

The Semantic Web

Craig Knoblock

(Thanks to Yolanda Gil for some of the slides!)

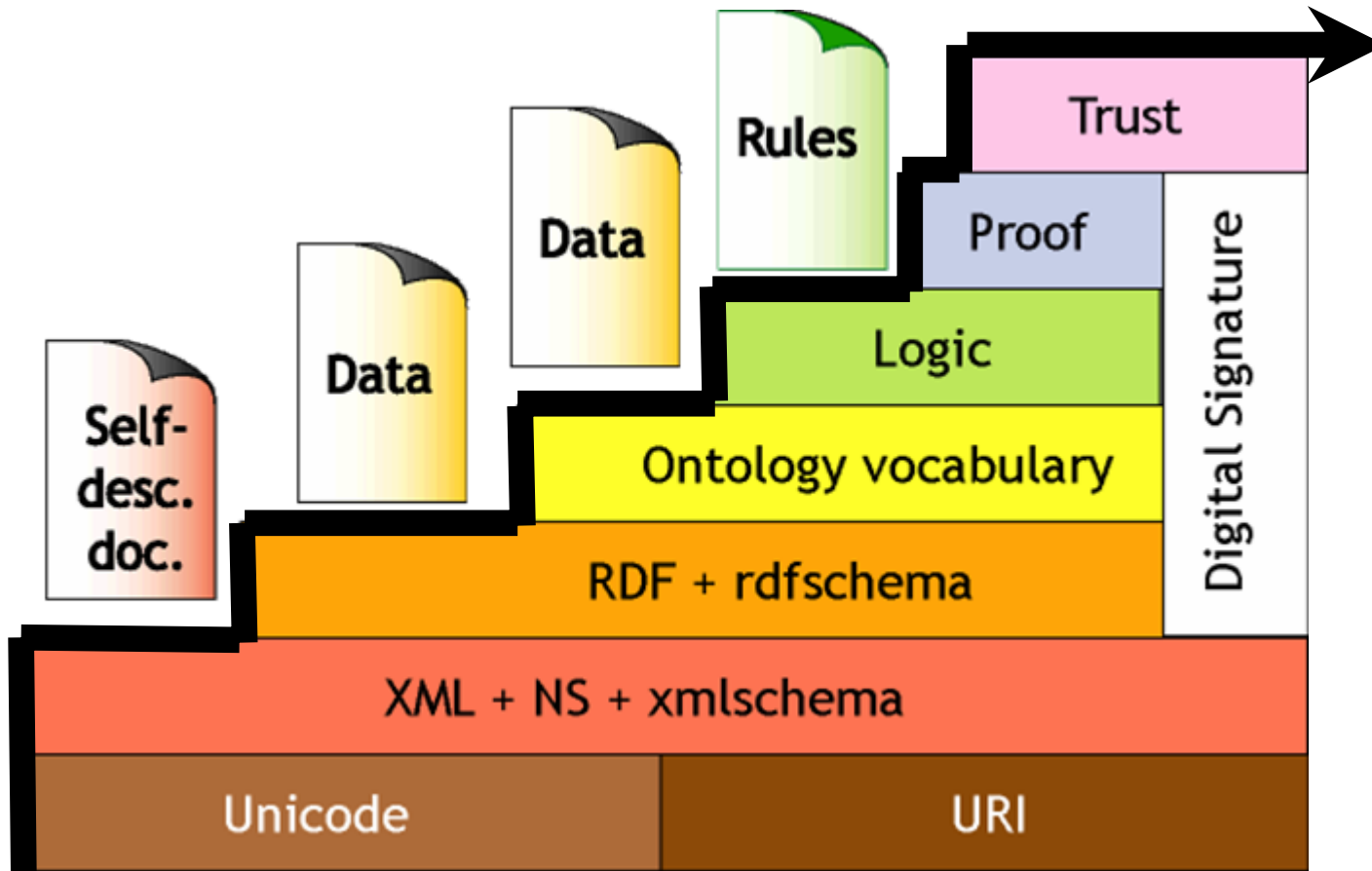
The Semantic Web

W3C's Tim Berners-Lee: "Weaving the Web":

"I have a dream for the Web... and it has two parts."

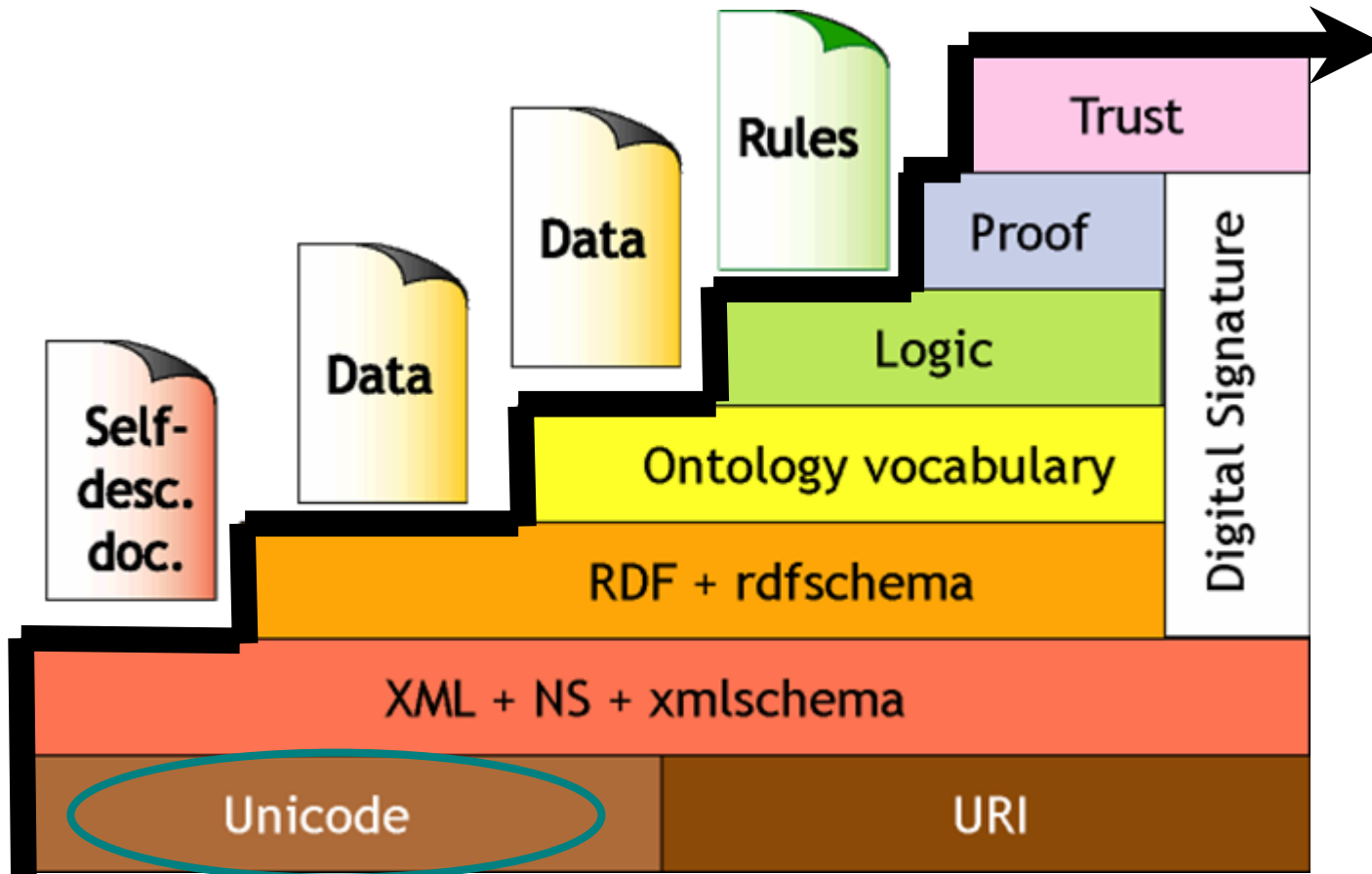
- *The first Web enables **communication between people***
 - *The Web shows how computers and networks enable the information space while getting out of the way*
- *The new Web will **bring computers into the action***
 - *Step 1 -- Describe: putting data on the Web in machine-understandable form -- a Semantic Web*
 - *RDF (based on XML)*
 - *Master list of terms used in a document (RDF schema)*
 - *Each document mixes global standards and local agreed-upon terms (namespaces)*
 - *Step 2 -- Infer and reason: apply logic inference*
 - *Operate on partial understanding*
 - *Answering why*
 - *Heuristics*

Web Semantics



Semantic Web LayerCake (Berners-Lee, 99; Swartz-Hendler, 2001)

Unicode

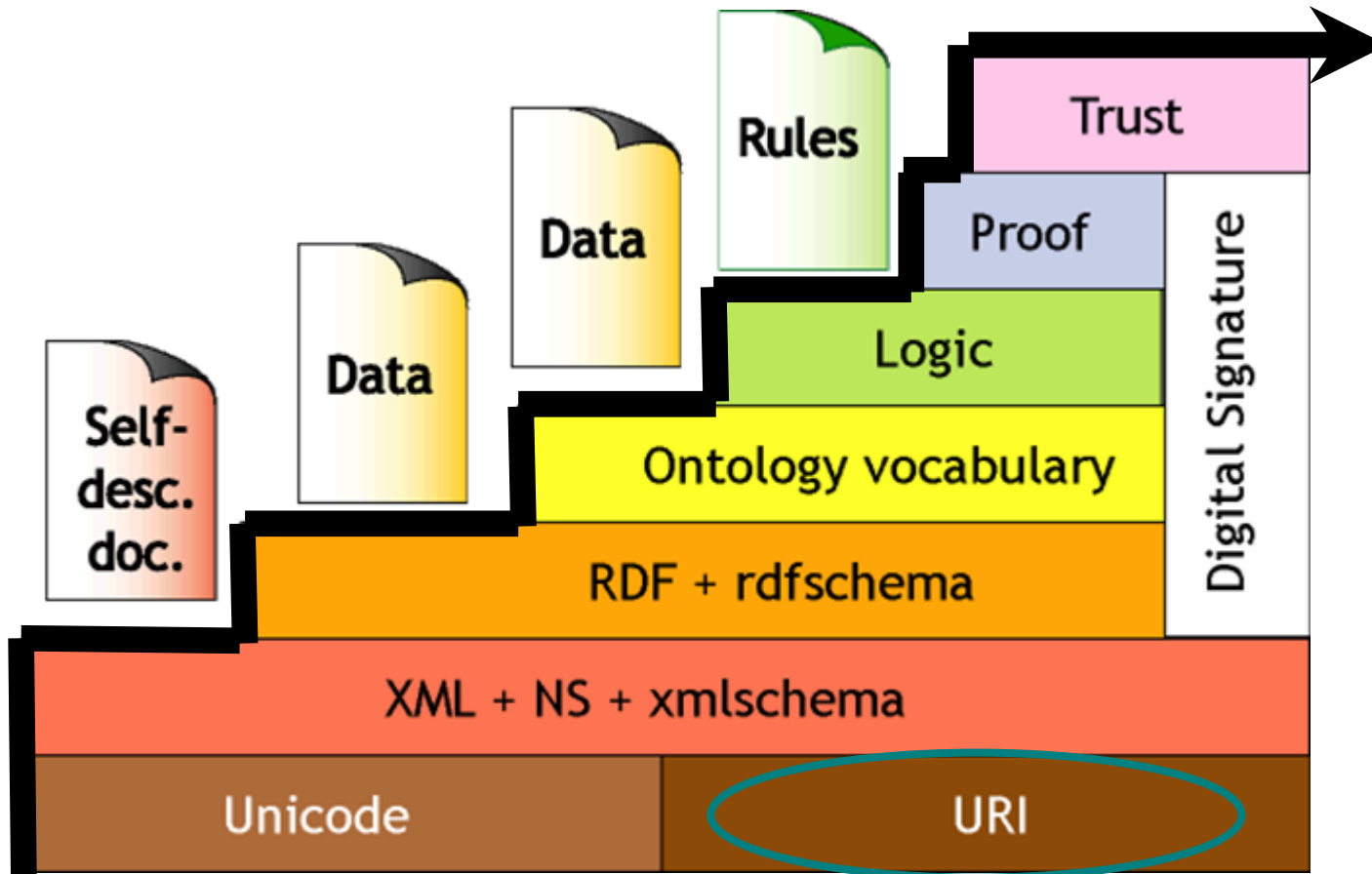


Semantic Web LayerCake (Berners-Lee, 99; Swartz-Hendler, 2001)

Unicode

- A character encoding system, like ASCII, designed to help developers who want to create software applications that work in any language in the world
- Unicode provides a unique number for every character, no matter what the platform, no matter what the program, no matter what the language

URI

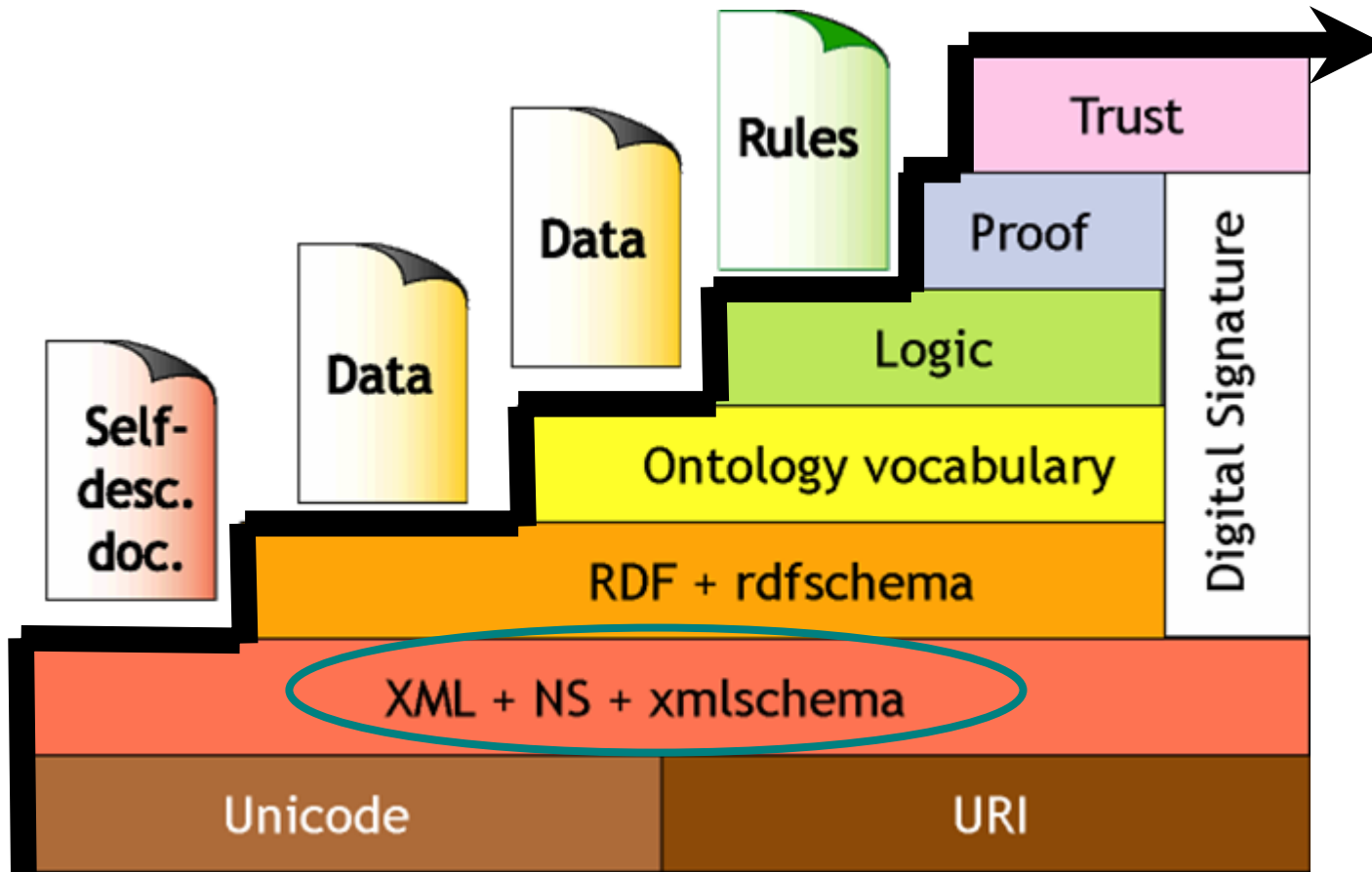


Semantic Web LayerCake (Berners-Lee, 99; Swartz-Hendler, 2001)

URIs: Uniform Resource Identifiers (aka URLs)

- The Web is an information space. URIs are the points in that space.
- Short strings that identify **resources** in the web: documents, images, downloadable files, services, electronic mailboxes, and other resources.
- They make resources addressable in the same simple way. They reduce the tedium of "log in to this server, then issue this magic command ..." down to a single click.

XML and Namespaces



Semantic Web LayerCake (Berners-Lee, 99; Swartz-Hendler, 2001)

Why XML (eXtensible Markup Language)

Problems with HTML

HTML design

- HTML is intended for presentation of information as Web pages.
- HTML contains a fixed set of markup tags.

This design is not appropriate for data:

- Tags don't convey meaning of the data inside the tags.
- Tags are not extensible.

The Design of XML

- Tags can be used to represent the meaning of data/information
 - separates syntax (structural representation) from semantics => **only syntax is considered in XML**
- There is no fixed set of markup tags - **new tags can be defined**
- Underlying data model is a **tree structure**
- “XML is the new ASCII” -- Tim Bray

<http://www.w3.org/TR/2000/REC-xml-20001006>

Simple XML Example

```
<Bookstore>
  <Book ID="101">
    <Author>John Doe</Author>
    <Title>Introduction to XML</Title>
    <Date>12 June 2001</Date>
    <ISBN>121232323</ISBN>
    <Publisher>XYZ</Publisher>
  </Book>
  <Book ID="102">
    <Author>Foo Bar</Author>
    <Title>Introduction to XSL</Title>
    <Date>12 June 2001</Date>
    <ISBN>12323573</ISBN>
    <Publisher>ABC</Publisher>
  </Book>
</Bookstore>
```

Make up your own tags

Sub-elements

XML by itself is just hierarchically structured text

An important diversion: Namespaces

- What is a Namespace ?

The Namespace of an element, is the scope within which, it (and thus it's name) is valid

- Why do we need Namespaces ?

- If elements were defined within a global scope, it becomes a problem when combining elements from multiple documents
- Modularity: If a markup vocabulary exists which is well understood and for which there is useful software available, it is better to reuse it

- Namespaces in XML:

An XML namespace is a collection of names, identified by a URI reference.

Names from XML namespaces may appear as qualified names, which contain a single colon, separating the name into a **prefix** and a **local** part. The prefix, which is mapped to a URI reference, selects a namespace

XSD: XML Schema Definition

- Written in the same syntax as XML documents (unlike XML DTDs!)
- Elements and attributes
- Enhanced set of primitive datatypes.
 - Wide range of primitive data types, supporting those found in databases (string, boolean, decimal, integer, date, etc.)
 - Can create your own datatypes (complexType)
- Can derive new type definitions on the basis of old ones (refinement)
- Can have constraints on attributes
 - Examples: maxLength, precision, enumeration, maxInclusive (upper bound), minInclusive (lower bound), etc.

XSD (XML Schema) Example

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://www.books.org"
            xmlns="http://www.books.org">
  <xsd:element name="Bookstore">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Book" minOccurs="1" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Book">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Title" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Author" minOccurs="1" maxOccurs="unbounded"/>
        <xsd:element ref="Date" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="ISBN" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Publisher" minOccurs="1" maxOccurs="1"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Title" type="xsd:string"/>
  <xsd:element name="Author" type="xsd:string"/>
  <xsd:element name="Date" type="xsd:date"/>
  <xsd:element name="ISBN" type="xsd:integer"/>
  <xsd:element name="Publisher" type="xsd:string"/>
</xsd:schema>
```

Prefix “**xsd**” refers to the XMLSchema namespace

“**xmlns**” refers to the **default** namespace

Defining the element “Bookstore” as a complex Type

Containing a sequence of 1 or more “Book” elements

When referring to another Element, use “ref”

The Author can be 1 or more

Element definitions

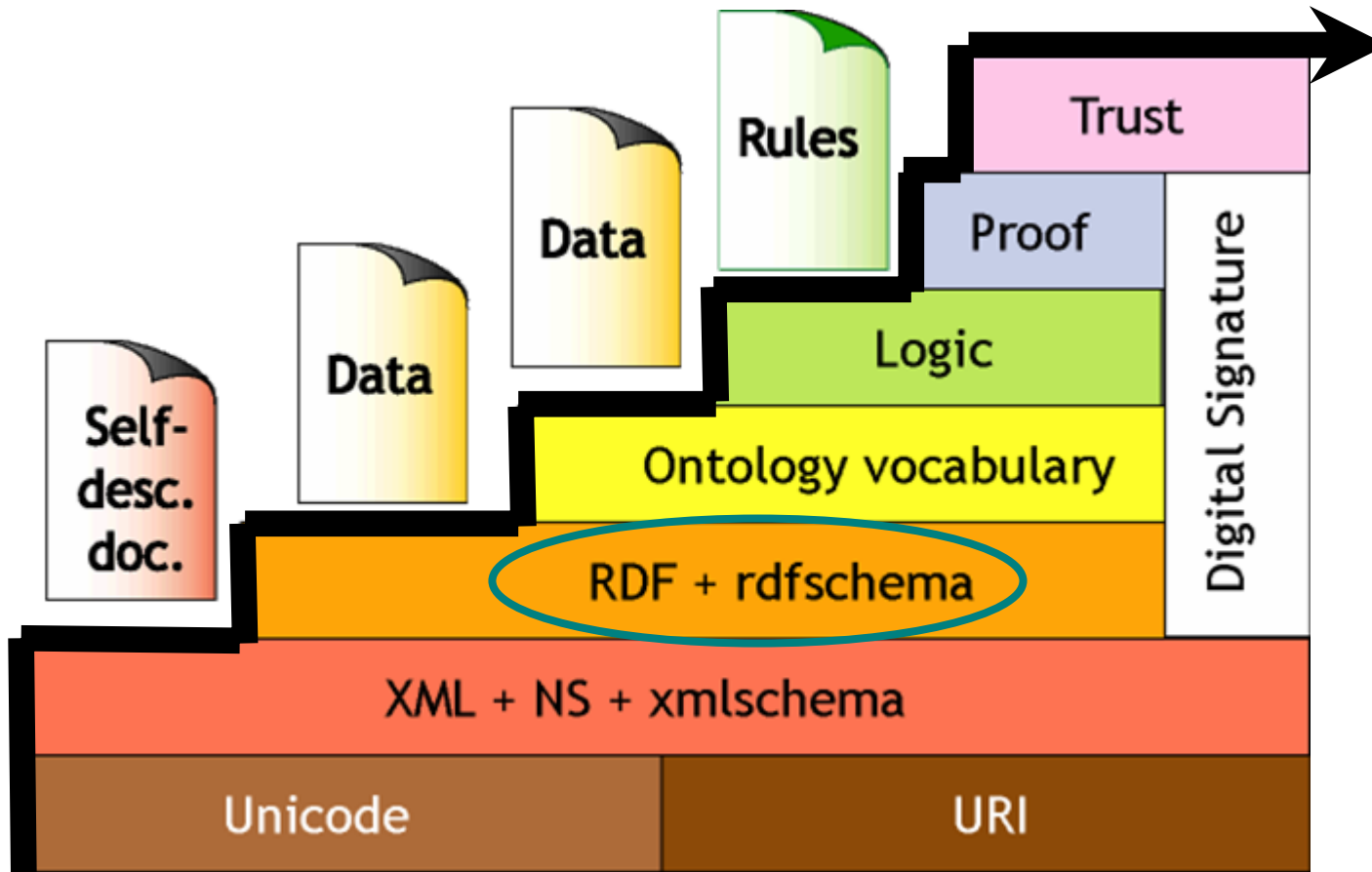
Notice the use of more meaningful data types

Summary of the XML+ NS +XSD Layer

The Power of Simplicity

- “When I designed HTML, I chose to avoid giving it more power than it absolutely needed – a “**principle of least power**”, which I have stuck to ever since. I could have used a language like Knuth’s Tex but...” -- TBL
- Keeps the principles of SGML in place but its spec is thin enough to wave 😊
- To say you are “Using XML” is sort of like saying you are using ASCII
- Using XSD (XML Schema) makes a lot more sense

Resource Description Framework



Semantic Web LayerCake (Berners-Lee, 99; Swartz-Hendler, 2001)

Where XML & XML Schemas Fail

- No semantics!

```
<book>
  <title> ... </title>
  <author> ... </author>
  <isbn> ... </isbn>
</book>
```

```
<bookstore>
  <book> ... </book>
  <mgzine> ... </mgzine>
</bookstore>
```

- Will XML scale in the metadata world?

1. The order in which elements appear in an XML document is often meaningful. This seems highly unnatural in the metadata world.

Furthermore, maintaining the correct order of millions of data items is impractical.

2. XML allows constructions that mix up some text along with child elements, which are hard to handle.

Ex.

```
<topelem>This is some character string data
  <elem>
    this is a child
    <subelem>this is another child</subelem>
  </elem>
</topelem>
```

RDF (Resource Description Framework)

- RDF provides a way of describing resources via metadata (data about data)
It restricts the description of resources to **triplets (subject, predicate, object)**
- It provides interoperability between applications that exchange machine understandable information on the Web.
- The original broad goal of RDF was to define a mechanism for describing resources that makes no assumptions about a particular application domain, nor defines (a priori) the semantics of any application domain.
- Uses XML as the interchange syntax.
- Provides a **lightweight** ontology system.

The formal specification of RDF is available at:

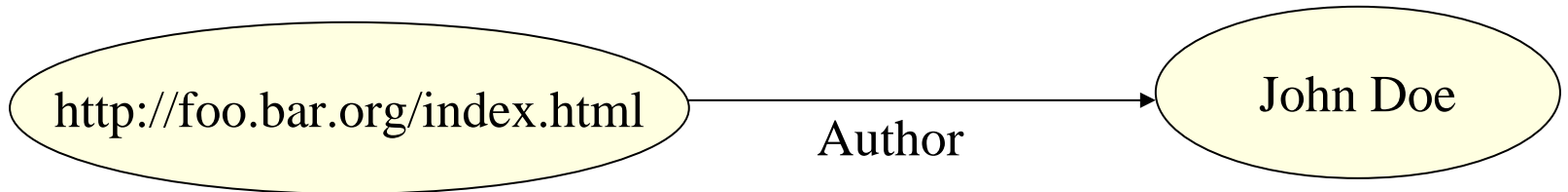
<http://www.w3.org/TR/REC-rdf-syntax/>

RDF Syntax

Subject, Predicate and Object Triplets (Tuples)

- Subject: The resource being described.
- Predicate: A property of the resource
- Object: The value of the property

A combination of them is said to be a Statement (or a rule)



A web page
being described

[Subject]

A property of the
web page (author)

[Predicate]

The value of the predicate
(here the author)

[Object]

RDF Example

```
<?xml version="1.0"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/TR/WD-rdf-syntax#"
  xmlns:s="http://description.org/schema/">
  <rdf:Description about="http://foo.bar.org/index.html">
    <s:Author>John Doe</s:Author>
  </rdf:Description>
</rdf:RDF>
```

Namespace for the RDF spec

Namespace 's', a custom namespace

Subject

Author (property of the subject)
(Also a resource)

Object. Can also point to a resource

The above statement says :

The Author of <http://foo.bar.org/index.html> is "John Doe"

In this way, we can have different objects (resources) pointing to other objects (resources) , thus forming a DLG (Directed Line Graph)

You can also make statements about statements - reification

Ex: 'xyz' says that ' The Author of <http://foo.bar.org/index.html> is John Doe'

RDF Schema

- A schema defines the terms that will be used in the RDF statements and gives specific meanings to them.

<http://www.w3.org/TR/rdf-schema/>

Example:

```
<rdf:RDF xml:lang="en"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
```

→ RDF Schema Namespace

```
<rdf:Description ID="MotorVehicle">
  <rdf:type resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
  <rdfs:subClassOf
    rdf:resource="http://www.w3.org/2000/01/rdf-schema#Resource"/>
</rdf:Description>
```

→ An "ID" attribute actually defines a new resource

→ "Resource" is the top level class

```
<rdf:Description ID="PassengerVehicle">
  <rdf:type resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
  <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
</rdf:Description>
```

→ PassengerVehicle is a **subclass** of MotorVehicle

```
<rdf:Description ID="Truck">
  <rdf:type resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
  <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
</rdf:Description>
```

Example (cont..)

```
<rdf:Description ID="Van">
  <rdf:type resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
  <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
</rdf:Description>
```

```
<rdf:Description ID="MiniVan">
  <rdf:type resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
  <rdfs:subClassOf rdf:resource="#Van"/>
  <rdfs:subClassOf rdf:resource="#PassengerVehicle"/>
</rdf:Description>
```

} → **Multiple Inheritance**

```
<rdf:Description ID="registeredTo">
  <rdf:type resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  <rdfs:domain rdf:resource="#MotorVehicle"/>
  <rdfs:range rdf:resource="#Person"/>
</rdf:Description>
```

→ **Domain** of a property

```
<rdf:Description ID="rearSeatLegRoom">
  <rdf:type resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  <rdfs:domain rdf:resource="#PassengerVehicle"/>
  <rdfs:domain rdf:resource="#Minivan"/>
  <rdfs:range rdf:resource="http://www.w3.org/2000/03/example/classes#Number"/>
</rdf:Description>
</rdf:RDF>
```

→ **Range** of a property

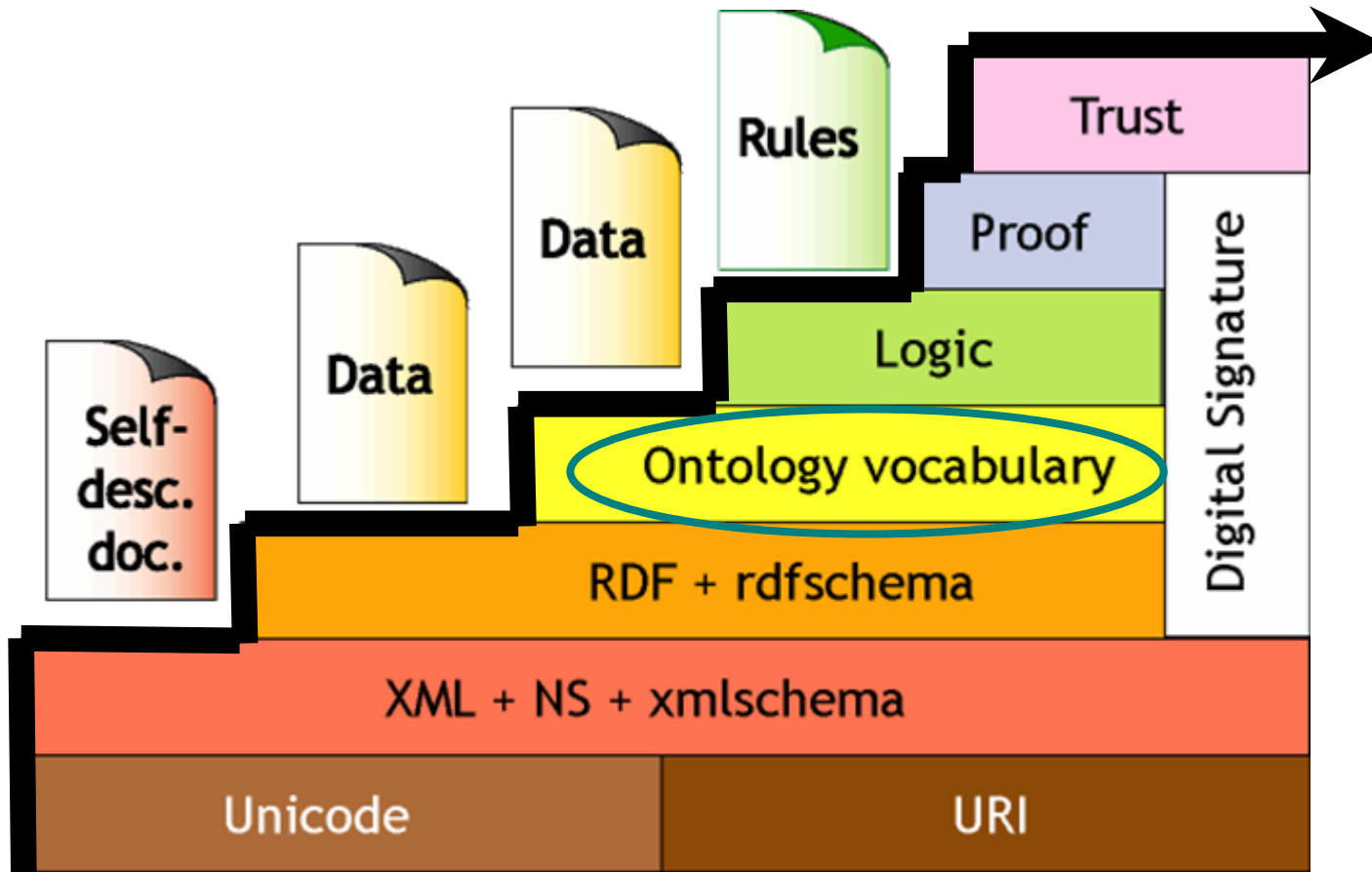
Summary: RDF & RDF Schema layer

- Minimalist model - (thing), Class, Property
- Subproperty, Subclass
- Domain & Range

- RDF Schema: a W3C recommendation
Feb'04

- Efficient storage and retrieval
 - “Triple store” using database backends

Ontology Vocabulary



Semantic Web LayerCake (Berners-Lee, 99; Swartz-Hendler, 2001)

Limitations of RDF

- Cannot define properties of properties (unique, transitive)
- No equivalence, disjointness, etc.
- No mechanism of specifying **necessary and sufficient conditions** for class membership.

Example:

If it is given that 'XYZ' has a 'car' which is '7ft high', has 'wide wheels' and 'loading space is 4 cub.m', then we should be able to reason that 'XYZ' has an 'SUV', as given by the necessary and sufficient conditions for being an 'SUV' :
height > 4ft & wide wheels & loading space > 2 cub.m

OWL: Web Ontology Language

- Takes DAML+OIL as input, follows standard W3C process
- OWL:
 - Extension of RDF schema
 - Ontology base
 - Description logic substrate
 - Language constructs similar to DAML+OIL
- OWL version 1.0 released Nov 2002
- W3C Recommendation (Feb 10, 2004)

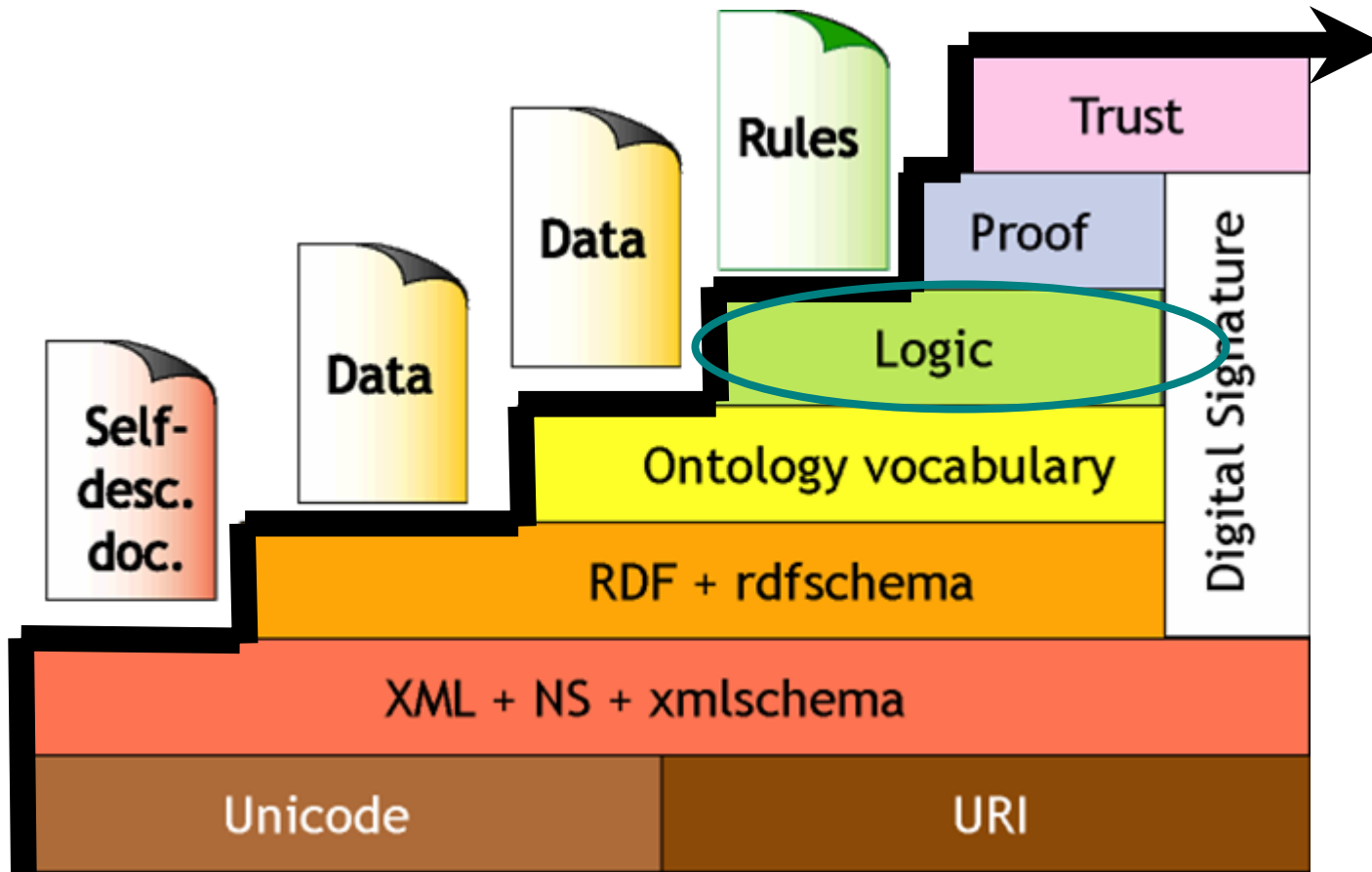
OWL features

- RDF schema (class, subclass, property, subproperty)
- Restrictions
 - Range, domain
 - Local, global
 - Existential
 - Cardinality
- Combinators
 - Union, intersection
 - Complement
 - Symmetry
 - Transitivity
- Mappings
 - Equivalent
 - inverse

OWL Lite

- A restricted version of OWL, designed to support efficient reasoning
- All of OWL except:
 - owl:oneOf
 - owl:unionOf
 - owl:complementOf
 - owl:hasValue
 - owl:disjointWith
 - owl:DataRange
 - ... and a few other restrictions
- Increasing expressivity is provided in OWL-DL and OWL-Full

Logic

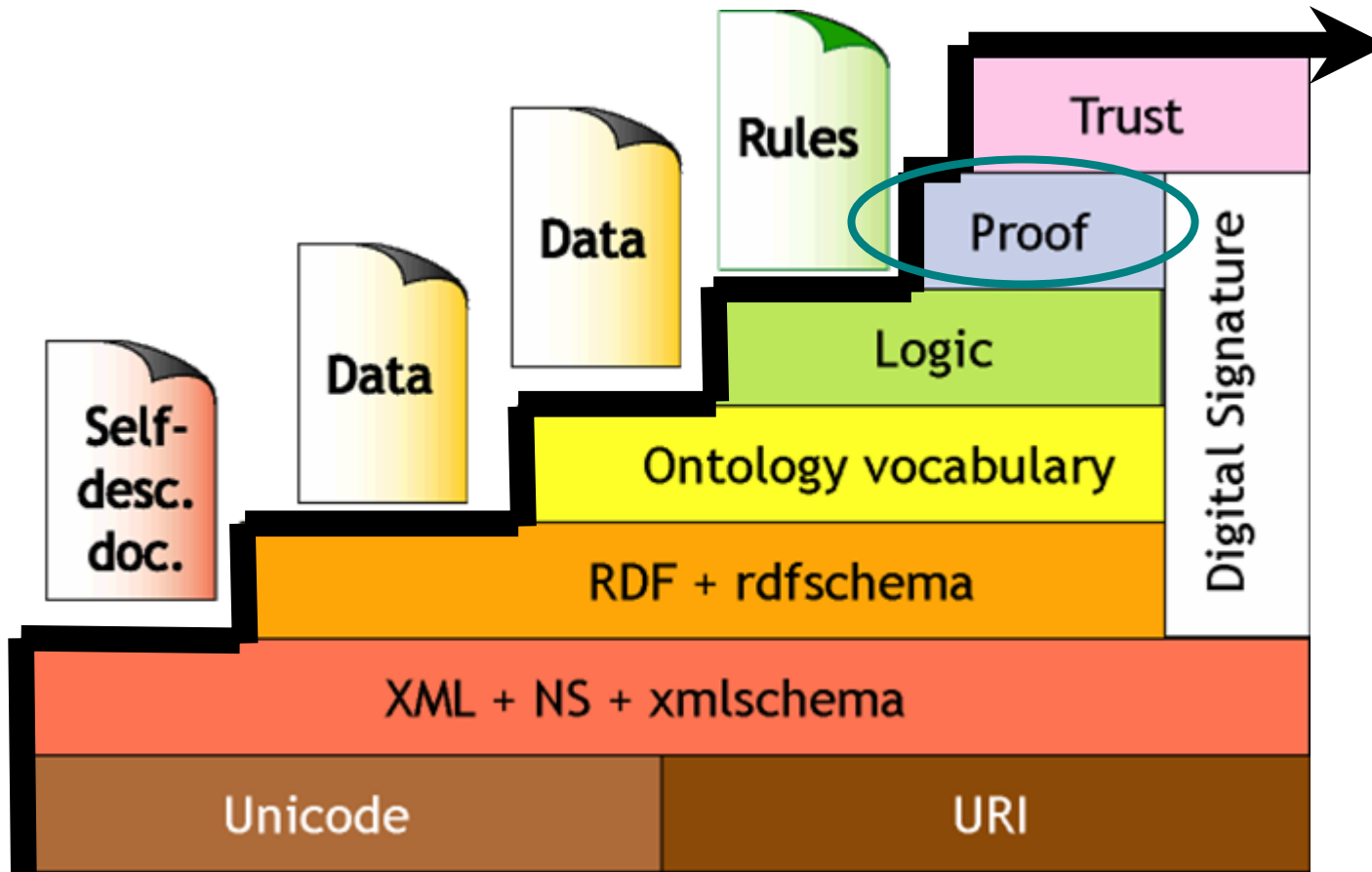


Semantic Web LayerCake (Berners-Lee, 99; Swartz-Hendler, 2001)

Description Logics

- Classes are defined in terms of other classes/relations
- Relations are first class citizens
- Powerful inference algorithms:
 - **Subsumption**: is classA a subclass of classB given their definitions?
 - **Recognition**: is instanceA of classA?
 - **Classification**: automatic reorganization of class hierarchy based on definitions of classes
- Provides a set of rules that can be used in proofs

Proof

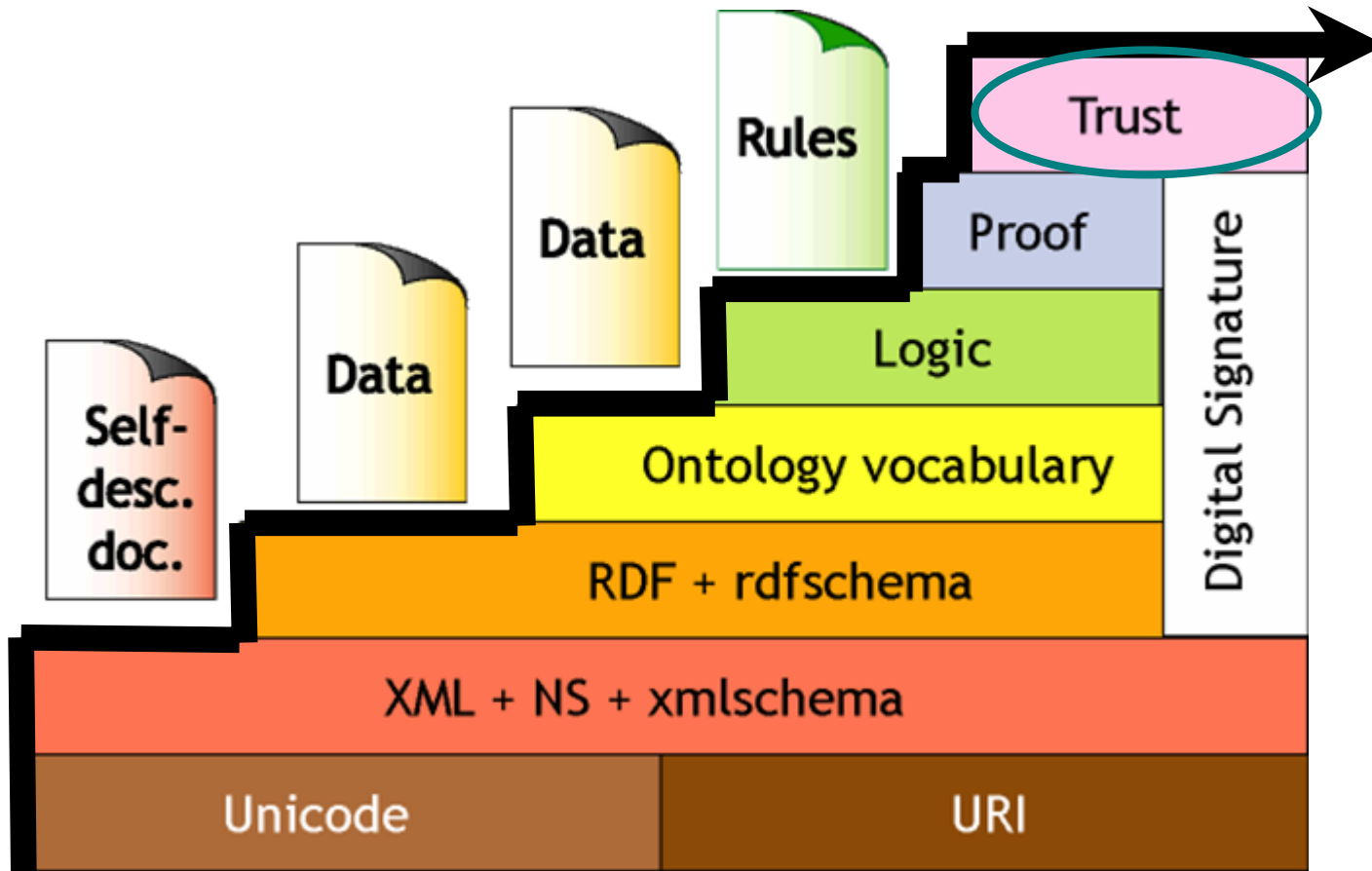


Semantic Web LayerCake (Berners-Lee, 99; Swartz-Hendler, 2001)

Proofs: Logical Derivations

- Use the logic to prove things given the set of facts provided
- The derivation of the proof provides the support for the derived facts
- Easier to verify a proof than it is to find one

Trust



Semantic Web LayerCake (Berners-Lee, 99; Swartz-Hendler, 2001)

Can We Trust the Result

- Need a mechanism to determine who to trust
- Exploit digital signatures to verify that information comes from a trusted source
- Define a “Web of Trust”
 - You tell the system who you want to trust

W3C's Semantic Web Principles

1. Everything identifiable is in the Semantic Web (URIs!)
2. Partial information
 - Anyone can say anything about anything
3. Web of trust
 - All statements on the Web occur in some context
4. Evolution
 - Allow combining independent work done by different communities
5. Minimalist design
 - Make the simple things simple, and the complex things possible
 - Standardize no more than is necessary

Hypertext: Then and Now

- SOTA circa 1990: Dynatext's electronic book
 - A book had to be compiled (like a program) in order to be displayed efficiently
 - A central link database, to make sure there were no broken links
 - Text that was fixed and consistent (a whole book)
- WWW:
 - Links can be added and used at any time
 - Distributed (must live with broken links!)
 - Decentralized

Knowledge Representation: Now and Tomorrow

“To webize KR in general is, in many ways, the same as to webize hypertext. Replace identifiers with URIs. Remove any requirement for global consistency. Put any significant effort into getting critical mass. Sit back.”

-- TBL