RDF Next Steps Workshop and RDF 1.1 WG

As discussed in Section 2.7.3 named graphs were an important topic in the RDF 1.1 standardization effort and a few proposals have been brought forward in preparation for and during the Working Group. Hayes in an invited talk at ISWC 2009 proposed to use graphs to extend the expressiveness of RDF towards First Order Logic. [Zim12b] and [Zim12a] explore options w.r.t. contextualized graph-local semantics and even optional quoting semantics whereas [Hay12c] and [Hay12a] propose a more general contextualization extension to RDF.

[GC10] advocates as an extension to the RDF/XML syntax a property to state graph membership of a triple. This allows for nesting of graph containment expressions in syntax, but not semantically as there is no inheritance or containment relation between inner and outer graph membership expressions.

[Ata10] present an approach that combines statement annotation and statement contextualization by adding a statement identifier to a graph based approach, as discussed below in Section 4.2.7.

4.2.5.3 Nested Graphs

[Ber+07] present Notation3 (N3), a syntax for RDF of which Turtle is a subset. Syntactically Notation3 goes beyond Turtle by providing a graph nesting mechanism via curly brackets enclosing RDF statements. Those nested graphs are called formulas and can occur in subject and object positions of triples.[85] N3 formulas might seem like an ideal syntactic vehicle to implement attributions on statements and graphs alike. However, they come with a semantics that makes them quite unsuitable for this task: formulas are defined as being referentially opaque and unasserted. This semantics enables quoting of statements which "allows users to distinguish between what they believe to be true and what someone else including a website states or believes". See Section 9.3.3.3 for how this arrangement also enables implementing different semantics per graph.

Graph literals, discussed in 4.2.1.2 above, are very similar to N3 formulas and have been proposed as a way to implement the logic features of formulas in RDF without needing an extension to the RDF model and syntax. They also have a usability advantage: a literal, which in probably all RDF syntaxes is enclosed in quotation marks, is quite unmistakeably a quote, whereas a formula looks like any other piece of RDF and might be mistaken for actually being asserted for someone not familiar with its specific semantics arrangements.

[Che+08] reports of the implementation of nested graphs in the *Corese* RDF store and query engine. Graph membership of triples is expressed in RDF/XML by the syntax extension discussed in [GC10] above. Graphs nesting is expressed via statements to that effect, using an ontology to define relations like e.g. `then` or `unless`. An extension to SPARQL enables querying over nested contexts.

Other projects have employed similar mechanisms to implement graph nesting 'virtually' by relating named graphs explicitly via statements to that effect. The following section on Graph Composition presents such an approach. As far as this author is aware Notation3 is the only attempt at a more integral nesting mechanism.

4.2.5.4 Graph Composition

Nanopublications [GGV10] propose nanopublications as a model to publish and transmit scientific data together with its provenance and other contextual data. They argue that any statement — like the simple assertion that "Malaria is transmitted by mosquitoes" — can exist many times, in many places but it can only be validated scientifically if its context is known. That context is traditionally the scientific publication. However scientific data is increasingly generated, aggregated and exchanged outside of and in addition to traditional publications in which case new mechanisms are needed to provide them with context about the way they were derived, how their claims are backed etc.

Nanopublications are serialized as named graphs in RDF. One named graph contains the assertion or set of assertions in question. E.g. that graph would contain only the one statement about Malaria being transmitted by mosquitoes. The second named graph contains the metadata: annotations about provenance, supporting evidence etc of the assertion(s) in the first graph. All statements in the second graph have as subject the IRI of the first graph. In its simplest

[85] See 6 for an example.

form a Nanopublication consists of just those two named graphs but annotations may as well be grouped into separate graphs, graphs containing annotations can themselves be annotated in further graphs etc. Technically this proposal has some interesting properties:

– it fits nicely within the model-theoretic semantics of RDF as it honours the distinction between an abstract triple — a triple S P O, that could be stated anywhere, anytime, by anyone — and a concrete occurrence of a statement.
– it provides a definition of what constitutes a concrete occurrence of a statement: a named graph — a crucial fixing arrangement that RDF leaves unspecified.
– it doesn't bother about introducing a special syntax for the reification of singleton statements. Named graphs are defined as sets of statements and a set containing only one member is a set just the same.[86]

Nanopublications can't contain each other because named graphs can't be nested. Therefore the connection between assertion graph and metadata graph(s) depends on links. [KD14] introduce cryptographically sound graph identifiers, called 'Trusty URIs', to mitigate this problem through strengthening the identification of the graphs that make up a Nanopublication. These identifiers allow self-references in the data and consequently can identify groups of nanopublications referencing each other, including graphs annotating other graphs that annotate an assertion. As an added benefit they even work across different presentation formats. To make nanopublications more self contained [Kuh+16] introduce the concept of a head graph that collects references to an assertion graph, a provenance graph and a publication info graph, the latter providing meta information about the Nanopublication itself. [Kuh+18] provides an overview of further work addressing e.g. collections of nanopublications, decentralized publishing structures and versioning. It also discusses performance aspects of the elaborate use of named graphs even for single statements. Nanopublications have been developed in the life sciences domain but are employed also in other areas like the humanities where they can be useful to keep track of interpretational aspects and factual uncertainties [Kuh+16].

Research Objects [Bel+12] pursue a similar goal as nanopublications: capture and describe scientific workflows. To that end they define a concept called "workflow-centric research object", that "bundles a workflow, the provenance of the results obtained by its enactment, other digital objects that are relevant for the experiment (papers, datasets, etc.), and annotations that semantically describe all these objects." They do rely mostly on RDF triples and specialized vocabularies, but group the resulting set of statements in graphs that can be annotated to foster discovery and reuse.

[86]This issue is discussed in more depth in Section 4.2.5.5.

4.2.5.5 Singleton Graphs aka Named Triples

Graphs are mostly understood as a grouping device to demarcate some kind of boundary around a set of statements and economically attribute those statements all together. Some approaches however make it explicitly clear that they consider graphs containing only one statement not as a mistake or regrettable nuisance, but as a valid approach to solve certain use cases.

[Car+05a] refers to Named Triples as a special case of named graphs and considers them a viable alternative to RDF standard reification.

[GGV10] maps the nanopublications model to named graphs by assigning each statement — here defined as a uniquely identifiable triple — to a separate named graph.

[TPR12] annotate each triple with a trust metric. They present two different methods to map this construct to RDF: "reification of the statement and Named Graphs", where named graphs are actually singleton graphs.

[HHK15] when discussing the mapping of Wikidata annotated statements to RDF consider the use of named graphs to represent individual statements, the graph name serving as identifier for further attributions.

[Sor+15] go a similar route and "allow each individual RDF statement [...] or set of statements [...] expressing a ContextAssertion to be wrapped within its own graph. Essentially, the graph URI becomes an identifier for the ContextAssertion. The ContextAnnotations are then expressed as RDF statements which have the graph URI (the identifier) as the subject."

[Tae+16] targets a need to annotate data for change detection. The aim is to improve performance in public SPARQL endpoints by caching fragments of RDF data. Those fragments, called Triple Pattern Fragments (TPF) are then annotated with the date they last changed to prevent unnecessary queries for eventual updates. "A TPF interface gives a unique IRI to each fragment corresponding to a triple pattern, including patterns without variables, i.e., actual triples." Each fragment is interpreted as a graph and called *implicit graph*. The IRI referring to such an implicit graph can then be used to attribute the fragment with the date it was last added.[87] However, this annotation of implicit graphs is only available for singleton graphs.

[Das20b] discusses extending RDF to support Named Triples as a means to facilitate statement annotation. He does this however in combination with and additionally to the support of named graphs and will therefore be discussed below in Section 4.2.7.2 together with other quintuple based approaches.

Named Graphs ("are not suited") to Annotate Single Triples Despite these works it is a recurring theme in the Semantic Web community that the sole purpose of named graphs is to group triples. According to some authors using named graphs to annotate a single triple, while possible in principle, would constitute bad engineering in practice. [Div+09] collect annotations on

[87]See 4.2.4.2 above for how the IRI is constructed from the statement

single triples into complex objects as otherwise they would have to "change the modelling approach drastically, i.e. by assigning each literal statement to a named graph of its own, which is undesirable." [Idr+17] dismiss the idea of using a named graph per triple to encode correspondences because they see a negative effect on query complexity equivalent to reification quads and "an additional negative effect because triple store indices are typically optimized for fewer named graphs." [Hof+13] argues that "to represent time and space annotations by named graphs, every RDF statement would have to form its own named graph. This seems like a huge overkill, and is not in the spirit of the original intention behind named graphs." [NS17] claims that "although the named graph could be used as a triple identifier, it was not intended to be used for that purpose. Instead, the named graph is mainly used for representing a set of triples." [Fer+18] assumes that "the number of graphs is much less than the number of unique subjects and objects. An optimization for extreme corner cases is devoted to future work." The triple store GraphDB discusses in its documentation the drawback of using the fourth element in a quad as a triple identifier instead of a graph name: "A significant drawback is the overload of the named graph parameter with an identifier instead of the file or source that produced the triple. The updates based on the triple source become more complicated and cumbersome to maintain."[88]

Plenty more statements to the same effect can be found on Semantic Web related mailing lists and the assumption seems to drive the design of recent approaches to meta-modelling in RDF like singleton properties and RDF-star. But this view might be rather a fallacy than a fact and it should be met with a healthy dose of scepticism. The original Named Graph proposal [Car+05b] only makes the case for graphs as a grouping device but, as noted above, an extended version of that paper, [Car+05a], also discusses 'Named Triples'. The query language SPARQL, that standardized a version of named graphs before RDF did in version 1.1, clearly concentrates on the grouping of triples, following the demands of database administrators to keep apart aggregations from different sources and manage ever increasing numbers of statements.

On a formal level one could argue that a statement, or triple, and a graph are two distinct concepts in RDF. On the other hand an RDF graph is defined as a set of triples, and there is little difference between graph containing one, two, three or many more triples. A set, even if it contains only one statement, is a set all the same. Given the prevalence of n-ary relations it may well be the case that annotations most often concern not a single but a small group of triples, but this author didn't become aware of any empirical study that investigated if annotations mostly come as singletons or in groups.

However, an important consideration should be that it won't help usability if some annotations are tied to single triples, using a triple-based approach, while others are tied to graphs, using a different syntax. Querying such a mix of modelling styles is tedious at best.

[88]https://graphdb.ontotext.com/documentation/9.2/free/devhub/rdf-sparql-star.html

4.2.6 Quads

As discussed in Section 4.2.4.3 above, quads are in this work defined as the extension of triples with a fourth field that serves one specific purpose, e.g. recording the provenance of a statement, a credibility value, the date of ingestion etc. In practice however most quad approaches either store a graph name or a statement identifier in the fourth field, both with the intent to provide a hook for a more flexible annotation facility.

[MK03] discuss approaches to contextualisation like RDF standard reification — which they dismisses as an approach that has never been employed in any significant knowledge representation technology — and instead argue for quads, saying: “A common argument against quads goes ‘We have triples, you want quads, where is it going to stop? Quintuples? Sextuples?’ The answer is, quadruples is all you need (this is a well-educated guess).” They propose to use the fourth position as a ‘context identifier’.

[Flo+09] propose to capture provenance information in the fourth field. More specifically they suggest to record in the fourth field for each statement “the set of graph names that participated in the derivation of the RDF triple through a limited subset of the RDFS entailment rules.”[89] In practice however this would rather be realized as a statement identifier and further statements, with that id as subject, naming the set of graphs participating in the derivation — or of course as a specialized data structure, but as such would be outside the scope of this section.

[KK10] present a data model called stRDF to represent spatial and temporal metadata. They define a stRDF quad as “an sRDF triple (a,b,c) with a fourth component τ which is a temporal constraint.” They introduce a “notation (a, b, c, τ), where the temporal constraint τ defines the set of time points that the fact represented by the triple (a, b, c) is valid in the real world.”

4.2.7 Quintuples

Quintuples, or quins, can like quads be differentiated into multiple approaches that all annotate a triple as the basic fact, but in quite different ways:

SPO G id A quite popular approach is to add an identifier to a statement in a graph. It starts out from the named graph approach and adds an additional identifier to the quad. However, multiple occurrences of the same triple (as type) in one graph are not supported and would be realized via multiple graphs.

SPO id G This approach, while very similar to the `SPO G id` above, explicitly caters for multiple occurrences of the same triple type in one and the same graph. Note that this differentiation does not depend on where the approach puts the identifier. Even if the identifier identifies an SPOG quad it may still allow multiple quads of that type, casting it into the `SPO id G` basket by this categorization.

[89] [Ana+14]

SPO key value In this variant the triple is attributed with property value pairs. Attributes can be freely chosen and multiple attributes to the same triple are possible, but not to different triples of the same type.

SPO temporal spatial Some approaches encode the temporal and spatial validity of a fact. Here the respective values are usually encoded as complex datatypes.

SPO start-time endtime The most specific approach encodes a time interval to describe the temporal validity of the fact.

4.2.7.1 SPO G id

[Sch+08] in an approach called *RDF*$^{+}$ implement quintuples as pairs of an identifier and a quad, the latter composed of an RDF statement and the named graph it is contained in. Identifier and quad are both keys in a bijective function and the statement is understood as a literal, i.e. referentially opaque. Statements are identified by their type, therefore multiple different annotations on the same statement in the same graph are not possible.

Meta data statements are kept separate from data, and can be represented either as mere triples (as in the example below) or as quins to make them amenable to further processing like e.g. calculations of combined probabilities.

The exact form of the identifiers is an implementation detail, as they are only used for internal processing. Mapping the internal quintuple representation to RDF with named graphs requires a dimensional reduction. Therefore sets of meta knowledge property values are grouped into one complex value, losing detail:

```
Knowledge:={
    (G5, JamesHendler, researchTopic, SemanticWeb, #1),
    (G5, JamesHendler, affiliatedWith, UnivMaryland, #2)
}
Metaknowledge:={
    (#1, mk:source, {<www.rpi.edu/report.doc>}),
    (#2, mk:source, {<www.cs.umd.edu/survey.pdf>})
}
```

is mapped to

```
G5 { JamesHendler researchTopic SemanticWeb .
     JamesHendler affiliatedWith UnivMaryland }
G6 { G5 mk:source <www.rpi.edu/report.doc>, <www.cs.umd.edu/survey.pdf>.}
```

The approach also includes an extension to SPARQL that facilitates the retrieval of metadata together with data queried for and an example implementation. [Div+09] expand on this work with a formal evaluation of the SPARQL extension, a discussion of soundness and completeness and more.

[Ata10] propose to separate provenance management from contextualization in a so-called *tripleset* model. The atomic element is a quintuple `<S,P,O,G,TS1,...,TSn>`, composed of the triple with subject `S`, predicate `P` and object `O`, a graph `G` containing the triple and a list of triplesets `TS1,...,TSn` that'tag' the quad `SPOG`.

Graphs ‘own’ a triple and are the domain of addition, update and removal operations. Their purpose is to record provenance. Triplesets are understood as being orthogonal to graphs and serve to contextualize, ‘tag’ or otherwise annotate the triples per graph. Triplesets may span over multiple graphs, conceptually rather encompassing the whole dataset. Adding a tag to a quad is interpreted as an association operation “which does not add a new arc in the graph” and accordingly removing a tag is not understood to change the underlying graph either.

[And19] present an RDF storage solution optimized for versioning. It “retains previous store states, in addition to the current state, as active addressable aspects of a dataset analogous to named graphs in a quad store by annotating statements with temporal attributes and permits to address these states as static projections through provision of a REVISION argument in SPARQL and Graph Store requests.” The implementation is flexible enough to encode other aspects instead of previous state.

4.2.7.2 SPO id G

[Das21] proposes *RDFn*, a “fully backward-compatible extension of RDF that addresses RDF’s lack of adequate support for modeling multi-edge (i.e., multiple instances of the same relationship) and edge-as-endpoint (where a statement, i.e., a triple or a quad, is used as the subject or object of another statement) by requiring that every statement must have a unique name.” An important feature of this approach is that existing data and queries remain unchanged when the knowledge base is updated with new annotations or even multi-edge annotations.

RDFn automatically generates one identifier per statement per graph, as a blank node, to refer to statements. If more than one multi-edge annotation is to be added to a statement, more identifiers have to be defined manually, as IRIs, by the user. This arrangement tackles use cases that can’t be solved by approaches like RDF$^+$ and RDF-star without requiring further constructs, because they address a statement as a type instead of as an occurrence. RDF standard reification does in principle follow the same path of providing an identifier per occurrence, albeit in a very verbose way. RDFn provides a syntax extension to Turtle that allows to specify the name manually as `(|<name>|)`.

The following example about Soccer World Cup (SWC) championships illustrates the syntax: automatically generated identifiers are of type `spogb` (`b` for blank node), manually generated ones of type `spogi` (`i` for IRI). Because the user wants to differentiate the second win in 2002 from the first one, fact A2 is provided with a manually generated identifier, an IRI (and therefore of type `spogi`:)

```
Fact                         | RDFn statement (ext. Turtle)  | id type
-----------------------------+-------------------------------+------------
A1  Brazil won SWC           | :Brazil :won :SWC.            | spogb
A2  Brazil won SWC (2nd win) | :Brazil :won :SWC (|:Bwin2|). | spogi
A3  France won SWC           | :France :won :SWC.            | spogb
A4  Brazil 2nd win in 2002   | :Bwin2 :year 2002.            | spogb
```

[Das21] also considers the case that the first occurrence (fact A1) is to be annotated. In this case an alias (again a blank node) is generated to refer to fact A1's identifier:

```
Fact                         | RDFn statement (ext. Turtle)  | id type
-----------------------------+-------------------------------+------------
A1  Brazil won SWC           | :Brazil :won :SWC (|_:Bwin|). | spogb/alias
A2  Brazil won SWC (2nd win) | :Brazil :won :SWC (|:Bwin2|). | spogi
A3  France won SWC           | :France :won :SWC.            | spogb
A4  Brazil 2nd win in 2002   | :Bwin2 :year 2002.            | spogb
A5  1st win preceded 2nd win | _:Bwin :preceded :Bwin2.      | spogb
```

4.2.7.3 SPO start-time end-time

[Kri12] replaces the RDF triple by a quintuple, encoding the start- and end time of a 'temporalized' fact as separate arguments. This, he argues, makes querying and reasoning with temporally annotated information much easier. An example implementation is provided.

4.2.7.4 SPO key value

[HHK15] discuss the representation of Wikidata annotated relations as quintuples, arguing that "[c]onceptually, one could view Wikidata as a 'Property Graph': a directed labelled graph where edges themselves can have attributes and values." An (attributed) RDF triple would then be represented by the first three arguments, whereas the annotation property and value are translated into the fourth and fifth argument. "All quins with a common primary relation would constitute a statement." However, they dismiss this approach because it can't encode a fact with multiple multi-value annotations without mixing up attribute-value pairs that belong to different compound annotations. They argue that RDF* can be interpreted as a quintuple based approach as well, and fails their use case for the same reason. They also discuss to fix this issue by adding a statement identifier as a sixth argument to what then becomes a sextuple, but dismiss that approach as too involved for reasons specific to the Wikidata use case.

4.2.8 N-tuples

N-tuples are rather rare and usually not intended to provide an alternative to the RDF data model but rather as an optimization strategy in the backend, often supported by extensions to SPARQL to facilitate querying of regular annotation dimensions like temporal and spatial constraints. Representations fall back to the usual candidates: RDF standard reification, n-ary relations, named graphs etc.

[SKW07] explains that the data of the YAGO knowledge graph is kept in text files and those files are organized in folders per property. If converted to a database table "[t]he table has the simple schema FACTS(factId, arg1, relation, arg2, confidence).” An example serialisation in RDF/XML makes no notion of a confidence value, but provides a provenance annotation, using RDF/XML’s elegant syntactic sugar for the reification quad, the `rdf:ID` attribute:

```
<rdf:Description rdf:about="http://mpii.mpg.de/yago#Albert_Einstein">
    <yago:bornInYear rdf:ID="f1">1879</yago:bornInYear>
</rdf:Description>
<rdf:Description rdf:about="http://mpii.mpg.de/yago#f1">
    <yago:foundIn rdf:ID="f2" rdf:resource="http:..."/>
</rdf:Description>
```

[Hof+13] introduce the YAGO2 knowledge graph, a follow-up to YAGO and experimenting with a different storage scheme. Here facts are composed of six arguments: subject, predicate, object, time, space and conteXt — SPOTLX for short. This allows to enhance facts with spatio-temporal scope. The SPOTLX representation format “can co-exist with SPO triples, but provide a much more convenient way of browsing and querying the YAGO2 knowledge base.” The context argument is selected from a curated list of keywords and keyphrases. The authors argue that “[a]s no knowledge base can ever be complete, the conteXtual annotations further enhance the capabilities for querying and interactive exploration. The full YAGO2 interface provides SPOTLX tuples to this end.” As in the preceding YAGO project the RDF representation uses RDF standard reification to express attributes like provenance etc. If a fact occurs multiple times then YAGO2 will contain it multiple times as well, with different identifiers. This is well within the expressive power of RDF standard reification and, if used e.g. for provenance, also in accordance with its semantics.

An example — the Grateful Dead performing ‘The Closing of Winterland’ in San Francisco on New Year’s eve of 1978 — illustrates the annotation of a sextuple statement with further details:

```
Id  S   P            O          T          L          X
id1 GD  performed    TCOW       1978-12-31 -37.5,122.3 "Wall of Sound..."
id2 id1 occursIn     SF                               "Golden Gate c..."
id3 id1 occursOnDate 1978-12-31
```

4.3 Collection

A single triple alone is often not able to fully convey the desired message. Even worse, if it happens to contain a blank node it is of limited use on its own. Expressing a complex subject with several statements that are then grouped together is one way to achieve higher expressivity. Another application is more efficient query and retrieval services. Grouping statements however is not well supported in RDF. Documents and named graphs are often too coarse and inflexible to group statements into sensible units of information or are already used for other purposes.

Various proposals have been brought forward on how to define a certain group of statements or a subgraph algorithmically. If such a group is addressable by an IRI it can itself be referenced and nested in other groupings of statements or become the subject of further attribution.

4.3.1 Constructing Subgraphs From Seeds

Concise Bounded Description [Sti05] defines an algorithm that, given some node in some graph, extracts a subgraph “consisting of those statements which together constitute a focused body of knowledge about the resource.” The Concise Bounded Description (CBD) algorithm extracts into a subgraph all statements that contain the given node as subject, recursively follows blank nodes in object position to the statements they are subject of, and of all the statements collected so far, their reifications and attributes associated to those. It is claimed that the description returned by the algorithm is of “reasonably general and broadly optimal form” to satisfy the needs of many applications. Alternative variants of the algorithm include statements that have the start node as object of a symmetric relation — called Symmetric CBD — or exclude statements that have a blank node as subject in an inverse functional relation — called Inverse Functional CBD. In case the returned subgraphs become too big because of extensive use of blank nodes in the source graph, it is advised to limit either the (over)use of blank nodes, the path length to be followed or the number of returned statements or to exclude reifications.

Minimum Self-Contained Graph [Tum+05] implement a similar approach. Their Minimum Self-Contained Graph (MSG) algorithm however starts from a statement, not an individual node. It follows blank nodes irrespective of their position as subject or object and recursively collects statements involving those blank nodes until no new blank nodes are found. Like with the CBD algorithm the MSG of a statement that contains no blank nodes is the statement itself. They also prove some interesting properties of the algorithm: each statement is contained in exactly one MSG and the constructed MSGs are always the same no matter from which statement the process is started. These properties are useful for applications like signing subgraphs and creating patches for update/delete operations. Related to [Sti05] they define the notion of an RDF Neighborhood (RDFN) which is composed of all MSGs that involve a certain node, equivalent to the start node of CBDs.

Named Graphs [Car+05a] compare MSGs and CBDs to named graphs. They find that MSGs tend to be too small to be useful, often comprising only one statement in case the graph uses blank nodes sparsely. CBDs however in their opinion “divide a knowledge base up into resource-centric subgraphs; whereas named graphs divide a knowledge base up in a publication oriented way. The two approaches are complementary and could be used together.”

4.3.2 Partitioning Graphs Into Subgraphs

Molecules [Din+05] present an algorithm to reversibly decompose an RDF graph into its minimal sub-graphs, called molecules. These molecules are either triples without blank nodes, or triples or sets of triples that 'ground' blank nodes through functional or inverse functional properties. The decomposition is lossless and reversible. An experimental implementation uses quads to keep track of the source from which each molecule was derived. The proposal notes as a disadvantage that the set of molecules derivable from a graph can become much larger than the graph itself depending on the number of ways in which blank nodes may be grounded.

Path Outlines [DCG20] propose an approach to generate path-based summaries from collections of named graphs. The goal is to provide "meaningful overviews at the right level of abstraction." Paths are formalized in terms of the concept of path outlines which are "first-class citizens that can be described, searched, browsed, and inspected." "A path outline is a conceptual object providing descriptive statistics about a sequence of statements relative to a set of entities. It consists of: 1) a set of entities sharing a similarity criteria (e.g. all the entities of class Person), for which at least one entity is the subject of a given sequence of properties, 2) the sequence of properties, 3) the set of objects at the end of this sequence, and 4) the set of measures relative to the entities and the object." Compared to the usually employed and basic-graph-pattern based follow-your-nose algorithm the path based approach may provide advantages for deeply nested and wide spanning structures.

JSON-LD Frames The JSON-LD Framing API[90], a proposal developed by the JSON-LD Working Group[91] allows to 'bundle' statements and thereby control their serialisation in JSON-LD, enabling "a developer to specify exactly how they would like data to be framed, such that statements about a particular subject are bundled together, delimited via { and }, such that the subjects they relate to 'nest' into a particular tree structure that matches what their application expects."

4.3.3 Triple Pattern Fragments

A related technique is Triple Pattern Fragments (TPF), designed to improve the performance and scalability of SPARQL endpoints by transforming complex queries into requests that servers can handle more easily and result sets that transfer load to the network and the client side. According to [Ver+14a] "triple pattern fragments, which additionally contain count metadata and hypermedia controls, can reduce the load on servers to less than 30% of the load on SPARQL endpoints. This happens by shifting the query-specific tasks to clients, at the

[90]`https://w3c.github.io/json-ld-framing/`
[91]`https://www.w3.org/2018/json-ld-wg/`

cost of slower query execution. Instead of sending one complex query, clients use a dynamic iterator pipeline to combine the results of several simpler queries, thereby also vastly improving the effectiveness of http caching."[92]

[Ver+14b] and [Ver+16] , outline the basic design principles of TPFs, [Tae15] introduces triple fragments for quads with identifiers, while [ATP20] and [Azz+21] introduce recent improvements and refinements. The concept is not immediately concerned with questions of modelling but if it proves successful such fragments might become resources in their own right.

4.4 Comparisons

Quite a few comparisons have been brought forward to evaluate the different approaches towards representation of complex knowledge in RDF, and with quite different foci. Often the main topics of interest are aspects of performance, triple count and such. Other works concentrate on usability, amenability to reasoning, expressivity etc. The following survey provides a cursory overview.

Comparisons often take existing approaches and stick to them as described by the authors. Such an approach can claim faithfulness and helps when comparing comparisons, but sometimes it leaves even obvious chances for optimizations unexploited. For example, although RDF standard reification in practice is most often implemented via three statements, not four, as the declaration of type can safely be omitted, comparisons regularly count four triples when comparing triple counts of different representations. [Fre+19b] is one of the laudable exceptions to this rule.

[Hay04a] discusses formalisms for spatio-temporal reasoning and compares the expressivity of annotations to statements, edges and vertices respectively.

[Gan11] and [GP13] compare OWL-compatible meta-modelling approaches w.r.t. several dimensions, including triple count and supported entailments, but also rather rare criteria like usability, idiomatic characteristics, pithiness, gestaltic criteria etc. [Sch+13] expand on that work with an extremely rare occurrence in the field of Semantic Web research: an empirical user study on usability.

[Hof+13] provide an extensive overview on works targeting the representation of spatial and temporal data and metadata.

[TKK13] discuss the representation of temporal aspects via RDF standard reification, named graphs, n-ary relations and several extensions to RDF(S).

[Fu+15] compare query performance when representing provenance annotations with n-ary relations, named graphs (as nanopublications) and singleton properties.

[HHK15] investigate the representation of Wikidata attributed graphs with RDF standard reification, named graphs, n-ary relations and singleton properties,

[92]See also an introductory video at `http://videolectures.net/iswc2014_verborgh_querying_datasets/`

comparing expressivity, ease of modelling and query performance. [Fre+19b] expand on that work, striving for a comparable test environment and extending the set of representational approaches by an optimized version of singleton properties, called Companion Property, and RDF*.

[Idr+17] use singleton properties to annotate `owl:sameAs`-correspondences but support conversion to n-ary relations and discuss also RDF standard reification, named graphs and RDF* with the help of very detailed examples.

[RMH17] compare the expressivity and usability of basic triples (as a "binary n-ary relation"), Davidsonian and neo-Davidsonian n-ary relations, singleton properties and name chaining approaches.

[GZM17] compare RDF standard reification, singleton properties, n-ary relations and NdFluents w.r.t. entailment preservation in contextualized reasoning. [ZG17] add their NdTerms approach, [GZ18] put in NdProperties and the Companion Property approach, that [Fre+19b] introduced.

[Tae+16] compare query performance of singleton properties, named graphs and implicit graphs (as Triple Pattern Fragments).

[Hog+20] analyse the expressivity of Labelled Property Graphs compared to RDF standard reification, n-ary relations, singleton properties and other modelling approaches.

[SP20] compare a very large set of approaches, among them Notation3, RDF Molecules, Annotated RDF, RDF$^+$, RDF*, PaCE, N-Quads along a quite unconventional set of criteria like the availability of formal semantics, tuple type, data model extension, compliance with standard RDF serialisations etc.

[Sen+20] provide an equally broad overview of representational approaches to meta-modelling.

[Las+] compare the OneGraph model of statements and statement-identifiers with RDF, RDF-star and Property Graphs.

[Das21] compares the RDFn quintuple approach with RDF, RDF-star and LPG, especially focusing on resilience w.r.t. updates to the data.

Chapter 5

Identification

I am reminded of the joke about comedians sitting in a room and just saying numbers and then laughing - (each number is a different joke that they all know).
So Identity Problem number 3? - "The world is infinitely complex, so you won't be able to accurately create a URI for anything."
Identity Problem number 5? - "The world changes, so none of your URIs will be stable over time."
Identity Problem number 4? - "Things have multiple identifiers - you won't know which to choose."
Identity Problem number 2? - "Yeah, but things don't look the same in other cultures."
Then there are others which crop up occasionally, but aren't actually problems, such as "But people will be able to make incorrect statements about my URIs".
These problems are not unique to SemWeb - they are common to any development anyone does.

Hugh Glaser[1]

Subject identification is already a can of worms, with questions that will never get definitive answers. A classic example is the Ship of Theseus: if all parts of a ship are replaced, one by one, over a prolonged period of time, is it in the end still the same ship or a different one? Another one: where exactly are the boundaries of Mount Everest? Or: is a person at the age of 5 and as an adult the same person? The best possible answer is most often that 'it depends', but that's not necessarily a helpful answer when the aim is to construct a semantic network of data spanning the whole planet, and dependable applications based on it.

The complications introduced by the Semantic Web machinery only add to the problem. The decentralized nature of the Semantic Web causes the most

[1]In a post to the Semantic Web mailing list, `https://lists.w3.org/Archives/Public/semantic-web/2018Dec/0073.html`

obvious set of problems: without a central naming authority it is inevitable that different datasets will refer to the same thing by different IRIs. Being able to express that two names refer to the same entity is essential to interlinking and interoperation of data from different sources, and data publishers are encouraged to do so.

But the question when exactly two names refer to the same thing gets trickier the closer one looks, as the introducing examples should illustrate. The ‘unique identification’ approach[2] would like identifiers to just work instead of the whole project risking getting bogged down in negotiations, check-up’s and unresolvable uncertainties. It therefore postulates some rather radical assumptions about the nature of identifiers on the Semantic Web. Luckily, for most applications the shared understanding of what an identifier identifies is available readily enough to fulfil those assumptions and proceed without delay. However, the question is if it’s good enough that ‘unique identification’ works *most* of the time, but sometimes fails without prior notice. And what to do about those failures: how to handle them if they are unavoidable? More problems arise with the aim of the Semantic Web to speak about anything whatsoever, not just web pages and other resources accessible over the network, as Section 2.1 on Identification discussed in some depth. The question of how to disambiguate access from reference so far hasn’t been answered to everybody’s satisfaction.

[Raa+19] survey numerous works that analyse and aim to solve the problem of identification on the Semantic Web and there’s a lot of overlap between that survey and this section. However, approaches to (semi-) automatic equality relation verification, entity matching and reconciliation solutions are outside the scope of this work.

5.1 Identity Services

[BSG+07] propose a system called OKKAM to provide a “naming service for entities”. They discriminate between *things* “in a very broad sense, ranging from electronic documents to bound books, from people to cars, from conferences to unicorns” and *logical resources*, “abstract objects (like predicates, relations, assertions)”, the latter however being out of scope of their effort. The distinction between access and reference would therefore be lost to their system. As each entity is complemented by a description but explicitly no further categorization, one would as before have to resort to out-of-band means to distinguish an entity referring to the city of Paris from an entity referring to a website about Paris.

Various other efforts try to improve interoperability and inter-linking between publicly available RDF datasets, often under the moniker of Linked Open Data, by scanning them for equivalence relations, computing the closure over all those equivalence relations and collecting them in one place. These services allow users to look up IRIs equivalent to a given resource identifier, thus either re-using established identifiers or improving recall in queries.

[2]See Section 2.1.2

[GJM09] present the Co-reference Resolution Service (CRS)[3]. Because of the problems of the strict semantics of `owl:sameAs`, discussed in the following section, they design CRSs as context-dependant services that applications can choose from as appropriate. The CRS approach is designed to support identity disambiguation via hash IRIs and HTTP status codes, as discussed above in Section 2.1, just as well as vocabulary based approaches, discussed below in Section 7.2. They also support other types of equivalence relations besides `owl:sameAs` that employ less strict semantics such as umbel:isLike and skos:closeMatch.[4]

[MT18] introduce the LODsyndesis service[5] which integrates hundreds of datasets from the Linked Open Data (LOD) cloud and adds features like discovery of most connected datasets, collecting entity features from different datasets etc. It provides the transitive closure of equivalence relationships between entities and schemas over all the datasets included by exploiting predefined properties such as `owl:sameAs`, owl:equivalentProperty and owl:equivalentClass.

[Bee+18] present an identity service named sameAs.cc[6] that exclusively uses `owl:sameAs` statements to establish equivalence between resources, thus ensuring full adherence to the OWL semantics.

Among the three aforementioned projects, sameAs.cc provides the largest collection of `owl:sameAs` statements with their equivalence closure.

5.2 owl:sameAs

The property `owl:sameAs` was introduced in the Web Ontology Language (OWL)[7] in 2004 for the purpose of interlinking datasets and reconciling different names referring to the same entity.[8] RDF(S) provides a slightly related property, `rdfs:seeAlso`[9], that is sometimes used when `owl:sameAs` seems too strong, but it is rather informally defined and with a very loose interpretation of relatedness at that.

As the preceding section already hinted at, `owl:sameAs` is indeed the most prominent identity relation on the Semantic Web. It is also the only equivalence relation, or, more specifically, coreference relation, with well-defined semantics contained in the central ontologies. The purpose of `owl:sameAs` according to the OWL specification is to state that "two URI references actually refer to

[3]Hosted at `http://sameas.org`

[4]`http://sameas.org/about.php` lists `<http://www.rkbexplorer.com/ontologies/coref#coreferenceData>`, `<http://umbel.org/umbel/sc/isLike>`, `<http://www.w3.org/2004/02/skos/core#exactMatch>`, `<http://www.w3.org/2004/02/skos/core#closeMatch>`, `<http://open.vocab.org/terms/similarTo>` and `<http://www.geneontology.org/formats/oboInOwl#hasExactSynonym>`

[5]Hosted at `http://www.ics.forth.gr/isl/LODsyndesis`

[6]Hosted at `https://www.sameas.cc/`

[7]`https://www.w3.org/TR/owl-overview/`

[8]The original OWL specification from 2004 elaborates: "The `owl:sameAs` statements are often used in defining mappings between ontologies. It is unrealistic to assume everybody will use the same name to refer to individuals. That would require some grand design, which is contrary to the spirit of the web."

[9]`https://www.w3.org/TR/2014/REC-rdf-schema-20140225/#ch_seealso`

the same thing: the individuals have the same 'identity"'[10], that "two names refer to (denote) the same individual"[11]. The definition of `owl:sameAs` is based on 'Leibniz's law'[12] which states that identical entities share an identical set of properties — the so-called 'Indiscernibility of Identicals' — and, conversely, entities with identical properties are themselves identical — the so-called *Identity of Indiscernibles*. This identity relation is necessarily reflexive, symmetric and transitive. Therefore it enables a good deal of inferences.

In theory this definition is crisp and straightforward, in practice however it is surprisingly unwieldy. In fact it is actually often too powerful to be useful in its pure interpretation. [Mel13] argues that "certainly not every application requires that sameAs links be strict in the Leibnizian sense. However, transferring properties across equivalent URIs can only safely be done if such a strict form of identity is guaranteed." However, just that the guarantee is given in theory doesn't mean that it is observed in practice. [Hal+10] provides some anecdotal evidence of problematic occurrences of `owl:sameAs` involving the city of Paris, noting that "dbpedia:Paris is asserted to be sameAs both `cyc:CityOfParisFrance` and `cyc:ParisDepartmentFrance` (and five other URIs). Yet OpenCyc explicitly states (in English!) that these two are distinct." Indeed a user looking for an IRI to tag a blog post about her latest trip to Paris will not care and happily use the DBpedia IRI. However, an analyst checking economical statistics about average housing costs in Paris published in the Linked Open Data cloud will have to wonder what definition of dbpedia:Paris the author of such data had in mind.

IRIs as names are inevitably minted with some application in mind and new applications that, as encouraged by Semantic Web best practices, reuse IRIs might embody a different interpretation of what the name means, what exactly it refers to — even if differs just so slightly. As [Hal+10] note these deviations in interpretation and intuition "almost always violate the rather strict logical semantics of identity demanded by `owl:sameAs` as officially defined."

If this constitutes misuse or is just unavoidable is in the eye of the beholder, but while a few alternatives to `owl:sameAs` have been proposed, discussed below in Section 7.2, most of them come with little or no formal semantics. It is indeed hard to imagine a less strict but equally tight formal definition of equivalence that would support automated inferences the way `owl:sameAs` does.

5.2.1 Contextualization

One way to control the power of `owl:sameAs` is to contextualize its usage. As [Raa+19] note, contextualization is an unsolved problem on the Semantic Web and "[a]t the moment, the way of encoding contexts on the Web is largely ad hoc, as contexts are often embedded in application programs, or implied by

[10]`https://www.w3.org/TR/owl-ref/#sameAs-def`
[11]`https://www.w3.org/TR/2012/REC-owl2-primer-20121211/#a_SameIndividual/`
[12]By the 'father of Knowledge Representation', Gottfried Wilhelm Leibniz

community agreement.” The following works have developed more principled approaches to contextualize the use of `owl:sameAs`.

[Raa+19] portray the development of representing contextual identity from [Bat+14], [BSH16] and [RPS17] to [Idr+17] as one of increasingly tighter definitions of what constitutes a context, and then also distinguishing between context definition and propagation instructions.

[Bat+14] propose a mechanism to define equivalence criteria under which same-ness of nodes in different datasets can be established. This enables defining equivalence according to operational aspects and use cases as is often done in practice, albeit in disregard of the strict semantics of `owl:sameAs`. Usability and reuse of vocabularies is improved as the semantics of otherwise too strong or too weak generic properties like `owl:sameAs` and `rdfs:seeAlso` can be refined and adapted to the task at hand without the necessity to introduce new, non-standard properties. Sets of equivalence relations can be integrated by following a scale where an `owl:sameAs` relation is considered the strongest equivalence possible whereas weaker relations are `skos:exactMatch`, `skos:closeMatch` and `rdfs:seeAlso`.

[BSH16] argue that because of the fundamental characteristics of the identity problem — they list modality, context-dependence, time-dependence and counterfactual conditions — “a real-world semantics of identity should be context-dependent and non-monotonic.” Instead of checking for all properties of two entities they “parametrise the identity relation over the set of properties that are taken into account for establishing identity”, allowing them to “define different identity relations in different contexts” ordered in a lattice of identity contexts. However those identity contexts are not represented in RDF and properties not belonging to them are not considered any further.

[RPS17] propose an algorithm to compute contexts and encode them in RDF. Such a context is composed of subsets of classes and properties from the source graph and can e.g. express that the equality of medications should be derived from certain ingredients but not from others and neither from their brand name(s). Similar to [BSH16] they further on ignore properties that are not part of a context definition.

[Idr+17] build on a proposal by [Bre+12] and improve on the aforementioned works by propagating also properties that do not take part in establishing contextualized equality. A detailed discussion of representational aspects — informing the choice of singleton properties over reification, named graphs and RDF-star — is provided. The proposal provides a rich meta-model to help users choose the right correspondences, a declarative representation and complex operators that allow the reuse of existing correspondences in new combinations, a detailed provenance trail on combined correspondences, on-demand integration of automatically generated correspondences and more.

[HHT15] argue that “decentralized knowledge representation systems such as the Semantic Web will require a theory of identity for the Web beyond current use of sameAs links between various data-sets, including the fact that entities can not be linked across semantic roles" (e.g. referentially opaque occurrences of the same entity). They thereby aim well beyond other approaches and to that end propose to embed identification in a full-blown contextualization mechanism, characterized as "a modest extension to RDF." Their proposed semantics is based on the concept of "pollarding", which understands occurrences of some IRI term in RDF as extensions of that term, similar to how the RDF semantics "distinguishes properties and classes considered as objects from their extensions - the sets of object-value pairs which satisfy the property, or things that are 'in' the class."[13] This allows the formally correct implementation of equality across different semantic roles, capturing e.g. the distinction between Frege's 'sense' and 'reference', discussed above in Section 2.1.2. However, the whole contextualization mechanism is only presented rather sketchily.[14]

5.2.2 Usage

The identity services presented above have been built not only as a service to users but also as platforms for analysis of `owl:sameAs` usage on the Web of Data.

5.2.2.1 Correctness

[Hal+10] provide some anecdotal evidence about misuses of `owl:sameAs` in practice. In a small empirical experiment they generate a random sample of 250 `owl:sameAs` statements from the LOD cloud and ask three judges to tell if the entities so related are 'identical', 'matching', 'similar', 'related', 'not related' or 'can't tell'. It turns out that while the judges agree that about 50% of entities related by `owl:sameAs` links are indeed 'identical', a considerable percentage of relations are considered rather just related in some other form. Interestingly, some judges rather prefer to choose 'can't tell' than some less strict form of relatedness than 'identical', hinting at insecurity about how aspects of context should be assessed.

[Hog+12] come to a much more optimistic result w.r.t. the semantic correctness of `owl:sameAs` relations 'in the wild', establishing through manual inspection a precision of 97.2%, whereas [Hal+10] would arrive at 72% maximum (depending on how lesser forms of similarity are counted).

[Raa+18] perform experiments on 28 billion statements, representing a large subset of the 2015 LOD cloud, from which they extract an identity network of about 331 million edges. Algorithmic testing suggests that about 75% of `owl:sameAs` statements adhere to the strict semantics defined by OWL while most of the rest were considered only 'related'. A sizeable portion of `owl:sameAs` links (about 1.2 million) in automatic testing raised flags as being incorrect. Manual inspection of a small subset of tested links revealed that positive testing did indeed proof correct in 100% of cases and also most of those considered

[13] [Hay04b], `https://www.w3.org/TR/rdf-mt/#technote`

[14] Referring to `http://www.slideshare.net/PatHayes/rdf-with-contexts` and `https://www.slideshare.net/baojie_iowa/2010-0624-rdfcontext` for further inspiration

incorrect by the algorithm could be confirmed to be so — often coming from large equality sets. Of the remaining 'related' `owl:sameAs` relations between 50 and 85% were judged as true identity links.

5.2.2.2 Extent

[Din+10] are the first to analyse a statistically significant part of the Web of Data for usage of `owl:sameAs`. They find that most `owl:sameAs` relations form star-like structures with one central node connected to some peripheral resources — mostly only two connected nodes, at average 2.4, only a few connected components with hundreds of nodes — and that a few central publishers like e.g. DBpedia are used in a lot of `owl:sameAs` relations.

[Bee+18] analyse a graph of over 500 million `owl:sameAs` relations, constructed from a 2015 crawl of the Semantic Web. In contrast to the earlier work by [Din+10] they find more components composed of more than two connected nodes, hinting at an increase in richer equality expressions. According to their analysis `owl:sameAs` relations are mainly used to express equivalence across namespaces, not within the same domain. Whereas [Din+10] found DBpedia to be the predominant hub in the center of the equivalence web of `owl:sameAs` links, in the 2015 crawl by [Bee+18] more hubs on more specific topics have emerged, like e.g. "geonames.org, for interlinking geographic datasets; bio2rdf.org, for interlinking biochemistry datasets."

[SBP14] analyse linkage relationships between datasets in the LOD cloud, finding that across topical domains `owl:sameAs` is the predominant relation overall (not specifically focusing on equivalence relations), with `rdfs:seeAlso` as a distant second.

Chapter 6

Serialisation

But the take-away from your note about blank nodes: use more turtle, and think about it as the turtle language more than the underlying triples.

Tim Berners-Lee[1]

RDF serialisation formats don't get much attention in scientific publications although they have an immense effect on usability — and in effect also on expressivity — of the Semantic Web. [TH20], however, do present a nice rundown.

RDF came first with RDF/XML as its standard serialisation since XML was at the time considered *the* most important, indispensable even, serialisation format for all things web. RDF/XML however was quite verbose and with its specific style, called 'striping', neither idiomatic XML nor easy to get the hang of.

Another syntax came into being when developers exchanged short snippets of RDF over IRC that eventually became formalized as Notation3 (N3). Turtle was proposed as a subset of N3, omitting the notion of nested graphs and some advanced logic features. Turtle quickly became the common syntax for authoring and reading RDF by hand and also became the basis of the syntax of the RDF query language SPARQL.

There exist more syntaxes like e.g. N-triples which is optimized for exchange of RDF data between applications, and variants of Turtle and N-Triples that support named graphs.

More recently JSON-LD, which caters to the needs and customs of web developers, became more visible. As JSON-LD is not a fresh start on an RDF serialisation but derived from JSON, it provides more syntactic freedom and easily incorporates non-RDF based data structures found in JSON itself.

As the introducing quote suggests, clever serialisations can help shield the user from some of the intricacies of RDF like its heavy reliance of blank nodes.

[1]In a post to the Semantic Web mailing list, `https://lists.w3.org/Archives/Public/semantic-web/2018Nov/0053.html`

The significance of serialisations to the topic of this work can be illustrated by some examples:

RDF/XML provides an id attribute to reify a statement that is syntactically much more convenient than the RDF reification quad. Some applications still use RDF/XML only because of this feature.[2]

Notation3 provides comparable and even more powerful support for meta-modelling with a bracketing syntax that allows to group statements into graphs, there called formulas, and even supporting recursive nesting of graphs within graphs.

RDF-star as described above in Section 4.2.4.5 is a recent effort to introduce better meta-modelling support in RDF, extending the RDF model with a new primitive, the 'quoted triple'. It gained quite some popularity because of the perceived elegance of that syntax in Turtle and SPARQL.

JSON-LD is an RDF serialisation based on JSON, a serialisation especially catered to and popular in web development. It targets Linked Data deployments on the web. [AP18] discuss how to succinctly represent RDF reification in JSON-LD, opting for an n-ary approach and exploiting the fact that JSON-LD syntax, like Turtle, does hide blank nodes rather well.

The following sections will not introduce those serialisations completely, but concentrate on such syntactic aspects of meta-modelling. Other serialisations like Turtle, N-Triples, Trig, N-Quads or RDFa will not be not discussed at all, because they provide no syntactic sugar for meta-modelling. However, it should be mentioned that the Lisp-inspired RDF collections vocabulary,[3] which is a real nuisance when written out in full, becomes a nice and concise appearance in Turtle with its rounded bracket syntax as syntactic sugar.

Sometimes the argument is made that working with triples is akin to coding in Assembler and a surface syntax that more successfully hides its conceptual complexities would solve most of the problems of RDF in practice — see for example the post "Toward easier RDF: a proposal", `https://lists.w3.org/Archives/Public/semantic-web/2018Nov/0036.html`, on the Semantic Web mailing list and the ensuing mega thread. The example of the ID attribute in RDF/XML discussed below certainly supports that view. Nonetheless the development of such a judicious syntax requires a profound understanding of the problem space too. Another problem with this vision is that the Semantic Web machinery does not only encompass authoring, digesting and exchanging data — the most obvious applications of a serialisation — but also querying and reasoning services, which sometimes have different requirements and different perspectives on which intricacies should be hidden from the user and what needs to be exposed explicitly.

[2]As reported in an RDF-star use case at `https://w3c.github.io/rdf-star/UCR/rdf-star-ucr.html#uniprot-attributed-evidenced-triples`

[3]`rdf:List`, `rdf:first`, `rdf:next` etc., see Section 2.4.2

6.1 Syntactic Sugar for the Reification Quad

RDF/XML, the prevalent syntax at the time the RDF reification vocabulary was defined, provides some syntactic sugar in the form of an ID attribute. That ID, when added to a statement, represents its reification, in exchange for the verbose quad of explicit subject, predicate, object and type declaration. The ID can then be annotated in further statements. Multiple reifications of the same triple require to state the triple multiple times in the source code and each time attach an ID. This properly reflects the semantics of RDF reification as referring to occurrences, not the type of statement. Of course an additional ability to attach multiple identifiers on the same syntactic triple e.g. as a list of values to the ID attribute would fix even that pain point. The following example shows the same triple stated two times, with different annotations attached — first in RDF/XML, then in Turtle:

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:dc="http://purl.org/dc/elements/1.1/"
        xmlns:ex="http://example.org/"
        xml:base="http://example.org/id#">
    <rdf:Description
            rdf:about="http://www.w3.org/TR/rdf-syntax-grammar"
            dc:title="RDF1.1 XML Syntax">
        <ex:editor>
            <rdf:Description ex:fullName="Dave Beckett">
                <ex:homePage
                        rdf:resource="http://purl.org/net/dajobe/"
                        rdf:ID="one" />
                <ex:homePage
                        rdf:resource="http://purl.org/net/dajobe/"
                        rdf:ID="two" />
            </rdf:Description>
        </ex:editor>
    </rdf:Description>
    <rdf:Description rdf:about="#one">
        <ex:accordingTo rdf:resource='http://example.org/source1'/>
    </rdf:Description>
    <rdf:Description rdf:about="#two">
        <ex:urlDate>2004-4-16</ex:urlDate>
    </rdf:Description>
</rdf:RDF>
```

The Turtle version is about the same size, but it would be much shorter if it didn't have to write the reification quads out in full:

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix dc11: <http://purl.org/dc/elements/1.1/> .
@prefix ex: <http://example.org/> .

<http://www.w3.org/TR/rdf-syntax-grammar>
    dc11:title "RDF1.1 XML Syntax" ;
    ex:editor _:genid1 .
_:genid1
    ex:fullName "Dave Beckett" ;
    ex:homePage <http://purl.org/net/dajobe/> .
```

```
<http://example.org/id#one>
    a rdf:Statement ;
    rdf:subject _:genid1 ;
    rdf:predicate ex:homePage ;
    rdf:object <http://purl.org/net/dajobe/> ;
    ex:accordingTo <http://example.org/source1> .
<http://example.org/id#two>
    a rdf:Statement ;
    rdf:subject _:genid1 ;
    rdf:predicate ex:homePage ;
    rdf:object <http://purl.org/net/dajobe/> ;
    ex:urlDate "2004-4-16" .
```

RDF/XML nowadays is itself eschewed for its verbosity and tediousness. No other popular syntax took up the idea of providing a statement identifier. Also the ID attribute is only available in the local context of an RDF/XML document. As [Eck13] puts it, "RDF-XML provides optional statement identifiers as a shortcut for reification, but they are usually not created for all statements in a document, not portable to other serializations and not part of the data model".

Another example of syntactic sugar to make RDF standard reification more tolerable is the following Turtle-bases approach (omitting the `rdf:Statement` type declaration because it is of little practical relevance and can be inferred anyway):

```
@prefix s: <http://www.w3.org/1999/02/22-rdf-syntax-ns#subject> .
@prefix p: <http://www.w3.org/1999/02/22-rdf-syntax-ns#predicate> .
@prefix o: <http://www.w3.org/1999/02/22-rdf-syntax-ns#object> .
@prefix : <http://example.com/> .

[ s: :employee_38; p: :jobTitle; o: "Assistant Designer" ]
    :accordingTo :employee_22 .
```

Once the namespace declarations are out of the way this syntactic trick does indeed lead to a quite civilized look.[4] And applying this approach to the example above, with multiple reifications of the same triple, leads to the following code:

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix dc11: <http://purl.org/dc/elements/1.1/> .
@prefix ex: <http://example.org/> .
@prefix s: <http://www.w3.org/1999/02/22-rdf-syntax-ns#subject> .
@prefix p: <http://www.w3.org/1999/02/22-rdf-syntax-ns#predicate> .
@prefix o: <http://www.w3.org/1999/02/22-rdf-syntax-ns#object> .

<http://www.w3.org/TR/rdf-syntax-grammar>
    dc11:title "RDF1.1 XML Syntax" ;
    ex:editor _:genid1 .
_:genid1
    ex:fullName "Dave Beckett" ;
    ex:homePage <http://purl.org/net/dajobe/> .
[ s: _:genid1; p: ex:homePage; o: <http://purl.org/net/dajobe/> ]
    ex:accordingTo <http://example.org/source1> .
[ s: _:genid1; p: ex:homePage; o: <http://purl.org/net/dajobe/> ]
    ex:urlDate "2004-4-16" .
```

[4]Example taken from `https://lists.w3.org/Archives/Public/public-rdf-star/2022Feb/0002.html`

Of course, if Turtle provided an ID attribute like RDF/XML does and even supported multiple identifiers, something like the following serialisation would become possible:

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix dc11: <http://purl.org/dc/elements/1.1/> .
@prefix ex: <http://example.org/> .
@prefix id: <http://rdf2.org/identifier#> .

<http://www.w3.org/TR/rdf-syntax-grammar>
    dc11:title "RDF1.1 XML Syntax" ;
    ex:editor [
        ex:fullName "Dave Beckett" ;
        ex:homePage <http://purl.org/net/dajobe/> id:one,two
    ] .
    id:one ex:accordingTo <http://example.org/source1> ;
           ex:urlDate "2004-4-16" .
    id:two ex:accordingTo <http://example.org/source2> ;
           ex:urlDate "2004-4-26" .
```

6.2 Notation3

[BC08] presents Notation3 (N3), a superset of the popular RDF serialisation Turtle. In fact, Turtle was derived from N3 by omitting formulae and logical operators that encode a rule language in N3. A *formula* is a set of statements enclosed in curly braces. It can occur in subject and object position of a statement. Statements contained in a formula can themself contain other formulae. For that reason formulae are also known as nested graphs. To give an example:

```
{
    { [ x:firstname  "Ora" ] dc:wrote [ dc:title  "Moby Dick" ] }
        a n3:falsehood
} a :example .
```

Formulae have a very specific semantics: they are not asserted and are referentially opaque. The purpose of this arrangement is to keep statements in those formulae separate from the knowledge that is known and accepted to being true. It also allows for handling contradicting statements isolated from each other and the main knowledge base. See also Sections 4.2.5.3 and 9.

6.3 Dark Triples

"Dark Triples" are a concept from the early years of RDF standardization that pops up now and then. [Hay02] ponders the possibility to separate triples that make factual assertions concerning the domain of interest from triples that merely put up some data structure like lists or n-ary relations — "but these two uses tend to trip over one another. Allowing data structures to be sets of dark triples frees them from accidentally making assertions that are inappropriate to the intentions of the user of the datatstructure." Another possible application would be OWL axiomatic constructs.[5] [Hay03] elaborates on how dark triples could prevent the problems that the Semantic Web's layered architecture generates when constructs that on a base layer only have syntactic purposes become 'meaningful' in higher layers and can lead to unwanted entailments, even resulting in inconsistencies as reported by [PF02].[6] In short "if you describe the syntax of the logic in the logic itself, you get into well known problems, paradoxes..."[7]

6.4 JSON-LD

JSON-LD[8] is a serialisation of RDF optimized for use in web development. In web contexts like browsers and JavaScript-driven programming, the JavaScript Object Notation (JSON) is the most prevalent data format. JSON-Linked Data (JSON-LD) tries to be as idiomatically JSON as possible to make it easy for web developers to tap into Semantic Web data sources. It has a certain similarity with Turtle although the conceptual mismatch between RDF's graph structure and JSON's name-value pairs unavoidably shows in the syntax. Because of this mismatch JSON-LD inherits some syntactic possibilities from JSON that Turtle and other native RDF serialisations don't provide. [Fre+19a] for example present a JSON-LD serialisation of DBpedia that takes some freedom in inventing new structural elements like an 'objects' node. Or, as Kellogg notes, "JSON-LD anonymous named graphs are already very much like [Notation3] formulae, given the way the graph value of a property (@container: @list) creates a blank node which is both the value of that property and the name of an associated named graph."[9] It is also nearer to generalized RDF,[10] e.g. allowing literals in subject position.

JSON is itself not without verbosity — which JSON-LD necessarily inherits — and YAML-LD[11] tries to address that problem.

[5] [Kly02] provides some more more detail.

[6] See also Section 2.6.1.3

[7] Taken from a chat log of an RDF Core WG meeting, `https://web.archive.org/web/20170717120641/http://chatlogs.planetrdf.com/rdfcore/2002-06-17.html`. Hard to follow, but an interesting read.

[8] `https://www.w3.org/TR/json-ld11/`

[9] `https://lists.w3.org/Archives/Public/semantic-web/2020Aug/0056.html`

[10] `https://www.w3.org/TR/rdf11-concepts/#section-generalized-rdf`

[11] `https://json-ld.github.io/yaml-ld/spec/`

Chapter 7

Ontologies

Ontologies play an immensely important role in the architecture of the Semantic Web. Ontologies represent shared conceptualizations of a domain. They define terms to describe the entities a domain is composed of and the types of relations and attributes that those entities can have. A car ontology would e.g. describe types of cars such as limousines and cabriolets, the parts they are composed of such as engine and wheels, attributes such as size and color, relations such as ownership, etc. The terms so defined would be documented in web accessible documents, accompanied by descriptions that humans can look up to understand what terms an ontology offers, how they are to be used for publishing data on the Semantic Web and what they are supposed to mean in case one comes across them on the web of data. RDF, RDFS and OWL provide vocabularies to define basic relations in ontologies like e.g. class hierarchies and property value spaces. However, apart from those basic building blocks the real work is done in domain-specific ontologies about any subject area whatsoever.

Without such shared domain ontologies it would be very hard to successfully navigate the Semantic Web, make sense of the data it offers and reuse it for new applications. But even seemingly straightforward ontologies are abstractions of often quite complex concepts. One might have an immediate intuition about what a postal address is and of what parts it is composed. Yet even this seemingly simple and commonplace concept is of astounding diversity, as the well-known "Falsehoods programmers believe about addresses"[1] blog entry documents. Another example is the property `foaf:knows` from the Friend-of-a-friend ontology: a detailed and exhaustive description of the meaning of being a friend could take any arbitrarily long amount of time and space, and still be incomplete. There is a lots and lots of room for misunderstandings, unintentional misuse and the introduction of hidden incompatibilities and contradictions. However nobody wants to wade through loads of documentation and vocabulary descriptions just to publish some snippet of data. Ontologies for the Semantic Web have to strive for generality and reusability. Ideally they provide a sparse and crisp core set of

[1]`https://www.mjt.me.uk/posts/falsehoods-programmers-believe-about-addresses/`

terms that gets the job done. Corner cases and outliers may then be covered by composition and specialized extensions.

In the context of this work ontologies are relevant in several respects. They can be used to:

– declare semantics
– disambiguate identification
– express structure
– facilitate annotation and contextualization
– provide vocabularies for popular tasks like provenance recording and mapping to external data sources

However, apart from these applications this section doesn't aim to present an overview of all ontologies developed for and published on the Semantic Web, as that would be a enormous task on its own.

7.1 Semantics

Despite the fact that a configurable semantics for RDF has been demanded, proposed and discussed many times — see e.g. Section 2.6.4 — there is very little vocabulary available to that effect. The W3C maintains a page with IRIs for Semantic Web Entailment Regimes[2] which in turn is referenced by the SPARQL 1.1 Service Description specification[3] to denote the entailment regime a SPARQL endpoint supports. The listed entailment regimes are those defined by the standard Semantic Web specifications: RDF, RDFS, OWL and RIF.

[Zim14] explains that the RDF 1.1 Working Group did not only fail to standardize a semantics of named graphs but couldn't even bring itself to define a mechanism to declare a desired semantics —"because of the lack of experience". The Note describes a possible solution based on existing vocabularies, namely the SPARQL 1.1 Service Description specification mentioned above:

```
@prefix er: <http://www.w3.org/ns/entailment> .
@prefix sd: <http://www.w3.org/ns/sparql-service-description\#> .
[]  a sd:Dataset ;
    sd:defaultEntailmentRegime er:RDF ;
    sd:namedGraph [
        sd:name "http://example.com/ng1" ;
sd:entailmentRegime er:RDFS ] .
```

The Note sketches further possible mechanisms but again hints that no current practice is available to provide some guidance.

[Bao+10b] points to the Rule Interchange Format (RIF)[4] suite of specifications. Specifically the RIF Framework for Logic Dialects (RIF-FLD)[5] is designed to create machine-readable specifications of logics with both model & syntax and a model-theoretic semantics.

[2]https://www.w3.org/ns/entailment/
[3]https://www.w3.org/TR/sparql11-service-description/#sd-EntailmentRegime
[4]https://www.w3.org/TR/rif-overview/
[5]http://www.w3.org/TR/rif-fld/

[AW19] show how to implement different semantics in Notation3.[6] Their approach takes advantage of the fact that in Notation3 nested graphs, there called *formulas*, are referentially opaque and therefore amenable to all kinds of more general semantics. That way different semantics can be implemented within a single document. They provide example implementations of four semantics that the RDF 1.1 WG considered as possible semantics for named graphs, as reported by [Zim14], namely:

- all graphs participate equally in the truth of the dataset and graph names have no meaning
- graphs are referentially opaque and the name denotes the graph,
- graphs are referentially transparent and the name denotes the graph,
- graph names refer to web resources (however, it's unspecified if they refer to the web resource itself or to what the web resource denotes).

Properties used from the N3 logic vocabulary are `log:equalTo`, `log:semantics`, `log:conjunction` and `log:conclusion`. Properties defined for this purpose, and implemented as N3 built-ins, are `sem:unionWithDefault`, `sem:mergeWithDefault`, `sem:combinedWithDefault`, `sem:quotedGraph`, `sem:isolatedGraph`, `sem:entailment`, `sem:onlineRelationTo` and `sem:onlineEquals`.

[RH22] describes an implementation of Hayes' BLogic proposal[7]. The system is build on top of a Notation3 rules engine and implements First Order Logic (FOL) via nested graphs,[8] here called *surfaces*. Predicates like `log:onPositiveSurface`, `log:onNegativeSurface` and `log:onNeutralSurface` are used to combine expressions 'on' surfaces to FOL logical formulas. Other operators like 'unasserted' or 'referentially opaque' are not implemented yet but would be possible as well. The following example states (in Notation3 syntax) that "If any cat X is alive, then X says meow":

```
@prefix : <http://example.org/ns#> .
@prefix log: <http://www.w3.org/2000/10/swap/log#> .
(_:X) log:onNegativeSurface {
    _:X a :Cat ;
        :is :Alive .
    () log:onNegativeSurface {
        _:X :says :Meow .
    } .
} .
```

The Semantic Web Application Platform (SWAP)[9] vocabulary[10] also provides properties like `log:equalTo`, `log:implies` and `log:conclusion`.

[6]See Section 4.2.5.3
[7]See 2.6.4.2
[8]See 4.2.5.3
[9]`https://www.w3.org/2000/10/swap/`
[10]`http://www.w3.org/2000/10/swap/log`

7.2 Identification

On the Semantic Web ontologies are the natural way to provide expressive means for anything that is not encodable with the means provided by the RDF model and syntax. Given the problematic semantics of identification in RDF and the dissatisfaction with solutions like hash IRIs and HTTP status codes as discussed at length in Section 2.1, it is not surprising that some extensions and alternatives to existing vocabularies have been proposed.

7.2.1 Sense, Reference & Access

Investigations into the use of ontologies to solve the 'identity crisis' of the Semantic Web are relatively rare.

[HH08] sketch a vocabulary to differentiate the usage of an IRI for reference or access as an alternative to encoding it in status codes in HTTP response headers or via hash signs appended to IRIs. A property `refersTo` relates a description of an entity to a reference to the entity itself. The former would probably address an information resource like some web page whereas the latter could refer to a not web-accessible entity like e.g. an abstract concept or a person (either living, imaginary, historic. . .)., but just as well to another web page. Another property called `describedBy` provides a relation in the opposite direction, from an IRI referring to an entity — again either web-accessible or abstract — to a web accessible resource describing it. A class `nonAccessible` serves to expressly state that an IRI denotes a not web-accessible entity. The proposed properties are considered similar in meaning to the properties `topic` and `page` from the FOAF[11] vocabulary. The authors point out that the distinction between access and reference is specific to the Semantic Web, which suggests that expressing it in RDF is more suitable than resorting to other technologies like hash IRIs and HTTP response headers. As an additional benefit they mention the possibility to embed such properties in web pages themselves using techniques such as RDFa and GRDDL.

[HP09] present the Identity of Resources on the Web (IRW)[12] ontology as a means to formally model concepts such as `identity` and `resource` and the debates about their meaning. The purpose of that vocabulary is to help resolve conflicting arguments about identification semantics as discussed above in Section 2.1, to "allow those involved in debates to model formally their positions using a common ontology as a starting point", to make the Semantic Web more self-describing and to support the implementation of practical solutions in a variety of different scenarios like e.g. expressing if an IRI is used to refer to a web page itself or to what the web page is about. To that end they differentiate between `direct reference position` and `logicist position` of identification, between `access` and `reference`, between `hash re-direction` and `303 re-direction`, between IRIs for non-accessible things on the Semantic

[11] `http://www.foaf-project.org/`
[12] `http://ontologydesignpatterns.org/ont/web/irw.owl`

Web as `non-information resources` and subclasses `physical entity resource`, `conceptual resource` and `abstract resource` and the normal use of IRIs on the hypertext Web as `information resources`, and so on.

The widely used Schema.org vocabulary defines a 'sameAs' property of its own. The definition reads: "URL of a reference Web page that unambiguously indicates the item's identity. E.g. the URL of the item's Wikipedia page, Wikidata entry, or official website."[13] and as example notes "Utilising Wikidata as a source of URIs for entities in a sameAs relationship." This is substantially different from the definition of `owl:sameAs` and seems to aim at addressing the 'Identity Crisis' of the Semantic Web.[14]

[Ber96a] outlines an ontology of types of representations of a resource[15], depending on aspects such as content negotiation over HTTP requests, temporal aspects etc.

7.2.2 Not Quite the sameAs

Various works have investigated alternatives and supplements to the `owl:sameAs` property, providing both weaker and stronger variants. They do however all struggle with the task of precisely defining clearly differentiated levels of relatedness and some come without any formal semantics to support inferences.

[HH10] see four different meanings embodied in popular usage of `owl:sameAs`: `Same Thing As But Referentially Opaque`, `Same Thing As But Different Con text`, `Represents`, and `Very Similar To`. They discuss how those meanings are represented in `owl:sameAs` as actually defined and in other properties from popular vocabularies like e.g. SKOS and FOAF, stating a need for more empirical analysis as well as considerable work on the semantics and core constructs of RDF to solve this problem.

[MM10] describe the properties of `owl:sameAs` as being isomorphic and also transitive, symmetric and reflexive. From the permutations of that set of properties they define the Similarity Ontology (SO) of identity relations, namely `identical`, `similar`, `claimsIdentical`, `claimsSimilar`, `exactlyMatches`, `related`, `matches` and `claimsRelated` and illustrate their use on the basis of well-known examples like the Superman problem[16]. These properties are posed into subproperty relationships with existing identity properties from RDFS, OWL and SKOS. This ontology is again prominently featured by [Hal+10].

[Mel13] suggests the introduction of a new property for strict identification to explicitly capture the precise semantics of `owl:sameAs`, as usage of

[13] `https://schema.org/sameAs`, accessed on Aug. 5th 2022

[14] The definition of schema:sameAs seems to have slightly changed over time. [Bee+18] describe the property as follows: "the semantics of this property is substantially different from that of `owl:sameAs`. It states that two terms 'are two pages with the same topic' and does not express equality." However, a cursory examination of different snapshots on archive.org, e.g. `https://web.archive.org/web/20180926060756/https://schema.org/sameAs`, couldn't confirm such strong deviations from the current textual definition.

[15] `http://www.w3.org/2006/gen/ont`

[16] As discussed in Section 2.6.2.8

`owl:sameAs` 'in the wild' can in general not be trusted to follow its actual semantics. The Lexvo.org ontology[17], presented by [MW08], to that end defines the term `strictlySameAs`[18], and in addition the terms `nearlySameAs`[19] and `somewhatSameAs`[20] to cater for less clear cut cases. [Bee+18] find similar constructs, e.g. a declaration that `bbc:sameAs rdfs:subPropertyOf owl:sameAs` but interpret them as erroneous modelling, suggesting that the intention was to define a semantically less strict version of `owl:sameAs`.

Wikidata defines a property named `exact match` (or P2888[21] in Wikidata nomenclature) that is described as "used to link two items or properties, indicating a high degree of confidence that the concepts can be used interchangeably. Also known as skos:exactMatch."

The Upper Mapping and Binding Exchange Layer (UMBEL), a now defunct project described as providing "a scaffolding to link and interoperate other datasets and domain vocabularies"[22] defines a property `isLike`[23] to "assert an associative link between similar individuals who may or may not be identical, but are believed to be so. [...] where there is some uncertainty that the two resources indeed refer to the exact same individual with the same identity."

The Simple Knowledge Organization System (SKOS)[24] provides a set of mapping relations to describe varying degrees of similarity between resources. The SKOS reference[25] describes those properties and also the design that drives their formal semantics — or rather the lack thereof — in detail. A `closeMatch` is to be used to "link two concepts that are sufficiently similar that they can be used interchangeably in *some* information retrieval applications." It is explicitly declared to *not* be a transitive property to avoid unexpected inferences when mapping over different sources. Using `exactMatch` indicates a "high degree of confidence that the concepts can be used interchangeably across a wide range of information retrieval application." It is defined as transitive. The properties `broadMatch` and `narrowMatch` can be used for hierarchical mappings between resources whereas `relatedMatch` is intended to express an 'associative' relation. SKOS is designed with an explicitly restrained approach to inference and is "*not* a formal knowledge representation language." Its purpose is to bring thesauri and other rather informal classification schemes to the Semantic Web. Those thesauri describe "a set of distinct ideas or meanings, which [. . .] may also be arranged and organized into various structures, most commonly hierarchies and association networks. These structures, however, do not have any formal semantics, and cannot be reliably interpreted as either formal axioms or facts

[17] `http://lexvo.org/`

[18] `http://lexvo.org/ontology#strictlySameAs`

[19] `http://lexvo.org/ontology#nearlySameAs`

[20] `http://lexvo.org/ontology#somewhatSameAs`

[21] `https://www.wikidata.org/wiki/Property:P2888`

[22] `https://web.archive.org/web/20190928031549/http://www.umbel.org/`, archived version from 28.9.2019

[23] `https://web.archive.org/web/20190928021401/http://techwiki.umbel.org/index.php/UMBEL_Vocabulary`

[24] `https://www.w3.org/TR/skos-primer/`

[25] `https://www.w3.org/TR/skos-reference`

about the world.” Transforming such structures into a set of formal axioms and facts requires a lot of effort. SKOS instead aims at the low hanging fruit, arguing that “[m]uch can be gained from using thesauri, etc., as-is, as informal, convenient structures for navigation within a subject domain.” And the reference warns that “`owl:sameAs`, `owl:equivalentClass` or `owl:equivalentProperty` would typically be inappropriate for linking SKOS concepts in different concept schemes, because the formal consequences that follow could be undesirable.”

[Bee+18] in their analyses of `owl:sameAs` usage report extensions to `owl:sameAs` like e.g. the introduction of super- and subproperties of `owl:sameAs` — some of them defined in ways that probably don’t properly reflect their intended meaning.

[Bat+14] employ a hierarchy of relatedness relations when consolidating equivalence relations between linksets as follows: `owl:sameAs` $\leq$ `skos:exactMatch` $\leq$ `skos:closeMatch` $\leq$ `rdfs:seeAlso`.

7.2.3 Contextualized owl:sameAs Ontologies

While none of the aforementioned works examine the requirements and effects of contextualization on vocabularies, [HHT15], discussed above in Section 5.2.1, suggest that “what is needed is a way to assert a graph in a context, a way to specify the reserved vocabulary of a context, a way to describe the semantic conditions imposed on the reserved vocabulary by the context, and a way to assert that one context inherits another.” On that basis a variant of `owl:sameAs`, e.g. `rdf2:sameAs`, with pollarding semantics could be defined. Beyond that however many more contextualized versions of `owl:sameAs` would become possible, as [HHT15] expands: “we can also imagine many different new domain specific usages of identity being created, such as sameAs within the context of geographical mapping, a certain historical era, or some other application that needs to be able to link data without making grand universalist claims. Contexts provide a degree of useful referential opacity, as an sameAs asserted in one context might cease to be true in a subcontext when more refined meanings are in use, such as in comparing chemical elements vs. chemical isotopes.”

7.3 Qualification

Qualification of terms has been discussed as one way to add detail and context to RDF statements. In terms of vocabulary this would require very little effort. More elaborate approaches introduce a concept of roles.

7.3.1 Qualification Vocabularies

The most widely deployed qualification vocabulary is the `rdf:value` property, as discussed above in Section 2.4.1. However, this property is defined without any formal semantics and with quite specific scope, which makes it rather unfit for a task as frequent but also semantically delicate as qualification.

[Lin22] sketches a small vocabulary consisting of four terms: the class `quid:QualifiedIdentity` and `quid:QualifiedValue` and the properties `quid:qualifies` and `quid:value`. Its aim is to explore "the possibility of using qualified entities as an alternative to the use of qualified relationships, to contrast the notions and challenge their differences." An example deploys this vocabulary to express that the town Abingdon for a few hundred years was part of the county of Berkshire, i.e. the object of the statement, the county Berkshire, is temporally qualified:[26]

```
prefix quid: <https://w3id.org/quid/>
<Abingdon> :partOf [ a quid:QualifiedIdentity ;
    quid:qualifies <Berkshire> ;
    :earliestYear "1600" ;
    :latestYear "1974" ;
    rdfs:label "Berkshire from 17c. to 1974"@en
] .
```

The property `https://w3id.org/quid/qualifies` is defined as equivalent to `prov:specializationOf`, the former just being more handy for the exploration of the ramifications of qualification, which the author indeed pursues in great detail.

The PROV-O ontology[27] defines the property `prov:specializationOf` as follows: "[a]n entity that is a specialization of another shares all aspects of the latter, and additionally presents more specific aspects of the same thing as the latter. In particular, the lifetime of the entity being specialized contains that of any specialization. Examples of aspects include a time period, an abstraction, and a context associated with the entity." This definition seems broad enough to cover any conceivable application of qualification.

7.3.2 Roles

[All+03] discuss how the concept of roles could be a suitable notion to extend the predominantly class-based domain models prevalent on the Semantic Web with contextual aspects. They define a 'role' type as describing a relationship into which an entity stands but which doesn't determine its identity. A 'natural' type on the other hand is constituent to the objects identity, much like a class membership relation.[28] An example in RDFS represents roles as domain and range declarations on properties.

[Bri14] proposes a data structure and vocabulary for richer descriptions than the simple and unqualified binary relations that the Schema.org[29] vocabulary provides. It is based on the notion of roles that the nodes in a relation can take and introduces a reification vocabulary quite similar to RDF standard reification, with properties `:roleSubject`, `:roleProperty` and `:roleObject`.

[26]While the example is reminiscent of fluents as discussed in Section 4.2.2 the approach has unconstrained applicability to location, viewpoint etc.

[27]`https://www.w3.org/TR/prov-o/`

[28][Ste00] provides an in-depth discussion of the concept of roles.

[29]`https://schema.org/`

Examples, however, seem to not be quite sure where to attach annotations:

```
{
    "@context": "http://schema.org" ,
    "@type": "AmericanFootballRole" ,
    "roleSubject": {
        "@type": "AmericanFootballTeam" ,
        "name": "San Francisco 49ers"
    },
    "roleProperty": {
        "http://schema.org/athlete"
    },
    "roleObject": {
        "@type": "Person" ,
        "name": "Joe Montana"
    },
    "startDate": "1979" ,
    "endDate": "1992" ,
    "position": "Quarterback"
}
```

annotates the whole object instead of the rolePredicate with start- and end date and position. Annotations on the enclosing object for administrative purposes would risk to be mingled with such qualifying annotations on the described fact itself.[30]

Schema.org implements the notion of qualifying roles, but in a quite different way[31]. Qualification can only be applied to objects in a relation, but it doesn't rely on a generic roles vocabulary. Rather, in a variation of the `rdf:value` approach, the value in a relation is replaced by a blank node acting as a liaison. The primary value is then related to that liaison by repeating the original property, e.g.:

```
@prefix schema: <http://schema.org/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
[
    a schema:Organization ;
    schema:member [               <--- original property and liaison
        a schema:OrganizationRole ;
        schema:member [           <--- original property repeated
            a schema:Person ;
            schema:name "Alice"^^xsd:string
        ] ;
        schema:startDate "1977"^^schema:Date
    ] ;
    schema:name "Cryptography Users"^^xsd:string .
]
```

Contrary to the proposal of [Bri14] this approach doesn't allow to qualify subject or predicate or the whole relation, but it is more idiomatic and easier to integrate into existing data structures, queries and reasoning procedures.

[30]The example is serialized as JSON-LD. The syntax is not valid, as the roleProperty value can't be encoded as an unary JSON object, but the point is clear enough.

[31]As documented at `https://schema.org/Role`

7.4 Structure

Apart from the RDF list vocabularies[32] and shape languages[33] there exist only few vocabularies to describe RDF structures in RDF.

[PRF12] explore ways to connect simple triples with n-ary representations of the same fact, but annotated with further detail. E.g. one may find a simple statement such as:

```
<mypage.html> tag:taggedWithTag ns:semweb .
```

represented as an n-ary relation like so:

```
[]  a tag:Tagging ;
    tag:taggedResource <mypage.html> ;
    tag:associatedTag ns:semweb .
```

for the purpose to add source or date of annotation. The simple statement might be understood as a shortcut version of the annotated n-ary relation.[34] The authors observe that "RDFS and OWL provide no way of expressing that both forms are equivalent. A Tag Ontology processor needs custom code to be able to deal with both forms." They draft a vocabulary to make such equivalences explicit, "thereby allowing reasoners to automatically convert between the two forms." A set of N3 rules is provided that performs such a conversion. The wiki also discusses alternative approaches e.g. based on OWL Property Chains, or s streamlined version of singleton properties:

```
:something :reifiedTag :tag .
:reifiedTag a ex:Reification ;
    rdfs:subPropertyOf tag:taggedWithTag ;
    dcterms:date "..." .
```

[Giu+21] present a model and accompanying vocabulary to describe n-ary relations as 'parametric properties' in a principled and uniform way. It can be applied equally well to unary, binary and n-ary relations. In this approach each type of n-ary relation is described as a numbered list of `n` relations, using the RDF collections vocabulary. The primary subject of the n-ary relation is attributed with the relation type. From that knowledge the whole n-ary relation can be gathered by following the chain of `n` relations that each represent the `n`'th relation in the `n`-ary relation. The approach is quite expressive and principled, but also has a few drawbacks. It is rather verbose, requiring the explicit definition of the relation type and its `n` properties. Multiple n-ary relations of the same type with the same subject node would require separately defined properties to be distinguishable from each other. Applying the approach to unary and binary relations would double the triple count with no apparent benefit except that the same 'parametric properties' model is applied to relations of all arities. Applying the parametric property to a relations with arity > 2 in practice would result in a chain of triples rather than a tightly connected object and therefore

[32]See Section 2.4.2
[33]See Section 4.1.3
[34]See Section 4.1.1.2

is not easily recognisable in an RDF serialisation and does not really qualify as idiomatic RDF. Practical implementations would probably ask for additional support in application logic and user interface.

[GP13] describe a similar approach, named Situation pattern, which defines an n-ary relation type by using the `owl:hasKey` property to define its attributes. It then creates instances of that type, attributed with those properties. It doesn't use sequences which, depending on application, may or may not be of merit.

7.5 Context

One of the most notorious, and vacuous, terms to come up in the context of meta-modelling is *context.* Everybody and her cat seems to have its own definition of what the term means[35] This section documents some of those ontologies. It starts with CYC,[36] a system that bets heavily on the notion of context and applies it many different ways, even basing its reasoning machinery on it.

[Len98] lists the 12 dimensions of context space that the CYC system uses to contextualize data:[37]

Absolute Time a particular time interval in which events occur
Type Of Time a non-absolute type of time period, such as "just after eating"
Absolute Place a particular location where events occur, such as "Paris"
Type Of Place a non-absolute type of place, such as "in bed"
Culture linguistic, religious, ethnic, age-group, wealth, etc. of typical actors
Sophistication/Security who already knows this, who could learn it, etc.
Topic/Usage drilling down into aspects and applications — not subsets
Granularity phenomena and details which are (and are not) ignored
Modality/Disposition/Epistemology who wants/believes this content to be true?
Argument-Preference local rules for how to resolve pro-con argument disputes
Justification are things in this context generally proven, observed, on faith...
Let's local bindings of variables etc. that hold true in that context

CYC even decades after its inception is still regarded as the gold standard in contextualized knowledge representation and its list of context dimensions is often referred to in the literature. Yet the overlap with dimensions gleaned from works about the Semantic Web is limited: *Absolute Time* maps to temporal situation, validity or provenance, *Absolute Place* likewise to spatial situation, validity or provenance. *Sophistication/Security* covers access control and rights management. *Topic/Usage* maps to tagging and keywords, although 'usage'

[35]This author's preferred, but by no means authoritative interpretation is that 'context' refers to further information external but related to the event or topic in question. Which of course begs the question of how *external* is defined...
[36]See Section 9.3.3.1
[37]According to a recent talk by Lenat, `https://www.youtube.com/watch?v=v2rK40bNrrY` they still use those 12 categories

seems like a rather different category. *Granularity* maps to context nesting, lifting and inheritance, *Modality/Disposition/Epistemology* to viewpoints and propositional attitudes. *Justification* may cover aspects of probability, plausibility and trust. On the other hand some aspects like “Type Of Time”, “Type Of Place”, “Argument-Preference” and “Let’s” are completely missing in Semantic Web oriented approaches. This can partly be explained by CYC’s focus on common sense knowledge which requires to incorporate for example indexicals, which are generally considered out of scope on the Semantic Web. The list of dimensions, as comprehensive and elegant as it undoubtedly is, will hardly feel intuitive and complete to everybody. It was probably a wise design decision that on the Semantic Web such conceptualizations are not baked into the core model but delegated to ontologies. So far any attempts to integrate even the seemingly most straightforward dimensions like time and space — which time, which space one might immediately ask, knowing e.g. CYC’s more differentiated approach — into the basic formalism have failed to garner wide and continued support.

Other authors have of course brought forward other classifications of contextual dimensions. [MK03] understand contextualization as referring to “sets of data attributes that vary according to the context in which they are viewed”, e.g. changes across time or different access levels depending on security settings. On the other hand metadata like provenance is not understood as context. [GM03b] list the following varieties of context: “Projection Contexts, Approximation Contexts, Ambiguity Contexts and Mental State Contexts.” They define each type, describe sub-types, discuss practical aspects and logical requirements. They also explore combinations of contexts, especially ‘lifting’ information from one context into another. [GMF04] argue that while a context mechanism is very useful when integrating data from different sources, the Semantic Web will not need a system as sophisticated as CYC’s. To that end they propose “a small number of varieties of context” to address different definitions of shared concepts, implicit assumptions based on situation and context, property type differences (eg design capacity vs actual capacity), differing viewpoints (eg in politics), implicit time and different levels of precision in measurements. [BGS13] in their Contextualized Knowledge Repository (CKR) framework[38] implement three context dimensions: time, location and topic. However, they consider the number of possible dimensions unlimited. [Pat18] provides an extremely flexible definition of context, as being “a meaningful unit that contains a set of facts, which jointly describe a portion of a structured domain to a certain level of approximation”, making any specific context vocabulary all but superfluous.

7.6 Provenance

Provenance is probably the most common use case when meta-modelling in RDF is discussed and it is indeed a dominant cross-cutting concern when integrating data from different sources, as the Semantic Web aims for. One of the earliest

[38]See Section 9

use cases for RDF was the annotation of web pages with metadata like author, keywords etc. and the predominant vocabulary to that effect was maintained by the Dublin Core Metadata Initiative (DCMI): The Dublin Core (DC) ontology.[39]

A more recent initiative with a much more extensive scope is PROV,[40] a whole suite of provenance related vocabularies, mappings and tools developed by a W3C Working Group. One of its central components is the PROV-O ontology[41] for mapping of the PROV data model to RDF. [ZH12] discuss interoperability issues between PROV and pre-existing standards like Dublin Core.

It would be of particular interest to this work if any such provenance vocabularies discuss or even implement some kind of disambiguation between a web-addressable resource itself and what that resource refers to, e.g. between a webpage about the Tour Eiffel and the Tour Eiffel itself, as discussed in Chapter 5. However, no such approach has been encountered.

7.7 Annotation

A thorough treatment of annotations is out of scope for this work, unfortunately. However, two vocabularies should at least be mentioned very briefly.

A W3C Working Group developed the Open Annotation Data Model (OA), later renamed to Web Annotation Data Model.[42] [SCS13] provide an introduction into the vocabulary, concentrating on a discussion of design goals and technical decisions. They describe the goal of OA as enabling annotations that "can easily be shared between platforms, with sufficient richness of expression to satisfy complex requirements while remaining simple enough to also allow for the most common use cases." Example use cases include comments, text highlighting, tagging, diffs and editing requests. The data model is composed of three components: annotation event, annotation body and annotation target. As the target is understood to be an addressable resource, textual or otherwise, problems of identification as discussed in Section 2.1 are not an issue. [Kim+15] investigate the application of OA to RDF statements and criticize that like with RDF standard reification the connection between statement and annotation is lost. They instead "advocate to put the statements [. . .] into named graphs rather than reifying them, so that the statements can be preserved as they are. [. . .] Then, meta information about the annotation can be attached to the named graphs." They claim that this approach adds a welcomed degree of flexibility to provenance recording e.g. when combining annotations from different domains and point out its compatibility with nanopublications.[43]

[39]Project homepage: urlhttp://dublincore.org/, namespace: `http://purl.org/dc/elements/1.1/`and `http://purl.org/dc/terms/`, documentation: `http://dublincore.org/specifications/`, more information: `https://www.w3.org/wiki/Good_Ontologies`

[40]`http://www.w3.org/TR/prov-overview/`

[41]`https://www.w3.org/TR/prov-o/`

[42]`https://www.w3.org/TR/annotation-model/`

[43]See Section 4.2.5.4

OWL provides an annotation vocabulary that doesn't support entailment. "[OWL] Annotation properties are used to encode information about (parts of) the ontology itself (like the author and creation date of an axiom) instead of the domain of interest."[44] This very specific semantics makes the vocabulary unavailable for any uses outside OWL ontology development.

7.8 Mappings to RDBMS's

Relational Data Base Management Systems (RDBMS) have been the mainstay of data processing for decades and aren't going anywhere anytime soon. Consequently it has been investigated how to integrate the data stored in those systems with the graph based data model that drives the Semantic Web: convert it, make it accessible per API, provide integrated views over both relational and graph data etc. The relational data model very directly maps to the neo-Davidsonian relation as already described in Section 2.7.4. Therefore only a very coarse overview of those works is given here.

As [Hog+14] report , "[g]iven an increasing interest in publishing relational data as RDF, the RDB2RDF W3C Working Group was tasked with standardizing a language for mapping relational data into RDF. As a result of the activity of this group, two languages were proposed: a direct mapping [RML, [Are+12]] that translates a relational database into RDF without any input about the transformation process from the user, and a general mapping language [R2RML, [DSC12]] where users can specify their own rules for translating a relational database into RDF." [Bon+18] elaborate that "R2RML is a language that allows the user to define custom mappings from a relational database schema to an RDF model. The R2RML document itself is written in RDF, in which mappings for each table can include a template for the desired output URI structure, specified ontology classes to be instantiated to, and relationships between columns. Such relationships can be used to link to other relational databases through, for example, primary keys. Since the input data are stored in relational databases, SQL queries can be used in the R2RML mapping files when constructing the desired RDF. An extension to R2RML is RML, which aims to be more generic by keeping the core model of the R2RML vocabulary, but excluding database specific concepts." RML is described in [Dim+14]. [SPV12] identify re-occurring patterns of mappings from RDBMS to RDF. [SAM12] investigate the use of OWL and Datalog to develop mappings that are monotonic and preserve semantics. They find that monotonicity and semantics preservation are incompatible.

More recently [TH20] provide a compact rundown of the state of the art and [SL21] offer a practical introduction of how to integrate data RDBMS into Knowledge Graphs.

[44] `https://www.w3.org/TR/owl2-primer/`

7.9 lowercase Upper Ontologies[45]

Upper ontologies are usually not directly used on the Semantic Web but may rather inform the design of Semantic Web vocabularies. However, among the many vocabularies that have been developed, one has some similarity to an upper ontology because of the breadth of topics it covers: the Schema.org vocabulary. Schema.org stems from a joint initiative of major web search engines to promote a unified vocabulary to describe web pages and the items they publish, from recipes to e-commerce items. [GBM15] provides an introduction into background and design of the vocabulary. [Pat14] analyses the semantics of the vocabulary, identifies gaps and contradictions in the definitions of terms and relations and proposes ways to correct those shortcomings, providing a formal model-theoretic semantics for schema.org.

Another example of ontologies of considerable breadth and depth are the structures of large knowledge graphs like Wikidata or DBpedia. [She+21] study the quality of Wikidata according to three indicators: community consensus, deprecation and constraint violations. They “combine these indicators to detect low-quality statements, revealing challenges with duplicate entities, missing triples, violated type rules, and taxonomic distinctions.”

[45]The title is a pun on an at least one decade old rallying cry to strive for a ‘lower case semantic web’, in differentiation to a Semantic Web that was perceived to be ruled by logicians and bogged down by their heavy handed rule of cumbersome formalisms and idiosyncratic arrangements.

Chapter 8

Querying

SPARQL is the predominant query language on the Semantic Web. It is dominated by Basic Graph Pattern matching. Work on the upcoming version 1.2 aims at better support for path querying and graph analytics. This would strengthen the graph aspect of RDF which so far is lagging behind Labelled Property Graphs (LPG), where especially the query language Gremlin provides prominent support for path queries and graph analytics. Implementation aspects however play a role too — e.g. path queries are often optimized for in-memory stored graphs which makes them rather impractical for massive knowledge graphs — but are not further discussed here.

[Ang+17] provide a comprehensive introduction into query languages for graph databases, covering both RDF and LPG flavours and the respective most popular query languages SPARQL for RDF and Cypher and Gremlin for LPG. They start with an introduction into graph data models, go on discussing graph pattern and navigational queries and the combination of the two paradigms, covering practical as well as theoretical aspects in considerable depth, and do not stop at questions of semantics like weighing the pros and cons of sets vs. bags.

SPARQL is the small door through which all meta-modelling techniques must pass: the more elaborate they get, the more joins they tend to require, and the less likely is it that involved queries will perform satisfyingly in practice. Degrees of freedom in expressivity quite directly correspond to inverse query performance. On the other hand more involved constructs may be harder to query, but easier to comprehend and use correctly — in authoring as well as in querying. Such benefits, performance or otherwise, are difficult to balance in design and hard to compare in practice.

One of the most prevalent meta-modelling techniques in RDF, named graphs, was first standardized in SPARQL, but without formal semantics, rather as a database administrators tool to group triples by source, access rights, etc. Nonetheless SPARQL's syntactic support for named graphs is often used for meta-modelling with application-specific semantics. Their particular query syntax gives them an edge in performance comparisons with triple-centric approaches, but at the expense of flexibility and generality: named graphs tend to be used for one sole meta-modelling dimension, dictated by the application at hand. Another indication of their rather one-dimensional design is that their implementation in SPARQL doesn't support path queries spanning multiple graphs.

However, an approach that improves modelling of involved knowledge structures is not necessarily easier to query, for which singleton properties may serve as an example. Some approaches introduce new features in the model of RDF, resulting in a need to expand SPARQL for their support like RDF-star and the accompanying SPARQL-star.

An important aspect to keep in mind is how resilient an approach is to changes in the data. If attributions require changes to existing data — and existing queries — usability is impaired and reliability put at risk. Ironically, the approach most prone to this disadvantage is the classical triple-based approach: n-ary relations.

8.1 Semantics

There are some fundamental differences between the paradigms under which RDF and SPARQL operate, leading to considerable tensions and frictions in practice.

Querying is necessarily a very application-focused activity, quite different from the focus on knowledge representation with a special knack for integration that drives RDF. Not surprisingly SPARQL has no model-theoretic semantics and operates under the closed world assumption. [AU17] discuss weakly monotonic queries as an approach to bridge the gap between RDF's open world semantics and SPARQL's closed world paradigm.

The RDF semantics is couched in set theory as is customary for model theories. Many practical applications however require bag semantics where multiple occurrences of the same triple are understood as different entities. The resulting tensions and mismatches have been discussed above e.g. as 'counting semantics' and have inspired some works to investigate how to bridge that gap. [AG16] investigate the tensions between RDF's set based semantics and SPARQL's multiset semantics and find parallels to well known problems in the field of relational databases. They claim that "[t]he incorporation of multisets (also called 'duplicates' or 'bags') into the semantics of query languages like SQL or SPARQL is essentially due to practical concerns: duplicate elimination is expensive and duplicates might be required for some applications, e.g. for aggregation." They discover that "the core fragment of SPARQL patterns matches precisely the multiset semantics of Datalog" and that "in SPARQL there is a coherent body of multiset operators." In conclusion they observe that their "study shows the complexities and challenges that the introduction of multisets brings to query languages, exemplified here in the case of SPARQL."

8.2 Meta-Modelling

[CF07] propose to model context hierarchies as named graphs that are organized via a special `:subStateOf` property. To make querying those hierarchies easier they extend SPARQL with a special `STATE` keyword, doing away with the need for cumbersome combinations of `FROM` and `FROM NAMED` clauses.

[Div+09] introduces an approach to meta-modelling called RDF$^+$, see 4.2.7 above. By separating the data and metadata (and extending SPARQL to support their representation format) they achieve that "one may request meta knowledge without modifying the query proper. Thus, our approach achieves highly flexible and automatically coordinated querying for data and meta knowledge, while completely separating the two areas of concern."

[Zim+12], as introduced above in Section 3.1.1.10, present an extension to SPARQL that 'enables conjunctive queries over annotated statements selected by annotation values, even by combinations of values from different annotation domains.' The syntactic representation is based on RDF standard reification, unlike the more elaborate approach of [Div+09] above, their approach is remarkable because of the breadth of problems it solves: it does not only cover a wide range of annotation domains, but also provides an extension to the RDFS inference rules and enables operations over combinations of annotation domains.

[Hof+13] describe an extension of the RDF data model with triple identifiers for the purpose of annotating statements with temporal and spatial validity and other contextualization. An extension to SPARQL allows to query for those annotations without requiring the user to formulate involved triple patterns. Additionally an implementation via 7-column tables in an RDBMS allows to improve join performance.

[Das+14] explore the modelling of LPG in RDF via singleton properties, named graphs and RDF standard reification and compare queries on nodes, edges, aggregates and paths. Standard reification performs worst while singleton properties and named graphs show largely similar performance.

[Ism+18] compare the knowledge graphs of Wikidata and DBpedia from the standpoint of the latter. DBpedia uses a different approach than Wikidata to represent annotations and the work discusses the respective advantages for querying over simple and reified statements, including aspects like transitive path queries.

[Sch+20] present an approach to analyse and query contextualized data in a way similar to Online Analytical Processing (OLAP) systems: information from related contexts is merged and entities are replaced by more general entities, generating "a more abstract, high-level view — a management summary — of the knowledge." The implementation extends the CKR framework, see 9.3.3.2.

[Las+] discusses problems with updates and deletes on annotated statements, e.g. if deleting an statement also deletes all its annotations or if an annotation on a statement with multiple occurrences should add the annotation to all those occurrences. "The common root cause for the ambiguity of all these scenarios is that we request updates over a simplified, dimensionally reduced view of the data.

[. . .] the semantics of such operations needs to be carefully designed in order to avoid unwanted side effects, as to obtain a 'natural' behavior in scenarios where data is queried and manipulated via different query languages."

Proposed extensions to SPARQL for the purpose of querying temporal and spatial annotations are largely out of scope of this work and only briefly mentioned here for the interested reader to further investigate: [KK10] has been introduced in Section 3. [KKK12] report on the implementation of the approach. [BK12] and [CH22] present GeoSPARQL and GeoSPARQL 1.1 respectively, the de-facto standards for querying geospatial data on the Semantic Web.

8.3 Usability

[WM18] present a survey of SPARQL users to complement other studies of SPARQL query logs w.r.t. to usage patterns, perceived usability and feature requests. Notably, and in accordance with earlier studies, it was found that many queries were composed of just one to three triples, suggesting a usability benefit of special meta-modelling primitives as opposed to e.g. involved n-ary relations.

[Das21] propose a quin solution, named RDFn,[1] in which every statement gets an identifier that is automatically derived from subject, predicate, object and graph, or manually created in case a statement is annotated multiple times (multi-edge). The author stresses that this approach provides "complete elimination of any need to rewrite pre-existing queries upon addition of new data. The resulting cost avoidance makes RDFn more cost-effective for use in enterprise settings." [GP13] investigate a similar aspect when they compare various approaches by their potential for *polymorphism*.

8.4 Applications

[Sah+09] propose query optimization through materialised provenance views. The generation of those views is controlled by their Provenir ontology schema.

[Kuh+16] argue that the named graph based Nanopublication approach[2] allows to reduce the load on SPARQL servers and therefore helps improve uptime of public SPARQL endpoints, a well-known problem and obstacle to the realization of the vision of a fully decentralized Semantic Web.

[DMM19] provide a very detailed treatment of modelling and querying lists in RDF in theory and practice. They present the vocabularies provided by RDF as well as ontology patterns to model lists, discuss their support in SPARQL and generate benchmarks over three popular triple stores.

[1]See 4.2.7.2 above

[2]See Section 4.2.5.4

8.4.1 Join Performance

RDF query performance suffers from the highly normalized structure of its data. Better support for, or rather exploitation of, object boundaries and shapes could help mitigate this problem. If larger parts of the data were well structured and those structures were easily recognisable by a query engine, optimizations could better handle complexities and order joins appropriately.

[HHK15], presented above in Section 3.2.2, find that different approaches to representing attributed statements may require different numbers of joins but those do not necessarily effect query performance in a negative way. Through measuring query performance in an experimental setup with different representations and databases they find that "[a]lthough [RDF standard reification, n-ary relations and singleton properties] would typically require more joins to be executed than named graphs, the joins in question are through the statement identifier, which is highly selective; assuming 'equivalent' query plans, the increased joins are unlikely to overtly affect performance, particularly when forming part of a more complex query. However, additional joins do complicate query planning, where aspects of different data models may affect established techniques for query optimization differently. In general, we speculate that in cases where a particular model was much slower for a particular query and engine, that the engine in question selected a (comparatively) worse query plan." [FH17] come to a similar conclusion. [Das+14] report that named graphs perform better than singleton properties when accessing edge key/value pairs because in such cases they require fewer joins.

8.4.2 Path Queries

[HHK15] point to a problem with path queries in RDF. When comparing different approaches to model Wikidata qualified relations in RDF they find that "property paths cannot be expressed in either the singleton properties model or the RDF standard reification model; they can only be expressed in the n-ary relation model [...], and the named graph model [...] assuming the default graph can be set as the union of all graphs (since one cannot do property paths across graphs in SPARQL, only within graphs)."

[Krö17] when analysing Wikidata query logs observes that a significant part of queries "are using regular path expressions: their share of user queries amounts to over 30% in several months, making this the most frequently used SPARQL feature after basic JOIN and SELECT, well before UNION and OPTIONAL (ranked at about 5-10%). This confirms the graph-based nature of Wikidata."

[WM18] in a survey of SPARQL users discusses users' demand for and usage of property path queries and suggests support for more features than SPARQL 1.1 offers.

Chapter 9

Reasoning

This chapter aims at a cursory overview of the topic of reasoning with attributed, annotated or contextualized relations. It provides a short introduction into the history of reasoning with formalisms that extend the simplistic concept of binary relations underlying RDF, and some pointers to lines of work and recent proposals in the area of the Semantic Web.

While the field of Artificial Intelligence started in the late nineteen fifties, only in the eighties of the last century the notion of context became a focus of interest. By then the range of information that AI systems were able to cover had expanded to sizes and degrees of complexity that asked for some structuring device to separate that which is of specific interest from that which is only of circumstantial importance.

John McCarthy's *Notes on formalizing Context*[1] were an important contribution to get a better grip on what context is exactly and how the notion may be formalized and incorporated into existing logic systems and made computable. His central proposal was the notion of a context in which a statement is true and accordingly the introduction of an `ist` operator — for *is true in* — to state such an arrangement. Other works expanded on this idea, e.g. [MB+97], [Hay97] and [GM03b], and also implementations followed, first and perhaps most notably by a PhD student of McCarthy, R.V. Guha, in the CYC system, there called microtheories. A second strand of conceptualizations of context was initiated in Italy by Fausto Giunchiglia in his work *Contextual reasoning*[2], developing its own line of thought and implementations, e.g. the Contextualized Knowledge Repository (CKR). [SB04] provide an in-depth comparison of the two approaches by McCarthy and Giunchiglia.

As that short history of context in AI may already suggest, the concept of context is quite elusive and hard to pin down. This can lead to formalisms that are not easy to understand and get right in practice. The ensuing complexity also makes contextual reasoning computationally demanding or even impractical

[1] McCa93
[2] [Giu93]

in larger environments. Consequently, although a contextual formalism and reasoning facility on a decentralized and distributed information system like the Semantic Web might obviously seem useful and desirable, all but very modest and well-designed attempts at contextualization might not scale to the requirements of its unprecedented size. Works by e.g. [GMF04], [JM10] and [BSH16] purposefully concentrate on certain aspects of contextualization like relating ontologies, statement types or individual terms. But computability is only one side of the problem. A shared understanding of what context is precisely will be indispensable as well to ensure that a context formalism provides useful results.

Another line of thinking tries to avoid the semantic intricacies and computational complexity of contextualization and rather follows a seemingly less involved concept of annotation. Such approaches often concentrate on individual statements — instead of contextualizing groups of statements, or whole 'theories' as in CYC — and aim for annotation domains with favourable computational properties, especially certain well-formedness characteristics and guarantees. Or, as [ZG17] puts it: "the approach of annotated logic programming [KS92] considers that a contextual annotation is just a value in an algebraic structure (e.g., a number or a temporal interval)." [URS06] and [URS10] provided important input, on which e.g. [Div+09] expanded, with [Zim+12] proposing a unified solution that integrates different annotation domains as long as they are reasonably (no pun intended) ordered. However, the requirement of some kind of order still rules out some arbitrary dimensions of contextuality as they are all too common on the open web, e.g. a strictly hierarchically organized conceptualization of topics or categories can be incorporated in such an annotation formalism, but a chaotic tag cloud with all kinds of interrelations — a normal graph, in other words — can not.

Approaches concentrating on term equivalence e.g. expressed via `owl:sameAs` are the obvious complement to contextualization of graphs and annotation of statements. However, proposals like those made by [WF06] and [ZG17] push the envelope and contextualize the smallest possible entity in RDF, the terms themselves. This has quite obvious pros and cons: on the upside it doesn't require any changes or extensions to model, syntax and semantics of RDF, but on the other hand it leads to rather verbose constructs that would need some support by interfaces or syntactic sugar to be bearable.

9.1 Graphs, Triples & Terms

As hinted above most works on contextualization concentrate on sets of triples, or graphs, sometimes also called theories or ontologies. Annotation-oriented approaches target statements. Terms are approached under the heading of the `owl:sameAs` problem or as fluents. However, as has been discussed already above with respect to representation formats under the moniker of *trickle down* *semantics*,[3] in the end it makes no difference in meaning: annotating a graph annotates all the statements it contains and all terms used in those statements as well.

[Sow03] provides a detailed account of "how any [nested graph model] NGM with a finite depth of nesting can be flattened to a Tarski-style model consisting of nothing but a set D of individuals and a set R of relations over D. Although the process of flattening shows that modalities can be represented in first-order logic, the flattening comes at the expense of adding extra arguments to each relation to indicate every context in which it is nested."

One has to be exact though: if the annotation states that the graph has a certain provenance then that doesn't mean that any term in any statement in that graph has that same provenance. Rather their usage in that graph has that provenance. However, if one is really correct, one has to realise that the graph itself is an abstract object in the semantics of RDF and therefore a provenance annotation on the graph is kind of an annotation on a usage itself. And then everything corresponds nicely and harmonically. That distinction between the abstract object — term, statement, graph — that means the same everywhere and its occurrence in certain circumstances is indeed a tricky subject.

Triple-icity Triples are the basic and dominant modelling primitive provided by RDF. RDF standard reification and named graphs and singleton properties on the other hand have no standardized model-theoretic semantics and are not supported by RDFS and OWL reasoning. It seems therefore that triple based representations would be the obvious modelling approach when the aim is to exploit the description logic based stack of Semantic Web reasoning machinery.

[WF06] use time-annotated reified triples as an example to describe the limited usefulness of reification in combination with OWL based reasoning. As reified relations are treated as classes, OWL's capabilities to describe them is limited. Some well-used and impactful OWL operators like inverse, transitive, symmetric, functional and inverse-functional can't be used on reified relations and consequently a lot of very basic and useful entailments are lost. It is also impossible to state restrictions on roles in relation to their time-annotation, e.g. to state that a person can have at most one manager at any given time.

However, the same is true for n-ary relations. In practice strictly triple-oriented approaches get very complicated very quickly when the complexity of the underlying data increases. OWL-conformant design patterns are not for the faint of heart, but require expert knowledge and training to master — scarce resources within the target audience of web developers and web users. So far no easy route to reasoning on annotated, contextualized, deeply nested metadata has been discovered.

Singleton properties would need special handling of the singleton property to ensure that OWL operators on relations work as desired. Ways out of this dilemma might be approaches to accompany n-ary constructs with shortcut

[3]See Section 4.2

relations as described e.g. in [Car+19] above, or attributed relations that encode the most central information in simple statement that is qualified by annotations providing further detail. Triple-centric reasoning approaches could then at least be applied to the core aspects of the data.

The way forward will probably not involve new breakthrough discoveries in logic, but rather some clever divide and conquer strategies based on new combinations of what's already there.

9.2 Surveys

As hinted at in the beginning this work cannot give a comprehensive survey of the field of reasoning with annotated and contextualized data. In the following a few works are presented that provide overviews.

[ZG17] give the following introduction: "The problem of being able to reason not only with logical formulas, but also about said formulas, is an old one in artificial intelligence. McCarthy [McC87] proposed to extend first order logic by reifying context and formulas to introduce a binary predicate ist(ϕ, c) satisfied if the formula ϕ is true (ist) in the context c. However, a complete axiomatization and calculus for McCarthy's contextual logic has never been formalized. Giunchiglia [Giu93] proposed the grouping of 'local' formulas in contexts, and then using other kinds of formulas to characterize how knowledge from multiple contexts is compatible. This idea of locality+compatibility [GS98] has led to several non standard formalisms for reasoning with multiple contexts [Zim13]."

As already noted above [SB04] provide a detailed treatment of the differences between these two dominant lines of approaches on contextualization, inspired by McCarthy and Giunchiglia respectively, and discusses intuitions as well as formalisms.

[Zim13] provides an overview of logical formalisms to reason over heterogeneous knowledge. He defines a concept of networked ontologies where logical systems are consistent locally but not necessarily among each other. Different proposals to manage the latter case are presented. Distributed Description Logics (DDL) [BS03] allow to define cross-context formulas, so-called bridge rules, to relate different terminologies. Cross-ontology inferencing is possible. As the formalism is based on Description Logics, it integrates well into the Semantic Web machinery. Package-Based Description Logics (P-DL) [BCH06] target the use case where one ontology needs to import another one in a well-controlled manner — see 9.3.4.3 below for more detail. ε-Connections [Kut+04] link ontologies, in principal similar to DDL, but "[i]nstead of expressing correspondences of ontological terms, an ontology can connect to another by using special terms (called links) which can be combined in conjunction with terms from another ontology. The semantics of links is very similar to the semantics of roles in Description Logics, except that instead of relating elements from the same domain of interpretation, they relate two different domains." The formalisms presented so far usually define cross-ontology relations from the point of view of one ontology. Integrated Distributed Description Logics (IDDL) [Zim07] in contrast to that defines correspondences from an outside perspective, external to the ontologies is connects. “One consequence of this approach is that correspondences can be manipulated and reasoned about independently of the ontologies, allowing operations like inversing or composing ontology alignments, as first class objects.” The author mentions other related formalisms. McCarthy’s approach is deemed unsuitable for his use case as it mixes knowledge about contexts with knowledge inside a context. An approach in the ‘Italian’ tradition by [SH12] “define[s] a contextual logic where cross-context knowledge does not take the form of correspondences between entities in different contexts, but expresses relationships between the contexts themselves” — an interesting parallel to the problems of identification discussed in other parts of this work. However, the work might be seen as an extension to the approaches above, simply bringing structure to alignments. [GMF04], which characterizes itself as a rather modest approach to contextualization in contrast to the elaborate formalisms employed in CYC, is portrayed as not separating the knowledge of different contexts, but instead ‘reifying’ context “such that multiple contexts can be described within the same knowledge base, and context itself can be described.” Finally a short overview over approaches to reason with inconsistent knowledge bases is given. However, they find that none of the proposed formalisms would allow treating a local knowledge base as sound while only considering and handling inconsistencies in, between and in relation to external sources.

[Zim+09] compare the aforementioned and more approaches at hand per the following criteria:

Context Awareness ability to identify and use context

Modularity ability to modularize ontologies (reusing multiple ontologies or parts of ontologies)

Profile/Policy Management ability to model internal policies or profiles distinctively

Correspondence Expressiveness ability to relate heterogeneous knowledge, either within or between contexts (in particular via more or less expressive ontology alignments)

Robustness to Heterogeneity i.e., ability to tolerate incompatibilities or inconsistencies within or between contexts

[BHS12] present a comprehensive list of criteria that a contextualization formalism should fulfil:

Encapsulation Knowledge that shares the same context should be encapsulated, easily identified and accessed

Explicit meta knowledge Knowledge about contexts should be explicitly represented in a logical language

Separation of meta knowledge and object knowledge Meta knowledge is clearly distinguished from object knowledge

Relations between contexts If one context is related to another, there should be a way how to represent this within the framework

Contextual reasoning Reasoning should take into the account as well the contextual meta knowledge and relations between contexts

Locality of knowledge In each context we should be able to state axioms with local effect that do not affect other contexts, and are not affected by other contexts

Knowledge lifting If needed, specific knowledge can be lifted from one context and reused by another

Overlapping and varying domains Objects can be present in multiple contexts, but not necessarily in all contexts

Complexity invariance The contextual layer should not increase the complexity of reasoning

They apply those criteria for a short but useful comparison of approaches by [URS10], [Str+10], [GMF04], [Bao+10b], [Tra+08], [KG11] and [SH12].

[ADP15] compares a large set of works by [Ana+13], [Bao+10b], [Ber+07], [Car+05a], [Del+08], [GL05], [GMF04], [JS11], [KT05], [Kly01] and [MK03] according to their support for the following aspects, features or dimensions:

- context,
- foreign contextual URIs,
- linking of contexts,
- nested RDF triples,
- metadata,
- partial importing of knowledge between contexts,
- selective transitivity of subClassOf and subPropertyOf relations between contexts,
- extension of the RDFS 1.1 model theory,
- extension of the SPARQL 1.0 query language,
- entailment between contextual structures,
- merging of contexts and subcontexts.

[ZG17] develop an interesting categorization of the problem space, different from the one employed in this work: they divide approaches into those that extend the data model, like named graphs, RDF+, RDF* and YAGO and, on the other hand, those that modify statements, like RDF standard reification, n-ary relations, singleton properties and NdFluents. Only approaches from the second category are compared in this work, considering how well they preserve entailments in the process of contextualization. The following entailment preservation properties are identified as desirable:

Soundness A contextualization is sound if, when contextualizing a consistent ontology, the result is also consistent.

Inconsistency preservation Inconsistency preservation means that a self-contradictory ontology in a given context is contextualized into an inconsistent ontology, such that bringing additional knowledge from other contexts would result in no more consistency. If something is inconsistent within a context, then it is not really worth to consider reasoning with this annotated ontology.

Entailment preservation A contextualization is entailment preserving if all the knowledge that could be inferred from the original ontology can also be inferred, in the same context, in the contextualized ontology.

[GZM17] in a related work compare different representations — RDF standard reification, n-ary relations, singleton properties and the NdFluents approach they present — and analyse which rules from the pD* fragment of OWL ter Horst[4] are preserved. For each representation format and rule they provide results to the following criteria:

- rule preservation,
- non-contextual rule preservation,
- risk of undesirable inferences.

Temporality Temporality is a dimension of contextualization often investigated in its own right. [AP08] presents a survey on models and query languages for temporally annotated RDF. [GP13] compare different design patterns to represent n-ary relations, their example including a temporal component. [Kri14] provides a comparison of seven approaches for the annotation of time-dependent factual knowledge in RDF and OWL. [GT17] investigates the representation of temporally restricted relations in OWL using only triples, disambiguating 'simple', 'standard' and 'philosophically-laden' patterns. [Bou+17] investigate the representation of changes in OWL 2 and provide an overview over related work from [WF06] to [BSK13].

9.3 Approaches

9.3.1 Identification

One way, although not the most intuitive one, to annotate/contextualize information is to address the smallest unit: the identifier of a concept that participates in a statement, be it as subject of discourse, as relation or as object value. This approach has become quite popular with temporal annotation — as mentioned above [Bou+17] provides an introductory overview — and is there referred to as fluents. It was originally discussed to the field of knowledge representation by [MH69], but [WF06] introduced it to the Semantic Web by a proposal to encode temporal parts of objects. [HJ03] describe the idea as follows: "objects

[4][Hor05]

are composed of so-called temporal parts. When we see an object here and now, we are seeing the parts of it that are now — but there are other parts of it at other times that we might have encountered or might yet encounter." [Bou+17] adds that "While objects are seen as 3D endurants through the endurantist approach, they appear otherwise as perdurantist worms, i.e., four dimensional "space-time worms" whose temporal parts are slices (snapshots) of the worms." This approach is often considered un-intuitive and verbose by laypersons, but liked by experts as being quite exact and unambiguous. [WF06] describe how they employ the approach quite successfully to encode temporal annotations in a way that is accessible to OWL reasoners and therefore providing desirable entailments that other representational approaches can't produce. [Wel10] presents an ontology design pattern based on this approach. [BP11] and [Bat+17] expand on [WF06], the latter introducing "a framework for handling temporal information in ontologies. The proposed framework handles both, time instants and time intervals (and also semi-closed intervals) equally well using a sound and complete inference procedure based on path consistency. Two alternative representations based on the 4D-fluents and the N-ary relations respectively are presented."

[GZM17] present NdFluents to discuss the application of the fluent approach "to any number of dimensions, provide guidelines and design patterns to implement it on actual data, and compare its reasoning power with alternative representations." They provide a very detailed analysis of how far desirable properties of OWL reasoning are preserved with their approach and compare the results to other popular approaches like singleton properties and n-ary relations. [ZG17] extend the NdFluents approach again with the proposal of NdTerms, an approach where "each and every term is contextualized, while in NdFluents only individuals are." As mentioned above they develop a set of criteria regarding desirable entailments and preservation of logic state through contextualization.

9.3.2 Annotation

Works focusing on annotation have already been introduced above: [URS06] opened this line of research in the context of the Semantic Web, [Div+09] and [Str+10] advanced it in several respects, [Zim+12] improved on the aforementioned and represents the most mature approach in this field.

9.3.3 Contextualization

Context and contextualization are one of the main paradigms when it comes to meta-modelling and managing complexity in the data. Some influential works have already been introduced in Section 3.1.2 above. This section gives some background and introduces a few longer running projects. It can't however provide a detailed discussion of the topics and issues that arise when reasoning over contextualized data — for such aspect the reader is encouraged to consult e.g. [Zim13], [ADP15] and [Suc20].

9.3.3.1 CYC

The vision of the Semantic Web has different aspects — some akin to a global e-commerce solution, some more inclined towards the high flying goals of Artificial Intelligence. Around the time the Semantic Web was designed AI had already gone through three decades of theory building and research, trials and errors and a few hype cycles. The then most ambitious project and sort of the culmination of AIs attempt to encode and operationalize common sense knowledge in a computer system was the CYC project [Len98], "the last-ditch Manhattan project of classical artificial intelligence" [Hal12]. This goal proved much harder to achieve than initially thought, and although thousands of man years have been spent in building the CYC ontology it still isn't fully achieved and probably won't be for the foreseeable future. It is nonetheless already useful not only in theory but also in practice as its initiator and primary driver, Doug Lenat, points out[5]. In a keynote to the International Semantic Web Conference 2017[6] Jamie Taylor, a lead Knowledge Graph scientist at Google, alluded to the CYC project again when he sketched the practical benefits of a possible future Knowledge Graph that captures all the common sense knowledge that one takes for granted like that birds can fly and vaccines prevent diseases.

There are personal continuities between CYC and the Semantic Web as well: R.V. Guha had designed and implemented the microtheories system that drives contextualization of knowledge representation in CYC, described in his dissertation [Guh92]. This design was based on theoretical work of his doctor father, John McCarthy, one of the founding fathers of AI. McCarthy had developed a theory of 'context' [McC93] that spurred various attempts of implementation in a scalable and practically usable system, of which Guha's implementation in CYC was probably the most consequential.

Guha later developed MCF and RSS, the precursors to RDF, and then became one of the editors of RDF Schema [BGM14]. He also proposed a strongly scaled down version of CYC's very powerful microtheories system that he deemed more appropriate for the Semantic Web [GMF04].

A close collaborator of McCarthy was Pat Hayes who joined the Semantic Web effort when a first attempt at standardization had hit the wall of incompatible implementations in the late 1990s and it had become clear that a mathematically founded formalization of the model underlying RDF was indispensable. Hayes became the leading author of a model-theoretic semantics for RDF and editor of the respective standard, the RDF Semantics [Hay04b].[7]

CYC with its very ambitious vision and manageable success remains both incentive and warning sign to the Semantic Web community. In AI-related endeavours it's all to easy to fall in the trap of high stakes and frustratingly

[5]`https://www.youtube.com/watch?v=v2rK40bNrrY`

[6]`https://videolectures.net/iswc2017_taylor_applied_semantics/`

[7]On a personal note, this author is heavily indebted to Pat Hayes always being around on several Semantic Web related mailing lists and tirelessly and dependably answering questions on RDF semantics, quirks, limitations and the meaning of it all.

little success. On the other hand goals that are not ambitious enough often fail to spur the kind of enthusiasm that is essential to bootstrap a Semantic Web of unprecedented, namely global, scale.

9.3.3.2 Contextualized Knowledge Repository (CKR)

The Contextualized Knowledge Repository (CKR) is a project that has been developed and refined over the course of the last decade. Its concept of contextualization follows the Giunchiglia-line of intuition. [Tam+10] present an implementation of the *context-as-a-box* approach that characterizes their intuitive concept of contextualization and "in which the contextual space is defined by a fixed number of partially-ordered contextual dimensions and the concrete context, containing RDF/OWL knowledge base, is defined by a vector of values the corresponding dimensions take." [HTS10] introduce a first version of CKR, based on OWL. [JS11] focus on RDFS rather than OWL in a system called Contextualized Knowledge Base (CKB). [SH12] implement the CKR based on OWL2/DL. [BGS13] show with the help of an example from the world of football how a contextual system like CKR can enhance expressivity, compactness and reasoning efficiency. [BS13] bring more structure to context organization and faster reasoning, now based on OWL2/RL. [BSC17] describe the steps required for full materialisation. [BSE18] present "a proposal, based on a recent principle for exception handling for inheritance in description logics, that allows CKRs with context dependent defeasible axioms which can be overridden by more specific local knowledge." [Sch+20] find that "[t]he multidimensional and hierarchical nature of context invites comparison with the OLAP cube model from multidimensional data analysis" and extend the CKR framework to support an OLAP-inspired slice-and-dice approach to navigating a contextualized knowledge graph. [BS20] extend the KG-OLAP approach to a distributed setting to overcome performance issues.

9.3.3.3 Nested Contexts

[ADP15] extend RDFS 1.1 model theory to support contexts implemented as named graphs. IRIs within a named graph are contextualized according to the context that the named graph belongs to. Contexts can be nested in contextual structures. They define entailments between contextual structures as an extension of RDFS entailment, with the same complexity as RDFS-entailment between RDF graphs.

[Ber+07] introduce configurable semantics via nested graphs, there called *formulas*[8] and a specific vocabulary to express the desired semantics of a formula. [Arn19] provides a comprehensive discussion of Notation3's logic extension. Formulas can be used to partition the knowledge space and implement fine-grained control over the semantics of each formula, enabling use cases such as managing contradictory information or explainable AI. [AW19] demonstrate

[8]See 4.2.5.3

how N3 formulas and built-ins can be used to implement some of the possible graph semantics that the RDF 1.1 WG worked on. The flexibility that this approach provides is limited only by the underlying reasoning engine: formulas are imported into the 'active', not merely quoted, knowledge base on demand, the importing properties determine their semantics, side by side with other formulas obeying other semantics. The property `log:notIncludes` for example enables default reasoning with non-monotonic behaviour.[9]

9.3.3.4 Other Projects

YAGO, like CKR, is a project that has evolved over several incarnations. Unlike CKR in each incarnation it creates a knowledge graph by importing data from existing sources like WordNet, DBpedia, Wikipedia or Wikidata, but each time experiments with different forms of representation and semantics. [SKW07] introduce the first incarnation, describing it as "based on a logically clean model, which is decidable, extensible, and compatible with RDFS. [...] Each fact is given a fact identifier. As RDFS, the YAGO model considers fact identifiers to be entities as well. This allows us to represent for example that a certain fact was found at a certain URL. For example, suppose that the above fact `(Albert Einstein, bornInYear, 1879)` had the fact identifier `#1`, then the following line would say that this fact was found in Wikipedia: `#1 foundIn http://www.wikipedia.org/Einstein` " However, this approach stays clear of the danger of introducing paradoxes as "YAGO uses fact identifiers, but it does not have built-in relations to make logical assertions about facts (i.e. it does not allow to say that a fact is false). If one relies on the denotation to map a fact identifier to the corresponding fact element in the universe, one can consider fact identifiers as simple individuals. This abandons the syntactic link between a fact identifier and the fact. In return, it opens up the possibility of mapping a YAGO ontology to an OWL ontology under certain conditions." [Hof+13] present YAGO2 which extends RDF's triple model to quins, including space and time as additional dimensions of each statement. It "makes extensive use of reification: every fact (SPO triple) is given an identifier, and this identifier can become the subject or the object of other facts. [...] If the same fact occurs more than once, then YAGO2 will contain it multiple times with different ids. " However, it can only reify facts that are already contained in the knowledge base. [MBS15] report a third version of YAGO that now incorporates Wikipedia entries in multiple languages. [TWS20] present the fourth incarnation of YAGO, switching data sources from Wikipedia and WordNet to Wikidata and schema.org and arriving at a sounder knowledge base that makes OWL-based reasoning feasible. Constraints modelled in SHACL help to ensure logical coherence.

[Nic07] introduces "a formal framework for the social acquisition of ontologies which are constructed dynamically from overhearing the possibly conflicting symbolic interaction of autonomous information sources, and an approach to the

[9] `https://www.w3.org/DesignIssues/N3Logic` provides a gentle introduction into some properties from the `log:` namespace, but the above cited papers are the more authoritative source.

pragmatics of communicated ontological axioms. Technically, the framework is based on distributed variants of description logic for the formal contextualization of statements wrt. their respective provenance, speaker's attitude, addressees, and subjective degree of confidence. Doing so, [the] approach demarcates from the dominating more or less informal approaches to context and provenance representation on the semantic web, and carefully distinguishes between communication attitudes such as public assertion and intention exhibited on the (semantic) web on the one hand, and mental attitudes such as private belief on the other. Furthermore, [the] framework provides formal means for the probabilistic fusion of controversial opinions, and presents a semantics of information publishing acts. [The] approach provides an incomplex and genuinely social approach to knowledge acquisition and representation, and is thus expected to be widely applicable in fields such as the semantic web and social software, and possibly also in other distributed environments such as P2P systems."

[Del+08] aim at scalable reasoning in a Semantic Web search engine application, but need to be able to "confine ontological assertions and reasoning tasks into contexts in order to track provenance of inference results". To that end they develop a reasoning strategy that is "context-dependent and incomplete by design in order to preserve the original meaning of a document." [DTP11] follow up on that effort with a report on "experiences and lessons learned in building a context dependent methodology for large scale web data reasoning." They give a detailed account of conceptual considerations and on that basis illustrate "how to create an ontology repository which can provide the materialization of implicit statements while keeping track of their provenance". Their implementation supports the ter Horst subset of RDFS and OWL.

9.3.4 Composition

9.3.4.1 N-ary Relations

[Hoe10] presents a design pattern for capturing n-ary relations in an OWL 2 DL conformant way. It is inspired by use cases that require to represent social reality, e.g. the concept of roles and other basic legal concept, but is generalizable to any n-ary relation.

[GRV10] "propose a novel encoding scheme enabling class-based metamodeling inside the domain ontology with full reasoning support through standard OWL 2 reasoning systems."

[KW15] discuss the extension of Semantic Web ontologies by *Cartesian* types to represent n-ary relations more naturally, pointing to a long tradition of such proposals dating back to KL-ONE and other works from the pre-Semantic Web era. To that end they investigate the extension of the standard entailment rules for RDFS and OWL as well as necessary adaptations of ontology editors such as the popular Protege offering.

[Art+17] "introduce DLR$^+$, an extension of the n-ary propositionally closed description logic DLR to deal with attribute-labelled tuples (generalising the positional notation), projections of relations, and global and local objectification of relations, able to express inclusion, functional, key, and external uniqueness dependencies. The logic is equipped with both TBox and ABox axioms. [They] show how a simple syntactic restriction on the appearance of projections sharing common attributes in a DLR$^+$ knowledge base makes reasoning in the language decidable with the same computational complexity as DLR. The obtained DLR$^\pm$ n-ary description logic is able to encode more thoroughly conceptual data models such as EER, UML, and ORM." [AF19] argue that talking about a statement, making it the subject of a new statement, can be considered as something different than the relation itself — they call this the "emergence of new entities". Consequently "the relation itself can be seen under two different perspectives: both as a relation and as a concept" and "via label renaming [. . .] a relation can be alternatively viewed as a set of tuples with different labels but same values: this provides the freedom to produce different reifications of a relation."

[CBR18] extend OWL DatatypeProperties with a RoleDomain to attribute statements and implement n-ary relations in OWL by allowing collections of classes to become the Domains of OWL ObjectProperties to model n-ary relations.

9.3.4.2 Prototypes

[CDP16] investigate "a syntax and formal semantics for a language based on prototypes for the purpose of enabling knowledge representation and knowledge sharing on the Web", arguing that such a system has distinctive advantages compared to other representation languages. "Prototypes avoid the distinction between classes and instances and provide a means for object-based data sharing and reuse." Prototype-based inheritance incorporates a notion of context.

9.3.4.3 Import

Identification of some snippet of RDF in RDF does not equal the import of said snippet into the referencing graph: making assertions about an ontology does not import the ontology; annotating a named graph in the default graph of a dataset does not import that named graph into the default graph; describing a statement via RDF standard reification and attributing it does not assert the statement described by the reification quad; addressing, as subject in an RDF statement, a web document that contains some snippet of RDFa or JSON-LD and attributing it does not necessarily attribute that snippet of RDFa data and doesn't make it part of the graph it is addressed in; and so on.

Importing an RDF graph into another graph is the domain of the `owl:imports` property which doesn't provide any advanced features like e.g. importing only certain parts of an ontology as [PP04] criticise.

[BCH06] presents Package-Based Description Logics (P-DL) as an alternative import mechanism that avoids "the complete overlapping of models required by the OWL importing mechanics. P-DL language features are aimed at providing fine-grained modular organization and controllable selective knowledge sharing of an ontology. P-DL syntax adopts a selective 'importing' approach that allows a subset of terms defined in one ontology module to be directly used in another module."

[JM10] propose to "introduce a new RDF predicate, `rdf:imports`, to enable contextualized knowledge reuse among RDF documents as needed in many applications, e.g., linked data." The proposal is an extension of a McCarthy-style contextualization mechanism presented in [Bao+10b]. It allows to "selectively control the transfer and non-transfer of knowledge between contexts, thus can avoid many 'out of context' misuse of knowledge" and "is compatible with existing RDF semantics as an extension of the current specification."

[AW19] discuss strategies to control the semantics under which some graph of RDF data is imported.

9.3.5 Collections

[BFL09] "formalize the intuitive semantics (as suggested by the RDF primer, the RDF Schema and RDF semantics specifications) of [collections in RDF] by two orthogonal extensions of the RDFS model theory (RDFCC for RDF containers and collections, and RDFR for RDF reification)." They also give a set of entailment rules that is sound and complete for the RDFCC and RDFR model theories and has the same complexity as simple RDF entailment.

[Ebe+21] describe problems in axiomatizing trees for use in OWL DL and provide a reusable ontology design pattern.

9.4 Rules in Context

Description logics (DL) are the predominant logic formalism employed on the Semantic Web as witnessed by e.g. the Web Ontology Language (OWL) and its derivates OWL DL, etc. Rule languages on the other hand have a small following too as they are considered more approachable and more readily employable in applications. [SD02], [YK03] and [Bru+05] all present proposals to base the Semantic Web logic on rules, before the decision for DL was made. [Bru+05] and [PH07] provide detailed comparisons of the pros and cons of both approaches, coming to different conclusions however.[10] [Krö17] analyses the shortcomings of rule languages as well as Description Logics when dealing with involved knowledge graphs. Nevertheless interesting and promising works investigating the use of rule based logics to reason about attributed and contextualized data continue to be developed.

[MF08] investigate "the development of hybrid systems that use OWL, [Semantic Web Rule Language] SWRL, and other semantic software, interoperating through a shared space of RDF triples" to record and reason about provenance and other 'semantic metadata'.

[10]See also Section 2.6.1.3 above

[Ana+11] present Extended RDF (ERDF) as a "semantic foundation of rule markup languages", extending RDF with weak and strong negation, as well as derivation rules. They say that "[a] distinctive feature of our theory, which is based on Partial Logic, is that both truth and falsity extensions of properties and classes are considered, allowing for truth value gaps. Our framework supports both closed-world and open-world reasoning through the explicit representation of the particular closed-world assumptions and the ERDF ontological categories of total properties and total classes." [ADA15] present complexity results for the underlying ERDF formalism. [Ana+13] propose a framework named Modular ERDF that extends ERDF and "supports local semantics and different points of view, local closed-world and open-world assumptions, and scoped negation-as-failure."

A whole line of work investigates the deployment of rule languages to reason about attributed relations in Wikidata, gradually extending their reach and expressivity. [Krö17] provides a good starting point into this research effort and [KT16] , [Krö+17] , [MKT17] and [Krö+18] document the progress. Further contributions come from [PM20] and [MP20] .

9.5 Scalability

Reasoning is a computationally very demanding and tricky terrain. The realisation that reasoning complexity is very hard to estimate and can behave quite contrary to intuitions has in the nineteen nineties started the whole field of Description Logics that became the main underpinning of reasoning on the Semantic Web. Surveying the tremendous amount of work in this area is out of scope for this thesis, but for a detailed a treatment of scalable and efficient reasoning algorithms in RDFS and OWL 2 RL that targets Linked Data deployments on the Semantic Web, the interested reader may for example consult [Hog14].

9.6 Outlook

[Suc20] stresses the need for more expressive formalisms to tackle tasks like the unravelling of fake news, modelling of controversies, analysis of processes etc. He provides a tour de force of the history of knowledge representation and reasoning formalisms, existing large knowledge bases and knowledge extraction systems. He then discusses the tensions between what is needed and what we have, stating that we "want the contexts to be first-class citizens, i.e., we want to be able to use a context C in a relation r(x, C),[. . .]. At the same time, we want the decidable properties of logics that do not permit this [. . .]. Thus, we have to find a trade-off between expressivity and tractability of our logic." But, he argues, "today, we know what type of data we have, and what type of questions real systems care about. Thus, we can cherry-pick the characteristics of the existing formalisms that are relevant for today‘s large-scale KBs and applications" and "narrow down the required properties of the desired formalism both from the data-driven side and the logical side. [. . .] [T]he key to the expressiveness- tractability trade-off could be to reduce the reasoning capabilities of the logic inside a context, and to increase the reasoning capabilities about contexts. [. . .] If both reasoning capabilities are kept limited, then one should arrive at a tractable formalism.”

Chapter 10

Implementation

A discussion of possible approaches for complex data representations in RDF would not be complete without a look at implementation requirements and strategies. Naturally this is the domain of databases and query engines. [Sak+18] offer a detailed overview of the technical aspects of RDF storage engine implementations, discussing strategies of storage, indexing, partitioning and encoding. [Ali+22] present a very comprehensive survey of RDF stores and SPARQL query engines. A popular resource to get an overview of the market share of various graph databases offerings for both RDF and LPGs is the *DB-Engines Ranking of Graph DBMS*.[1]

10.1 Performance Comparisons

Several works have compared different approaches to statement attribution and meta-modelling in RDF for performance by measures such as triple count, load time and query performance, sometimes even comparing different approaches on different triple stores. It turns out that results are not always easy to compare. Triple stores can for example behave quite differently depending on indexing strategies: a store that trades performance for low memory consumption and chooses a sparse indexing scheme may perform badly on an approach that creates unusually diverse data in usually quite homogeneous subsets like property values or graph names. An even more intricate aspect are query planners and join optimizations that can have huge and hard to analyse effects on query performance. Last not least the structure and size of the example data may play an important role. Consequently all these measurements have to be closely studied and taken with a grain of salt before jumping to conclusions — not to mention that another important performance factor, usability, is even harder to measure, but equally important and not necessarily aligned with crunchability prowess.

[1] https://db-engines.com/en/ranking/graph+dbms

[GP13] compare several modelling approaches w.r.t. multiple dimensions of usability, but also triple count and suitability to OWL entailment.

[Kri14] compares three approaches to temporal annotations — quins (triples + start and end time), n-ary relations and fluents — by triple count, storage space, load times and query performance on a database specialized for temporal reasoning.

[Fu+15] compare the triple count and query performance of n-ary relations, singleton properties and nanopublications on three databases: Virtuoso, Stardog and Blazegraph.

[HHK15] compare RDF standard reification, n-ary relations, singleton properties and named graphs to represent Wikidata in RDF.[2] Performance of the respective approaches is measured on five different RDF databases: 4store, BlazeGraph, GraphDB, Jena TDB and Virtuoso.

[Her+16] present a comparison similar to [HHK15], again modelling attributed relations as RDF standard reification, n-ary relations, singleton properties or named graphs respectively, but on a slightly different set of databases — in this case Virtuoso, Blazegraph, PostgreSQL and Neo4J — and with a more idiomatic perspective on queries.

[RMH17] describe the integration of data modelled in various styles — basic triples, RDF standard reification, singleton properties, various dialects of n-ary relations and name chaining — into a system based on neo-Davidsonian n-ary relations.[3] They compare triple counts along multiple measures and analyse query characteristics.

[Fre+19b] compare RDF standard reification, n-ary relations, named graphs, RDF*, singleton properties and *Companion Properties*, a variant of singleton properties that they developed to improve join performance of the latter. Experiments are performed on the Virtuoso, Blazegraph and Stardog databases, measuring loading time, storage size, triple count and query performance. Queries were the same as in [Her+16] and performed on a well-sorted variety of example datasets. They compare their results to [Her+16] and provide a detailed discussion of possible reasons for some strong deviations in the performance measurements that they find.

[KFH22] compare query performance on LPG–to–RDF mappings proposed by [Har19] and [Ngu+19] — the former a straightforward mapping from LPG to RDF-star, the latter an evolution of singleton properties. Neither approach proved superior in all aspects. Notably a Property Graph database querying the original LPG didn't perform much better as well.

10.2 Combined Applications

[FH17] present an abstraction layer to provide unified serialisation and query access to different metadata representation approaches. They argue that because

[2]But they dismiss RDF* because it can't disambiguate multiple relations of the same type.
[3]See 2.7.4.1

the various proposed metadata representation approaches have been shown to perform very differently depending on data structure and database it would be beneficial to establish an abstraction layer that facilitates "flexible storage and querying of data and its metadata independent of the applied Metadata Representation Model" (MRM). They define their own data model in JSON which "allows among others to express different granularity and share levels, an easy way of nesting metadata and the definition of logical metadata groups." "Once the source dataset is converted into the JSON representation, this intermediate format can be used to create NQuads files for the various MRMs." An accompanying tool, called meta-SPARQL, "allows automatic rewriting of SPARQL queries for different MRMs. The idea is, to replace every triple pattern within a SPARQL query by a set of special annotations, which will be translated by meta-SPARQL into the appropriate format. Every query needs to be written as a template in an intermediate SPARQL dialect based on these annotations. [. . .] The template can be converted into query instances of the various MRMs."

[Gim+18] publish the Never-Ending Language Learning (NELL) dataset[4] as Linked Data. The original dataset contains rich metadata documenting its origins, but because NELL is only available in an ad hoc CSV format the data is difficult to use outside the NELL system. The authors therefore convert the dataset according to five well-known reification models, namely RDF standard reification, n-ary relation, named graphs, singleton properties, and NdFluents, as well as to a dataset without annotations, and make those converted datasets publicly available.[5]

Some works like e.g. [Das+14], [Vrg+21], [Ang+22] and [Las+] discuss ways and obstacles to representing both RDF and LPG in the same database.

10.3 Strategies

This section doesn't aim at giving a complete picture of implementation strategies but concentrates on a few aspects that are particularly interesting.

10.3.1 Indexing

Proper indexing plays always an essential part in efforts to enhance the performance of a database. It is especially important with the highly normalized data structure of RDF. "[I]ndexes for different collation orders are crucial for the performance of triple stores. They determine which join orders are possible and which triple patterns are cheap to resolve.[... But] there is always a trade-off

[4]The authors describe NELL as "a system that continuously reads the Web to extract knowledge in the form of entities and relations between them. It has been running since January 2010 and extracted over 450 million candidate statements, 28 million of which remain in iteration 1100. NELL's generated data comprises all the candidate statements, together with detailed metadata information about how it was generated".

[5]`http://w3id.org/nellrdf`

between querying speed on the one hand and memory consumption and maintenance cost on the other hand."[6] Indexing is discussed in many works in more or less detail. [Sak+18], already mentioned in the introduction to this chapter, puts it in context with other implementation specific aspects.

[Big+20] provide a nice overview over indexing strategies implemented in popular RDF stores, from fully indexed triple stores like RDF-3X over quad stores providing specific subsets of indexes, e.g. Virtuoso and GraphDB, to solutions like RDFox that augment such column oriented indexes with adjacency lists to improve path traversal.

[Fer+18] provide a similar outline, slightly more focused on quads. They come from another strand of work that concentrates on compact self-indexed and queryable serialisations — omitting the create, update and delete functionality that full database systems provide — and present the first quad-based serialisation, dubbed HDTQ (Header-Dictionary-Triples Quads), implementing SPARQL/RDF 1.1 datasets.

10.3.2 Statement Identification

Statement identifiers as syntactic sugar in RDF/XML[7] have been one of the first attempts at equipping meta-modelling in RDF with acceptable usability. Some RDF databases like e.g. AllegroGraph and Stardog implement and expose them through proprietary APIs alongside named graph support, effectively providing a quin store. By and large however meta-modelling approaches focused on graphs as a means to annotate groups of triples, as only that approach seemed to provide sufficient scalability. More recently, as singleton properties and RDF-star approaches re-focused meta-modelling to individual triples, statement identifiers are discussed again at least as internal representation of statements.

Approaches to statement identification can roughly be grouped into two categories. The more straightforward approaches add an identifier to a triple and let annotations on the triple, via the identifier, take it from there. Others don't want to give up on the performance benefits and administrative semantics of grouping statements in named graphs and add statement identifiers as a fifth element to quad stores, effectively extending them to quins.

10.3.2.1 Quads

Implementation efforts have more recently increasingly turned to simple Quad-models where each entry consists of a subject-predicate-object triple and an accompanying identifier — the statement identifier replacing the named graph identifier that in the past one-and-a-half decades regularly occupied the fourth position if the database went beyond pure triples (which most RDF databases did). The most important drivers of this development were the increasing commercial success of LPG which generally do not support named graph-like

[6] [Big+20]
[7] See Section 6.1

partitioning of datasets and Wikidata's Property Graph-style data model that heavily relies on edge annotations.

[Ili+20] developed a toolkit to "represent, create, transform, enhance and analyse" Knowledge Graphs (KG). Because of the huge size of KGs like Wikidata classic RDF databases come to their limits and incur high costs in terms of processing time and hardware requirements. The proposed toolkit instead serializes RDF data into a file format that can be processed with popular data science tools, and sometimes even resorts to highly optimized UNIX utilities, making it possible to work efficiently with massive KGs like Wikidata on a standard laptop. Their implementation is not constrained to quads: if the use case suggests it, more columns can be added to capture additional values, eschewing the need for joins on very regular annotation dimensions. Datatypes are addressed too.

[Vrg+21] present a new graph database that is targeting RDF as well as LPG and Wikidata. They motivate the lack of support for named graphs as follows: "Named graphs were proposed to represent multiple RDF graphs for publishing and querying. SPARQL thus uses syntax — such as FROM, FROM NAMED, GRAPH — that is unintuitive when querying singleton named graphs representing higher-arity relations. Moreover, SPARQL does not support querying paths that span named graphs; in order to support path queries over singleton named graphs, all edges would need to be duplicated (virtually or physically) into a single graph." They hint at possible support for named graphs via reserved relations and do not rule out proper support via quins in future development.

[Las+] describe the Amazon Neptune cloud graph database offering which supports both RDF and LPG. To support both approaches the internal representation is again based on triple statements and statement identifiers. It is however not yet possible, but a goal of the project, to support switching between the two formalisms within one application and without having to convert the data first.

[Ang+22], in an article co-authored by some of the authors of the aforementioned works, argue for the "need for a unifying data model that can support popular graph formats such as RDF, RDF* and property graphs, while at the same time being powerful enough to naturally store information from complex knowledge graphs, such as Wikidata, without the need for a complex reification scheme." To that end they propose *multilayer graphs* as an 'intermediate solution': "an abstract graph model that extends directed labeled graphs with edge ids. This model largely removes the need for reification when modeling complex data, and yet adds minimal complexity versus directed labeled graphs." The authors provide two formal definitions of multilayer graphs, as function and as relation, the core feature being "to reference an edge as it if were a node (or possibly multiple nodes). A multilayer graph captures this feature through edge ids." However, no model-theoretic definition is given.

10.3.2.2 Quins

[Div+09] employ statement identifiers internally as an optimization strategy in implementation but don't provide them in serialisations. Consequently certain annotations are only available internally but get lost when publishing the data on the Semantic Web. [Knu21] discusses IRI-encoded statements as a low-cost implementation strategy for RDF-star but in other comments also hints at statement IDs as a possible approach. [Fre+19b] report on statement identifiers in the Stardog triple store[8] which supports statement annotations and is queryable via a proprietary SPARQL extension, taking an approach very similar to AllegroGraph. The statement identifiers also support Stardog's implementation of Property Graphs. As [Das21] reports a similar quin approach is also taken by Oracle to support both RDF and LPG. [Hof+13] provides statement identifiers besides other extensions to model dimensions like time, space and context as discussed above.

10.3.2.3 Quads & Quins Combined

[Kuh+18] walk on the edge between quads and quins: they use named graphs extensively, even for storing (and annotating) single triples and for (virtually) nesting graphs within graphs, ignoring the conventional wisdom that named graphs should rather be used for grouping triples for the sake of administrative purposes than for real meta-modelling. They provide a discussion of the performance aspects of this elaborate use of named graphs.

[Las+] suggest the possibility to extend the basic and straightforward Triple+ID model with grouping features on demand, evading the need for hard choices and arguing for an overall more flexible approach that can cater to the needs of applications while preserving clear semantics.

[Fer+18] present an extension of HDT (Header-Dictionary-Triples), a "compressed representation of RDF data that supports retrieval features without prior decompression" to HDTQ, representing not triples but quads. They provide two versions: in the case of annotated triples a bitmap is assigned to each triple, pointing to the graphs in which the triple occurs, whereas in the case of annotated graphs a bitmap is assigned to each graph, marking the triples present in the corresponding graph.

10.3.3 Miscellaneous

As hinted above this work can't provide a comprehensive overview of implementation strategies. However, a few aspects should be mentioned to at least give a vague idea about the size and shape of the terrain.

BigData & NoSQL Many works investigate in how far recent advances in database technology dubbed BigData and NoSQL databases can be exploited for

[8]`https://www.stardog.com/`

the Semantic Web — e.g. [Che+13] present an implementation of provenance-annotated named graphs in HBase, [Sch+15] provide a mapping from RDF data and SPARQL to the property graph model of Spark/GraphX and its query engine, to name just two of those works, targeting the prominent BigData databases HBase and Spark.

Querying & Decentralization The vision of a decentralized cloud of Linked Open Data facilitated through publicly accessible SPARQL endpoints is jeopardized by the enormous strain that involved queries can impose on servers and, sometimes as a result of such overtax, the low availability of servers. As a result strategies have been developed that try to better balance the load between servers and clients by serving larger chunks of data, possibly cached or pre-computed, that clients then post-process to arrive at the final result. [Tae+18] note that "Linked Data is increasingly published through different Web interfaces, such as data dumps, Linked Data documents, SPARQL endpoints and Triple Pattern Fragments (TPF) interfaces. This has led to entirely different query evaluation strategies, such as server-side, link-traversal-based, shared client-server query processing, and client-side (by downloading data dumps and loading them locally)." [Pol+20] find that "[i]t seems that in order to avoid both such discrepancies and bottlenecks for downloads and query processing, a combination of (i) dumps provided in HDT, a compressed and queryable RDF format, as well as (ii) Triple Pattern Fragments (TPF) endpoints as the standard access method for Linked Datasets could alleviate some of these problems: the triple-patterns fragment interface essentially limits queries to an endpoint to simple triple format that allows such triple pattern queries without decompression and also guarantees duplicate-freeness."

Old School & New School [Hof+13] extend the RDF data model with further dimensions that they find general enough to record for every statement, namely time, location and context. They implement that model on PostgreSQL, a well-proven relational database system, as a single table of 7-tuples: subject S, predicate P, object O, time T, location L, context C, tuple identifier id. Indexing follows proven strategies for the SPO columns plus "additional indexes specifically suited to the respective data type" in columns T, L and C.

[Big+20] on the other hand pursue a rather exotic path with a storage solution based on order-3 tensors. Tensors, or matrices, are sparsely populated data structures, here implemented as a novel approach called hyper-trie. The presented implementation, named Tentris, does not use multiple indexes like almost all other triple stores but "relies on the novel hypertrie tensor data structure that unifies multiple indexes into a single data structure." It is an in-memory store, which somehow impedes its applicability to very large knowledge graphs. In benchmark testing it performs very well against established triple stores Blazegraph, Fuseki, GraphDB, gStore and Virtuoso. Currently Tentris doesn't support named graphs, but discusses the addition of statement id's.

Chapter 11

Graph Models

Alternative graph models for RDF, although an important discussion of a low level aspect with possibly wide-ranging consequences, can only be presented in passing here. However [AG08a] give a detailed overview over the state of the art in graph database models around the time that RDF was conceived and first standardized. More recently [MH19] recap the history of graph models in RDF (and informed the following categorization).

11.1 Labelled Directed Graph

The graph model of RDF is based on sets of triples. The conceptual model is designed as a labelled directed graph which can straightforwardly be implemented in relational database management systems. This is indeed the predominant approach in the Semantic Web software ecosystem but, as [MH19] point out, the RDF graph model differs from the graph theory definition of graphs in a number of ways and therefore can't easily take advantage from advances and techniques developed in graph theory and implementations for typical applications such as graph traversal and shortest path.

[Ngu+16] propose a modified graph model for RDF, a Labeled Directed Multigraph with Triple Nodes (LDM-3N). They argue that although properties can be subjects in RDF triples, idiomatic "follow your nose" algorithms will never come across them as they jump from node to node. In the model they propose properties become nodes themselves and RDF triples are decomposed into [subject-propertyNode,propertyNode-object] pairs.

11.2 Bipartite Graph

For that reason [HG04] argue for augmenting the graph model of RDF with an intermediate representation that is better suited to apply methods from graph theory, namely bipartite graphs. They present such a model formally, discuss transformations to and from RDF, explore formalizations of the intuitive notion of "semantic relation" and discuss implementation in the Semantic Web software ecosystem. However [MH19] report of experiments showing that reasoning is not well supported but rather made even more complicated through the use of bipartite graphs.

11.3 Hypergraph

[Mor07] point out that labelled directed graphs do not harmonize well with established graph theory. Instead they propose directed hypergraphs and present preliminary experimental results suggesting that their approach scales better than labelled directed graphs. [Ang09] employ hypergraphs as what they call nested graphs to improve exploration and visualization of RDF data and discuss implementation of nested graphs via plain, flattened representations in graphsets.

11.4 Metagraph

[Che+17] propose metagraphs as an alternative to 'flat' labelled graphs and hypergraphs. Like with most graph models a metagraph consists of vertices and edges connecting those vertices, but additionally it can also address graph fragments, composed of vertices and edges, in so-called *metavertices* and use those in relations just like normal vertices. They show how this facilitates complex tasks like reification and n-ary relations and leads to more natural and intuitive knowledge representations.

Also of interest in this context is [YK02] who early on argued for basing the RDF semantics on F-Logic, an around the time popular vehicle for implementing complex inferences over Semantic Web resources. As F-Logic lacks support for reification they propose a respective model-theoretic extension.

11.5 Layered Graph

[MH19] present a layered graph model optimized for neural learning purposes. RDF graphs and their RDFS inference graphs are encoded in the form of layered, 3D adjacency matrices. This mapping is however irreversible and two non isomorphic graphs may have the same layered directed graph representation. Consequently this model is not suitable or storing and querying RDF data.

Chapter 12

Example Use Cases

Some works introduce quite elaborate examples to motivate and illustrate their approach. Recurring themes are presented and discussed below. Even proposals for a standard use case or example scenario have been brought forward but haven't seen much uptake so far, probably due to the many facets of the problem space. This might be an area worthy of more effort in future work.

12.1 Actors & Roles

A popular example is that of a person holding different positions at different times. In standard RDF this can be modelled in a tree-like structure with intermediary nodes (probably blank nodes) but the resulting structure is cumbersome to author and query. If discriminating detail like the temporal perspective is not present in the original model merging data from different periods may result in nonsensical or misleading information.

[Sch+08] tackles the problem by collecting statements from different contexts in different graphs, annotating those graphs with attributes like time and certainty:

```
G1 { JamesHendler researchTopic SemanticWeb .
     JamesHendler affiliatedWith RensselaerPI  }
G2 { JamesHendler researchTopic Robotics .
     JamesHendler affiliatedWith UnivMaryland .
     RudiStuder researchTopic SemanticWeb .
     RudiStuder affiliatedWith UnivKarlsruhe  }
G3 { G1 mk:source <www.rpi.edu/report.doc> .
     G1 mk:certainty "0.9" .
     G1 mk:timestamp "5/5/2007" }
G4 { G2 mk:source <www.cs.umd.edu/survey.pdf> .
     G2 mk:certainty "0.6" .
     G2 mk:timestamp "6/6/2001" }
```

[Fer+19] present a variation of the theme in which the same actor plays different roles at different times: "The use case is [. . .] an evolving RDF graph with three versions V1, V2 and V3 : the initial version V1 models two students `ex:S1` and `ex:S2` of a course `ex:C1`, whose professor is `ex:P1`. In V2, the `ex:S2` student disappeared in favour of a new student, `ex:S3`. Finally, the former professor `ex:P1` leaves the course to a new professor `ex:P2`, and the former student `ex:S2` reappears also as a professor."

[Zim+12] elaborate how such a use case can be handled successfully by applying proper temporal annotations and the proposed framework.

12.2 Multisets

Another recurring topic is how to handle multiple occurrences of the same type but with different attributes. A popular example is that of relations that most usually occur only once but nonetheless may occur more often: Stephen Grover Cleveland was the 22nd and 24th president of the United States, the only president to serve two non-consecutive terms. Elizabeth Taylor and Richard Burton where married from 1964 to 1974 and then again briefly in 1975, an example used by [Krö17] among others. More mundane examples of the same issue exist such as repeating events that need to be annotated with the circumstance of occurrence as for example discussed by [Las+21]. [Das21] combines an example of multi-edges — "Cleveland was President for two non-consecutive terms" — with one for edges-as-endpoints — "Cleveland's first term was happier than his second term".

12.3 Annotations on Properties

[Gan11] uses as example the so-called *Expedition of the 1000*, the historic event of Garibaldi landing in Sicily in 1861 with 1000 soldiers, and discusses a whole range of quite different approaches to model that not too complex fact.

[Vrg+21] define the following list of desired features:

- Edge type/label: assign a type or label to an edge.
- Node label: assign labels to nodes.
- Edge annotation: assign property-value pairs to an edge.
- Node annotation: assign property-value pairs to a node.
- External annotation: nodes/edges can be annotated without adding new nodes or edges.
- Edge as node: an edge can be referenced as a node (this allows edges to be connected to nodes of the graph).
- Edge as nodes: a single unique edge can be referenced as multiple nodes.
- Nested edge nodes: an edge involving an edge node can itself be referenced as a node, and so on, recursively.
- Graph as node: a graph can be referenced as a node.

Translating the Property Graph inspired wording into RDF speak they require among others the notorious (and notoriously difficult to realise) features of multisets, unasserted assertions, annotations on annotations and annotations on properties.

12.4 Contextualization

[Car+05a] list as example uses: data syndication, restricting information usage, access control, signing RDF graphs, expressing propositional attitudes, scoping assertions and logic, ontology versioning and evolution. They work out in great detail the example of a publishing workflow that handles aspects of trust, provenance tracking and authority guarantees.

[HTS10] develop a detailed example based on a football World Cup scenario, with national leagues etc, featuring:

- a structured domain, information publicly available
- general facts: 11 players per team
- context dependent facts: players change per team per match, players can change positions/roles, teams can play against each other multiple times and with different scores, referees can't have same nationality as team in World Cup
- connections between contexts: players per team don't change per competition

They promote this as an example that others could use as well to further comparability between different approaches.

[Bao+10a] define model-theoretic semantics per context, differentiate between applicable and non-applicable contexts and also relate contexts (in the last line):

```
:sp { :EricCartman :livesIn :SouthPark . }
:sp rdf:label "A South Park Context"^^rdf:string .
:sp a :TelevisionSeries .
:sp context:semantics <http://www.w3.org/TR/rdf-mt/>
:sp1 { :Kenny :livesIn :SouthPark . }
:sp1 context:extend :sp .
```

12.5 Contextualized Equality

[Bre+12] model personae of researchers that need different views on their data depending on use case e.g. 'Chris the Integrative Systems Biologist': "When searching for data he needs to be able to switch between species depending on the kind of experimental (protein and mRNA) data he wants to compare in pathways, i.e. sometimes he will be interested in just mice data, sometimes just human, and at other times the combination of both. However, he always insists that genes and proteins are kept distinct[...] The scientific user personae have very different needs about the notion of equivalence that should be applied between datasets. The users need a simple mechanism by which they can change the operational equivalence applied between datasets." [Bat+14] enforce the need for different notions of equality as follows: "Scientific data is messy. It is stored in multiple datasets, each of which has been created with its own focus. [...] The challenge is identifying when two entries should be considered equivalent to meet specific scientific needs, particularly when these needs change on a per use case basis."

12.6 Evolving Data

[Das20a] stresses the importance of modelling primitives that are unaffected by changes and updates, neither at the schema level nor w.r.t. existing queries — an important criterion in enterprise settings according to his experience. He gives the following example of cumulative updates to a graph, illustrating different attribution styles and requirements:

Vertex, Edge, Vertex-Property John, whose net worth is $1 billion, donated to Top University. Mary, a child of John, got admitted to Top University.

Duplicate Edge John ... donated twice to Top University. ...

Edge-Property John ... donated twice to Top University, in the years 2010 and 2012, respectively. Mary ... got admitted to Top University in 2011.

Edge as Endpoint of an Edge ... Bob suspects that John's 2010 donation helped Mary's admission.

Edge-Property as Endpoint of an Edge John ... donated twice to Top University, in the years 2010 and 2012 (the year 2012 was confirmed by Alex) ...

Vertex-Property as Endpoint of an Edge John, whose net worth is $1 billion (according to Dan), ...

[Las+] discuss problems with updates and deletes of attributed data.

12.7 Time, Space and Provenance

Many works use examples with temporal, spatial or provenance attributions to illustrate their approach. Such examples have the benefit of evoking strong intuitions and being familiar to the reader, however their semantics are often not as simple and clear cut as they may seem. Temporal and spatial annotations may be interpreted as qualifying the time and location that an assertion is made, or is ingested into the system, or is approved, or is claimed to be valid etc. Likewise provenance may refer to the syntactic representation of a statement or to the fact it asserts. If those scenarios are not clearly defined and disambiguated — and they seldom are — such examples are on shaky semantic grounds. The RDF-star (aka RDF*) proposal used for years and in numerous peer-reviewed scientific papers an example about statement provenance where it asserted the provenance of a statement occurrence although the reference so annotated pointed to a statement as an abstract type. This only became apparent in the process of a

W3C Community Group authoring a report on RDF-star[1](and in 2021 renamed it to RDF-star). A semantically correct solution required re-modelling the use case and consequently the proposal lost a good deal of its perceived simplicity and elegance.

12.8 Inadequacy

[Suc20] quotes a Wikipedia article about vaccines — "In February 1998, Andrew Wakefield published a paper in the medical journal The Lancet, which reported on twelve children with developmental disorders. The parents were said to have linked the start of behavioural symptoms to vaccination. The resulting controversy became the biggest science story of 2002. As a result, vaccination rates dropped sharply. In 2011, the BMJ detailed how Wakefield had faked some of the data behind the 1998 Lancet article." — and subsequently shows that the information from this article cannot be represented in any of the well established knowledge representation formalisms, from Frames to Semantic Web and beyond.

[1] `https://w3c.github.io/rdf-star/cg-spec/2021-12-17.html#the-seminal-example`

Part III

SEMANTIC GRAPHS

This part, *Semantic Graphs*, points in two directions: For one it develops different notions of the *problem* space, different points of view, different angles and aspects. The multitude of approaches presented so far have not always reached for the same goals or followed the same intuitions. Their goals and intuitions don't always form a coherent whole; they much rather do overlap, diverge and even contradict each other. It seems necessary to abstract away from the practical, the targeted and the nitty-gritty and develop a better understanding of the defining structural characteristics of the problem space.

While this part tries to provide such a better understanding, it also tries to grasp a glimpse of a possible *solution*, not as a worked out proposal but rather as a perspective on how an actual solution might be found. The stack of technologies and techniques that the Semantic Web employs and has at its disposal is enormous: from ontologies to reasoning, from serialisations to query engines, from modelling patterns to heuristics, from model level extensions to syntactic sugar. These tools can be combined in many different ways, to many different effects. That takes some creativity, an open mind, and some discipline, but the room for experimentation is there, and it is vast...

There is however one major obstacle to such grand visions, and that is the state of semantics on the Semantic Web, or, more precisely, the at-risk state of shared understanding of what it means to have a semantics, what the semantics of what one tries to express is or may be, what the semantics of RDF actually *means*. This problem runs deep and without some work on these fundamentals it will be hard to develop and agree on some fitting representational strategies that help the Semantic Web to express more than just isolated facts, but can represent shared knowledge in faceted and flexible ways.

Chapter 13 on *Schrödinger's Triple* rubs some salt into the wound of the Semantic Web: it is indeed often quite hard to realise and be aware of the informal semantics embedded in the ways facts are woven into knowledge. Discussions about the meaning of certain constructs and arrangements tend to get lost in downward spiralling chatter and innumerable ramifications. Yet: there's no way around this topic if the 'Semantic' part of the 'Semantic Web' endeavour is to be taken seriously.

Chapter 14 on *The Gestalt of it All* aims to shed some light on hidden assumptions, to disambiguate competing aspects and to identify the main fault lines along which intuitions and actual semantics diverge, investigating where they use different techniques to achieve the same result or where they use similar techniques while striving for very different goals. As it turns out, the underlying semantics are sometimes not well understood and many different intuitions are at work, pulling in quite, even contradictory directions, and they do not fit easily under the umbrella of one unifying design. In this chapter semantics is not discussed in terms of a formal model-theoretic account, but in a more comprehensive sense: the meaning of what we aim to express and the abstractions we develop to structure our universe of thoughts and information, transforming facts into knowledge. This is easily the most daring chapter in this work, but probably also the most important one. In the opinion of this author this subject needs much more attention if a sensible solution to the expressive needs of the Semantic Web is to be found.

Chapter 15, *Perspectives*, connects a few dots between low hanging technical fruit, basic usability requirements, the very nature of the Semantic Web endeavour and some well-proven design principles. Section 15.5 then presents a very provisional idea of how a solution might look like. That approach, named *Silhouettes*, depicts rather a stimulus than an actual proposal, combining some of the insights and achievements found in the works surveyed. It investigates how NdTerms, an approach to representation of attributed data that shows rather favourable entailment properties but suffers from usability issues, might be made amenable to the Semantic Web at large. This is not intended to be *the* proposal to solve all problems described so far. Rather it should be understood as an experiment, an investigation into the combination of a rather radical representational approach like term-based annotation, a straightforward annotation approach based on statements and statement-identifiers, some vocabulary and a late-binding approach to disambiguation. A lot more of such dabbling and prototyping will probably be necessary to arrive at a sustainable solution that meets all demands on expressiveness, simplicity and practical viability reasonably well.

Finally, Appendix A on *Abstractions* provides a principled, systematic and exhaustive overview of aspects and criteria that informed and guided the approaches presented in the Part II, either im- or explicitly. However, it trades readability for systemic purity.

Chapter 13

Schrödinger's Triple

[Las+] discuss possible convergence strategies for the two most popular graph technologies applied in the field of Knowledge Representation today, RDF (including the upcoming RDF-star extension) and Labelled Property Graphs (LPG). At hand of a brief example they illustrate the pitfalls of modelling decisions that simplify the data for the sake of 'intuitive' usage, but against the semantics inherent in those constructs. Their example, translated to a Turtle-ish notation,[1] is:

```
:Alice :knows :Bob           id:1 .
:Alice :knows :Bob           id:2 .
id:1 :statedBy :NYTimes      id:3 .
id:2 :statedBy :TheGuardian  id:4 .
id:1 :since 2020             id:5 .
id:2 :since 2021             id:6 .
```

The example illustrates a problem originating in a mismatch between RDF' set based semantics and LPG's instance-focused approach. RDF doesn't distinguish two occurrences of the same statement — in RDF thinking the only thing that counts that it is the same type of statement — whereas LPG does. To RDF, statements `id:1` and `id:2` are identical and naïvely modelled annotations will get smushed, leading to something like:

```
:Alice :knows :Bob           id:7.
id:7 :statedBy :NYTimes      id:3 ;
     :statedBy :TheGuardian  id:4 ;
     :since 2020             id:5 ;
     :since 2021             id:6 .
```

Obviously this 'dimensional reduction', as [Las+] call it, loses information. The example is a popular one, and has surfaced multiple times in previous chapters. Depending on the details it can be more complex than meets the eye. The underlying problem in the example above is two-fold.

[1]As already employed in other Sections, an ID attribute is added as a forth element to the triples syntax of Turtle. This is of course non-standard, but should be easy enough to comprehend.

First, these annotations are categorically different:

- different occurrences are annotated
- different facts are qualified

The former is handled correctly by RDF standard reification, but in a way too cumbersome to be attractive and too involved to be plausible. As [Las+] point out "[in RDF-star] all statements involving said `(s, p, o)` are understood to reference the same (quoted) triple. In contrast, the original RDF reification mechanism [...] is more expressive: each statement model has its own identity." This characterization is debatable insofar as the RDF reification mechanism is not *more* expressive, but rather expresses something *different*: while RDF-star refers to the triple as an abstract type, RDF standard reification refers to the act of stating a triple. It addresses not so much what was said, but rather the act of saying it. In particular it doesn't refer to any specific triple. RDF-star n the other hand refers to all triples of that type. The latter, qualification of statements, is not allowed in RDF as it would change the truth value of the qualified statement. [Las+] don't discuss the consequences of this issue. RDF may be unnecessarily strict in that regard and this issue probably is resolvable, although formalizations of a more nuanced approach like singleton properties haven't received a critical amount of uptake so far. However, it has to be discussed and settled in order for a solution to be semantically sound. Also, as it is a problem quite close to the core of the RDF semantics, it will probably be at the center of any possible solution and attacking it shouldn't be postponed for too long.

Second, the main point [Las+] want to make is a different one: they would like to represent statements as occurrences instead of types because they would like to spare users the effort of introducing intermediary nodes that represent occurrences. This makes sense as such intermediaries are particularly counter-intuitive and consequently are experienced by users as an unnecessary nuisance. This makes them prone to being ignored and omitted in practice, leading to unsound data modelling. On the other hand the reasons why RDF does and arguably should settle for set semantics have been discussed at length in Section 2.6.2.5 — in short: integration requires to concentrate on the meaning of the triple itself, what it says, whereas applications might be interested to control integration along criteria of who said it, when, why etc, all referring to the *act* of saying something, not on *what* was said. With respect to qualification the case is even clearer: a Knowing that started in 2020 is different from a Knowing starting in 2021. Consequently qualification differentiates the two statements. Just because it took an extra triple to assert the time constraint allowed the triples with `id:1` and `id:2` to look the same, but actually they aren't: they amount to different assertions. However, they still do have a common core which depending on context might be all that counts. So how to express this in a way that is easy to author and query, but doesn't lose detail and context?

There might just be no easy way out of this dilemma, but it can't be ignored either. The authors give an example which saves the information that got lost in the example above:

```
:Alice :knows :Bob            id:1 .
:Alice :knows :Bob            id:2 .
id:1 :since 2020              id:5 .
id:5 :statedBy :NYTimes       id:3 .
id:2 :since 2021              id:6 .
id:6 :statedBy :TheGuardian   id:4 .
```

But they criticise that this amounts to "falling back on more complex modeling approaches (such as introducing intermediate nodes or using named graphs to attach custom statement identifiers)." They elaborate that "one could easily argue that our interpretation of the above model is wrong, because there is no logical dependency between the *since* and *statedBy* statements, and that one should in fact add more 'nesting', [. . .] but this is not the typical pattern employed with LPGs. It also makes query-writing more difficult." However, the authors find that dimensional reduction does not only lose some information when querying — which could be regarded as a regrettable but tolerable loss — but also results in unclear semantics of update and delete operations, stemming from ambiguous references. What should for example happen if a statement with multiple annotations gets updated or deleted: should all annotations follow suite, is it possible to re-construct which occurrence was annotated and which one is to be deleted, is the issuer up the update even aware of the problem? [Las+] caution that "the semantics of such operations needs to be carefully designed in order to avoid unwanted side effects, as to obtain a 'natural' behavior." However, such 'careful designs' always are in danger of getting too closely coupled to specific use cases and cause new and even more intricate trouble in the long run. Sometimes some upfront precision and ensuing tediousness is just unavoidable.

[Las+]'s preference for a usability-focused approach to semantics is not without justification: discussions about semantics notoriously get sidetracked in meandering expanse and ever-descending rabbit-holes. It is a difficult area, prone to misunderstandings, shaky intuitions, strong opinions and little firm ground. And indeed it is largely avoided: this author is not aware of many serious treatments of semantics in RDF on an informal level — ones that don't require sound knowledge of formal logic and don't content themselves with formal correctness, but are concerned about what makes sense to non-logicians, and why, and what can be done about the limits of formal semantics instead of just leaving the last word to applications. The RDF 1.0 specifications provide some notable exceptions in the Primer and Model Theory[2] documents. Pat Hayes' posts to Semantic Web related mailing lists, not without reason cited in this work so often, are an immense source of wisdom with respect to the actual meaning of the formal semantics of RDF in particular and issues of semantics in knowledge representation in general. Still, arguably and obviously, this is not enough. Semantic soundness is an essential ingredient for the Semantic Web to

[2][Hay04b]

succeed as a data integration technology on a global scale. Too much uninformed sloppiness and application-specific intuitions undermine that effort.

Well-informed sloppiness on the other hand is indispensable. Users need simple solutions that are easy to understand and deploy. [Las+] are perfectly right when they refuse to aim for impeccable semantics through rigorously exact modelling. Semantics is a field very easy to get lost in, and quite prone to bigotry. The expressive richness, the malleability and contextuality of natural language is hard to comprehend and quite impossible to formalize to its full extent, much less in an easy to use way. The true and full meaning of a statement is an elusive thing. No description is ever complete. There is no such thing as a perfectly valid and fully understood triple and there never can be. Compromises have to be made, cow paths have to be paved and conservative reading is most advisable (see Section 14.2 below).

However, and this is a big however, there can be no good compromises, no elegant streamlining, no genius shortcuts if the underlying complexity isn't thoroughly examined and well understood. Too often is one aspect taken for the whole, are apples mixed with oranges, are categorically different entities linked without noticing, or caring, or even arguing that one *shouldn't* care. Such nonchalance works in natural language, not only because natural language is very expressive, but also because the context is always rather well defined (and contextuality is presumed). However, when data stemming from different contexts and following different intuitions is integrated, such contextual assumptions may no longer hold. It works in application-centric environments that implicitly rely on the same pre-conditions, even if they integrate data from distant or even unknown sources. However, it starts to fall apart outside of such focused environments, although the aim of the Semantic Web is precisely to go out there and use, re-use and integrate data on a truly global scale and in a fully decentralised manner, without assuming much common context beyond shared vocabularies. The original design of the Semantic Web tried to not aim too high, to not repeat prior mistakes in the field of Artificial Intelligence where goals were often as lofty as actual results disappointing. Example scenarios targeted relatively easy to define domains. A lot of meaning (and disambiguation) was delegated to and encapsulated in ontologies. A model theory was meant to iron out the kinks in the basic RDF formalism itself. Yet, the world is not a flat and well-organized data space. People tend to argue, differ and disagree. Fact, events, theories and descriptions all have more than one facet and can be viewed and described from different angles equally well. Fake news just as well as respectable hypotheses need to be handled with care. Application environments are based on implicit assumptions that may change from task to task. Integrating mission-critical data requires tracking of provenance to ensure accountability and provability.

There are many different ways to say something and they do not all mesh easily with each other. Some leave this out, some take that for granted. They may start from the beginning, the end or in between — how to identify the main actors in a neo-Davidsonian n-ary relation? When resorting to more advanced modelling techniques — because simple triples don't cut it and n-ary relations are too unwieldy — the act of saying something needs to be disambiguated from what was said. Reference to something that was said has to disambiguate between the assumed meaning and the exact wording, and also between a single word or the whole conversation or anything in between — a shape, a graph? The problem of what a word — or an IRI for that matter — means precisely in the given context adds another level of complexity. This can all get incredibly complicated very quickly, and is indeed often a rather hopeless endeavour if not guided by some pragmatism and sound conventions. There is still much room for improvement and better understanding of the complexities involved. On that basis new designs can find better compromises, more elegant shortcuts, easier interfaces that make sense. But such solutions can't be obtained by glossing over those shallows and quagmires, or by bricolage at the open heart of an already quite extensive body of specifications, code bases, and data. They won't show themselves in use cases and they will most probably not arise from a stroke of genius. They require analyses and abstraction. The discussion of semantics in a non-formal sense may be tedious, but it is indispensable.

Chapter 14

The Gestalt Of It All

I am convinced that there is real value to the work that is being done in semantic network representations and that there is much to be learned from it. However, I feel the major discoveries are yet to be made and what is currently being done is not really understood.

William A. Woods[1]

One of the oldest controversies about Aristotle's categories was whether they represent the kinds of things that exist or the way people perceive, think, and talk about things that exist. Theophrastus, Aristotle's successor as head of the Lyceum, said that the categories were intended in all those ways — in modern terms, ontological, epistemological, and lexical. Yet the fragmented methodologies of those subjects are scattered across the fields of philosophy, linguistics, and artificial intelligence, in each of which the researchers who work on formal semantics or lexical semantics are disjoint sets.

John F. Sowa[2]

Woods' assessment in his seminal paper from 1975 seems to still very much hold today. Despite the huge corpus and wide range of approaches presented in Section II something is missing, something essential: a sense of the breadth of the problem and attempts to get a grasp on all its different aspects and facets. Such a discussion may seem quite challenging, but also overdue.

This discussion has to concentrate on cross-cutting concerns. The preceding chapter about possible abstractions made it quite clear that there are too many competing plausible perspectives that one could hope for a single elegant and simple solution to cover all relevant aspects. While this has been tried over and over again and while there has been progress and interesting solutions have emerged, no approach manages to cover a reasonable wide problem space to be considered *the* solution. It may very well be necessary to settle for not only one

[1] [Woo75]
[2] [Sow06]

but a number of approaches, cleverly combined to an intuitive whole. Hopefully a deeper understanding about what the underlying problems, intuitions and information desires are can help find such a combination of techniques to tame the beast. This will undoubtedly require some effort and perseverance.

This is a rather theoretical but at the same time very practical discussion: how do we conceptualize in our minds and what would be the most efficient way to represent those conceptualizations in RDF? Tackling this question requires to observe oneself when thinking, it demands scrutinizing what seems natural for hidden presuppositions, it has to employ a keen sense for the essential and its subtle differentiations, it needs to wander on slippery paths through semantic quagmires, it has to develop and use abstractions without descending in quibblish rabbit holes. It has to be pragmatic, but not afraid of philosophy.

14.1 The Principle of Most Surprise

One should probably not be surprised that a concept as innocent looking as blank nodes comes with so many subtle differentiations and deviations that can seriously hamper interoperability when one expects it the least. The pattern common to diverging interpretations is interesting and typical for discussions about semantics on the Semantic Web: natural language provides many more subtle distinctions and differentiations than one is normally aware of. It's all too easy to get caught in the perception that one knows the right or even the only possible interpretation of the meaning of a concept when there are indeed a few more of them and only context can provide certainty about which interpretation is the most accurate or fitting one for the task at hand.

Knowledge representation is ripe with such shallows and the average Semantic Web practitioner is little equipped to navigate them. The Semantic Web effort endeavoured into the area without much ado, with a very pragmatic attitude and little respect for problems that have plagued philosophers for millennia. That's not a bad thing per se and the call for "a little less hysteria"[3] reflects that hands-on approach. Indeed it is the only possible approach if one doesn't want to get lost in philosophical debate but put knowledge representation to work. However even such a realistic and pragmatic approach requires a solid shared understanding of the right balance between expressivity and verbosity. That in turn can only be developed from an awareness of the quagmires and rabbit-holes of semantics. It is an endless terrain as interesting as tiring and many that entered it unprepared would have preferred to wholeheartedly throw out the semantic baby with the bath tube. A little more education on what problems will inevitably be encountered — like some introductory discussion in the RDF Primer — might have helped to avoid some frustrations and hard feelings.

[3]In Section 2.5.2 on blank nodes above

14.2 Conservative Reading Required

Annotating a statement with further detail does not necessarily have to be understood as constraining its validity, but may rather be interpreted as providing additional information, only adding to what is already known about the world. This depends heavily on the perspective the user takes: e.g. the statement that "Bob likes Alice" can't reasonably be understood as universal and eternal truth. We all know that persons don't live forever, that likings take time to develop and sometimes also fade etc. But even if Bob doesn't like Alice anymore now or in some distant future the liking relation did exists at some point in time. One may argue that the statement never stated anything more than the existence of some liking relation (at some unspecified point in time). Any annotation considering time span, source, verification, the exact nature of that specific instance of liking etc. under such a perspective doesn't impose a constraint but provides additional detail. This would seem like a quite healthy attitude as in fact most statements don't (and can't) claim to state eternal, universal and self-contained truths. A suitable attitude might be to never interpret too much into a statement but rather to 'curb your enthusiasm'.

Some annotations seem to not only add further detail but clearly don't change the truth value of the annotated statement: annotating a statement that Alice bought a car with the payment method or the reason for the purchase should in no way influence the truth value of the annotated fact itself. But the question is if there really is a categorical difference and if yes, how could it be captured in a formally precise way? Or, if that is not possible, how serious a problem is it if one statement does indeed make another one false? Contradicting statements are unavoidable on the Semantic Web and we have to — and regularly do — deal with them out-of-band. That's not nice but it's not the end of the world, or rather the Semantic Web, neither. Why not take the same stance towards conflicts that arise by statements annotating other statements?

[Pat18] distinguishes qualifying annotations from those that merely add additional detail and argues that a qualifying temporal constraint to an "isMarriedTo" relation poses an unwarranted burden on the consumer. On the other hand he sees no problem when the fact that some actor was cast in some movie is annotated with the role the actor played. Such examples seem intuitively clear at first sight but don't necessarily hold up under closer inspection as they rely heavily on implicit assumptions: the property's name suggests a now-ness that isn't backed up by the example data he uses it on (two marriages of Bob Dylan, both divorced by now). Would that property be renamed to something rather awkward sounding like `isOrHasBeenOrIsSupposedToBeOrToHaveBeenInSomeMarriage-RelationWith` things would look and feel quite different. What the relation really aims to convey is that some sort of marriage relation between two persons exists, and understood as an abstract concept rather than a currently valid contract that can indeed be true of past, present and maybe even future marriages. Consequently this might be more a problem of how properties are named (especially those that are freshly minted for use in examples, one might add)? And who expects marriages to last forever, who expects data from the vast Semantic Web to always be up to date, who queries for presently active marriages without adding some kind of temporal argument to the query? In consequence we might have to give up on the hope to find some clear cut distinction between additional fact and limiting constraint, but rather learn to distinguish between solid intuitions and unreasonable expectations. Or, to slightly rephrase Postel's law: be conservative in what you assume to express, be liberal in what you understand from others.[4]

14.3 Meta & Fact

> *There is no useful distinction between the representational needs of data and metadata. The kinds of information that need to be represented in metadata and data are very similar. Furthermore, every item of information, without exception, is likely to be regarded by some applications as ancillary and never to be displayed, and by others as core content that needs to be formatted, printed, or searched.*
>
> R.V. Guha and Tim Bray[5]

Sections 3 on *Demand* and II on *Approaches* have presented a wide range of investigations and proposals to the problem of how to represent complex facts and involved data structures in RDF. Despite this inherent richness most works focus on specific domains with a limited set of metadata dimensions and often also a very specific understanding of how the realms of data and metadata relate and may interact. While many argue for the intuitiveness of their specific approach, those intuitions on closer inspection often don't relate to or even contradict each other. The meta dimensions covered vary widely but many proposals can't be extended to other dimensions, at least not out of the box. Not even the above cited claim that the distinction between data and metadata is largely arbitrary and application dependant is uncontentious — which, given the obvious diversity of this domain, is surprising.

This section collects the different dimensions, idioms and intuitions that drive the works presented so far. The picture that it paints is however not uniform and nicely structured, quite to the contrary: "metadata" is a topic with many facets, cross-cutting concerns and strong undercurrents, and "intuitions" are not always what they seem. Instead of pressing this diversity into yet another rigid scheme it seems more useful to first provide a rather faithful account of the many aspects that have so far been articulated or at least implied. In conclusion an attempt will be made to capture the gist of it, pointing to possible opportunities for a (more) unifying architecture of meta-modelling.

[4]Originally: "Be conservative in what you do, be liberal in what you accept from others."

[5]From the specification of the pre-cursor to RDF, the Meta Content Framework (MCF), 1997, `https://www.w3.org/TR/NOTE-MCF-XML-970624/`

14.4 Facts & Acts

Facts concern things that happen or are arranged in a certain way, as statements. Acts concern the stating of facts, or speech acts, or occurrences of statements.

Some examples focus on occurrences, the same statement stated twice:

```
:john :loves :mary id:1 .
id:1  :source :alice ;
      :credibility 0.8 .
id:1  :source :theGoldenLeaf ;
      :credibility 0.1 .
```

Some examples focus on — not quite — the same fact stated twice:

```
:richardBurton :marriedTo :lizTaylor id:2 .
id:2  :start 1964 ;
      :end 1974 .
id:2  :start 1975 ;
      :end 1976 .
```

The first example refers to the speech act itself, whereas the second example qualifies the statement. But to a user the source of a statement and its especially its credibility might make all the difference, thereby indeed qualifying the statement itself also in the first example. Should both examples be modelled differently? Would that hep or even be possible given that the disambiguation depends on the use case which may well not be known when the statement is made?

14.5 Updates & Merges

What starts out as a simple fact may get ambiguous with the addition of other simple facts. Taking a simple example:

```
:bob :loves :cake ;
     :for :breakfast .
```

Let's add another one:

```
:bob :loves :steak.
```

Now merge the two:

```
:bob :loves :cake ,
            :steak ;
     :for :breakfast .
```

Granted, this mix-up could have been avoided by introducing an intermediate (blank) node for the multipart statement that Bob loves cake for breakfast. Should such a modelling style become mandatory to avoid possible problems in the future? But who would follow such a rule? Or could an extra construct be introduced to disambiguate existing facts and multipart facts on demand, but without changing them? How can ambiguity be avoided without demanding overly involved modelling styles? How does modelling scale from simple to involved without requiring re-modelling?

14.6 Structure & Meta-Modelling

Two problem spaces penetrate and interact with each other: the simplistic primitive, the triple, on which RDF is based requires modelling strategies that introduce complexity and complications, like n-ary relations and blank nodes. No mechanism is provided to manage that complexity, e.g. to demarcate complex objects by some sort of boundary. On the other hand complex knowledge constructs ask for means to structure the information, to differentiate between primary fact and secondary detail, or between fact and context, between data and metadata. All those meta-modelling desires compete with structural requirements for the same modelling primitives, be it what RDF already provides or be it extensions like named graphs etc. Mechanisms meant to structure data are used to annotate it, annotation mechanisms are repurposed to introduce some structure.

14.7 Integration & Application

The Semantic Web wants to have the cake and eat it too. Integration of all the data requires the open world semantics that RDF specifies. But using the data in applications needs a different set of fixings. It is quite problematic that RDF provides so little support to close down a world, make blank nodes nominals, be referentially opaque, treat statements as occurrences instead of references to their abstract type etc.

While the RDF semantics is targeted towards data integration, SPARQL is optimized for applications. Both designs are in principle well justified, but the resulting rift requires practitioners to understand the underlying machinery very well and navigate its fault lines keenly and with great care.

The referentially opaque semantics of Notation3, named graphs as proposed by [Car+05a] and RDF-star attempt to integrate application semantics into the open world of RDF, but are optimized for reasoning tasks, not for general-purpose Semantic Web application development.

14.8 Triples & More

The Semantic Web would not be itself without the triple, the very basic relation linking a subject by a property to an object. It was a design decision made very early on in the development of the Semantic Web, that a directed labelled graph would be the most efficient way to integrate all kinds and sources of data.[6] As [GMF04] notes, "[t]he ease with which web sites could link to each other doubtless contributed to the rapid adoption of the World Wide Web. It is hoped that as the Semantic Web becomes more prevalent, programs will be able to similarly

[6]See R.V.Guha's keynote at ISWC 2013, "Light at the End of the Tunnel", `http://videolectures.net/iswc2013_guha_tunnel/` at around minute 5:00, for a lively account of that early phase of Semantic Web engineering.

weave together data from diverse sites. Indeed, the data model behind RDF was significantly motivated by the fact that directed labelled graphs provided a simple, but effective model for aggregating data from different sources."

More recently in a proposal for OKN, "an open web-scale knowledge network [. . .] envisioned as a distributed, federated system that anyone can participate in", [GM16] stress the importance of the basic RDF formalism, the statement triple, when developing a decentralized Knowledge Graph. The bare triple, linking well known entities from disparate sources through well known properties, may be the only practical way to successfully navigate a decentralized Knowledge Graph. Any more involved modelling primitive constitutes an obstacle that requires extra, possibly prohibitive, upfront effort and domain specific knowledge to overcome.

But despite the glorious achievements of the simple triple: the mighty graph might not be the data structure to rule them all. A small graph is almost always easy to understand and feels intuitive. A big graph however tends to be messy, noisy and unwieldy. It doesn't provide any orientation, there's no up and down, no start and finish. Semantic Web enthusiasts tend to gloss over the fact that simple lists are a very useful tool: they provide just a little bit of structure but are almost not perceptible. They are very easily extensible too: a tree is just a list of lists, a table is a list of uniform lists. Those structures are immensely powerful, they guide the eye just as well as an algorithm, yet they are easy to implement and use. Sure and true, they don't provide the flexibility of graphs. They can't match the ease with which graphs can accommodate unforeseen requirements. They don't provide means to express more than one perspective on the same data. They are closely tied to specific uses and applications of the data they encode. But it's important to realize the amount of useful semantic boilerplate that a list or tree or table provide out of the box and that a graph has to explicitly construct if needed in a cumbersome effort. That's one reason why there are so many approaches to structure the graph with metadata as data about date. It might be useful to tackle this issue in a more straightforward fashion and follow the proposal by [LW10] to extend the RDF model with nested lists. This is a relatively easy to implement extension, certainly less involved and probably more usable than employing qualifying annotations or contextualization to emulate such structures.

14.9 Data & Metadata

Conceptualisations of meta-modelling are prone with dualities: annotation and contextualisation, attribution and qualification, attribution and composition, types and instances, contextualisation and administrative housekeeping, internal detail and external context, content and provenance, and so on and on until the overwhelming arch-duality, data and metadata.

It is interesting to see how from the very beginning of the design and development of the Semantic Web it has been stated that the distinction between data and metadata is essentially arbitrary and application-dependent and that RDF should be agnostic to such aspects, but still the obvious preferences in shared practices tend to creep into the basic abstractions and intuitions. Provenance is a perfect example: it was one of the first applications of metadata on the web in the form of the Dublin Core metadata set, then became via the Prov ontology one of the central vocabularies on the Semantic Web — and indeed, when integrating data from sources near and far keeping track of provenance is an important administrative task. Yet, to a knowledge representation formalisms that goes beyond just integration of facts but aims at discovering new connections and perspectives and turning facts into knowledge, it is important to have maximum flexibility and no predefined dichotomies in the data.

There is a difference between hardcoding such dualities into the core formalism or making them available at the application level. Named graphs understood as a mere administrative tool that may streamline certain operations or speed up queries for aspects common to a dataset are certainly a sensible addition to the engineering toolbox. Anything more definite will probably rather lead to fragmentation than to easier integration.

Annotation and contextualisation might seem like a promising way to order the chaos: annotations annotating the single fact and adding intrinsic detail, contextualisation grouping statements in graphs according to external and commonly shared situational (or, again, administrative) circumstances. Yet, as intuitive this may seem at first sight, there will hardly be any broader agreement on what aspects constitutes detail, context etc in all cases. Probably the same user will look at the same data quite differently when tasks change or familiarity with the data grows. Again, hardcoding any such categories into the basic formalism will at best be short-term advantages, but generate pain in the long run.

Encoding knowledge tends quite necessarily to impose some structure and some implicit assumptions about relations and priorities: that is as true for relational databases as it is for the mighty graph, which despite all its flexibility is not without structure itself, as the discussions about n-ary relations, Davidsonian and Neo-Davidsonian alike, should have illustrated.

It would certainly help if the meta-modelling facilities of RDF could be improved, but also standardized. In the end it doesn't matter *that much* if the result is Davidsonian n-ary relations or annotated triples or quads or quins. What's important is that *the same* formalism is used everywhere and for everything. There's no bigger nuisance than having to query for the same fact in multiple ways, as e.g. an annotation on a statement and a contextualisation in a graph, because one wouldn't know in advance which aspects in the data where mapped to which perspective. That won't stop the discussions for which representation mechanism is *the best* for now, but hopefully rather sooner than later. Standardization of meta-modelling approaches combined with ways to express their application in the data might become a means to ease transformation of data, but also queries from one approach to the other.

Chapter 15

Perspectives

There is a kind of universal understanding about what an RDF triple 'means': it says that a relation holds between two things, and it is therefore an assertion about a world consisting of these things with relations holding between them. This consensus of intended meaning is perhaps a bit shaky at the edges and under strain in some places, but still is kind of universally understood and accepted. But there is no such consensus AT ALL about what a list is supposed to mean, or what an array is supposed to mean, when used as part of an assertion about this common 'world' of things bound together by relations. And because there is no such consensus, as soon as these structures are given to developers to build with, what they build will have, at best, an idiosyncratic meaning private to the community in which the developer is working. At which point, the entire purpose of RDF is lost. Which is why any new syntactic extension to RDF should be given a semantics as part of its normative definition, and this should be one that is likely to survive the pressures of how developers are wanting to use the new structure. For example, if it is clear that some folk really want to use arrays to express n-ary relations, while others really want to use them to abbreviate conjunctions but with a closed-world assumption, then these two groups should be given distinct extensions to RDF syntax which will not be confused with each other, and each given a nice crisp semantics and tutorial examples, etc...

Good luck, y'all.

Pat Hayes[1]

[1] In a post to the Semantic Web mailing list, `https://lists.w3.org/Archives/Public/semantic-web/2022Sep/0084.html`

Hayes' comment touches the pain point of semantics, no matter if formalized in a model theory or not: if it doesn't capture what users expect and need, it won't survive the pressure of practice. Nobody is served with a powerful and involved design that excels in specializations but makes ordinary, middle-of-the-road tasks harder than necessary — it will just be ignored, rendering the result semantically worthless and the whole effort futile. But what can be done, then, as long as no saving Grand Design has revealed itself to the Semantic Web faithful?

15.1 Pragmatics

The Semantic Web is *Triples all the Way.* Basing meta-modelling on another primitive, e.g. on named graphs or even n-ary relations, must weigh eventual benefits against the many costs that may incur:

– databases favour straightforward indexing and struggle with unexpectedly large value spaces like those induced by singleton properties and singleton graphs (see [Fre+19b])
– databases favour predictable joins (see [Fre+19b])
– paths can't be queried across named graphs (see [HHK15])
– navigation and query construction favour a minimal set of primitives
– reasoning vocabularies like OWL are optimized for simple triples
– new aspects shouldn't change the footprint of a relation, they shouldn't require re-modelling of data and adjusting existing queries (see [GP13] and [Das21])

The reason why fluents, annotations on the subject- and objects-terms of triples themselves, perform so favourably in terms of entailments is that they don't interfere with the triple-based formalisms that drive the Semantic Web. Their usability issues with novice users however show that they as well don't provide an easy solution on their own.

15.2 Principles

Any design aiming to improve the expressiveness of RDF should try to meet the following principles:

15.2.1 Intuitive

Formal semantics have to follow intuitions, not break them. formalizations trying to achieve some greater goal at the cost of immediate naturalness are bound to be ignored in practice. Consequently they lead to semantically unsound statements, adding to confusion rather than clarity, as they should. A good guide to intuitions is observing and drawing parallels to natural language, even observing oneself when developing thoughts and formulating statements.

15.2.2 Idiomatic

Triples must be at the heart. Navigation is based on triples, ontologies are designed for triples, querying focuses on simple triple patterns and triple paths, reasoning is optimized for triples. Implementations based on triples provide exciting opportunities as recent contributions like Tentris[2] and MilleniumDB[3] show.

15.2.3 Resilience

Updates to the data already present — be it qualifications, annotations or simply additional detail — should not change the structure of what's already there. They should not require an update to existing queries and they should also not force a new, navigational query to try multiple variants of how a fact might be modelled.

Notably, n-ary relations tend to break that principle: adding additional detail to what was first a simple statement tends to require the addition of an indirection, a blank node probably, changing the structure of existing data. However, better support for a property distinguishing the primary value from additional detail, like `rdf:value` does, could go a long way.

15.2.4 Curb Your Enthusiasm

This is probably the hardest part: always being aware, or rather taking care to making oneself aware of the hidden assumptions, the intuitions driven by not explicitly stated context, the facts that are taken for granted but are not explicitly shared.

How easy it is to assume that a stated fact is valid *now*, and overlooking that a source might have gone stale, or indeed might never have subscribed to that concept of *now-ness* that so many seem to take for granted on the Semantic Web, although it is neither enshrined in any RDF specification — quite to the contrary, actually — nor a very reasonable expectations by any means, given the rather chaotic state of the web (and the world, one might add) as a whole.

"Be liberal in what you accept, be conservative in what you assume" is a good guide not only when consuming data, but also when designing data representation formalisms.

15.2.5 Premature Optimization

"...is the root of all evil" [Knu97] and it is the other side of the quest for simplicity and scalability that drives the design of the Semantic Web. Not repeating the mistakes of overly ambitious endeavours in Artificial Intelligence is of course a good goal to have, and catering for users that are not trained logicians but rather lay database practitioners, web developers and even mere

[2][Big+20]
[3][Vrg+21]

content authors requires considerable restraint when balancing expressiveness with ease of use. However, and the Semantic Web seems to learn that the hard way, a formalisms that is successful, can't be too simplistic: if it doesn't meet certain needs they will find their own way, but probably not in the most elegant manner imaginable, maybe even rather crude and lastingly unfinished like named graphs, and Collections even.

15.2.6 Late Binding

Many, if not most, possible interpretations of complex constructs — from n-ary relations to attributions and qualifications even — vary in more or less subtle ways. The preceding chapter on *the Gestalt of it all* gave some examples of how even mainstays of computer science like the distinction between instances and subtypes are dependent on context, on points of view or maybe even just a matter of personal preference and modelling style. It seems rather impossible to fix and enforce such interpretations. Ontologists may reasonably be asked to consider such detail when designing vocabularies and categorizations. Trying to make users to ponder semantic subtleties when all they want is to publish some data to get an interaction going is a rather hopeless endeavour.

The design of the Semantic Web tried to avoid such quagmires and like the architecture of web itself — which was dismissed in the beginning by hypertext specialists as too simplistic and missing crucial features to ensure sound and solid operations — often opted for the easy way out, trusting users and applications to disambiguate just well, and on demand. Identification is one example that has been discussed at length in Section 2.1 above. This late-binding style of operations can carry very far, as both the web and the Semantic Web have proven. However, it wouldn't hurt if some more means were available to disambiguate on demand, when the need arises — and again: without changing what's already there, without disturbing any established processes and structures. Again, identification is one example where this might be really useful.

15.3 Possibilities

Some clarity about reasonable aims might have been established. The goal is an approach to representing complexity that leaves the simplicity and straightforwardness of basic RDF intact and only chimes in when needed, honouring the late-binding style of web architecture. Specifically, a solution should:

- leave the basic triple-focused structure intact
- leave shared concepts in nodes and vocabularies intact (in contrast to n-ary relations, fluents and singleton properties)
- provide intuitive defaults (catered to the usual, rather closed world application scenario)
- lend itself to easy on-demand expressiveness when need arises

- *not* try to disambiguate context from annotations, attribution from contextualisation, subtypes from instances
- be careful how it mixes structure with content (unavoidable, but risky)

Some of the approaches discussed so far show potential, each with its own Pro's and Con's.

15.3.1 N-ary Relations and Shortcuts

Some have asked for one definitive style to model n-ary relations or for the introduction of n-tuples into the RDF model even.[4] Neo-Davidsonian n-ary relations are the most generic form of n-ary relations and could serve as the backbone to model anything more complex than a simple relation between two things that are more or less on equal footing. Shapes might help to describe those n-ary relations and make them easier to optimize for in storage, indexing, query planning, sharing etc.

Shortcut relations might add the usability that involved n-ary relations tend to lack, making easy connections in the spirit of the basic graph formalism that RDF is based on. Those shortcuts should have a link back to the complex shapes they are derived from, to make it easy to zoom into a detailed view when navigating shortcuts has led to an interesting intersection.

15.3.2 Fluents

Fluents and their favourable properties for reasoning, but also their unfavourable usability characteristics, have been discussed already. Chapter 15.5 presents a brainstorming proposal on how to overcome the usability issues and put the triple-friendliness of the approach to good use.

Syntactically it is based on annotating statements via statement identifiers. Under the hood however it qualifies terms and interprets annotated triples as a sort of surface syntax for a very blank-node heavy but sound and unambiguous representation.

15.3.3 Property Annotations

The singleton property approach has the advantage that it stays true to the triple and can be implemented without requiring any kind of extension to the RDF model. It met some resistance in the database community because some databases are optimized for relatively sparsely populated property fields, and lost momentum and favours compared to the then upcoming RDF* approach. However, with the right optimization it has proven to perform quite well. As no solution to the representational issues of RDF comes for free, the singleton property approach might be worth a second look.

[4]See Section 2.7.4.1 above

It would however benefit from the adoption of generalized RDF and a qualifying property that distinguishes the primary value — the property from which the Singleton is derived — and its attributes.

15.3.4 Quads as Triples + ID

Quads, here understood as statements with an accompanying statement identifier that carries no meaning of its own, have of late garnered support in big commercial offerings of graph data — however, and rather tellingly, not pure RDF environments but combined RDF and LPG instalments. The RDF world has long eschewed statement level annotation as overly fine-grained and, if in doubt, already supported by RDF standard reification, even given the verbosity of that approach. While semanticists are not overly fond of statement identifiers because they fear the implications of self-reference, RDF practitioners have been asking for grouping devices to streamline operations and improve performance, not yet another addition to the model that needs to be indexed and updated.

The onslaught of LPG databases on RDF's market share, but also the success of Wikidata with its roots in the Semantic Web, but its modelling style firmly in LPG territory, seems to have changed the calculations rather drastically. Quads provide some unique modelling opportunities, as they allow to refer to statements very easily, facilitating recursive operations and late binding disambiguation. They also may challenge some very fundamental principles of RDF, namely that statements can't be qualified by other statements.

15.3.5 Named Graphs

Named graphs were once the go-to solution for meta-modelling needs in RDF: they are widely supported, provide a succinct syntax and are easy to reference, and, as a grouping mechanism, have immediate appeal to those looking for streamlined and scalable meta-modelling. Even nesting of graphs, although supported natively only in Notation3, can be simulated, as found in some approaches.

The lack of a standardized model-theoretic semantics for named graphs and the inability of the RDF 1.1 WG to come to a consensus about such a semantics has however curbed enthusiasm. Although there are ways to express and refer to named graphs with sound semantics, the common intuition seems to have shifted towards a position that accepts their use for application specific demands like data management and close-to-the-metal optimizations and prefers to keep a distance from that. Also, as with singleton properties there is a reluctance to overload named graphs, which are mainly used for grouping and therefore usually not fully indexed, with triple level functionality.

15.3.6 Quintuples

Quintuples, or quins for short, advocate to bite the bullet and accept that two mechanisms are needed: statement identification to support Labelled Property Graph-style annotation, and named graphs to group statements (and annotations on them) for management and optimization purposes. While some databases have supported this approach for a long time already, it was very niche and has only recently been mentioned more often — reflecting the increasing importance of Labelled Property Graphs also in the RDF world.

A thorough comparison of these alternatives, along a well defined set of tasks and criteria w.r.t. expressivity, ease of modelling and querying, resilience to change, entailment preservation and overall implementation performance, should be an interesting task for future work.

15.4 Practicalities

Despite the deeper and quite hard to solve problems that this work may make seem to abound on the Semantic Web (which they of course do, but obviously not to its imminent demise) there are some low hanging fruit that should be harvested as soon as possible.

These are possible, plausible and reasonable improvements that a slightly agile standardization process could tackle rather swiftly. They neither require much implementation effort nor do they impose grave changes to model, syntax or semantics of RDF, if any at all. Of course, there will be no completely free lunch, but the advantages seem to outweigh the costs by far.

15.4.1 Blank Nodes

Blank Nodes, as extensively discussed in Section 2.5.2, still need some polishing — not so much a general overhaul, but education, best practices and maybe, maybe even a property to expressly state if one should not be leaned.

15.4.2 Lists

An `rdfx:length` property could be added to the `rdfs:Container` vocabulary. As discussed in Section 2.4.2 this would improve the utility of `rdf:Bag`, `rdf:Seq` and `rdf:Alt` to applications and make their advantages in syntactic conciseness and raw performance more usable and useful.

15.4.3 RDF Literal Datatype

Defining an RDF Literal Datatype, as discussed in 4.2.1.2, would require only minimal effort and no extension to the core of RDF itself, but could proof beneficial in multiple ways. For one it enables to implement quoted RDF, RDF that is referentially opaque, as in Notation3 formulas, named graphs per

[Car+05a] and RDF-star quoted triples, but without the fuss and with very clear and intuitive semantics: a literal enclosed by quotation marks is quite unmistakably as a quote, and not asserted. It is a very pragmatic approach to syntactically faithful representation. Such literals could as well be used to encode closed lists, multi-statement OWL axioms, signed pieces of RDF data or any shape of data, together with some guarantee of integrity and wellformedness. SPARQL would of course have to recognise and query such literals to harness their full potential.

15.4.4 Import

An appropriate `rdfx:import` property could *unfold* before mentioned RDF Literals into a graph, ensuring their syntactic integrity and fidelity to the original as published. Other import mechanisms akin to those proposed by Notation3 may define imports with different semantics.

15.4.5 Generalized RDF

Generalized RDF, allowing literals as subjects and blank nodes as predicates, has for long been on the list of rather obvious improvements to RDF with little to no cost at all. This would make some sensible idioms possible that currently require extra effort. Blank nodes in predicate position would for example make annotating statements via their properties very straightforward.

15.4.6 Semantics Ontology

The lack of vocabulary to define semantic attributes like the identification semantics of a term, a local unique name assumption or a locally closed world has been discussed in Section 7. Such an ontology could be helpful to expressly state intended semantics even if it depends on out-of-band application logic to implement. It could also be used to make certain changes in semantics explicit like when counting semantics are employed in SPARQL or if blank node leaning is applied or not. This might not make a huge practical difference, but could serve to increase awareness and confidence among developers.

15.4.7 Named Graph Semantics

One practical application of such a vocabulary would be to expressly declare the semantics of named graphs, per individual graph or for a whole dataset. The way named graphs are defined in RDF 1.1 leaves applications maximal freedom w.r.t. semantics, and given the complications and practicalities that applications have to deal with that may well be a sensible default arrangement. However, providing the option to define sound and unambiguous semantics would certainly not harm anybody, but increase chances for successful integration across application borders. Here again a useful and sensible default might be a SPARQL-friendly semantics of referentially transparent occurrences, or speech-acts.

The discussion about naming semantics of RDF named graphs could be resolved in a similar vein: the SPARQL 1.1 Graph Store HTTP Protocol allows to unambiguously address a named graph. This makes graph identification as weak or strong as any identification by IRI. The `<>` syntax is used to refer to the dataset from inside a named graph. Add to that a new `<.>` syntax to refer to the named graph itself from inside a named graph, as a means for self-reference, and any named graph can declare its own semantics, no matter if it is aware of the dataset address or graph name under which it is stored, and without the verbosity that is incurred by the spelling out the whole IRI composed of dataset address, query parameters and graph name.

15.4.8 Identification

Identification semantics in RDF could be vastly improved by a small vocabulary that allows to express per node in a statement if it is meant to reference the resource itself or what the resource refers (what it means). The same vocabulary can be used to disambiguate graph naming semantics.

15.5 Proposal: RDF Silhouettes

With *RDF Silhouettes* we present a proposal inspired by NdTerms[5]. [ZG17] introduce the NdTerms approach and show that it has quite favourable reasoning properties and stays true to the basic triple formalism of RDF. It seems to confirm the insight expressed by [Sow03] and [GP13] that annotations and contextualisations can all be applied at any level of representation: adding them to groups of triples just trades some space efficiency for expressivity, but doesn't change the underlying semantics — assuming one uses appropriate terminology to differentiate between annotations on a construct (triple or graph) from its constituents (vertices and edges).

However, without some supporting surface syntax, the approach has usability issues, like e.g. [GM03a],[WF06] and [Sch+13] report them for similar approaches with fluents. Our proposal investigates ways to model fluents just like normal triples, keeping the complexity inducing annotations under the hood but nearby, within easy reach. We don't claim to present a formalism ready for deployment on the Semantic Web and neither for it to solve all problems and challenges identified so far. Rather we understand this proposal as a thought experiment, investigating possible intuitions and designs to make an approach that is quite radical in theory usable in practice.

[5]See Section 9

15.5.1 The General Idea

In this approach the familiar triples as seen in interfaces are just silhouettes of a much more differentiated knowledge structure. Every node and property seen in a silhouette triple is indeed a primary value of a more complex, annotated value. The statements on the surface are called *silhouette triples* and the subjects, predicates and objects *shortcuts*. The shortcut nodes and edges in the silhouette triples are connected to their 'real', or rather more precisely defined value, via an `IS` relation, in which they are the object. The subject of such an `IS` relation is a blank node. All attributes that annotate and further specify the shortcut are attached to that blank node. The user-facing silhouette triple is just a placeholder for that expanded, blank-node based representation. The expanded version, also called a *write out*, is composed of blank nodes, each with an `IS` relation to a privileged primary value and optionally further secondary attributes (as also the property is a blank node this requires Generalized RDF).

The meaning of a silhouette triple is defined by its expanded, blank node based representation.

Let's see a first example, without any further annotations except the primary values (this silhouette triple is actually just an ordinary triple and would in practice not get the blank node treatment):

```
:s :p :o .                          // the so-called silhouette triple
```

is just a placeholder for

```
_:s _:p _:o .                       // the expanded version or "write out"
_:s IS :s .                         // and its relations to shortcuts :s,
_:p IS :p .                         // :p and
_:o IS :o .                         // :o
```

`IS` is a new property, central to this proposal, and in its range always pointing to the primary value of a blank node in its domain (much like `rdf:value`, but with stricter semantics).

In the write out annotations are attached to those blank nodes. Those blank nodes represent fluents as defined in [ZG17]. Silhouette triples look and behave like ordinary triples, and their relations to each other are captured via ordinary triples. In fact a silhouette triple without annotations is an ordinary triple.

The written out, expanded representation requires lots of unwieldy blank nodes. The shortcut syntax instead uses a statement identifier to capture annotations on the silhouette triple. That identifier will be discussed in detail in the next section as there's a lot to consider. For now let's just assume the identifier is unambiguously attached to an occurrence of a statement and readily available in syntax and application layers.

The following is an example of a silhouette representation of an annotated statement. It describes Alice buying her first car at the age of 21. The car itself is second-hand and 12 years old, Alice pays cash, and all this is known for sure as reported by `:Bob` in 2001. The syntax of our example is a derivation of Turtle, with a statement identifier added in front of every statement.

```
id:1    :Alice :buys :Car .         // id and silhouette triple
id:2    id:1#Subject :age 21 .      // annotations on shortcut nodes
id:3    id:1#Predicate :funds :Cash .
id:4    id:1#Object :age 12 .
id:5    id:1 :creator :Bob ;        // annotating the whole statement
id:6         sc:inScene :S_1 .      // contextualisation by some scene
id:7    :S_1 :year 2001 .           // a scene can itself be annotated
id:8    id:2 :trust :High .         // an annotation on an annotation
```

Similarities to Labelled Property Graphs are intended. Property Graphs usually offer annotations on individual nodes while approaches to annotations in RDF most often only allow to annotate the whole statement. The common workaround to attach statement annotations to the property e.g. in the singleton properties approach has obvious shortcomings w.r.t. expressivity. Our approach facilitates annotations on any part as well as the whole of a statement.

Scene containment is a special kind of attribute meant to group statements. This is similar to RDF graphs. However the term graph in RDF is quite contested and has different meanings. We name our grouping facility a *scene* instead to emphasize that unlike graphs encoded in RDF documents in a file system or as named graphs in RDF Datasets a scene has no physical aspect to it. Scenes themselves can be attributed, so all sorts of groups, group memberships, group attributes, group attribute inheritance etc. can be described. This is discussed in more detail below. Another related term that might come to mind is 'context' but that again is a very loaded term with a lot of different interpretations and connotations. It does however fit the concept of scene better than the term 'graph' and could indeed be considered quite equivalent. Additionally we use the term *source* to refer to documents and named graphs as concrete locations from which RDF statements can be retrieved.

Adding that Alice bought a new car when she was older and had started to earn some serious money requires introducing a new annotated occurrence of the statement (introducing multisets):

```
id:9    :Alice :buys :Car .         // new occurrence of same triple
id:10   id:9#Subject :age 28 .
id:11   id:9#Predicate :funds :Card .
id:12   id:9#Object :age :BrandNew .
id:13   id:9 sc:inScene :S_1 ;
                :purpose :Business .
```

In other statements we might want to annotate just one node:

```
id:14   :Alice :loves :Madonna .
id:15   id:14#Subject :age 12 .
id:16   :Alice :hates :Car .
id:17   id:16#Object :age :Wreck .
```

These silhouette triples should look familiar, modulo the statement identifier. However the nodes they are composed of are just shortcuts. Their expanded version, written out fully in standard RDF, is quite alienating, as the following section shows.

15.5.2 Mapping to Standard RDF

A silhouette triple is mapped to its written out, expanded representation by the following algorithm:

– every annotated node is converted to a blank node,
– an `IS` relation annotates each blank node with its shortcut value,
– all annotations on shortcut nodes are attributed to the respective blank nodes and
– every statement that is annotated as a whole is reified using RDF standard reification and statement annotations are attributed to the subject of the reification quad.

The silhouette triples

```
id:1    :Alice :buys :Car .
id:2    id:1#Subject :age 21 .
id:3    id:1#Predicate :funds :Cash .
id:4    id:1#Object :age 12 .
```

expand to

```
_:s _:p _:o .
_:s IS :Alice ;
    :age 21 .
_:p IS :buys ,
    :funds :Cash .
_:o IS :Car ;
    :age 12 .
```

To refer to the whole statement we employ RDF standard reification.

```
id:5    id:1 :creator :Bob ;
id:6         sc:inScene :S_1 .
```

expands to

```
id:1 rdf:subject _:s ;
        rdf:predicate _:p ;
        rdf:object _:o ;
        sc:inSource <> ;           // <> referencing the local graph
        :creator :Bob ;
        sc:inScene :S_1 .
```

RDF standard reification is syntactically unattractive, but in a triple based syntax it's a straightforward solution. RDF standard reification misses a notion of the location where a statement occurs, like e.g. a reference to some graph or document, local or elsewhere. However in our case the nodes described are all blank nodes, so necessarily local. Still an explicit reference to the local graph as in the example above might be useful. RDF-star would offer a more concise encoding, with `IS` defined as by default pointing to the local graph and as a Transparency-Enabling Property (TEP) per the RDF-star CG report:

```
id:1 IS << _:s _:p _:o >> ;
        :creator :Bob ;
        sc:inScene :S_1 .
```

More options will be discussed below after other details have been introduced. In practice however users should be able to work with the surface syntax of annotated silhouette triples most of the time and won't have to bother about the employed reification technique.

The above example fully written out, id:1 to id:17:

```
_:s1 _p:1 _:o1 .
_:s1 IS :Alice ;
        :age 21 .
_:p1 IS :buys ;
        :funds :Cash .
_:o1 IS :Car ;
        :age 12 .
id:1 a rdf:Statement ;
        rdf:subject _:s1 ;
        rdf:predicate _:p1 ;
        rdf:object _:o1 ;
        :creator :Bob ;
        sc:inScene :S_1 .
:S_1 :year 2001 .
id:2 a rdf:Statement ;
        rdf:subject _:s1 ;
        rdf:predicate :age ;
        rdf:object 21 ;
        :trust :High .

_:s2 _:p2 _:o2 .
_:s2 IS :Alice ;
        :age 28 .
_:p2 IS :buys ;
        :funds :Card .
_:o2 IS :Car ;
        :age :BrandNew .
id:9 a rdf:Statement ;
        rdf:subject _:s2;
        rdf:predicate _:p2 ;
        rdf:object _:o2 ;
        sc:inScene :S_1 ;
        :purpose :Business .

_:s3 :loves :Madonna ;
        IS :Alice ;
        :age 12 .

:Alice :hates _:o4 .
_:o4 IS :Car ;
        :age :Wreck .
```

Note that:

- statements without annotations won't get modified at all,
- only annotated nodes require the blank node treatment,
- only statement level annotations require reification.

The written out, expanded RDF should never be visible to the regular user. It is however important for unambiguously capturing the semantics of annotated statements and controlling entailment.

15.5.3 Merits and Pitfalls of the Silhouette Syntax

Even the silhouette syntax, although much more usable than the expanded write out, may look complicated at first, but that's because it is able to represent a lot of information in quite compact ways. The user is however free to concentrate on the main relations between silhouette shortcuts and ignore their annotations in a first run. This is the main usability feature of annotated relations: they separate the central facts from additional detail but keep the additional details nearby. All the user has to do to check for annotations is to query for triples with the statement identifier as subject.

On the other hand, the last example illustrates that annotations may be important: Alice doesn't hate cars in general (quite to the contrary), she only hates old and wrecked cars. So not paying attention to annotations on the silhouette triple may be misleading — after all it just gives a rough idea, omitting the details. However a reader, or an application, not aware of the annotation syntax may miss the annotation. This issue might need more attention.

Also note the details and pitfalls of multi-level annotation:

– statement `id:2` is an annotation on an annotation and has to reify the first annotation — a possibly confusing and hard to read indirection,
– we know that `:Bob` created statement `id:1` but we don't know who created statement `id:2`,
– if we wanted to record the provenance of the trust annotation, we would have to reify statement `id:5`.

15.5.4 Statement Identifiers

The statement identifier and its silhouette representation, the `id:` attribute as a fourth node, are the most visible novelty of this proposal. Statement identifiers have however been part of RDF since the early days, namely as an 'id' attribute in RDF/XML, providing syntactic sugar for the RDF standard reification quad. Our approach is indeed very similar. Some aspects of statement identifiers that are closely related to graph semantics will be touched upon here rather superficially and then revisited after we have discussed all graph-related issues below. We will then also explore options to implement storage of statement identifiers in the quad store dominated Semantic Web ecosystem later below. Note that we will from now on mostly use the technically more correct term *source* rather than the customary, but ambiguous term 'graph'. The RDF specification differentiates a 'source' as a "persistent yet mutable source or container of RDF graphs" from a 'graph' as an abstract set-theoretic concept. It's quite common to use the term 'graph' when speaking about sources, but from now on we need to be more precise.

15.5.4.1 Semantics

The semantics of the `id:` attribute is based on RDF standard reification where the statement identifier is the subject of an RDF standard reification quad. As such it refers to the referentially transparent occurrence of a triple. However our proposal avoids some of the problems of the (non-normative) semantics of RDF standard reification, or rather: it moves those problems into the area of blank node semantics. As the expanded representation of an annotated triple is based on blank nodes, the question of whether or not two annotated triples are the same or different is a question of merging blank nodes.

The notion of an 'occurrence' is problematic in the set based semantics of RDF. It was not an oversight but a result of a mismatch between semantic theory and practical demands that RDF's standard reification vocabulary is underspecified. RDF allows to describe the subject, predicate and object of an occurrence of a triple, but omits a crucial part: the notion of the location of an occurrence. It provides no property that would allow to specify a document or named graph in which a statement occurs.

In our proposal this question becomes obsolete as in the written out, expanded syntax no two statements share the same blank nodes, no matter if they all refer to the same shortcut value. Annotated triples may either be considered the same — then their blank nodes are merged — or different — then their blank nodes are kept apart. The decision to merge or not to merge is out-of-band for RDF: it can only be guided by application specific means. While deferring this decision may seem a bit whimsical, it is indeed a pragmatic as well as prudent approach: the answer to the question if two entities are the same is often context dependent. Questions of identity like if a thing is the same as all of its parts have been discussed at least since the old Greeks and the answer is all too often that "it depends". Our proposal allows to make that dependency explicit but it doesn't let this problem — which is often subtle and idiosyncratic — overshadow everything else.

The proposed design strives for a balanced solution: on the silhouette surface it is very trigger happy in establishing equality, as the shortcut value is positioned very prominently. The underlying written out version on the other hand is extremely cautious and provides maximum control and flexibility. The user is able to choose in a very transparent way which annotations to consider when establishing equality: a user may take an extreme position and consider no two shortcut nodes equal, or may in the other extreme discard all annotations as inferior. An attentive user will take some route in between and conditionalize equality along the lines of specific annotations and depending on the use case at hand.

15.5.4.2 Internal Structure

Statement identifiers are IRIs. To account for various usage scenarios and requirements a statement identifier is structured into three parts:

source e.g. an RDF document on the web, a named graph in an RDF dataset
triple a specific occurrence of a statement
handle either a specific node (subject, predicate or object) or a complex object

The source is described by an IRI. The IRI may refer to an RDF document or a named graph in a dataset or any other resource addressable by that IRI. By default it has no other meaning than being the address from which the statement can be retrieved on the web. One syntactic constraint is imposed on source IRIs: they may contain query parameters but no fragment identifiers as those are reserved for the 'handle' part.

The triple identifier identifies an occurrence of a statement, not the abstract type as per the set semantics of RDF. The identifier can be generated or authored by hand. It has to adhere to the syntactic constraints on IRIs. Generated identifiers have to strive only for local uniqueness and therefore don't have to be very long. They can however not solely be a function of the triple they identify. The hash should begin with a string identifying the algorithm used and disambiguating it from hand authored identifiers. Choosing an appropriate hashing function is outside the scope of this proposal, but it will have to balance the need for usability and conciseness with the requirement for local uniqueness per triple occurrence. Hand authored identifiers will probably provide better usability but may not always be available, e.g. when a statement is annotated in a GUI that hides the identifier. Also having to mint them explicitly may be considered tedious in some use cases. When merging graphs, autogenerated identifiers are by default standardized apart (similar to bnodes) while annotations on hand authored identifiers are by default merged if the triples so identified are the same in both graphs. Otherwise they also need to be standardized apart.

A handle may amend the triple identifier. It's a string from a reserved vocabulary that serves two purposes: it may address the subject, predicate or object of the identified statement. Possible values are 'Subject'/'Sub'/'S', 'Predicate'/'Pre'/'P' and 'Object'/'Obj'/'O'. It may also address an algorithmically defined more complex object like e.g. a Concise Bounded Description (CBD). We will discuss such complex objects in more detail below.

The full statement identifier is composed by amending the source IRI with a query parameter containing the triple identifier, possibly followed by a # and a handle identifier, like so:

```
<source_IRI>?id=<triple_REF>#<handle_CLASS>
```

from which the following combinations can be used to refer to a statement in the local graph:

```
id:<triple_REF>
id:<triple_REF>#<handle_CLASS>
```

or in a globally unambiguous way:

```
<source_IRI>?id=<triple_REF>
<source_IRI>?id=<triple_REF>#<handle_CLASS>
```

The prefix `id:` will always resolve to the IRI of the local source, e.g. a document on the web or the name of a named graph in an RDF dataset. Consequently the triple identifier alone can be used to refer to a statement in the same source if it is prefixed by `id:`.

15.5.4.3 Syntax

This proposal benefits from proper support in syntaxes which is why we sketch extensions to Turtle and TriG below. But triple identifiers can also be used rather comfortably with unmodified Turtle (and by extension TriG) and using RDF-star instead of RDF standard reification. Only one extra triple is needed to declare an identifier if the `id:` attribute is not supported natively:

```
:s :p :o .
id:10 IS << :s :p :o >> ;
        :annotations :Galore .
```

Note that actually asserting (and endorsing) the triple requires an extra statement.

Annotating a triple in another source that provides no statement identifier requires defining an appropriate identifier first. The following example uses an RDF-star embedded triple instead of RDF standard reification:

```
id:11 IS << :s :p :o >> ;
        OF :SomeSourceIRI ;       // for more on the OF property see below
        AS sc:Denotation .        // identification semantics, see below
```

Here `id:11` addresses a triple in another source. Note that `< :s :p :o >` is again not actually asserted in this source (and may not be asserted in the other source as well).

15.5.4.4 Multiset Syntax and Semantics

As already mentioned the triple identifier disambiguates different occurrences of the same statement in a graph. As such it provides an important means to overcome the limitations imposed on RDF by its set based semantics. This corresponds to the way multiple occurrences of the same triple are declared with the RDF standard reification vocabulary. In our proposal those “multiset identifiers” are just syntactic sugar for the expanded blank node based representation, by which their meaning is well defined.

15.5.4.5 Syntactic Extensions

We propose to extend Turtle and TriG to Scenic Turtle and Scenic TriG by adding an optional triple identifier as a fourth element to each triple. Thereby

```
:Alice :buys :Car .
id:1 IS << :Alice :buys :Car >> .
id:1#Subject :age 21 .
id:1#Predicate :funds :Cash .
id:1#Object :age 12 .
id:1 :creator :Bob .
```

becomes

```
:Alice :buys :Car id:1 .
id:1#Subject :age 21 .
id:1#Predicate :funds :Cash .
id:1#Object :age 12 .
id:1 :creator :Bob .
```

A human readable identifier like the `id:` attribute that we use in our examples is preferable in human facing data, but the following example is just as valid:

```
:Alice :buys :Car http://someGraphIRI?id=someTripleID .
http://someGraphIRI?id=someTripleID#Subject :age 21 .
http://someGraphIRI?id=someTripleID#Predicate :funds :Cash .
http://someGraphIRI?id=someTripleID#Object :age 12 .
http://someGraphIRI?id=someTripleID :creator :Bob .
```

Note that triples that aren't referenced in other statements don't need to declare an identifier. That rule is valid also in other syntaxes that aim for succinct representations, like RDF/XML and JSON-LD. RDF/XML doesn't require any changes at the syntactic level as it already provides an `id:` attribute. However it is also important to know if the identifier abides by the structural constraints of Scenic RDF. A declaration at the top level of an RDF graph to that effect could provide some desirable reassurance when exchanging data.

15.5.5 Scenic Graphs

RDF defines an RDF graph as a set of RDF triples. A set is a mathematical abstraction and set theory is used to define the model theoretic semantics of RDF, as customary in that field. In practice however the Semantic Web is rather not a mathematical abstraction, but composed of snippets of RDF scattered all over the place and accessed from documents on the web, from named graphs in RDF databases, from IoT devices, streaming services and public query endpoints etc. The RDF specification refers to such concrete installations as 'sources', without providing any semantics to them. It treats them as a technical means to partition the Semantic Web into application specific sub spaces, and as such out of scope of the RDF semantics.

This situation is unsatisfactory insofar as it makes an important part of data modelling and governance — grouping data items and partitioning data spaces — inaccessible to RDF itself. Attempts to standardize semantics for sources like named graphs have failed in the past as different needs for grouping have different needs for semantics and it has proven impossible to standardize on just one of them. From this conundrum we draw a few conclusions that guide our proposal:

- grouping triples for technical, application-specific reasons is a legitimate use case and shouldn't be burdened with overly specific semantics,
- more elaborate use cases need means to explicitly declare semantics for groups of triples and
- absent any such declaration or other application specific fixing arrangements one can, if only tentatively, assume that a source follows the standard RDF semantics.

Named graphs as standardized in RDF and SPARQL have no model theoretic semantics, but RDF per default is composed of referentially transparent triples, where IRIs name the resource they return when addressed. Whatever semantics can be extracted from the operations of SPARQL adheres to a somewhat more specific, application centric intuition in which triples are understood as occurrences, differentiating the same triple in two different named graphs. In that spirit we treat RDF sources in the most unassuming way: as mere addresses of and access points to RDF statements which semantically are to be understood as interpreted occurrences of triples, where two occurrences of a triple refer to the same interpretation in the big, unified knowledge space that the Semantic Web puts up. Such a semantics should in our opinion form the basis for the definition of a graph in RDF.

To that end we introduce the concept of a Scenic Graph, or *scene* for short — so called to differentiate it from graphs, named graphs etc, but also from sources. We aim at a small vocabulary that allows to:

- group triples into Scenic Graphs to structure annotations,
- declare non-standard semantics on those Scenic Graphs, and
- upgrade existing sources to proper Scenic Graphs.

We want Scenic Graphs to be able to e.g. inherit annotations from other Scenic Graphs, thereby implementing nesting and other forms of composition. Scenic Graphs are identified by an IRI and have identity on their own. Two scenes may have the same attributes and be assigned to the same set of triples, but they still are different entities (unless of course an `owl:sameAs` statement says otherwise). Their semantics is by default that of regular RDF. Note that this is different from the semantics proposed in the original Named Graphs paper by [Car+05a], in which named graphs are also occurrences, but contrary to our approach are referentially opaque. Declaring other semantics such as referential opacity, CWA, UNA, type instead of occurrence etc requires:

- a vocabulary to explicitly declare semantics different from the standard RDF semantics, e.g.:

  ```
  :SomeScenicGraph sc:semantics sc:ReferentiallyOpaque .
  ```

- a means to declare the semantics of sources without semantics like named graphs, e.g.:

  ```
  <someNamedGraphIRI> a sc:Scene .
  ```

- As the `rdfs:domain` of `sc:semantics` is `sc:Scene`,

  ```
  <someNamedGraphIRI> sc:semantics sc:ReferentiallyTransparent ,
                                   sc:Occurrence .
  ```

  would lead to the same result.
- an encoding of RDF data that hides sources governed by other than the default semantics from systems that don't understand and adhere to such a semantics. RDF literals implemented as a datatype could serve such purposes, e.g.:

  ```
  ":s :p :o . :q :r :s ."^^rdf:turtle
  ```

- a mechanism to introduce such hidden data to systems that can handle it:

  ```
  [ IF sc:ReferentiallyOpaque ;
       sc:expands ":s :p :o ."^^rdf:turtle ]
  ```

Nesting, inheriting, 'lifting' or otherwise composing scene attributes from other scenes is another area of application that requires some vocabulary. We plan to take [GMF04] as a starting point for further work.

We'd like to stress once more the guiding design principle behind Scenic Graphs: the most intuitive and main stream semantics should be the default while more elaborate needs should be satisfiable with little extra effort. Grouping is a very basic tool for information modelling, an ubiquitous need, and named graphs as the only means available in RDF databases to group statements can't be burdened with any other purpose. The original Named Graphs paper overloaded this grouping mechanism with more specific demands and corresponding semantics. Rather unsurprisingly the result was that while the syntax caught on the semantics in practice was ignored. With Scenic Graphs we have a clean slate, but we can't afford to make the same mistake again. This time the default is explicitly the most unsurprising option: a scene is by default a referentially transparent occurrence.

15.5.5.1 Sources and Scenes

Not everything is smooth sailing though — differentiating physical from logical grouping devices has some interesting properties.

Let's assume some triple is annotated with `sc:inScene` membership and also with other annotations that themselves are not explicitly declared to belong to any Scenic Graph. One might ask if those other annotations annotate the triple 'itself' or the triple 'in some scene'. The answer is that all annotations, `sc:inScene` like any other, are in the same league. Belonging to some scene is a property like any other and it annotates the blank nodes that form the core of the expanded representation. Some applications like e.g. declaring a non-standard semantics may profit from or even depend on addressing a set of statements by source rather than annotating each statement with `sc:inScene` attributes individually. As is often the case with formal semantics this behaviour may be seen as welcome or problematic, depending on the situation and use case.

Sadly it's impossible to devise a simple scheme that covers all aspects, niches and corner cases.

Another question that may come up is what happens if a statement is annotated with two `sc:inScene` attributes: does it belong to two scenes — thought of as sources — at the same time? Again the answer is that scenes, although a grouping device, are not locations. A statement belonging to two Scenic Graphs has the attributes of both — nothing more, nothing less. A user may then request statements that share all those attributes or at least some of them or none of them at all. Declaring that a statement either belongs to one Scenic Graph or another would require to create two different occurrences of the statement and annotate them accordingly.

And what happens when two sources contain the same statement, e.g.:

```
<someNamedGraphIRI>?id=<theOneTriple>
<someOtherNamedGraphIRI>?id=<theOneTriple>
```

— do the two identifiers refer to the same statement? We wouldn't want that to be the case per se as it would seem unintuitive in many situations, so: no. And as stated above an automatic merge would need to standardize those identifiers apart. It was also said above that the source IRI has no meaning unless and until it is declared to be of type Scenic Graph but that wasn't completely correct. A source obviously has *some* meaning and if it's only that it is a specific source. The meaning of a source IRI is its ability to address, or put a little more formally:

```
<someNamedGraphIRI> sc:addressOf <someNamedGraphIRI> .
```

Anything more definitive concerning the meaning of those sources of triples has to be declared explicitly in further RDF statements. It is at the discretion of the user to treat those two statement identifiers as referring to the same triple because they represent the same statement or to treat them as referring to different statements because they come from different sources.

15.5.5.2 Storage: Quad, Quad-and-a-half, Quint

A superficial look over the installed base of Semantic Web enabled databases may give the impression that everybody uses a quad store where in addition to the subject, predicate and object that form an RDF statement a fourth element is used to record the named graph in which a statement appears. However the situation is not so clear cut in reality. Some RDF stores, e.g. AllegroGraph and Stardog, support quins, storing a statement identifier and a graph with each triple. Some RDF stores like Virtuoso are technically based on relational databases and therefore can support quins with relatively little effort by adding another column to their table. Some RDF stores like Jena internally provide row level identifiers that could be made accessible externally. Some RDF stores like Amazon Neptune and the analytics tool KGTK use the fourth column to store not the graph but a statement identifier, effectively turning each statement into a singleton named graph. Consequently it might well be possible to support the proposed statement identifier on a broad range of systems without requiring a prohibitive amount of work and effort, maybe even less than say supporting RDF-star (which, per anecdotal evidence, is often implemented via triple identifiers). However, the more interesting and also imperative to pursue question is how well the proposed statement identifier can be supported in regular SPARQL dataset enabled quad stores, which represent without doubt the backbone and common denominator of today's Semantic Web. Any solution will have to provide means to:

- declare a statement identifier,
- understand handles addressing individual nodes and complex objects,
- declare scene membership and
- declare source membership, preserving named graphs functionality.

The most unassuming solution is to declare statement identifiers in extra triples, leaving all established implementations and machinery untouched:

```
:g { :s :p :o .
    :q :r :s .
    id:ex IS << :s :p :o >> ; \todo{RDF-star is NOT part of the established machinery}
        :creator :Bob ;
        sc:inScene :S_1 ,
                    :S_2 .
}
:g sc:inScene :S_3 ;
   a sc:Scene .                    // not necessary, but maybe helpful
```

Obviously this solution leads to more triples than one might wish for, but for moderate annotation demands it might be a worthwhile approach. Note that annotating all triples in the source graph at once is easily available and doesn't require any extra reification steps.

Another approach repurposes the fourth element in a quad store as a statement identifier, effectively turning all triples into singleton graphs. Now source membership will have to be declared explicitly, just like scene membership. In the following example :s1, :s2, :s3 . . . represent identifiers of the singleton triple:

```
:s1        { :s :p :o }
:s2        { :q :r :s }
:s3        { :s1 :creator :Bob }
:s4        { :s1 sc:inScene :S_1 }
:s5        { :s1 sc:inScene :S_2 }
:s6        { :s1 sc:inScene :S_3 }
:s7        { :s2 sc:inScene :S_3 }
:s8        { :s3 sc:inScene :S_3 }
:s9        { :s4 sc:inScene :S_3 }
:s10       { :s5 sc:inScene :S_3 }
:s11       { :s1 sc:inSource :g }
:s12       { :s2 sc:inSource :g }
:s13       { :s3 sc:inSource :g }
:s14       { :s4 sc:inSource :g }
:s14       { :s5 sc:inSource :g }
```

This quickly becomes verbose and unintuitive. We would get even more singleton graphs if we had to also declare membership in the default graph of a dataset explicitly.

A third approach takes advantage of the fact that each triple is tied to exactly one source. Consequently source identifier and statement identifier can both be stored in the same field. One could call this a quad.5 or quad-and-a-half approach.

```
:g?s1       { :s :p :o }
:g?s2       { :q :r :s }
:g?s3       { :s1 :creator :Bob }
:g?s4       { :s1 sc:inScene :S_1 }
:g?s5       { :s1 sc:inScene :S_2 }
:g?s6       { :s1 sc:inScene :S_3 }
:g?s7       { :s2 sc:inScene :S_3 }
:g?s8       { :s3 sc:inScene :S_3 }
:g?s9       { :s4 sc:inScene :S_3 }
:g?s10      { :s5 sc:inScene :S_3 }
```

This approach requires parsing source identifiers and will probably need extra optimization efforts in indexing. It is less readable than the first approach, but has about the same triple count. Certainly triple count can change quickly with more involved examples. Readability is not the most important consideration as the syntax of this approach is much closer to the metal than the first approach. It can easily be represented like the first approach.

In a quin store statement identifiers can be stored separately from the source identifier, thus avoiding the need for costly parsing operations. However annotating a statement with scene membership still requires an extra statement. One could also extend the quad.5 approach to quins and store the statement identifier together with the source as the fourth element, the scene however in the fifth attribute. Multiple `:inScene` annotations on one and the same annotated statement would then require to consolidate them in one new, combined scene, to avoid having to duplicate the annotated statement including all other annotations besides the `:inScene` annotation (as that could too easily lead to ambiguous semantics). This approach should be investigated further.

We only hint at yet another approach which would be to extend the RDF-star syntax by a fourth element to record a source. That would allow to refer to local occurrences as well as to those in other sources and much like the third approach it stuffs both aspects - statement and source - into one identifier, again requiring extra parsing effort. The RDF-star embedded triple is quite readable but verbose. It also doesn't scale well for repeated annotations on annotations when e.g. tracking multiple transformation steps in an ETL pipeline.

15.5.6 Complex Objects

We need ways to address a triple with all its annotations, similar to how e.g. Concise Bound Descriptions address groups of connected statements. This is a way of grouping independent of scenes and more fine-grained than graphs. For silhouettes as a special kind of object such a device might also help to answer the question of whether or not an annotation annotates only the silhouette triple itself or if it includes the silhouette as well as all its annotations.

Labelled Property Graphs do not only annotate relations but also nodes with secondary information. In effect this leads to a differentiation between relations that connect nodes and relations that describe nodes, in other words local and global relations, or in yet other words:

objects (with properties) and
relations (with properties) between those objects

We can replicate that expressivity if we treat silhouette relations as those that connect objects whereas all other relations describe those objects and are somehow 'integral' to them. We can even annotate silhouette relations as specific for a certain use case, e.g. 'navigation' or 'topic' relations that connect areas, 'object' relations, 'shortcut' relations etc. This is much more flexible and expressive than the LPG approach. Of course such flexibility is not always a boon, but can be overwhelming as well. Defining appropriate algorithms — starting from Concise Bounded Descriptions, on to lists, tables, tress, shapes etc. — will be the topic of future work.

15.5.7 Basic Vocabulary

We have introduced a few basic vocabulary terms in the discussions and examples above that shall in the following be more properly defined.

Basics

```
sc:ScenicRDF a rdfs:Class ;
    rdfs:comment "declares that identifiers follow Scenic RDF rules" .
sc:expands a rdf:Property ;
    rdfs:comment "interprets an RDF literal + imports it into a graph" .
```

Source

```
sc:Source a rdfs:Class ;
    rdfs:comment "May be a document, a named graph" .
sc:inSource a rdf:Property ;
    rdfs:range sc:Source .
```

Scene

```
sc:Scene a rdfs:Class ;
    rdfs:comment "aka Scenic Graph" .
sc:inScene a rdf:Property ;
    rdfs:range sc:Scene ;
    rdfs:comment "to declare membership in a scene" .
```

Nodes

```
sc:Subject a rdfs:Class ;
    rdfs:comment "addresses the subject of a triple" .
sc:Predicate a rdfs:Class ;
    rdfs:comment "addresses the predicate of a triple" .
sc:Object a rdfs:Class ;
    rdfs:comment "addresses the object of a triple" .
rdf:subject rdfs:range sc:Subject .
rdf:predicate rdfs:range sc:Predicate .
rdf:object rdfs:range sc:Object .
```

Complex Objects

```
sc:CBD a rdfs:Class ;
    rdfs:comment "addresses a Concise Bounded Description" .
```

Semantics

```
sc:semantics a rdf:Property ;
    rdfs:domain sc:Scene ;
    rdfs:comment "defines semantics of a scene" .
```

Defining terms to specify semantics and semantic properties is future work. Possible terms are e.g. `sc:StandardRDFSemantics`, `sc:ReferentiallyTransparent`, `sc:ReferentiallyOpaque`, `sc:Type`, `sc:Occurrence` . The issue is discussed in more detail in Section 7.

15.5.8 Structural Vocabulary

This proposal so far concentrated on statement annotation and LPG style modelling, which is also the basis for graph containment and contextualization. However there are also other nagging problems on the Semantic Web and in addition to the basic vocabulary terms this proposal introduces some properties that target RDF's lack of expressivity at the structural level and addresses some of its fundamental issues like:

– proper identification (the "identity crisis")
– a need for modelling primitives beyond the triple, like lists, nested lists, n-ary relations and even more complex objects
– quoted representations, avoiding the vaguenesses of interpretation (and favouring CWA application semantics over OWA integration facilitation)

All these properties serve structural, logical and modelling purposes. There exist other prominent attributes like time, location and provenance but those are covered by regular vocabularies already. Some special annotation types are named by reserved keywords as explained below. We already saw `IS`, the property central to this proposal, but there is more:

IS a stricter version of `rdf:valueOf`
AS disambiguates identification semantics through a notion of role
IN provides pointers from nodes into lists that contain them
OF provides pointers from statements and nodes into complex objects like CBDs or RDBMS
BY documents entailments
CC links to quoted syntactical representation
IF a stub for some sort of scripting extension

These properties shall now be introduced in some more detail.

IS

`IS` was already introduced and is the property most central to this proposal. It is defined as a subproperty of `rdf:valueOf`, with the same meaning — primary value — but a stricter interpretation: whenever a node is an object of an `IS` statement its meaning is only fully described by taking into account all information from the statements with which it shares a common subject via the `IS` relation.

The subject of an `IS` relation can be attributed in any way deemed useful. It is the real subject of the relation to which the shortcut relation just provides the more usable surface. The object of an `IS` relation is a shortcut in a silhouette statement and can only be considered as a hint to the real thing. That real thing is the blank node that is subject of the `IS` relation.

```
_:b IS :Alice ;
    :age 8 ;
    :meets :CheshireCat
```

If more than one node is related to a subject via an `IS` relation then the resulting meaning is not the intersection or combination of the meanings of those nodes but rather something that could be described as a local `owl:sameAs` relation: all those nodes have the same meaning in the context of all other annotations attributed to the subject of the `IS` relations, e.g.:

```
_:b IS :Alice ,
       :ChildAlice ;
    :age 8 .
```

AS

`AS` addresses a long standing problem at the very base of RDF: disambiguating the various forms of identification that an IRI supports. Possible values of `AS` are `:Denotation` and `:Indication` but others may be defined as needed. In the following example the `AS` relation declares that the IRI `:Alice` is used to refer to the person Alice that is subject of the resource addressed by the IRI `:Alice`:

```
_:b IS :Alice ;
    AS :Denotation .
```

However if we wanted to annotate Alice's website of which that very IRI addresses the homepage, we could employ some more elaborate disambiguation like so:

```
_:b IS :Alice ;
    AS :Indication , :Website .
```

IN

`IN` is a 'shortcut relation' (like OF, discussed next) used to express that the node is part of a list. This is useful to model a frequent type of n-ary relations where one node is actually a list of nodes, like e.g. a group of friends, listed by name, that together go to the movies:

```
_:b IS :Alice ;
    IN _:c .
_:c a rdf:List ;
    rdf:first :Alice ;
    rdf:rest ...                // more list members
```

The advantage of such a construct in practice is that it makes it easy to query if Alice went viewing a certain movie and subsequently retrieve further detail, namely that she didn't go alone. Without the shortcut relation one would have to query if Alice went alone *and* if she went as part of a group, a much more tedious query task.

Yet another open question is if also RDF container types and possible future extensions to the RDF list machinery like native list objects should be possible targets of the `IN` relation. Probably: yes.

OF

`OF` is the second shortcut relation, targeting complex objects like tree-, star- or snowflake-structures, algorithmically defined objects like Concise Bounded Descriptions, relational data mapped via RML etc. `OF` can be used predominantly on whole statements but also, like `IN`, on nodes.

Using `OF` on statements is useful to introduce prominent navigational statements that tap into much more complex structures, making them more accessible to a user that is not familiar with a given knowledge space. For example, OWL safe data ontology patterns can get very involved and need expert knowledge to navigate and query. Seemingly ordinary objects like a train itinerary can become very convoluted and confusing. A simple statement, connected to the full extent of data by an `OF` relation, can help to provide easy access and entry points, e.g.:

```
:Alice :travelsTo :Brussels id:7 .
id:7 OF [ IS :TrainNo5 ;
            a :CBD ] .
:TrainNo5 :start :Paris ;
            :at "5.30 am" ...
```

`OF` relations can provide entry points into n-ary relations imported from an RDBMS, emphasizing the most important aspects in a table. Another possible application scenario is to model n-ary relations in the most generic way, as neo-Davidsonian n-ary relations, making them verbose but easy to construct and query, and deliberately rely on `OF` relations to create a usability-focused layer of navigational, user-facing relations on top of and 'through' them. This provides a new interpretation of Shneiderman's data visualization mantra: *Overview first, zoom in, detail on demand.*[6] `OF` relations also help dealing with the problem that it it is often very subjective and situation dependent if some extra information is understood as internal detail of a topic or external context. The `OF` relation takes no side in this dichotomy, but just provides a link to additional information.

Using `OF` on nodes is slightly tricky and should be done only when the node and related nodes in the target object are of the same and also quite specific type. Otherwise the reference probably wouldn't make much sense.

BY

`BY` links a node or statement to another node or statement that it is derived from by entailment. It can be used e.g. to investigate the origin of some assertion or to roll back changes and their entailments. To record further details like the entailment regime `BY` can be combined with `IS`, where `IS` identifies the primary value and another relation adds further detail like the entailment regime, e.g.:

```
_:b IS :Alice
    BY [ IS :AliceSprings ;
            :entailmentRegime :OWL ;
            :entailmentRelation owl:sameAs ] .
```

CC

`CC` (for Carbon Copy) links a node, statement, graph or complex object to its original literal representation. This can be useful to record specific states of some piece of RDF for versioning purposes or establish fine grained control and documentation of entailment regimes and entailments made. It can also be used to replace the RDF standard reification quad, e.g. instead of:

```
_:1 a rdf:Statement ;
    rdf:subject :s ;
    rdf:predicate :p ;
    rdf:object :o .
```

just write:

```
_:1 CC ":s :p :o ."^^rdf:turtle
```

The value of a `CC` relation is a literal of datatype RDF.[7] This omits the need for resorting to RDF-star quoted triples, as employed in some examples above.

IF

`IF` is an extension point with the purpose of embedding scripted instructions, SPARQL filter expressions, predefined rules or other dynamic content like

[6] [Shn96]
[7] See Section 4.2.1.2

LDscript[8] and SPIN[9] into RDF. Details still need to be fleshed out, but the general idea is that the annotated resource — probably a statement — should only be considered, used, returned from a query etc `IF` specific conditions are met:

```
:Alice a :happyPerson id:1 .
id:1 IF :goodWeather .
```

However this runs counter to the monotonicity principle of RDF and even applications that are aware of the `IF` extension might have problems implementing it correctly e.g. with streaming data sources. To guarantee monotonicity RDF literals could be used, e.g.:

```
":Alice a :happyPerson ."^^rdf:turtle IF :goodWeather .
```

that only get expanded into regular RDF statements if the condition is met.

`IF` can control individual nodes only if it provides a default value as otherwise the node and by extension the containing statement would initially be in an undefined state.

```
:Alice a [ IF [ IS [ CC ":nicePerson"^^rdf:turtle ] ;
                :weatherReport [ :forecasts :goodWeather ;
                                     IS :happyPerson ] ,
                               [ :forecasts :badWeather ;
                                     IS :miserablePerson ] ] ] .
```

Here `IF-IS-CC` automatically converts the quoted literal into its interpreted representation which may or may not be replaced by the following instructions before the resulting node is being added to the graph. To an RDF application that doesn't understand the `IF` construct all this is of course just well formed gibberish.

[8][CFG17]

[9]`https://www.w3.org/Submission/spin-overview/`

Bibliography

[22] *The Semantic Web - ISWC 2022 - 21st International Semantic Web Conference, Virtual Event, October 23-27, 2022, Proceedings.* 2022. ISBN: 978-3-031-19432-0. URL: `https://doi.org/10.1007/978-3-031-19433-7` (cit. on p. 86).

[Abu+22] Ghadeer Abuoda et al. “Transforming RDF-star to Property Graphs: A Preliminary Analysis of Transformation Approaches–extended version”. In: *arXiv preprint arXiv:2210.05781* (2022) (cit. on pp. 159, 164).

[ADA15] Anastasia Analyti, Carlos Viegas Damásio, and Grigoris Antoniou. “Extended RDF: Computability and complexity issues”. In: *Ann. Math. Artif. Intell.* (2015). URL: `https://doi.org/10.1007/s10472-015-9451-0` (cit. on p. 233).

[ADP15] Anastasia Analyti, Carlos Viegas Damásio, and Ioannis Pachoulakis. “Nested contextualised views in the web of data”. In: *Int. J. Web Eng. Technol.* (2015). URL: `https://doi.org/10.1504/IJWET.2015.069360` (cit. on pp. 224, 226, 228).

[AF19] Alessandro Artale and Enrico Franconi. “Towards a Logical Foundation of Reification in Modelling Languages”. In: *Ontology Makes Sense - Essays in honor of Nicola Guarino.* 2019. URL: `https://doi.org/10.3233/978-1-61499-955-3-242` (cit. on pp. 110, 231).

[AG08a] Renzo Angles and Claudio Gutiérrez. “Survey of graph database models”. In: *ACM Comput. Surv.* (2008). URL: `https://doi.org/10.1145/1322432.1322433` (cit. on p. 243).

[AG08b] Renzo Angles and Claudio Gutiérrez. “The Expressive Power of SPARQL”. In: *The Semantic Web - ISWC 2008, 7th International Semantic Web Conference, ISWC 2008, Karlsruhe, Germany, October 26-30, 2008. Proceedings.* 2008. URL: `https://doi.org/10.1007/978-3-540-88564-1%5C_8` (cit. on p. 70).

[AG16] Renzo Angles and Claudio Gutiérrez. “The multiset semantics of SPARQL patterns”. In: *CoRR* (2016). URL: `http://arxiv.org/abs/1610.04315` (cit. on p. 214).

[Ali+22] Waqas Ali et al. “A survey of RDF stores & SPARQL engines for querying knowledge graphs”. In: *VLDB J.* (2022). URL: `https://doi.org/10.1007/s00778-021-00711-3` (cit. on p. 235).

[All+03] Heidrun Allert et al. “Role-oriented models for hypermedia construction-conceptual modeling for the semantic web”. In: *Learning* (2003) (cit. on p. 204).

[Ana+11] Anastasia Analyti et al. “Extended RDF as a Semantic Foundation of Rule Markup Languages”. In: *CoRR* (2011). URL: `http://arxiv.org/abs/1111.0055` (cit. on p. 233).

[Ana+13] Anastasia Analyti et al. “A framework for modular ERDF ontologies”. In: *Ann. Math. Artif. Intell.* (2013). URL: `https://doi.org/10.1007/s10472-013-9350-1` (cit. on pp. 224, 233).

[Ana+14] Anastasia Analyti et al. “Why-provenance information for RDF, rules, and negation”. In: *Ann. Math. Artif. Intell.* (2014). URL: `https://doi.org/10.1007/s10472-013-9396-0` (cit. on pp. 173, 348).

[And19] James Anderson. “RDF Graph Stores as Convergent Datatypes”. In: *Companion of The 2019 World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*. 2019. URL: `https://doi.org/10.1145/3308560.3316517` (cit. on pp. 115, 175).

[Ang+17] Renzo Angles et al. “Foundations of Modern Query Languages for Graph Databases”. In: *ACM Comput. Surv.* (2017). URL: `https://doi.org/10.1145/3104031` (cit. on pp. 55, 213, 355).

[Ang+22] Renzo Angles et al. “Multilayer graphs: a unified data model for graph databases”. In: *GRADES-NDA ’22: Proceedings of the 5th ACM SIGMOD Joint International Workshop on Graph Data Management Experiences & Systems (GRADES) and Network Data Analytics (NDA), Philadelphia, Pennsylvania, USA, 12 June 2022*. 2022. URL: `https://doi.org/10.1145/3534540.3534696` (cit. on pp. 154, 237, 239).

[Ang09] Renzo Angles. “A Nested Graph Model for Visualizing RDF Data”. In: *Proceedings of the 3rd Alberto Mendelzon International Workshop on Foundations of Data Management, Arequipa, Peru, May 12-15, 2009*. 2009. URL: `http://ceur-ws.org/Vol-450/paper10.pdf` (cit. on p. 244).

[AP08] Anastasia Analyti and Ioannis Pachoulakis. “A survey on models and query languages for temporally annotated RDF”. In: *International Journal of Advanced Computer Science & Applications* (2008) (cit. on p. 225).

[AP18] Abdullah Abbas and Gilles Privat. “Bridging Property Graphs and RDF for IoT Information Management”. In: *Proceedings of the 12th International Workshop on Scalable Semantic Web Knowledge Base Systems co-located with 17th International Semantic Web Conference, SSWSISWC 2018, Monterey, California, USA, October 9, 2018*. 2018. URL: `http://ceur-ws.org/Vol-2179/SSWS2018_paper6.pdf` (cit. on pp. 162, 192).

[Are+12] Marcelo Arenas et al. “A direct mapping of relational data to RDF”. In: *W3C recommendation* (2012) (cit. on p. 210).

[Arn19] Dörthe Arndt. “Notation3 as the unifying logic for the semantic web”. PhD thesis. Ghent University, 2019 (cit. on p. 228).

[Art+17] Alessandro Artale et al. “A Decidable Very Expressive Description Logic for Databases”. In: *The Semantic Web - ISWC 2017 - 16th International Semantic Web Conference, Vienna, Austria, October 21-25, 2017, Proceedings, Part I*. 2017. URL: `https://doi.org/10.1007/978-3-319-68288-4_3` (cit. on p. 230).

[Asp+19] Luigi Asprino et al. “Observing LOD Using Equivalent Set Graphs: It Is Mostly Flat and Sparsely Linked”. In: *The Semantic Web - ISWC 2019 - 18th International Semantic Web Conference, Auckland, New Zealand, October 26-30, 2019, Proceedings, Part I*. 2019. URL: `https://doi.org/10.1007/978-3-030-30793-6_4` (cit. on pp. 34, 119).

[Ata10] Vassil Momtchev Atanas Kiryakov. *Triplesets: Tagging and Grouping in RDF Datasets*. 2010. URL: `https://www.w3.org/2009/12/rdf-ws/papers/ws24` (visited on 08/11/2022) (cit. on pp. 168, 174).

[ATP20] Amr Azzam, Ruben Taelman, and Axel Polleres. “Towards Cost-Model-Based Query Execution over Hybrid Linked Data Fragments Interfaces”. In: *The Semantic Web: ESWC 2020 Satellite Events - ESWC 2020 Satellite Events, Heraklion, Crete, Greece, May 31 - June 4, 2020, Revised Selected Papers*. 2020. URL: `https://doi.org/10.1007/978-3-030-62327-2_2` (cit. on p. 180).

[ATT20] Renzo Angles, Harsh Thakkar, and Dominik Tomaszuk. “Mapping RDF Databases to Property Graph Databases”. In: *IEEE Access* (2020). URL: `https://doi.org/10.1109/ACCESS.2020.2993117` (cit. on p. 163).

[AU17] Marcelo Arenas and Martín Ugarte. “Designing a Query Language for RDF: Marrying Open and Closed Worlds”. In: *ACM Trans. Database Syst.* (2017). URL: `https://doi.org/10.1145/3129247` (cit. on p. 214).

[AW19] Dörthe Arndt and William Van Woensel. “Towards Supporting Multiple Semantics of Named Graphs Using N3 Rules”. In: *Proceedings of the 13th RuleML+RR 2019 Doctoral Consortium and Rule Challenge, September 16-19, 2019 - Bolzano, Italy co-located with 3rd International Joint Conference on Rules and Reasoning (RuleML+RR 2019) 5th Global Conference on Artificial Intelligence (GCAI 2019) 15th Reasoning Web Summer School (RW 2019) DecisionCAMP 2019 (DecisionCAMP 2019), Bolzano, Italy, September 16-24, 2019*. 2019. URL: http://ceur-ws.org/Vol-2438/paper6.pdf (cit. on pp. 86, 158, 199, 228, 232).

[Azz+21] Amr Azzam et al. “WiseKG: Balanced Access to Web Knowledge Graphs”. In: *WWW ’21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021*. 2021. URL: https://doi.org/10.1145/3442381.3449911 (cit. on p. 180).

[Bak+13] Thomas Baker et al. “Key choices in the design of Simple Knowledge Organization System (SKOS)”. In: *J. Web Semant.* (2013). URL: https://doi.org/10.1016/j.websem.2013.05.001 (cit. on p. 67).

[Bao+10a] Jie Bao et al. “A formal context representation framework for network-enabled cognition”. In: (2010) (cit. on pp. 117, 247).

[Bao+10b] Jie Bao et al. “Context representation for the semantic web”. In: (2010) (cit. on pp. 118, 198, 224, 232).

[Bat+14] Colin R. Batchelor et al. “Scientific Lenses to Support Multiple Views over Linked Chemistry Data”. In: *The Semantic Web - ISWC 2014 - 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part I*. 2014. URL: https://doi.org/10.1007/978-3-319-11964-9_7 (cit. on pp. 114, 115, 187, 203, 247).

[Bat+17] Sotiris Batsakis et al. “Temporal representation and reasoning in OWL 2”. In: *Semantic Web* (2017). URL: https://doi.org/10.3233/SW-160248 (cit. on pp. 143, 226).

[BC08] Tim Berners-Lee and Dan Connolly. *Notation 3 (N3): a readable RDF syntax*. 2008. URL: http://www.w3.org/TeamSubmission/n3/ (visited on 08/23/2022) (cit. on p. 195).

[BCH06] Jie Bao, Doina Caragea, and Vasant G. Honavar. “Package-Based Description Logics - Preliminary Results”. In: *The Semantic Web - ISWC 2006, 5th International Semantic Web Conference, ISWC 2006, Athens, GA, USA, November 5-9, 2006, Proceedings*. 2006. URL: https://doi.org/10.1007/11926078%5C_72 (cit. on pp. 222, 231).

[Bee+18] Wouter Beek et al. “sameAs.cc: The Closure of 500M owl: sameAs Statements”. In: *The Semantic Web - 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3-7, 2018, Proceedings.* 2018. URL: https://doi.org/10.1007/978-3-319-93417-4_5 (cit. on pp. 185, 189, 201, 202, 203).

[Bel+12] Khalid Belhajjame et al. “Workflow-Centric Research Objects: A First Class Citizen in the Scholarly Discourse.” In: *SePublica ESWC.* 2012 (cit. on p. 170).

[Ber+07] Tim Berners-Lee et al. “N3Logic: A Logical Framework For the World Wide Web”. In: *CoRR* (2007). URL: http://arxiv.org/abs/0711.1533 (cit. on pp. 77, 168, 224, 228).

[Ber02] Tim Berners-Lee. *What do HTTP URIs Identify?* 2002. URL: https://www.w3.org/DesignIssues/HTTP-URI.html (visited on 08/23/2022) (cit. on pp. 30, 32).

[Ber09] Tim Berners-Lee. *A Short History of "Resource" in web architecture.* 2009. URL: https://www.w3.org/DesignIssues/TermResource.html (visited on 08/23/2022) (cit. on pp. 29, 31, 47).

[Ber96a] Tim Berners-Lee. *Generic Resources – Axioms of Web Architecture.* 1996. URL: https://www.w3.org/DesignIssues/Generic.html (cit. on p. 201).

[Ber96b] Tim Berners-Lee. *Universal Resource Identifiers – Axioms of Web Architecture.* 1996. URL: https://www.w3.org/DesignIssues/Axioms.html (cit. on p. 30).

[BFL09] François Bry, Tim Furche, and Benedikt Linse. “Model theory and entailment rules for rdf containers, collections and reification”. In: (2009) (cit. on p. 232).

[BGM14] Dan Brickley, Ramanathan V. Guha, and Brian McBride. *RDF Schema 1.1.* 2014. URL: https://www.w3.org/TR/rdf-schema/ (visited on 09/01/2022) (cit. on pp. 41, 227).

[BGS13] Loris Bozzato, Chiara Ghidini, and Luciano Serafini. “Comparing contextual and flat representations of knowledge: a concrete case about football data”. In: *Proceedings of the 7th International Conference on Knowledge Capture, K-CAP 2013, Banff, Canada, June 23-26, 2013.* 2013. URL: https://doi.org/10.1145/2479832.2479842 (cit. on pp. 208, 228, 355, 358).

[BHS12] Loris Bozzato, Martin Homola, and Luciano Serafini. “Context on the semantic web: Why and how”. In: *ARCOE-12* (2012) (cit. on pp. 118, 223).

[Big+20] Alexander Bigerl et al. “Tentris–A Tensor-Based Triple Store”. In: *International Semantic Web Conference.* Springer. 2020 (cit. on pp. 146, 238, 241, 271).

[BK12] Robert Battle and Dave Kolas. "Enabling the geospatial Semantic Web with Parliament and GeoSPARQL". In: *Semantic Web* (2012). URL: `https://doi.org/10.3233/SW-2012-0065` (cit. on p. 216).

[Bol19] Jerven Bolleman. *Can SA be extension of RDF+reification? [was Re: PG mode and SA mode]*. 2019. URL: `https://lists.w3.org/Archives/Public/public-rdf-star/2019Sep/0081.html` (visited on 10/12/2022) (cit. on p. 152).

[Bon+18] Piero Andrea Bonatti et al. "Knowledge Graphs: New Directions for Knowledge Representation on the Semantic Web (Dagstuhl Seminar 18371)". In: *Dagstuhl Reports* (2018). URL: `https://doi.org/10.4230/DagRep.8.9.29` (cit. on p. 210).

[Bou+04] Paolo Bouquet et al. "Contextualizing ontologies". In: *J. Web Semant.* (2004). URL: `https://doi.org/10.1016/j.websem.2004.07.001` (cit. on pp. 117, 353).

[Bou+17] Jean-Rémi Bourguet et al. "Empirically Evaluating Three Proposals for Representing Changes in OWL2". In: *Proceedings of the Joint Ontology Workshops 2017 Episode 3: The Tyrolean Autumn of Ontology, Bozen-Bolzano, Italy, September 21-23, 2017*. 2017. URL: `http://ceur-ws.org/Vol-2050/DEW_paper_4.pdf` (cit. on pp. 225, 226).

[BP11] Sotiris Batsakis and Euripides G. M. Petrakis. "Representing temporal knowledge in the semantic web: The extended 4d fluents approach". In: *Combinations of intelligent methods and applications*. 2011 (cit. on p. 226).

[Bre+12] Christian Y. A. Brenninkmeijer et al. "Scientific Lenses over Linked Data: An Approach to Support Task Specific Views of the Data. A Vision". In: *Proceedings of the Second International Workshop on Linked Science 2012 - Tackling Big Data, Boston, MA, USA, November 12, 2012*. 2012. URL: `http://ceur-ws.org/Vol-951/paper5.pdf` (cit. on pp. 187, 247).

[Bri14] Dan Brickley. *Roles in schema.org - final draft*. 2014. URL: `https://www.w3.org/wiki/images/c/c8/RolesinSchema.orgMay8.pdf` (visited on 09/01/2022) (cit. on pp. 204, 205).

[Bru+05] Jos de Bruijn et al. "OWL DL vs. OWL flight: conceptual modeling and reasoning for the semantic Web". In: *Proceedings of the 14th international conference on World Wide Web, WWW 2005, Chiba, Japan, May 10-14, 2005*. 2005. URL: `https://doi.org/10.1145/1060745.1060836` (cit. on pp. 69, 232).

[Bru+21] Julian Bruyat et al. "PREC: semantic translation of property graphs". In: *CoRR* (2021). URL: `https://arxiv.org/abs/2110.12996` (cit. on p. 163).

[BS03] Alexander Borgida and Luciano Serafini. “Distributed Description Logics: Assimilating Information from Peer Sources”. In: *J. Data Semant.* (2003). URL: `https://doi.org/10.1007/978-3-540-39733-5%5C_7` (cit. on p. 222).

[BS13] Loris Bozzato and Luciano Serafini. “Materialization Calculus for Contexts in the Semantic Web”. In: *Informal Proceedings of the 26th International Workshop on Description Logics, Ulm, Germany, July 23 - 26, 2013.* 2013. URL: `http://ceur-ws.org/Vol-1014/paper_51.pdf` (cit. on p. 228).

[BS20] Loris Bozzato and Christoph G. Schuetz. “Towards Distributed Contextualized Knowledge Repositories for Analysis of Large-Scale Knowledge Graphs”. In: *Proceedings of the 35th Italian Conference on Computational Logic - CILC 2020, Rende, Italy, October 13-15, 2020.* 2020. URL: `http://ceur-ws.org/Vol-2710/short1.pdf` (cit. on p. 228).

[BSC17] Loris Bozzato, Luciano Serafini, and Gaetano Calabrese. “CKR: Live Demo: Using Contexts and Exceptions for Representing Evolving Knowledge States.” In: *Description Logics.* 2017 (cit. on p. 228).

[BSE18] Loris Bozzato, Luciano Serafini, and Thomas Eiter. “Reasoning with Justifiable Exceptions in Contextual Hierarchies”. In: *Principles of Knowledge Representation and Reasoning: Proceedings of the Sixteenth International Conference, KR 2018, Tempe, Arizona, 30 October - 2 November 2018.* 2018. URL: `https://aaai.org/ocs/index.php/KR/KR18/paper/view/18032` (cit. on p. 228).

[BSG+07] Paolo Bouquet, Heiko Stoermer, Daniel Giacomuzzi, et al. “OKKAM: Enabling a Web of Entities.” In: *I3* (2007) (cit. on p. 184).

[BSH16] Wouter Beek, Stefan Schlobach, and Frank van Harmelen. “A Contextualised Semantics for owl: sameAs”. In: *The Semantic Web. Latest Advances and New Domains - 13th International Conference, ESWC 2016, Heraklion, Crete, Greece, May 29 - June 2, 2016, Proceedings.* 2016. URL: `https://doi.org/10.1007/978-3-319-34129-3_25` (cit. on pp. 187, 220).

[BSH18] Marco Brandizi, Ajit Singh, and Keywan Hassani-Pak. “Getting the best of Linked Data and Property Graphs: rdf2neo and the KnetMiner use case.” In: *SWAT4LS.* 2018 (cit. on pp. 160, 163).

[BSK13] Konstantina Bereta, Panayiotis Smeros, and Manolis Koubarakis. “Representation and Querying of Valid Time of Triples in Linked Geospatial Data”. In: *The Semantic Web: Semantics and Big Data, 10th International Conference, ESWC 2013, Montpellier, France, May 26-30, 2013. Proceedings.* 2013. URL: `https://doi.org/10.1007/978-3-642-38288-8_18` (cit. on pp. 112, 225).

[Car+05a] Jeremy J. Carroll et al. “Named graphs”. In: *J. Web Semant.* (2005). URL: https://doi.org/10.1016/j.websem.2005.09.001 (cit. on pp. 38, 62, 63, 91, 92, 93, 98, 158, 166, 167, 168, 171, 172, 178, 224, 247, 266, 276, 287).

[Car+05b] Jeremy J. Carroll et al. “Named graphs, provenance and trust”. In: *Proceedings of the 14th international conference on World Wide Web, WWW 2005, Chiba, Japan, May 10-14, 2005.* 2005. URL: https://doi.org/10.1145/1060745.1060835 (cit. on pp. 77, 172, 344).

[Car+19] Valentina Anita Carriero et al. “ArCo: The Italian Cultural Heritage Knowledge Graph”. In: *The Semantic Web - ISWC 2019 - 18th International Semantic Web Conference, Auckland, New Zealand, October 26-30, 2019, Proceedings, Part II.* 2019. URL: https://doi.org/10.1007/978-3-030-30796-7_3 (cit. on pp. 121, 132, 222, 350).

[CBR18] Asmaa Chebba, Thouraya Bouabana-Tebibel, and Stuart H. Rubin. “Attributed and n-ary relations in OWL for knowledge modeling”. In: *Comput. Lang. Syst. Struct.* (2018). URL: https://doi.org/10.1016/j.cl.2018.06.001 (cit. on pp. 140, 231, 355).

[CDP16] Michael Cochez, Stefan Decker, and Eric Prud’hommeaux. “Knowledge Representation on the Web Revisited: The Case for Prototypes”. In: *The Semantic Web - ISWC 2016 - 15th International Semantic Web Conference, Kobe, Japan, October 17-21, 2016, Proceedings, Part I.* 2016. URL: https://doi.org/10.1007/978-3-319-46523-4_10 (cit. on p. 231).

[CF07] Olivier Corby and Catherine Faron-Zucker. “RDF/SPARQL Design Pattern for Contextual Metadata”. In: *2007 IEEE / WIC / ACM International Conference on Web Intelligence, WI 2007, 2-5 November 2007, Silicon Valley, CA, USA, Main Conference Proceedings.* 2007. URL: https://doi.org/10.1109/WI.2007.162 (cit. on p. 215).

[CFG17] Olivier Corby, Catherine Faron-Zucker, and Fabien Gandon. “LD-Script: A Linked Data Script Language”. In: *The Semantic Web - ISWC 2017 - 16th International Semantic Web Conference, Vienna, Austria, October 21-25, 2017, Proceedings, Part I.* 2017. URL: https://doi.org/10.1007/978-3-319-68288-4%5C_13 (cit. on p. 297).

[CH22] Nicholas John Car and Timo Homburg. “GeoSPARQL 1.1: Motivations, Details and Applications of the Decadal Update to the Most Important Geospatial LOD Standard”. In: *ISPRS Int. J. Geo Inf.* (2022). URL: https://doi.org/10.3390/ijgi11020117 (cit. on p. 216).

[Che+08] Hacène Cherfi et al. "Semantic Annotation of Texts with RDF Graph Contexts". In: *Supplementary Proceedings of the 16th International Conference on Conceptual Structures, ICCS 2008, Toulouse, France, July 7-11, 2008.* 2008. URL: `http://ceur-ws.org/Vol-354/p40.pdf` (cit. on p. 169).

[Che+12] Lei Chen et al. "Blank Nodes in RDF". In: *J. Softw.* (2012). URL: `https://doi.org/10.4304/jsw.7.9.1993-1999` (cit. on p. 51).

[Che+13] Artem Chebotko et al. "Storing, Indexing and Querying Large Provenance Data Sets as RDF Graphs in Apache HBase". In: *IEEE Ninth World Congress on Services, SERVICES 2013, Santa Clara, CA, USA, June 28 - July 3, 2013.* 2013. URL: `https://doi.org/10.1109/SERVICES.2013.32` (cit. on p. 241).

[Che+17] Valeriy Chernenkiy et al. "Using the metagraph approach for addressing RDF knowledge representation limitations". In: *2017 Internet technologies and applications (ITA).* IEEE. 2017 (cit. on p. 244).

[CP14] Paolo Ciccarese and Silvio Peroni. "The Collections Ontology: Creating and handling collections in OWL 2 DL frameworks". In: *Semantic Web* (2014). URL: `https://doi.org/10.3233/SW-130121` (cit. on p. 45).

[CS04] Jeremy J. Carroll and Patrick Stickler. "RDF Triples in XML". In: *Proceedings of the Extreme Markup Languages® 2004 Conference, 2-6 August 2004, Montréal, Quebec, Canada.* 2004. URL: `http://www.mulberrytech.com/Extreme/Proceedings/html/2004/Stickler01/EML2004Stickler01.html` (cit. on pp. 77, 167).

[Cyg+14] Richard Cyganiak et al. *RDF 1.1 concepts and abstract syntax.* 2014. URL: `https://www.w3.org/TR/rdf11-concepts/` (visited on 09/01/2022) (cit. on pp. 47, 52).

[Cyg10] Richard Cyganiak. "Next steps for RDF: Keep the core and pave the cowpaths". In: (2010). URL: `www.w3.org/2009/12/rdf-ws/papers/ws30` (cit. on pp. 81, 82).

[CYM20] Hirokazu Chiba, Ryota Yamanaka, and Shota Matsumoto. "G2GML: Graph to Graph Mapping Language for Bridging RDF and Property Graphs". In: *Proceedings of the ISWC 2020 Demos and Industry Tracks: From Novel Ideas to Industrial Practice co-located with 19th International Semantic Web Conference (ISWC 2020), Globally online, November 1-6, 2020 (UTC).* 2020. URL: `http://ceur-ws.org/Vol-2721/paper591.pdf` (cit. on pp. 163, 164).

[Das+14] Souripriya Das et al. "A Tale of Two Graphs: Property Graphs as RDF in Oracle". In: *Proceedings of the 17th International Conference on Extending Database Technology, EDBT 2014, Athens, Greece, March 24-28, 2014.* 2014. URL: `https://doi.org/10.5441/002/edbt.2014.82` (cit. on pp. 161, 215, 217, 237).

[Das20a] Souripriya Das. *Modeling Evolving Data in Graphs: The Power of RDF Quads.* 2020. URL: https://blogs.oracle.com/oraclespatial/post/modeling-evolving-data-in-graphs-the-power-of-rdf-quads (visited on 08/11/2022) (cit. on p. 248).

[Das20b] Souripriya Das. *RDFn - Extending RDF to Support Named Triples.* 2020. URL: https://blogs.oracle.com/oraclespatial/post/rdfn-extending-rdf-to-support-named-triples (visited on 08/11/2022) (cit. on p. 171).

[Das21] Souripriya Das. *RDFn: Name Every Triple or Quad – Manually if Member of a Multi-Edge, Automatically Otherwise.* Tutorial slid-edeck. 2021. URL: https://www.linkedin.com/pulse/rdfn-name-every-triple-quad-manually-member-otherwise-das-ph-d-?trk=public_profile_article_view (visited on 08/11/2022) (cit. on pp. 175, 176, 181, 216, 240, 246, 270).

[Dav01] Donald Davidson. *Essays on Actions and Events: Philosophical Essays Volume 1.* 2001 (cit. on p. 103).

[DCG20] Marie Destandau, Olivier Corby, and Alain Giboin. "Path Outlines: Browsing Path-Based Summaries of Knowledge Graphs". In: *CoRR* (2020). URL: https://arxiv.org/abs/2002.09949 (cit. on p. 179).

[DD11] Leigh Dodds and Ian Davis. "Linked data patterns". In: (2011). URL: http://patterns.dataincubator.org/book (visited on 08/11/2022) (cit. on pp. 106, 135).

[Del+08] Renaud Delbru et al. "Context dependent reasoning for semantic documents in sindice". In: *Proc. of 4th SSWS Workshop.* 2008 (cit. on pp. 224, 230).

[Del+21] Thomas Delva et al. "RML-star: A Declarative Mapping Language for RDF-star Generation". In: *Proceedings of the ISWC 2021 Posters, Demos and Industry Tracks: From Novel Ideas to Industrial Practice co-located with 20th International Semantic Web Conference (ISWC 2021), Virtual Conference, October 24-28, 2021.* 2021. URL: http://ceur-ws.org/Vol-2980/paper374.pdf (cit. on p. 157).

[Dim+14] Anastasia Dimou et al. "RML: A Generic Language for Integrated RDF Mappings of Heterogeneous Data". In: *Proceedings of the Workshop on Linked Data on the Web co-located with the 23rd International World Wide Web Conference (WWW 2014), Seoul, Korea, April 8, 2014.* 2014. URL: http://ceur-ws.org/Vol-1184/ldow2014_paper_01.pdf (cit. on p. 210).

[Din+05] Li Ding et al. "Tracking rdf graph provenance using rdf molecules". In: *TR-CS-05-06* (2005) (cit. on pp. 113, 179).

[Din+10] Li Ding et al. "SameAs Networks and Beyond: Analyzing Deployment Status and Implications of owl: sameAs in Linked Data". In: *The Semantic Web - ISWC 2010 - 9th International Semantic Web Conference, ISWC 2010, Shanghai, China, November 7-11, 2010, Revised Selected Papers, Part I.* 2010. URL: `https://doi.org/10.1007/978-3-642-17746-0%5C_10` (cit. on p. 189).

[Div+09] Renata Queiroz Dividino et al. "Querying for provenance, trust, uncertainty and other meta knowledge in RDF". In: *J. Web Semant.* (2009). URL: `https://doi.org/10.1016/j.websem.2009.07.004` (cit. on pp. 116, 171, 174, 215, 220, 226, 240).

[DMM19] Enrico Daga, Albert Meroño-Peñuela, and Enrico Motta. "Modelling and Querying Lists in RDF. A Pragmatic Study". In: *Proceedings of the QuWeDa 2019: 3rd Workshop on Querying and Benchmarking the Web of Data co-located with 18th International Semantic Web Conference (ISWC 2019), Auckland, New Zealand, October 26-30, 2019.* 2019. URL: `http://ceur-ws.org/Vol-2496/paper2.pdf` (cit. on pp. 44, 216).

[Dru+06] Nick Drummond et al. "Putting OWL in Order: Patterns for Sequences in OWL". In: *Proceedings of the OWLED*06 Workshop on OWL: Experiences and Directions, Athens, Georgia, USA, November 10-11, 2006.* 2006. URL: `http://ceur-ws.org/Vol-216/submission_12.pdf` (cit. on p. 45).

[DSC12] Souripriya Das, Seema Sundara, and Richard Cyganiak. "R2RML: RDB to RDF mapping language". In: *http://www.w3.org/TR/r2rml/* (2012) (cit. on p. 210).

[DTG19] Ahmed El Amine Djebri, Andrea G. B. Tettamanzi, and Fabien Gandon. "Publishing Uncertainty on the Semantic Web: Blurring the LOD Bubbles". In: *Graph-Based Representation and Reasoning - 24th International Conference on Conceptual Structures, ICCS 2019, Marburg, Germany, July 1-4, 2019, Proceedings.* 2019. URL: `https://doi.org/10.1007/978-3-030-23182-8%5C_4` (cit. on p. 114).

[DTP11] Renaud Delbru, Giovanni Tummarello, and Axel Polleres. "Context-Dependent OWL Reasoning in Sindice - Experiences and Lessons Learnt". In: *Web Reasoning and Rule Systems - 5th International Conference, RR 2011, Galway, Ireland, August 29-30, 2011. Proceedings.* 2011. URL: `https://doi.org/10.1007/978-3-642-23580-1_5` (cit. on p. 230).

[EB21] Marieke van Erp and Victor de Boer. "A Polyvocal and Contextualised Semantic Web". In: *The Semantic Web - 18th International Conference, ESWC 2021, Virtual Event, June 6-10, 2021, Proceedings.* 2021. URL: `https://doi.org/10.1007/978-3-030-77385-4_30` (cit. on p. 116).

[Ebe+21] Aaron Eberhart et al. “Seed Patterns for Modeling Trees”. In: *Advances in Pattern-Based Ontology Engineering, extended versions of the papers published at the Workshop on Ontology Design and Patterns (WOP)*. 2021. URL: https://doi.org/10.3233/SSW210006 (cit. on pp. 45, 232).

[Eck13] Kai Eckert. “Provenance and Annotations for Linked Data”. In: *Proceedings of the 2013 International Conference on Dublin Core and Metadata Applications, DC 2013, Lisbon, Portugal, September 2-6, 2013*. 2013. URL: http://dcpapers.dublincore.org/pubs/article/view/3669 (cit. on pp. 113, 151, 152, 194).

[Ehr+14] Maud Ehrmann et al. “Representing Multilingual Data as Linked Data: the Case of BabelNet 2.0”. In: *Proceedings of the Ninth International Conference on Language Resources and Evaluation, LREC 2014, Reykjavik, Iceland, May 26-31, 2014*. 2014. URL: http://www.lrec-conf.org/proceedings/lrec2014/summaries/810.html (cit. on p. 114).

[Erx+14] Fredo Erxleben et al. “Introducing Wikidata to the Linked Data Web”. In: *The Semantic Web - ISWC 2014 - 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part I*. 2014. URL: https://doi.org/10.1007/978-3-319-11964-9_4 (cit. on pp. 120, 131, 132, 150).

[ESA08] Mikel Egaña, Robert Stevens, and Erick Antezana. “Transforming the Axiomisation of Ontologies: The Ontology Pre-Processor Language”. In: *Proceedings of the Fourth OWLED Workshop on OWL: Experiences and Directions, Washington, DC, USA, 1-2 April 2008*. 2008. URL: http://ceur-ws.org/Vol-496/owled2008dc%5C_paper%5C_14.pdf (cit. on p. 135).

[FDG18] Marco Fossati, Emilio Dorigatti, and Claudio Giuliano. “N-ary relation extraction for simultaneous T-Box and A-Box knowledge base augmentation”. In: *Semantic Web* (2018). URL: https://doi.org/10.3233/SW-170269 (cit. on p. 131).

[Fer+18] Javier D. Fernández et al. “HDTQ: Managing RDF Datasets in Compressed Space”. In: *The Semantic Web - 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3-7, 2018, Proceedings*. 2018. URL: https://doi.org/10.1007/978-3-319-93417-4_13 (cit. on pp. 172, 238, 240).

[Fer+19] Javier D. Fernández et al. “Evaluating query and storage strategies for RDF archives”. In: *Semantic Web* (2019). URL: https://doi.org/10.3233/SW-180309 (cit. on pp. 115, 245).

[FH17] Johannes Frey and Sebastian Hellmann. "MaSQue: An Approach for Flexible Metadata Storage and Querying in RDF". In: *Proceedings of the Posters and Demos Track of the 13th International Conference on Semantic Systems - SEMANTiCS2017 co-located with the 13th International Conference on Semantic Systems (SEMANTiCS 2017), Amsterdam, The Netherlands, September 11-14, 2017.* 2017. URL: `http://ceur-ws.org/Vol-2044/paper8/` (cit. on pp. 217, 236).

[Flo+09] Giorgos Flouris et al. "Coloring RDF Triples to Capture Provenance". In: *The Semantic Web - ISWC 2009, 8th International Semantic Web Conference, ISWC 2009, Chantilly, VA, USA, October 25-29, 2009. Proceedings.* 2009. URL: `https://doi.org/10.1007/978-3-642-04930-9_13` (cit. on pp. 113, 173).

[Fre+19a] Johannes Frey et al. "DBpedia FlexiFusion The Best of Wikipedia > Wikidata > Your Data". In: *The Semantic Web - ISWC 2019 - 18th International Semantic Web Conference, Auckland, New Zealand, October 26-30, 2019, Proceedings, Part II.* 2019. URL: `https://doi.org/10.1007/978-3-030-30796-7_7` (cit. on p. 196).

[Fre+19b] Johannes Frey et al. "Evaluation of metadata representations in RDF stores". In: *Semantic Web* (2019). URL: `https://doi.org/10.3233/SW-180307` (cit. on pp. 121, 146, 154, 180, 181, 236, 240, 270).

[Fro+16] Marvin Frommhold et al. "Towards Versioning of Arbitrary RDF Data". In: *Proceedings of the 12th International Conference on Semantic Systems, SEMANTICS 2016, Leipzig, Germany, September 12-15, 2016.* 2016. URL: `https://doi.org/10.1145/2993318.2993327` (cit. on p. 115).

[Fu+15] Gang Fu et al. "Exposing Provenance Metadata Using Different RDF Models". In: *CoRR* (2015). URL: `http://arxiv.org/abs/1509.02822` (cit. on pp. 113, 180, 236).

[Gan+16] Aldo Gangemi et al. "Framester: A Wide Coverage Linguistic Linked Data Hub". In: *Knowledge Engineering and Knowledge Management - 20th International Conference, EKAW 2016, Bologna, Italy, November 19-23, 2016, Proceedings.* 2016. URL: `https://doi.org/10.1007/978-3-319-49004-5_16` (cit. on p. 132).

[Gan05] Aldo Gangemi. "Ontology Design Patterns for Semantic Web Content". In: *The Semantic Web - ISWC 2005, 4th International Semantic Web Conference, ISWC 2005, Galway, Ireland, November 6-10, 2005, Proceedings.* 2005. URL: `https://doi.org/10.1007/11574620_21` (cit. on p. 134).

[Gan11] Aldo Gangemi. "Super-duper schema: an OWL2+RIF DnS pattern". In: *Proceedings of DeepKR Challenge Workshop at KCAP11. Edited by Chaudry.* 2011 (cit. on pp. 144, 146, 180, 246).

[Gar02] Lars Marius Garshol. “Q: A model for topic maps Unifying RDF and Topic Maps”. In: *EXTREME MARKUP LANGUAGES 2005*. Citeseer. 2002 (cit. on p. 133).

[Gar03] Lars Marius Garshol. “Living with topic maps and RDF”. In: *Online only* (2003) (cit. on p. 133).

[Gay+17] José Emilio Labra Gayo et al. *Validating RDF Data*. 2017. URL: `https://doi.org/10.2200/S00786ED1V01Y201707WBE016` (cit. on pp. 13, 136).

[Gay22] Jose Emilio Labra Gayo. “WShEx: A language to describe and validate Wikibase entities”. In: *arXiv preprint arXiv:2208.02697* (2022) (cit. on p. 137).

[GBM15] Ramanathan V. Guha, Dan Brickley, and Steve MacBeth. “Schema.org: Evolution of Structured Data on the Web: Big data makes common schemas even more necessary.” In: *Queue* (2015) (cit. on p. 211).

[GC10] Fabien Gandon and Olivier Corby. *Name That Graph or the need to provide a model and syntax extension to specify the provenance of RDF graphs*. 2010. URL: `https://www.w3.org/2009/12/rdf-ws/papers/ws06` (visited on 08/11/2022) (cit. on pp. 168, 169).

[GGV10] Paul Groth, Andrew Gibson, and Jan Velterop. “The anatomy of a nanopublication”. In: *Inf. Serv. Use* (2010). URL: `https://doi.org/10.3233/ISU-2010-0613` (cit. on pp. 113, 169, 171).

[GHV05] Claudio Gutiérrez, Carlos A. Hurtado, and Alejandro A. Vaisman. “Temporal RDF”. In: *The Semantic Web: Research and Applications, Second European Semantic Web Conference, ESWC 2005, Heraklion, Crete, Greece, May 29 - June 1, 2005, Proceedings*. 2005. URL: `https://doi.org/10.1007/11431053_7` (cit. on p. 112).

[GHV07] Claudio Gutiérrez, Carlos A. Hurtado, and Alejandro A. Vaisman. “Introducing Time into RDF”. In: *IEEE Trans. Knowl. Data Eng.* (2007). URL: `https://doi.org/10.1109/TKDE.2007.34` (cit. on pp. 112, 151).

[Gim+18] José M. Giménez-García et al. “NELL2RDF: Reading the Web, Tracking the Provenance, and Publishing it as Linked Data”. In: *Joint Proceedings of the International Workshops on Contextualized Knowledge Graphs, and Semantic Statistics co-located with 17th International Semantic Web Conference (ISWC 2018)*. 2018. URL: `http://ceur-ws.org/Vol-2317/article-02.pdf` (cit. on p. 237).

[Giu+21] Marco Giunti et al. “Representing n-ary relations in the Semantic Web”. In: *Log. J. IGPL* (2021). URL: `https://doi.org/10.1093/jigpal/jzz047` (cit. on p. 206).

[Giu93] Fausto Giunchiglia. “Contextual reasoning”. In: *Epistemologia, special issue on I Linguaggi e le Macchine* (1993) (cit. on pp. 219, 222).

[GJM09] Hugh Glaser, Afraz Jaffri, and Ian Millard. “Managing Co-reference on the Semantic Web”. In: *Proceedings of the WWW2009 Workshop on Linked Data on the Web, LDOW 2009, Madrid, Spain, April 20, 2009*. 2009. URL: `http://ceur-ws.org/Vol-538/ldow2009%5C_paper11.pdf` (cit. on p. 185).

[GL05] Manolis Gergatsoulis and Pantelis Lilis. “Multidimensional rdf”. In: *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*. Springer. 2005 (cit. on p. 224).

[GM03a] Aldo Gangemi and Peter Mika. “Understanding the Semantic Web through Descriptions and Situations”. In: *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE - OTM Confederated International Conferences, CoopIS, DOA, and ODBASE 2003, Catania, Sicily, Italy, November 3-7, 2003*. 2003. URL: `https://doi.org/10.1007/978-3-540-39964-3_44` (cit. on p. 277).

[GM03b] Ramanathan V. Guha and John McCarthy. “Varieties of Contexts”. In: *Modeling and Using Context, 4th International and Interdisciplinary Conference, CONTEXT 2003, Stanford, CA, USA, June 23-25, 2003, Proceedings*. 2003. URL: `https://doi.org/10.1007/3-540-44958-2_14` (cit. on pp. 208, 219).

[GM16] Ramanathan V. Guha and Andrew Moore. *The OKN White Paper: Open Knowledge Network: Creating the Semantic Information Infrastructure for the Future*. 2016. URL: `https://ichs.ucsf.edu/wp-content/uploads/2017/08/OKN-White-Paper..docx` (cit. on pp. 122, 267, 352).

[GMF04] Ramanathan V. Guha, Rob McCool, and Richard Fikes. “Contexts for the Semantic Web”. In: *The Semantic Web - ISWC 2004: Third International Semantic Web Conference, Hiroshima, Japan, November 7-11, 2004. Proceedings*. 2004. URL: `https://doi.org/10.1007/978-3-540-30475-3_4` (cit. on pp. 37, 117, 118, 208, 220, 223, 224, 227, 266, 288, 347).

[GP13] Aldo Gangemi and Valentina Presutti. “A Multi-dimensional Comparison of Ontology Design Patterns for Representing *n*-ary Relations”. In: *SOFSEM 2013: Theory and Practice of Computer Science, 39th International Conference on Current Trends in Theory and Practice of Computer Science, Špindlerův Mlýn, Czech Republic, January 26-31, 2013. Proceedings*. 2013. URL: `https://doi.org/10.1007/978-3-642-35843-2_8` (cit. on pp. 102, 103, 130, 134, 138, 144, 180, 207, 216, 225, 236, 270, 277, 355).

[GP16] Aldo Gangemi and Valentina Presutti. “Multi-layered n-ary Patterns”. In: *Ontology Engineering with Ontology Design Patterns - Foundations and Applications*. 2016. URL: `https://doi.org/10.3233/978-1-61499-676-7-105` (cit. on p. 134).

[GRV10] Birte Glimm, Sebastian Rudolph, and Johanna Völker. “Integrated Metamodeling and Diagnosis in OWL 2”. In: *The Semantic Web - ISWC 2010 - 9th International Semantic Web Conference, ISWC 2010, Shanghai, China, November 7-11, 2010, Revised Selected Papers, Part I*. 2010. URL: `https://doi.org/10.1007/978-3-642-17746-0_17` (cit. on p. 230).

[GS98] Chiara Ghidini and Luciano Serafini. “Distributed first order logics”. In: (1998) (cit. on p. 222).

[GT17] Pawel Garbacz and Robert Trypuz. “Representation of Tensed Relations in OWL - A Survey of Philosophically-Motivated Patterns”. In: *Metadata and Semantic Research - 11th International Conference, MTSR 2017 Tallinn, Estonia, November 28 - December 1, 2017, Proceedings*. 2017. URL: `https://doi.org/10.1007/978-3-319-70863-8_6` (cit. on p. 225).

[Gua09] Nicola Guarino. “The Ontological Level: Revisiting 30 Years of Knowledge Representation”. In: *Conceptual Modeling: Foundations and Applications - Essays in Honor of John Mylopoulos*. 2009. URL: `https://doi.org/10.1007/978-3-642-02463-4_4` (cit. on p. 352).

[Guh92] Ramanathan V. Guha. “Contexts: a formalization and some applications”. PhD thesis. Stanford University, 1992 (cit. on p. 227).

[GZ18] José M. Giménez-García and Antoine Zimmermann. “NdProperties: Encoding Contexts in RDF predicates with Inference Preservation”. In: *Joint Proceedings of the International Workshops on Contextualized Knowledge Graphs, and Semantic Statistics co-located with 17th International Semantic Web Conference (ISWC 2018)*. 2018. URL: `http://ceur-ws.org/Vol-2317/article-04.pdf` (cit. on pp. 149, 181).

[GZM17] José M. Giménez-García, Antoine Zimmermann, and Pierre Maret. “NdFluents: An Ontology for Annotated Statements with Inference Preservation”. In: *The Semantic Web - 14th International Conference, ESWC 2017, Portorož, Slovenia, May 28 - June 1, 2017, Proceedings, Part I*. 2017. URL: `https://doi.org/10.1007/978-3-319-58068-5_39` (cit. on pp. 149, 181, 225, 226).

[Hal+10] Harry Halpin et al. “When owl: sameAs Isn’t the Same: An Analysis of Identity in Linked Data”. In: *The Semantic Web - ISWC 2010 - 9th International Semantic Web Conference, ISWC 2010, Shanghai, China, November 7-11, 2010, Revised Selected Papers, Part I*. 2010. URL: `https://doi.org/10.1007/978-3-642-17746-0_20` (cit. on pp. 186, 188, 201).

[Hal12] Harry Halpin. *Social semantics: the search for meaning on the web*. 2012 (cit. on pp. 36, 65, 67, 68, 78, 227).

[Har+21] Olaf Hartig et al. *RDF-star and SPARQL-star*. 2021. URL: `https://w3c.github.io/rdf-star/cg-spec` (visited on 08/23/2022) (cit. on pp. 77, 156, 157).

[Har14] Olaf Hartig. "Reconciliation of RDF*and Property Graphs". In: *CoRR* (2014). URL: `http://arxiv.org/abs/1409.3288` (cit. on pp. 161, 162).

[Har17] Olaf Hartig. "Foundations of RDF⋆ and SPARQL⋆ (An Alternative Approach to Statement-Level Metadata in RDF)". In: *Proceedings of the 11th Alberto Mendelzon International Workshop on Foundations of Data Management and the Web, Montevideo, Uruguay, June 7-9, 2017*. 2017. URL: `http://ceur-ws.org/Vol-1912/paper12.pdf` (cit. on pp. 155, 156).

[Har19] Olaf Hartig. "Foundations to Query Labeled Property Graphs using SPARQL". In: *Joint Proceedings of the 1st International Workshop On Semantics For Transport and the 1st International Workshop on Approaches for Making Data Interoperable co-located with 15th Semantics Conference (SEMANTiCS 2019), Karlsruhe, Germany, September 9, 2019*. 2019. URL: `http://ceur-ws.org/Vol-2447/paper3.pdf` (cit. on pp. 162, 236).

[Haw+12] Sandro Hawke et al. *RDF Graph Identification*. 2012. URL: `https://www.w3.org/2012/08/RDFNG.html` (visited on 09/01/2022) (cit. on p. 98).

[Haw12] Sandro Hawke. *RDF Spaces and Datasets*. 2012. URL: `https://dvcs.w3.org/hg/rdf/raw-file/tip/rdf-spaces` (visited on 09/01/2022) (cit. on p. 98).

[Hay02] Patrick J. Hayes. *Unasserted triples, Contexts and things that go bump in the night*. 2002. URL: `https://lists.w3.org/Archives/Public/w3c-rdfcore-wg/2002Mar/0253.html` (visited on 08/11/2022) (cit. on p. 195).

[Hay03] Patrick J. Hayes. *Re: Dark triples, motivating examples*. 2003. URL: `https://lists.w3.org/Archives/Public/w3c-rdfcore-wg/2002Apr/0186.html` (visited on 08/11/2022) (cit. on p. 196).

[Hay04a] Patrick J. Hayes. *Formal unifying standards for the representation of spatiotemporal knowledge*. Tech. rep. Technical report, IHMC, 2004, 2004. URL: `http://www.ihmc.us/users/phayes/Trickledown2004.pdf` (visited on 08/18/2022) (cit. on p. 180).

[Hay04b] Patrick J. Hayes. *RDF Semantics*. 2004. URL: `https://www.w3.org/TR/rdf-mt/` (visited on 09/01/2022) (cit. on pp. 36, 65, 71, 73, 78, 85, 88, 188, 227, 257).

[Hay07] Patrick J. Hayes. "Context Mereology". In: *Logical Formalizations of Commonsense Reasoning, Papers from the 2007 AAAI Spring Symposium, Technical Report SS-07-05, Stanford, California, USA, March 26-28, 2007*. 2007. URL: `http://www.aaai.org/Library/Symposia/Spring/2007/ss07-05-011.php` (cit. on p. 349).

[Hay09] Patrick J. Hayes. *How Quaint the Ways of Paradox*. 2009. URL: `https://www.slideshare.net/PatHayes/how-quaint-the-ways-of-paradox` (visited on 08/14/2022) (cit. on p. 80).

[Hay12a] Patrick J. Hayes. *Another Spin*. 2012. URL: `https://www.w3.org/2011/rdf-wg/wiki/AnotherSpin` (visited on 09/01/2022) (cit. on pp. 36, 78, 97, 98, 168).

[Hay12b] Patrick J. Hayes. *RDF with Contexts*. 2012. URL: `https://www.slideshare.net/PatHayes/rdf-with-contexts` (visited on 09/01/2022) (cit. on pp. 94, 95).

[Hay12c] Patrick J. Hayes. *RDFC: RDF with Contexts*. 2012. URL: `https://www.w3.org/2011/rdf-wg/wiki/RDFwithContexts` (visited on 08/11/2022) (cit. on pp. 97, 98, 168).

[Hay77] Patrick J. Hayes. "In defence of logic". In: *Proc. IJCAI-77*. 1977 (cit. on pp. 67, 68).

[Hay97] Pat Hayes. "Contexts in context". In: *Context in knowledge representation and natural language, AAAI Fall Symposium*. 1997 (cit. on p. 219).

[HB10] Jim Hendler and Tim Berners-Lee. "From the Semantic Web to social machines: A research challenge for AI on the World Wide Web". In: *Artif. Intell.* (2010). URL: `https://doi.org/10.1016/j.artint.2009.11.010` (cit. on p. 117).

[HB11] Tom Heath and Christian Bizer. *Linked Data: Evolving the Web into a Global Data Space*. 2011. URL: `https://doi.org/10.2200/S00334ED1V01Y201102WBE001` (cit. on pp. 57, 119).

[HC14] Harry Halpin and James Cheney. "Dynamic provenance for SPARQL updates". In: *International Semantic Web Conference*. Springer. 2014 (cit. on p. 115).

[Her+16] Daniel Hernández et al. "Querying Wikidata: Comparing SPARQL, Relational and Graph Databases". In: *The Semantic Web - ISWC 2016 - 15th International Semantic Web Conference, Kobe, Japan, October 17-21, 2016, Proceedings, Part II*. 2016. URL: `https://doi.org/10.1007/978-3-319-46547-0_10` (cit. on pp. 121, 146, 236).

[Her10] Ivan Herman. *RDF Graph Literals and Named Graphs*. 2010. URL: `https://www.w3.org/2009/07/NamedGraph.html` (visited on 09/01/2022) (cit. on pp. 98, 99, 141).

[HG04] Jonathan Hayes and Claudio Gutiérrez. “Bipartite Graphs as Intermediate Model for RDF”. In: *The Semantic Web - ISWC 2004: Third International Semantic Web Conference,Hiroshima, Japan, November 7-11, 2004. Proceedings*. 2004. URL: `https://doi.org/10.1007/978-3-540-30475-3_5` (cit. on p. 243).

[HH08] Patrick J. Hayes and Harry Halpin. “In defense of ambiguity”. In: *International Journal on Semantic Web and Information Systems (IJSWIS)* (2008) (cit. on pp. 30, 32, 34, 35, 200).

[HH10] Harry Halpin and Patrick J. Hayes. “When owl: sameAs isn’t the Same: An Analysis of Identity Links on the Semantic Web”. In: *Proceedings of the WWW2010 Workshop on Linked Data on the Web, LDOW 2010, Raleigh, USA, April 27, 2010*. 2010. URL: `http://ceur-ws.org/Vol-628/ldow2010%5C_paper09.pdf` (cit. on pp. 28, 63, 201).

[HHK15] Daniel Hernández, Aidan Hogan, and Markus Krötzsch. “Reifying RDF: What Works Well With Wikidata?” In: *Proceedings of the 11th International Workshop on Scalable Semantic Web Knowledge Base Systems co-located with 14th International Semantic Web Conference (ISWC 2015), Bethlehem, PA, USA, October 11, 2015*. 2015. URL: `http://ceur-ws.org/Vol-1457/SSWS2015_paper3.pdf` (cit. on pp. 121, 150, 151, 153, 155, 156, 171, 176, 180, 217, 236, 270, 354).

[HHT15] Harry Halpin, Patrick J. Hayes, and Henry S. Thompson. “When owl: sameAs isn’t the Same Redux: Towards a Theory of Identity, Context, and Inference on the Semantic Web”. In: *Modeling and Using Context - 9th International and Interdisciplinary Conference, CONTEXT 2015, Lanarca, Cyprus, November 2-6, 2015. Proceedings*. 2015. URL: `https://doi.org/10.1007/978-3-319-25591-0_4` (cit. on pp. 36, 187, 203).

[Hit+17] Pascal Hitzler et al. “Towards a Simple but Useful Ontology Design Pattern Representation Language”. In: *Proceedings of the 8th Workshop on Ontology Design and Patterns (WOP 2017) co-located with the 16th International Semantic Web Conference (ISWC 2017), Vienna, Austria, October 21, 2017*. 2017. URL: `http://ceur-ws.org/Vol-2043/paper-09.pdf` (cit. on p. 134).

[HJ03] Steven D Hales and Timothy A Johnson. “Endurantism, perdurantism and special relativity”. In: *The Philosophical Quarterly* (2003) (cit. on p. 225).

[Hoe09] Rinke Hoekstra. *Ontology Representation - Design Patterns and Ontologies that Make Sense*. 2009. ISBN: 978-1-60750-013-1. URL: `https://doi.org/10.3233/978-1-60750-013-1-i` (cit. on pp. 108, 134).

[Hoe10] Rinke Hoekstra. "Representing Social Reality in OWL 2". In: *Proceedings of the 7th International Workshop on OWL: Experiences and Directions (OWLED 2010), San Francisco, California, USA, June 21-22, 2010*. 2010. URL: http://ceur-ws.org/Vol-614/owled2010_submission_29.pdf (cit. on p. 230).

[Hof+13] Johannes Hoffart et al. "YAGO2: A spatially and temporally enhanced knowledge base from Wikipedia". In: *Artif. Intell.* (2013). URL: https://doi.org/10.1016/j.artint.2012.06.001 (cit. on pp. 75, 112, 120, 172, 177, 180, 215, 229, 240, 241).

[Hog+12] Aidan Hogan et al. "Scalable and distributed methods for entity matching, consolidation and disambiguation over linked data corpora". In: *J. Web Semant.* (2012). URL: https://doi.org/10.1016/j.websem.2011.11.002 (cit. on p. 188).

[Hog+14] Aidan Hogan et al. "Everything you always wanted to know about blank nodes". In: *J. Web Semant.* (2014). URL: https://doi.org/10.1016/j.websem.2014.06.004 (cit. on pp. 49, 50, 54, 55, 58, 59, 210).

[Hog+20] Aidan Hogan et al. "Knowledge Graphs". In: *CoRR* (2020). URL: https://arxiv.org/abs/2003.02320 (cit. on pp. 32, 181).

[Hog+21] Aidan Hogan et al. *Knowledge Graphs*. English. 2021. ISBN: 9781636392363. URL: https://kgbook.org/ (cit. on p. 122).

[Hog14] Aidan Hogan. *Reasoning Techniques for the Web of Data*. 2014. ISBN: 978-1-61499-382-7. URL: https://doi.org/10.3233/978-1-61499-383-4-i (cit. on pp. 116, 233).

[Hog15] Aidan Hogan. "Skolemising Blank Nodes while Preserving Isomorphism". In: *Proceedings of the 24th International Conference on World Wide Web, WWW 2015, Florence, Italy, May 18-22, 2015*. 2015. URL: https://doi.org/10.1145/2736277.2741653 (cit. on p. 61).

[Hog17] Aidan Hogan. "Canonical Forms for Isomorphic and Equivalent RDF Graphs: Algorithms for Leaning and Labelling Blank Nodes". In: *ACM Trans. Web* (2017). URL: https://doi.org/10.1145/3068333 (cit. on p. 61).

[Hog18] Aidan Hogan. "Context in graphs". In: *Proceedings of the 1st International Workshop on Conceptualized Knowledge Graphs. RWTH Aachen University, Aachen*. 2018 (cit. on p. 153).

[Hor05] Herman J. ter Horst. "Completeness, decidability and complexity of entailment for RDF Schema and a semantic extension involving the OWL vocabulary". In: *J. Web Semant.* (2005). URL: https://doi.org/10.1016/j.websem.2005.06.001 (cit. on p. 225).

[HP03] Ian Horrocks and Peter F. Patel-Schneider. "Three theses of representation in the semantic web". In: *Proceedings of the Twelfth International World Wide Web Conference, WWW 2003, Budapest, Hungary, May 20-24, 2003*. 2003. URL: `https://doi.org/10.1145/775152.775159` (cit. on pp. 68, 70).

[HP09] Harry Halpin and Valentina Presutti. "An Ontology of Resources: Solving the Identity Crisis". In: *The Semantic Web: Research and Applications, 6th European Semantic Web Conference, ESWC 2009, Heraklion, Crete, Greece, May 31-June 4, 2009, Proceedings*. 2009. URL: `https://doi.org/10.1007/978-3-642-02121-3_39` (cit. on pp. 29, 32, 35, 36, 200).

[HP14] Patrick J. Hayes and Peter F. Patel-Schneider. *RDF 1.1 Semantics*. 2014. URL: `https://www.w3.org/TR/rdf11-mt/` (visited on 09/01/2022) (cit. on pp. 37, 52, 53, 65, 91).

[HT14] Olaf Hartig and Bryan Thompson. "Foundations of an Alternative Approach to Reification in RDF". In: *CoRR* (2014). URL: `http://arxiv.org/abs/1406.3399` (cit. on pp. 154, 155, 156).

[HTS10] Martin Homola, Andrei Tamilin, and Luciano Serafini. "Modeling Contextualized Knowledge". In: *Proceedings of the Second Workshop on Context, Information and Ontologies, CIAOEKAW 2010, Lisbon, Portugal, October 11, 2010*. 2010. URL: `http://ceur-ws.org/Vol-626/regular5.pdf` (cit. on pp. 118, 228, 247, 355).

[Idr+17] Al Koudous Idrissou et al. "Is my: sameAs the same as your: sameAs?: Lenticular Lenses for Context-Specific Identity". In: *Proceedings of the Knowledge Capture Conference, K-CAP 2017, Austin, TX, USA, December 4-6, 2017*. 2017. URL: `https://doi.org/10.1145/3148011.3148029` (cit. on pp. 114, 148, 172, 181, 187).

[Ili+20] Filip Ilievski et al. "KGTK: A Toolkit for Large Knowledge Graph Manipulation and Analysis". In: *The Semantic Web - ISWC 2020 - 19th International Semantic Web Conference, Athens, Greece, November 2-6, 2020, Proceedings, Part II*. 2020. URL: `https://doi.org/10.1007/978-3-030-62466-8%5C_18` (cit. on pp. 121, 239).

[Ism+18] Ali Ismayilov et al. "Wikidata through the eyes of DBpedia". In: *Semantic Web* (2018). URL: `https://doi.org/10.3233/SW-170277` (cit. on pp. 120, 150, 152, 215).

[ISZ21] Filip Ilievski, Pedro A. Szekely, and Bin Zhang. "CSKG: The CommonSense Knowledge Graph". In: *The Semantic Web - 18th International Conference, ESWC 2021, Virtual Event, June 6-10, 2021, Proceedings*. 2021. URL: `https://doi.org/10.1007/978-3-030-77385-4%5C_41` (cit. on p. 121).

[JM10] Li Ding Jie Bao and Deborah L. McGuinness. *Contextualized RDF Importing*. 2010. URL: www.w3.org/2009/12/rdf-ws/papers/ws33 (visited on 09/09/2022) (cit. on pp. 81, 220, 232).

[JS11] Mathew Joseph and Luciano Serafini. "Simple Reasoning for Contextualized RDF Knowledge". In: *Modular Ontologies - Proceedings of the Fifth International Workshop, WoMO 2011, Ljubljana, Slovenia, August 2011*. 2011. URL: https://doi.org/10.3233/978-1-60750-799-4-79 (cit. on pp. 224, 228).

[KC04] Graham Klyne and Jeremy J. Carroll. *RDF Concepts and Abstract Syntax*. 2004. URL: https://www.w3.org/TR/rdf-concepts/ (visited on 09/01/2022) (cit. on p. 38).

[KD14] Tobias Kuhn and Michel Dumontier. "Trusty URIs: Verifiable, Immutable, and Permanent Digital Artifacts for Linked Data". In: *The Semantic Web: Trends and Challenges - 11th International Conference, ESWC 2014, Anissaras, Crete, Greece, May 25-29, 2014. Proceedings*. 2014. URL: https://doi.org/10.1007/978-3-319-07443-6_27 (cit. on pp. 113, 114, 170).

[KD15] Hans-Ulrich Krieger and Thierry Declerck. "An OWL Ontology for Biographical Knowledge. Representing Time-Dependent Factual Knowledge". In: *Proceedings of the First Conference on Biographical Data in a Digital World 2015, Amsterdam, The Netherlands, April 9, 2015*. 2015. URL: http://ceur-ws.org/Vol-1399/paper16.pdf (cit. on p. 112).

[KFH22] Shahrzad Khayatbashi, Sebastián Ferrada, and Olaf Hartig. "Converting property graphs to RDF: a preliminary study of the practical impact of different mappings". In: *GRADES-NDA '22: Proceedings of the 5th ACM SIGMOD Joint International Workshop on Graph Data Management Experiences & Systems (GRADES) and Network Data Analytics (NDA), Philadelphia, Pennsylvania, USA, 12 June 2022*. 2022. URL: https://doi.org/10.1145/3534540.3534695 (cit. on pp. 159, 236).

[KG11] Szymon Klarman and Víctor Gutiérrez-Basulto. "Two-Dimensional Description Logics for Context-Based Semantic Interoperability". In: *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2011, San Francisco, California, USA, August 7-11, 2011*. 2011. URL: http://www.aaai.org/ocs/index.php/AAAI/AAAI11/paper/view/3688 (cit. on p. 224).

[Khe+19] Abdallah Khelil et al. "Should We Be Afraid of Querying Billions of Triples in a Graph-Based Centralized System?" In: *Model and Data Engineering - 9th International Conference, MEDI 2019, Toulouse, France, October 28-31, 2019, Proceedings*. 2019. URL: https://doi.org/10.1007/978-3-030-32065-2%5C_18 (cit. on p. 39).

[Kim+15] Jin-Dong Kim et al. "Semantic representation of annotation involving texts and linked data resources". In: *Semantic Web journal* (2015) (cit. on p. 209).

[KK10] Manolis Koubarakis and Kostis Kyzirakos. "Modeling and Querying Metadata in the Semantic Sensor Web: The Model stRDF and the Query Language stSPARQL". In: *The Semantic Web: Research and Applications, 7th Extended Semantic Web Conference, ESWC 2010, Heraklion, Crete, Greece, May 30 - June 3, 2010, Proceedings, Part I.* 2010. URL: https://doi.org/10.1007/978-3-642-13486-9_29 (cit. on pp. 111, 173, 216).

[KKK12] Kostis Kyzirakos, Manos Karpathiotakis, and Manolis Koubarakis. "Strabon: A Semantic Geospatial DBMS". In: *The Semantic Web - ISWC 2012 - 11th International Semantic Web Conference, Boston, MA, USA, November 11-15, 2012, Proceedings, Part I.* 2012. URL: https://doi.org/10.1007/978-3-642-35176-1_19 (cit. on p. 216).

[Kla13] Szymon Klarman. "Reasoning with contexts in description logics". PhD thesis. Citeseer, 2013 (cit. on p. 118).

[Kly01] Graham Klyne. "Information Modelling using RDF-Constructs for Modular Description of Complex Systems". In: (2001) (cit. on p. 224).

[Kly02] Graham Klyne. *Re: Unasserted triples, Contexts and things that go bump in the night.* 2002. URL: https://lists.w3.org/Archives/Public/w3c-rdfcore-wg/2002Mar/0267.html (visited on 08/11/2022) (cit. on p. 196).

[Knu21] Holger Knublauch. *DASH Reification Support for SHACL.* 2021. URL: https://datashapes.org/reification.html (visited on 10/12/2022) (cit. on pp. 152, 240).

[Knu97] Donald Ervin Knuth. *The art of computer programming.* Pearson Education, 1997 (cit. on p. 271).

[KPS19] Christian Kindermann, Bijan Parsia, and Uli Sattler. "Comparing Approaches for Capturing Repetitive Structures in Ontology Design Patterns". In: *Proceedings of the 10th Workshop on Ontology Design and Patterns (WOP 2019) co-located with 18th International Semantic Web Conference (ISWC 2019), Auckland, New Zealand, October 27, 2019.* 2019. URL: http://ceur-ws.org/Vol-2459/paper2.pdf (cit. on p. 135).

[KR02] Ralph Kimball and Margy Ross. *The data warehouse toolkit: the complete guide to dimensional modeling, 2nd Edition.* 2002. ISBN: 9780471200246. URL: https://www.worldcat.org/oclc/49284159 (cit. on p. 39).

[Kri12] Hans-Ulrich Krieger. “A Temporal Extension of the Hayes/ter Horst Entailment Rules and an Alternative to W3C’s N-ary Relations”. In: *Formal Ontology in Information Systems - Proceedings of the Seventh International Conference, FOIS 2012, Gray, Austria, July 24-27, 2012*. 2012. URL: `https://doi.org/10.3233/978-1-61499-084-0-323` (cit. on pp. 112, 176).

[Kri14] Hans-Ulrich Krieger. “A detailed comparison of seven approaches for the annotation of time-dependent factual knowledge in rdf and owl”. In: *Proceedings 10th joint ISO-ACL SIGSEM workshop on interoperable semantic annotation*. 2014 (cit. on pp. 112, 225, 236).

[Krö+17] Markus Krötzsch et al. “Reasoning with Attributed Description Logics”. In: *Proceedings of the 30th International Workshop on Description Logics, Montpellier, France, July 18-21, 2017*. 2017. URL: `http://ceur-ws.org/Vol-1879/paper47.pdf` (cit. on p. 233).

[Krö+18] Markus Krötzsch et al. “Attributed Description Logics: Reasoning on Knowledge Graphs”. In: *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*. 2018. URL: `https://doi.org/10.24963/ijcai.2018/743` (cit. on p. 233).

[Krö17] Markus Krötzsch. “Ontologies for Knowledge Graphs?” In: *Proceedings of the 30th International Workshop on Description Logics, Montpellier, France, July 18-21, 2017*. 2017. URL: `http://ceur-ws.org/Vol-1879/invited2.pdf` (cit. on pp. 119, 121, 217, 232, 233, 246, 352, 353).

[KS92] Michael Kifer and V. S. Subrahmanian. “Theory of Generalized Annotated Logic Programming and its Applications”. In: *J. Log. Program.* (1992). URL: `https://doi.org/10.1016/0743-1066(92)90007-P` (cit. on p. 220).

[KT05] Oleksiy Khriyenko and Vagan Terziyan. “Context description framework for the semantic web”. In: *Proceedings Context 2005 Context representation and reasoning workshop, Paris (FR)*. 2005 (cit. on p. 224).

[KT16] Markus Krötzsch and Veronika Thost. “Ontologies for Knowledge Graphs: Breaking the Rules”. In: *The Semantic Web - ISWC 2016 - 15th International Semantic Web Conference, Kobe, Japan, October 17-21, 2016, Proceedings, Part I*. 2016. URL: `https://doi.org/10.1007/978-3-319-46523-4_23` (cit. on p. 233).

[Kuh+16] Tobias Kuhn et al. “Decentralized provenance-aware publishing with nanopublications”. In: *PeerJ Comput. Sci.* (2016). URL: `https://doi.org/10.7717/peerj-cs.78` (cit. on pp. 113, 170, 216).

[Kuh+17] Tobias Kuhn et al. "Reliable Granular References to Changing Linked Data". In: *The Semantic Web - ISWC 2017 - 16th International Semantic Web Conference, Vienna, Austria, October 21-25, 2017, Proceedings, Part I*. 2017. URL: `https://doi.org/10.1007/978-3-319-68288-4_26` (cit. on p. 113).

[Kuh+18] Tobias Kuhn et al. "Nanopublications: A Growing Resource of Provenance-Centric Scientific Linked Data". In: *CoRR* (2018). URL: `http://arxiv.org/abs/1809.06532` (cit. on pp. 170, 240).

[Kut+04] Oliver Kutz et al. "E-connections of abstract description systems". In: *Artif. Intell.* (2004). URL: `https://doi.org/10.1016/j.artint.2004.02.002` (cit. on p. 222).

[KW15] Hans-Ulrich Krieger and Christian Willms. "Extending owl ontologies by cartesian types to represent n-ary relations in natural language". In: *Proceedings of the 1st Workshop on Language and Ontologies*. 2015 (cit. on p. 230).

[Las+] Ora Lassila et al. "The OneGraph Vision: Challenges of Breaking the Graph Model Lock-In". In: *Semantic Web* (). URL: `https://www.semantic-web-journal.net/content/onegraph-vision-challenges-breaking-graph-model-lock-0` (visited on 10/08/2022) (cit. on pp. 154, 181, 215, 237, 239, 240, 248, 255, 256, 257, 258, 356).

[Las+21] Ora Lassila et al. "Graph? Yes! Which one? Help!" In: *CoRR* (2021). URL: `https://arxiv.org/abs/2110.13348` (cit. on pp. 45, 75, 160, 165, 246).

[LDV20] Sven Lieber, Anastasia Dimou, and Ruben Verborgh. "Statistics about Data Shape Use in RDF Data". In: *Proceedings of the ISWC 2020 Demos and Industry Tracks: From Novel Ideas to Industrial Practice co-located with 19th International Semantic Web Conference (ISWC 2020), Globally online, November 1-6, 2020 (UTC)*. 2020. URL: `http://ceur-ws.org/Vol-2721/paper584.pdf` (cit. on p. 137).

[Leh+15] Jens Lehmann et al. "DBpedia - A large-scale, multilingual knowledge base extracted from Wikipedia". In: *Semantic Web* (2015). URL: `https://doi.org/10.3233/SW-140134` (cit. on pp. 33, 120).

[Len98] Doug Lenat. *The dimensions of context-space*. 1998 (cit. on pp. 207, 227).

[Lin22] Niklas Lindström. *The Qualified Identity Ontology (QuID)*. 2022. URL: `https://niklasl.github.io/quid/` (visited on 11/03/2022) (cit. on pp. 42, 204).

[Liv+13] Kevin M. Livingston et al. “Representing annotation compositionality and provenance for the Semantic Web”. In: *J. Biomed. Semant.* (2013). URL: `https://doi.org/10.1186/2041-1480-4-38` (cit. on pp. 142, 151).

[LS99] Ora Lassila and Ralph R. Swick. *Resource Description Framework (RDF) Model and Syntax Specification.* 1999. URL: `http://www.w3.org/TR/PR-rdf-syntax` (visited on 09/01/2022) (cit. on pp. 68, 81, 82, 86, 93).

[LT07] Hsien-Chou Liao and Chien-Chih Tu. “A RDF and OWL-based temporal context reasoning model for smart home”. In: *Information Technology Journal* (2007) (cit. on p. 112).

[LW10] James Leigh and David Wood. “An ordered RDF list. W3C workshop–RDF next steps; June 26–27, 2010”. In: *National Center for Biomedical Ontology (NCBO), Stanford, Palo Alto, CA, USA: W3C* (2010). URL: `https://www.w3.org/2009/12/rdf-ws/papers/ws14` (cit. on pp. 45, 267).

[LZ16] Maxime Lefrançois and Antoine Zimmermann. “Supporting Arbitrary Custom Datatypes in RDF and SPARQL”. In: *The Semantic Web. Latest Advances and New Domains - 13th International Conference, ESWC 2016, Heraklion, Crete, Greece, May 29 - June 2, 2016, Proceedings.* 2016. URL: `https://doi.org/10.1007/978-3-319-34129-3_23` (cit. on pp. 140, 141).

[LZ18] Maxime Lefrançois and Antoine Zimmermann. “The Unified Code for Units of Measure in RDF: cdt: ucum and other UCUM Datatypes”. In: *The Semantic Web: ESWC 2018 Satellite Events - ESWC 2018 Satellite Events, Heraklion, Crete, Greece, June 3-7, 2018, Revised Selected Papers.* 2018. URL: `https://doi.org/10.1007/978-3-319-98192-5_37` (cit. on p. 141).

[Mal+11] Alejandro Mallea et al. “On Blank Nodes”. In: *The Semantic Web - ISWC 2011 - 10th International Semantic Web Conference, Bonn, Germany, October 23-27, 2011, Proceedings, Part I.* 2011. URL: `https://doi.org/10.1007/978-3-642-25073-6_27` (cit. on pp. 50, 58, 70, 119).

[MB+97] John McCarthy, Sasa Buvac, et al. “Formalizing context”. In: *Computing Natural Language, Stanford University* (1997) (cit. on p. 219).

[MBS15] Farzaneh Mahdisoltani, Joanna Biega, and Fabian M. Suchanek. “YAGO3: A Knowledge Base from Multilingual Wikipedias”. In: *Seventh Biennial Conference on Innovative Data Systems Research, CIDR 2015, Asilomar, CA, USA, January 4-7, 2015, Online Proceedings.* 2015. URL: `http://cidrdb.org/cidr2015/Papers/CIDR15%5C_Paper1.pdf` (cit. on pp. 152, 229).

[McC87] John McCarthy. “Generality in Artificial Intelligence”. In: *Commun. ACM* (1987). URL: `https://doi.org/10.1145/33447.33448` (cit. on p. 222).

[McC93] John McCarthy. “Notes on formalizing context”. In: (1993) (cit. on p. 227).

[MD06] Mauro Mazzieri and Aldo Franco Dragoni. “A fuzzy semantics for the resource description framework”. In: *Uncertainty Reasoning for the Semantic Web I*. 2006 (cit. on p. 114).

[Mee+17] Ben De Meester et al. “Detailed Provenance Capture of Data Processing”. In: *Proceedings of the First Workshop on Enabling Open Semantic Science co-located with 16th International Semantic Web Conference (ISWC 2017), Vienna, Austria, October 21st, 2017*. 2017. URL: `http://ceur-ws.org/Vol-1931/paper-05.pdf` (cit. on pp. 113, 143).

[Mel13] Gerard de Melo. “Not Quite the Same: Identity Constraints for the Web of Linked Data”. In: *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence, July 14-18, 2013, Bellevue, Washington, USA*. 2013. URL: `http://www.aaai.org/ocs/index.php/AAAI/AAAI13/paper/view/6491` (cit. on pp. 186, 201).

[MF08] Robert E. McGrath and Joe Futrelle. “Reasoning about Provenance with OWL and SWRL Rules”. In: *AI Meets Business Rules and Process Management, Papers from the 2008 AAAI Spring Symposium, Technical Report SS-08-01, Stanford, California, USA, March 26-28, 2008*. 2008. URL: `http://www.aaai.org/Library/Symposia/Spring/2008/ss08-01-011.php` (cit. on p. 232).

[MH19] Bassem Makni and James A. Hendler. “Deep learning for noise-tolerant RDFS reasoning”. In: *Semantic Web* (2019). URL: `https://doi.org/10.3233/SW-190363` (cit. on pp. 243, 244).

[MH69] John McCarthy and Patrick J. Hayes. “Some Philosophical Problems from the Standpoint of Artificial Intelligence”. In: *Machine Intelligence 4*. 1969 (cit. on pp. 143, 225).

[MK03] Robert M. MacGregor and In-Young Ko. “Representing Contextualized Data using Semantic Web Tools”. In: *PSSS1 - Practical and Scalable Semantic Systems, Proceedings of the First International Workshop on Practical and Scalable Semantic Systems, Sanibel Island, Florida, USA, October 20, 2003*. 2003. URL: `http://ceur-ws.org/Vol-89/macgregor-et-al.pdf` (cit. on pp. 151, 173, 208, 224).

[MKT17] Maximilian Marx, Markus Krötzsch, and Veronika Thost. “Logic on MARS: Ontologies for Generalised Property Graphs”. In: *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017.* 2017. URL: https://doi.org/10.24963/ijcai.2017/165 (cit. on p. 233).

[MM10] Jamie P. McCusker and Deborah L. McGuinness. “Towards Identity in Linked Data”. In: *Proceedings of the 7th International Workshop on OWL: Experiences and Directions (OWLED 2010), San Francisco, California, USA, June 21-22, 2010.* 2010. URL: http://ceur-ws.org/Vol-614/owled2010_submission_12.pdf (cit. on p. 201).

[Mor07] Amadis Antonio Martinez Morales. “A Directed Hypergraph Model for RDF”. In: *Proceedings of the KWEPSY 2007 Knowledge Web PhD Symposium 2007, Innsbruck, Austria, June 6, 2007.* 2007. URL: http://ceur-ws.org/Vol-275/paper24.pdf (cit. on p. 244).

[MP20] David Martin and Peter F. Patel-Schneider. “Wikidata Constraints on MARS”. In: *Proceedings of the 1st Wikidata Workshop (Wikidata 2020) co-located with 19th International Semantic Web Conference(OPub 2020), Virtual Conference, November 2-6, 2020.* 2020. URL: http://ceur-ws.org/Vol-2773/paper-12.pdf (cit. on p. 233).

[MPP12] Boris Motik, Peter F. Patel-Schneider, and Bijan Parsia. *OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax (Second Edition).* 2012. URL: https://www.w3.org/TR/owl2-syntax/ (visited on 09/01/2022) (cit. on p. 78).

[MT18] Michalis Mountantonakis and Yannis Tzitzikas. “LODsyndesis: The Biggest Knowledge Graph of the Linked Open Data Cloud that Includes all Inferred Equivalence Relationships”. In: *ERCIM News* (2018). URL: https://ercim-news.ercim.eu/en114/r-i/lodsyndesis-the-biggest-knowledge-graph-of-the-linked-open-data-cloud-that-includes-all-inferred-equivalence-relationships (cit. on p. 185).

[MTS15] Michael R. Margitus, Gregory Tauer, and Moises Sudit. “RDF versus attributed graphs: The war for the best graph representation”. In: *18th International Conference on Information Fusion, FUSION 2015, Washington, DC, USA, July 6-9, 2015.* 2015. URL: https://ieeexplore.ieee.org/document/7266563/ (cit. on pp. 163, 164).

[MW08] Gerard de Melo and Gerhard Weikum. “Language as a Foundation of the Semantic Web”. In: *Proceedings of the Poster and Demonstration Session at the 7th International Semantic Web Conference (ISWC2008), Karlsruhe, Germany, October 28, 2008.* 2008. URL: http://ceur-ws.org/Vol-401/iswc2008pd%5C_submission%5C_77.pdf (cit. on p. 202).

[NBS14] Vinh Nguyen, Olivier Bodenreider, and Amit P. Sheth. “Don’t like RDF reification?: making statements about statements using singleton property”. In: *23rd International World Wide Web Conference, WWW ’14, Seoul, Republic of Korea, April 7-11, 2014*. 2014. URL: `https://doi.org/10.1145/2566486.2567973` (cit. on pp. 145, 146, 147, 149).

[New02] Steven R. Newcomb. “Preemptive reification”. In: *International Semantic Web Conference*. Springer. 2002 (cit. on p. 133).

[Ngu+15] Vinh Nguyen et al. “On Reasoning with RDF Statements about Statements using Singleton Property Triples”. In: *CoRR* (2015). URL: `http://arxiv.org/abs/1509.04513` (cit. on pp. 147, 148).

[Ngu+16] Vinh Nguyen et al. “A Formal Graph Model for RDF and Its Implementation”. In: *CoRR* (2016). URL: `http://arxiv.org/abs/1606.00480` (cit. on pp. 149, 164, 243).

[Ngu+19] Vinh Nguyen et al. “Singleton Property Graph: Adding A Semantic Web Abstraction Layer to Graph Databases”. In: *Proceedings of the Blockchain enabled Semantic Web Workshop (BlockSW) and Contextualized Knowledge Graphs (CKG) Workshop co-located with the 18th International Semantic Web Conference, BlockSW/CKGISWC 2019, Auckland, New Zealand, October 27, 2019*. 2019. URL: `http://ceur-ws.org/Vol-2599/CKG2019_paper_4.pdf` (cit. on pp. 149, 164, 236).

[Nic07] Matthias Nickles. “Social acquisition of ontologies from communication processes”. In: *Appl. Ontology* (2007). URL: `http://content.iospress.com/articles/applied-ontology/ao040` (cit. on pp. 115, 229).

[Noy+06] Natasha Noy et al. *Defining N-ary Relations on the Semantic Web: Use With Individuals*. 2006. URL: `https://www.w3.org/TR/swbp-n-aryRelations/` (cit. on pp. 42, 45, 48, 103, 104, 105).

[Noy+19] Natasha F. Noy et al. “Industry-scale Knowledge Graphs: Lessons and Challenges”. In: *ACM Queue* (2019). URL: `https://doi.org/10.1145/3329781.3332266` (cit. on p. 121).

[NS17] Vinh Nguyen and Amit P. Sheth. “Logical Inferences with Contexts of RDF Triples”. In: *CoRR* (2017). URL: `http://arxiv.org/abs/1701.05724` (cit. on pp. 145, 147, 148, 172).

[OGO20] Fabrizio Orlandi, Damien Graux, and Declan O’Sullivan. “How Many Stars Do You See in This Constellation?” In: *The Semantic Web: ESWC 2020 Satellite Events - ESWC 2020 Satellite Events, Heraklion, Crete, Greece, May 31 - June 4, 2020, Revised Selected Papers*. 2020. URL: `https://doi.org/10.1007/978-3-030-62327-2_30` (cit. on p. 156).

[Par90] Terence Parsons. “Events in the semantics of English: A study in subatomic semantics”. In: (1990) (cit. on p. 103).

[Pat05] Peter F. Patel-Schneider. “Building the Semantic Web Tower from RDF Straw”. In: *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30 - August 5, 2005*. 2005. URL: http://ijcai.org/Proceedings/05/Papers/0319.pdf (cit. on pp. 70, 81, 82).

[Pat10] Peter F. Patel-Schneider. “RDF: Back to the Graph”. In: (2010). URL: www.w3.org/2009/12/rdf-ws/papers/ws05 (cit. on pp. 67, 81, 82).

[Pat14] Peter F. Patel-Schneider. “Analyzing Schema.org”. In: *The Semantic Web - ISWC 2014 - 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part I*. 2014. URL: https://doi.org/10.1007/978-3-319-11964-9%5C_17 (cit. on p. 211).

[Pat15] Peter F. Patel-Schneider. “Using Description Logics for RDF Constraint Checking and Closed-World Recognition”. In: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*. 2015. URL: http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9531 (cit. on p. 136).

[Pat18] Peter F. Patel-Schneider. “Contextualization via Qualifiers”. In: *Joint Proceedings of the International Workshops on Contextualized Knowledge Graphs, and Semantic Statistics co-located with 17th International Semantic Web Conference (ISWC 2018)*. 2018. URL: http://ceur-ws.org/Vol-2317/article-01.pdf (cit. on pp. 146, 208, 263).

[Pep00] Steve Pepper. *The TAO of topic maps*. 2000. URL: https://www.ontopia.net/topicmaps/materials/tao.pdf (visited on 09/13/2022) (cit. on p. 133).

[Per85] Donald Perlis. “Languages with self-reference I: Foundations”. In: *Artificial Intelligence* (1985) (cit. on pp. 80, 86).

[Pet+12] David Peterson et al. *W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes*. 2012. URL: https://www.w3.org/TR/xmlschema11-2/ (visited on 09/28/2022) (cit. on p. 140).

[PF02] Peter F. Patel-Schneider and Dieter Fensel. “Layering the Semantic Web: Problems and Directions”. In: *The Semantic Web - ISWC 2002, First In ternational Semantic Web Conference, Sardinia, Italy, June 9-12, 2002, Proceedings*. 2002. URL: https://doi.org/10.1007/3-540-48005-6%5C_4 (cit. on pp. 70, 80, 110, 196).

[PH07] Peter F. Patel-Schneider and Ian Horrocks. “A comparison of two modelling paradigms in the Semantic Web”. In: *J. Web Semant.* (2007). URL: `https://doi.org/10.1016/j.websem.2007.09.004` (cit. on pp. 65, 69, 71, 76, 84, 85, 232, 355).

[PM20] Peter F. Patel-Schneider and David Martin. “Wikidata on MARS”. In: *CoRR* (2020). URL: `https://arxiv.org/abs/2008.06599` (cit. on p. 233).

[Pol+20] Axel Polleres et al. “A more decentralized vision for Linked Data”. In: *Semantic Web* (2020). URL: `https://doi.org/10.3233/SW-190380` (cit. on p. 241).

[PP04] Bijan Parsia and Peter F. Patel-Schneider. “Meaning and the semantic web”. In: *Proceedings of the 13th international conference on World Wide Web - Alternate Track Papers & Posters, WWW 2004, New York, NY, USA, May 17-20, 2004*. 2004. URL: `https://doi.org/10.1145/1013367.1013448` (cit. on pp. 78, 79, 231).

[Pre+16] Valentina Presutti et al. “The Role of Ontology Design Patterns in Linked Data Projects”. In: *Conceptual Modeling - 35th International Conference, ER 2016, Gifu, Japan, November 14-17, 2016, Proceedings*. 2016. URL: `https://doi.org/10.1007/978-3-319-46397-1_9` (cit. on p. 134).

[PRF12] Jiri Prochazka, RichardCyganiak, and Bob Ferris. *Property Reification RDF Vocabulary*. 2012. URL: `https://www.w3.org/wiki/PropertyReificationVocabulary` (visited on 11/03/2022) (cit. on p. 206).

[PS08] Eric Prud’hommeaux and Andy Seaborne. *SPARQL Query Language for RDF*. 2008. URL: `https://www.w3.org/TR/rdf-sparql-query/` (visited on 09/01/2022) (cit. on p. 92).

[Raa+18] Joe Raad et al. “Detecting Erroneous Identity Links on the Web Using Network Metrics”. In: *The Semantic Web - ISWC 2018 - 17th International Semantic Web Conference, Monterey, CA, USA, October 8-12, 2018, Proceedings, Part I*. 2018. URL: `https://doi.org/10.1007/978-3-030-00671-6%5C_23` (cit. on p. 188).

[Raa+19] Joe Raad et al. “The sameAs Problem: A Survey on Identity Management in the Web of Data”. In: *CoRR* (2019). URL: `http://arxiv.org/abs/1907.10528` (cit. on pp. 28, 184, 186, 187).

[Reb+16] Thomas Rebele et al. “YAGO: A Multilingual Knowledge Base from Wikipedia, Wordnet, and Geonames”. In: *The Semantic Web - ISWC 2016 - 15th International Semantic Web Conference, Kobe, Japan, October 17-21, 2016, Proceedings, Part II*. 2016. URL: `https://doi.org/10.1007/978-3-319-46547-0_19` (cit. on pp. 120, 357).

[RH22] Jos de Roo and Patrick Hochstenbach. *RDF Surfaces Primer.* 2022. URL: https://josd.github.io/surface/ (visited on 11/04/2022) (cit. on p. 199).

[RLH22] Kashif Rabbani, Matteo Lissandrini, and Katja Hose. "SHACL and ShEx in the Wild: A Community Survey on Validating Shapes Generation and Adoption". In: (2022) (cit. on pp. 136, 137).

[RMH17] Jacobo Rouces, Gerard de Melo, and Katja Hose. "FrameBase: Enabling integration of heterogeneous knowledge". In: *Semantic Web* (2017). URL: https://doi.org/10.3233/SW-170279 (cit. on pp. 132, 181, 236).

[RPS17] Joe Raad, Nathalie Pernelle, and Fatiha Saís. "Detection of Contextual Identity Links in a Knowledge Base". In: *Proceedings of the Knowledge Capture Conference, K-CAP 2017, Austin, TX, USA, December 4-6, 2017.* 2017. URL: https://doi.org/10.1145/3148011.3148032 (cit. on p. 187).

[Rul+12] Anisa Rula et al. "On the Diversity and Availability of Temporal Information in Linked Open Data". In: *The Semantic Web - ISWC 2012 - 11th International Semantic Web Conference, Boston, MA, USA, November 11-15, 2012, Proceedings, Part I.* 2012. URL: https://doi.org/10.1007/978-3-642-35176-1_31 (cit. on pp. 113, 355).

[Sah+09] Satya S. Sahoo et al. "Prom: A semantic web framework for provenance management in science". In: (2009) (cit. on p. 216).

[Sah+10] Satya Sanket Sahoo et al. "Provenance Context Entity (PaCE): Scalable Provenance Tracking for Scientific RDF Data". In: *Scientific and Statistical Database Management, 22nd International Conference, SSDBM 2010, Heidelberg, Germany, June 30 - July 2, 2010. Proceedings.* 2010. URL: https://doi.org/10.1007/978-3-642-13818-8_32 (cit. on pp. 142, 150).

[Sak+18] Sherif Sakr et al. "Provenance Management for Linked Data". In: *Linked Data.* 2018 (cit. on pp. 235, 238).

[SAM12] Juan F. Sequeda, Marcelo Arenas, and Daniel P. Miranker. "On Directly Mapping Relational Databases to RDF and OWL (Extended Version)". In: *CoRR* (2012). URL: http://arxiv.org/abs/1202.3667 (cit. on p. 210).

[SB04] Luciano Serafini and Paolo Bouquet. "Comparing formal theories of context in AI". In: *Artif. Intell.* (2004). URL: https://doi.org/10.1016/j.artint.2003.11.001 (cit. on pp. 219, 222).

[SBP14] Max Schmachtenberg, Christian Bizer, and Heiko Paulheim. "Adoption of the Linked Data Best Practices in Different Topical Domains". In: *The Semantic Web - ISWC 2014 - 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Pro-*

ceedings, Part I. 2014. URL: `https://doi.org/10.1007/978-3-319-11964-9%5C_16` (cit. on p. 189).

[SC08] Leo Sauermann and Richard Cyganiak. *Cool URIs for the Semantic Web.* 2008. URL: `https://www.w3.org/TR/cooluris/` (visited on 09/01/2022) (cit. on p. 32).

[Sch+08] Bernhard Schueler et al. "Querying for meta knowledge". In: *Proceedings of the 17th International Conference on World Wide Web, WWW 2008, Beijing, China, April 21-25, 2008.* 2008. URL: `https://doi.org/10.1145/1367497.1367582` (cit. on pp. 174, 245).

[Sch+13] Andreas Scheuermann et al. "An empirical perspective on representing time". In: *Proceedings of the 7th International Conference on Knowledge Capture, K-CAP 2013, Banff, Canada, June 23-26, 2013.* 2013. URL: `https://doi.org/10.1145/2479832.2479854` (cit. on pp. 113, 139, 180, 277, 355).

[Sch+15] Alexander Schätzle et al. "S2X: Graph-Parallel Querying of RDF with GraphX". In: *Biomedical Data Management and Graph Online Querying - VLDB 2015 Workshops, Big-O(Q) and DMAH, Waikoloa, HI, USA, August 31 - September 4, 2015, Revised Selected Papers.* 2015. URL: `https://doi.org/10.1007/978-3-319-41576-5_12` (cit. on p. 241).

[Sch+16] Alexander Schätzle et al. "S2RDF: RDF Querying with SPARQL on Spark". In: *Proc. VLDB Endow.* (2016). URL: `http://www.vldb.org/pvldb/vol9/p804-schaetzle.pdf` (cit. on p. 39).

[Sch+20] Christoph G Schuetz et al. "Knowledge Graph OLAP". In: *Semantic Web* (2020) (cit. on pp. 215, 228).

[SCS13] Robert Sanderson, Paolo Ciccarese, and Herbert Van de Sompel. "Designing the W3C Open Annotation Data Model". In: *CoRR* (2013). URL: `http://arxiv.org/abs/1304.6709` (cit. on p. 209).

[SD02] Michael Sintek and Stefan Decker. "TRIPLE - A Query, Inference, and Transformation Language for the Semantic Web". In: *The Semantic Web - ISWC 2002, First International Semantic Web Conference, Sardinia, Italy, June 9-12, 2002, Proceedings.* 2002. URL: `https://doi.org/10.1007/3-540-48005-6%5C_28` (cit. on pp. 69, 232).

[Sen+20] Sangeeta Sen et al. "State-of-the-art approaches for meta-knowledge assertion in the web of data". In: *IETE Technical Review* (2020) (cit. on p. 181).

[Seq+19] Juan F. Sequeda et al. "A Pay-as-you-go Methodology to Design and Build Enterprise Knowledge Graphs from Relational Databases". In: *The Semantic Web - ISWC 2019 - 18th International Semantic Web Conference, Auckland, New Zealand, October 26-30, 2019,* *Proceedings, Part II.* 2019. URL: https://doi.org/10.1007/978-3-030-30796-7_32 (cit. on p. 352).

[SH12] Luciano Serafini and Martin Homola. "Contextualized knowledge repositories for the Semantic Web". In: *J. Web Semant.* (2012). URL: https://doi.org/10.1016/j.websem.2011.12.003 (cit. on pp. 223, 224, 228).

[She+21] Kartik Shenoy et al. "A Study of the Quality of Wikidata". In: *CoRR* (2021). URL: https://arxiv.org/abs/2107.00156 (cit. on p. 211).

[SHH19] Cogan Shimizu, Quinn Hirt, and Pascal Hitzler. "MODL: A Modular Ontology Design Library". In: *Proceedings of the 10th Workshop on Ontology Design and Patterns (WOP 2019) co-located with 18th International Semantic Web Conference (ISWC 2019), Auckland, New Zealand, October 27, 2019.* 2019. URL: http://ceur-ws.org/Vol-2459/paper4.pdf (cit. on p. 135).

[Shn96] Ben Shneiderman. "The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations". In: *Proceedings of the 1996 IEEE Symposium on Visual Languages, Boulder, Colorado, USA, September 3-6, 1996.* 1996. URL: https://doi.org/10.1109/VL.1996.545307 (cit. on p. 296).

[Skj+18] Martin G. Skjæveland et al. "Practical ontology pattern instantiation, discovery, and maintenance with reasonable ontology templates". In: *International Semantic Web Conference.* Springer. 2018 (cit. on p. 135).

[SKW07] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. "Yago: a core of semantic knowledge". In: *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007.* 2007. URL: https://doi.org/10.1145/1242572.1242667 (cit. on pp. 120, 132, 176, 229).

[SL21] Juan Sequeda and Ora Lassila. "Designing and Building Enterprise Knowledge Graphs". In: *Synthesis Lectures on Data, Semantics, and Knowledge* (2021) (cit. on pp. 122, 210).

[SLS21] Philipp Seifer, Ralf Lämmel, and Steffen Staab. "ProGS: Property Graph Shapes Language (Extended Version)". In: *CoRR* (2021). URL: https://arxiv.org/abs/2107.05566 (cit. on p. 159).

[Sor+15] Alexandru Sorici et al. "CONSERT: Applying semantic web technologies to context modeling in ambient intelligence". In: *Comput. Electr. Eng.* (2015). URL: https://doi.org/10.1016/j.compeleceng.2015.03.012 (cit. on p. 171).

[Sow00] John F. Sowa. *Knowledge representation: logical, philosophical, and computational foundations.* 2000. ISBN: 0534949657. URL: https://www.worldcat.org/oclc/38239202 (cit. on p. 20).

[Sow03] John F Sowa. "Laws, facts, and contexts: Foundations for multimodal reasoning". In: *Knowledge Contributors.* 2003 (cit. on pp. 221, 277).

[Sow06] John F. Sowa. "Worlds, Models and Descriptions". In: *Stud Logica* (2006). URL: https://doi.org/10.1007/s11225-006-9012-y (cit. on p. 261).

[Sow07] John F. Sowa. "Fads and Fallacies about Logic". In: *Intelligent Systems, IEEE* (Apr. 2007) (cit. on pp. 66, 67).

[SP14] Ralph Schäfermeier and Adrian Paschke. "Aspect-Oriented Ontologies: Dynamic Modularization Using Ontological Metamodeling". In: *Formal Ontology in Information Systems - Proceedings of the Eighth International Conference, FOIS 2014, September, 22-25, 2014, Rio de Janeiro, Brazil.* 2014. URL: https://doi.org/10.3233/978-1-61499-438-1-199 (cit. on p. 134).

[SP18] Ralph Schäfermeier and Adrian Paschke. "Aspect-Oriented Ontology Development". In: *Synergies Between Knowledge Engineering and Software Engineering.* 2018 (cit. on p. 134).

[SP20] Leslie F. Sikos and Dean Philp. "Provenance-Aware Knowledge Representation: A Survey of Data Models and Contextualized Knowledge Graphs". In: *Data Sci. Eng.* (2020). URL: https://doi.org/10.1007/s41019-020-00118-0 (cit. on p. 181).

[SPH19] Ralph Schäfermeier, Adrian Paschke, and Heinrich Herre. "Ontology Design Patterns for Representing Context in Ontologies using Aspect Orientation". In: *Proceedings of the 10th Workshop on Ontology Design and Patterns (WOP 2019) co-located with 18th International Semantic Web Conference (ISWC 2019), Auckland, New Zealand, October 27, 2019.* 2019. URL: http://ceur-ws.org/Vol-2459/paper3.pdf (cit. on p. 134).

[SPV12] Juan F. Sequeda, Freddy Priyatna, and Boris Villazón-Terrazas. "Relational Database to RDF Mapping Patterns." In: *WOP.* 2012 (cit. on p. 210).

[SR14] Guus Schreiber and Yves Raimond. *RDF 1.1 Primer.* 2014. URL: https://www.w3.org/TR/rdf11-primer/ (visited on 09/01/2022) (cit. on p. 52).

[Ste00] Friedrich Steimann. "On the representation of roles in object-oriented and conceptual modelling". In: *Data Knowl. Eng.* (2000). URL: https://doi.org/10.1016/S0169-023X(00)00023-9 (cit. on p. 204).

[Sti05] Patrick Stickler. *CBD - Concise Bounded Description.* 2005. URL: http://www.w3.org/Submission/CBD/ (visited on 09/13/2022) (cit. on p. 178).

[Sto+06] Heiko Stoermer et al. “rdf and Contexts: Use of sparql and Named Graphs to Achieve Contextualization”. In: *Proc. of 2006 Jena User Conference*. 2006 (cit. on pp. 166, 168).

[Str+10] Umberto Straccia et al. “A General Framework for Representing and Reasoning with Annotated Semantic Web Data”. In: *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010*. 2010. URL: `http://www.aaai.org/ocs/index.php/AAAI/AAAI10/paper/view/1590` (cit. on pp. 224, 226, 349).

[Str09] Umberto Straccia. “A Minimal Deductive System for General Fuzzy RDF”. In: *Web Reasoning and Rule Systems, Third International Conference, RR 2009, Chantilly, VA, USA, October 25-26, 2009, Proceedings*. 2009. URL: `https://doi.org/10.1007/978-3-642-05082-4_12` (cit. on p. 114).

[Suc20] Fabian M. Suchanek. “The Need to Move beyond Triples”. In: *Proceedings of Text2Story - Third Workshop on Narrative Extraction From Texts co-located with 42nd European Conference on Information Retrieval, Text2StoryECIR 2020, Lisbon, Portugal, April 14th, 2020 [online only]*. 2020. URL: `http://ceur-ws.org/Vol-2593/paper12.pdf` (cit. on pp. 110, 115, 226, 233, 249).

[Tae+16] Ruben Taelman et al. “Continuous Client-Side Query Evaluation over Dynamic Linked Data”. In: *The Semantic Web - ESWC 2016 Satellite Events, Heraklion, Crete, Greece, May 29 - June 2, 2016, Revised Selected Papers*. 2016. URL: `https://doi.org/10.1007/978-3-319-47602-5_44` (cit. on pp. 112, 148, 152, 171, 181).

[Tae+17] Ruben Taelman et al. “Versioned Triple Pattern Fragments: A Low-cost Linked Data Interface Feature for Web Archives”. In: *Joint proceedings of the 3rd Workshop on Managing the Evolution and Preservation of the Data Web (MEPDaW 2017) and the 4th Workshop on Linked Data Quality (LDQ 2017) co-located with 14th European Semantic Web Conference (ESWC 2017), Portorož, Slovenia, May 28th-29th, 2017*. 2017. URL: `http://ceur-ws.org/Vol-1824/mepdaw_paper_1.pdf` (cit. on p. 115).

[Tae+18] Ruben Taelman et al. “Comunica: A Modular SPARQL Query Engine for the Web”. In: *The Semantic Web - ISWC 2018 - 17th International Semantic Web Conference, Monterey, CA, USA, October 8-12, 2018, Proceedings, Part II*. 2018. URL: `https://doi.org/10.1007/978-3-030-00668-6_15` (cit. on p. 241).

[Tae15] Ruben Taelman. “Continuously Updating Queries over Real-Time Linked”. In: (2015) (cit. on p. 180).

[Tam+10] Andrei Tamilin et al. "Context-Driven Semantic Enrichment of Italian News Archive". In: *The Semantic Web: Research and Applications, 7th Extended Semantic Web Conference, ESWC 2010, Heraklion, Crete, Greece, May 30 - June 3, 2010, Proceedings, Part I.* 2010. URL: `https://doi.org/10.1007/978-3-642-13486-9_25` (cit. on p. 228).

[Tao+10] Jiao Tao et al. "Integrity Constraints in OWL". In: *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010.* 2010. URL: `http://www.aaai.org/ocs/index.php/AAAI/AAAI10/paper/view/1931` (cit. on p. 136).

[TAT20] Dominik Tomaszuk, Renzo Angles, and Harsh Thakkar. "PGO: Describing Property Graphs in RDF". In: *IEEE Access* (2020). URL: `https://doi.org/10.1109/ACCESS.2020.3002018` (cit. on pp. 160, 161, 163).

[TB09] Jonas Tappolet and Abraham Bernstein. "Applied Temporal RDF: Efficient Temporal Querying of RDF Data with SPARQL". In: *The Semantic Web: Research and Applications, 6th European Semantic Web Conference, ESWC 2009, Heraklion, Crete, Greece, May 31-June 4, 2009, Proceedings.* 2009. URL: `https://doi.org/10.1007/978-3-642-02121-3_25` (cit. on p. 112).

[Tch+19] Andon Tchechmedjiev et al. "ClaimsKG: A Knowledge Graph of Fact-Checked Claims". In: *The Semantic Web - ISWC 2019 - 18th International Semantic Web Conference, Auckland, New Zealand, October 26-30, 2019, Proceedings, Part II.* 2019. URL: `https://doi.org/10.1007/978-3-030-30796-7%5C_20` (cit. on p. 115).

[TH20] Dominik Tomaszuk and David Hyland-Wood. "RDF 1.1: Knowledge Representation and Data Integration Language for the Web". In: *CoRR* (2020). URL: `http://arxiv.org/abs/2001.00432` (cit. on pp. 110, 191, 210).

[Tha+18] Harsh Thakkar et al. "Two for one: querying property graph databases using SPARQL via gremlinator". In: *Proceedings of the 1st ACM SIGMOD Joint International Workshop on Graph Data Management Experiences & Systems (GRADES) and Network Data Analytics (NDA), Houston, TX, USA, June 10, 2018.* 2018. URL: `https://doi.org/10.1145/3210259.3210271` (cit. on p. 165).

[TKK13] Johannes Trame, Carsten Keßler, and Werner Kuhn. "Linked Data and Time - Modeling Researcher Life Lines by Events". In: *Spatial Information Theory - 11th International Conference, COSIT 2013, Scarborough, UK, September 2-6, 2013. Proceedings.* 2013. URL: `https://doi.org/10.1007/978-3-319-01790-7_12` (cit. on pp. 103, 113, 150, 180).

[Tom16] Dominik Tomaszuk. "RDF Data in Property Graph Model". In: *Metadata and Semantics Research - 10th International Conference, MTSR 2016, Göttingen, Germany, November 22-25, 2016, Proceedings.* 2016. URL: https://doi.org/10.1007/978-3-319-49157-8_9 (cit. on p. 163).

[TPR12] Dominik Tomaszuk, Karol Pąk, and Henryk Rybińsk. "Trust in RDF Graphs". In: *Advances in Databases and Information Systems - 16th East European Conference, ADBIS 2012, Poznán, Poland, September 18-21, 2012. Proceedings II.* 2012. URL: https://doi.org/10.1007/978-3-642-32741-4_25 (cit. on p. 171).

[Tra+08] Thanh Tran et al. "Metalevel Information in Ontology-Based Applications". In: *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008.* 2008. URL: http://www.aaai.org/Library/AAAI/2008/aaai08-196.php (cit. on p. 224).

[Tum+05] Giovanni Tummarello et al. "Signing individual fragments of an RDF graph". In: *Proceedings of the 14th international conference on World Wide Web, WWW 2005, Chiba, Japan, May 10-14, 2005 - Special interest tracks and posters.* 2005. URL: https://doi.org/10.1145/1062745.1062848 (cit. on p. 178).

[TWS20] Thomas Pellissier Tanon, Gerhard Weikum, and Fabian M. Suchanek. "YAGO 4: A Reason-able Knowledge Base". In: *The Semantic Web - 17th International Conference, ESWC 2020, Heraklion, Crete, Greece, May 31-June 4, 2020, Proceedings.* 2020. URL: https://doi.org/10.1007/978-3-030-49461-2_34 (cit. on pp. 120, 157, 229).

[URS06] Octavian Udrea, Diego Reforgiato Recupero, and V. S. Subrahmanian. "Annotated RDF". In: *The Semantic Web: Research and Applications, 3rd European Semantic Web Conference, ESWC 2006, Budva, Montenegro, June 11-14, 2006, Proceedings.* 2006. URL: https://doi.org/10.1007/11762256_36 (cit. on pp. 116, 144, 220, 226).

[URS10] Octavian Udrea, Diego Reforgiato Recupero, and V. S. Subrahmanian. "Annotated RDF". In: *ACM Trans. Comput. Log.* (2010). URL: https://doi.org/10.1145/1656242.1656245 (cit. on pp. 116, 220, 224).

[Ver+14a] Ruben Verborgh et al. "Querying Datasets on the Web with High Availability". In: *The Semantic Web - ISWC 2014 - 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part I.* 2014. URL: https://doi.org/10.1007/978-3-319-11964-9_12 (cit. on p. 179).

[Ver+14b] Ruben Verborgh et al. “Web-Scale Querying through Linked Data Fragments”. In: *Proceedings of the Workshop on Linked Data on the Web co-located with the 23rd International World Wide Web Conference (WWW 2014), Seoul, Korea, April 8, 2014*. 2014. URL: http://ceur-ws.org/Vol-1184/ldow2014_paper_04.pdf (cit. on p. 180).

[Ver+16] Ruben Verborgh et al. “Triple Pattern Fragments: A low-cost knowledge graph interface for the Web”. In: *J. Web Semant.* (2016). URL: https://doi.org/10.1016/j.websem.2016.03.003 (cit. on p. 180).

[Ver19] Ruben Verborgh. *Shaping Linked Data apps*. 2019. URL: https://ruben.verborgh.org/blog/2019/06/17/shaping-linked-data-apps/ (visited on 10/02/2022) (cit. on pp. 136, 137).

[Vol+05] Max Volkel et al. “Semversion: A versioning system for rdf and ontologies”. In: *Proc. of ESWC*. 2005 (cit. on p. 115).

[Vrg+21] Domagoj Vrgoc et al. “MillenniumDB: A Persistent, Open-Source, Graph Database”. In: *CoRR* (2021). URL: https://arxiv.org/abs/2111.01540 (cit. on pp. 153, 164, 237, 239, 246, 271).

[VS20] Ruben Verborgh and Miel Vander Sande. “The Semantic Web identity crisis: In search of the trivialities that never were”. In: *Semantic Web* (2020). URL: https://doi.org/10.3233/SW-190372 (cit. on p. 84).

[Wan+13] Dong Wang et al. “S-store: An Engine for Large RDF Graph Integrating Spatial Information”. In: *Database Systems for Advanced Applications, 18th International Conference, DASFAA 2013, Wuhan, China, April 22-25, 2013. Proceedings, Part II*. 2013. URL: https://doi.org/10.1007/978-3-642-37450-0_3 (cit. on p. 112).

[Wel10] Chris Welty. “Context Slices: Representing Contexts in OWL”. In: *Proceedings of the 2nd International Workshop on Ontology Patterns - WOP2010, Shanghai, China, November 8, 2010*. 2010. URL: http://ceur-ws.org/Vol-671/pat01.pdf (cit. on p. 226).

[WF06] Christopher A. Welty and Richard Fikes. “A Reusable Ontology for Fluents in OWL”. In: *Formal Ontology in Information Systems, Proceedings of the Fourth International Conference, FOIS 2006, Baltimore, Maryland, USA, November 9-11, 2006*. 2006. URL: http://www.booksonline.iospress.nl/Content/View.aspx?piid=2209 (cit. on pp. 112, 139, 143, 220, 221, 225, 226, 277).

[WM18] Paul Warren and Paul Mulholland. “Using SPARQL - The Practitioners’ Viewpoint”. In: *Knowledge Engineering and Knowledge Management - 21st International Conference, EKAW 2018, Nancy, France, November 12-16, 2018, Proceedings*. 2018. URL: https://doi.org/10.1007/978-3-030-03667-6_31 (cit. on pp. 216, 217).

[Woo75] William A. Woods. “What’s in a link: Foundations for semantic networks”. In: *Representation and understanding.* 1975 (cit. on pp. 68, 261).

[Wyl+17] Marcin Wylot et al. “Storing, Tracking, and Querying Provenance in Linked Data”. In: *IEEE Trans. Knowl. Data Eng.* (2017). URL: `https://doi.org/10.1109/TKDE.2017.2690299` (cit. on pp. 113, 348).

[YK02] Guizhen Yang and Michael Kifer. “On the Semantics of Anonymous Identity and Reification”. In: *On the Move to Meaningful Internet Systems, 2002 - DOA/CoopIS/ODBASE 2002 Confederated International Conferences DOA, CoopIS and ODBASE 2002 Irvine, California, USA, October 30 - November 1, 2002, Proceedings.* 2002. URL: `https://doi.org/10.1007/3-540-36124-3_67` (cit. on p. 244).

[YK03] Guizhen Yang and Michael Kifer. “Reasoning about Anonymous Resources and Meta Statements on the Semantic Web”. In: *J. Data Semant.* (2003). URL: `https://doi.org/10.1007/978-3-540-39733-5_4` (cit. on pp. 66, 69, 80, 232).

[Zar09] Gian Piero Zarri. *Representation and Management of Narrative Information - Theoretical Principles and Implementation.* 2009. ISBN: 978-1-84800-077-3. URL: `https://doi.org/10.1007/978-1-84800-078-0` (cit. on p. 103).

[ZG17] Antoine Zimmermann and José M. Giménez-García. “Integrating Context of Statements within Description Logics”. In: *CoRR* (2017). URL: `http://arxiv.org/abs/1709.04970` (cit. on pp. 143, 181, 220, 222, 224, 226, 277, 278).

[ZH12] Jun Zhao and Olaf Hartig. “Towards Interoperable Provenance Publication on the Linked Data Web”. In: *WWW2012 Workshop on Linked Data on the Web, Lyon, France, 16 April, 2012.* 2012. URL: `http://ceur-ws.org/Vol-937/ldow2012-paper-03.pdf` (cit. on p. 209).

[Zim+09] Antoine Zimmermann et al. “Heterogeneity and Context in Semantic-Web-Enabled HCLS Systems”. In: *On the Move to Meaningful Internet Systems: OTM 2009, Confederated International Conferences, CoopIS, DOA, IS, and ODBASE 2009, Vilamoura, Portugal, November 1-6, 2009, Proceedings, Part II.* 2009. URL: `https://doi.org/10.1007/978-3-642-05151-7_30` (cit. on p. 223).

[Zim+12] Antoine Zimmermann et al. “A general framework for representing, reasoning and querying with annotated Semantic Web data”. In: *J. Web Semant.* (2012). URL: `https://doi.org/10.1016/j.websem.2011.08.006` (cit. on pp. 116, 151, 215, 220, 226, 246, 350).

[Zim07] Antoine Zimmermann. "Integrated Distributed Description Logics". In: *Proceedings of the 2007 International Workshop on Description Logics (DL2007), Brixen-Bressanone, near Bozen-Bolzano, Italy, 8-10 June, 2007*. 2007. URL: `http://ceur-ws.org/Vol-250/paper%5C_37.pdf` (cit. on p. 223).

[Zim12a] Antoine Zimmermann. *Dataset-semantics*. 2012. URL: `https://www.w3.org/2011/rdf-wg/wiki/TF-Graphs/Dataset-semantics` (visited on 09/01/2022) (cit. on pp. 98, 168).

[Zim12b] Antoine Zimmermann. *RDF-Datasets-Proposal*. 2012. URL: `https://www.w3.org/2011/rdf-wg/wiki/TF-Graphs/RDF-Datasets-Proposal` (visited on 09/01/2022) (cit. on pp. 98, 168).

[Zim13] Antoine Zimmermann. "Logical formalisms for agreement technologies". In: *Agreement technologies*. 2013 (cit. on pp. 222, 226).

[Zim14] Antoine Zimmermann. *RDF 1.1: on semantics of RDF datasets*. 2014. URL: `https://www.w3.org/TR/rdf11-datasets/` (visited on 09/01/2022) (cit. on pp. 48, 93, 98, 101, 198, 199).

Appendix A

Abstractions

The approaches presented in Part II employ a wide range of different perspectives, intuitions, techniques and lines of attack to manage complex knowledge structures in RDF. This appendix attempts to collect those aspects in one place. At first sight the sheer range of aspects covered may seem dizzying, if not outright disheartening. A closer look reveals that the situation is indeed even worse as many of those aspects don't work well along or even contradict each other. As unpleasant as this may appear to be, it can explain to some degree why it seems so hard to come up with a convincing proposal for one unifying metaphor and accompanying syntax for all meta-modelling problems in RDF. The abstractions collected in this appendix may serve as a guide to a thorough analysis of the problem space. They may help establish useful patterns and partitions and foster informed decisions about priorities. Hopefully they can guide comparisons of approaches and maybe even help identify possible combinations and extensions to cover also the more niche use cases. In any case they should help develop a good understanding of the complexity of the problem space.

A.1 Primitives

Chapter 4 on Representations has been structured by the kind of element that an approach uses to attribute specific parts of RDF data. On closer inspection however there are differentiations worth recording:

- A model may be based on quads or quins but nonetheless target individual statements. Extensions that extend the triple-based structure of RDF to quads or quins may technically be quite similar but nonetheless serve to address very different parts (and purposes) of some chunk of RDF data.
- A statement may be addressed in many different ways: by a property, an identifier, a graph containing it (and maybe only it), a quin that in the same step attributes it with start and end date or in many other ways.
- A property may be annotated itself without annotating the whole triple. Of course the triple then changes its meaning as well but only implicitly. This categorization is only concerned about explicit targets.

Approaches differ in which element(s) they make accessible to further attribution, ranging from individual terms to arbitrary groups of triples, from types to multiple occurrences, and of course combinations thereof. An annotation to a node in a triple implicitly also annotates the triple as a whole. An annotation to a node itself is not annotating a triple in which it occurs (it is quite ordinarily a part of the triple, just as any other node might be).

A.1.1 Model

While some approaches don't touch the basic triple formalism of RDF, many do stretch its boundaries and some even extend it by introducing new primitives.

triple
triple + id
quad e.g. named graphs per [Car+05b] or RDF 1.1. DataSet
quad + id e.g. RDF$^+$
n-tuple quins and beyond
extension e.g. RDF-star

A.1.2 Handle

The handle is the element by which some piece of RDF is addressed.

node
term
property
triple
quad

triple with id

- **triple in graph**

graph sets of triples, including singleton graphs that contain only one triple
quin triple in graph with id
n-tuple quins other than triple in graph with id, and beyond
struct groups of triples other than graphs

- **algorithmically specified set** e.g. Concise Bounded Description
- **arbitrary set** e.g. described through some syntactic feature like a pair of braces
- **list** as a native object
- **tree** as a native object
- **table** as a native object

model extension/new primitive e.g. rdf-star quoted triple as new node type
absolute ref distant reference possible or write access needed

A.1.3 Target

The target is the piece of RDF that is addressed by the handle. Contrary to the more syntactic perspective of Handle, the Target category also covers semantic aspects like the type/token distinction.[1]

node
term
property
triple

- **triple type**
- **triple occurrence** without handling multiset
- **triple multiset** occurrence including handling multiset

graph

- **graph type**
- **graph occurrence**

struct (same as in Handle)

[1]Note that if the target of an attribution is e.g. a term, the meaning of the statement that contains that term is affected too, naturally. That however is an implicit change of meaning, not an explicit attribution, and therefore not the topic of this categorization.

A.2 Content

Content is quite necessarily the broadest category of them all. It assembles all perspectives that are concerned about what is expressed — even if they consider structural elements part of the expression.

A.2.1 Dimensions

The following categories capture a good deal of contextual dimensions discussed in the literature. Determining how well they reflect actual need and usage 'in the wild' would require further investigation.

content topic ranging from informal tagging to categorization in a highly formalized upper ontology

- **viewpoint**
  - **bias**
  - **perspective**
  - **propositional attitude**
  - **underlying assumptions and theory**
- **situation**
  - **temporal** e.g. for what time the is statement true
  - **spatial** e.g. for what place the is statement true
  - **circumstances** e.g. under what circumstances the is statement true
- **background**
  - **assumptions implicit in context**
  - **common sense knowledge**
  - **modality** e.g. a fictional world like Southpark

metadata

- **provenance** of the statement in the realm of interpretation
  - **temporal** e.g. when was the statement asserted
  - **spatial** e.g. where does the statement originate from
  - **source** e.g. who asserted the statement
- **validity**
  - **temporal**
  - **spatial**
- **assessment**
  - **certainty**
  - **probability**
  - **credibility**
  - **reliability**
  - **fuzzyness**
  - **trust**

administration application & housekeeping

- **provenance** of the statement in the data
  - **temporal** e.g. ingest date
  - **spatial** e.g. sensor location
  - **source** e.g. IRI of data provider on the web
  - **process** e.g. derivation, inference
- **version**
- **access right**

technicalities although most of this could be encoded in datatypes

- **granularity** e.g. mappings as described by [GMF04]
- **language** e.g. of a web page
- **format** e.g. the encoding of the data

semantics

- **identity** either representation/syntax or interpretation/meaning

other

The less technical the categories, the harder they are to define:[2] e.g. the different subcategories of "viewpoint" or "background" might not seem immediately convincing to the reader and may well be understood as rather illustrating the range of possible semantics (although they were all taken from the approaches discussed before). That is a problem space of its own that is discussed in a bit more detail in Section 7 on Ontologies. A few clarifying remarks however should be added right here:

Identification It makes quite a difference if a provenance annotation is meant to refer to the author of the assertion described by a triple or to the author of the triple or to the publisher of that triple on the web. Yet, in practice, most of the time the distinction between content data, metadata and administrative data is only implicitly expressed at best and therefore in danger of being misunderstood or spurring unfounded conclusions.[3]

Viewpoints As discussed in Section 3.1.1.9 the representation of viewpoints is an important requirement on the Semantic Web. Naturally this field is so wide that it is impossible to abstract much further independently from any specific application.

Reasoning The subcategories 'process' (under 'administration-provenance') and especially 'identity' (under 'semantics') are motivated by the work of the RDF-star community group (see Section 4.2.4.5) that chose a quite peculiar semantics for the RDF-star quoted triple, namely that it be referentially opaque:

[2]Not to mention 'topic' for which it would seem quite impossible to give a sensible account of possible categories of values

[3]See Section 2.1

referring to an entity in the realm of interpretation, but only through one specific syntactic representation. The issue of explainable AI that that proposal hopes to facilitate has however seen broader interest e.g. in the work of [Ana+14] and also relates to e.g. the tracing of lineage discussed by [Wyl+17].

Temporal and Spatial Temporal and spatial dimensions are often considered as rather technical, well defined and easy to handle dimensions. However they do not only apply to categorically very different aspects of an assertion like when or where an assertion is valid, when or where it originates from, when or where it was processed in the application. They also can take quite different forms: a location may be rather abstractly described by geo-coordinates or datetime — but is the thing so described really a point in space and time, and how large or rather small is that point exactly? Is it indefinitely small, or does it rather stand in for a location or a time span? Is is described by exact coordinates or a specific time(span) or is it named as e.g. the Tour Eiffel or the Russian Revolution? Or is it a type like 'at sea', or even an indexical like 'around the corner', 'a few minutes ago' or 'a few minutes from here'? It's obvious how — as so often in the field of semantics — this seemingly clear-cut topic under closer inspection splinters into a sheer infinite spectrum of variations. Consequently simplification is indeed not only a virtue but a necessity but many of the works in Section 3 on Demand that claim to cover temporal or spatial aspects don't provide more than the most rudimentary semantics, leaving probably too many aspects undefined to be useful outside of well confined application scenarios. On the Semantic Web this can only work if those semantics are defined and expressed in either very specific vocabularies or through meta assertions — but then the question might become what was gained by such often quite involved temporal/spatial facilities in the first place.

temporal

- **point** e.g. a year, a date, an exact time
- **interval** defined by a pair of start and end points
- **period** e.g. 'WorldWar 2'
- **type** e.g. 'around noon'

spatial

- **point** e.g. precise geo coordinates
- **sector** e.g. a polygon defined by geo coordinates
- **area** e.g. a town square, a country
- **type** e.g. 'ìn bed'

A.2.2 Dimensionality

Some approaches cover only one or a specific set of dimensions while others are very flexible and extensible. Some require ordered domains, but on that basis support useful inferences.

arity of dimensions

- **single**
- **multiple**
- **unconstrained**

malleability

- **fixed**
- **extensible**
- **resilient** extensible by a third party
- **arbitrary**

metaMeta metadata as first class citizen

- **integration** no separation of object and meta knowledge
  Some approaches strive for seamless integration of data and metadata while others consider it a virtue to keep them apart
- **structure** attribution domain is not just flat
  - **order** the attribution domain is ordered
    Special orders are predominantly discussed in the context of annotating individual statements.
    - **partial order** AnnotatedRDF, Contextualized Knowledge Repository
    - **lattice** (see [Str+10])
    - **distributive** some context dimensions lend themselves to weak or strong distributive ordering, some not — see [Hay07]
    - **consecutive** temporal
    - **neighbouring** spatial
  - **composition** attribution dimensions can be composed
    Composition of attribution domains is often found with contextualization-focused approaches, presumably as the effort required to compose an attribution domain seems more justified for sets of statements than for individual triples.
    - **nesting**
      - **inheritance**
      - **negation** nested negated contexts enable FOL expressivity
    - **mapping**
    - **merging**
    - **lifting**

A.2.3 Structure

Some approaches focus on structural aspects as a means to partition content for easier annotation. Some take a more opportunistic stance and interpret structural characteristics as attributional domains themselves. Some approaches aim at mitigating the structural weaknesses of RDF — especially w.r.t. n-ary relations and the demarcation of complex objects — via attribution techniques. Some focus on overcoming those structural weaknesses as a means to disambiguate between attributes of objects and relations between objects.

arity of statements how many statements can be attributed

- **singular** a single statement
- **n-ary relation** a statement with e.g. a structured value as object[4]
- **multiple** statements grouped as graphs or by other means

structures

- **relations**
  - **n-ary relation with primary value** (davidsonian)
  - **n-ary relation without a primary value** (neo-Davidsonian)
  - **n-ary relation with a list as value**
  - **n-ary relations branching out further than one level**
  - **objectified/classified/complex relation** e.g. Topic Maps
  - **attributed relations** e.g. LPG
  - **shortcut relations** e.g. [Car+19]
- **groups of statements**
  - **grouping syntax** e.g. named graphs
  - **grouping algorithm** e.g. Concise Bounded Description
  - **grouping axioms** e.g. OWL, Shapes
  - **grouping attributions** to demarcate complex objects
- **regularity**
  - **regular**
  - **irregular**

Honouring and exploiting regularity can go both ways. Regular data structures may constitute the main facts while irregular additional detail may be hard to normalize and is relegated to a secondary annotation space. On the other hand irregularly shaped data may constitute the interesting part and main fact whereas regular and additional, maybe purely administrative data is grouped in a context object. [Zim+12] is a good example for a focus on regularly shaped and ordered dimensions. Work on irregular dimensions like viewpoints, bias, argumentation etc. is much more rare.

[4]Lindström gave the examples `<book> bibo:authorList ( <alice> <bob> ).` and `<book> bf:title [bf:mainTitle 'The Bog of Meaning' ; bf:subtitle 'Adventures in semantic space'].`, `https://lists.w3.org/Archives/Public/public-rdf-star/2022Jan/0101.html`

A.3 Form

This section concentrates on more external aspects like design philosophies and usability.

A.3.1 Epistemology

Many approaches are guided by some overarching design idea — often a perceived foundational dichotomy — aiming at intuitive understanding and easy to use idioms. Those idioms do in general have merit and immediate appeal, but neither do they necessarily harmonize with or complement one another nor can any of them claim universal priority.

scope

- **annotation**
- **contextualization**

grouping

- **one or more facts** that share the same annotation
- **one or more annotations** as a situation/context/theory

mode

- **describe** attribute on a node
- **relate** relation between nodes

specificity

- **specificity of metadata**
  - **general dimensions** like time and space, also provenance
  - **specific dimensions** dependant on application, situation or topic
- **specificity of data**
  - **intrinsic detail**
  - **extrinsic context**

prioritization

- **primary** the essential core information that should be immediately visible and easy to query
- **secondary** additional details, nice-to-have but not essential, maybe administrative and/or repetitive

reach

- **global** a model of some domain commonly shared by all parties
- **local** a model informed by one party, but not shared by (and with) others

structureal regularity

- **regular**
- **irregular**

Scope The term ‘annotation’ often refers to attributions on single statements whereas ‘contextualization’ mostly refers to sets of statements that share a common context. That context may be composed of several attributes. But also a single statement may be annotated with a context that is composed of several attributes. So while intuitions about the meaning of those categories may be strong, in practice the distinction is less clear-cut.

Grouping This category is closely related to the category of scope. Interestingly it goes both ways: grouping can mean grouping of several annotations on one statement or grouping of several statements under the same annotation - or both.

Mode [Gua09] assesses that “the main practical problem of binary relations, from the KR point of view, is [...]: how to distinguish between the relations which contribute to the internal structure of a concept and those which do not? Or, in other words, how to decide whether a piece of information should be modelled in terms of an attribute-value pair or in terms of a genuine relation?”[5] In the simplistic world of RDF every statement is equally weighted as one triple among many. But, as Guarino notes, some statements are more tightly connected in meaning as they commonly ‘describe’ an object whereas other statements ‘relate’ such objects to each other, as [Krö17] puts it. Other formalisms like Topic Maps (Section 4.1.1.3) and Labelled Property Graphs (LPG) (Section 4.2.4.6) have this distinction hard coded in their very core, inspiring the development of a dedicated design methodology by [Seq+19]. This category is not to be confused with attributions of relations themselves, which again Topic Maps and LPGs support to a much higher degree than RDF.

Specfcity The degree to which a piece of information is considered special in some context may be interpreted as a very natural way to partitioning an information space. On closer look however being ‘special’ has many different forms. Metadata dimensions vary greatly in ubiquity. Some approaches let this guide their design. When information does not belong to established metadata categories it is rather understood as ‘more data’ that may provide more detail to the subject at hand itself or add context and external attributes.

Prioritization A more general take on structuring the information domain is to distinguish between primary facts and secondary annotations. This view doesn’t subscribe to the special importance of any specific ordering scheme but focuses on the economies of attention in general. In standard RDF this would best be captured by n-ary relations with an `rdf:value` property pointing out the primary object. [GM16] stresses the importance of such easy to follow topical relations for realizing large integrated and decentralized knowledge

[5][Gua09] dives deep into the problems of formalizing knowledge representation and provides some useful hints w.r.t. to its fundamentals.

spaces. [Krö17] gives examples how context can be understood as subsuming all secondary aspects that 'enrich' the 'data', such as "temporal validity, provenance, trustworthiness, or other side conditions and details."

Reach [Bou+04] establishes the concept of global or local models, essential to contextualization as a means to disambiguate different world views, theories etc.

Structureal Regularity [Krö17] sees the distinction between regular and irregular data driving contextualization: "Contextual information is introduced for a variety of reasons. It may simply be convenient for storing additional details that would otherwise be hard to represent in normalisation, or it may by used for capturing meta-information such as provenance, trust, and temporal validity." He also notes that the irregular parts are sometimes "more similar to the modelling of n-ary relations."

A.3.2 Usability

There exists very little research on usability or on actual use of different representational approaches, not to mention empirically sound works. One might claim that the closest thing to an empirically sound usability study of the Semantic Web is the tremendous success of Property Graphs relative to RDF, at least when measured in sheer market share.

design philosophy

- **minimalistic flexibility**
- **opinionated guidance**

modelling

- **idiomatic** triple-centric, meets general expectations
- **intuitive** (even if not idiomatic)
- **unambiguous** attributions are precisely targeted
- **concise** syntactic elegance, support for namespaces, succinct identifiers
- **taciturnity** how many newly generated constants need to be introduced
- **late binding** stays out of view when not needed
- **polymorphism** attribution doesn't require re-write

querying

- **use cases** expressivity
  - **gathering all annotations on a specific...**
    - **node**
    - **triple**
    - **graph**
  - **gathering all ... with the same annotations**
    - **nodes**
    - **triples**
    - **graphs**
- **ease of use** self-evident or reliant on special query techniques
- **complexity**
- **query dominating** retrieve core and metadata together in one go[6]
- **property path support**

reasoning

- **OWL-friendliness**
- **expressivity**
- **ease of use**

Modelling Many authors mention for example the usability-problems of RDF standard reification, namely the introduction of 'heavy', involved and hard to comprehend data structures and indirections, that — although arguably

[6][HHK15]

idiomatic being based on triples — can hardly be considered intuitive and easy to use. However, attributions, which are considered more intuitive and concise, are often not precisely targeted. For example, most approaches that target either statements or graphs can't annotate an individual term without also annotating its peers. [CBR18] point out possible mismatches between provided syntax and intended meaning when an approach "proposes to add statements about statements. However, in an n-ary relation, the additional arguments usually characterize information about the relation instance itself, not the statements." W.r.t. polymorphism [Ang+17] note how Property Graphs (see Section 4.2.4.6) make it easy to add properties to an edge whereas in RDF the graph would need to be restructured, new blank nodes added etc. [GP13] call this *relation footprint preservation*: attributions preserve the relation topology and do not require changes to e.g. the parse tree and/or vocabulary constants

In an attempt to provide a sound basis for comparing different approaches [HTS10] propose to settle on a common example scenario covering a wide range of aspects of contextualization (see Section 12.4). [BGS13] uses that example to discuss the benefits of contextualization.

[Rul+12] investigate how temporal data is encoded in Linked Open Data and provide a both quantitative and qualitative analysis of approaches used in practice. [GP13] compare seven different ways to represent n-ary relations in OWL 2 along different axes, on the basis of a realistic example, via a qualitative discussion and in relation to the results of [Rul+12] mentioned above. [Sch+13] apply the design patterns developed by [GP13] in an empirical study to the representation of time, asking users to represent temporal information using the provided patterns.

A.3.2.1 Tool Support

Proper tooling can be very effective in improving the expressivity and usability of a language, especially for a rather low level formalism like RDF, as it may provide additional support where syntactic features alone become too cumbersome and involved. Some have been comparing editing RDF to coding in Assembler to stress the need for more user-friendly surface syntaxes that hide the complexities of triple-based structures. [PH07] provide a few hints in this direction.

A lot of effort has been spent on ontology visualization and ontology editors, which however this work doesn't cover at all. The VOILA! workshop series[7] may serve as an entry point for further exploration. Graph visualizations however are notorious for looking good with small example datasets but to not scale very well when the datasets get bigger.

A.4 Function

This section gathers aspects concerning functionality, features and performance.

[7]See `http://voila2021.visualdataweb.org/`

A.4.1 Semantics

Semantics was discussed in some detail in Section 2.6, especially Section 2.6.2 on the model theory of RDF. Fitting this complex topic into a rigid schema is quite impossible and the following should be understood as a list of issues to keep in mind rather than a stringent categorization.

- **monotonicity**
  - **annotation**
  - **qualification**
- **inference**
  - **paradigm**
    - **axiom** inference of new knowledge
    - **constraint** type safety
  - **reasoning support**
    - **supported entailment**
    - **fidelity of entailment**
- **instantiation**
  - **type**
  - **abstract (unasserted|hypothetical) type**
  - **token** occurrence
    - **set**
    - **multiset** annotating multiple occurrences of a triple
      - **merge**
      - **update/delete**
- **subclass relations**
  - **inheritance**
- **domain of discourse**
  - **representation** indication | expression | syntax | annotation | triple
  - **interpretation** denotation | proposition | meaning | qualification | fact
- **co-denotation**
  - **referentially opaque**
  - **referentially transparent**
- **on demand semantics** support for different entailment regimes

Multisets [Las+] discuss possible issues when merging, updating or deleting annotated statements, especially when the annotations actually target occurrences, not triple types.

A.4.2 Compatibility

Backward compatibility has always been a high priority in the development of RDF. In a decentralized setting aiming for integration of data from widely varying sources it is very important to make the entry barriers low and and don't break things that have been working already. On the other hand sometimes not breaking something requires so much effort and downstream expenditure that making a cut is the reasonable thing to do even if it hurts temporarily. So far RDF development has always stayed on the conservative side with a few extensions and very little, mostly bug-fixing changes. But there's also been forks like Property Graphs and calls for "NoRDF" solutions like [Reb+16] that suggest that some radical change may be unavoidable.

Model

- **backward compatible** reform rather than revolution, extension rather than change
- **idiomatic** composing classic triples rather than extending the model
- **ontology patterns** established patterns remain valid and well supported

Syntax

- **compatibility with syntaxes** especially N-Triples, N-Quads, Turtle, TriG, JSON-LD
- **extensions required**

Semantics compatibility with RDF 1.x semantics

- **model theory adopted unchanged**
- **model theory completed/extended**
  - **statement types and tokens** disambiguation provided
  - **graph naming semantics**
  - **identification per denotation and indication** disambiguation provided
  - **extensible to full FOL**
- **model theory modified**

Querying

- **resilience** do pre-existing queries remain valid when data is attributed
- **SPARQL extension required**
- **SPARQL extension provided**

Reasoning standard OWL reasoning supported

Implementations

- **no extensions required** runs on standard RDF software
- **extensions required**
- **extensions provided**
- **reference implementation provided**

A.4.3 Metrics

Measurements are a delicate subject as they depend on so many factors that are very hard to get right. A slight variation in a scenario may lead to very different results. Differences in implementations can make an approach seem untractable on one system while perfectly valid on others: e.g. singleton properties perform miserably on some triple stores that use indexing schemes relying on the absolute number of properties being low — a not unreasonable assumption when the number of vocabularies used is limited, but performance-wise the death kiss to said approach. Even RDF standard reification performs very differently on different stores, hinting at the presence (or absence) of some optimizations under the hood.

Another question is how important differences in triple count or query performance are in practice. As hardware gets faster, interfaces can hide triple bloat to some extent. More important is probably the usability of an approach, its intuitiveness, clear semantics, a syntax that goes out of the way when not needed, easy queryability and straightforward reasoning support. All those factors can have a huge impact on the overall performance of a system where the human is more often than not the limiting factor, especially in a domain as hard to automate as knowledge representation. Of course, as far as really big data is concerned, nothing beats raw performance on the implementation level, but then again optimizations under the hood may be a more sensible approach than hard to author/query syntactic means.

efficiency

- **triple count** with or without syntactic sugar
  - **modelling**
  - **querying**
  - **reasoning**
- **usability** again very syntax dependent
  - **modelling** how involved is authoring an annotation
  - **querying** how involved is formulating a query
  - **reasoning** how involved is formulating an inference
  - **bnode count** as surrogate measure ([BGS13])
- **compatability**
  - **properties introduced** e.g. `singletonPropertyOf`
  - **constructs introduced** e.g. named graphs, quoted triples
- **implementation**
  - **additional indices required** e.g. on predicates

performance

- **querying**
- **reasoning** is decidability guaranteed