Logics and Reasoning - SESA Tutorial

Darko Anicic
darko.anicic@deri.org

Digital Enterprise Research Institute (DERI)
Leopold-Franzens Universität
Innsbruck, Austria

Seoul – September 10-13, 2007
Logics and Reasoning

Agenda for day 2nd: Logics and Reasoning

1. Ontologies and Reasoning
2. The WSML Language
3. IRIS Reasoner
4. Hands-on Session
Ontology Definition

Figure: Ontology Definition
Ontology Example

**Concept**
conceptual entity of the domain

**Property**
attribute describing a concept

**Relation**
relationship between concepts or properties

**Axiom**
coherent description between Concepts / Properties / Relations via logical expressions

```
holds (Professor, Lecture) :-
Lecture.topic ∈ Professor.researchField
```
Reasoning

- Reasoning is an act of deriving a conclusion from certain premises.
- The process of inferring implicit knowledge about some domain of discourse from a given (formal) knowledge of that domain.
- Reasoning is a critical success factor for Semantic Web applications.
- Reasoning problems: satisfiability, model finding, deduction...
Practical Reasons

Given key role of ontologies in e-Science and Semantic Web, it is essential to provide tools and services to help users:

- Design and maintain high quality ontologies, e.g.:
  - Meaningful - all named classes can have instances
  - Correct - captured intuitions of domain experts
  - Minimally redundant - no unintended synonyms
  - Richly axiomatised - (sufficiently) detailed descriptions

- Store (large numbers) of instances of ontology classes, e.g.:
  - Annotations from web pages (or gene product data)

- Answer queries over ontology classes and instances, e.g.:
  - Find more general/specific classes
  - Retrieve annotations/pages matching a given description

- Integrate and align multiple ontologies
Ontology Languages

**Requirements:**

- **Expressivity**
  - knowledge representation and ontology theory support
  - Dimensions to Consider: Open vs. Closed World Semantics, Classical vs. Default neg, performance etc.

- **Reasoning support**
  - sound (unambiguous, decidable)
  - support reasoners / inference engines (query answering, classification, logical ent., consistency check...)

**Semantic Web languages:**

- **Web compatibility**
- **Existing W3C Recommendations**: XML, RDF, OWL
Figure: Tim Berners Lee, International Semantic Web Conference, 2005
Simplified Stack (from a logicians point of view)
Open-World Assumption vs. Closed-World Assumption

Logic Programming

hasFather(peter, john)
hasFather(john, michael)
grandchild(x) ← hasFather(x, y), hasFather(y, z), Person(x).

- Every person is a grandchild is not deducible due to closed domain of interpretation!
- Domain bounded to known objects: peter is a grandchild.
- No tree model property required
Open-World Assumption vs. Closed-World Assumption

Description Logic

\[ \text{Person(peter)} \]
\[ \text{Person} \sqsubseteq \exists \text{father.Person} \]
\[ \exists \text{father.}(\exists \text{father.Person}) \sqsubseteq \text{Grandchild} \]

- Peter is a person.
- Each person has a father who is a person.
- Things having a father of a father who is a person are grandchildren.
- Every person is a grandchild is deducible due to open domain of interpretation!
- hasAunt(x,y) ← hasParent(x,z), hasSibling(z,y), Female(y)
- Arbitrary rule is not possible to express due to the need for tree model property

\[ \text{Darko Anicic} \]
\[ \text{SESA Tutorial - Logics and Reasoning} \]
Classical vs. Default Negation

Default Negation (i.e., negation as failure)

Rules:

\[
\text{male}(x) \leftarrow \text{father}(x).
\]
\[
\text{female}(x) \leftarrow \text{person}(x), \neg \text{male}(x).
\]
\[
\text{grandchild}(x) \leftarrow \text{hasFather}(x, y), \text{hasFather}(y, z), \text{Person}(x).
\]

Facts:

\[
\text{person}(joe), \text{person}(bill), \text{father}(joe)
\]

Value Computation:

\[
\text{male}(joe)
\]
\[
\text{female}(bill)
\]
Classical vs. Default Negation

Classical Negation

Rules:

\[
\text{male}(x) :- \text{father}(x).
\]
\[
\text{female}(x) :- \text{person}(x), \neg \text{male}(x).
\]
\[
\text{grandchild}(x) \leftarrow \text{hasFather}(x,y), \text{hasFather}(y,z), \text{Person}(x).
\]

Facts:

\[
\text{person}(joe), \text{person}(bill), \text{father}(joe)
\]

Value Computation:

\[
\text{male}(joe)
\]
Basic Model of RDF (S)

RDF triples:

- A model is a set of statements
- Statement := (predicate, subject, object)
- Predicate is a resource
- Subject is a resource
- Object is either a resource or a literal
RDF(S) - Core Classes and Properties

- **RDFS CLASS** represents the generic concept of a type or category and can be defined to represent almost everything, e.g. Web pages, people, document types...

- **RDFS PROPERTY** represents the subset of RDFS resources that are properties

- **RDFS SubClassOf** defines a subset or superset relation between classes. This property is transitive!

- **RDFS SubPropertyOf** is used to indicate that one property is a specialization of another property

- **RDFS RANGE** is used to define that the values of a property are instances of one or more stated classes

- **RDFS DOMAIN** is used to state that any resource that has a given property is an instance of one or more classes
RDF(S) - Schema Example

[Diagram showing an ontology with classes and properties such as Resource, Class, Property, Vehicle, Company, LandVehicle, SeaVehicle, Hovercraft, Number, and properties like subClassOf and producedBy.]
RDF(S) - Summary

- Simple Triple Data Model
- Basic Ontology Modeling Primitives
- URIs for Identification
- RDFS too weak to describe resources in sufficient detail
  - No localised range and domain constraints
  - No existence/cardinality constraints
  - No transitive, inverse or symmetrical properties
- Difficult to provide reasoning support (No "native" reasoners for non-standard semantics)
OWL - DL based ontology language

- DLs are a family of logic based KR formalisms
- DLs - languages mainly characterised by:
  - Set of constructors for building complex concepts and roles from simpler ones
  - Set of axioms for asserting facts about concepts, roles and individuals
- DL is a monotonic Logic
- This is due to the open world assumption.
OWL - Language

- Three species of OWL[3]
  - OWL full is union of OWL syntax and RDF
  - OWL DL restricted to DL fragment (1/4 DAML+OIL)
  - OWL Lite is "simpler" subset of OWL DL

- Semantic layering
  - OWL DL based on OWL full within DL fragment
  - OWL DL based on $SHOIN(D)$ Description Logic
  - OWL DL benefits from many years of DL research
    - Well defined semantics
    - Formal properties well understood (complexity, decidability)
    - Known reasoning algorithms
    - Implemented systems (highly optimized)
DL Concept and Role Constructors

Range of other constructors found in DLs, including:

- Number restrictions (cardinality constraints) on roles, e.g., >3 hasChild
- Qualified number restrictions, e.g., >2 hasChild.Female
- Nominals (singleton concepts), e.g., Korea
- Concrete domains (datatypes), e.g., hasAge.(21)
- Inverse roles, e.g., hasChildVAL (hasParent)
- Transitive roles, e.g., locatedIn
- Role composition, e.g., hasParent o hasBrother (uncle)
# OWL Class Constructors

<table>
<thead>
<tr>
<th>Constructor</th>
<th>DL Syntax</th>
<th>Example</th>
<th>FOL Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>intersectionOf</td>
<td>$C_1 \cap \ldots \cap C_n$</td>
<td>Human $\cap$ Male</td>
<td>$C_1(x) \land \ldots \land C_n(x)$</td>
</tr>
<tr>
<td>unionOf</td>
<td>$C_1 \cup \ldots \cup C_n$</td>
<td>Doctor $\cup$ Lawyer</td>
<td>$C_1(x) \lor \ldots \lor C_n(x)$</td>
</tr>
<tr>
<td>complementOf</td>
<td>$\neg C$</td>
<td>$\neg$ Male</td>
<td>$\neg C(x)$</td>
</tr>
<tr>
<td>oneOf</td>
<td>${x_1} \cup \ldots \cup {x_n}$</td>
<td>${john} \cup {mary}$</td>
<td>$x = x_1 \lor \ldots \lor x = x_n$</td>
</tr>
<tr>
<td>allValuesFrom</td>
<td>$\forall P.C$</td>
<td>$\forall$ hasChild.Doctor</td>
<td>$\forall y. P(x, y) \rightarrow C(y)$</td>
</tr>
<tr>
<td>someValuesFrom</td>
<td>$\exists P.C$</td>
<td>$\exists$ hasChild.Lawyer</td>
<td>$\exists y. P(x, y) \land C(y)$</td>
</tr>
<tr>
<td>maxCardinality</td>
<td>$\leq nP$</td>
<td>$\leq 1$ hasChild</td>
<td>$\exists n \ y. P(x, y)$</td>
</tr>
<tr>
<td>minCardinality</td>
<td>$\geq nP$</td>
<td>$\geq 2$ hasChild</td>
<td>$\exists n \ y. P(x, y)$</td>
</tr>
</tbody>
</table>
### OWL as DL: Axioms

<table>
<thead>
<tr>
<th>Axiom</th>
<th>DL Syntax</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>subClassOf</td>
<td>$C_1 \sqsubseteq C_2$</td>
<td>Human $\sqsubseteq$ Animal $\sqcap$ Biped</td>
</tr>
<tr>
<td>equivalentClass</td>
<td>$C_1 \equiv C_2$</td>
<td>Man $\equiv$ Human $\sqcap$ Male</td>
</tr>
<tr>
<td>disjointWith</td>
<td>$C_1 \sqsubseteq \neg C_2$</td>
<td>Male $\sqsubseteq \neg$ Female</td>
</tr>
<tr>
<td>sameIndividualAs</td>
<td>${x_1} \equiv {x_2}$</td>
<td>${\text{President Bush}} \equiv {\text{G.W. Bush}}$</td>
</tr>
<tr>
<td>differentFrom</td>
<td>${x_1} \sqsubseteq \neg {x_2}$</td>
<td>${\text{john}} \sqsubseteq \neg {\text{peter}}$</td>
</tr>
<tr>
<td>subPropertyOf</td>
<td>$P_1 \sqsubseteq P_2$</td>
<td>hasDaughter $\sqsubseteq$ hasChild</td>
</tr>
<tr>
<td>equivalentProperty</td>
<td>$P_1 \equiv P_2$</td>
<td>cost $\equiv$ price</td>
</tr>
<tr>
<td>inverseOf</td>
<td>$P_1 \equiv P_2^-$</td>
<td>hasChild $\equiv$ hasParent$^-$</td>
</tr>
<tr>
<td>transitiveProperty</td>
<td>$P^+ \sqsubseteq P$</td>
<td>ancestor$^+$ $\sqsubseteq$ ancestor</td>
</tr>
<tr>
<td>functionalProperty</td>
<td>$T \sqsubseteq \leq_1 P$</td>
<td>$T \sqsubseteq \leq_1$ hasMother</td>
</tr>
<tr>
<td>inverseFunctionalProperty</td>
<td>$T \sqsubseteq \leq_1 P^-$</td>
<td>$T \sqsubseteq \leq_1$ hasSSN$^-$</td>
</tr>
</tbody>
</table>

Axioms (mostly) reducible to inclusion ($\sqsubseteq$):

$C \equiv D$ iff both $C \sqsubseteq D$ and $D \sqsubseteq C$
OWL - Inference Tasks

- Knowledge is **correct** (captures intuitions)
  - Does $C$ **subsume** $D$ w.r.t. ontology $\mathcal{O}$? (in every model $\mathcal{I}$ of $\mathcal{O}$, $C^\mathcal{I} \subseteq D^\mathcal{I}$)

- Knowledge is **minimally redundant** (no unintended synonyms)
  - Is $C$ **equivalent** to $D$ w.r.t. $\mathcal{O}$? (in every model $\mathcal{I}$ of $\mathcal{O}$, $C^\mathcal{I} = D^\mathcal{I}$)

- Knowledge is **meaningful** (classes can have instances)
  - Is $C$ **satisfiable** w.r.t. $\mathcal{O}$? (there exists some model $\mathcal{I}$ of $\mathcal{O}$, s.t. $C^\mathcal{I} \neq \emptyset$)
OWL - Inference Tasks (2)

- **Querying** knowledge
  - Is \( x \) an **instance** of \( C \) w.r.t. \( O \)? (in **every model** \( \mathcal{I} \) of \( O \), \( x^I \in C^I \))
  - Is \( <x,y> \) an **instance** of \( R \) w.r.t. \( O \)? (in **every model** \( \mathcal{I} \) of \( O \), \( (x^I, y^I) \in R^I \))

- **Knowledge base consistency**
  - A KB \( \mathcal{K} \) is consistent iff there exists some model \( \mathcal{I} \) of \( \mathcal{K} \)

- All inference tasks reducible to KB satisfiability or concept satisfiability w.r.t. a KB
OWL DL Architecture

Knowledge Base

Tbox (schema)

\[ \text{Man} \equiv \text{Human} \sqcap \text{Male} \]
\[ \text{Happy-Father} \equiv \text{Man} \sqcap \exists \text{has-child} \]
\[ \text{Female} \sqcap \ldots \]

Abox (data)

\[ \text{John} : \text{Happy-Father} \]
\[ \langle \text{John, Mary} \rangle : \text{has-child} \]
First Order Logic - Basic Facts

- In opposition to DLs and HLs it provides significant more flexibility in writing down required axioms.
- This language is not easy to support in terms of scalable reasoning service.
- Many interesting theorems can be proven and the theorem prover community has made significant progress over the last years.
- First order language could define the common umbrella, where DLs and HLs are sublanguages and unified.
Aims to provide a language (or a set of interoperable languages) for representing the elements of WSMO: Ontologies, Web services, Goals, Mediators

WSML provides a formal language for the conceptual elements of WSMO, based on:

- Description Logics
- Logic Programming
- First-Order Logic
- F-Logic
Rationale of WSML

1. Provide a Web Service Modeling Language based on the WSMO conceptual model
   - Concrete syntax
   - Semantics

2. Provide a Rule Language for the Semantic Web

3. Many current Semantic Web languages have
   - undesirable computational properties
   - unintuitive conceptual modeling features
   - inappropriate language layering
Variants of WSML

Figure: Variants of WSML
Syntaxes for WSML

- **Human-readable syntax:**
  - Modular syntax
    - WSMO-syntax functions as "umbrella"
    - Modules for different WSML variants with clear layering
  - Syntax:
    - Inspired by OIL/OWL and F-Logic
    - Conceptual syntax
    - Logical Expression Syntax
  - Semantics is fixed in WSML variants

- **XML syntax**
  - Provides syntax transport layer

- **RDF syntax**
  - Provides basic relational language and simple ontological primitives
Syntaxes for WSML

- Ontologies
- Namespaces
- Imported Ontologies
- Used Mediators

**Extra-Logical declarations**

- Concepts
- Relations
- Instances
  - Explicitly defined in ontology
  - Retrieved from external instance store
- Axioms

**Logical Declarations**

**Non-Functional Properties**
WSML - Example

Conceptual Syntax

Man subConceptOf Human,
nonFunctionalProperties
  dc#related hasValue disjointManWoman
EndNonFunctionalProperties
  hasWife inverseOf(hasHusband) ofType (01) Wife

axiom disjointManWoman definedBy
  !- ?Human memberOf Man and ?Human memberOf Woman.

Logical Expression Syntax
WSML Logical Expressions

- Frame and first order based concrete syntax
- Elements:
  - Function symbols (e.g. f())
  - Molecules (e.g. Human subClassOf Animal, John memberOf Human, John[name hasValue "John Smith"]).
  - Predicates (e.g. distance(?x,?y,?z))
  - Logical connectives (or, and, not, implies, equivalent, impliedBy, forall, exists)

- Example: axiom WomanDefinition
  definedBy ?x memberOf Woman equivalent
  ?x memberOf Female and ?x memberOf Human.
WSML-Core

- Intersection of Description Logics and Horn Logic: *description logic programs*
- Features:
  - classes
  - attributes
  - binary relations
  - instances
  - classes and relation hierarchies
  - datatype and datatype predicates
- Compatible with OWL
- Some limitations in conceptual modeling of Ontologies:
  - no cardinality constraints
  - only "inferring" range of attributes
  - no meta-modeling
Example WSML-Core

```xml
namespace "http://www.example.org#"

ontology AboutAudioSystems

concept AudioSystem

  isFunctioning ofType Answer

concept iPod subConceptOf AudioSystem

instance no memberOf Answer
instance yes memberOf Answer
instance myipod memberOf iPod

  isFunctioning hasValue no
```
Example WSML-Core (2)

- Queries:
  
  \[
  ?x \text{ memberOf AudioSystem} \\
  ?x[\text{isFunctioning hasValue } ?Answer]
  \]

- Extra rules:

  \[
  \text{axiom fixing definedBy} \\
  \quad ?x \text{ memberOf AudioSystem} \\
  \quad \text{and } ?x[\text{isFunctioning hasValue no}] \\
  \quad \text{implies } ?x \text{ memberOf NeedsFixing.}
  \]
WSML-DL

- Extension of WSML-Core capturing the Description Logic $SHIQ(D)$
  - entailment is decidable
  - close to DL species of Web Ontology Language OWL
  - many efficient subsumption reasoners (FaCT++, Pellet, Racer)

- Some limitations in conceptual modeling of Ontologies:
  - no cardinality constraints
  - only "inferring" range of attributes
  - no meta-modeling

- Limitations in logical expressions: -From Logic Programming point-of-view, there is a lack of:
  - n-ary predicates
  - chaining variables over predicates
  - (default) negation
WSML-Flight

- Extension of WSML-Core, equivalent to Datalog with inequality and (stratified) default negation

- Features:
  - powerful rule-based language (n-ary predicates allowed)
  - meta-modeling
  - constraints
  - nonmonotonic negation (conclusions can be retracted at later points of time)

- Query answering supported by WSML2Reasoner and underlying Datalog reasoner
Example WSML-Flight

axiom con definedBy
    !- ?x[isFunctioning hasValue no] and
    ?x[isFunctioning hasValue yes].

concept AudioSystem
    isFunctioning ofType (1 1) Answer

axiom fixing2 definedBy
    ?x memberOf NeedsFixing :-
    ?x memberOf AudioSystem and
    naf ?x[isFunctioning hasValue yes].
WSML-Rule

- Extension of WSML-Flight in the direction of Logic Programming
- Features:
  - function symbols
  - unsafe rules
- Query answering is undecidable
WSML-Full

- WSML-Full unifies WSML-DL and WSML-Rule under a First-Order umbrella
- Very expressive
- Query answering is undecidable
- Specification of WSML-Full is now in progress
Mission

WSML Goal

Efficient and extensible reasoning engine for expressive rule-based languages: WSML Core/Flight/Rule.

Research Goal

Framework consisting of a collection of components which cover various aspects of reasoning with formally represented knowledge. Development of effective optimization algorithms as well as memory and storage management strategies for reasoning with large data sets.
Mission

**WSML Goal**
Efficient and extensible reasoning engine for expressive rule-based languages: WSML Core/Flight/Rule.

**Research Goal**
Framework consisting of a collection of components which cover various aspects of reasoning with formally represented knowledge. Development of effective optimization algorithms as well as memory and storage management strategies for reasoning with large data sets.
Research Methodology

WSML Core/Flight Reasoner

Datalog extended with locally stratified negation.

1. Full Datalog support
2. Support for (stratified) default negation
3. Built-in predicates
4. Integrity constraints (for checking datatypes)
Integration of IRIS with WSML

Reasoning Task

Result
(Variable Bindings)

WSML Reasoner Interface

Normalization Steps

Facades

WSML DL
WSML DL
WSML Flight
WSML Rule
WSML Core
WSML Flight
WSML Full (FOL)

Pellet
KAON2
MINS
IRIS
Theorem Prover

Darko Anicic
SESA Tutorial - Logics and Reasoning
Query-Sub-Query Recursive Framework

1. Top-down, direct evaluation [1].
2. Avoid the calculation of tuples that are not used for deriving answer.
3. Begin with constants in a query "pushing" them from goals to subgoals.
4. Use "sideways information passing" to pass constant binding information from one atom to the next in subgoals.
Consider the rule:
\[ rsg(X, Y) : \neg \text{up}(X, X_1), rsg(Y_1, X_1), \text{down}(Y_1, Y). \]

Suppose that a subquery involving \(rsg^{bf}\) is invoked (e.g. \(rsg(\text{`a'}, Y)\)):
\[
\begin{align*}
rsg^{bf}(X, Y) & : \neg \text{up}(X, X_1), rsg^{fb}(Y_1, X_1), \text{down}(Y_1, Y). \\
rsg^{fb}(X, Y) & : \neg \text{up}(X, X_1), rsg^{fb}(Y_1, X_1), \text{down}(Y_1, Y).
\end{align*}
\]
Supplementary Relations

Goal: to store information during intermediate stages of evaluation.

Consider the rule:
\[ rsg^{bf}(X,Y) \ :- \ up(X,X_1), \ rsg^{fb}(Y_1,X_1), \ down(Y_1,Y). \]

Supplementary relations, for head adorned by \( rsg^{bf} \), are:
- \( sup_0[X] \)
- \( sup_1[X,X_1] \)
- \( sup_2[X,Y_1] \)
- \( sup_3[X,Y] \)
QSQR Evaluation - Example

\[
\begin{align*}
\text{rsg}(X,Y) & \leftarrow \text{flat}(X,Y). \\
\text{rsg}(X,Y) & \leftarrow \text{up}(X,X_1), \text{rsg}(Y_1,X_1), \text{down}(Y_1,Y).
\end{align*}
\]

?- rsg('a',Y).

**Figure:** Program
QSQR Evaluation - Example

Figure: QSQR evaluation
Magic Sets

- Bottom-up technique whose efficiency rivals with top-down approaches [1, 2].
- Given a query, the magic set rewriting method generates a modified program in order to take advantage of bound variables from the query.
- A bottom-up evaluation procedure is used to evaluate the new program (e.g., seminaive evaluation).
- The bottom-up evaluation produces only the set of facts produced by the top-down approaches such as QSQ.
Relational Algebra and Logical Rules

**Relational Algebra**

Compute relations for logical rules and evaluate them using relational algebra operations [4].

Relational algebra expressions:

1. Take given relations as arguments and produce relations as results;
2. Can be combined to form complex expressions;
3. Can be efficiently evaluated using RDBMS.
Computing the Relations

Input: Datalog rules
q(X,Y) :- p(X,b) & X=Y.
q(X,Y) :- p(X,Z) & s(Z,Y).

Output: Relational algebra expressions
\[ \pi_X(\sigma_{Z=b}(P(X,Z))) \]
\[ \sigma_{X=Y}(\pi_X(\sigma_{Z=b}(P(X,Z))) \times \pi_Y(P(Y,W))) \]
\[ Q(X,Y) = \sigma_{X=Y}(\pi_X(\sigma_{Z=b}(P(X,Z))) \times \pi_Y(P(Y,W))) \]
\[ \cup \pi_{X,Y}(P(X,Z) \bowtie S(Z,Y)) \]
Rectification

Non-rectified rules:
- q('a',X,Y) :- r(X,Y).
- q(X,Y,X) :- r(Y,X).

Making heads equal:
- q(U,V,W) :- r(X,Y) & U=a & V=X & W=Y.
- q(U,V,W) :- r(Y,X) & U=X & V=Y & W=X.

Rectified rules:
- q(U,V,W) :- r(V,W) & U=a.
- q(U,V,W) :- r(V,U) & W=U.
Semi-Naive Algorithm

Computes the least fixed point of the equations to which it is applied, with respect to the given EDB relations [5]:

```plaintext
for i := 1 to m do
    P_i := ∅;
 repeat
    for i := 1 to m do
        Q_i := P_i; //save old values
    for i := 1 to m do
        P_i := EVAL(p_i, R_1,...,R_k,Q_1,...,Q_m);
    until P_i = Q_i for all i, 1 ≤ i ≤ m;
 output P_i's
```

Figure: Naive Evaluation
Algorithm

Computes the least fixed point based on incremental relations for the IDB predicates [5]:

\[
\begin{align*}
\text{for } i & := 1 \text{ to } m \text{ do} \\
\quad \Delta P_i & := \text{EVAL}(p_i, R_1, \ldots, \emptyset, \ldots, \emptyset); \\
\quad P_i & := \Delta P_i; \\
\text{end;} \\
\text{repeat} \\
\quad \text{for } i & := 1 \text{ to } m \text{ do} \\
\quad \quad \Delta Q_i & := \Delta P_i; \quad // \text{save old } \Delta P\text{'s} \\
\quad \text{for } i & := 1 \text{ to } m \text{ do begin} \\
\quad \quad \Delta P_i & := \text{EVAL-INCR}(p_i, R_1, \ldots, R_k, P_1, \ldots, P_m, \Delta Q_1, \Delta Q_m); \\
\quad \quad \Delta P_i & := \Delta P_i - P_i \quad // \text{Removes } "\text{new}" \text{ tuples that actually appeared before} \\
\text{end} \\
\quad \text{for } i & := 1 \text{ to } m \text{ do} \\
\quad \quad P_i & := P_i \cup \Delta P_i; \\
\text{until } \Delta P_i = \emptyset \text{ for all } i; \text{ output } P_i\text{'s}
\end{align*}
\]
Negation

Consider the rules:

\[ p(X) \dashv \vdash r(X) \& \neg q(X). \]
\[ q(X) \dashv \vdash r(X) \& \neg p(X). \]

For \( R=\{1\} \) the least fixed point is not unique:

\[ S_1: P=1, \ Q=\emptyset \]
\[ S_2: Q=1, \ P=\emptyset \]

IRIS supports safe, stratified rules! Algorithms for checking safeness and stratification are implemented.
Handling Negation

Consider the rule:
\[
\text{trueCousin}(X,Y) \leftarrow \text{cousin}(X,Y) \land \neg \text{Sibling}(X,Y).
\]

The rule may be evaluated as:
\[
\text{C}(X,Y) \bowtie S(X,Y)
\]

DOM - union of the symbols appearing in the EDB relations and in the rules themselves.
\[
\text{DOM} \times \text{DOM} - S(X,Y)
\]

In general for \(\neg s(X_1,...,X_n)\):
\[
\text{DOM} \times ... \times \text{DOM} \text{ (n times)} - S
\]
The IRIS Roadmap

1. Query optimization
2. Performance and scalability
3. Storage management
4. Support for function symbols
5. Wellfounded semantics implementation
Combining Reasoning with Search

1. **Goal:** Provide an infrastructure that scales for realistic semantic computing applications

2. **What:** A plugable & distributed infrastructure for Web-reasoning and search

3. **Project:** LarCK

4. **Like:** Search for Extraterrestrial Intelligence (SETI), Folding@Home and Google techniques for large scale parallelized computing
IRIS Code Access

IRIS is an open source project developed under LGPL and available at:

http://sourceforge.net/projects/iris-reasoner


Ian Horrocks and Alan Rector. The semantic web: Ontologies and owl.


Jeffrey D. Ullman.
Principles of Database and Knowledge-Base Systems, Volume II.
Thank you!
Questions, comments, suggestions please...
Examples Using WSMT

1. Set up WSMT;
2. Browsing ontologies using WSMT GUI
3. Editing ontologies using WSML Text Editor
4. Execute a query;
Examples Using Java API

1. Set up required WSMO4J factories;
2. Parse and load an ontology (written in WSML) in the reasoner;
3. Create a query;
4. Execute a query;
5. Print the loaded ontology and results after the query execution.