

A Comparison of Two Modelling Paradigms

Peter F. Patel-Schneider¹ and Ian Horrocks²

¹ Bell Labs Research
Lucent Technologies
Email: pfps@research.bell-labs.com

² School of Computer Science
University of Manchester
Email: horrocks@cs.man.ac.uk

Abstract. Standard logics and Datalog-related logics have both been proposed as underlying formalisms for the Semantic Web. Although these two different formalism groups have some commonalities, and look similar in the context of expressively-impooverished languages like RDF, their differences become apparent at more expressive language levels. After considering some of these differences, we argue that, although some of the characteristics of Datalog have their utility, the open environment of the Semantic Web is better served by standard logics.

1 Introduction

Two very different modelling paradigms have been proposed for the Semantic Web. One paradigm is based on notions from standard logics, such as propositional logic, first-order logic, and Description Logics [Baader *et al.* 2003]. This paradigm is embodied in the W3C-recommended Semantic Web languages, the Resource Description Framework (RDF) [Manola and Miller 2004], RDF Schema [Dan Brinkley and R. V. Guha 2004], and the OWL Web Ontology Language [Dean *et al.* 2004]. We will call this the Classical paradigm. The other paradigm is based on notions from object-oriented databases [Agrawal and Gehani 1989] and rule languages [Ullman 1988]. This paradigm is embodied in a previous version of RDF [Ora Lassila and Ralph R. Swick 1999] and several proposals for Semantic Web languages, including OWL Flight [de Bruijn *et al.* 2004]. We will call this the Datalog paradigm.

The best versions of both paradigms can be given a formal definition. The formal definition for the Classical paradigm comes from the standard model-theoretic basis for standard logics, as exemplified in the model theories for both RDF [Hayes 2004] and OWL [Patel-Schneider *et al.* 2004]. For the Datalog paradigm various versions of the Datalog formal basis [Ullman 1988] can be given, including formal bases that incorporate non-monotonic extensions to Datalog [Gelfond and Lifschitz 1988; Gelder *et al.* 1991].

There are significant differences between the two paradigms. These differences range from computational aspects of the paradigms, in various guises; to the expressive power of the paradigms; to the naturalness of modelling in the paradigms. Competing claims have been made concerning which of the two paradigms are better for representation and reasoning in the Semantic Web.

A recent paper by de Bruijn *et al.* [2005] has argued in favour of the Datalog paradigm, as embodied in OWL Flight. In this paper we argue that the paradigm based on standard logical notions is a better paradigm for the Semantic Web than the Datalog paradigm. We are not (just) arguing that OWL DL is better than OWL Flight. Instead we are arguing that the notions underlying standard logics are better suited to the Semantic Web than those underlying Datalog.

2 The Semantic Web

The World Wide Web has been a tremendous success, making an incredible range of information and services accessible to billions of users worldwide. In some respects, however, the Web is a victim of its own success, as it has become more and more difficult to manage the ever-increasing volume of available data, and to use it to perform more complex tasks than keyword based search and retrieval.

This problem is exacerbated by the unstructured nature of the web, and the focus of HTML on presentation rather than content. It has been suggested that Web resources could be made much more usable if “information was given a well defined meaning” [Berners-Lee *et al.* 2001] so that it formed a “logical web of data” [Berners-Lee 1998]; this idea/dream has become known as the Semantic Web.

Early work on realising the Semantic Web has focused on the development of languages such as RDF and OWL that could be used both to augment web content with “semantic markup” and to establish ontologies—vocabularies of terms with formally specified and machine accessible meanings that can be used in semantic markup.

One of the key factors in the success of the existing Web is its decentralised nature; this allows the Web to grow rapidly (sometimes in unforeseen directions) without the inertia that would inevitably result from the need to coordinate with a centralised authority. As an extension of the existing Web, the Semantic Web will have to exist within the same loosely organised framework, and so without the benefit of canonical names or authoritative sources of meaning: there may be many ontologies providing different, perhaps even conflicting, meanings for the same term. Using formal languages to define ontologies will mean, however, that such differences can be detected and analysed.

3 The Two Paradigms

3.1 The Classical Paradigm

The Classical paradigm, embodied in RDF and OWL, has its formal basis in standard model-theoretic accounts. The basic idea here, without going into too much detail, is that the domain being modelled is abstractly represented as a set of objects and relationships between them. There can be many (often potentially infinite) states of affairs, often called interpretations or models, each describing one possible state of the domain. For example, if we are modelling familial relationships, a person “Sam” might be represented by an object, and relationships like “sibling” and “parent” by relationships between (in this case pairs of) objects. In the absence of any other information, there are interpretations for every possible way that people can be related, including all sorts

of nonsensical sibling relationships such as someone being the sibling of one of their parents.

Ontologies consist of sets of statements (often called axioms) that describe characteristics that must be satisfied by (the ontology designer’s idea of) “reasonable” states of the world. Formally, such statements correspond to logical sentences, and an ontology corresponds to a logical theory. For example, our family ontology might include axioms asserting that all persons have exactly two parents, both of whom are themselves persons. Such an ontology would rule out interpretations where people had three parents, or where one of their parents is not a person. A “perfect” ontology, if such a thing were ever possible, would admit *all* reasonable interpretations and rule out *all* unreasonable ones. Note that this does not mean that only one interpretation would remain—our ontology may, for example, precisely describe the characteristics of familial relationships without saying anything about who is the parent of whom. Further details of this model-theoretic account of meaning are not relevant to this paper. What is relevant is that this account admits a multiplicity of (hopefully “reasonable”) interpretations, leaving open which one is the actual situation.

Information (e.g., an ontology) is separate from interpretations in this paradigm. The meaning of information is carried in the relationship between the information and the interpretations that support it. It is often useful to think of the meaning of information as corresponding to the set of interpretations that are consistent with it. Retrieval in this paradigm comes down to the task of checking if some situation holds in *all* interpretations that are consistent with the available information (i.e., logical entailment).

For example, if an ontology contains the information that Joe is married to Sam, that they are both employees of NewCo, and that only persons of opposite gender can be married, then we can return NewCo in response to a query for companies with both male and female employees, even if we don’t know the gender of either Sam or Joe—this is because in all interpretations where Sam is male, Joe must be female, and vice versa. On the other hand, we may not be able to return NewCo in response to a query for companies with at least two employees, as our ontology may not rule out interpretations in which people are both male and female at the same time and where Joe and Sam could be two names for the same person.

Throughout the paper we will be using a hybrid, informal notation for formulae (including rules), taking bits of syntax commonly used in accounts of first-order logic, Datalog, and Description Logics. We will use italicised letters (e.g., *x*) for variables, with implicit universal quantification in rules. Constants and properties will be written in typewriter font (e.g., `mother`).

3.2 The Datalog Paradigm

The relational model underlying databases also models the domain in terms of objects and relationships between them, but makes several simplifying assumptions. In particular, it is assumed that the only objects and relationships that exist in the domain are those that are explicitly represented in the database (the Closed World Assumption), and that names uniquely identify objects in the domain (the Unique Name Assumption). The result of these assumptions is that there is a *single* (canonical) model, where objects and

relationships are in a one to one correspondence with the data in the database. Given this close correspondence, it is often convenient to think of the data and the model as being the same thing.

In the database setting, retrieval only requires checking the structure of this single model (i.e., of the database itself). While this is *much* easier (requiring time polynomial in the size of the data) than computing entailment with respect to an ontology (at least PSpace-complete in the size of the data), it is not able to capture situations like the NewCo one above: it would, e.g., be necessary to state the gender of persons and/or be assumed that any person not known to be male must be female.

Datalog is a formalisation of the database approach in which Horn-like rules are used to capture both the *schema* (i.e., structural constraints on the data) and the data itself. For example, the rule

$$\text{Person}(y) \leftarrow \text{Person}(x) \wedge \text{parent}(x, y),$$

can be read as stating that, for any x and y , if x is a person, and y is the parent of x , then y is also a person. The antecedent ($\text{Person}(x) \wedge \text{parent}(x, y)$ in this case) is often called the *body* of the rule, and the consequent ($\text{Person}(y)$ in this case) is often called the *head* of the rule.

A rule with an empty head is often called a *constraint*,³ and is used to express the fact that interpretations satisfying the condition described in the body of the rule are not admitted. For example, the rule

$$\leftarrow \text{Person}(x) \wedge \text{parent}(x, x)$$

can be read as stating that no person can be their own parent.

Finally, rules with empty bodies are used to capture data (often called ground facts). For example, the rule

$$\text{married}(\text{Joe}, \text{Sam}) \leftarrow$$

can be read as stating that Joe is married to Sam.

It is normal to restrict rules to be *safe*. A safe rule is one where all of the variables in the head of a rule also occur in (positive atoms in) the body of the rule. For example, the rule

$$\text{mother}(x, y) \leftarrow \text{Person}(x)$$

(whose intended meaning is that every person has a mother) is unsafe, because the variable y in the head of the rule does not occur in the body of the rule.

The semantics of Datalog relies on minimal Herbrand models: essentially, interpretations where the objects and relationships are limited to those mentioned in rules, and the only facts are those that are implied by the rules (i.e., the closed world assumption). When combined with the unique name assumption, this means that a collection of

³ We will use the terminology from the database literature, and call these sorts of constructs constraints. We view other limitations on the permissible models as constraints as well, but have refrained from using this useful word in its more general meaning in order to emphasize the difference between database-style constraints and logical axioms.

Datalog rules (sometimes called a Datalog program) still admits (at most) only one interpretation, and so has similar characteristics to a database: retrieval is relatively easy (polynomial), but only relatively simple situations can be modelled, i.e., situations in which complete information is available. It cannot, for example, be used to capture information such as the fact that all persons are either male or female, or the fact that all persons have exactly two parents.

Datalog can be extended in a variety of directions, e.g., with negation as failure (in the body of rules) and disjunction (in the head of rules).⁴ Such extensions are often indicated using superscripted operators, e.g., $\text{Datalog}^{-\vee}$ for Datalog with negation as failure (NAF) and disjunction. For expressive extensions such as $\text{Datalog}^{-\vee}$ there may no longer be a single minimal interpretation. In this case the most commonly adopted semantics is to restrict attention to so-called “stable models” [Gelfond and Lifschitz 1991], i.e., interpretations where the interpretation of negated terms is fixed such that they are consistent with the rules.

For $\text{Datalog}^{-\vee}$ retrieval is no longer so easy (co-NP-complete in the general case), but it is possible to model some kinds of incomplete information, e.g., the fact that all persons are either male or female. It is still not possible, however, to capture the fact that all persons have exactly two parents. Other reasoning tasks in $\text{Datalog}^{-\vee}$, including many reasoning tasks useful when working with ontologies, are even harder (NEXPTIME^{NP}-complete in the general case; in fact, reasoning in the Description Logic *SHIQ* can be reduced to reasoning in Datalog^{\vee} [Hustadt *et al.* 2004]).

4 Conceptual Modelling in the Semantic Web

The differences between the Classical and Datalog paradigms have important consequences for modelling. The Datalog paradigm is, not surprisingly, well suited to closed and highly structured environments such as databases, where it is reasonable to assume that all relevant information is available. The Classical paradigm may have advantages, however, in an open and loosely structured environment such as the Semantic Web. In the following sections we will compare and contrast various aspects of the two paradigms, with particular reference to the kinds of situation that we can expect to arise in the Semantic Web.

4.1 Identifiers

An identifier is a name that is used to reference an individual (or property or class). Identifiers are not exactly part of the domain being modelled, instead providing a vocabulary of names that can be used to describe and refer to various aspects of the domain. They also provide a simple mechanism for establishing common references between different sources of information. One of the strengths of the Semantic Web is that it provides a nicely structured collection of identifiers in the form of IRI references.

As we saw in Section 3.2, the Datalog paradigm (along with many other representation formalisms) requires that different identifiers reference distinct individuals (the

⁴ Allowing disjunction in the body of rules does not extend the expressive power of the language, as such a rule can simply be rewritten as multiple rules without disjunction [Lloyd 1987].

unique name assumption). That is, the person referenced via the identifier `John_Smith` is different from the person referenced via the identifier `Bill_Jones`.

The Semantic Web is a very hostile environment for the unique name assumption. There are many and varied sources of information in the Semantic Web, even in the same area, and these sources are free to coin their own identifiers (IRIs) for anything they choose. For example, there are many providers of FOAF information,⁵ each of which may choose to use different identifiers to refer to the same individuals. One such information source may use one identifier to refer to a particular person, as in

```
mbox(http://ex1.org/John\_Smith, "mailto:John.Smith@ex.com")6
```

while another may use a different identifier for the same person, as in

```
mbox(http://ex2.org/Jack\_Smith, "mailto:John.Smith@ex.com")
```

Assuming that these two identifiers *necessarily* reference different individuals precludes the possibility that they may simply be two different “names” for the same individual.

In this sort of situation it is possible, and useful, to provide descriptions of situations in which two identifiers can be recognised as referencing the same individual. This is done in FOAF, where the `mbox`⁷ property is defined as an inverse functional property, which means that if two names are linked via `mbox` to the same mailbox, then these two names *must* reference the same individual. (This is a slightly suspect modelling decision, as mailboxes are not always unique, so not all mailboxes can be used as FOAF `mboxes`.) The above two information sources would be inconsistent in FOAF if the unique name assumption were in force, but are not inconsistent without it; without the unique name assumption it would simply follow that http://ex1.org/John_Smith and http://ex2.org/Jack_Smith are two names for the same person.

Further, FOAF information sources need not provide any identifier for the people their information describes, instead only providing other uniquely identifying information (such as the name of their `mbox`), as in

```
mbox(j, "mailto:John.Smith@ex.com")
```

The use of such anonymous individuals does not fit well with the unique name assumption. Either each such individual is assumed to be different, which would clearly not be the intention if they have the same identifying information, or it must be possible for multiple anonymous references to identify a single individual, in which case the Datalog simplifying assumptions and resulting complexity benefits may no longer hold.

This is not to say that it is not convenient to have a unique name assumption in many situations. For example, it is the case in many settings that a single information source will indeed use different identifiers to identify distinct individuals. Here, the unique

⁵ See <http://www.foaf-project.org> more for information about FOAF.

⁶ This is the only place that we will use fully-written out IRI references for identifiers as they are *very* long and interrupt the page layout. We will also generally eschew the use of qualified names, instead assuming that the unqualified short names we use are shorthand for an appropriate fully-written out IRI reference.

⁷ Of course, we mean here the mailbox property defined in the FOAF ontology.

name assumption is a useful shortcut for a potentially large number of statements such as

```
John_Smith ≠ Bill_Smith
John_Smith ≠ Susan_Jones
Bill_Smith ≠ Susan_Jones
...
```

that are required in the Classical paradigm in order to capture situations where different names necessarily refer to distinct individuals.

4.2 Open World

The Semantic Web is a very open environment. In the Semantic Web there is no requirement that information sources be comprehensive in any way, or even that a collection of information sources be comprehensive. In such a setting it is often incorrect to assume that lack of information is equivalent to negative information, as in assuming that a person mentioned at some FOAF site knows no other person simply because there are no `knows` relationships for that person listed at the site. Further, even if the person has his or her own FOAF page that lists some of the people that they know, it is not necessarily the case that *all* of the people that they know are listed there. For example, if

```
knows(John_Smith,Bill_Jones)
knows(John_Smith,William_Jones)
Bill_Jones ≠ William_Jones
```

are the only `knows` relationships found in (even) the web page of `John_Smith`, it is not appropriate to infer that `John_Smith` knows only two other people.

This is not to say that it is not useful to be able to state or even infer such comprehensive information. In most cases, however, this closure of information should be explicitly stated as an *addition* to the positive information. It is quite possible to do this in the Classical paradigm, for example by adding

```
John_Smith ∈ ≤2 knows.
```

Situations where this kind of closure information is valid are, however, mainly limited to database-like applications, such as an employee database, where it is reasonable to assume that all employees of the company are listed in its database.

4.3 Incomplete Information

In an open environment such as the Semantic Web, it may also be important to allow other kinds of incomplete information.

For example, it is useful to be able to state something about the people that `John_Smith` knows without providing complete information about them, or even saying who they are. We might like to say that `John_Smith` knows (at least) two other people without having to identify them. This is easy in the Classical paradigm, using a statement such as

$\text{John.Smith} \in \geq 2 \text{ knows,}$

but it is not possible in the strict Datalog paradigm, where it is required that relationships be between particular named individuals.

One might imagine that it would be possible to express this sort of information in a Datalog setting by introducing two new individuals and stating that they are known by `John.Smith`. Due to the unique name assumption, however, these two individuals would be *in addition* to any other individuals that are known to `John.Smith`, and could lead to incorrect inferences about the *total* number of people that he knows; they would also lead to an incorrect answer to a query asking for all of the people that `John.Smith` knows.

A similar problem exists with respect to required information for classes. If we want every person to have a birthplace, then the Classical paradigm does not require us to be able to identify the birthplace of any given person. We can simply state, for example, that every `Person` has exactly one birthplace, which must be a `Location`, as in

$\text{Person} \sqsubseteq = 1 \text{ birthplace}$
 $\text{Person} \sqsubseteq \forall \text{ birthplace. Location.}$

It is then perfectly acceptable for there to be instances of `Person` whose birthplace is not known, for example

`Person(John.Smith).`

This does not work in the Datalog paradigm, where the closed world assumption causes the requirement that every `Person` has a birthplace to become the requirement that every `Person` has a *known* birthplace.

4.4 Restrictions vs Constraints

Value Restrictions vs Value Constraints As the Semantic Web is an open environment, information can come from a variety of different sources. For example, there may be a base ontology about familial relationships that provides the class `Person` and property `parent`, further requiring that all parents of instances of `Person` also be instances of `Person`, as in

$\text{Person} \sqsubseteq \forall \text{ parent. Person.}$ (1)

A second information source may extend this ontology by adding the class `Adult`, and further requiring that all parents of people are instances of the `Adult` class, as in

$\text{Person} \sqsubseteq \forall \text{ parent. Adult.}$ (2)

In the Classical paradigm, information sources written for the first ontology, such as

`Person(John.Smith)` (3)
`parent(John.Smith, Fred.Smith),`

can be used in the second ontology, because their use in the second ontology will simply result in additional inferences based on the additional information (often called restrictions) provided by the second ontology. In the above case, for example, an additional inference would be

```
Adult(Fred.Smith).
```

Some of this sort of treatment of restrictions can also be provided in the Datalog paradigm, but it is common to instead use a different modelling methodology based on constraints. As we saw in Section 3.2, a constraint in the Datalog paradigm is a rule with an empty head, whose meaning is that we do not admit models satisfying the conditions expressed in the body of the rule.

A typical use of this methodology would be to model the situation described by the ontology axiom (1) above using a constraint

```
← parent(x,y) ∧ ¬Person(y),
```

which states that it is “illegal” for any individual to be a `parent` without also being a `Person`. Note that, because of the closed world assumption, any individual that is not provably a `Person` is taken *not* to be a person, so the information given in (3) above would not lead to the inference that `Fred.Smith` must be a `Person`, but would instead be treated as “illegal” (because it violates the constraint). Any valid information source would, therefore, have to include (explicitly or implicitly) the information that `Fred.Smith` is a `Person`, e.g.,

```
Person(John.Smith)
Person(Fred.Smith)
parent(John.Smith,Fred.Smith). (4)
```

This is not too onerous a burden if the information source is written with the first ontology/constraint in mind, and even, as de Bruijn *et al.* [2005] claim, has some modelling benefits having to do with lack of surprises.⁸

However, this modelling methodology breaks down in the presence of extended ontologies such as the one that includes the axiom (2) above, which would be rendered in the constraint modelling methodology as

```
← parent(x,y) ∧ ¬Adult(y). (5)
```

⁸ de Bruijn *et al.* [2005] provide the following example (slightly modified) of a potentially surprising inference:

```
FlightSeat ⊆ ∀hasPassenger.Passenger
FlightSeat(seat3)
FlightSeat(seat2)
hasPassenger(seat2, seat3)
```

There is no contraction here; instead it is inferred that `seat3` is a member of `Passenger`, which might not be what is wanted. The solution, of course, is to explicitly state the disjointness of `Passenger` and `FlightSeat`, in which case the above forms a contradiction, as was probably desired.

Even the extended information source (4) above would not be valid in this extended ontology, because it does not include the required information that `Fred.Smith` is an `Adult`.

The constraints modelling methodology thus seems to be in conflict with the open nature of the Semantic Web, which encourages the sharing, reuse and extension of information.

Such constraints also introduce a limited form of non-monotonicity. As we have seen above, for example, the augmented information source (4) above is inconsistent with respect to the extended ontology including the constraint (5); if

```
Adult(Fred.Smith)
```

is added, however, there is no longer any inconsistency.

Cardinality Restrictions vs Cardinality Constraints In the Classical paradigm, cardinality restrictions are an important way of inferring that two identifiers refer to (i.e., are different names for) the same individual. This sort of inference is particularly important in the Semantic Web, where different sources may use different identifiers for the same individual.

As mentioned above, FOAF mailboxes form a unique identifier for members of `Person` (i.e., `mbox` is an inverse functional property in FOAF). So, if one information source includes

```
mbox(Bill.Jones, "mailto:Bill.Jones@ex.com")
```

and another includes

```
mbox(William.Jones, "mailto:Bill.Jones@ex.com"),
```

then it can be inferred from the two sources that `Bill.Jones` and `William.Jones` identify the same individual.

This would not be possible in a strict constraint setting, as the two different identifiers would necessarily identify different individuals, leading to a contradiction.

Of course, this power does have its dangers. Consider the example from de Bruijn *et al.* [2005] (slightly modified)

```
FlightSeat  $\sqsubseteq \leq 1$  hasPassenger  
FlightSeat(seat1)  
hasPassenger(seat1, mary)  
hasPassenger(seat1, john).
```

This is *not* a contradiction, as `mary` and `john` could be the same, and due to the cardinality restriction it is inferred that they are, indeed, the same.

This may not be the intent of the modeler. However, if it is not the intent, then there is an easy solution—simply state that these two individuals are different, as in

```
mary  $\neq$  john.
```

With the addition of this information, the above example does produce the desired contradiction.

Existential Restrictions vs Existential Constraints Other common situations cause even more problems when rendered as constraints. For example, consider trying to model how a `mother` property should work, i.e., that every element of `Person` has exactly one `mother`, who is also a `Person`. In the Classical paradigm this is quite easy, e.g., using the axioms

$$\begin{aligned} \text{Person} \sqsubseteq = 1 \text{mother} \\ \text{Person} \sqsubseteq \forall \text{mother. Person.} \end{aligned}$$

This ensures that `mother` has the desired characteristics, and allows for situations where the `mother` of a `Person` is known, as in

$$\text{mother}(\text{John_Smith}, \text{Mary_Smith}),$$

as well as situations where little or nothing is known about the `mother` of a given `Person`, as in

$$\text{Person}(\text{John_Smith}).$$

In the Datalog paradigm, this situation is extremely problematic. Rendering it in restriction-like rules results in unsafe rules such as

$$\text{mother}(x, y) \leftarrow \text{Person}(x),$$

which are not allowed in the Datalog paradigm. Rendering it as constraints, as in

$$\begin{aligned} \leftarrow \text{Person}(x) \wedge \neg \text{mother}(x, y) \\ \leftarrow \text{Person}(x) \wedge \text{mother}(x, y) \wedge \neg \text{Person}(y) \end{aligned}$$

looks better initially, or at least fits within the Datalog paradigm.

It is not possible, however, to use this reasonable-looking ontology. Consider any instance of `Person` in this information source. This individual has to have a *known* `mother`, who has to be a `Person` as well, and so on. This means that the information source either has to include an infinite chain of `mother` links and an infinite number of `Persons`, which is obviously not possible, or there has to be a loop in the `mother` links (i.e., some person whose mother is one of their own descendants), which clearly conflicts with the desired meaning.⁹

4.5 Datatypes

It is, of course, vital to have a treatment of what are generally called datatypes in a Semantic Web modelling language. This has been provided in RDF and OWL by utilizing certain datatypes from XML Schema, including strings and integers.

In standard logical accounts of meaning, datatypes are treated in the same way as classes, and datatype values are treated in the same way as individual identifiers (i.e., a data value is treated as referring to an object).¹⁰ Much more is known, however, about

⁹ The Classical model may not rule out such cyclical `mother` links, but it does not force *all* interpretations to include such cycles.

¹⁰ RDF and OWL DL *do* restrict where these values can appear in their syntax, but this does not affect their meaning.

the possible interpretation of datatypes, data values and relationships between them. For example, it is known that the integer values “1” and “2” cannot refer to the same object, and that the objects that these two values refer to must be in a “<” relationship. It is possible, however, that two different data values are interpreted as the same object, e.g., the decimal values “1” and “1.0”.

Restrictions and constraints work in the standard way with datatypes and values. As much more is known about possible interpretations of datatype domains, however, it is much more likely that non-trivial inferences will cause a conflict (i.e., be incompatible with any possible interpretation).

4.6 The Role of Tools

Finally, it should be noted that there is indeed a preference amongst users for the Datalog paradigm in some areas. It is often easier to start modelling many domains using the Datalog paradigm—provided that the domain can be modelled in the Datalog paradigm *at all*—as the Classical paradigm requires more specification (e.g., stating that certain names denote distinct individuals, providing local closed world information, etc.).

Ontology-building tools with good user interfaces can help here. For example, it is not difficult to generate the statements required to explicitly state the unique name assumption for a group of identifiers. In a good tool it would not even be necessary to pick out the relevant identifiers: it would be sufficient to state that, in the information source being constructed, the identifiers referring to instances of a particular class all identify distinct individuals. The tool can then generate the required inequality axioms.

Good user interfaces can also help in determining the appropriate information to add to provide closure information, such as determining that

$$\text{John_Smith} \in \leq 2 \text{ knows}$$

would provide local closure for the people that `John.Smith` knows. They can also be used to point out potential “surprises” resulting from the inferences performed in the Classical paradigm, such as the ones mentioned by de Bruijn *et al.* [2005], thus suggesting improvements to the deficient ontology or other information source.

5 Reasoning in the Semantic Web

5.1 Complexity

At first glance, the Datalog paradigm seems to offer significant advantages with respect to the complexity of reasoning. Reasoning in the Classical paradigm *is* difficult for any reasonably expressive ontology language. For example, reasoning in the OWL DL Web Ontology Language is NEXPTIME-complete [Horrocks and Patel-Schneider 2004].

The polynomial-time reasoning result for Datalog looks very attractive when compared to this difficult reasoning in the Classical paradigm. However, the caveats associated with the result make it much less attractive than might first be imagined. One of these caveats is that the result only applies to standard query answering, i.e., queries about individuals, against a fixed program—as soon as changes to the program are allowed, then complexity becomes exponential (in fact EXPTIME-complete).

The polynomial-time complexity result does not, therefore, apply to many interesting schema level queries, as reductions to standard query answering (e.g., for subclass queries) require changing the program in the general case.

The polynomial-time reasoning results also do not hold for the more expressive versions of Datalog such as $\text{Datalog}^{\neg\vee}$. Unrestricted use of negation as failure, for example pushes even query answering beyond polynomial time, and full reasoning in this version of Datalog is NEXPTIME-complete [Eiter *et al.* 1997]. This is the same complexity as reasoning in OWL DL, and, as mentioned earlier, reasoning in the Description Logic *SHIQ* can be reduced to reasoning in Datalog^{\vee} [Hustadt *et al.* 2004].

5.2 Inferencing

As discussed in Section 4, one of the attractive characteristics of the Classical paradigm is the inferencing that it supports. For example, the large Semantic Web namespace means that different information sources may use different names for the same individuals, and thus inferring equality is a useful inference. The unique name assumption of the Datalog paradigm prevents this kind of inference, whereas the Classical paradigm can easily support it.¹¹ Moreover, information to the effect that different identifiers reference different individuals does not affect complexity in the Classical paradigm; reasoning gets no harder computationally as more identifiers are known to reference different individuals and often becomes easier in practice.

Even the basic formulations of class-level inference (such as identifying instance and subclass relationships) are suspect in the Datalog paradigm. What does it mean to ask whether a class defined using constraints is a subclass of another class, or if an individual is an instance of such a class? It is not correct to assume that constraints have no consequences. For example,

$$\begin{aligned} &\leftarrow \text{Person}(x) \wedge \text{child}(x, y) \wedge \neg \text{Person}(y) \\ &\text{Person}(\text{John_Smith}) \end{aligned}$$

implies both

$$\text{Person} \sqsubseteq \forall \text{child. Person}$$

and

$$\text{John_Smith} \in \forall \text{child. Person},$$

but accounts that use the Datalog paradigm to build ontology formalisms, such as OWL Flight [de Bruijn *et al.* 2005], do not provide coherent accounts of how to determine such relationships.

¹¹ Certain systems, such as RACER [Haarslev and Möller 2000], do have a unique name assumption built in, but the unique name assumption is certainly not a required part of the Classical paradigm.

6 Conclusion

We see two very different ways of modelling the world. At one extreme there is the Classical paradigm, where unstated information is left open. At the other extreme there is the Datalog paradigm, where unstated information is assumed to be false. The Datalog paradigm has some allure, particularly as it is closer to the common database view, but we argue that this closed view is not very compatible with the Semantic Web.

We argue that the Classical paradigm is better for modelling in the Semantic Web. We do admit that this paradigm has some pitfalls for those used to database modelling, but we believe that most of these can be easily handled with the help of good ontology building tools, which can be used to generate much of the extra information needed for the Classical paradigm. We also admit that the openness of the Classical paradigm can result in additional computational requirements, at least in some areas.

There is, of course, the possibility of augmenting the Classical paradigm with constructs that provide a Datalog-like flavour for *portions* of the Semantic Web. For example, there is no conceptual problem in providing constructs that state that certain information sources abide by the unique name assumption or are complete in some way. There have, indeed, been proposals to add such constructs to Description Logics [Baader *et al.* 2003]. The unrestricted use of these constructs does *increase* the difficulty of reasoning, but if they are limited to database-like sources, then no extra computational load is generated.

Good 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.

References

- [Agrawal and Gehani 1989] R. Agrawal and N. H. Gehani. Ode (object database and environment): The language and the data model. In *Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data*, pages 36–45. Association for Computing Machinery, June 1989.
- [Baader *et al.* 2003] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, implementation, and applications*. Cambridge University Press, 2003.
- [Berners-Lee *et al.* 2001] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic Web. *Scientific American*, 284(5):34–43, 2001.
- [Berners-Lee 1998] Tim Berners-Lee. Semantic web road map, September 1998. Available at <http://www.w3.org/DesignIssues/Semantic.html>.
- [Dan Brinkley and R. V. Guha 2004] RDF vocabulary description language 1.0: RDF schema. W3C Recommendation, <http://www.w3.org/TR/rdf-schema>, 2004.
- [de Bruijn *et al.* 2004] Jos de Bruijn, Axel Polleres, Rubén Lara, and Dieter Fensel. OWL Flight. Working Draft D20.3 v0.1, WSMML, 23 August 2004. Available at <http://www.wsmo.org/2004/d20/d20.3/v0.1/>.
- [de Bruijn *et al.* 2005] Jos de Bruijn, Axel Polleres, Rubén Lara, and Dieter Fensel. OWL DL vs. OWL Flight: Conceptual modeling and reasoning for the semantic web. In *Proceedings of the Nth World Wide Web Conference*, Japan, May 2005.

- [Dean *et al.* 2004] Mike Dean, Guus Schreiber, Sean Bechhofer, Frank van Harmelen, Jim Hendler, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein. OWL web ontology language: Reference. W3C Recommendation, <http://www.w3.org/TR/owl-ref/>, 2004.
- [Eiter *et al.* 1997] Thomas Eiter, Georg Gottlob, and Heikki Mannilla. Disjunctive datalog. *ACM Transactions on Database Systems*, 22(3):364–418, 1997.
- [Gelder *et al.* 1991] Allen Van Gelder, Kenneth Ross, and John S. Schlipf. The well-founded semantics for general logic programs. *Journal of the ACM*, 38(3):620–650, 1991.
- [Gelfond and Lifschitz 1988] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In Robert A. Kowalski and Kenneth Bowen, editors, *Proceedings of the Fifth International Conference on Logic Programming*, pages 1070–1080, Cambridge, Massachusetts, 1988. The MIT Press.
- [Gelfond and Lifschitz 1991] Michael Gelfond and Vladimir Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9(3/4), 1991.
- [Haarslev and Möller 2000] Volker Haarslev and Ralf Möller. Expressive ABox reasoning with number restrictions, role hierarchies, and transitively closed roles. In Anthony G. Cohn, Fausto Giunchiglia, and Bart Selman, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Seventh International Conference (KR2000)*, pages 273–284. Morgan Kaufmann Publishers, San Francisco, California, April 2000.
- [Hayes 2004] Patrick Hayes. RDF semantics. W3C Recommendation, <http://www.w3.org/TR/rdf-mt/>, 2004.
- [Horrocks and Patel-Schneider 2004] Ian Horrocks and Peter Patel-Schneider. Reducing OWL entailment to description logic satisfiability. *Journal of Web Semantics*, 1(4):345–357, 2004.
- [Hustadt *et al.* 2004] Ullrich Hustadt, Boris Motik, and Ulrike Sattler. Reducing SHIQ-description logic to disjunctive datalog programs. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Ninth International Conference (KR2004)*, pages 152–162, June 2004.
- [Lloyd 1987] John W. Lloyd. *Foundations of Logic Programming (Second, Extended Edition)*. Springer-Verlag, Berlin, Heidelberg, 1987.
- [Manola and Miller 2004] Frank Manola and Eric Miller. RDF primer. W3C Recommendation, <http://www.w3.org/TR/rdf-primer/>, 2004.
- [Ora Lassila and Ralph R. Swick 1999] Resource description framework (RDF): Model and syntax specification. W3C Recommendation, 22 February 1999, <http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/>, February 1999.
- [Patel-Schneider *et al.* 2004] Peter F. Patel-Schneider, Patrick Hayes, and Ian Horrocks. OWL web ontology language: Semantics and abstract syntax. W3C Recommendation, <http://www.w3.org/TR/owl-semantics/>, 2004.
- [Ullman 1988] Jeffrey D. Ullman. *Principles of Database and Knowledge-Base Systems*. Computer Science Press, 1988.