A General Framework for Representing and Reasoning with Annotated Semantic Web Data

Umberto Straccia
ISTI-CNR
Pisa, Italy

Nuno Lopes, Gergely Lukácsy, and Axel Polleres
Digital Enterprise Research Institute
National University of Ireland, Galway
RDFS is both a logic and standard W3C Semantic Web Language

- basic ingredient: triples \((\text{subject}, \text{predicate}, \text{object})\)

But triples alone are often not enough . . .

RDFS statements are true with respect to a certain domain

- Time
  - \((\text{umberto}, \text{workedFor}, \text{ISTI})\)
  - true since 2001

- Vagueness
  - \((\text{AAAI10Hotel}, \text{closeTo}, \text{OlimpicPark})\)
  - true to some degree

- Provenance
  - \((\text{umberto}, \text{knows}, \text{axel})\)
  - true in http://www.straccia.info/foaf.rdf
RDFS variants are emerging including some specific domains such as
time, fuzziness, provenance, . . .

Our contribution:
- A very general framework for annotating RDFS triples
- A deductive system, which straightforwardly extends the one for classical RDFS
  - Implementation is simple
- Crisp RDF is a special case
  - Backward compatibility is guaranteed
- Computational complexity and scalability: as for crisp RDFS
  - . . . if domain computations are not too expensive
Outline

- Annotated RDF
- Query answering
- Summary & Outlook
From RDFS to Annotated RDFS
RDFS Syntax

- Pairwise disjoint alphabets
  - \( U \) (RDF URI references)
  - \( B \) (Blank nodes)
  - \( L \) (Literals)

- For simplicity we will denote unions of these sets simply concatenating their names

- We call elements in **UBL terms** (denoted \( t \))
- We call elements in \( B \) **variables** (denoted \( x \))
RDF triple (or RDF atom):

\((s, p, o) \in UBL \times U \times UBL\)

- \(s\) is the subject
- \(p\) is the predicate
- \(o\) is the object

Example:

\((umberto, workedFor, IEI)\)
\( \rho \text{df} \) (restricted RDFS) [Munoz et al., 2007]

- \( \rho \text{df} \) (read rho-df, the \( \rho \) from restricted rdf)
- \( \rho \text{df} \) is defined as the following subset of the RDFS vocabulary:

\[
\rho \text{df} = \{ \text{sp}, \text{sc}, \text{type}, \text{dom}, \text{range} \}
\]

- \((p, \text{sp}, q)\)
  - property \( p \) is a sub property of property \( q \)
- \((c, \text{sc}, d)\)
  - class \( c \) is a sub class of class \( d \)
- \((a, \text{type}, b)\)
  - \( a \) is of type \( b \)
- \((p, \text{dom}, c)\)
  - domain of property \( p \) is \( c \)
- \((p, \text{range}, c)\)
  - range of property \( p \) is \( c \)
Graph (or Knowledge Base) is a set of triples $\mathcal{T}$

The universe of a graph $G$, denoted by $universe(G)$, is the set of elements in $UBL$ that occur in the triples of $G$.

The vocabulary of $G$, denoted by $voc(G)$ is the set $universe(G) \cap UL$.

A graph is ground if it has no blank nodes (i.e. variables).
Annotated RDFS: Syntax

- Statement (triples) may have attached a value $\lambda$ taken from an *Annotation Domain*

  \[(s, p, o): \lambda\]

- For instance,

  *(umberto, workedFor, IEI): [1992, 2001]*

  *(AAAI10Hotel, closeTo, OlimpicPark): 0.8*

  *(umberto, knows, axel): http://www.straccia.info/foaf.rdf*
Annotated RDFS: Semantics

- What do annotations mean for RDFS semantics?
- How do I combine, annotated triples semantically?

\[(umberto, \text{type}, \text{IEIEmployee}) : [1992, 2001]\]
\[(\text{IEIEmployee}, \text{sc}, \text{PisaCenterEmployee}) : [1968, 2000]\]
\[(umberto, \text{type}, \text{PisaCenterEmployee}) : [?, ?]\]
Annotation Domains: Informally

Illustration by Example: Time

- An Annotation Domain consists of
  - A lattice $L$ of annotation values
    - e.g. $[1968, 2000]$ and $\{[1968, 2000], [2003, 2004]\}$
  - An order between elements:
    - if $\lambda \preceq \lambda'$, then $\tau: \lambda$ is true to a lesser extent than $\tau': \lambda'$
    - e.g. $[1968, 2000] \preceq [1952, 2007]$ ($\preceq$ is $\subseteq$)
  - Top and bottom elements:
    - $\top = [-\infty, +\infty], \bot = \emptyset$
  - “Conjunction” function $\otimes$
    - $[1992, 2001] \otimes [1968, 2000] = [1992, 2000]$ ($\otimes$ is $\cap$)
  - “Combination” function $\lor$
Annotation Domains: Informally

Illustration by Example: Time

- An Annotation Domain consists of
  - A lattice $L$ of annotation values
    - e.g. $[1968, 2000]$ and $\{[1968, 2000], [2003, 2004]\}$
  - An order between elements:
    - if $\lambda \preceq \lambda'$, then $\tau : \lambda$ is true to a lesser extent than $\tau' : \lambda'$
    - e.g. $[1968, 2000] \preceq [1952, 2007]$ ($\preceq$ is $\subseteq$)
  - Top and bottom elements:
    - $\top = [-\infty, +\infty]$, $\bot = \emptyset$
  - “Conjunction” function $\otimes$
    - $[1992, 2001] \otimes [1968, 2000] = [1992, 2000]$ ($\otimes$ is $\cap$)
  - “Combination” function $\vee$
Annotation Domains: Informally

Illustration by Example: Time

▶ An Annotation Domain consists of
  ▶ A lattice $L$ of annotation values
    ▶ e.g. [1968, 2000] and \{[1968, 2000], [2003, 2004]\}
  ▶ An order between elements:
    ▶ if $\lambda \leq \lambda'$, then $\tau: \lambda$ is true to a lesser extent than $\tau': \lambda'$
    ▶ e.g. [1968, 2000] $\leq$ [1952, 2007] ($\leq$ is $\subseteq$)
  ▶ Top and bottom elements:
    ▶ $\top = [-\infty, +\infty]$, $\bot = \emptyset$
  ▶ “Conjunction” function $\otimes$
    ▶ [1992, 2001] $\otimes$ [1968, 2000] = [1992, 2000] ($\otimes$ is $\cap$)
  ▶ “Combination” function $\vee$
Annotation Domains: Informally

Illustration by Example: Time

▶ An Annotation Domain consists of
  ▶ A lattice $L$ of annotation values
    ▶ e.g. $[1968, 2000]$ and $\{[1968, 2000], [2003, 2004]\}$
  ▶ An order between elements:
    ▶ if $\lambda \preceq \lambda'$, then $\tau : \lambda$ is true to a lesser extent than $\tau' : \lambda'$
    ▶ e.g. $[1968, 2000] \preceq [1952, 2007]$ ($\preceq$ is $\subseteq$)
  ▶ Top and bottom elements:
    ▶ $\top = [−\infty, +\infty], \bot = \emptyset$
  ▶ “Conjunction” function $\otimes$
    ▶ $[1992, 2001] \otimes [1968, 2000] = [1992, 2000]$ ($\otimes$ is $\cap$)
  ▶ “Combination” function $\vee$
Annotation Domains: Informally

Illustration by Example: Time

▶ An Annotation Domain consists of
  ▶ A lattice $L$ of annotation values
    ▶ e.g. $[1968, 2000]$ and $\{[1968, 2000], [2003, 2004]\}$
  ▶ An order between elements:
    ▶ if $\lambda \preceq \lambda'$, then $\tau: \lambda$ is true to a lesser extent than $\tau': \lambda'$
    ▶ e.g. $[1968, 2000] \preceq [1952, 2007]$ ($\preceq$ is $\subseteq$)
  ▶ Top and bottom elements:
    ▶ $\top = [-\infty, +\infty]$, $\bot = \emptyset$
  ▶ “Conjunction” function $\otimes$
    ▶ $[1992, 2001] \otimes [1968, 2000] = [1992, 2000]$ ($\otimes$ is $\cap$)
  ▶ “Combination” function $\lor$
Annotation Domain (Formally)

- An annotation domain is an algebraic structure that is well-known for Many-Valued FOL
- **Annotation Domain**: is a *residuated bounded lattice*

\[
D = \langle L, \preceq, \land, \lor, \otimes, \Rightarrow, \bot, \top \rangle,
\]

*i.e.*

1. \(\langle L, \preceq, \land, \lor, \bot, \top \rangle\) is a bounded lattice, where \(\bot\) and \(\top\) are bottom and top elements, \(\land\) and \(\lor\) are the meet and join operators;
2. \(\langle L, \otimes, \top \rangle\) is a commutative monoid;
3. \(\Rightarrow\) is the so-called residuum implication of \(\otimes\), i.e. for all \(x, y, z\),

\[
z \preceq (x \Rightarrow y) \text{ iff } x \otimes z \preceq y.
\]

Remark: \(x \Rightarrow y = \sup \{z \mid x \otimes z \preceq y\}\)
Annotation Domain (Formally)

- An annotation domain is an algebraic structure that is well-known for Many-Valued FOL
- **Annotation Domain**: is a *residuated bounded lattice*

\[ D = \langle L, \preceq, \wedge, \vee, \otimes, \Rightarrow, \bot, \top \rangle, \]

i.e.

1. \[ \langle L, \preceq, \wedge, \vee, \bot, \top \rangle \] is a bounded lattice, where \( \bot \) and \( \top \) are bottom and top elements, \( \wedge \) and \( \vee \) are the meet and join operators;
2. \[ \langle L, \otimes, \top \rangle \] is a commutative monoid;
3. \( \Rightarrow \) is the so-called residuum implication of \( \otimes \), i.e. for all \( x, y, z \),

\[ z \preceq (x \Rightarrow y) \iff x \otimes z \preceq y. \]

Remark: \( x \Rightarrow y = \sup \{ z \mid x \otimes z \preceq y \} \)
Annotation Domain (Formally)

- An annotation domain is an algebraic structure that is well-known for Many-Valued FOL
- **Annotation Domain**: is a *residuated bounded lattice*

\[ D = \langle L, \preceq, \land, \lor, \otimes, \Rightarrow, \bot, \top \rangle, \]

i.e.

1. \( \langle L, \preceq, \land, \lor, \bot, \top \rangle \) is a bounded lattice, where \( \bot \) and \( \top \) are bottom and top elements, \( \land \) and \( \lor \) are the meet and join operators;
2. \( \langle L, \otimes, \top \rangle \) is a commutative monoid;
3. \( \Rightarrow \) is the so-called residuum implication of \( \otimes \), i.e. for all \( x, y, z \),

\[ z \preceq (x \Rightarrow y) \text{ iff } x \otimes z \preceq y. \]

Remark: \( x \Rightarrow y = \sup \{ z \mid x \otimes z \preceq y \} \)
Other domains: Example

- **Fuzzy**: $(AAA10\text{Hotel}, \text{closeTo}, \text{OlimpicPark}): 0.8$
  - $L = [0, 1]$
  - $\otimes =$ any t-norm
  - $\lor =$ max

- **Provenance**: $(\text{umberto}, \text{knows}, \text{axel}): p$
  - $L =$ DNF propositional formulae over URIs
  - $\otimes =$ $\land$
  - $\lor =$ $\lor$

- **Multiple Domains**: our frameworks allows to combine domains
  
  $(\text{CountryXXX}, \text{type}, \text{Dangerous}): ([1975, 1983], 0.8, 0.6)$

$Time \times \text{Fuzzy} \times \text{Trust}$
Other domains: Example

- **Fuzzy**: \((AAAI10\text{Hotel}, closeTo, OlimpicPark): 0.8\)
  - \(L = [0, 1]\)
  - \(\otimes = \text{any t-norm}\)
  - \(\lor = \text{max}\)

- **Provenance**: \((umberto, knows, axel): p\)
  - \(L = \text{DNF propositional formulae over URIs}\)
  - \(\otimes = \land\)
  - \(\lor = \lor\)

- **Multiple Domains**: our frameworks allows to combine domains
  \((CountryXXX, type, Dangerous): \langle [1975, 1983], 0.8, 0.6 \rangle\)

\(Time \times Fuzzy \times Trust\)
Other domains: Example

- **Fuzzy**: \((\text{AAA}10\text{Hotel}, \text{closeTo}, \text{OlimpicPark}): 0.8\)
  - \(L = [0, 1]\)
  - \(\otimes = \text{any t-norm}\)
  - \(\lor = \text{max}\)

- **Provenance**: \((\text{umberto}, \text{knows}, \text{axel}): p\)
  - \(L = \text{DNF propositional formulae over URIs}\)
  - \(\otimes = \land\)
  - \(\lor = \lor\)

- **Multiple Domains**: our frameworks allows to combine domains
  \((\text{CountryXXX}, \text{type}, \text{Dangerous}): \langle[1975, 1983], 0.8, 0.6\rangle\)

\(Time \times Fuzzy \times Trust\)
Annotated RDFS Semantics

- Semantics generalises that of crisp RDFS
- **Annotated RDF interpretation** $\mathcal{I}$ over a vocabulary $V$ is a tuple

\[ \mathcal{I} = \langle \Delta_R, \Delta_P, \Delta_C, \Delta_L, P[\cdot], C[\cdot], \cdot \mathcal{I} \rangle, \]

where
- $\Delta_R, \Delta_P, \Delta_C, \Delta_L$ are the finite interpretations domains of $\mathcal{I}$
- $P[\cdot], C[\cdot], \cdot \mathcal{I}$ are the interpretation functions of $\mathcal{I}$
\[ \mathcal{I} = \langle \Delta_R, \Delta_P, \Delta_C, \Delta_L, P[\cdot], C[\cdot], \cdot^\mathcal{I} \rangle \]

Common parts between Crisp RDFS and Annotated RDFS

1. $\Delta_R$ is a nonempty set of resources, called the domain or universe of $\mathcal{I}$
2. $\Delta_P$ is a set of property names (not necessarily disjoint from $\Delta_R$)
3. $\Delta_C \subseteq \Delta_R$ is a distinguished subset of $\Delta_R$ identifying if a resource denotes a class of resources
4. $\Delta_L \subseteq \Delta_R$, a set of literal values, $\Delta_L$ contains all plain literals in $L \cap V$
5. $\cdot^\mathcal{I}$ maps each $t \in UL \cap V$ into a value $t^\mathcal{I} \in \Delta_R \cup \Delta_P$, i.e. assigns a resource or a property name to each element of $UL$ in $V$, and such that $\cdot^\mathcal{I}$ is the identity for plain literals and assigns an element in $\Delta_R$ to elements in $L$
6. $\cdot^\mathcal{I}$ maps each variable $x \in B$ into a value $x^\mathcal{I} \in \Delta_R$, i.e. assigns a resource to each variable in $B$

7. What are $P[\cdot]$ and $C[\cdot]$?
\[ \mathcal{I} = \langle \Delta_R, \Delta_P, \Delta_C, \Delta_L, P[\cdot], C[\cdot], I \rangle \]

Common parts between Crisp RDFS and Annotated RDFS

1. \( \Delta_R \) is a nonempty set of resources, called the domain or universe of \( \mathcal{I} \)
2. \( \Delta_P \) is a set of property names (not necessarily disjoint from \( \Delta_R \))
3. \( \Delta_C \subseteq \Delta_R \) is a distinguished subset of \( \Delta_R \) identifying if a resource denotes a class of resources
4. \( \Delta_L \subseteq \Delta_R \), a set of literal values, \( \Delta_L \) contains all plain literals in \( L \cap V \)
5. \( I \) maps each \( t \in UL \cap V \) into a value \( t^I \in \Delta_R \cup \Delta_P \), i.e. assigns a resource or a property name to each element of \( UL \) in \( V \), and such that \( I \) is the identity for plain literals and assigns an element in \( \Delta_R \) to elements in \( L \)
6. \( I \) maps each variable \( x \in B \) into a value \( x^I \in \Delta_R \), i.e. assigns a resource to each variable in \( B \)
7. What are \( P[\cdot] \) and \( C[\cdot] \)?
Common parts between Crisp RDFS and Annotated RDFS

1. $\Delta_R$ is a nonempty set of resources, called the domain or universe of $\mathcal{I}$
2. $\Delta_P$ is a set of property names (not necessarily disjoint from $\Delta_R$)
3. $\Delta_C \subseteq \Delta_R$ is a distinguished subset of $\Delta_R$ identifying if a resource denotes a class of resources
4. $\Delta_L \subseteq \Delta_R$, a set of literal values, $\Delta_L$ contains all plain literals in $L \cap V$
5. $\cdot^\mathcal{I}$ maps each $t \in UL \cap V$ into a value $t^\mathcal{I} \in \Delta_R \cup \Delta_P$, i.e. assigns a resource or a property name to each element of $UL$ in $V$, and such that $\cdot^\mathcal{I}$ is the identity for plain literals and assigns an element in $\Delta_R$ to elements in $L$
6. $\cdot^\mathcal{I}$ maps each variable $x \in B$ into a value $x^\mathcal{I} \in \Delta_R$, i.e. assigns a resource to each variable in $B$
7. What are $P[\cdot]$ and $C[\cdot]$?
\[ \mathcal{I} = \langle \Delta_R, \Delta_P, \Delta_C, \Delta_L, P[\cdot], C[\cdot], \cdot^\mathcal{I} \rangle \]

Common parts between Crisp RDFS and Annotated RDFS

1. \( \Delta_R \) is a nonempty set of resources, called the domain or universe of \( \mathcal{I} \)
2. \( \Delta_P \) is a set of property names (not necessarily disjoint from \( \Delta_R \))
3. \( \Delta_C \subseteq \Delta_R \) is a distinguished subset of \( \Delta_R \) identifying if a resource denotes a class of resources
4. \( \Delta_L \subseteq \Delta_R \), a set of literal values, \( \Delta_L \) contains all plain literals in \( L \cap V \)
5. \( \cdot^\mathcal{I} \) maps each \( t \in UL \cap V \) into a value \( t^\mathcal{I} \in \Delta_R \cup \Delta_P \), i.e. assigns a resource or a property name to each element of \( UL \) in \( V \), and such that \( \cdot^\mathcal{I} \) is the identity for plain literals and assigns an element in \( \Delta_R \) to elements in \( L \)
6. \( \cdot^\mathcal{I} \) maps each variable \( x \in B \) into a value \( x^\mathcal{I} \in \Delta_R \), i.e. assigns a resource to each variable in \( B \)
7. What are \( P[\cdot] \) and \( C[\cdot] \)?
$$\mathcal{I} = \langle \Delta_R, \Delta_P, \Delta_C, \Delta_L, P[\cdot], C[\cdot], \cdot^\mathcal{I} \rangle$$

Common parts between Crisp RDFS and Annotated RDFS

1. $\Delta_R$ is a nonempty set of resources, called the domain or universe of $\mathcal{I}$
2. $\Delta_P$ is a set of property names (not necessarily disjoint from $\Delta_R$)
3. $\Delta_C \subseteq \Delta_R$ is a distinguished subset of $\Delta_R$ identifying if a resource denotes a class of resources
4. $\Delta_L \subseteq \Delta_R$, a set of literal values, $\Delta_L$ contains all plain literals in $L \cap V$
5. $\cdot^\mathcal{I}$ maps each $t \in UL \cap V$ into a value $t^\mathcal{I} \in \Delta_R \cup \Delta_P$, i.e. assigns a resource or a property name to each element of $UL$ in $V$, and such that $\cdot^\mathcal{I}$ is the identity for plain literals and assigns an element in $\Delta_R$ to elements in $L$
6. $\cdot^\mathcal{I}$ maps each variable $x \in B$ into a value $x^\mathcal{I} \in \Delta_R$, i.e. assigns a resource to each variable in $B$
7. What are $P[\cdot]$ and $C[\cdot]$ ?
\[ \mathcal{I} = \langle \Delta_R, \Delta_P, \Delta_C, \Delta_L, \mathcal{P}[\cdot], \mathcal{C}[\cdot], \cdot^\mathcal{I} \rangle \]

Common parts between Crisp RDFS and Annotated RDFS

1. \( \Delta_R \) is a nonempty set of resources, called the domain or universe of \( \mathcal{I} \)
2. \( \Delta_P \) is a set of property names (not necessarily disjoint from \( \Delta_R \))
3. \( \Delta_C \subseteq \Delta_R \) is a distinguished subset of \( \Delta_R \) identifying if a resource denotes a class of resources
4. \( \Delta_L \subseteq \Delta_R \), a set of literal values, \( \Delta_L \) contains all plain literals in \( L \cap V \)
5. \( \cdot^\mathcal{I} \) maps each \( t \in UL \cap V \) into a value \( t^\mathcal{I} \in \Delta_R \cup \Delta_P \), i.e. assigns a resource or a property name to each element of \( UL \) in \( V \), and such that \( \cdot^\mathcal{I} \) is the identity for plain literals and assigns an element in \( \Delta_R \) to elements in \( L \)
6. \( \cdot^\mathcal{I} \) maps each variable \( x \in B \) into a value \( x^\mathcal{I} \in \Delta_R \), i.e. assigns a resource to each variable in \( B \)
7. What are \( \mathcal{P}[\cdot] \) and \( \mathcal{C}[\cdot] \) ?
\[ \mathcal{I} = \langle \Delta_R, \Delta_P, \Delta_C, \Delta_L, P[\cdot], C[\cdot], \mathcal{I} \rangle \]

Common parts between Crisp RDFS and Annotated RDFS

1. \( \Delta_R \) is a nonempty set of resources, called the domain or universe of \( \mathcal{I} \)
2. \( \Delta_P \) is a set of property names (not necessarily disjoint from \( \Delta_R \))
3. \( \Delta_C \subseteq \Delta_R \) is a distinguished subset of \( \Delta_R \) identifying if a resource denotes a class of resources
4. \( \Delta_L \subseteq \Delta_R \), a set of literal values, \( \Delta_L \) contains all plain literals in \( L \cap V \)
5. \( \mathcal{I} \) maps each \( t \in UL \cap V \) into a value \( t^\mathcal{I} \in \Delta_R \cup \Delta_P \), i.e. assigns a resource or a property name to each element of \( UL \) in \( V \), and such that \( \mathcal{I} \) is the identity for plain literals and assigns an element in \( \Delta_R \) to elements in \( L \)
6. \( \mathcal{I} \) maps each variable \( x \in B \) into a value \( x^\mathcal{I} \in \Delta_R \), i.e. assigns a resource to each variable in \( B \)
7. What are \( P[\cdot] \) and \( C[\cdot] \)?
Crisp \( P[\cdot] \): \( P[\cdot] \) maps each property name \( p \in \Delta_P \) into a subset \( P[p] \subseteq \Delta_R \times \Delta_R \), \( i.e. \) assigns an extension to each property name; \( i.e. \)

\[
P[p] : \Delta_R \times \Delta_R \rightarrow \{0, 1\}
\]

Annotated \( P[\cdot] \): \( P[\cdot] \) maps each property name \( p \in \Delta_P \) into a function \( P[p] : \Delta_R \times \Delta_R \rightarrow L \), \( i.e. \) assigns an annotation term to each pair of resources;

Crisp \( C[\cdot] \): \( C[\cdot] \) maps each class \( c \in \Delta_C \) into a subset \( C[c] \subseteq \Delta_R \), \( i.e. \) assigns a set of resources to every resource denoting a class; \( i.e. \)

\[
C[c] : \Delta_R \rightarrow \{0, 1\}
\]

Annotated \( C[\cdot] \): \( C[\cdot] \) maps each class \( c \in \Delta_C \) into a function \( C[c] : \Delta_R \rightarrow L \), \( i.e. \) assigns an annotation term to every resource
Crisp $P[\cdot]$: $P[\cdot]$ maps each property name $p \in \Delta_P$ into a subset $P[p] \subseteq \Delta_R \times \Delta_R$, i.e. assigns an extension to each property name; i.e.

$$P[p] : \Delta_R \times \Delta_R \rightarrow \{0, 1\}$$

Annotated $P[\cdot]$: $P[\cdot]$ maps each property name $p \in \Delta_P$ into a function $P[p] : \Delta_R \times \Delta_R \rightarrow \mathcal{L}$, i.e. assigns an annotation term to each pair of resources;

Crisp $C[\cdot]$: $C[\cdot]$ maps each class $c \in \Delta_C$ into a subset $C[c] \subseteq \Delta_R$, i.e. assigns a set of resources to every resource denoting a class; i.e.

$$C[c] : \Delta_R \rightarrow \{0, 1\}$$

Annotated $C[\cdot]$: $C[\cdot]$ maps each class $c \in \Delta_C$ into a function $C[c] : \Delta_R \rightarrow \mathcal{L}$, i.e. assigns an annotation term to every resource
Crisp $P[\cdot]$ : $P[\cdot]$ maps each property name $p \in \Delta_P$ into a subset $P[p] \subseteq \Delta_R \times \Delta_R$, i.e. assigns an extension to each property name; i.e.

$$P[p] : \Delta_R \times \Delta_R \rightarrow \{0, 1\}$$

Annotated $P[\cdot]$ : $P[\cdot]$ maps each property name $p \in \Delta_P$ into a function $P[p] : \Delta_R \times \Delta_R \rightarrow L$, i.e. assigns an annotation term to each pair of resources;

Crisp $C[\cdot]$ : $C[\cdot]$ maps each class $c \in \Delta_C$ into a subset $C[c] \subseteq \Delta_R$, i.e. assigns a set of resources to every resource denoting a class; i.e.

$$C[c] : \Delta_R \rightarrow \{0, 1\}$$

Annotated $C[\cdot]$ : $C[\cdot]$ maps each class $c \in \Delta_C$ into a function $C[c] : \Delta_R \rightarrow L$, i.e. assigns an annotation term to every resource
Crisp \( P[\cdot] \): \( P[\cdot] \) maps each property name \( p \in \Delta_P \) into a subset \( P[p] \subseteq \Delta_R \times \Delta_R \), i.e. assigns an extension to each property name; i.e.

\[
P[p] : \Delta_R \times \Delta_R \rightarrow \{0, 1\}
\]

Annotated \( P[\cdot] \): \( P[\cdot] \) maps each property name \( p \in \Delta_P \) into a function \( P[p] : \Delta_R \times \Delta_R \rightarrow L \), i.e. assigns an annotation term to each pair of resources;

Crisp \( C[\cdot] \): \( C[\cdot] \) maps each class \( c \in \Delta_C \) into a subset \( C[c] \subseteq \Delta_R \), i.e. assigns a set of resources to every resource denoting a class; i.e.

\[
C[c] : \Delta_R \rightarrow \{0, 1\}
\]

Annotated \( C[\cdot] \): \( C[\cdot] \) maps each class \( c \in \Delta_C \) into a function \( C[c] : \Delta_R \rightarrow L \), i.e. assigns an annotation term to every resource.
Crisp RDFS : For ground triples, $\mathcal{I} \models (s, p, o)$ if
- $p$ is interpreted as a property name
- $s$ and $o$ are interpreted as resources
- the interpretation of the pair $(s, o)$ belongs to the extension of the property assigned to $p$

Annotated RDF : For ground triples, $\mathcal{I} \models (s, p, o) : \lambda$ if
- $p$ is interpreted as a property name
- $s$ and $o$ are interpreted as resources
- the interpretation of the pair $(s, o)$ belongs to the extension of the property assigned to $p$ to a wider extent than $\lambda$
Models (Intuitively)

Crisp RDFS : For ground triples, $I \models (s, p, o)$ if
- $p$ is interpreted as a property name
- $s$ and $o$ are interpreted as resources
- the interpretation of the pair $(s, o)$ belongs to the extension of the property assigned to $p$

Annotated RDF : For ground triples, $I \models (s, p, o): \lambda$ if
- $p$ is interpreted as a property name
- $s$ and $o$ are interpreted as resources
- the interpretation of the pair $(s, o)$ belongs to the extension of the property assigned to $p$ to a wider extent than $\lambda$
Let $G$ be a graph over $\rho_{df}$.

- An interpretation $\mathcal{I}$ is a model of $G$ under $\rho_{df}$, denoted $\mathcal{I} \models G$, iff
  - $\mathcal{I}$ is an interpretation over the vocabulary $\rho_{df} \cup \text{universe}(G)$
  - $\mathcal{I}$ satisfies the following conditions:
Crisp Simple:

1. for each \((s, p, o) \in G, p^I \in \Delta_P\) and \((s^I, o^I) \in P[p^I]\);

Annotated Simple:

1. for each \((s, p, o) : \lambda \in G, p^I \in \Delta_P\) and \(P[p^I](s^I, o^I) \geq \lambda\);

Crisp Subclass:

1. \(P[sc^I]\) is transitive over \(\Delta_C\);
2. if \((c, d) \in P[sc^I]\) then \(c, d \in \Delta_C\) and \(C[c] \subseteq C[d]\);

Annotated Subclass:

1. \(P[sc^I]\) is transitive over \(\Delta_C\);
2. \(P[sc^I](c, d) = \min_{x \in \Delta_R} C[c](x) \Rightarrow C[d](x)\).
Crisp Simple:

1. for each \((s, p, o) \in G, p^\mathcal{T} \in \Delta_P\) and \((s^\mathcal{T}, o^\mathcal{T}) \in P[p^\mathcal{T}]\);

Annotated Simple:

1. for each \((s, p, o): \lambda \in G, p^\mathcal{T} \in \Delta_P\) and \(P[p^\mathcal{T}](s^\mathcal{T}, o^\mathcal{T}) \geq \lambda\);

Crisp Subclass:

1. \(P[sc^\mathcal{T}]\) is transitive over \(\Delta_C\);
2. if \((c, d) \in P[sc^\mathcal{T}]\) then \(c, d \in \Delta_C\) and \(C[c] \subseteq C[d]\);

Annotated Subclass:

1. \(P[sc^\mathcal{T}]\) is transitive over \(\Delta_C\);
2. \(P[sc^\mathcal{T}](c, d) = \min_{x \in \Delta_R} C[c](x) \Rightarrow C[d](x)\).
Crisp Simple:

1. for each \((s, p, o) \in G\), \(p^I \in \Delta_P\) and \((s^I, o^I) \in P[p^I]\);

Annotated Simple:

1. for each \((s, p, o)\): \(\lambda \in G\), \(p^I \in \Delta_P\) and \(P[p^I](s^I, o^I) \succeq \lambda\);

Crisp Subclass:

1. \(P[sc^I]\) is transitive over \(\Delta_C\);
2. if \((c, d) \in P[sc^I]\) then \(c, d \in \Delta_C\) and \(C[c] \subseteq C[d]\);

Annotated Subclass:

1. \(P[sc^I]\) is transitive over \(\Delta_C\);
2. \(P[sc^I](c, d) = \min_{x \in \Delta_R} C[c](x) \Rightarrow C[d](x)\).
Crisp Simple:
1. for each \((s, p, o) \in G, p^I \in \Delta_P\) and \((s^I, o^I) \in P[p^I]\);

Annotated Simple:
1. for each \((s, p, o): \lambda \in G, p^I \in \Delta_P\) and
   \[P[p^I](s^I, o^I) \geq \lambda;\]

Crisp Subclass:
1. \(P[sc^I]\) is transitive over \(\Delta_C\);
2. if \((c, d) \in P[sc^I]\) then \(c, d \in \Delta_C\) and \(C[c] \subseteq C[d]\);

Annotated Subclass:
1. \(P[sc^I]\) is transitive over \(\Delta_C\);
2. \(P[sc^I](c, d) = \min_{x \in \Delta_R} C[c](x) \Rightarrow C[d](x)\).
Models (cont.)

- In the crisp case, if \( c \) is a sub-class of \( d \) then we impose that \( C[c] \subseteq C[d] \).
- This may be seen as the formula

\[
\forall x. c(x) \Rightarrow d(x),
\]

- In the annotated framework this is (\( \forall x \equiv \min_{x \in \Delta_R} \))

\[
P[sc^I](c, d) = \min_{x \in \Delta_R} C[c](x) \Rightarrow C[d](x);
\]

- Transitivity: for a set \( \Delta \subseteq \Delta_R \cup \Delta_P \), we say that a function \( f: \Delta \times \Delta \to L \) is transitive) over \( \Delta \) iff for all \( x, z \in \Delta \),

\[
f(x, y) \succeq \max_{z \in \Delta} \{ f(x, z) \otimes f(z, y) \}\]
Crisp Subproperty:

1. $P[\text{sp}^T]$ is transitive over $\Delta_P$;
2. if $(p, q) \in P[\text{sp}^T]$ then $p, q \in \Delta_P$ and $P[p] \subseteq P[q]$;

Annotated Subproperty:

1. $P[\text{sp}^T]$ is transitive over $\Delta_P$;
2. $P[\text{sp}^T](p, q) = \min_{(x, y) \in \Delta_R \times \Delta_R} P[p](x, y) \Rightarrow P[q](x, y)$
Crisp Typing I:

1. \( x \in \mathbb{C}[c] \) iff \((x, c)\) \(\in\) \(P[type^I]\);
2. if \((p, c)\) \(\in\) \(P[dom^I]\) and \((x, y)\) \(\in\) \(P[p]\) then \(x \in \mathbb{C}[c]\);
3. if \((p, c)\) \(\in\) \(P[range^I]\) and \((x, y)\) \(\in\) \(P[p]\) then \(y \in \mathbb{C}[c]\);

Annotated Typing I:

1. \( \mathbb{C}[c](x) = P[type^I](x, c) \);
2. \( P[dom^I](p, c) = \inf_{(x,y) \in \Delta_R \times \Delta_R} P[p](x,y) \Rightarrow \mathbb{C}[c](x) \);
3. \( P[range^I](p, c) = \inf_{(x,y) \in \Delta_R \times \Delta_R} P[p](x,y) \Rightarrow \mathbb{C}[c](y) \);
Crisp Typing II:

1. For each $e \in \rho \text{df}$, $e^\mathcal{I} \in \Delta_P$
2. if $(p, c) \in P[\text{dom}^\mathcal{I}]$ then $p \in \Delta_P$ and $c \in \Delta_C$
3. if $(p, c) \in P[\text{range}^\mathcal{I}]$ then $p \in \Delta_P$ and $c \in \Delta_C$
4. if $(x, c) \in P[\text{type}^\mathcal{I}]$ then $c \in \Delta_C$

Annotated Typing II:

1. For each $e \in \rho \text{df}$, $e^\mathcal{I} \in \Delta_P$
2. $P[\text{dom}^\mathcal{I}](p, c)$ is defined only for $p \in \Delta_P$ and $c \in \Delta_C$
3. $P[\text{range}^\mathcal{I}](p, c)$ is defined only for $p \in \Delta_P$ and $c \in \Delta_C$
4. $P[\text{type}^\mathcal{I}](x, c)$ is defined only for $c \in \Delta_C$
$G$ entails $H$ under $\rho$ df, denoted $G \models H$, iff
- every model under $\rho$ df of $G$ is also a model under $\rho$ df of $H$

Proposition (Consistency)

Any annotated RDFS graph has a finite model.
Deduction System for Annotated RDFS (excerpt)

1. Crisp Subproperty:

\[
\begin{align*}
(a) & \quad \frac{(A, \text{sp}, B), (B, \text{sp}, C)}{(A, \text{sp}, C)} \\
(b) & \quad \frac{(A, \text{sp}, B), (X, A, Y)}{(X, B, Y)}
\end{align*}
\]

2. Annotated Subproperty:

\[
\begin{align*}
(a) & \quad \frac{(A, \text{sp}, B) : \lambda_1, (B, \text{sp}, C) : \lambda_2}{(A, \text{sp}, C) : \lambda_1 \otimes \lambda_1} \\
(b) & \quad \frac{(A, \text{sp}, B) : \lambda_1, (X, A, Y) : \lambda_2}{(X, B, Y) : \lambda_1 \otimes \lambda_2}
\end{align*}
\]
1. Crisp Subproperty:

   (a) \( \frac{(A, sp, B), (B, sp, C)}{(A, sp, C)} \)

   (b) \( \frac{(A, sp, B), (X, A, Y)}{(X, B, Y)} \)

2. Annotated Subproperty:

   (a) \( \frac{(A, sp, B): \lambda_1, (B, sp, C): \lambda_2}{(A, sp, C): \lambda_1 \otimes \lambda_1} \)

   (b) \( \frac{(A, sp, B): \lambda_1, (X, A, Y): \lambda_2}{(X, B, Y): \lambda_1 \otimes \lambda_2} \)
1. Crisp Subclass:

(a) \( \frac{(A,sc,B),(B,sc,C)}{(A,sc,C)} \)  
(b) \( \frac{(A,sc,B),(X,type,A)}{(X,type,B)} \)

2. Annotated Subclass:

(a) \( \frac{(A,sc,B): \lambda_1,(B,sc,C): \lambda_2}{(A,sc,C): \lambda_1 \otimes \lambda_2} \)  
(b) \( \frac{(A,sc,B): \lambda_1,(X,type,A): \lambda_2}{(X,type,B): \lambda_1 \otimes \lambda_2} \)

3. Crisp Typing:

(a) \( \frac{(A,dom,B),(X,A,Y)}{(X,type,B)} \)  
(b) \( \frac{(A,range,B),(X,A,Y)}{(Y,type,B)} \)

4. Annotated Typing:

(a) \( \frac{(A,dom,B): \lambda_1,(X,A,Y): \lambda_2}{(X,type,B): \lambda_1 \otimes \lambda_2} \)  
(b) \( \frac{(A,range,B): \lambda_1,(X,A,Y): \lambda_2}{(Y,type,B): \lambda_1 \otimes \lambda_2} \)
1. Crisp Subclass:

\[
\begin{align*}
(a) & \quad \frac{(A, \text{sc}, B), (B, \text{sc}, C)}{(A, \text{sc}, C)} \quad (b) & \quad \frac{(A, \text{sc}, B), (X, \text{type}, A)}{(X, \text{type}, B)}
\end{align*}
\]

2. Annotated Subclass:

\[
\begin{align*}
(a) & \quad \frac{(A, \text{sc}, B) : \lambda_1, (B, \text{sc}, C) : \lambda_2}{(A, \text{sc}, C) : \lambda_1 \otimes \lambda_2} \quad (b) & \quad \frac{(A, \text{sc}, B) : \lambda_1, (X, \text{type}, A) : \lambda_2}{(X, \text{type}, B) : \lambda_1 \otimes \lambda_2}
\end{align*}
\]

3. Crisp Typing:

\[
\begin{align*}
(a) & \quad \frac{(A, \text{dom}, B), (X, A, Y)}{(X, \text{type}, B)} \quad (b) & \quad \frac{(A, \text{range}, B), (X, A, Y)}{(Y, \text{type}, B)}
\end{align*}
\]

4. Annotated Typing:

\[
\begin{align*}
(a) & \quad \frac{(A, \text{dom}, B) : \lambda_1, (X, A, Y) : \lambda_2}{(X, \text{type}, B) : \lambda_1 \otimes \lambda_2} \quad (b) & \quad \frac{(A, \text{range}, B) : \lambda_1, (X, A, Y) : \lambda_2}{(Y, \text{type}, B) : \lambda_1 \otimes \lambda_2}
\end{align*}
\]
1. Crisp Subclass:

(a) \[ \frac{(A, sc, B), (B, sc, C)}{(A, sc, C)} \] 
(b) \[ \frac{(A, sc, B), (X, type, A)}{(X, type, B)} \]

2. Annotated Subclass:

(a) \[ \frac{\lambda_1 \cdot (B, sc, C)}{(A, sc, C)} : \lambda_2 \] 
(b) \[ \frac{(A, sc, B) : \lambda_1 \cdot (X, type, A)}{(X, type, B)} : \lambda_2 \]

3. Crisp Typing:

(a) \[ \frac{(A, dom, B), (X, A, Y)}{(X, type, B)} \] 
(b) \[ \frac{(A, range, B), (X, A, Y)}{(Y, type, B)} \]

4. Annotated Typing:

(a) \[ \frac{(X, type, B)}{(A, dom, B) : \lambda_1 \cdot (X, A, Y) : \lambda_2} \] 
(b) \[ \frac{(Y, type, B)}{(A, range, B) : \lambda_1 \cdot (X, A, Y) : \lambda_2} \]
1. Crisp Implicit Typing:

(a) \[ \frac{(A, \text{dom}, B), (C, \text{sp}, A), (X, C, Y)}{(X, \text{type}, B)} \]

(b) \[ \frac{(A, \text{range}, B), (C, \text{sp}, A), (X, C, Y)}{(Y, \text{type}, B)} \]

2. Annotated Implicit Typing:

(a) \[ \frac{(A, \text{dom}, B): \lambda_1, (C, \text{sp}, A): \lambda_2, (X, C, Y): \lambda_3}{(X, \text{type}, B): \lambda_1 \otimes \lambda_2 \otimes \lambda_3} \]

(b) \[ \frac{(A, \text{range}, B): \lambda_1, (C, \text{sp}, A): \lambda_2, (X, C, Y): \lambda_3}{(Y, \text{type}, B): \lambda_1 \otimes \lambda_2 \otimes \lambda_3} \]
The annotated rules carry over all RDFS rules:

- If a classical RDFS triple $\tau$ can be inferred by applying a classical RDFS inference rule to triples $\tau_1, \ldots, \tau_n$

$$\{\tau_1, \ldots, \tau_n\} \vdash_{\text{RDFS}} \tau$$

then the annotation term of $\tau$ will be $\bigotimes_i \lambda_i$, where $\lambda_i$ is the annotation of triple $\tau_i$

- That is:

$$(A) \quad \frac{\tau_1 : \lambda_1, \ldots, \tau_n : \lambda_n, \{\tau_1, \ldots, \tau_n\} \vdash_{\text{RDFS}} \tau}{\tau : \bigotimes_i \lambda_i}$$

- Eventually, we need also the Generalisation Rule:

$$\tau : \lambda_1, \tau : \lambda_2 \quad \frac{}{\tau : \lambda_1 \lor \lambda_2} \quad \text{(and remove } \tau : \lambda_1, \tau : \lambda_2 \text{ )}$$
Deduction System for Annotated RDFS (cont.)

- Notion of proof (as for crisp RDFS)
- Closure

\[ cl(G) = \{ \tau : \lambda \mid G \vdash \tau : \lambda \} \]

Proposition (Soundness, Completeness, Complexity)

For an annotated graph, the proof system \( \vdash \) is sound and complete for \( \models \), that is,

1. if \( G \vdash \tau : \lambda \) then \( G \models \tau : \lambda \)
2. if \( G \models \tau : \lambda \) then there is \( \lambda' \succeq \lambda \) with \( G \vdash \tau : \lambda' \)
3. Computational complexity: is as for RDFS, plus the cost of the operations \( \otimes \) and \( \lor \) in \( L \)
Example (Proof)

\[ G = \{ (\text{audiTT}, \text{type}, \text{SportsCar}) : 0.8, (\text{SportsCar}, \text{sc}, \text{PassengerCar}) : 0.9 \} \quad \otimes \text{is product} \]

Let us proof that

\[ G \models (\text{audiTT}, \text{type}, \text{PassengerCar}) : 0.72 \]

\begin{align*}
G & \models (\text{audiTT}, \text{type}, \text{SportsCar}) : 0.8, \quad (1) \quad \text{Hypothesis} \\
G & \models (\text{SportsCar}, \text{sc}, \text{PassengerCar}) : 0.9 \quad (2) \quad \text{Hypothesis} \\
G & \models (\text{audiTT}, \text{type}, \text{PassengerCar}) : 0.72 \quad (3) \quad \text{Rule SubClass (b) applied to (1) + (2) using product t-norm} \\
\end{align*}

Similarly, we get

\[
(\text{umberto}, \text{type}, \text{IEIEmployee}) : [1992, 2001] \\
(\text{IEIEmployee, sc, PisaCenterEmployee}) : [1968, 2000] \\
\hline
(\text{umberto, type, PisaCenterEmployee}) : [1992, 2000] \\
\]

where \([1992, 2000] = [1992, 2001] \otimes [1968, 2000] \quad (\otimes = \cap)\]
Conjunctive query:

\[ q(x, v) \leftarrow \exists y \exists v'. \varphi(x, v, y, v') \]

where

- \( \varphi(x, v, y, v') \) is a conjunction of annotated triples and built-in predicates
- \( x, y \) range over RDFS terms
- \( v, v' \) range over annotation values
- \( x, v, y \) and \( v' \) are pairwise disjoint

Example: “sports car drivers between 1975 and 1985 and the temporal term at which this was true”

\[ q(x, v) \leftarrow (x, \text{type}, \text{SportsCarDriver}) : v \land (v \leq [1975, 1985]) \]

\[ G \models q(t, c) \text{ iff for any } I \models G \text{ there is a vector } t' \text{ of terms and a vector } c' \text{ of annotation values such that } I \models \varphi(t, c, t', c') \]

Answer Set:

\[ \text{ans}(G, q) = \{ \langle t, c \rangle \mid G \models q(t, c) \text{ and for any } c' \neq c \text{ such that } G \models q(t, c'), c' \preceq c \text{ holds} \} \]

Proposition

Given a graph \( G \), \( \langle t, c \rangle \) is an answer to \( q \) iff \( \exists y \exists v'. \varphi(t, c, y, v') \) is true in the closure of \( G \).
Annotated RDFS Query Answering (cont.)

- A simple query answering procedure is the following:
  - Represent annotated triples as reified RDFS triples
  - Compute the closure of a graph off-line
  - Store the annotated RDFS triples into a relational database
  - Translate the query into SQL statement
  - Execute the SQL statement over the relational database

- A prototype has been implemented (in SWI-Prolog):
  - [http://anql.deri.org](http://anql.deri.org)
We have presented Annotated RDFS:

- It’s general and flexible
  - define an annotation domain with operations $\otimes$ and $\lor$

- Conservative extension of RDFS
- Deductive system generalises crisp RDFS
- Conservative extension of conjunctive query answering
- Implementation relatively easy (prototype already available)

Forthcoming:

- AnQL: a conservative SPARQL (1.1) extension to query annotated RDFS graphs

Questions? Ask him...