WELCOME! to the RuleML-2015 Tutorial (Aug. 02 2015)

Powerful Practical Semantic Rules in Rulelog: Fundamentals and Recent Progress

presented by Benjamin Grosof*, Michael Kifer**, and Paul Fodor**

* Coherent Knowledge, and Benjamin Grosof & Associates
** Stony Brook University, and Coherent Knowledge

http://coherentknowledge.com

Smart Rules for Smart Data
Preface; For More Info

• This is based on a longer previous half-day tutorial about Semantic Web Rules, given ~10 times at conferences during 2004-2013, most recently at AAAI-13

• To make this: Shortened; Reshaped focus; Updated

• This slideset will soon be available on the web, including at the Coherent publications link below

• For more info, including the longer tutorial:
  • http://coherentknowlege.com/publications

• Hoping to turn the tutorial material into a book, suitable as a course unit, at some point.
Outline of Tutorial

A. Introduction and Overview

B. Case Study Demo and Features Tour
   - Financial regulatory/policy compliance

C. Concepts and Foundations
   - Overview level, with selective drill down, on:
     expressive features, semantics, algorithms;
     relationships to natural language and machine learning

D. Conclusions and Future Work
   - Background Assumed: basic knowledge of first-order logic, databases, XML, RDF and semantic web concepts
A KR S is defined as a triple (LA, LC, |=), where:

- LA is a formal language of sets of **assertions** (i.e., premise expressions)
- LC is a formal language of sets of **conclusions** (i.e., conclusion expressions)
- LC is not necessarily even a subset of LA. E.g., in LP and Rulelog.
- |= is the **entailment** relation.

- Conc(A,S) stands for the set of conclusions that are entailed in KR S by a set of premises A

- We assume here that Conc is a functional relation.

- Typically, e.g., in FOL and declarative Logic Programs, entailment is defined formally in terms of **models**, i.e., truth assignments that satisfy the premises and meet other criteria.
Practical Logic, vs. Classical Logic

• Support IT, not mathematics
• Databases
• Rules
• Scalable computationally
• Robust in face of human errors and miscommunications

• Thus: Humble -- avoid general reasoning by cases and general proof by contradiction

What is “reasoning by cases”: (background)
Assertions: if A then C. if B then C. A or B.
Conclude: C.
Main Kinds of Practical Logic

• Databases: relational (SQL), graph (SPARQL, XQuery)
• Production rules, Event-Condition-Action rules, Prolog
  • (subset of their functionality is a subset of LP)
• First Order Logic (Common Logic) subset of classical.
• Description Logic (OWL) is subset of FOL.
• Rulelog (RIF dialect in draft)
• Well-founded declarative logic programs (LP) is a subset of Rulelog. Ditto most RuleML & RIF dialects
• Probabilistic LP is a subset of Rulelog
• Others not so commercially/practically prominent
  • Answer Set Programs, MKNF
    • Related to LP and Rulelog, but closer to classical. Not as scalable. Less robust.
More Practical Logic Context for Rulelog

- Also subsets of LP and thus of Rulelog:
  - Databases
  - Production rules, ECA rules, Prolog (their logical subsets)
  - OWL-RL (Rules profile)

- "Smart Data" is hot in industry:
  - Graph / linked database, with explicit schemas and a bit of semantics. Also as input to machine learning.

- Next step: "Smart Rules", using Rulelog and subsets
  - Rules that chain, deeper reasoning and semantics, meta flexibility with scalability
  - Enterprises leverage investments in smart data

- Keys: semantics, agile schema, meta data (incl. linking), meta knowledge; simplicity, flexibility, reusability
Semantic

• “Semantic” rules/technology/web is a way to describe, i.e., it’s based on logic
• Advantages for communication across systems and organizational boundaries
• Meaning is shared notion of what is/is-not inferrable
• Abstracts away from implementation

• Relational DB was 1st successful semantic tech
• LP theory was invented to formalize it and unify it with the pure subset of Prolog
Overview of Rulelog: Highly Expressive

• Extends LP with strong meta (knowledge and reasoning)
  • Higher-order logic formulas
    • Higher-order syntax via reduction to first-order
    • General formulas: all usual quantifiers/connectives
      • Head existentials via skolemization
      • Head disjunction via “omni-directionality”
  • Defeasible (incl. negation) – flexible approach
  • Probabilistic – flexible approach
  • Restraint bounded rationality via undefined truth value
• Rule ID’s, provenance
• Reification
• External queries
• Frame/object-oriented syntax
Overview of Rulelog (II)

- Computationally scalable, nevertheless
  - Database logic (LP) spirit + bounded rationality
- Has capable efficient algorithms AND implementations
  - Compilation, transformation, indexing, cacheing, dependency-awareness, subgoal reordering
  - Leverages database and “tabling” techniques
- Supports automatic full explanations
- Supersumes expressiveness and closely integrates with: RDF & SPARQL, relational DB & SQL, OWL-RL
Overview of Rulelog (III)

• Closely integrates with OWL-DL

• Closely integrates with natural language processing
  • Text interpretation: map text to logic
  • Text generation: map logic to text

• Closely integrates with machine learning (ML)
  • Import ML results as probabilistic knowledge
  • Export conclusions to ML

→ → practical, easier to build and evolve KB’s
**Example Architecture**

Optional Custom Solutions

---

**Ergo Suite**

- **Ergo Studio**
  - Rule Editor and Query UI
  - (Integrated Development Environment)

- **Ergo Reasoner**
  - Knowledge Base

---

External Info (multi-source)
- Complex Information
  - English Doc.'s etc.
  - Policies, Regulations
  - Financial, Legal, Science
- Data
- Views, Rules
- Schemas & Ontologies
- Results of ML

---

External Services & Frameworks
- Relational DB
- RDF/Graph DB
- Other Sem. Tech
- Machine Learning
- Apps, Docker, ...

---

Users

---

Other Sem. Tech
- App Actions
  - events, decisions

---


© Copyright Coherent Knowledge Systems, LLC, 2015. All Rights Reserved.
Rulelog: Software Tools

❖ Lots of Rulelog expressiveness:
  • Flora-2: Large subset of Rulelog. Open source.
  • Ergo (from Coherent): Most of Rulelog. Has IDE.
    • Available free for research use on case-by-case basis
      (support time by Coherent may cost something, tho’)

❖ Much smaller subsets of Rulelog expressiveness:
  • XSB Prolog: most of LP -- with functions and well-founded negation. Plus a bit more. Open source.
  • Jena: function-free negation-free LP, focused on RDF. Plus a bit more. Open source.
    • Similar: misc. other, e.g., that implement SWRL or SPIN
Example Applications of Rulelog

- Horizontally: policy-based decisions, info integration, analytics, business intelligence, business process flow
- E-commerce: pricing/promotions, contracts, ads, product catalog integration, ordering
- Financial: regulatory/policy compliance, business reporting
- E-Learning: personalized tutoring via explanations
- Security: confidentiality, defense intelligence analysis
- E-science: model causal processes in life/physical sciences
- Health: treatment guidance, insurance
Textual Rulelog (TR) – approach

• Leverage Rulelog to much more simply and closely map between natural language (NL) and logic
• Rulelog’s high expressiveness is much closer to conceptual abstraction level used in NL
• English sentence ↔ Rulelog sentence (rule)
• **Textual terminology:**
  – English phrase ↔ logical term in Rulelog
  – English word ↔ logical functor in Rulelog
  – Basis for textual templates
Textual Rulelog – approach (II)

• TR text interpretation:
  Rulelog rules map from NL to logic

• TR text generation:
  Rulelog rules map from logic to NL

• TR terminology mapping:
  Rulelog rules map between phrasings and ontologies – in NL or logic
  – “moving a bomb” implies “transporting weaponized material”
  – isBomb(\(?x\)) implies rdftriple(\(?x,rdftype,bomb\))
Knowledge Authoring Process using Textual Rulelog

• Start with source text in English – e.g., textbook or policy guide
  • A sentence/statement can be an assertion or a query

• Articulate: create encoding sentences (text) in English.
  As necessary:
  • Clarify & simplify – be prosaic and grammatical, explicit and self-contained
  • State relevant background knowledge – that’s not stated directly in the source text

• Encode: create executable logic statements
  • Each encoding text sentence results in one executable logic statement (“rules”)
  • Use IDE tools and methodology

• Test and debug, iteratively
  • Execute reasoning to answer queries, get explanations, perform other actions
  • Find and enter missing knowledge
  • Find and fix incorrect knowledge
  • Optionally: further optimize reasoning performance, where critical
Knowledge Authoring Steps using Textual Rulelog

Source sentences

**Articulate** *(mainly manual)*

Encoding sentences

**Encode** *(partly automatic)*

Logic statements

**Test** – execute reasoning *(mainly automatic)*

R&D direction: methods to greatly increase the degree of automation in encoding
Actively Reason over Today’s Gamut of Knowledge

![Diagram of database technologies and processing methods]

- **Graph DB & Semantic tech**
- **Relational DB**
- **Machine Learning**
- **Domain Apps & Legacy**
- **Spreadsheets**

Probabilistic engines
\[ p(H|C) = \prod_{i=1..C} \frac{\sum p(x_i|Y_j)}{\sum C \prod p(x_i|Y_j)} \]

"Business Rules"
\[ \text{tier}(X,1) \land \text{supply}(Y,X) \Rightarrow \text{tier}(Y,2) \]

Text & Natural Language processing

**ERGO**

- **KB Libraries**
- **Queries**
- **Assertions**
- **Edits**

Answers, Views, Decisions, Alerts, Explanations

**Application**

© Copyright Coherent Knowledge Systems, LLC, 2015. All Rights Reserved.
Outline of Tutorial

A. Introduction and Overview

B. Case Study Demo and Features Tour
   - Financial regulatory/policy compliance

C. Concepts and Foundations
   - Overview level, with selective drill down, on: expressive features, semantics, algorithms; relationships to natural language and machine learning

D. Conclusions and Future Work
   - Background Assumed: basic knowledge of first-order logic, databases, XML, RDF and semantic web concepts
Case Study: Automated Decision Support for Financial Regulatory/Policy Compliance

Problem: Current methods are expensive and unwieldy, often inaccurate

Solution Approach – using Textual Rulelog software technology:
• Encode regulations and related info as semantic rules and ontologies
• Fully, robustly automate run-time decisions and related querying
• Provide understandable full explanations in English
  • **Proof**: Electronic audit trail, with provenance
• Handles increasing complexity of real-world challenges
  • Data integration, system integration
  • Conflicting policies, special cases, exceptions
  • What-if scenarios to analyze impact of new regulations and policies

Business Benefits – compared to currently deployed methods:
• More Accurate
• More Cost Effective – less labor; subject matter experts in closer loop
• More Agile – faster to update
• More Overall Effectiveness: less exposure to risk of non-compliance
Demo of Ergo Suite for Compliance Automation: US Federal Reserve Regulation W

• EDM Council Financial Industry Consortium Proof of Concept – successful and touted pilot
  – Enterprise Data Management Council (Trade Assoc.)
  – Coherent Knowledge Systems (USA, Technology)
  – SRI International (USA, Technology)
  – Wells Fargo (Financial Services)
  – Governance, Risk and Compliance Technology Centre (Ireland, Technology)

• Reg W regulates and limits $ amount of transactions that can occur between banks and their affiliates. Designed to limit risks to each bank and to financial system.

• Must answer 3 key aspects:
  1. Is the transaction’s counterparty an affiliate of the bank?
  2. Is the transaction contemplated a covered transaction?
  3. Is the amount of the transaction permitted?

Determining Whether Regulation W Applies

Two initial questions need to be answered in determining whether a transaction is subject to Regulation W. The first is whether the transaction is between a bank and an “affiliate” of the bank. The second is whether the transaction is a “covered transaction.”

Affiliate Definition. Regulation W applies to covered transactions between a bank and an affiliate of the bank.

The definition of an affiliate for purposes of Regulation W is set forth in section 223.2. The definition is broad, and includes:

- Any company that controls the bank,
- Any company that is controlled by a company that controls the bank,
- Any company that is controlled, directly or indirectly, by trust or otherwise, by or for the benefit of shareholders who beneficially or otherwise control, directly or indirectly, by trust or otherwise, the bank or any company that controls the bank,
- Any company in which a majority of its directors, trustees, or general partners (or individuals exercising similar functions) constitute a majority of the persons holding any such office with the bank or any company that controls the bank,
- Any company, including a real estate investment trust, that is sponsored and advised on a contractual basis by the bank or an affiliate of the bank,
- Any registered investment company for which the bank or any affiliate of the bank serves as an investment adviser,
- Any unregistered investment fund for which the bank or any affiliate of the bank serves as an investment adviser, if the bank and its affiliates own or control in the aggregate more than 5 percent of any class of voting securities or more than 5 percent of the equity capital of the fund.

The Starting Point - Text of Regulation W
Demo goes here
Executable Assertions: non-fact Rules

/* A company is controlled by another company when the first company is a subsidiary of a subsidiary of the second company. */
@!{rule103b} /* declares rule id */
@@{defeasible} /* indicates the rule can have exceptions */
controlled(by)(?x1,?x2)
:- /* if */
  subsidiary(of)(?x1,?x3) \and
  subsidiary(of)(?x3,?x2).

/* A case of an affiliate is: Any company that is advised on a contractual basis by the bank or an affiliate of the bank. */
@!{rule102b} @@{defeasible}
affiliate(of)(?x1,?x2) :-
  ( advised(by)(?x1,?x2)
  \or
  (affiliate(of)(?x3,?x2) \and advised(by)(?x1,?x3))).
Executable Assertions: Exception Rule

@!{rule104e}
@{‘ready market exemption case for covered transaction'} /* tag for prioritizing */
\neg covered(transaction)(by(?x1))(with(?x2))
   (of(amount(?x3)))(having(id(?Id))) :-
   affiliate(of)(?x2, ?x1) \and
   asset(purchase)(by(?x1))(of(asset(?x6)))(from(?x2))(of(amount(?x3)))
      (having(id(?Id))) \and
   asset(?x6)(has(ready(market))).

/* prioritization info, specified as one tag being higher than another */
\overrides(‘ready market exemption case for covered transaction',
   'general case of covered transaction').

/* If a company is listed on the New York Stock Exchange (NYSE), then the
   common stock of that company has a ready market. */
@!{rule201} @@{defeasible}
asset(common(stock)(of(?Company)))(has(ready(market))) :-
   exchange(listed(company))(?Company)(on('NYSE')).
Executable Assertions: Import of OWL

:- iriprefix fibof = /* declares an abbreviation */
   "http://www.omg.org/spec/FIBO/FIBO-Foundation/20120501/ontology/".

/* Imported OWL knowledge: from Financial Business Industry Ontology (FIBO) */
    rdfs#subClassOf(fibob#BankingAffiliate, fibob#BodyCorporate).
    rdfs#range(fibob#whollyOwnedAndControlledBy, fibob#FormalOrganization).
    owl#disjointWith(edmc#Broad_Based_Index_Credit_Default_Swap_Contract,
                     edmc#Narrow_Based_Index_Credit_Default_Swap_Contract).

/* Ontology Mappings between textual terminology and FIBO OWL vocabulary */
    company(?co) :- fibob#BodyCorporate(?co).
    fibob#whollyOwnedAndControlledBy(?sub,?parent) :- subsidiary(of)(?sub,?parent).

/* Semantics of OWL - specified as general Rulelog axioms */
    ?r(?y) :- rdfs#range(?p,?r), ?p(?x,?y).
    ?p(?x,?y) :- owl#subPropertyOf(?q,?p), ?q(?x,?y).
Outline of Tutorial

A. Introduction and Overview

B. Case Study Demo and Features Tour
   ➢ Financial regulatory/policy compliance

C. Concepts and Foundations
   ➢ Overview level, with selective drill down, on:
     expressive features, semantics, algorithms;
     relationships to natural language and machine learning

D. Conclusions and Future Work
   ➢ Background Assumed: basic knowledge of first-order logic,
     databases, XML, RDF and semantic web concepts
Outline of Part C. Concepts & Foundations

1. Horn LP, with Functions
2. Well-Founded Negation
3. Tabling Algorithms for LP
4. Restraint: semantic bounded rationality
5. Frame syntax (a.k.a. F-Logic), Object Oriented style
7. Rule ID’s
9. General Formulas, Existentials and Skolems, Omni-directional Disjunction
   - Representing Text. Importing full OWL. FOL-Soundness.
10. Probabilistic knowledge and reasoning
11. External Querying
12. Reactiveness
14. Terminology/Ontology Mapping
15. Justification/Explanation
**Horn FOL**

- The Horn subset of FOL is defined relative to clausal form of FOL

- A Horn clause is one in which there is at most one positive literal. It takes one of the two forms:

  1. \( H \lor \neg B_1 \lor \ldots \lor \neg B_m \) . A.k.a. a **definite clause / rule**

- Fact \( H \) . is special case of rule (H ground, \( m=0 \))

  2. \( \neg B_1 \lor \ldots \lor \neg B_m \) . A.k.a. an **integrity constraint**

where \( m \geq 0 \), \( H \) and \( B_i \)’s are atoms. (An atom = \( \text{pred(term}_1,\ldots,\text{term}_k) \) where \( \text{pred} \) has arity \( k \), and functions may appear in the terms.)

- A definite clause (1.) can be written equivalently as an **implication**:

- Rule := \( H \Leftarrow B_1 \land \ldots \land B_m \) . where \( m \geq 0 \), \( H \) and \( B_i \)’s are atoms

  head if body ;

- An integrity constraint (2.) can likewise be written as:

  \( \bot \Leftarrow B_1 \land \ldots \land B_m \) . A.k.a. **empty-head** rule (\( \bot \) is often omitted). For refutation theorem-proving, represent a **negated goal** as (2.).
Horn LP Syntax and Semantics

- **Horn LP syntax** is similar to implication form of Horn FOL
  - The implication connective’s semantics are a bit weaker however. We will write it as $\leftarrow$ (or as :- ) instead of $\Leftarrow$.
  - **Declarative LP with model-theoretic semantics**
    - Same for forward-direction (“derivation” / “bottom-up”) and backward-direction (“query” / “top-down”) inferencing
  - **Model** $M(P) = \text{a set of (concluded) ground atoms}$
    - Where $P = \text{the set of premise rules}$
- **Semantics is defined via the least fixed point of an operator $T_P$.**
  - $T_P$ outputs conclusions that are immediately derivable (through some rule in $P$) from an input set of intermediate conclusions $I_j$.
    - $I_{j+1} = T_P(I_j)$ ; $I_0 = \emptyset$ (empty set)
      - $I_{j+1} = \{\text{all head atoms of rules whose bodies are satisfied by } I_j\}$
    - $M(P) = \text{LeastFixedPoint}(T_P)$ ; where $\text{LFP} = \text{the } I_m \text{ such that } I_{m+1} = I_m$
    - Simple algorithm: $\varphi \square \{\text{run each rule once}\} \clubsuit \spadesuit \blacklozenge \bigcirc \bigcirc$ {quiescence}
Example of Horn LP vs. Horn FOL

Let P be:
- DangerousTo(?x,?y) ← PredatorAnimal(?x) ∧ Human(?y);
- PredatorAnimal(?x) ← Lion(?x);
- Lion(Simba);
- Human(Joey);

I1 = {Lion(Simba), Human(Joey)}
I2 = {PredatorAnimal(Simba), Lion(Simba), Human(Joey)}
I3 = {DangerousTo(Simba,Joey), PredatorAnimal(Simba), Lion(Simba), Human(Joey)}
I4 = I3. Thus M(P) = I3.

Let P’ be the Horn FOL rulebase version of P above, where ← replaces ⇐.
Then the ground atomic conclusions of P’ are exactly those in M(P) above.
P’ also entails various non-ground-atom conclusions, including:
1. Non-unit derived clauses, e.g., DangerousTo(Simba,?y) ← Human(?y).
2. All tautologies of FOL, e.g., Human(?z) ∨ ¬Human(?z).
3. Combinations of (1.) and (2.), e.g., ¬Human(?y) ← ¬DangerousTo(Simba,?y).
Horn LP Compared to Horn FOL

- **Fundamental Theorem connects Horn LP to Horn FOL:**
  - \( M(P) = \{ \text{all ground atoms entailed by } P \text{ in Horn FOL} \} \)

- Horn FOL has additional non-ground-atom conclusions, notably:
  - non-unit derived clauses; tautologies

- Can thus view Horn LP as the **f-weakening** of Horn FOL.
  - “f” here stands for “fact-form conclusions only”
  - A restriction on form of conclusions (not of premises).

- **Horn LP – differences from Horn FOL:**
  - Conclusions \( \text{Conc}(P) = \text{essentially a set of ground atoms.} \)
    - Can extend to permit more complex-form queries/conclusions.
  - Consider Herbrand models only, **in typical formulation and usage.**
    - \( P \) can then be replaced equivalently by \( \{ \text{all ground instantiations of each rule in } P \} \)
    - But can extend to permit: extra unnamed individuals, beyond Herbrand universe
  - Rule has non-empty head, **in typical formulation and usage.**
    - Can extend to detect violation of integrity constraints
The “Spirit” of LP

The following summarizes the “spirit” of how LP differs from FOL:

- **“Avoid Disjunction”**
  - Avoid disjunctions of positive literals as expressions
    - In premises, intermediate conclusions, final conclusions
    - (conclude (A or B)) only if ((conclude A) or (conclude B))
  - Permitting such disjunctions creates exponential blowup
    - In propositional FOL: 3-SAT is NP-hard
    - In the leading proposed approaches that expressively add disjunction to LP with negation, e.g., propositional Answer Set Programs
  - No “reasoning by cases”, therefore

- **“Stay Grounded”**
  - Avoid (irreducibly) non-ground conclusions

LP, unlike FOL, is straightforwardly extensible, therefore, to:

- Nonmonotonicity – defaults, incl. NAF
- Procedural attachments, esp. external actions
Requirements Analysis for Logical Functions

- Function-free is a commonly adopted restriction in practical LP/Web rules today
  - DB query languages: SQL, SPARQL, XQuery
  - RDFS
  - Production rules, and similar Event-Condition-Action rules
  - OWL

- BUT functions are often needed for Web (and other) applications. Uses include:
  - HiLog and reification – higher-order syntax
    - For meta-reasoning, e.g., in knowledge exchange or introspection
      - Ontology mappings, provenance, KB translation/import, multi-agent belief, context
      - KR macros, modals, reasoning control, KB modularization, navigation in KA
      - Meta-data is important on the Web
  - Skolemization – to represent existential quantifiers
    - E.g., RDF blank nodes
  - Convenient naming abstraction, generally
    - steering_wheel(my_car)
Functions in LP Lead to Undecidability; but Restraint Solves this

- Functions lead to undecidability, due to potentially infinite number of conclusions

- Example:
  - Assert: num(succ(?x)) :- num(?x). num(0).
  - Conclusions: num(0), num(succ(0)), num(succ(succ(0))), …

- In Rulelog, restraint bounded rationality solves this
  - Specify radial restraint with radius of 3, for example
  - Then num(succ(succ(succ(succ(0))))), … all have truth value u

- For more info on restraint, see
  - RuleML-2013 paper “Advanced Knowledge Debugging for Rulelog” by C. Andersen et al.
  - Both are available at http://coherentknowledge.com/publications/
Well Founded Semantics for LP

- Uses 3 truth values: \( t = \text{true}, f = \text{weak-negation (naf)}, u = \text{undefined} \)
  - \( f \) intuition: “I know I do not believe it”
  - \( u \) intuition: “I don’t want to figure it out”
    - Original motivation: represent paradoxicality, e.g., \( p : - \text{naf} \ p \).
    - Also used for restraint bounded rationality
- Always exactly one set of conclusions (entailed ground atoms)
- **Tractable** to compute all conclusions, for broad cases:
  - \( O(n^2) \) for Propositional case of Normal LP
  - \( O(n) \) if restricted to naf-free (i.e., Horn)
  - \( O(n^{2v+2}) \) for function-free case (\( v = \max \# \text{variables per rule} \))
  - NAF only moderately increases computational complexity compared to Horn (frequently linear, at worst quadratic)
- By contrast, for Stable Semantics / Answer Set Programs (ASP):
  - There may be zero, or one, or a few, or very many alternative conclusion sets
  - Intractable even for Propositional case


**Tabling Algorithms for LP & Rulelog**

- Builds and maintains a forest of saved subgoal attempts and results
- Thus heavily caches. Is mixed-direction, not just backward-direction.
- Efficient indexing and low level data structures

- Hilog (higher-order syntax) is a challenge, e.g., for indexing

- Nonmonotonicity of naf and defeasibility is a challenge

- **Incremental** tabling adds more dependency-awareness
  - Enables fast updating
  - E.g., for interactive rule authoring edit-test loop

- Highly sophisticated, optimized over last two decades
**Hilog: Higher-Order Syntax**

- Permit predicate or function to be a variable
- Permit predicate or function to be a complex functional term
- Elegant transformation defines the semantics, and is used to implement
- Intuition: \( ?\text{pred}(?\text{arg1}, ?\text{arg2}) \rightarrow \rightarrow \text{believe}(?\text{pred}, ?\text{arg1}, ?\text{arg2}) \)
Probabilistic Knowledge & Reasoning, in Rulelog

- Leverage Hilog and restraint

- Probabilistic knowledge has tuple of parameters
  - Prob(<formula-term>, <parameters>)
  - Flexible in regard to what are the <parameters>:
    - Point value
    - Interval
    - Mean, standard-deviation
    - Interval, confidence-level, sample-size, statistical-technique

- Evidential reasoning: weighted or prioritized combination

- Distribution semantics: semantics/foundation of Probabilistic LP
Outline of Part C. Concepts & Foundations

1. Horn LP, with Functions
2. Well-Founded Negation
3. Tabling Algorithms for LP
4. Restraint: semantic bounded rationality
5. Frame syntax (a.k.a. F-Logic), Object Oriented style
7. Rule ID’s
9. General Formulas, Existentials and Skolems, Omni-directional Disjunction
   ➢ Representing Text. Importing full OWL. FOL-Soundness.
10. Probabilistic knowledge and reasoning
11. External Querying
12. Reactiveness
14. Terminology/Ontology Mapping
15. Justification/Explanation
For more info:
SEE AAAI-13 tutorial Part B


• See its Part B “Concepts and Foundations”
A. Introduction and Overview

B. Case Study Demo and Features Tour
   - Financial regulatory/policy compliance

C. Concepts and Foundations
   - Overview level, with selective drill down, on:
     expressive features, semantics, algorithms;
     relationships to natural language and machine learning

D. Discussion and Future Work
   - Background Assumed: basic knowledge of first-order logic, databases, XML, RDF and semantic web concepts
Rulelog KR: Advantages for Knowledge Management

- Unprecedented flexibility in the kinds of complex info that can be stated as assertions, queries, and conclusions (highly expressive “knowledge” statements)
  - Almost anything you can say in English – concisely and directly
  - Just-in-time introduction of terminology
  - Statements about statements (meta knowledge)
  - State and view info at as fine a grain size as desired

- Probabilistic info combined in principled fashion, tightly combined with logical
  - Tears down the wall between probabilistic and non-probabilistic

- Unprecedented ease in updating knowledge
  - Map between terminologies as needed, including from multiple sources

- Conflict between statements is robustly handled (often arises during integration)
  - Resolved based on priority (e.g., authority), weighting, or else tolerated as an impasse

- Scalable and computationally well-behaved
Open Research Topics in the KR itself (I)

• Reactive: semantics, event handling/dispatching
  • Relate to Reaction RuleML, Prova, production/ECA rules, Transaction Logic

• Probabilistic: distribution semantics, hookups to ML approaches

• Reasoning by cases: theory/semantics, algorithms
  • Soundness/relationship to: FOL, ASP, MKNF

• Hypothetical reasoning, abduction
Open Research Topics in the KR itself (II)

- Equality: axiomatic semantics, efficient algorithms
- Aggregates – handle indefiniteness, unstratified cases
- “Constraints” – cf. constraint LP: theory, algorithms
- Distributed reasoning: algorithms and testbeds
  - Finely parallelized too. Leverage persistent stores.
- Optimizations: e.g., subgoal re-ordering for efficiency
Research Directions – Other Aspects

❖ Applications
  • Text interpretation and generation, NLP and HCI
  • Legal
  • Biomedical
  • In tandem with ML, relationship to induction
  • There are many more

❖ Standards design – with RuleML
  • (In draft): RIF-Rulelog
  • RuleML-Rulelog; relate to Oasis Legal RuleML
  • Profiles (subsets) incl. intersect with OWL
  • Rulelog output from SBVR
Thank You

Disclaimer: The preceding slides represent the views of the authors only. All brands, logos and products are trademarks or registered trademarks of their respective companies.