Analyzing Schema.org

Peter F. Patel-Schneider

Nuance Communications

pfpschneider@gmail.com

24 October 2014

Motivation

- This paper started when I was trying to determine whether Nuance should use schema.org information.
- I couldn't fully find out how to
  - create schema.org information,
  - consume schema.org information, or
  - change schema.org (the way I think it should be changed).
- Both technical and non-technical issues contributed to the difficulties.
- I asked around, and failing to get answers decided to perform an analysis of schema.org and then, like any academic, write a paper.
- My hope is that this analysis will make schema.org better, and easier to use.

What is Schema.org?

- An object-centered ontology language
- An extension mechanism—not analyzed here
- A single general-purpose ontology written in the ontology language language, available at *http://schema.org/*
  - This ontology must be used when creating schema.org information.

Schema.org Language (Class and Property Portion)

- URLs as identifiers
- Types—a.k.a. classes (*s:Movie*), datatypes (*s:Integer*), enumerations (*s:PhysicalExam*), and properties (*s:director*)
- Subclass relationship between classes
  s:Movie is a subclass of *s:CreativeWork*
- Maybe subproperty relationship between properties*
  s:collection is a subclass of *s:object*
- Disjunctive expected subject and value types for properties
  - *s:director* has expected value type of *s:Person* and expected subject type of *s:Episode*, *s:Movie*, and several others

NUANCE

Schema.org Language (Object Portion)

- Multi-typed items—a.k.a. objects
  http://www.avatarmovie.com/index.html of type *s:Movie*
- An item can have multiple identifiers,
  w:Avatar_(2009_film) and *http://www.avatarmovie.com/index.html*
- An item does not need to have an identifier
- Property-value pairs, values either other items or data values,
  http://www.avatarmovie.com/index.html has “Avatar” for *s:name*
  and *w:James_Cameron* for *s:director*
- A string can be provided as a value where an item is expected,
  ”James Cameron” as the value for *s:director* of
  http://www.avatarmovie.com/index.html

Non-technical Issues Limiting Utility of Schema.org

- **Documentation is limited.**
- Change may happen at any time.
- Continued ability to use schema.org is not guaranteed.
- **Bad name choices abound.**
  - *s:Abdomen* is a medical physical examination of the abdomen
- Extension mechanism status

No solutions are proposed here for these issues.

NUANCE

Technical Issues Limiting Utility of Schema.org

- Can content pages affect types, properties, etc.?
- Can foundational types and properties be changed (e.g., *s:Thing*)?
- Are types, properties, items, and data values disjoint?
- Are enumeration values all distinct items?
- **Do different URLs identify different items?**
- Must all types and properties be from *http://schema.org/*?
- **The datatypes are unspecified, both lexical and value.**
- Floats and integers are commingled.
- Are domains and ranges specified with the property or on types?
- Are subproperties allowed? (YES)
- **Are domains and ranges axioms or constraints?**
- **How does “strings as things” work?**

How to Solve the Technical Issues

Provide a firm basis for schema.org,
in both formal and non-formal forms

- To resolve the open issues, defining how schema.org works
- To provide guidance for producers of schema.org content
- To provide guidance for consumers of schema.org content
- To provide guidance when changing schema.org ontology

What follows is a non-formal description and formal definition of schema.org, *as I think it should be*

Non-formal Description—Follow Your Nose

- The occurence of a URL identifying an item indicates that the document available at that URL might have further information relevant to the item.
  - Note the qualification here.
- Information from such documents doesn't have to be taken into account.
  - However, it is often a good idea to do so.

Non-formal Description—Types

- **Each type has exactly one identifier, from *http://schema.org/* namespace, and is defined on its page there.**
- There is a multi-parent taxonomy of types.
  - Two roots, *s:Thing* and *s:Datatype* (should be *s:Literal*).
  - Subtypes of *s:Thing* and *s:Datatype* are disjoint.
- Types with ancestor *s:Datatype* are datatypes, and provide a lexical space and a lexical-to-value mapping.
- **Types with ancestor *s:Enumeration* are enumerations, and provide the (distinct) items that directly belong to the type.**
- Other types are regular types, and can have instances, which are items, and provide properties for their instances.
- *s:Thing* is a type with properties *s:description* and *s:name*.
- *s:Text* is a datatype with lexical and value space being Unicode strings.

Non-formal Description—Properties

- Each property has exactly one identifier, from *http://schema.org/* namespace, and is defined on its page there.
- Properties are disjoint from types.
- There is a multi-parent taxonomy of properties.
- Each property has one or more expected value types (ranges).
- **For each range each parent property must have a range that is an ancestor of that range**, e.g., if *s:episode* is a property with *s:Episode* as a range and *tvEpisode* is a subproperty of *s:episode*, then it is legal for *tvEpisode* to have *s:TVEpisode* as a range.
- *s:description* and *s:name* are properties with range *s:Text*

Non-formal Description—Data values

- Data values belong to one or more datatypes, e.g, 7 belongs to both *s:Integer* and *s:Number*.
- Data values are disjoint from types and properties.

Non-formal Description—Items

- Items have zero or more URLs identifying them,
  i.e., a URL identifies at most one item.
- **Different URLs need not identify different items.**
- **Items are disjoint from types, properties, and data values**,
  e.g., *http://www.avatarmovie.com/index.html* cannot be a type, a property, or an integer.
- Items belong to one or more non-datatype types.
- All items belong to *s:Thing*.
- If an item belongs to a type it also belongs to that type's parents.

Non-formal Description—Property-value pairs

- Property-value pairs provide values for properties of items, e.g., *http://www.imdb.com/name/nm0000116/* as a value of *s:director* for *http://www.avatarmovie.com/index.html*
- A property value is also a value for the parents of the property, e.g., if *ex:BBTEpisode7* is a value for *tvEpisode* of *ex:BBT* then it is also a value for *s:episode* of *ex:BBT*
- For each property value on an item,
  - **the item belongs to a type that has the property as one of its properties**, e.g., because *s:director* is a property of *s:Episode*, *s:Movie*, and several other types *http://www.avatarmovie.com/index.html* belongs to *s:Episode*, *s:Movie*, or one of the other types,
  - **the value belongs to one of the ranges of the property**, e.g., because *s:director* has range *s:Person*, *http://www.imdb.com/name/nm0000116/* belongs to *s:Person*.

Non-formal Description—Strings as things

- Bare text can be used as if it was the value for any property
- If the property has range *s:Text* or *s:Datatype*, the value is the text.
- Otherwise, treat the value as from one of the datatype ranges of the property if reasonable, so a property with range of *s:Integer* would treat text values that look like integers as integers.
- Otherwise, **use an item with the text as a** *s:description* **and belonging to some non-datatype range of the property** if possible, so *"James Cameron"* as a value for *s:director* would result in an item of type *s:Person*. (The property *s:description* is used instead of *s:name*, as the text might not truly be a name for the item.)
- Otherwise the actual value for the property is the text itself.

Non-formal Description—Surface syntaxes

- Allow all possible data values (as long as they are not too big).
- **Allow items with any number of types, including none**, and use *s:additionalType* to provide types.
- **Allow items with any number of identifying URLs, including none**, and use *s:url* and *s:sameAs* to provide identifying URLs.
- Allow property values for any property and any value on an item.
- **Allow multiple values for a property on an item.**
- Forbid unintended use of built-in RDF and OWL classes and properties.

Formal Definition of Schema.org

- What matters is only the abstract syntax, not the surface form.
- The abstract syntax builds up a knowledge base, incorporating the information at schema.org and information from various web pages, often including many of those web pages that are used as item identifiers.
- A knowledge base has unique definitions of types (regular, enumerations, and datatypes) and properties plus information about items.
- **The meaning of a schema.org knowledge base is given by a model-theoretic semantics**:
  - Standard model-theoretic semantics (even more standard than RDF)
  - Just implements the description above

For more information see the paper.

Formal Definition of Schema.org—Definitions

- A regular type definition provides an identifier (from *http://schema.org* namespace), parents, and properties, e.g., $\langle$ *s:Movie*, {*s:CreativeWork*}, {*s:actor*, *s:director*, *s:producer*, *s:duration*, *s:musicBy*, *s:productionCompany*, *s:trailers:author*, *s:copyrightYear*} $\rangle$
- An enumeration definition provides provides an identifier (from *http://schema.org* namespace), parents, properties, and items, e.g., $\langle$ *s:BookFormatType*, {*s:Enumeration*}, {}, {*s:EBook*, *s:Hardcover*, *s:Paperback*} $\rangle$
- A datatype definition provides an identifier (from *http://schema.org* namespace), parents, lexical space, and lexical-to-value mapping, e.g., $\langle$*s:URL*, {*s:Text*}, *U*, $id^U\rangle$
- A property definition provides an identifier (from *http://schema.org* namespace), parents, and ranges, e.g., $\langle$*s:collection*, {*s:object*}, {*s:Thing*}$\rangle$

Formal Definition of Schema.org—Items

- Item information provides identifying URLs, types, and property-value pairs (a property and an item or typed value or text), e.g.,

$$
\begin{aligned}
&\langle \{\textit{http://www.avatarmovie.com/index.html}, \textit{w:Avatar\_(2009\_film)}\},\\
&\ \{\textit{s:Movie}\},\\
&\ \{\langle \textit{s:name}, \textit{"Avatar"}\rangle ,\\
&\quad \langle \textit{s:director}, \langle\ \{\}, \{\textit{s:Person}\}, \{\langle \textit{s:name}, \textit{"James Cameron"}\rangle\}\ \rangle\ \rangle ,\\
&\quad \langle \textit{s:actor}, \textit{"Sam Worthington"}\rangle ,\\
&\quad \langle \textit{s:actor}, \textit{w:Sigourney\_Weaver}\rangle\}\\
&\quad \langle \textit{s:year}, \textit{"2009"}\rangle ,\\
&\quad \langle \textit{s:author}, \textit{"James Cameron"}\rangle\}\ \rangle
\end{aligned}
$$

Formal Definition of Schema.org—Model Theory

- A standard model-theoretic semantics
  (even more standard than RDF)
- Devfines domain, mappings, axioms, interpretations, models, ...
- Provides formal basis for the non-formal description above.

See the paper.

NUANCE®

What Has Been Done

- Introduce abstract syntax and standard model-theoretic semantics for schema.org.
- Provide a formal foundation for schema.org ontology language, including
  - type, property, and data type definitions,
  - items, data values, and property-value pairs, and
  - “strings as things”.
- Define the meaning of the ontology at *http://schema.org*.
- Define the meaning of schema.org content.

Aspects of the Account

- Multiple URLs for items gives an equality mechanism for schema.org
- schema.org can represent disjunctive information
  - via unnamed elements of enumerations
  - via undertyped subjects (values) of properties with multiple domains (ranges)
- schema.org is translatable into OWL
  - but needs entire ontology to construct domains and ranges
  - not quite OWL 2 DL
- Reasoning in schema.org is probably not easy
  - decidable - very close to OWL 2 DL
  - in PSpace - no new types, properties, or items needed
  - probably PSpace hard

Useful Extensions to the schema.org Ontology Language

- Local ranges,
  e.g., an episode of a TV series is a TV episode
  - For better modelling of specialized types
- Disjointness between types
  e.g., people are disjoint from actions
  - For better modelling
  - For better detection of errors in content

Conclusion

- Motivation: desire to consume (and produce) schema.org information.
- Result: non-formal description and formal definition of what schema.org should be.
  - Provides a resource for use by both producers and consumers of schema.org content.
- Non-technical issues also need to be addressed.
- Schema.org continues to change—recent changes and proposals need similar analysis,
  - e.g., roles, Sports, *http://schema.org/FictionalThing*, ...
- Schema.org content is not (just) linked data, nor should it be.