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

Introduction

- RDFS is both a logic and standard W3C Semantic Web Language
  - basic ingredient: triples (*subject*, *predicate*, *object*)
- But triples alone are often not enough ...
- RDFS statements are true with respect to a certain domain
  - Time
    - (*umberto*, *workedFor*, *ISTI*)
    - true since 2001
  - Vagueness
    - (*AAAI*10*Hotel*, *closeTo*, *OlimpicPark*)
    - true to some degree
  - Provenance
    - (*umberto*, *knows*, *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):

$$(s, p, o) \in \mathbf{UBL} \times \mathbf{U} \times \mathbf{UBL}$$

  - s is the subject
  - p is the predicate
  - o is the object
- Example:

$$(\mathit{umberto}, \mathit{workedFor}, \mathit{IEI})$$

ρdf (restricted RDFS) [Munoz et al., 2007]

- ρdf (read rho-df, the ρ from restricted rdf)
- ρdf is defined as the following subset of the RDFS vocabulary:

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

- (p, sp, q)
  - property p is a sub property of property q
- (c, sc, d)
  - class c is a sub class of class d
- (a, type, b)
  - a is of type b
- (p, dom, c)
  - domain of property p is c
- (p, range, c)
  - range of property p is c

- Graph (or Knowledge Base) is a set of triples τ
- 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 λ taken from an *Annotation Domain*

$$(s, p, o) : \lambda$$

- For instance,

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

$$(AAAI10Hotel, closeTo, OlimpicPark) : 0.8$$

$$(umberto, knows, axel) : \texttt{http://www.straccia.info/foaf.rdf}$$

Annotated RDFS: Semantics

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

$$\frac{\begin{array}{l}(\textit{umberto}, \text{type}, \textit{IEIEmployee}) : [1992, 2001] \\ (\textit{IEIEmployee}, \text{sc}, \textit{PisaCenterEmployee}) : [1968, 2000]\end{array}}{(\textit{umberto}, \text{type}, \textit{PisaCenterEmployee}) : [?, ?]}$$

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 ellements:
    - $\top = [-\infty, +\infty], \bot = \emptyset$
  - “Conjunction” function $\otimes$
    - $[1992, 2001] \otimes [1968, 2000] = [1992, 2000]$ ($\otimes$ is $\cap$)
  - “Combination” function $\vee$
    - $[1992, 2001] \vee [1995, 2003] = [1992, 2003]$
    - $[1992, 1996] \vee [2001, 2009] = \{[1992, 1996], [2001, 2009]\}$

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 ellements:
    - $\top = [-\infty, +\infty], \bot = \emptyset$
  - "Conjunction" function $\otimes$
    - $[1992, 2001] \otimes [1968, 2000] = [1992, 2000]$ ($\otimes$ is $\cap$)
  - "Combination" function $\vee$
    - $[1992, 2001] \vee [1995, 2003] = [1992, 2003]$
    - $[1992, 1996] \vee [2001, 2009] = \{[1992, 1996], [2001, 2009]\}$

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 ellements:
    - $\top = [-\infty, +\infty], \bot = \emptyset$
  - "Conjunction" function $\otimes$
    - $[1992, 2001] \otimes [1968, 2000] = [1992, 2000]$ ($\otimes$ is $\cap$)
  - "Combination" function $\vee$
    - $[1992, 2001] \vee [1995, 2003] = [1992, 2003]$
    - $[1992, 1996] \vee [2001, 2009] = \{[1992, 1996], [2001, 2009]\}$

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 ellements:
    - $\top = [-\infty, +\infty], \bot = \emptyset$
  - "Conjunction" function $\otimes$
    - $[1992, 2001] \otimes [1968, 2000] = [1992, 2000]$ ($\otimes$ is $\cap$)
  - "Combination" function $\vee$
    - $[1992, 2001] \vee [1995, 2003] = [1992, 2003]$
    - $[1992, 1996] \vee [2001, 2009] = \{[1992, 1996], [2001, 2009]\}$

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 ellements:
    - $\top = [-\infty, +\infty], \bot = \emptyset$
  - "Conjunction" function $\otimes$
    - $[1992, 2001] \otimes [1968, 2000] = [1992, 2000]$ ($\otimes$ is $\cap$)
  - "Combination" function $\vee$
    - $[1992, 2001] \vee [1995, 2003] = [1992, 2003]$
    - $[1992, 1996] \vee [2001, 2009] = \{[1992, 1996], [2001, 2009]\}$

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,

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

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,

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

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,
$$z \preceq (x \Rightarrow y) \text{ iff } x \otimes z \preceq y \ .$$

Remark: $x \Rightarrow y = \sup \{z \mid x \otimes z \preceq y\}$

Other domains: Example

- Fuzzy: $(AAAI10Hotel, closeTo, OlimpicPark)\colon 0.8$
  - $L = [0, 1]$
  - $\otimes =$ any t-norm
  - $\vee =$ max
- Provenance: $(umberto, knows, axel)\colon p$
  - $L =$ DNF propositional formulae over URIs
  - $\otimes = \wedge$
  - $\vee = \vee$
- Multiple Domains: our frameworks allows to combine domains

$$(CountryXXX, \text{type}, Dangerous)\colon \langle [1975, 1983], 0.8, 0.6 \rangle$$

$Time \times Fuzzy \times Trust$

Other domains: Example

- Fuzzy: $(AAAI10Hotel, closeTo, OlimpicPark): 0.8$
  - $L = [0, 1]$
  - $\otimes =$ any t-norm
  - $\vee = \max$
- Provenance: $(umberto, knows, axel): p$
  - $L =$ DNF propositional formulae over URIs
  - $\otimes = \wedge$
  - $\vee = \vee$
- Multiple Domains: our frameworks allows to combine domains

$$(CountryXXX, \text{type}, Dangerous): \langle [1975, 1983], 0.8, 0.6 \rangle$$

$$Time \times Fuzzy \times Trust$$

Other domains: Example

- Fuzzy: $(AAAI10Hotel, closeTo, OlimpicPark): 0.8$
  - $L = [0, 1]$
  - $\otimes =$ any t-norm
  - $\vee = \max$
- Provenance: $(umberto, knows, axel): p$
  - $L =$ DNF propositional formulae over URIs
  - $\otimes = \wedge$
  - $\vee = \vee$
- Multiple Domains: our frameworks allows to combine domains

$$(CountryXXX, \mathrm{type}, Dangerous): \langle [1975, 1983], 0.8, 0.6 \rangle$$

$Time \times Fuzzy \times Trust$

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. Δ_R is a nonempty set of resources, called the domain or universe of $\mathcal{I}$
2. Δ_P is a set of property names (not necessarily disjoint from Δ_R)
3. $\Delta_C \subseteq \Delta_R$ is a distinguished subset of Δ_R identifying if a resource denotes a class of resources
4. $\Delta_L \subseteq \Delta_R$, a set of literal values, Δ_L contains all plain literals in $\mathbf{L} \cap V$
5. $\cdot^{\mathcal{I}}$ maps each $t \in \mathbf{UL} \cap V$ 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 Δ_R to elements in **L**
6. $\cdot^{\mathcal{I}}$ maps each variable $x \in \mathbf{B}$ 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. Δ_R is a nonempty set of resources, called the domain or universe of $\mathcal{I}$
2. Δ_P is a set of property names (not necessarily disjoint from Δ_R)
3. $\Delta_C \subseteq \Delta_R$ is a distinguished subset of Δ_R identifying if a resource denotes a class of resources
4. $\Delta_L \subseteq \Delta_R$, a set of literal values, Δ_L contains all plain literals in $\mathbf{L} \cap V$
5. $\cdot^{\mathcal{I}}$ maps each $t \in \mathbf{UL} \cap V$ 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 Δ_R to elements in **L**
6. $\cdot^{\mathcal{I}}$ maps each variable $x \in \mathbf{B}$ 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. Δ_R is a nonempty set of resources, called the domain or universe of $\mathcal{I}$
2. Δ_P is a set of property names (not necessarily disjoint from Δ_R)
3. $\Delta_C \subseteq \Delta_R$ is a distinguished subset of Δ_R identifying if a resource denotes a class of resources
4. $\Delta_L \subseteq \Delta_R$, a set of literal values, Δ_L contains all plain literals in $\mathbf{L} \cap V$
5. $\cdot^{\mathcal{I}}$ maps each $t \in \mathbf{UL} \cap V$ 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 Δ_R to elements in **L**
6. $\cdot^{\mathcal{I}}$ maps each variable $x \in \mathbf{B}$ 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. Δ_R is a nonempty set of resources, called the domain or universe of $\mathcal{I}$
2. Δ_P is a set of property names (not necessarily disjoint from Δ_R)
3. $\Delta_C \subseteq \Delta_R$ is a distinguished subset of Δ_R identifying if a resource denotes a class of resources
4. $\Delta_L \subseteq \Delta_R$, a set of literal values, Δ_L contains all plain literals in $\mathbf{L} \cap V$
5. $\cdot^{\mathcal{I}}$ maps each $t \in \mathbf{UL} \cap V$ 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 Δ_R to elements in **L**
6. $\cdot^{\mathcal{I}}$ maps each variable $x \in \mathbf{B}$ 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. Δ_R is a nonempty set of resources, called the domain or universe of $\mathcal{I}$
2. Δ_P is a set of property names (not necessarily disjoint from Δ_R)
3. $\Delta_C \subseteq \Delta_R$ is a distinguished subset of Δ_R identifying if a resource denotes a class of resources
4. $\Delta_L \subseteq \Delta_R$, a set of literal values, Δ_L contains all plain literals in $\mathbf{L} \cap V$
5. $\cdot^{\mathcal{I}}$ maps each $t \in \mathbf{UL} \cap V$ 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 Δ_R to elements in **L**
6. $\cdot^{\mathcal{I}}$ maps each variable $x \in \mathbf{B}$ 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. Δ_R is a nonempty set of resources, called the domain or universe of $\mathcal{I}$
2. Δ_P is a set of property names (not necessarily disjoint from Δ_R)
3. $\Delta_C \subseteq \Delta_R$ is a distinguished subset of Δ_R identifying if a resource denotes a class of resources
4. $\Delta_L \subseteq \Delta_R$, a set of literal values, Δ_L contains all plain literals in $\mathbf{L} \cap V$
5. $\cdot^{\mathcal{I}}$ maps each $t \in \mathbf{UL} \cap V$ 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 Δ_R to elements in **L**
6. $\cdot^{\mathcal{I}}$ maps each variable $x \in \mathbf{B}$ 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. Δ_R is a nonempty set of resources, called the domain or universe of $\mathcal{I}$
2. Δ_P is a set of property names (not necessarily disjoint from Δ_R)
3. $\Delta_C \subseteq \Delta_R$ is a distinguished subset of Δ_R identifying if a resource denotes a class of resources
4. $\Delta_L \subseteq \Delta_R$, a set of literal values, Δ_L contains all plain literals in $\mathbf{L} \cap V$
5. $\cdot^{\mathcal{I}}$ maps each $t \in \mathbf{UL} \cap V$ 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 Δ_R to elements in **L**
6. $\cdot^{\mathcal{I}}$ maps each variable $x \in \mathbf{B}$ 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.*

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

Annotated $P[\![\cdot]\!]$: $P[\![\cdot]\!]$ maps each property name $p \in \Delta_P$ into a function $P[\![p]\!] : \Delta_R \times \Delta_R \to L$, *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.*

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

Annotated $C[\![\cdot]\!]$: $C[\![\cdot]\!]$ maps each class $c \in \Delta_C$ into a function $C[\![c]\!] : \Delta_R \to L$, *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.*

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

Annotated $P[\![\cdot]\!]$: $P[\![\cdot]\!]$ maps each property name $p \in \Delta_P$ into a function $P[\![p]\!] : \Delta_R \times \Delta_R \to L$, *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.*

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

Annotated $C[\![\cdot]\!]$: $C[\![\cdot]\!]$ maps each class $c \in \Delta_C$ into a function $C[\![c]\!] : \Delta_R \to L$, *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.*

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

Annotated $P[\![\cdot]\!]$: $P[\![\cdot]\!]$ maps each property name $p \in \Delta_P$ into a function $P[\![p]\!] : \Delta_R \times \Delta_R \to L$, *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.*

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

Annotated $C[\![\cdot]\!]$: $C[\![\cdot]\!]$ maps each class $c \in \Delta_C$ into a function $C[\![c]\!] : \Delta_R \to L$, *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.*

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

Annotated $P[\![\cdot]\!]$: $P[\![\cdot]\!]$ maps each property name $p \in \Delta_P$ into a function $P[\![p]\!] : \Delta_R \times \Delta_R \to L$, *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.*

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

Annotated $C[\![\cdot]\!]$: $C[\![\cdot]\!]$ maps each class $c \in \Delta_C$ into a function $C[\![c]\!] : \Delta_R \to L$, *i.e.* assigns an annotation term to every resource

Models (Intuitively)

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 λ

Models (Intuitively)

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, $\mathcal{I} \models (s, p, o) \colon \lambda$ 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 λ

Models

Let G be a graph over ρdf.

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

Crisp Simple:

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

Annotated Simple:

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

Crisp Subclass:

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

Annotated Subclass:

1. $P[\![\mathrm{sc}^{\mathcal{I}}]\!]$ is transitive over Δ_C;
2. $P[\![\mathrm{sc}^{\mathcal{I}}]\!](c, d) = \min_{x \in \Delta_R} C[\![c]\!](x) \Rightarrow C[\![d]\!](x)$.

Crisp Simple:

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

Annotated Simple:

1. for each $(s, p, o)\colon \lambda \in G$, $p^{\mathcal{I}} \in \Delta_P$ and $P[\![p^{\mathcal{I}}]\!](s^{\mathcal{I}}, o^{\mathcal{I}}) \succeq \lambda$;

Crisp Subclass:

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

Annotated Subclass:

1. $P[\![\mathrm{sc}^{\mathcal{I}}]\!]$ is transitive over Δ_C;
2. $P[\![\mathrm{sc}^{\mathcal{I}}]\!](c, d) = \min_{x \in \Delta_R} C[\![c]\!](x) \Rightarrow C[\![d]\!](x)$.

Crisp Simple:

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

Annotated Simple:

1. for each $(s, p, o) \colon \lambda \in G$, $p^{\mathcal{I}} \in \Delta_P$ and $P[\![p^{\mathcal{I}}]\!](s^{\mathcal{I}}, o^{\mathcal{I}}) \succeq \lambda$;

Crisp Subclass:

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

Annotated Subclass:

1. $P[\![\mathrm{sc}^{\mathcal{I}}]\!]$ is transitive over Δ_C;
2. $P[\![\mathrm{sc}^{\mathcal{I}}]\!](c, d) = \min_{x \in \Delta_R} C[\![c]\!](x) \Rightarrow C[\![d]\!](x)$.

Crisp Simple:

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

Annotated Simple:

1. for each $(s, p, o) \colon \lambda \in G$, $p^{\mathcal{I}} \in \Delta_P$ and $P[\![p^{\mathcal{I}}]\!](s^{\mathcal{I}}, o^{\mathcal{I}}) \succeq \lambda$;

Crisp Subclass:

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

Annotated Subclass:

1. $P[\![\mathrm{sc}^{\mathcal{I}}]\!]$ is transitive over Δ_C;
2. $P[\![\mathrm{sc}^{\mathcal{I}}]\!](c, d) = \min_{x \in \Delta_R} C[\![c]\!](x) \Rightarrow C[\![d]\!](x)$.

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 ($\forall x \equiv \min_{x \in \Delta_R}$)

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

- Transitivity: for a set $\Delta \subseteq \Delta_R \cup \Delta_P$, we say that a function $f \colon \Delta \times \Delta \to L$ is *transitive*) over Δ 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. $P[\![\mathsf{sp}^{\mathcal{I}}]\!]$ is transitive over Δ_P;
2. if $(p, q) \in P[\![\mathsf{sp}^{\mathcal{I}}]\!]$ then $p, q \in \Delta_P$ and $P[\![p]\!] \subseteq P[\![q]\!]$;

Annotated Subproperty:

1. $P[\![\mathsf{sp}^{\mathcal{I}}]\!]$ is transitive over Δ_P;
2. $P[\![\mathsf{sp}^{\mathcal{I}}]\!](p, q) = \min_{(x,y) \in \Delta_R \times \Delta_R} P[\![p]\!](x, y) \Rightarrow P[\![q]\!](x, y)$

Crisp Typing I:

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

Annotated Typing I:

1. $C[\![c]\!](x) = P[\![\mathrm{type}^{\mathcal{I}}]\!](x, c)$;
2. $P[\![\mathrm{dom}^{\mathcal{I}}]\!](p, c) = \inf_{(x,y)\in\Delta_R\times\Delta_R} P[\![p]\!](x, y) \Rightarrow C[\![c]\!](x)$;
3. $P[\![\mathrm{range}^{\mathcal{I}}]\!](p, c) = \inf_{(x,y)\in\Delta_R\times\Delta_R} P[\![p]\!](x, y) \Rightarrow C[\![c]\!](y)$;

Crisp Typing II:

1. For each e $\in \rho$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 $\in \rho$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$c \in \Delta_C$

Models (cont.)

- G entails H under ρdf, denoted $G \models H$, iff
  - every model under ρdf of G is also a model under ρdf of H

Proposition (Consistency)

Any annotated RDFS graph has a finite model.

Deduction System for Annotated RDFS (excerpt)

1. Crisp Subproperty:

$$(a) \quad \frac{(A, \mathrm{sp}, B), (B, \mathrm{sp}, C)}{(A, \mathrm{sp}, C)} \qquad (b) \quad \frac{(A, \mathrm{sp}, B), (X, A, Y)}{(X, B, Y)}$$

2. Annotated Subproperty:

$$(a) \quad \frac{(A, \mathrm{sp}, B)\colon \lambda_1, (B, \mathrm{sp}, C)\colon \lambda_2}{(A, \mathrm{sp}, C)\colon \lambda_1 \otimes \lambda_1} \qquad (b) \quad \frac{(A, \mathrm{sp}, B)\colon \lambda_1, (X, A, Y)\colon \lambda_2}{(X, B, Y)\colon \lambda_1 \otimes \lambda_2}$$

Deduction System for Annotated RDFS (excerpt)

1. Crisp Subproperty:

$$(a) \quad \frac{(A,\mathrm{sp},B),(B,\mathrm{sp},C)}{(A,\mathrm{sp},C)} \qquad (b) \quad \frac{(A,\mathrm{sp},B),(X,A,Y)}{(X,B,Y)}$$

2. Annotated Subproperty:

$$(a) \quad \frac{(A,\mathrm{sp},B)\colon \lambda_1,(B,\mathrm{sp},C)\colon \lambda_2}{(A,\mathrm{sp},C)\colon \lambda_1 \otimes \lambda_1} \qquad (b) \quad \frac{(A,\mathrm{sp},B)\colon \lambda_1,(X,A,Y)\colon \lambda_2}{(X,B,Y)\colon \lambda_1 \otimes \lambda_2}$$

1. Crisp Subclass:

$$(a) \quad \frac{(A, \mathrm{sc}, B), (B, \mathrm{sc}, C)}{(A, \mathrm{sc}, C)} \qquad (b) \quad \frac{(A, \mathrm{sc}, B), (X, \mathrm{type}, A)}{(X, \mathrm{type}, B)}$$

2. Annotated Subclass:

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

3. Crisp Typing:

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

4. Annotated Typing:

$$(a) \quad \frac{(A, \mathrm{dom}, B) : \lambda_1, (X, A, Y) : \lambda_2}{(X, \mathrm{type}, B) : \lambda_1 \otimes \lambda_2} \qquad (b) \quad \frac{(A, \mathrm{range}, B) : \lambda_1, (X, A, Y) : \lambda_2}{(Y, \mathrm{type}, B) : \lambda_1 \otimes \lambda_2}$$

1. Crisp Subclass:

$$(a) \quad \frac{(A, \mathrm{sc}, B), (B, \mathrm{sc}, C)}{(A, \mathrm{sc}, C)} \qquad (b) \quad \frac{(A, \mathrm{sc}, B), (X, \mathrm{type}, A)}{(X, \mathrm{type}, B)}$$

2. Annotated Subclass:

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

3. Crisp Typing:

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

4. Annotated Typing:

$$(a) \quad \frac{(A, \mathrm{dom}, B) : \lambda_1, (X, A, Y) : \lambda_2}{(X, \mathrm{type}, B) : \lambda_1 \otimes \lambda_2} \qquad (b) \quad \frac{(A, \mathrm{range}, B) : \lambda_1, (X, A, Y) : \lambda_2}{(Y, \mathrm{type}, B) : \lambda_1 \otimes \lambda_2}$$

1. Crisp Subclass:

$$(a) \quad \frac{(A, \mathrm{sc}, B), (B, \mathrm{sc}, C)}{(A, \mathrm{sc}, C)} \qquad (b) \quad \frac{(A, \mathrm{sc}, B), (X, \mathrm{type}, A)}{(X, \mathrm{type}, B)}$$

2. Annotated Subclass:

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

3. Crisp Typing:

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

4. Annotated Typing:

$$(a) \quad \frac{(A, \mathrm{dom}, B) : \lambda_1, (X, A, Y) : \lambda_2}{(X, \mathrm{type}, B) : \lambda_1 \otimes \lambda_2} \qquad (b) \quad \frac{(A, \mathrm{range}, B) : \lambda_1, (X, A, Y) : \lambda_2}{(Y, \mathrm{type}, B) : \lambda_1 \otimes \lambda_2}$$

1. Crisp Implicit Typing:

$$(a) \quad \frac{(A,\mathrm{dom},B),(C,\mathrm{sp},A),(X,C,Y)}{(X,\mathrm{type},B)} \qquad (b) \quad \frac{(A,\mathrm{range},B),(C,\mathrm{sp},A),(X,C,Y)}{(Y,\mathrm{type},B)}$$

2. Annotated Implicit Typing:

$$(a) \quad \frac{(A,\mathrm{dom},B)\colon \lambda_1,(C,\mathrm{sp},A)\colon \lambda_2,(X,C,Y)\colon \lambda_3}{(X,\mathrm{type},B)\colon \lambda_1 \otimes \lambda_2 \otimes \lambda_3}$$

$$(b) \quad \frac{(A,\mathrm{range},B)\colon \lambda_1,(C,\mathrm{sp},A)\colon \lambda_2,(X,C,Y)\colon \lambda_3}{(Y,\mathrm{type},B)\colon \lambda_1 \otimes \lambda_2 \otimes \lambda_3}$$

The annotated rules carry over all RDFS rules:

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

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

  then the annotation term of τ will be $\bigotimes_i \lambda_i$, where λ_i is the annotation of triple τ_i
- That is:

  $$(A) \quad \frac{\tau_1 : \lambda_1, \ldots, \tau_n : \lambda_n, \quad \{\tau_1, \ldots \tau_n\} \vdash_{\mathrm{RDFS}} \tau}{\tau : \bigotimes_i \lambda_i}$$
- Eventually, we need also the *Generalisation Rule*:

$$\frac{\tau : \lambda_1, \ \ \tau : \lambda_2}{\tau : \lambda_1 \vee \lambda_2 \ \ \text{(and remove } \tau : \lambda_1, \tau : \lambda_2 \text{)}}$$

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 $\vee$ in L*

Example (Proof)

$$G = \{(\mathit{audiTT}, \text{type}, \mathit{SportsCar}) : 0.8, (\mathit{SportsCar}, \text{sc}, \mathit{PassengerCar}) : 0.9\}$$

$\otimes$ is product

Let us proof that

$$G \models (\mathit{audiTT}, \text{type}, \mathit{PassengerCar}) : 0.72$$

$$\begin{array}{lllll}
G & \vdash & (\mathit{audiTT}, \text{type}, \mathit{SportsCar}) : 0.8, & (1) & \text{Hypothesis} \\
G & \vdash & (\mathit{SportsCar}, \text{sc}, \mathit{PassengerCar}) : 0.9 & (2) & \text{Hypothesis} \\
G & \vdash & (\mathit{audiTT}, \text{type}, \mathit{PassengerCar}) : 0.72 & (3) & \text{Rule SubClass (b) applied to (1) + (2) using product t-norm}
\end{array}$$

Similarly, we get

$$\frac{\begin{array}{l}(\mathit{umberto}, \text{type}, \mathit{IEIEmployee}) : [1992, 2001] \\ (\mathit{IEIEmployee}, \text{sc}, \mathit{PisaCenterEmployee}) : [1968, 2000]\end{array}}{(\mathit{umberto}, \text{type}, \mathit{PisaCenterEmployee}) : [1992, 2000]}$$

where $[1992, 2000] = [1992, 2001] \otimes [1968, 2000]$ ($\otimes = \cap$)

Annotated RDFS Query Answering (excerpt)

- Conjunctive query:

$$q(\mathbf{x},\mathbf{v}) \leftarrow \exists \mathbf{y} \exists \mathbf{v}'.\varphi(\mathbf{x},\mathbf{v},\mathbf{y},\mathbf{v}')$$

  where
  - $\varphi(\mathbf{x},\mathbf{v},\mathbf{y},\mathbf{v}')$ is a conjunction of annotated triples and built-in predicates
  - $\mathbf{x}, \mathbf{y}$ range over RDFS terms
  - $\mathbf{v}, \mathbf{v}'$ range over annotation values
  - $\mathbf{x}, \mathbf{v}, \mathbf{y}$ and $\mathbf{v}'$ are pairwise disjoint
- Example: “sports car drivers between 1975 and 1985 and the temporal term at which this was true”

$$q(x,v) \leftarrow (x, \mathrm{type}, \mathit{SportsCarDriver}) \colon v \ \wedge \ (v \preceq [1975, 1985])$$

- $G \models q(\mathbf{t},\mathbf{c})$ iff for any $\mathcal{I} \models G$ there is a vector $\mathbf{t}'$ of terms and a vector $\mathbf{c}'$ of annotation values such that $\mathcal{I} \models \varphi(\mathbf{t},\mathbf{c},\mathbf{t}',\mathbf{c}')$
- Answer Set:

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

Proposition

Given a graph G, $\langle \mathbf{t},\mathbf{c}\rangle$ is an answer to q iff $\exists \mathbf{y} \exists \mathbf{v}'.\varphi(\mathbf{t},\mathbf{c},\mathbf{y},\mathbf{v}')$ 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`

Summary & Outlook

- We have presented Annotated RDFS:
  - It's general and flexible
    - define an annotation domain with operations $\otimes$ and $\vee$
  - 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 ...