HERZUM SOFTWARE

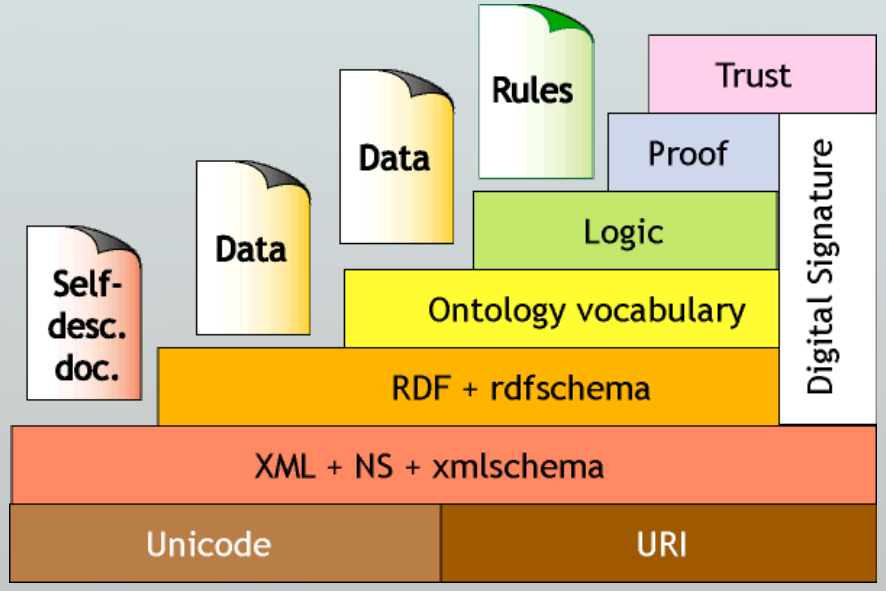
Resource Description Framework and Web Ontology Language

RDF / OWL

HERZUM SOFTWARE

What is RDF

- W3C standard - http://www.w3.org/RDF/
- A language for describing *resources*
- Able to describe any data structure of arbitrary complexity
- Heritage in Description Logic systems
- The basis of the Semantic Web



HERZUM SOFTWARE

RDF for Metadata

- **RDF *describes* resources**
  - **PersonA:** *first name = “Fred”*
    surname = “Flintstone”
    lives in = “Bedrock”
    friends with = PersonB
- **Resources can be *anything***
  - **documents**
  - **network services**
  - **computers**
  - **people**
  - **locations**
  - ***etc.***
- **Resources are identified by URI (URL or URN)**
- **RDF is defined with an “Open World” assumption**

HERZUM SOFTWARE

RDF for Data

- Original intent of RDF was to describe resources (metadata) which were accessible through other programs.
  - Properties of resources
  - Links between resources
- RDF is flexible enough to describe *any* data structure.
- More useful for describing free-form data. Often called *Schema-less.*
- Consistent records are more efficient in an RDBMS.

HERZUM SOFTWARE

Subject, Predicate, Object

- **RDF is formed by a set of statements, of the form:**
  - **(subject, predicate, object)**

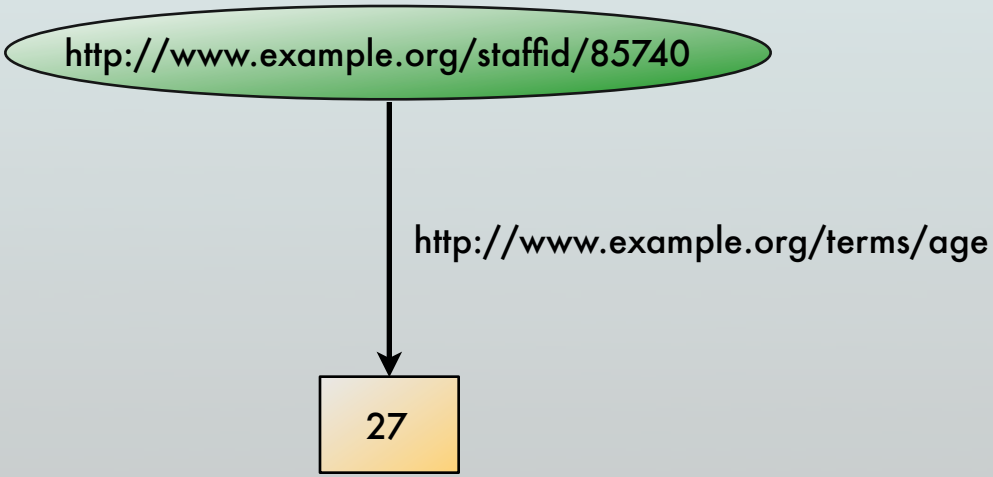
(people:FF, people:hasName, "*Fred Flintstone*")
(people:BR, people:hasName, "*Barney Rubble*")
(people:FF, people:hasFriend, people:BR)

- **Subjects may be a:**
  - **URI**
  - ***blank node* (a node without an identifier)**
- **Predicates can only be URIs**
- **Objects may be a:**
  - **URI**
  - ***blank node***
  - **Literal (String, number, date, ...)**

HERZUM SOFTWARE

Ball and Stick Diagrams

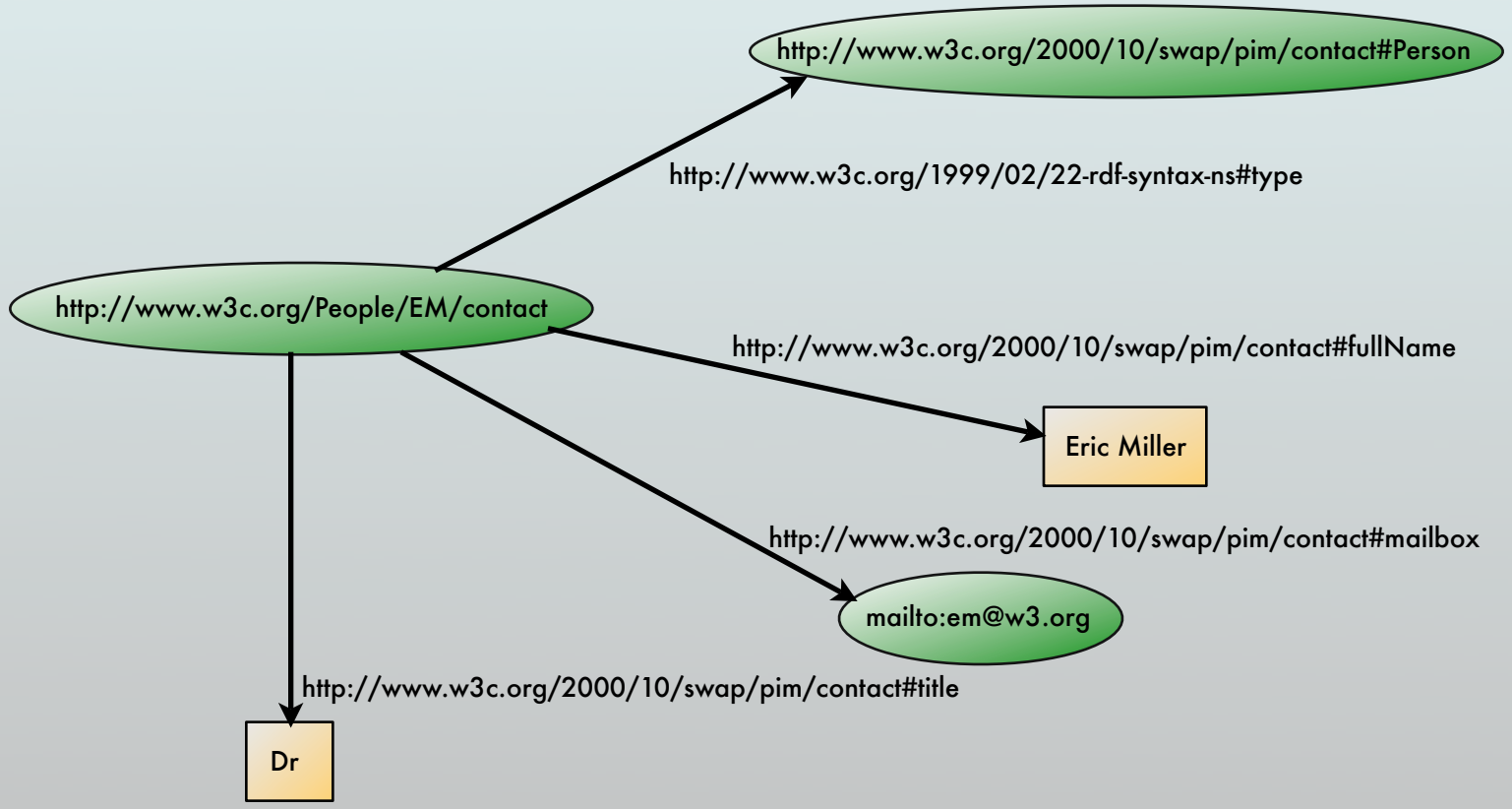
- Treats *subjects* and *objects* as nodes in a graph, with *predicates* as the connecting edges
- Useful to conceptualize simple structures



HERZUM SOFTWARE

Ball and Stick Diagrams

- Treats *subjects* and *objects* as nodes in a graph, with *predicates* as the connecting edges
- Useful to conceptualize simple structures



HERZUM SOFTWARE

RDF/XML

- Most common encoding
- Often mistaken as RDF
- Efficient, but difficult to learn

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
             xmlns:contact="http://www.w3.org/2000/10/swap/pim/contact#">

  <contact:Person rdf:about="http://www.w3.org/People/EM/contact#me">
    <contact:fullName>Eric Miller</contact:fullName>
    <contact:mailbox rdf:resource="mailto:em@w3.org"/>
    <contact:personalTitle>Dr.</contact:personalTitle>
  </contact:Person>

</rdf:RDF>
```

HERZUM SOFTWARE

Other Syntax

- **N3**

```
<#pat> <#child>    <#al> .
<#pat> <#child>    <#chaz> .
<#pat> <#child>    <#mo> .
<#pat> <#age>      24 .
<#pat> <#eyecolor> "blue" .
```

- **N-Triples**
  - **fixed subset of N3**
- **Turtle (Terse RDF Triple Language)**
  - **Extension of N-Triples**

```
<http://www.w3.org/TR/rdf-syntax-grammar> <http://purl.org/dc/elements/1.1/title> "RDF/
XML Syntax Specification (Revised)" .

<http://www.w3.org/TR/rdf-syntax-grammar> <http://example.org/stuff/1.0/editor> _:123 .

_:123 <http://example.org/stuff/1.0/fullname> "Dave Beckett" .

_:123 <http://example.org/stuff/1.0/homePage> <http://purl.org/net/dajobe/> .
```

HERZUM SOFTWARE

Other Syntax

- **N3**

```
<#pat> <#child>  <#al>, <#chaz>, <#mo> ;
       <#age>    24 ;
       <#eyecolor> "blue" .
```

- **N-Triples**
  - **fixed subset of N3**
- **Turtle (Terse RDF Triple Language)**
  - **Extension of N-Triples**

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix ex: <http://example.org/stuff/1.0/> .

<http://www.w3.org/TR/rdf-syntax-grammar>
  dc:title "RDF/XML Syntax Specification (Revised)" ;
  ex:editor [
    ex:fullname "Dave Beckett";
    ex:homePage <http://purl.org/net/dajobe/>
  ] .
```

HERZUM SOFTWARE

RDF Schema

- **Provides structure to RDF documents**
- **Defines types with** rdf:type**:**
  - rdfs:Class rdfs:subClassOf rdfs:Property rdfs:subPropertyOf

```
<contact:em> <rdf:type> <contact:Person> .
<contact:Person> <rdf:type> <rdfs:Class> .
```

- **Defines domains and ranges:**

```
<contact:surname> <rdfs:domain> <contact:Person> .
<contact:surname> <rdfs:range> <xsd:string> .
<contact:knows> <rdfs:domain> <contact:Person> .
<contact:knows> <rdfs:range> <contact:Person> .
```

- **Defines Containers and Collections:**

```
<rdf:Description rdf:about="http://example.org/courses/6.001">
   <s:students>
      <rdf:Bag>
         <rdf:li rdf:resource="http://example.org/students/Amy"/>
         <rdf:li rdf:resource="http://example.org/students/Mohamed"/>
         <rdf:li rdf:resource="http://example.org/students/Johann"/>
         <rdf:li rdf:resource="http://example.org/students/Maria"/>
         <rdf:li rdf:resource="http://example.org/students/Phuong"/>
      </rdf:Bag>
   </s:students>
</rdf:Description>
```

HERZUM SOFTWARE

OWL - Web Ontology Language

- **Various levels of functionality:**
  - **OWL Lite**
  - **OWL DL**
  - **OWL Full**
- **RDFS was released as “that part of OWL that everyone could agree to”**
- **Introduces class and object relations**
  - owl:sameAs owl:disjointWith owl:oneOf owl:intersectionOf owl:unionOf owl:complementOf
- **Introduces properties of properties**
  - owl:inverseOf owl:TransitiveProperty owl:FunctionalProperty owl:InverseFunctionalProperty owl:SymmetricProperty
- **Introduces property restriction classes**
  - owl:cardinality (min/max) owl:allValuesFrom owl:someValuesFrom owl:hasValue

HERZUM SOFTWARE

OWL Semantics

- OWL still uses RDF syntax
- Open World model
- Can provide inferred RDF statements

```
<contact:hasFriend> <rdf:type> <owl:SymmetricProperty> .
<contact:FF> <contact:hasFriend> <contact:BR> .
```

⇒

```
<contact:hasFriend> <rdf:type> <owl:ObjectProperty> .
<contact:BR> <contact:hasFriend> <contact:FF> .
```

- Can check for inconsistencies

```
<person:Parent> <rdf:type> <owl:Class> .
<person:Male>   <rdf:type> <owl:Class> .
<person:Female> <rdf:type> <owl:Class> .
<person:Male>   <owl:disjointWith> <person:Female> .

<person:Father> <owl:IntersectionOf> <_:blank1> .
<_:blank1> <rdf:type> <rdf:List> .
<_:blank1> <rdf:first> <person:Male> .
<_:blank1> <rdf:rest> <_:blank2> .
<_:blank2> <rdf:first> <person:Parent> .
<_:blank2> <rdf:rest> <rdf:nil>;

<person:Wilma> <rdf:type> <person:Female> .
<person:Wilma> <rdf:type> <owl:Father> .
```

HERZUM SOFTWARE

OWL Semantics

- **OWL still uses RDF syntax**
- **Open World model**
- **Can provide inferred RDF statements**

SymmetricProperty(hasFriend)
hasFriend(FF,BR)

⇒ ObjectProperty(hasFriend)
hasFriend(BR,FF)

- **Can check for inconsistencies**

Class(Parent)
Class(Male)
Class(Female)
disjointWith(Male,Female)

Father ⇐ IntersectionOf(Male,Parent)

Female(Wilma)
Father(Wilma)

HERZUM SOFTWARE

RDF and Logic

$C(x) \Rightarrow$ `<x> <rdf:type> <C>`

$P(x,y) \Rightarrow$ `<x> <P> <y>`

$P(x,y,z) \Rightarrow$

```
<x> <P> <_:blank1>
<_:blank1> <rdf:first> <y>
<_:blank1> <rdf:rest> <_:blank2>
<_:blank2> <rdf:first> <z>
<_:blank2> <rdf:rest> <rdf:nil>
```

or

```
<_:blank1> <rdf:type> <P>
<_:blank1> <rdf:first> <x>
<_:blank1> <rdf:rest> <_:blank2>
<_:blank2> <rdf:first> <y>
<_:blank2> <rdf:rest> <_:blank3>
<_:blank3> <rdf:first> <z>
<_:blank3> <rdf:rest> <rdf:nil>
```

HERZUM SOFTWARE

Querying

- Previous languages
  - RDQL (Jena, Sesame)
  - iTQL (Kowari, Mulgara)
- New Standard: SPARQL
  - similar to RDQL and iTQL
  - Subset of iTQL is SPARQL compatible
- All languages are SQL-like
  - SELECT clause
  - FROM clause
  - WHERE clause
- Principal difference is in WHERE constraints:
  - SPARQL: bind variable names to column position
  - SQL: existing variable bindings, select on variable values

HERZUM SOFTWARE

Query Example

```
select $x
from <myGraph>
where $x <rdf:type> <hs:Employee>
   and $x <hs:firstName> “Peter”
```

```
select $property $value
from <myGraph>
where $obj <rdf:type> <hs:MyClass>
  and $obj $key “keyvalue”
  and $key <rdf:type> <owl:InverseFunctionalProperty>
  and $obj $property $value
```

Logic via Queries

```
<contact:hasFriend> <rdf:type> <owl:SymmetricProperty> .
<contact:FF> <contact:hasFriend> <contact:BR> .
```

⇒ `<contact:BR> <contact:hasFriend> <contact:FF> .`

```
insert
 select $o2 $pred $o1
 from <myGraph>
 where $pred <rdf:type> <owl:SymmetricProperty>
   and $o1 $pred $o2
into <myGraph>
```

HERZUM SOFTWARE

Links

- RDF and OWL standards
  - http://www.w3.org/RDF/
  - http://www.w3.org/2004/OWL/
  - http://www.w3.org/TR/rdf-sparql-query/
- All linked from http://www.w3.org/

- Open Source Mulgara RDF data store
  - http://mulgara.org/