WELCOME! to the RuleML+RR-2017 Tutorial

Rulelog: Highly Expressive Rules with Deep Scalable Reasoning

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

*   *Accenture ; work in part done while previously at Coherent Knowledge*
http://benjamingrosof.com

** *Stony Brook University, and Coherent Knowledge*
http://www.cs.stonybrook.edu/~kifer
http://www.cs.stonybrook.edu/~pfodor

1

Preface ; For More Info

- Previous tutorials on Rulelog and its forerunners were presented at AAAI-17, IJCAI-16, RuleML-2016, RuleML-2015, AAAI-13, and ~9 other times at conferences since 2004.

- This slideset is, or soon will be, available on the web, at http://benjamingrosof.com/misc-publications/#RuleMLRR2017RulelogTutorial , and perhaps also at the RuleML+RR-2017 website

- For more info beyond the slideset: see the authors' websites and http://coherentknowlege.com/publications
  - References are in the short proceedings paper of this tutorial, also in the paper of the RuleML-2015 tutorial, and in the AAAI-13 tutorial slides near the end

- Hoping to turn the tutorial material into a book, suitable as a course unit, at some point. (Interest from several publishers.)

Bio – Benjamin Grosof

- AI researcher, executive, and entrepreneur
- Research Fellow & Principal Director in AI at Accenture
- Co-founder & Board member of Coherent Knowledge

- Previously:
  - CTO and CEO of Coherent Knowledge – AI KRR software startup
  - Directed advanced AI research program for Paul Allen
    - Developed Rulelog KRR theory, algorithms, UI approach
  - MIT Sloan professor and DARPA PI
    - Co-Founder of RuleML, key contributor to W3C OWL-RL and RIF standards
  - IBM Research, creator IBM Common Rules
    - 1st successful semantic rules product in industry
  - Stanford AI PhD, combining ML with logical and probabilistic reasoning

- http://www.linkedin.com/in/benjamingrosof
- http://benjamingrosof.com

Outline of Tutorial

A. Overview

  ➢ Practical logic. Applications. Features. Software. Textual.

  ➢ Case Study Demo and Features Tour

    ➢ Financial regulatory/policy compliance

B. Concepts and Foundations: Drill-downs

  ➢ Expressive features, semantics, algorithms; relationships to natural language and machine learning

C. Conclusions and Future Work

– Background Assumed: basic knowledge of first-order logic, databases, rules. Helpful: RDF and semantic web concepts

Practical Logic, vs. Classical Logic

- Goal: support IT, vs. mathematics
  - E.g., Databases, Rules
  - Central: declarative logic programs (LP) KR

- Requirements:
  - Scalable computationally
  - Robust in face of human errors and miscommunications
  - $\rightarrow\rightarrow$ "Humble"
    - Avoid general reasoning by cases
    - Avoid general proof by contradiction

What is "reasoning by cases":   (background)
Assertions: if A then C. if B then C. A <u>or</u> B.
Conclude: C.

Semantic

- "Semantic" rules/tech/web means: 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

Concept of logical Knowledge Representation (KR)

A given KR logical system S has …

1. Formal language L_S for <u>assertions</u> and <u>conclusions</u>
   - Assertions LA_S and conclusions LC_S may be different!!
     - E.g., in LP and Rulelog

2. Semantics: <u>entailment</u> relation $\models_S$
   - An assertion set *A* entails a* conclusion set *C*
     - * We assume here exactly 1
   - Typically, entailment is defined formally in terms of <u>models</u>
     - Truth assignments on LC_S that meet criteria based on the assertions *A*
       - E.g., in FOL and LP and Rulelog

Reasoning implements the semantics, e.g., to answer queries
   - KR<u>R</u> <u>software</u> system: knowledge representation & <u>r</u>easoning
   - Typically KRR systems are sound but often incomplete

Industry Landscape of Practical Logic

- Rulelog extends LP. LP (well-founded) is a subset of Rulelog.
- ***LP is the core KR of structured knowledge management today***
- Subsets of LP and thus of Rulelog:
  - Relational databases (SQL)   *[Datalog subset of LP]*
  - Graph databases, a.k.a. knowledge graphs (SPARQL, XQuery)   *[Datalog]*
  - Production rules, Event-Condition-Action rules. More precisely: their logical subsets.
  - Prolog. More precisely: its "pure" logical subset.
  - Many RuleML & RIF dialects, e.g., RIF-BLD, RIF-Core, SWRL
  - Many ontology standards, e.g., OWL-RL, RDF-Schema   *[Datalog]*
- Other:
- Subsets of Classical Logic:
  - Propositional. E.g., hardware circuit design, satisfiability for planning.
  - First Order Logic (Common Logic)
  - Description Logic (OWL) subset of FOL. For ontologies.
- Emerging:
  - Larger subsets of Rulelog (RIF-Rulelog dialect in draft)
    - Covers: most RuleML & RIF dialects; Probabilistic LP (under various names).
  - Others not so commercially/practically prominent
    - Answer Set Programs, MKNF. Related to LP & Rulelog, but closer to classical, less humble.

Outline of Tutorial

A. Overview

  ➢ Practical logic. Applications. Features. Software. Textual.

  ➢ Case Study Demo and Features Tour

    ➢ Financial regulatory/policy compliance

B. Concepts and Foundations: Drill-downs

  ➢ Expressive features, semantics, algorithms; relationships to natural language and machine learning

C. Conclusions and Future Work

– Background Assumed: basic knowledge of first-order logic, databases. Helpful: XML, RDF and semantic web concepts

Overview of Rulelog (I)

- A major research advance in KRR theory & algorithms
  - Culminated in 2012. Contributions from many researchers since 1990's.

- Very high/flexible expressiveness
  - Overall: extends LP with strong meta (knowledge and reasoning)
  - Higher-order logic formulas
    - Higher-order syntax via reduction to first-order. Reification.
    - General formulas: all usual quantifiers/connectives
      - Head existentials via skolemization
      - Head disjunction via "omni-directionality"
  - Defeasibility/exceptions (incl. negation) – flexible approach
  - Probabilistic – flexible approach
  - Restraint bounded rationality via *undefined* truth value
  - Rule ID's, provenance
  - External queries
  - Frame/object-oriented syntax
- Strongly supports federation & orchestration of KRR
  - Via external queries and high expressiveness

Overview of Rulelog (II)

- Computationally scalable, despite very high expressiveness
    - Database logic (LP) spirit + bounded rationality
    - Reasoning is polynomial time* (as in databases)

- Has capable efficient algorithms AND implementations
    - Dynamic compilation/transformation stack architecture
    - Cacheing of successful & failed inferences, with dependency-awareness, subgoal reordering, analysis of cycles and depths
    - Indexing, tries, other low-level data structures
    - Leverages database and "tabling" techniques

- Supports automatic full explanations

- Supersumes expressiveness and closely integrates with: RDF & SPARQL, relational DB & SQL, OWL-RL

* if, as is typical, one employs the radial restraint feature

Overview of Rulelog (III)

- Closely integrates with OWL-DL ontologies

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

- Complements and integrates with machine learning (ML)
  - Import ML results as probabilistic knowledge
  - Export conclusions to ML

→ → *practical, easier to build and evolve KB's*

Uses of Rulelog – Overview

- Good for complex, commonly-arising kinds of knowledge, combined with simpler kinds of info
  - Mappings between different terminologies, ontologies, or schemas
  - Policies
  - Legal: regulations, contracts
  - Causal pathways, e.g., in science or biz processes

- Use for decision automation, question-answering, and other analytics, esp. involving
  - Deep reasoning
  - Integration of diverse info sources and types

Outline of Tutorial

A. Overview

  ➢ Practical logic. Applications. Features. Software. Textual.

  ➢ Case Study Demo and Features Tour

    ➢ Financial regulatory/policy compliance

B. Concepts and Foundations: Drill-downs

  ➢ Expressive features, semantics, algorithms; relationships to natural language and machine learning

C. Conclusions and Future Work

– Background Assumed: basic knowledge of first-order logic, databases. Helpful: 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 Transparent – better explanations
- 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 Rulelog 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 <u>affiliate</u> of the bank?*

2. *Is the transaction contemplated a <u>covered transaction</u>?*

3. *Is the amount of the transaction <u>permitted</u> ?*

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."

<u>Affiliate Definition.</u> 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:
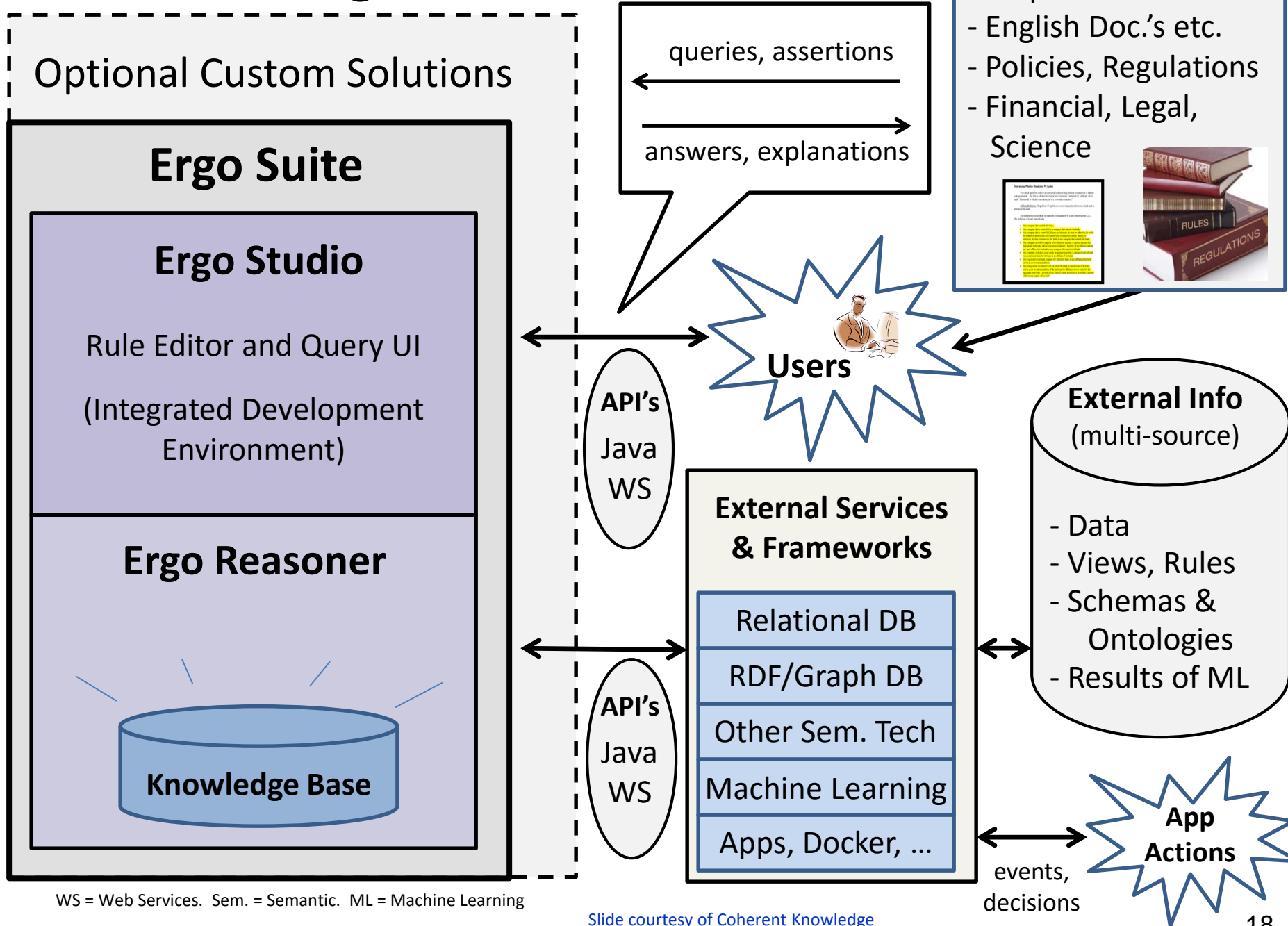
- Any company that controls the bank;
- Any company that is controlled by a company that controls the bank;
- Any company that is controlled, directly or indirectly, by trust or otherwise, by or for the benefit of shareholders who beneficially or otherwise control, directly or indirectly, by trust or 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[1];

The Starting Point - Text of Regulation W

Reg W Demo Drill-down – outline

- Uses Ergo Suite implementation of Rulelog

- Video available at
  http://coherentknowledge.com

- We show here: screenshots, example KB rules

Ergo Architecture

Optional Custom Solutions

Ergo Suite

Ergo Studio

Rule Editor and Query UI

(Integrated Development Environment)

Ergo Reasoner

Knowledge Base

queries, assertions

answers, explanations

Users

Complex Information
- English Doc.'s etc.
- Policies, Regulations
- Financial, Legal, Science

API's
Java
WS

API's
Java
WS

External Services & Frameworks

Relational DB

RDF/Graph DB

Other Sem. Tech

Machine Learning

Apps, Docker, ...

External Info
(multi-source)

- Data
- Views, Rules
- Schemas & Ontologies
- Results of ML

App Actions

events, decisions

WS = Web Services. Sem. = Semantic. ML = Machine Learning

Slide courtesy of Coherent Knowledge

18

Query is asked in English

19

User Clicks the handles to expand the Explanations

Why 'What proposed transactions are prohibited by RegW? Show '('Pacific Bank','Maui Sunset',23.0) ?
Edit Operations Windows

- 'What proposed transactions are prohibited by RegW? Show '('Pacific Bank','Maui Sunset',23.0)
  - RegW prohibits the proposed transaction by Pacific Bank with Maui Sunset of $23.0 million

Why 'What proposed transactions are prohibited by RegW? Show '('Pacific Bank','Maui Sunset',23.0) ?
Edit Operations Windows

- 'What proposed transactions are prohibited by RegW? Show '('Pacific Bank','Maui Sunset',23.0)
  - RegW prohibits the proposed transaction by Pacific Bank with Maui Sunset of $23.0 million
    - The proposed transaction by Pacific Bank with Maui Sunset of $23.0 million is a RegW covered transaction
    - There is a limit of $10.0 million for any proposed RegW covered transaction by Pacific Bank with Maui Sunset
    - The proposed transaction of $23.0 million is greater than the RegW limit of $10.0 million

Why 'What proposed transactions are prohibited by RegW? Show '('Pacific Bank','Maui Sunset',23.0) ?
Edit Operations Windows

- 'What proposed transactions are prohibited by RegW? Show '('Pacific Bank','Maui Sunset',23.0)
  - RegW prohibits the proposed transaction by Pacific Bank with Maui Sunset of $23.0 million
    - The proposed transaction by Pacific Bank with Maui Sunset of $23.0 million is a RegW covered transaction
      - Maui Sunset is a RegW affiliate of Pacific Bank
      - There is a proposed loan from Pacific Bank to Maui Sunset of $23.0 million
    - There is a limit of $10.0 million for any proposed RegW covered transaction by Pacific Bank with Maui Sunset
    - The proposed transaction of $23.0 million is greater than the RegW limit of $10.0 million

Slide courtesy of:

Coherent Knowledge

20

Why is the proposed transaction prohibited by Regulation W?

1. *Is the transaction's counterparty an "<u>affiliate</u>" of the bank?*

YES.

And here's why …

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. Overview

➢ Practical logic. Applications. Features. Software. Textual.

➢ Case Study Demo and Features Tour

  ➢ Financial regulatory/policy compliance

B. Concepts and Foundations: Drill-downs

➢ Expressive features, semantics, algorithms; relationships to natural language and machine learning

C. Conclusions and Future Work

– Background Assumed: basic knowledge of first-order logic, databases. Helpful: XML, RDF and semantic web concepts

Series of Advances → Rulelog's Core Expressive Features

- Well-founded semantics; basic tabling algorithms
  - *Undefined* for paradox; smart cacheing; intuitionistic disjunction
- Higher-order syntax (Hilog); frame syntax
  - Associated optimizations of LP tabling etc. algorithms
- Statement id's for meta; argumentation meta-rules for defeasibility; provenance
- General formulas with all usual classical connectives and quantifiers (omniformity)
- Restraint bounded rationality
  - Use 3rd truth value *undefined* for "don't-care"
  - Radial, skipping; naf unsafety; external-query unsafety, unreturn

Rulelog: Software Tools

❖ Lots of Rulelog expressiveness:

- Ergo Lite: Large subset of Rulelog. Open source.

  - A.k.a. as Flora-2, originally

- Ergo (from Coherent): Most of Rulelog. Has IDE.

  - **The most complete & highly optimized implementation available.** Free for academic research use. Free trials available for everyone. (Support time may cost, 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

Ergo Suite: Reasoner, Studio, Connectors

- Ergo Reasoner has sophisticated algorithms & data structures
  - Smart cacheing with dependency-aware updating. Leverages LP & DBMS techniques.
  - Transformation, compilation, reordering, indexing, modularization, dependency/loop analysis, performance monitoring/analysis, pausing, virtual machine, programming kernel, external import/querying
  - Java API. Other interfaces: command line, web, C.
  - Scales well: Millions of sentences on 1 processor; Trillions on distributed nodes

- Ergo Studio is a graphical Integrated Development Environment
  - Interactive editing, querying, explanation, visualization of knowledge
  - Fast edit-test loop with award-winning advanced knowledge debugging/monitoring

- Ergo Connectors federate knowledge & reasoning
  - Import/query dynamically via: SPARQL, OWL, RDF; SQL; CSV; JSON; and more
  - Federation distributes reasoning (i.e., its processing) across multiple nodes

- Open, standards-based approach; a portion is open source
  - Rulelog is draft industry standard from RuleML (submission to W3C & Oasis)

KRR Features Comparison: Rulelog Shines

Feature \ System	Rulelog Rules - e.g., Ergo	Datalog Rules - e.g., Jena, SWRL, Ontobroker, SPIN	Production Rules - e.g., IBM, Oracle, Red Hat	Prolog - e.g., SWI, SICStus, XSB	FOL & OWL-DL - e.g., Vampire, Pellet, Prover9	ASP Solvers - e.g., DLV, CLASP
Semantic & on standardization path	✓	✓	restricted case	restricted case	✓	✓
Basic expressiveness						
• Datalog LP	✓	✓	✓	✓	✓	✓
• Logical functions	✓	✗	✗	✓	✓	restricted
• Quantified formulas (genl.)	✓	✗	✗	✗	✓	✗
Full Meta expressiveness						
• Higher-order syntax, provenance	✓	✗	✗	✗ (except XSB a little)	✗	✗
• Defeasibility & well founded negation	✓	✗	✗	✗	✗	some have restricted
• Restraint bounded rationality	✓	✗	✗	✗	✗	✗
• Probabilistic	✓	✗	✗	✗	✗	✗
Efficiency						
• Goal-directed	✓	✗ (except Jena)	✗	✓	✓	✓
• Full LP tabling with dependency-aware updating	✓	✗	✗	✗ (except XSB)	✗	✗
• Polynomial time complexity	✓	✓	✓	✗	✗	✗

Notes on KRR Features Comparison

- "System" means system type / approach of logical knowledge representation and reasoning (KRR).
- "Semantic" means in the sense of KRR, i.e., fully declarative and having a model theory in the logical sense.
- "FOL" means First Order Logic. "ASP" means Answer Set Programs.
  - ASP is recently emerging. The tasks for which it's suitable are more similar to FOL than to the other systems here.
- "Standardization" here means industry standardization. "On path to" means in process of being, or already, standardized.
- "Restricted case" means for a syntactic/expressive subset.
- Event-condition-action rules in this context are similar to, and lumped in with, production rules.
- "LP" means declarative logic programs.
- Datalog means LP without logical functions. Usually this is restricted to Horn. But here we permit negation(-as-failure).
- OWL-RL is pretty much a restricted case of Datalog LP.
- "Higher-order syntax" means Hilog, which enables probabilistic – and also 1) fuzzy and 2) frame syntax cf. F-Logic.
- "Provenance" means provenance info about assertions, via properties of rule id's that are within the logical language / KRR.
- "Full" applies to all four of the meta expressiveness features.
- Defeasibility includes flexible argumentation theories.
- "General formulas" means classical-logic-like formulas, including with head existentials and with head disjunction.
- "LP tabling" includes sophisticated: cacheing of intermediate reasoning results, inference control, and indexing.
- "Dependency-aware updating" means that when assertions are added or deleted, saved inferences are only recomputed if they depend on the changes to the assertions.
- Polynomial time "complexity" means worst-case computational complexity, with constant-bounded number of variables per rule. Polynomial-time is similar to database querying, and is a.k.a. "tractable".

- Datalog X defeasibility: Ontobroker has full well founded negation.
- Prolog X defeasibility: XSB has full well founded negation.
- ASP X defeasibility: Some ASP systems have restricted defeasibility & well founded negation. ASP systems essentially have wf negation inside (i.e., as part of) their semantics/reasoning, and some ASP systems even expose it to the user.
- Datalog X goal-directed: Jena has a backward engine as well as a forward engine.
- ASP X general formulas: ASP has head disjunction.
- FOL X full LP tabling with dependency-aware updating: Some FOL theorem-provers cache intermediate results in a way that is analogous to LP tabling, and some do dependency tracking but we're not sure how analogous or sophisticated.
- Prolog X higher-order syntax: XSB has some support for this (i.e., for Hilog), although it is not integrated well.

Concept: Virtual Data Stores

- Rulelog orchestrates overall federated reasoning
  by sub-goaling dynamically

- A variety of other ("external") structured information systems are treated as virtual data/knowledge stores,
  via Rulelog federation connectors, which import/query and translate

- Each virtual data/knowledge store is treated as having expressiveness that is a subset of Rulelog. E.g.,

  – An external database fact is treated as a logical atom in Rulelog

  – An OWL axiom is treated as a fact but also supplemented by semantic axioms about OWL's constructs in general

Kinds of Virtual Data in Ergo

- Graph databases: via SPARQL/RDF connector
  - Description logic ontologies: via OWL connector
- Relational databases: via SQL connector
- Spreadsheets and web logs: via DSV connector
- JSON connector; XML connector; Web services via those
- Extensible to almost any kind of (semi-)structured info
  - E.g., Machine Learning (ML) and NLP systems
    - Represent prob(content_sentence, lower_bound, upper_bound, confidence_level, statistical_procedure)   as an Ergo sentence
  - E.g., legacy applications in Java
    - Get method is treated like a query

• Importing RDF & OWL knowledge into Ergo

• Screenshot of Ergo OWL connector part of Ergo Studio

Translates RDF & OWL to Ergo
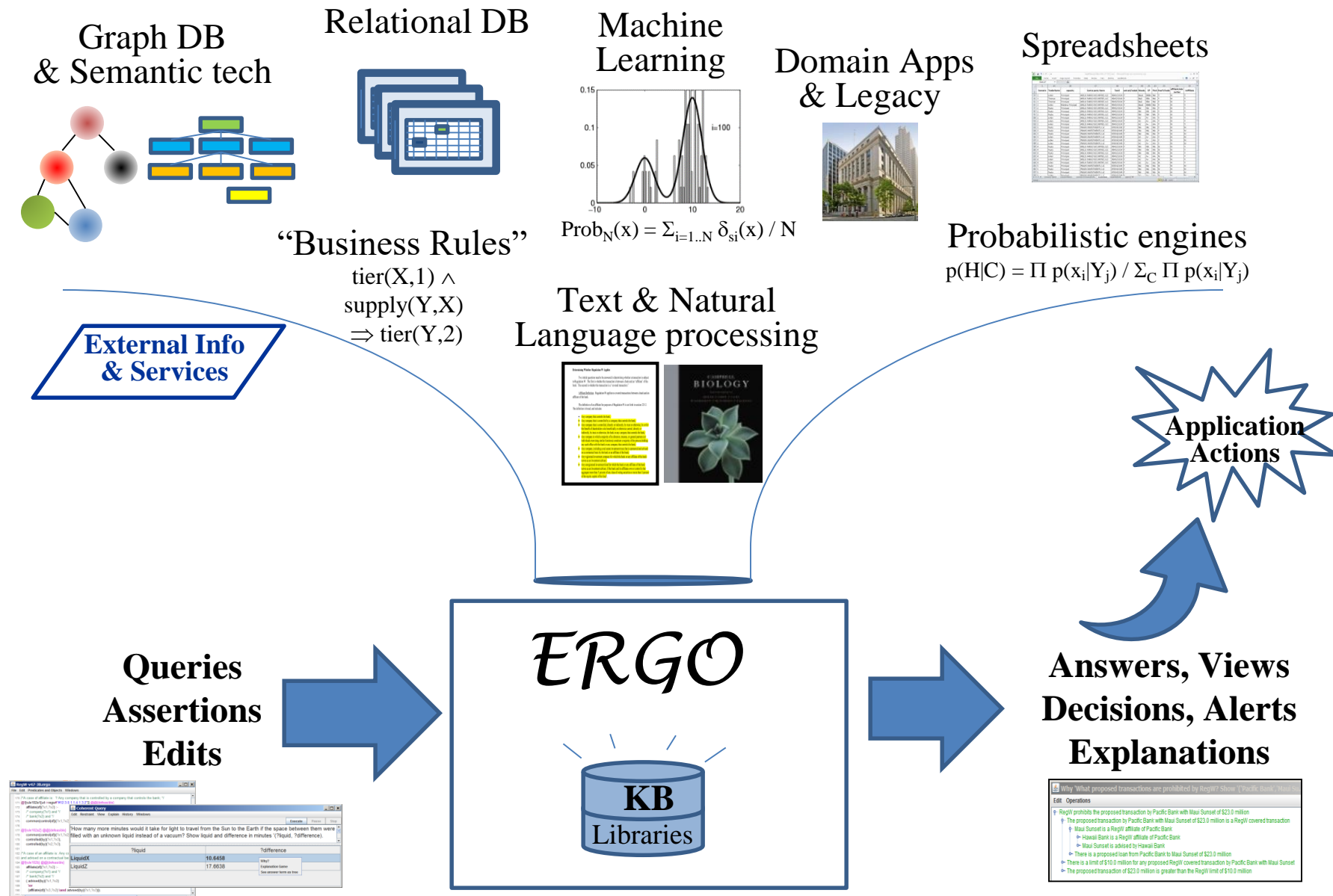
Define IRIs in Ergo Studio
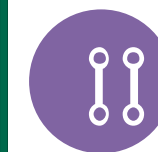
N-triples and N-quads

RDF/OWL XML, JSON-LD, or Turtle as input. Predicate or Frame syntax output.

33

Actively Reason over Today's Gamut of Knowledge

Graph DB & Semantic tech

Relational DB

Machine Learning

Domain Apps & Legacy

Spreadsheets

$Prob_N(x) = \Sigma_{i=1..N}\, \delta_{si}(x) / N$

"Business Rules"

$tier(X,1) \wedge$
$supply(Y,X)$
$\Rightarrow tier(Y,2)$

Probabilistic engines

$p(H|C) = \Pi\, p(x_i|Y_j) / \Sigma_C\, \Pi\, p(x_i|Y_j)$

Text & Natural Language processing

External Info & Services

Application Actions

Queries Assertions Edits

ERGO

KB
Libraries

Answers, Views Decisions, Alerts Explanations

Slide courtesy of Coherent Knowledge

Outline of Tutorial

A. Overview

- ➤ Practical logic. Applications. Features. Software. Textual.
- ➤ Case Study Demo and Features Tour
  - ➤ Financial regulatory/policy compliance

B. Concepts and Foundations: Drill-downs

- ➤ Expressive features, semantics, algorithms; relationships to natural language and machine learning

C. Conclusions and Future Work

– Background Assumed: basic knowledge of first-order logic, databases. Helpful: XML, RDF and semantic web concepts

Example Application Areas for Rulelog

- Confidentiality policies
- Financial/business reporting
- Contracts
- E-commerce pricing/promotion policies
- E-commerce product catalog integration, supply chain
- Financial regulatory/policy compliance
- Health treatment guidance, insurance
- Defense intelligence analysis
- Education/e-learning: personalized tutoring in sciences
- Info/system integration, e.g., in financial, defense

- Potentially many more: natural language interaction, business intelligence, games, workflow, social media,…

Problem: Analytics for *Complex* Knowledge

Examples: policies, regulations, contracts; terminology mappings; science, causality

Existing *Non-Semantic* Technologies tend to be:

- Shallow
- Siloed
- Costly, and Slow
- Patchily automated
- Opaque
- Inaccurate
- End users not empowered to modify

Based on:
- *Conventional programming languages*
- *Production/ECA rules*
- *Prolog*

Benefits of Semantic Approach to analytics & decision automation

- Modeling, declaratively, rather than programming
- First steps – state of art:
  - Decision Tables (cf. DMN)
  - Ontologies (cf. OWL)
- Benefits:
  - Greater integration and reusability
  - More transparent, i.e., explainable
  - Easier to modify, end users* more empowered
  - More cost-effective and agile

* esp. subject matter experts (SMEs)

Rulelog is a Next Step on Semantic

- Compared to decision tables:
  - Deeper in reasoning & knowledge
    - Support many-step inferencing
    - Model complex sentences with high fidelity,
      via high expressiveness, e.g., higher-order, existentials
    - Map to/from natural language
    - Map between ontologies, schemas, terminologies
    - Principled defeasibility (exceptions)
  - Fuller, more understandable explanations
  - Greater scope of automation

  ➔ Extends the benefits of the semantic approach

Outline of Tutorial

A. Overview

- Practical logic. Applications. Features. Software. Textual.

- Case Study Demo and Features Tour

  - Financial regulatory/policy compliance

B. Concepts and Foundations: Drill-downs

- Expressive features, semantics, algorithms;
  relationships to natural language and machine learning

C. Conclusions and Future Work

- Background Assumed: basic knowledge of first-order logic, databases. Helpful: XML, RDF and semantic web concepts

Concept of *Humagic* Knowledge

- Humagic = <u>hu</u>man-<u>ma</u>chine lo<u>gic</u>
- A humagic KB consists of a set of linked sentences
  - Assertions, queries, conclusions (answers & explanations)
- NL-syntax sentence may have 1 or more logic-syntax sentences associated with it
  - E.g., that encode it, or give its provenance, or represent its text interpretation
- Logic-syntax sentence may have 1 or more NL-syntax sentences associated with it
  - E.g., results of text generation on it
  - E.g., source sentence in text interpretation, that produced it
- Other sentences can be in a mix of NL-syntax and logic-syntax
  - Using textual templates, for text interpretation and text generation

Textual extension of Rulelog

- High-level concept of approach: *Textual* Rulelog (TR)
- Extends Rulelog with natural language processing (NLP)
  - Start with English as the NL
- Rulelog logic itself is utilized to map:

  Rulelog logic syntax ← → NL syntax
  - I.e., use logic to help do: *text interpretation*   and   *text generation*
- Mapping is much simpler and closer than with other KR's
  - Rulelog's high expressiveness is much closer to NL's conceptual abstraction level
  - More often doable and useful:

    1 English sentence   ↔   1 Rulelog sentence (rule)
- In principle, almost any NL sentence can be represented with deep semantics as a logical sentence in Rulelog
  - Leverage the general quantified formulas expressive feature of Rulelog

Some Key Tasks in Textual Rulelog

- 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)

Some Techniques for Textual Rulelog

1. Word as functor (WAF): treat a NL word as a logical functor

2. Phrasal terms (phrasts): treat a NL phrase as a logical term

3. Paraphrase knowledge (PAK): e.g., synonyms, hypernyms, hyponyms, equivalent named entities; other one-directional implications between phrases

4. Textual templates (TET): hybrid of text syntax and logic syntax

5. Quantification of NL determiners (QUD): e.g., treat "every" and "some" as relativized universal and existential logical quantifiers

6. Deep extended NL parsing (DEP): logically represent dependency parse tree extended with coreference analysis and named entity recognition

- *These can be combined*
  - *Many interesting directions & open areas for research! E.g., DEP and QUD.*

Motivation for Textual Templates (I)

The knowledge acquisition problem:

Domain expert

Knowledge engineer

Can't communicate effectively

Cannot represent knowledge formally, can't code
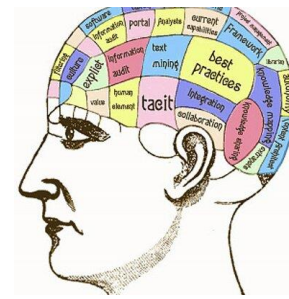
Doesn't have domain knowledge

Motivation for Textual Templates (II)

Typical knowledge acquisition flow:

domain knowledge

knowledge base rules

questions, clarifications
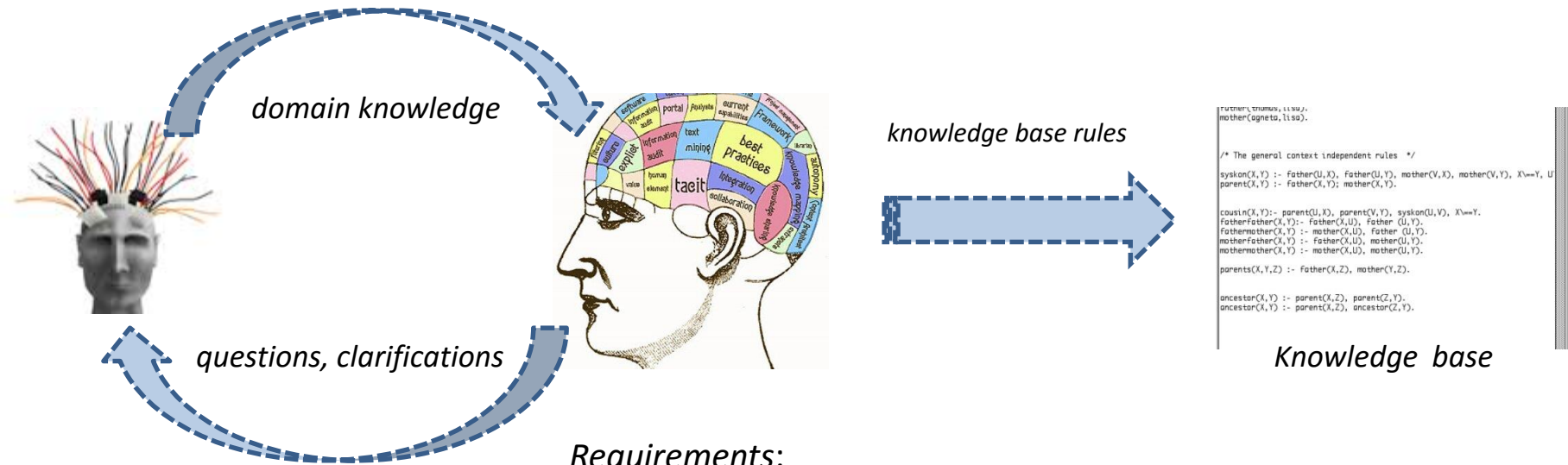
Knowledge base

Requirements:
- <u>high-quality</u> knowledge (facts, rules),
- <u>reliable</u>, <u>justifiable</u> inferences.

Domain knowledge can be communicated in:
- ~~English?~~
- ~~Machine learning?~~
- **Textual Templates**

Textual Templates Example in *Ergo* (I)

\(?X is a father of ?Y\) : −
  \(?X is a parent of ?Y and a male\).

\(if ?X is a penguin then it is a bird\).
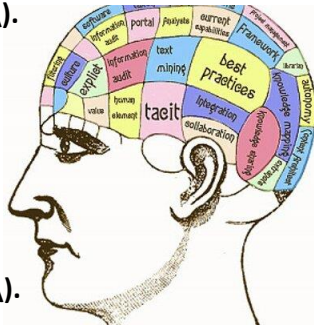
template(\(?X is a father of ?Y\), father(?X,?Y)).
template(\(?X is a parent of ?Y and a male\),
  (parent(?X,?Y), ?X:Male)).

template(\(if ?X is a penguin then it is a bird\),
  (bird(?X) : − penguin(?X))).

- The *domain expert* operates with a corpus of stylized English sentences, which s/he
  - designs and communicates to the knowledge engineer
  - plus some elementary logic (and, or, if, variables)

- The *knowledge engineer* translates into *Ergo* using the *ErgoText* templates feature.

- Any <u>inferences can be explained</u> using the <u>phrases of the</u> aforesaid <u>English corpus</u>.

Textual Templates Example in Ergo (II)

- **KB.ergo** – the actual KB

:- ergotext{KBtempl}.

\(?X is a father of ?Y\) :-

    \(?X is a parent of ?Y and a male\).

\(if ?X is a penguin then it is a bird\).

- **KBtempl.ergotxt** – templates

template(headbody,
       \(?X is a father of ?Y\),
       father(?X,?Y)).
template(headbody,
       \(?X is a parent of ?Y and a male\),
       (parent(?X,?Y), ?X:Male)).
template(rule,
       \(if ?X is a penguin then it is a bird\),
       (bird(?X) :– penguin(?X))).

Reg W Example Sentence using Templates

- Example of hybrid-syntax sentence – <u>executable</u> in Ergo:

  \(The individual affiliate threshold for transaction under Regulation W

  by ?Bank with ?Counterparty is ?Amount\) :-

    \(?Counterparty is deemed an affiliate of ?Bank under Regulation W\) \and

    \(?Bank has capital stock and surplus ?Capital\) \and

    \(the threshold percentage for an individual affiliate is ?Percentage\) \and

    ?Amount \is ?Capital * ?Percentage/100.

Reg W example Template definition in Ergo

template(headbody,
  \(The proposed transaction ?Id by ?Bank with ?Affiliate of $?Amount
   is a RegW covered transaction\),

   covered(proposed(transaction))(by(?Bank))(with(?Affiliate))
    (of(amount(?Amount)))(having(id(?Id)))
).

- The templates are self-documenting

Quantification of Determiners

- Determiners in NL often indicate quantification
  - E.g., treat "*each*" via a logical relativized *universal* quantifier
    - forall(?var)^((?var \isa *fooNounPhrase*)) *==> barVerbPhraseAbout?Var*)
  - E.g., treat "*some*" via a logical relativized *existential* quantifier
    - exists(?var)^((?var \isa *fooNounPhrase*)) \and *barVerbPhraseAbout?Var*)

- Ex.: "each large company has some talented CEO"

  exists(?y)^((?y \isa \(talented CEO\)) \and

  (\?x has ?y\))

  :- ?x \isa \(large company\) .

  /* above has implicit outermost universal quantification of ?x */

Deep Extended NL Parsing (DEP) in TR

- Common useful form of NLP output is "parse++" in following form:
  - NL dependency parsing,
  - extended with coreference analysis (Coref) and named entity recognition (NER)
- DEP: Represent the results of parse++ as a set of logical facts specifying the parse++ tree in full detail
  - Tree structure: dependency edges, left vs. right sequencing, edge labels
  - Node labels: word token, part of speech (PoS), NER, Coref, other word sense
  - Provides grist for deep semantic text interpretation & representation in Rulelog

- ErgoNLP is a recently released open source tool implementing DEP
  - It uses the popular Stanford CoreNLP toolset, which is open source (GPL)
  - Inputs a passage of 1 or more English sentences
  - Outputs Ergo facts that represent the parse++ of each sentence
    - Actually uses only the Ergo Lite subset of Ergo's syntax & expressiveness
  - https://bitbucket.org/coherentknowledge/ergonlp
  - Implemented in Java. Original authors Coherent Knowledge. Apache 2.0 license.

Example of DEP in ErgoNLP

// Input Sentence 1: Each large company has some talented CEO.

```
ph(104)[ ws(wd)->'has',
 root -> \true,
 ws(PoS)->'VBZ',
 ldp(1)->dp(nsubj,ph(103)),
 rdp(1)->dp(dobj,ph(107))].


ph(103)[ ws(wd)->'company',
 ws(PoS)->'NN',
 ldp(1)->dp(det,ph(101)),
 ldp(2)->dp(amod,ph(102))].


ph(107)[ ws(wd)->'CEO',
 ws(PoS)->'NN',
 ldp(1)->dp(det,ph(105)),
 ldp(2)->dp(amod,ph(106)),
 rdp(1)->dp(rcmod,ph(109))].


ph(101)[ ws(wd)->'Each',
 ws(PoS)->'DT'].


ph(102)[ ws(wd)->'large',
 ws(PoS)->'JJ'].


ph(105)[ ws(wd)->'some',
 ws(PoS)->'DT'].


ph(106)[ ws(wd)->'talented',
 ws(PoS)->'JJ'].
```

// Input Sentence 2: IBM is a huge company.

```
ph(205)[ ws(wd)->'company',
 root -> \true,
 ws(PoS)->'NN',
 ldp(1)->dp(nsubj,ph(201)),
 ldp(2)->dp(cop,ph(202)),
 ldp(3)->dp(det,ph(203)),
 ldp(4)->dp(amod,ph(204)),
 ws(coref)->ph(201)].


ph(201)[ ws(wd)->'IBM',
 ws(PoS)->'NNP',
 ws(ne)->ORGANIZATION].


ph(202)[ ws(wd)->'is',
 ws(PoS)->'VBZ'].


ph(203)[ ws(wd)->'a',
 ws(PoS)->'DT'].


ph(204)[ ws(wd)->'huge',
 ws(PoS)->'JJ'].
```

The example's input text passage is the 2 English sentences.

There's one frame set of facts for each word.

There's one set of frame sets for each English sentence.

In Ergo frame syntax:

   subject[property->value]

is a fact triple that is similar to

   property(subject,value)

in predicate calculus syntax; and

   subj[prop1->val1, prop2->val2, prop3->val3]

is logically equivalent to the 3 fact triples

   subj[prop1->val1] \and subj[prop2->val2] \and subj[prop3->val3].

ph(104) stands for phrasal term number 104. Each such phrast represents a <u>node</u> in the parse++ tree, and also corresponds to a parse++ <u>subtree</u> rooted at that node.

ws stands for word sense info.

ws(wd) stands for the word token.

ws(PoS) stands for part of speech.

ldp stands for left dependency list.

rdp stands for right dependency list.

ldp(1) stands for first left dependency, as one goes left-to-right; ldp(2) stands for the second, etc.
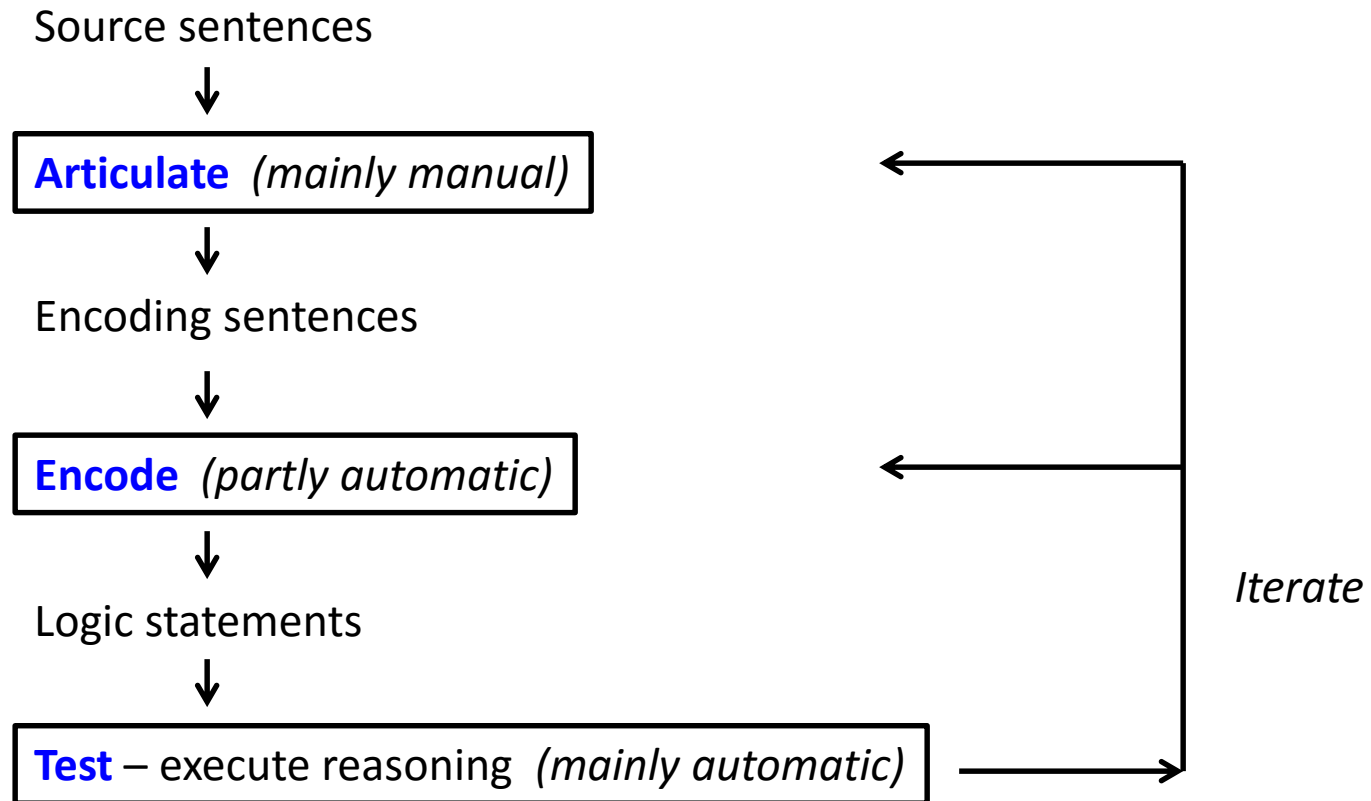
rdp(1) stands for first right dependency, as one goes left-to-right; rdp(2) stands for the second, etc.

dp(*dependency_label,* ph(*number*)) stands for a dependency edge, with a particular edge label, to a particular node.

The dependency edge labels, and PoS node labels, are the usual Penn TreeBank ones. E.g., nsubj for subject NP, JJ for adjective.

Knowledge Authoring Steps using Textual Rulelog

Source sentences

↓

| **Articulate** *(mainly manual)* |

↓

Encoding sentences

↓

| **Encode** *(partly automatic)* |

↓

Logic statements

↓

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

Iterate

R&D direction: methods to greatly increase the degree of automation in encoding

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 <u>encoding</u> 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

Outline of Tutorial

A. Overview

  ➢ Practical logic. Applications. Features. Software. Textual.

  ➢ Case Study Demo and Features Tour

   ➢ Financial regulatory/policy compliance

B. Concepts and Foundations: Drill-downs

  ➢ Expressive features, semantics, algorithms;
      relationships to natural language and machine learning

C. Conclusions and Future Work

– Background Assumed: basic knowledge of first-order logic,
    databases. Helpful: XML, RDF and semantic web concepts

Outline of Part B. Concepts & Foundations

- Spirit of LP
- Higher-Order Syntax via Hilog. Reification.
- Rule ID's
- Default Negation and Well Founded Semantics
- Restraint: semantic bounded rationality
- Tabling Algorithms. Knowledge Debugging Methods.
- Defeasibility via Argumentation Rules. Remedying FOL's Fragility.
- General Formulas, Existentials and Skolems, Omni-directional Disjunction
  - Representing Text. Importing full OWL.
- Uncertain/Probabilistic knowledge and reasoning

- *Other features covered in Parts (A.) and (C.):*
  - External Querying. Federation & orchestration.
  - Textual Rulelog: combining closely with Natural Language Processing.
  - Explanation. Terminology/ontology mapping.
- *Other features not covered much due to limitations of time & focus:*
  - Frame syntax (a.k.a. F-Logic), Object Oriented style
  - Reactiveness
  - Misc. Lesser Features: Datatypes, Aggregation, Integrity Constraints, Inheritance, Equality, "Constraints", "Modules"

Horn LP Compared to Horn FOL

- Fundamental Theorem connects Horn LP to Horn FOL:
  - M(P) = {all ground atoms entailed by P 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 Conc(P) = 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 {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"**
  - <u>Avoid disjunctions of positive literals</u> 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
  - Avoid general/unlimited "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

Outline of Part B. Concepts & Foundations

- Spirit of LP
- Higher-Order Syntax via Hilog. Reification.
- Rule ID's
- Default Negation and Well Founded Semantics
- Restraint: semantic bounded rationality
- Tabling Algorithms. Knowledge Debugging Methods.
- Defeasibility via Argumentation Rules. Remedying FOL's Fragility.
- General Formulas, Existentials and Skolems, Omni-directional Disjunction
    - Representing Text. Importing full OWL.
- Uncertain/Probabilistic knowledge and reasoning

- *Other features covered in Parts (A.) and (C.):*
    - External Querying. Federation & orchestration.
    - Textual Rulelog: combining closely with Natural Language Processing.
    - Explanation. Terminology/ontology mapping.
- *Other features not covered much due to limitations of time & focus:*
    - Frame syntax (a.k.a. F-Logic), Object Oriented style
    - Reactiveness
    - Misc. Lesser Features: Datatypes, Aggregation, Integrity Constraints, Inheritance, Equality, "Constraints", "Modules"

Hilog: Higher-Order Syntax

- A higher-order extension of predicate logic, which has a tractable first-order syntax
    - Allows certain forms of logically clean, yet tractable, meta-programming
    - Syntactically appears to be higher-order, but semantically is first-order and tractable

- 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

HiLog Transformation

- High-level Spirit:
  ?pred(?arg1,?arg2) $\rightarrow$ $\rightarrow$ believe(?pred,?arg1,?arg2)

- A simplified version of the transformation, which gives intuition:

  - Rewrite each atom   p(a,b) $\rightarrow$ holds_2(p,a,b)
    - Generic predicate constants holds_1, holds_2, …

  - Treat each term in similar manner
    - f(a,b) $\rightarrow$ apply_2(f,a,b)
    - Generic function constants apply_1, apply_2, …

Examples of Hilog (I)

Hilog permits variables over predicates and function symbols:

p(?X,?Y) :– ?X(a,?Z) and ?Y(?Z(b)).

Higher-order variable (a.k.a. meta-variable): ranges over predicate names of arity 2

Higher-order variable: ranges over function names of arity 1

Hilog also permits variables over atomic formulas. This is a kind of reification:

p(q(a)).
r(?X) :– p(?X) and ?X.

HiLog's Diffusion

- Used in ISO Common Logic to syntactically extend FOL
  - Also appears promising for OWL Full and its use of RDF [Kifer; Hayes]

- Implemented in Ergo, Flora-2 / Ergo Lite, SILK
  - Also partially exists in XSB, others

- [Chen, Kifer, Warren, "HiLog: A Foundation for Higher-Order Logic Programming", J. of Logic Programming, 1993]

Reification

- Reification makes a term out of a formula:

    believes(john, ${marylikes(mary,bob)})

- Introduced in [Yang & Kifer, ODBASE 2002]

- Rules can also be reified

Object made out of
the formula
mary[likes -> bob]

Rule ID's

- Simple, but important, feature
- Each (assertion) statement gets a unique rule id
- The id can be explicitly specified
  - @!{myRule17} H :- B.
- Or if implicit, is a skolem essentially
  - H :- B.   →   gets treated as: @!{gensym0897} H :- B.
- Enables various useful kinds of meta-knowledge, by asserting properties of the rule id
  - Provenance, e.g., createdBy(myRule17, Benjamin)
  - Defeasibility
  - Rule-based transformations, e.g., for language extensibility, UI, NLP
- Hidlog (pronounced "High-Dee-Log") = Hilog + rule id's

Representational Uses of HiLog

- *(We lump together with it also: reification and rule id's)*
- For <u>meta-</u> reasoning, e.g., in knowledge exchange or introspection
  - Meta-data is central to the Web
- Modals, e.g., believe, permit. Multi-agent belief. Deontics.
- Defeasibility: principles of argumentation/debate
- Restructuring in mapping of schemas, ontologies, and terminologies
  - E.g., in knowledge integration, federation, KB translation/import
- Conciseness. Simple example: transitive closure of a relation.
- Reasoning control
- Modularization of KB's
- Context, incl. provenance
- KR macros, modals, reasoning control, KB modularization, context
- Defeasibility: argumentation rules
- Probabilistic/uncertainty range and confidence about a sentence
- Representing natural language, e.g., compositionality of phrases
  - Compounding of nouns; salesman(?x); (insurance(salesman))(?x); ((life(insurance))(salesman))(?x)
  - Adverbial modification: quickly(give)(?thing,?recipient)

Outline of Part B. Concepts & Foundations

68

Logical Nonmonotonicity – Motivations

- Pragmatic reasoning is, in general, nonmonotonic
  - E.g., policies for taking actions, exception handling, legal argumentation, Bayesian/statistical/inductive, etc.
  - Monotonic is a special case – simpler in <u>some</u> regards
- Most commercially important rule systems/applications use nonmon
  - A basic expressive construct is ubiquitous there:
    - <u>Default Negation</u>   a.k.a. Negation-As-Failure (NAF)
      - BUT with varying semantics – <u>often not fully declarative</u> cf. LP
        - Primarily due to historical hangovers and lack of familiarity with modern algorithms
  - Another expressive construct, almost as ubiquitous there, is:
    - <u>Priorities</u> between rules
- Such nonmonotonicity enables:
  - Modularity and locality in revision/updating/merging

Default Negation: Intro

- *Default* negation, a.k.a. "*weak*" negation, a.k.a. "*negation as failure (NAF)*" is the most common form of negation in commercially important rule and knowledge-based systems.

- Normal LP, a.k.a. Ordinary LP, adds NAF to Horn LP
  - May appear in body literals, but not in head

- Concept/Intuition for \naf q   default negation
  - q is not derivable from the available premise info
  - fail to believe q . "I know I do not believe q".
  - … but might also not believe q to be false
  - A.k.a. "*weak*" negation, or NAF. In Ergo: "\naf"

- Contrast with: **\neg** q   strong negation, a.k.a. "classical" negation
  - q is believed to be false ; the opposite of q is true
  - Brings potential for logical contradiction / conflict: p vs. \neg p

Semantics for Default Negation

- Semantics of NAF has subtleties for the fully general case
  - Paradox can arise from "unstratified" NAF:
    - Cyclic dependencies among atoms/predicates: thru the rules, involving NAF
    - Example 1:   p :- \naf p.   Example 2: p :- \naf q. q :- \naf p.
  - Well-understood theoretically since 1994 – after 10 years of work
- <u>*Well Founded* Semantics</u> (WFS): popular; major commercial focus
  - Quadratic (often linear -- time for propositional. No reasoning by cases.
  - Employs a 3rd <u>truth value</u> u ("undefined") for paradox
    - Intuition: "I cannot figure out whether I believe p or do not believe p"
  - Definition uses <u>iterated</u> minimality: Horn-case then close-off; repeat til done.

  - Operational semantics for NAF, in actual practice, is often "sloppy" (incomplete / cut-corners / obsolete) relative to canonical semantics for NAF, in most Prolog and other currently commercially important rule systems that express NAF; required WFS algorithms are more complex.

ASP Semantics for Default Negation

- *Answer Set Programs* (ASP): popular as research topic
  - A different KR than WFS, departs particularly in presence of paradox
  - Intractable for propositional; does reasoning by cases
  - Enables a limited kind of disjunction in heads & conclusions
  - Good for combinatorial KRR problems requiring nonmonotonicity
  - Lacks undefined truth value $\Rightarrow$ sometimes ill-defined: no <u>set</u> of conclusions

Brief Examples of Unstratified Normal LP

- RB3:
  - a.
  - c :- a \and \naf b.
  - p :- \naf p.
- Well Founded Semantics (WFS) for RB3 entails conclusions {a,c}.
  p is not entailed.   p has "*undefined*" (u) truth value (in 3-valued logic).
- ASP Semantics for RB3: ill defined; there *is no* set of conclusions.
  - *(NOT   there is a set of conclusions that is empty.)*

- RB4:
  - a.
  - c :- a \and \naf b.
  - p :- \naf q.
  - q :- \naf p.
- WFS for RB4 entails conclusions {a,c}. p,q have truth value u.
- ASP Semantics for RB4 results in two alternative conclusion sets: {a,c,p} and {a,c,q}. Note their intersection {a,c} is the same as the WFS conclusions.

Outline of Part B. Concepts & Foundations

- Spirit of LP
- Higher-Order Syntax via Hilog. Reification.
- Rule ID's
- Default Negation and Well Founded Semantics
- Restraint: semantic bounded rationality
- Tabling Algorithms. Knowledge Debugging Methods.
- Defeasibility via Argumentation Rules. Remedying FOL's Fragility.
- General Formulas, Existentials and Skolems, Omni-directional Disjunction
    - Representing Text. Importing full OWL.
- Uncertain/Probabilistic knowledge and reasoning

- *Other features covered in Parts (A.) and (C.):*
    - External Querying. Federation & orchestration.
    - Textual Rulelog: combining closely with Natural Language Processing.
    - Explanation. Terminology/ontology mapping.
- *Other features not covered much due to limitations of time & focus:*
    - Frame syntax (a.k.a. F-Logic), Object Oriented style
    - Reactiveness
    - Misc. Lesser Features: Datatypes, Aggregation, Integrity Constraints, Inheritance, Equality, "Constraints", "Modules"

74

Requirements Analysis for Logical Functions

- Function-free is a commonly adopted restriction in practical LP/Web rules today
  - DB query languages: SQL, SPARQL, XQuery
  - Production rules, and similar Event-Condition-Action rules
  - OWL (including OWL-RL), RDF-Schema
  - RIF Basic Logic Dialect, RIF-Core (and SWRL)
  - Jena, SPIN

- BUT functions are often needed for Web (and other) applications.
  Uses include:
  - Skolemization – to represent existential quantifiers (incl. RDF blank nodes)
    - E.g., exists(?p)^(part_of(?p,?c) \and nucleus(?p)) :- eukaryotic_cell(?c).
      - → → part_of(sk1(?c),?c) \and nucleus(sk1(?c)) :- eukaryotic_cell(?c).
    - Related: convenient naming. E.g., steering_wheel(my_car)
  - HiLog and reification – higher-order syntax
    - For many purposes!

Functions in LP Lead to Undecidability

- Functions in LP 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
  – AAAI-13 paper "Radial Restraint: A Semantically Clean Approach to Bounded Rationality" by B. Grosof and T. Swift
  – RuleML-2013 paper "Advanced Knowledge Debugging for Rulelog" by C. Andersen et al.
  – Both are available at http://coherentknowlege.com/publications/

Functions in LP Lead to Undecidability

- Functions in LP 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)), …

- Practical effects: runaway computations during inferencing to answer a query

- A significant pain point for :
    - FOL and other classical-logic reasoning (e.g., higher-order)
    - LP
    - Prolog

- Heart of problem: the semantics, not the proof procedure or the implementation

Restraint – Bounded Rationality (I)

- Restraint: utilize the *undefined* (u) truth value
  - A fully semantic kind of bounded rationality
  - Intuition: <u>don't-care</u>, can't figure it out or don't want to figure it out
  - Intuition: <u>misty</u> regions of the set of all literals, in regard to truth value t vs. f

- Radial restraint:
  - Specify a fixed depth radius r on literals, e.g., r = 10
  - Every literal whose depth exceeds the radius is concluded to have truth value u
  - Depth here is a metric which could be nesting depth, term size, or similar

- Example revisited, when specify radial restraint
  - Then num(succ(succ(succ(succ(0))))), …   all have truth value u

- Radial restraint has a proof theory, incorporates straightforwardly into algorithms
  - Modest computational overhead, ~~10% in experiments

- Paradox – unstratified NAF – is a kind of restraint

Restraint (II)

- Several other kinds of restraint
  - Skipping: don't-care about particular instances of particular rules
    - Allows for more nuance and conditions about when to apply restraint
    - E.g., restrain a causal projection rule, when the successor-state step count exceeds 1000 and the causal effect predicate is microscopic
  - Sensor unreturn
    - E.g., an external query does not return, due to network or server failure
  - NAF unsafety
    - NAF'd literal must be evaluated with 1 or more unbound variables
  - Sensor unsafety
    - Externally queried literal has too many unbound variables for what the external knowledge source is willing/able to answer as a query

- Modest computational overhead required for the above

- For more info on restraint, see
  - AAAI-13 paper "Radial Restraint: A Semantically Clean Approach to Bounded Rationality" by B. Grosof and T. Swift
  - RuleML-2013 paper "Advanced Knowledge Debugging for Rulelog" by C. Andersen et al.
  - Both are available at http://coherentknowlege.com/publications/

Outline of Part B. Concepts & Foundations

- Spirit of LP
- Higher-Order Syntax via Hilog. Reification.
- Rule ID's
- Default Negation and Well Founded Semantics
- Restraint: semantic bounded rationality
- Tabling Algorithms. Knowledge Debugging Methods.
- Defeasibility via Argumentation Rules. Remedying FOL's Fragility.
- General Formulas, Existentials and Skolems, Omni-directional Disjunction
    - Representing Text. Importing full OWL.
- Uncertain/Probabilistic knowledge and reasoning

- *Other features covered in Parts (A.) and (C.):*
    - External Querying. Federation & orchestration.
    - Textual Rulelog: combining closely with Natural Language Processing.
    - Explanation. Terminology/ontology mapping.
- *Other features not covered much due to limitations of time & focus:*
    - Frame syntax (a.k.a. F-Logic), Object Oriented style
    - Reactiveness
    - Misc. Lesser Features: Datatypes, Aggregation, Integrity Constraints, Inheritance, Equality, "Constraints", "Modules"

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.

- Challenges include:
    - Logical functions: complex terms and unifiers
    - Hilog (higher-order syntax): e.g., indexing
    - NAF, non-monotonicity, and defeasibility is a challenge
- Techniques have been developed to meet all of these
    - Efficient search control, indexing and low level data structures
    - <u>Incremental</u> tabling adds more dependency-awareness to nonmon.
        - Enables fast updating via reuse of most previous inferences
        - E.g., for interactive rule authoring edit-test loop

- Highly sophisticated, optimized over last two decades

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.

- Challenges include:
  - Logical functions: complex terms and unifiers
  - Hilog (higher-order syntax): e.g., indexing
  - NAF, non-monotonicity, and defeasibility is a challenge
- Techniques have been developed to meet all of these
  - Efficient search control, indexing and low level data structures
  - <u>Incremental</u> tabling adds more dependency-awareness to nonmon.
    - Enables fast updating via reuse of most previous inferences
    - E.g., for interactive rule authoring edit-test loop

- Highly sophisticated, optimized over last two decades

Advanced Knowledge Base Debugging Methods in Rulelog

- Explanation – as we say in the Reg W case study in Ergo
  - Helps to identify missing knowledge,
    as well as wrong knowledge

- Pause/resume
- Performance monitor
- Analysis of attempted infinite loops ("terminyzer")
- Analysis of sizes of tables (subgoal attempts)

- Ergo implements these
  - Several were pioneered in Vulcan SILK

Outline of Part B. Concepts & Foundations

- Spirit of LP

Key features

- Higher-Order Syntax via Hilog. Reification.
- Rule ID's
- Default Negation and Well Founded Semantics
- Restraint: semantic bounded rationality
- Tabling Algorithms. Knowledge Debugging Methods.
- Defeasibility via Argumentation Rules. Remedying FOL's Fragility.
- General Formulas, Existentials and Skolems, Omni-directional Disjunction
  - Representing Text. Importing full OWL. FOL-Soundness.
- Uncertain/Probabilistic knowledge and reasoning

- *Other features covered in Parts (A.) and (C.):*
  - External Querying. Federation & orchestration.
  - Textual Rulelog: combining closely with Natural Language Processing.
  - Explanation. Terminology/ontology mapping.
- *Other features not covered much due to limitations of time & focus:*
  - Frame syntax (a.k.a. F-Logic), Object Oriented style
  - Reactiveness
  - Misc. Lesser Features: Datatypes, Aggregation, Integrity Constraints, Inheritance, Equality, "Constraints"

The following couple sections are TO-SKIM

- Defeasibility
- Omniform rules
- I.e., slides ~86-113

- (Ideally, these would have been compressed to be much shorter.)

Defeasible Knowledge & Reasoning

- **Concept of defeasibility: Rules can be true by default but may be *defeated*, i.e., have exceptions**
  - A form of commonsense reasoning

- **Rulelog has defeasibility. Most previous KRR languages do not.**
  - Its approach to defeasibility is the most flexible, efficient, and practical
    - "Argumentation rules": the principles of defeat/debate are specified via a set of general meta-flavor rules

- **"A cell has a nucleus."   … Except when it doesn't ☺**
  - A cell has no nucleus during anaphase. Red blood cells have no nuclei.
  - A cell has two nuclei between mitosis and cytokinesis. Some fungi are multinucleate.

Knowledge often has Exceptions

- Exceptions / special cases are inevitably realized over time
  - E.g., knowledge is incomplete, multiple authors contribute, …

- Requiring entered knowledge to be strictly / universally true (exception-free) is impractical
  - Precludes stating generalities (the typical), and limits the author pool
  - "The perfect is the enemy of the good"

- Exceptions manifest as contradictions, i.e., conflict

- Leveraging multiple sources of knowledge (e.g., KB merging) requires conflict resolution
  - Errors. Confusions. Omitted context.

Defeasible Reasoning

- **Rules can be true by default but may be defeated**
  - A form of commonsense reasoning

- **Application domains:**
  - policies, regulations, and law
  - actions, change, and process causality
  - Web services
  - inductive/scientific learning
  - natural language understanding
  - …

- **Previous approaches (i.e., previous to Rulelog/Ergo):**
  - Courteous Logic Programs (Grosof , 1997)
    - The main approach used commercially (IBM Common Rules, 1999)
  - Defeasible logic (Nute, 1994) *[similar to Courteous LP]*
  - "Prioritized defaults" (Gelfond & Son, 1997)
  - Preferred answer sets (Brewka & Eiter, 2000)
  - Compiling preferences (Delgrande et al., 2003)
  - …

Defeasibility is Always Prepared for Exceptions

- **Recognizes and handles conflict**

- **Avoids unreliable conclusions from inconsistent knowledge**

  - Unlike FOL

- **Represents meta-knowledge which resolves conflicts**

  - Minimize need to modify prior knowledge, by acquiring additional meta-knowledge

- **… Notably: priorities (partially-ordered) between rules**

  - Some rules have higher priority than others

- **Priorities arise naturally from: specificity, recency, authority, causality, reliability**

- **Prioritization tames conflict $\Rightarrow$ aids modularity**

  - Analogy: the gift of fire

Defeasibility is Indicated When…

- **Useful generalities – <u>and</u> potential exceptions – coexist**
    - Specify knowledge in detail/precision <u>appropriate</u> for various circumstances

- **Governing doctrine, definitions, or other knowledge, cannot be assured to be conflict-free, e.g.:**
    - Multiple sources of governing doctrine exist
        - Typically, no central authority resolves all conflict promptly
    - Truth depends on context
        - Yet context is rarely made fully explicit

- **Many broad realms are full of exceptions**
    - Policies, regulations, laws   —   and the workflows they drive
        - Multiple jurisdictions, organizations, contracts, origins
    - Learning and science. Updating. Debate.
        - May falsify previous hypotheses after observation or communication
    - Causal processes: changes to state, from interacting/multiple causes
    - Natural language (text interpretation): "there's a gazillion special cases"

Ubiquity of Priorities
in Commercially Important Rules -- and Ontologies

- Updating in relational databases
  - more recent fact   *overrides*   less recent fact
- Static rule ordering in Prolog
  - rule earlier in file   *overrides* rule later in file
- Dynamic rule ordering in production rule systems (OPS5)
  - "meta-"rules can specify agenda of rule-firing sequence
- Event-Condition-Action rule systems rule ordering
  - often static or dynamic, in manner above
- Exceptions in default inheritance in object-oriented/frame systems
  - subclass's property value   *overrides*   superclass's property value, e.g., method redefinitions
- All lack Declarative KR Semantics

Semantic KR Approaches to Prioritized LP

The currently most important for Semantic Web are:

1. Courteous LP
    - KR extension to Ordinary LP
    - In RuleML, since 2001
    - Commercially implemented and applied
        – IBM's CommonRules, 1999-2009
        – Coherent Knowledge's Ergo, 2013*-present

2. Defeasible Logic
    - Closely related to Courteous LP
        – Less general wrt typical patterns of prioritized conflict handling needed in e-business applications
        – In progress: theoretical unification with Courteous LP [Wan, Kifer, Grosof RR-2010 / Semantic Web Journal 2015]

* and earlier in Flora-2 (a.k.a. Ergo Lite) 2009-2013

Courteous LP: the What

- **Updating/merging of rule sets: is crucial, often generates conflict.**
- **<u>Courteous</u> LP's feature prioritized handling of conflicts.**
- **Specify scope of conflict via a set of <u>exclusion</u> constraints**
  - Each is a <u>preventive</u> spirit integrity constraint on a set of competing literals
    - It says that not all of the competing literals can be entailed as true.
    - opposes(p, q) $\approx$ ($\perp$:- p and q)   // Case of 2 competing literals
  - opposes(discount(?product,"5%"), discount(?product,"10%"));
  - opposes(loyalCustomer(?cust,?store), premiereCustomer(?cust,?store));
- **Permit <u>strong negation of atoms</u>:   (NB: a.k.a. (quasi-) <u>"classical"</u> negation.)**
  - \neg p means p has truth value *false* .
  - implicitly, for every atom p: opposes(p, \neg p);
- <u>Priorities</u> between rules: <u>partially-ordered</u>.
  - Represent priorities via <u>reserved predicate</u> that compares <u>rule tags</u>:
    - \overrides(rule1, rule2)   means rule1 is higher-priority than rule2.
    - Each rule optionally has a rule tag whose form is a functional term.
    - \overrides   <u>can be reasoned about</u>, just like any other predicate.

Courteous LP: Advantages

- Facilitate updating and merging, modularity and locality in specification.

- Expressive: strong negation, exclusions, partially-ordered prioritization, reasoning to infer prioritization.

- Guarantee consistent, unique set of conclusions.
  - **Exclusion is enforced**. E.g., never conclude discount is both 5% and that it is 10%, nor conclude both p and ¬p.

- Scalable & Efficient: low computational overhead beyond ordinary LPs.
  - Tractable given reasonable restrictions (VB, and radial restraint or no functions):
    - extra cost is equivalent to increasing v to (v+2) in Ordinary LP, worst-case.
  - By contrast, more expressive prioritized rule representations (e.g., Prioritized Default Logic) add NP-hard overhead.

- Modular software engineering:
  - Transform: CLP $\rightarrow \rightarrow$ OLP.   Via "argumentation theory" approach.

EECOMS Example of Conflicting Rules: Ordering Lead Time

- Vendor's rules that prescribe how buyer must place or modify an order:
- A) 14 days ahead if the buyer is a qualified customer.
- B) 30 days ahead if the ordered item is a minor part.
- C) 2 days ahead if the ordered item's item-type is backlogged at the vendor, the order is a modification to reduce the quantity of the item, and the buyer is a qualified customer.
- D) 45 days ahead if the buyer is a walk-in customer.

- Suppose more than one of the above applies to the current order? **Conflict!**
- Helpful Approach: __precedence__ between the rules.
  - E.g., D is a catch-case: A > D , B > D , C > D
- Often only *partial* order of precedence is justified.
  - E.g., C > A , but no precedence wrt B vs. A, nor wrt C vs. B.

Ordering Lead Time Example in LP with Courteous Defaults

@{prefCust} orderModifNotice(?Order,14days)   :-

> preferredCustomerOf(?Buyer,SupplierCo),   purchaseOrder(?Order,?Buyer,SellerCo) .

@{smallStuff} orderModifNotice(?Order,30days)   :-

> minorPart(?Buyer,?Seller,?Order),   purchaseOrder(?Order,?Buyer,SupplierCo) .

@{reduceTight} orderModifNotice(?Order,2days)   :-

> preferredCustomerOf(?Buyer,SupplierCo) and

> orderModifType(?Order,reduce) and

> orderItemIsInBacklog(?Order) and

> purchaseOrder(?Order,?Buyer,SupplierCo) .

\overrides(reduceTight, prefCust) .   // *reduceTight has higher priority than prefCust*

// *The below <u>exclusion</u> constraint specifies that orderModifNotice is unique, for a given order.*

\opposes(orderModifNotice(?Order,?X), orderModifNotice(?Order,?Y))   :-   ?X != ?Y .

- Rule D, and prioritization about it, were omitted above for sake of brevity.
- Above rules are represented in Logic Programs KR, using the <u>Courteous defaults</u> feature
- Notation:
  - ":-" means "if". "@…" declares a rule tag. "?" prefixes a logical variable.
    "\overrides" predicate specifies prioritization ordering.
    An exclusion constraint specifies what constitutes a conflict.
    "!=" means ≠ .

Courteous LP Semantics: Prioritized argumentation in an opposition locale.

Conclusions from opposition-locales <u>previous</u> to this opposition-locale {p1, p2}

(p1, p2 are each a ground strong literal, e.g., q, neg q)

⬇

Run Rules for p1, p2

⬇

Set of <u>Candidates</u> for p1, p2:
Team for p1, ..., Team for pk

⬇

Prioritized Refutation

⬇

Set of <u>Unrefuted</u> Candidates for p1, p2:
Team for p1, Team for p2

⬇

Skepticism

⬇

<u>Conclude</u> Winning Side if any: at most one of {p1, p2}

Argumentation Rules approach to Defaults in LP

- **Combines Courteous + HiLog, and generalizes**
- **New approach to defaults: "argumentation rules"**
  - Meta-rules, in the LP itself, that specify when rules ought to be defeated
  - [Wan, Grosof, Kifer, *et al.* ICLP-2009; RR-2010; SWJ 2015]
- **Extends straightforwardly to combine with other key features**
  - E.g., Frame syntax, external Actions
- **Significant other improvements on previous Courteous**
  - Eliminates a complex transformation
  - Much simpler to implement
    - 20-30 background rules instead of 1000's of lines of code
  - Much faster when updating the premises
  - More flexible control of edge-case behaviors
  - Much simpler to analyze theoretically

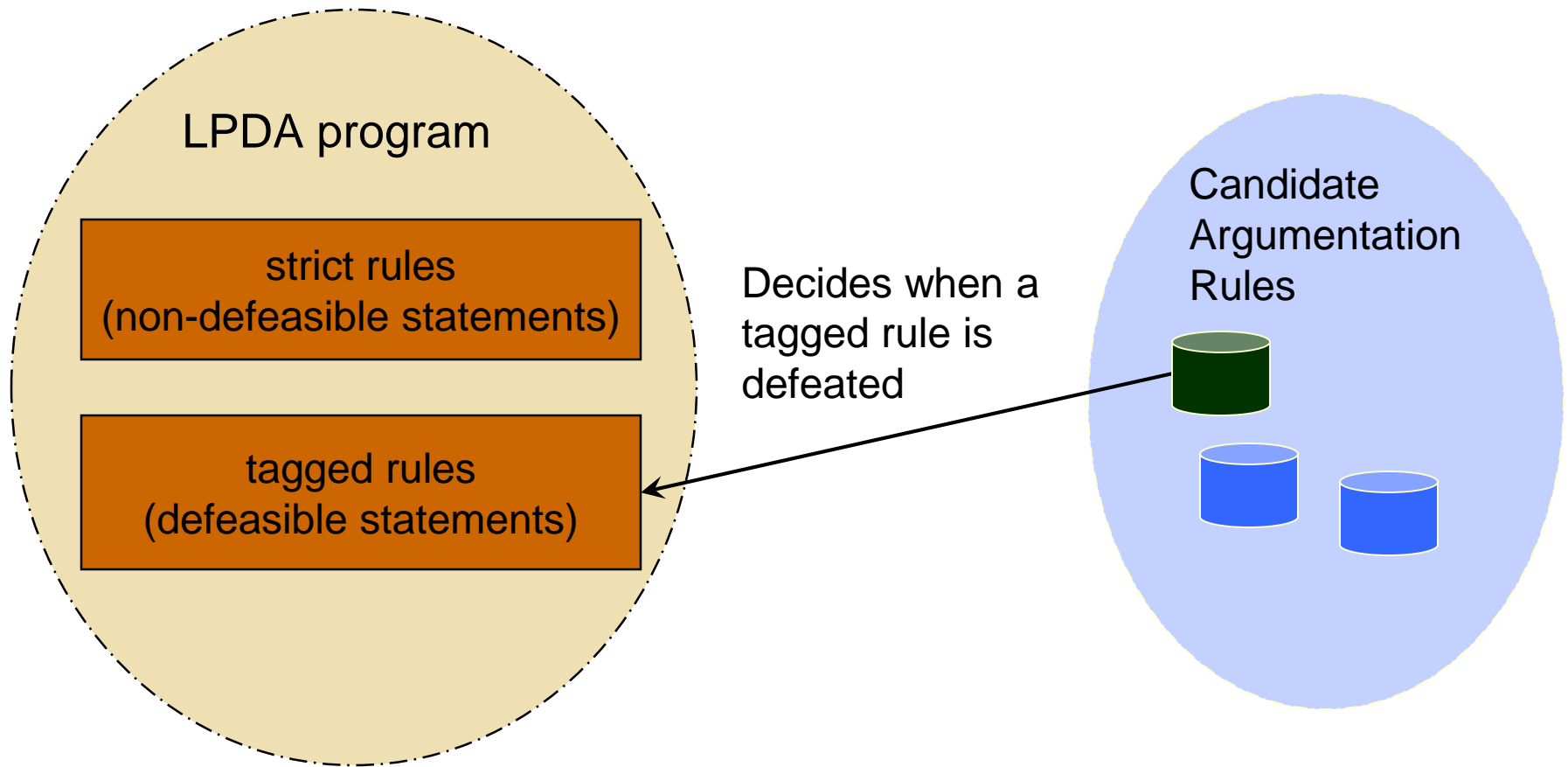# *LPDA* *Approach, Continued*

- **More Advantages**
  - 1st way to generalize defeasible LP, notably Courteous, to HiLog higher-order and F-Logic frames
  - Well-developed model theory, reducible to normal LP
  - Reducibility results
  - Well-behavior results, e.g., guarantees of consistency
  - Unifies almost all previous defeasible LP approaches
    - Each reformulated as an argumentation theory
    - E.g., Defeasible Logic (see Wan, Kifer, and Grosof RR-2010 / SWJ 2015 paper)
  - Cleaner, more flexible and extensible semantics
    - Enables smooth and powerful integration of features
    - Applies both to well founded LP (WFS) and to Answer Set Programs (ASP)
  - Leverages most previous LP algorithms & optimizations

- **Implemented** in Ergo, also earlier in Flora-2 and used in SILK

LPDA Framework

- **L**ogic **P**rograms with **D**efaults and **A**rgumentation rules

LPDA program

strict rules
(non-defeasible statements)

tagged rules
(defeasible statements)

Decides when a
tagged rule is
defeated

Candidate
Argumentation
Rules

Example – AT for Courteous ($\mathcal{AT}^{\mathcal{GCLP}}$)

defeated(?R)   :- defeats(?S, ?R).

defeats(?R, ?S) :- refutes(?R, ?S) or $rebuts(?R, ?S).

> Prioritization (user specified)

refutes(?R, ?S) :- conflict(?R, ?S), \overrides(?R, ?S).

refuted(?R)   :- refutes(?R2, ?R).

rebuts(?R, ?S) :- conflict(?R, ?S),

> Default negation (NAF)

naf refuted(?R), \naf refuted(?S).

> Meta predicates ("Reflection")

candidate(?R)   :- body(?R, ?B), call(?B).

conflict(?R, ?S) :- candidate(?R), candidate(?S),

\opposes(?R, ?S).

\opposes(?R, ?S)   :- \opposes(?S, ?R).

> Exclusion (user specified)

\opposes(?L1,?L2) :- head(?L1, ?H), head(?L2, \neg ?H).

> Explicit negation

101

Ecology Ex. of Causal Process Reasoning

/*   Toxic discharge into a river causes fish die-off.   */

/* Init. facts, and an "exclusion" constraint that fish count has a unique value */

  occupies(trout,Squamish).

  fishCount(0,Squamish,trout,400). / * 1st argument of fishCount is an integer time */

  \opposes(fishCount(?s,?r,?f,?C1), fishCount(?s,?r,?f,?C2)) :- ?C1 != ?C2.

/* Action/event description that specifies causal change, i.e., effect on next state */

 @{tdf1} fishCount(?s+1,?r,?f,0) :- occurs(?s,discharge,?r) \and occupies(?f,?r).

/* Persistence ("frame") axiom */

  @{pefc1} fishCount(?s+1,?r,?f,?p) :- fishCount(?s,?r,?f,?p).

/* Action effect axiom has higher priority than persistence axiom */

 \overrides(tdf1,pefc1).

/* An action instance occurs */

  @{UhOh} occurs(1,toxicDischarge,Squamish).

As desired: |= fishCount(1,Squamish,trout,400),
                        fishCount(2,Squamish,trout,0)

Notes: @… declares a rule tag. ? prefixes a variable. :- means if. != means ≠.   \opposes indicates an
exclusion constraint between two literals, which means "it's a conflict if".

Physics Ex. of Contextual Assumptions

/* "P8: Joe drops a glove from the top of a 100m cliff.

      How long does the fall take in seconds?" */

// Initial problem-specific facts

AP_problem(P8); fall_event(P8); P8[height->100].

// Action description that specifies causal implications on the continuous process

?e[time->((2 * ?h / ?n)^0.5)] :- fall_event(?e) \and ?e[height->?h, net_accel->?n].

?e[net_accel->(?g - ?a)] :- fall_event(?e) and

                    ?e[gravity_accel->?g, air_resistance_accel->?a].

 // Other facts

?e[gravity_accel->9.8] :- loc(?e, Earth).

?e[gravity_accel->3.7] :- loc(?e, Mars).

// Contextual assumptions for answering Advanced Placement exam (AP) problems

@{implicit_assumption} loc(?e, Earth) :- AP_problem(?e).

\opposes(loc(?e, Earth), loc(?e, Mars)).

@{implicit_assumption} ?e[air_resistance_accel->0] :- AP_problem(?e).

\overrides(explicitly_stated, implicit_assumption).

As desired: |= P8[net_accel->9.8, time->4.52]   // 4.52 = (2*100/9.8)^0.5

Physics Ex. of Contextual Assumptions (in Ergo)

/* "P8: Joe drops a glove from the top of a 100m cliff <u>on Mars</u>.

How long does the fall take in seconds?" */

/* Initial problem-specific facts*/

AP_problem(P8). fall_event(P8). P8[height->100].

@{explicitly_stated} loc(P8,Mars).

…

As desired: |= P8[net_accel-><u>3.7</u>, time-><u>7.35</u>]   // 7.35 = (2*100/<u>3.7</u>)^0.5

Example: Ontology Translation, leveraging hilog and exceptions

/* Company BB reports operating earnings using R&D operating cost which includes price of a small company acquired for its intellectual property. Organization GG wants to view operating cost more conventionally which excludes that acquisition amount. We use rules to specify the contextual ontological mapping. */

@{normallyBringOver} ?categ(GG)(?item) :- ?categ(BB)(?item).

@{acquisitionsAreNotOperating} \neg ?categ(GG)(?item) :-
    acquisition(GG)(?item) \and (?categ(GG) :: operating(GG)).

\overrides(acquisitionsAreNotOperating, normallyBringOver). /* exceptional */

acquisition(GG)(?item) :- price_of_acquired_R_and_D_companies(BB)(?item).

R_and_D_salaries(BB)(p1001). p1001[amount -> $25,000,000].

R_and_D_overhead(BB)(p1002). p1002[amount -> $15,000,000].

price_of_acquired_R_and_D_companies(BB)(p1003). p1003[amount -> $30,000,000].

R_and_D_operating_cost(BB)(p1003). /* BB counts the acquisition price item in this category */

R_and_D_operating_cost(GG) :: operating(GG).

Total(R_and_D_operating_cost)(BB)[amount -> $70,000,000]. /* rolled up by BB cf. BB's definitions */

Total(R_and_D_operating_cost)(GG)[amount -> ?x] :- … . /* roll up the items for GG cf. GG's definitions */

As desired: |= R_and_D_salaries(GG)(p1001)

|= \neg R_and_D_operating_cost(GG)(p1003) /* GG doesn't count it */

|= Total(R_and_D_operating_cost)(GG)[amount -> $40,000,000]

Notation: @{…} declares a rule tag. ? prefixes a variable. :- means if. X :: Y means X is a subclass of Y.
\overrides(X,Y) means X is higher priority than Y.

Trust Mgmt. Ex. of Higher-Order Defaults
illustrating also basic Knowledge-level Communication, and Frame syntax

In Frame syntax: subject[property -> object] *stands for* property(subject,object).

/*   Trust policy administration by multiple agents, about user permissions */
/* Admin. Bob controls printing privileges including revocation (neg). */
  Bob[controls -> print];   Bob[controls -> \neg print].   /* \neg print means it is disallowed.*/
  Cara[controls -> ?priv]; /* Cara is the most senior admin., so controls all privileges. */
/* If an administrator controls a privilege and states at a time (t) that a user has a privilege,
   then the user is granted that privilege. Observe that ?priv is a higher-order variable. */
  @{grant(?t)} ?priv(?user) :- ?admin[states(?t) -> ?priv(?user)] and ?admin[controls(?priv)].
/* More recent statements have higher priority, in case of conflict. */
  \overrides(grant(?t2), grant(?t1)) :- ?t2 > ?t1.
/* Admins Bob and Cara make conflicting statements over time about Ann's printing */
  Cara[states(2007) -> print(Ann)];   Cara[states(2007) -> webPage(Ann)].
  Bob[states(2008) -> \neg print(Ann)].

As desired: |= \neg print(Ann),   webPage(Ann)
                        /* Currently, Ann is permitted a webpage but not to print. */

Notes: @[…] declares a rule tag. ? prefixes a variable. :- means if. != means ≠. neg is strong negation.
There is an implicit exclusion (\opposes) between P and neg P, for every literal P.

Outline of Part B. Concepts & Foundations

Key features

- Spirit of LP
- Higher-Order Syntax via Hilog. Reification.
- Rule ID's
- Default Negation and Well Founded Semantics
- Restraint: semantic bounded rationality
- Tabling Algorithms. Knowledge Debugging Methods.
- Defeasibility via Argumentation Rules. Remedying FOL's Fragility.
- General Formulas, Existentials and Skolems, Omni-directional Disjunction
  - Representing Text. Importing full OWL.
- Uncertain/Probabilistic knowledge and reasoning

- *Other features covered in Parts (A.) and (C.):*
  - External Querying. Federation & orchestration.
  - Textual Rulelog: combining closely with Natural Language Processing.
  - Explanation. Terminology/ontology mapping.
- *Other features not covered much due to limitations of time & focus:*
  - Frame syntax (a.k.a. F-Logic), Object Oriented style
  - Reactiveness
  - Misc. Lesser Features: Datatypes, Aggregation, Integrity Constraints, Inheritance, Equality, "Constraints", "Modules"

Omniform Rules: Clausal case

- **Rulelog introduces the concept of an *omniform ("omni") rule*.**
- **Basic case is <u>clausal</u>. Here, the clause is treated *omni-directionally*.**
  - @{G} FC . where FC has the syntactic form of a FOL clause
    - The prioritization tag (@{G}) is optional. Outer universal quantification is implicit.
  - E.g., @{hi} wet(lawn, nextMorning(?night)) \or \neg occur(rain, ?night) ;

- **A clausal omni rule is transformed, i.e., *directionalized*, from**
  @{G} L_1 \or L_2 \or … \or L_k; where each L_i is an atom or the \neg of an atom
  into a set of k <u>directed</u> rules, one for each choice of head literal:
  @{G} L_1 :- \neg L_2 \and \neg L_3 \and … \and \neg L_k .
  @{G} L_2 :- \neg L_1 \and \neg L_3 \and … \and \neg L_k .
  …
  @{G} L_k :- \neg L_1 \and \neg L_2 \and … \and \neg L_{k-1} .

\naf-free !

- **This is called the set of *directional <u>variant</u>* rules.**

- **Avoids unrestricted reasoning by cases!!!**
  - Cf. unit/linear resolution strategy in FOL

- **<u>Sound</u> with respect to FOL** semantics of the clauses. I.e., "hypermonotonic".

Examples of Directionalization

- **@{hi} wet(lawn, nextMorning(?night)) <== Occur(rain, ?night) .   /* Causal */**
  is transformed into:
  - **@{hi} Wet(lawn, nextMorning(?night) :- Occur(rain, ?night) ;**
  - **@{hi} \neg Occur(rain, ?night) :- \neg Wet(lawn, nextMorning(?night) ;**

- **\neg (Cat(?x) \and Bird(?x)) .   /* OWL-DL disjoint classes */**
  is transformed into:
  - **\neg Cat(?x) :- Bird(?x) .**
  - **\neg Bird(?x) :- Cat(?x) .**

- **\neg Approved(?p) <== \neg Validated(?p) ;   /* SBVR: Car Rental Constraint */**
  is transformed into:
  - **\neg Approved(?p) :-   \neg Validated(?p) .**
  - **Validated(?p)   :-   Approved(?p) .**

- **mtg(3p) \or mtg(4p) \or mtg(5p) .   /* Scheduling: Joe's meeting time */**
  is transformed into:
  - **mtg(5p) :- \neg mtg(3p) \and \neg mtg(4p) .**
  - **mtg(4p) :- \neg mtg(3p) \and \neg mtg(5p) .**
  - **mtg(3p) :- \neg mtg(4p) \and \neg mtg(5p) .**

Omnis: General case

- **Permit the formula *F* to:**
  - Have the form of any FOL formula ("FOL-like")
  - Also use HiLog and Frame features
- **Permit a rule body too**
  - @*G   F* :- *B* .
  - Adds *B* to the body of each directional variant rule
  - *B* is similar in form to F, but also permits NAF
  - Special case: *F* is a literal
- **Semantics of existentials has subtleties**
  - Use skolemization, via a *tight* normal form (TNF) that's a bit different from Skolem NF. Argumentation theory is tweaked.
- **Omni feature raises the KR abstraction level**
  - Hide directionality (:-) as well as NAF (\naf)
  - Use instead: \neg (strong negation), <== (strong/material implication), and defeasibility (courteous)

Remedying FOL Semantics' Lack of Scalability

- **Rulelog handles conflict robustly – get consistent conclusions**
  - **Whereas FOL is a "Bubble" – it's <u>perfectly</u> brittle semantically in face of contradictions from quality problems or merging conflicts.**
    - Any contradiction is totally contagious – the conclusions all become garbage

    E.g., OWL beyond the RL subset suffers this problem. So does Common Logic. (Technically, RIF-BLD and RDF(S) are defined via FOL semantics too, although their typical implementations are essentially LP.)

A KB with a million or billion axioms formed by merging from multiple Web sources, is unlikely to have <u>zero</u> KB/KA conflicts from:
  - Human knowledge entry/editing
  - Implicit context, cross-source ontology interpretation
  - Updating cross-source
  - Source trustworthiness

- *Rulelog's approach provides a <u>critical</u> advantage for KB scalability*
  - *<u>semantically</u>, as well as computationally*

Extreme sensitivity to conflict limits its scalability in # of axioms and # of merges

© Richard Heeks / Barcroft Media

Left:

http://www.dailymail.co.uk/sciencetech/article-1199149/Super-slow-motion-pictures-soap-bubble-bursting-stunning-detail.html

Above:

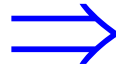http://img.dailymail.co.uk/i/pix/2007/11_03/BubblePA_468x585.jpg

KR Conflict Handling – A Key to Scalability

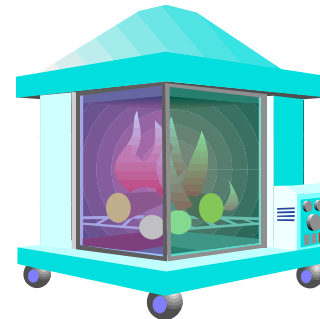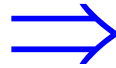## BEFORE

AFTER

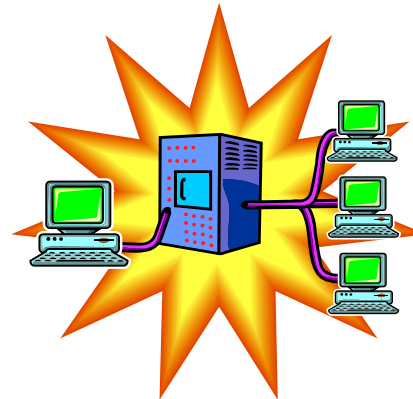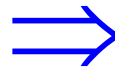KR: Classical Logic (FOL, OWL) $\Rightarrow$ KR: LP with Defaults (Courteous-style)

Contradictory conflict is globally contagious, invalidates all results. $\Rightarrow$ Contradictory conflict is contained locally, indeed tamed to aid modularity.

Knowledge integration involving conflict is labor-intensive, slow, costly. $\Rightarrow$ Knowledge integration involving conflict is highly automated, faster, cheaper.

Outline of Part B. Concepts & Foundations

- Spirit of LP
- Higher-Order Syntax via Hilog. Reification.
- Rule ID's
- Default Negation and Well Founded Semantics
- Restraint: semantic bounded rationality
- Tabling Algorithms. Knowledge Debugging Methods.
- Defeasibility via Argumentation Rules. Remedying FOL's Fragility.
- General Formulas, Existentials and Skolems, Omni-directional Disjunction
  - Representing Text. Importing full OWL.
- Uncertain/Probabilistic knowledge and reasoning

- *Other features covered in Parts (A.) and (C.):*
  - External Querying. Federation & orchestration.
  - Textual Rulelog: combining closely with Natural Language Processing.
  - Explanation. Terminology/ontology mapping.
- *Other features not covered much due to limitations of time & focus:*
  - Frame syntax (a.k.a. F-Logic), Object Oriented style
  - Reactiveness
  - Misc. Lesser Features: Datatypes, Aggregation, Integrity Constraints, Inheritance, Equality, "Constraints", "Modules"

Uncertainty, including Probability

- Many important and popular kinds of uncertain knowledge and reasoning fall within Rulelog's expressiveness

- Syntactic representation of uncertainty – including probability – "comes for free" with Hilog and strong meta

  - Represent uncertain knowledge as

    uncertainty(*?formula*, *?parameters)*

    - E.g., parameters might be point value, lower/upper bounds, confidence level, sample-size, statistical technique, etc.

- Rulelog can then be used to axiomatize desired reasoning principles with such statements – "roll-your-own"

  - E.g., leveraging arithmetic operations, defeasibility, restraint, etc.

- Ergo also has been developing optimized support for specific kinds of probabilistic uncertainty

  - Evidential reasoning: weighted or prioritized combination

  - Distribution semantics: semantics/foundation of Probabilistic LP

115

Reasoning with Uncertainty using Rulelog

Mature:

- Treat probabilistic statements as a special case of general logical statements and approximate the desired type of reasoning by incorporating certainty factors (that represent probability values), into the general Ergo reasoning facilities. I.e., "roll-your-own" uncertain reasoning.

  - Leverages highly expressive features of Rulelog including higher-order syntax
  - Examples of reasoning with uncertainty that can be incorporated into rules:
    - Lower and upper bounds on probability value.
      - E.g., *prob_range(needs_repair(part32), 0.89, 0.94).*
    - Confidence level. E.g., *prob_range_with_confidence(needs_repair(part32), 0.89, 0.94, 0.001).*
    - Source and provenance info about the statistical or ML method used to derive probability value/range. This can be a basis for certainty factors.
      - E.g., source(prob_range(needs_repair(part32), 0.89, 0.93), ML_episode(myFavoriteMLClassifier, 'Michael Kifer', 'Feb 11, 2017', http://mycompany.com/dataset41)).

Additional Ways of Reasoning with Uncertainty in Ergo

In development:

- "Evidential reasoning": combines information about probabilities based on different conditions and associated data sets
  - Addresses the probability whether a particular "thing" (i.e, a person, situation, etc.) belongs to a particular class.
  - Examples:
    - The probability that person X has a given disease
    - The probability that a given transaction is risky
    - The probability that a given airplane part will fail given its age, type, and manufacturer
  - This type of reasoning does not require complete or even consistent knowledge about probabilities and is scalable

- Restricted "distribution semantics" (DS) for probability*
  - ~~5 major approaches to probabilistic LP are based on DS, under various names, e.g., ProbLog, PRISM
  - DS's expressiveness supersumes Bayesian networks

- Triangular-norm style weighted uncertainty*
  - Similar to the uncertainty methods used for neural networks, fuzzy logic, some other ML/KRR systems

* less mature than evidential reasoning

# *Outline of Part B. Concepts & Foundations*

Key features

- Spirit of LP
- Higher-Order Syntax via Hilog. Reification.
- Rule ID's
- Default Negation and Well Founded Semantics
- Restraint: semantic bounded rationality
- Tabling Algorithms. Knowledge Debugging Methods.
- Defeasibility via Argumentation Rules. Remedying FOL's Fragility.
- General Formulas, Existentials and Skolems, Omni-directional Disjunction
  - Representing Text. Importing full OWL.
- Uncertain/Probabilistic knowledge and reasoning

- *Other features covered in Parts (A.) and (C.):*
  - External Querying. Federation & orchestration.
  - Textual Rulelog: combining closely with Natural Language Processing.
  - Explanation. Terminology/ontology mapping.
- *Other features not covered much due to limitations of time & focus:*
  - Frame syntax (a.k.a. F-Logic), Object Oriented style
  - Reactiveness
  - Misc. Lesser Features: Datatypes, Aggregation, Integrity Constraints, Inheritance, Equality, "Constraints", "Modules"

Integrity Constraints

Two styles with quite different semantics:

1. Alarm: Rule that detects a violation
   - Typical: the rule reports/notifies that constraint is violated
     - Other rules infer resulting actions to take
   - E.g., many BRMS, Ergo Lite, Ergo
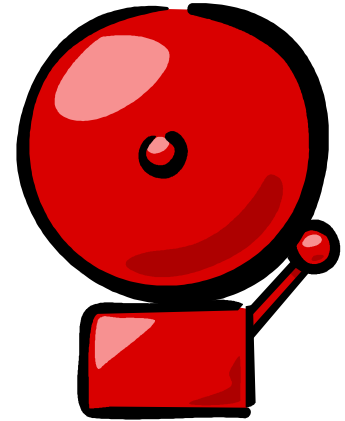
…VERSUS…

2. Model-cutting: Rule that forces global contradiction when axiom is violated
   - Typical: no model, lose all useful entailments!!
   - E.g., FOL

Additional Expressive Features in Rulelog

- Explicit <u>equality</u> (and equivalence) reasoning
  - In head of non-fact rules, therefore <u>derived</u>
  - Interaction with <u>nonmonotonicity</u>
  - Key characteristic: <u>substitutivity</u> of equals for equals
  - Related to Herbrand aspect of LP semantics

- <u>Existentials</u>, <u>skolemization</u>
  - RDF blank-nodes, anonymous individuals [Yang & Kifer]
  - Related to Herbrand aspect of LP semantics

- <u>Aggregation</u> (operate on entailed lists): count, total, min, max, etc.
  - Depends on <u>nonmonotonicity</u>, stratification

- <u>Datatypes</u> – they are basic but fairly straightforward

- "<u>Constraints</u>" (e.g., equation/inequality systems)
  - Commonly: via external query/assert to specialized solver

- "<u>Modules</u>": sub- KBs with information hiding and optimizations

- *Also implemented in Ergo Lite & Ergo, tho' not part of Rulelog itself: Transaction Logic [Kifer et al] – including KRR about <u>actions' results, transactionality</u>, and some kinds of hypotheticals*

Outline of Part B. Concepts & Foundations

- Spirit of LP
- Higher-Order Syntax via Hilog. Reification.
- Rule ID's
- Default Negation and Well Founded Semantics
- Restraint: semantic bounded rationality
- Tabling Algorithms. Knowledge Debugging Methods.
- Defeasibility via Argumentation Rules. Remedying FOL's Fragility.
- General Formulas, Existentials and Skolems, Omni-directional Disjunction
  - Representing Text. Importing full OWL.
- Uncertain/Probabilistic knowledge and reasoning

- *Other features covered in Parts (A.) and (C.):*
  - External Querying. Federation & orchestration.
  - Textual Rulelog: combining closely with Natural Language Processing.
  - Explanation. Terminology/ontology mapping.
- *Other features not covered much due to limitations of time & focus:*
  - Frame syntax (a.k.a. F-Logic), Object Oriented style
  - Reactiveness
  - Misc. Lesser Features: Datatypes, Aggregation, Integrity Constraints, Inheritance, Equality, "Constraints", "Modules"

Outline of Tutorial

A. Overview

  ➤ Practical logic. Applications. Features. Software. Textual.

  ➤ Case Study Demo and Features Tour

    ➤ Financial regulatory/policy compliance

B. Concepts and Foundations: Drill-downs

  ➤ Expressive features, semantics, algorithms; relationships to natural language and machine learning

C. Conclusions and Future Work

  – Background Assumed: basic knowledge of first-order logic, databases. Helpful: XML, RDF and semantic web concepts

Some Recap Points

- Expressiveness
- Federation & orchestration
- Case study, application benefits
- Drilled down
  - Esp.: Textual
  - Also: Higher-order, restraint, tabling, defeasibility, uncertainty, general quantified formulas

Ontologies in Rulelog

- Ontology is a purpose and aspect within KRR, not a language or expressive subset
  - Some expressive subsets of KRR are esp. common and useful for ontology
- The most frequent kinds of ontological modeling/KRR are doable <u>within</u> Rulelog
  - Using the same overall KRR language/tools as non-ontological rules etc.
  - Advantage: <u>reduce effort required for system integration</u>
  - Rulelog expressiveness <u>supersumes</u> RDF-S, OWL-RL, and most of OWL-DL
  - Rulelog strength: computational <u>scalability</u>, e.g., compared to OWL-DL
  - E.g., Ergo imports & translates those into Rulelog
    - Still an open research direction: maximizing treatment of OWL-DL
- Rulelog has pre-defined <u>frame syntax</u> – in object-oriented style, which is good for basic ontological modeling. E.g., wolf :: animal. // subclass
  - Subclass hierarchy, property domain/range, subproperty hierarchy, inheritance along classes, property cardinality (using integrity constraints)
- Rulelog strength: <u>mappings</u> between ontologies
  - Perspective: most of a dictionary's content is about mappings between words
- Rulelog strength: <u>textual</u> and higher-order, e.g., to capture NL terminologies
- Rulelog strength: <u>defeasibility</u>, which increases reuse, e.g., in inheritance
- Rulelog enables methodology of "<u>as-you-go</u>" ontological KA
  - Reduce effort waste, delay. risk

Explanations in Rulelog

- Rulelog has methods for powerful explanations
  - E.g., illustrated in Reg W case study using Ergo system
  - Fully detailed, automatically generated in English, interactively navigable
  - Comprehensible to ordinary users
  - Handles also: why-**not**, and argumentation

- Intrinsic strength: logical; and potentially close to NL abstractions
  - Contrast with most machine learning (ML) methods – esp. neural networks!
    - Heavily numerical, large fanout, often/typically far from NL
      $\rightarrow$ $\rightarrow$ cognitively difficult for humans to understand

- Intrinsic strength: natural deduction style proofs
  - Expressive transformations produce within the reasoning procedures a series of LP style steps, each deriving a head literal from a conjunction of literals
  - Most classical-logic proof procedures lack natural deduction style
    - E.g., resolution, tableau

- Textual templates work well for text generation

Machine Learning with Rulelog

- Goal for whole field of AI: bring together ML with KRR
- ML methods typically produce uncertain/probabilistic knowledge
- Machine learning (ML) is a motivation for uncertainty/probability in Rulelog
    - It has been a driver for work on probabilistic LP

- Rulelog can complement ML in several ways
    - Import knowledge from ML: at load time or via external querying
    - Integrate knowledge from multiple ML techniques/systems, episodes/datasets
    - Integrate learned knowledge with human-authored knowledge
    - Feed inferred conclusions as data to ML systems
    - Pose questions, with control parameters, to ML systems

- Rulelog often humans to specify complex knowledge concisely, as in NL
    - Say it once
    - … rather than via labeling tons of examples for a ML method
      (and providing other supervision to ML)

Standardization of Rulelog

- Rulelog is on standardization path
  - Through RuleML with W3C and Oasis
- Existing draft of large subset as RIF dialect
  - Under RIF Framework for Logical Dialects (FLD)

- Rulelog is an expressive superset of several RIF dialects and OWL profiles
  - E.g., RIF-Core, OWL-RL

- Also relevant to explore relationships to:
  - LegalRuleML
  - HiLog feature used in ISO Common Logic
  - NLP used in SBVR

Rulelog KRR: 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

Some Other Case Studies

- Financial regulatory compliance decisions:
  with databases/ontologies

- Health care treatment guidance:
  making decisions with policy-flavored protocols

- Defense intelligence analysis:
  with text extraction, databases/ontologies

- Personalized tutoring in continuing/higher ed:
  answering science questions

- E-commerce marketing:
  with product databases/ontologies, promotion/pricing policies

- *Info on most of these is available at Coherent Knowledge website*

Lessons Learned from Case Studies

Users in these multiple domains benefited from:

- Agility: Flexibility and ease of authoring, fast updating

- High accuracy and transparency

  - Explanations and provenance

  - Lower risk of non-compliance or confusion

- More Cost Effectiveness – less labor, SMEs in closer loop

- Leveraging investment in semantic tech: RDF, SPARQL, OWL

Dream for field of AI:
Answer Questions using logic **and** NLP

- What if it was "*cheap*" to acquire massive volumes of knowledge formally encoded as logical formulas?
  - Say, only a small integer multiple of cost to write quality text, i.e., NL sentences, e.g., about science, policies, medicine, …

- What if it was "*easy*" to understand natural language questions well enough to exploit such formal encodings?

- *How much could Textual Rulelog help?*

Rulelog enriches Text Extraction

- Leverage Rulelog's high expressiveness and flexibility

- Mappings between multiple terminologies or ontologies

"moving a bomb" — *implies* → "transporting weaponized material"
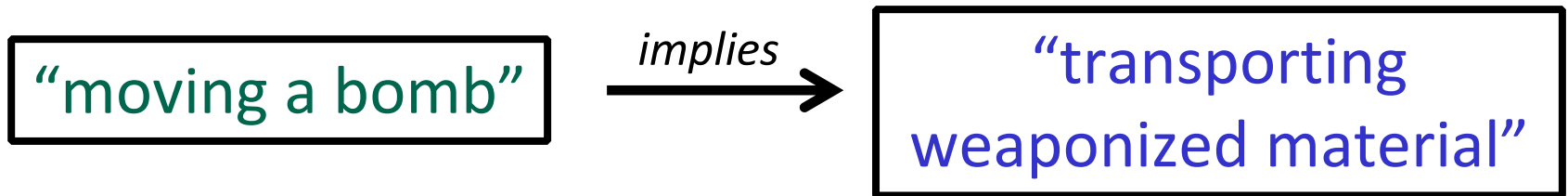
Chatbots, HCI, IoT, and NL Understanding

- *The Promise:* Able to converse with and assist humans in many facets of our lives
  - Provide advice
  - Perform tasks
  - Inform us proactively
  - Explain why

- *Required to fulfill this promise:* Flexible deep reasoning
  - Using logical/probabilistic knowledge representation and reasoning (KRR)
  - Combined with natural language processing (NLP) and machine learning (ML)
  - Treat the **deep** semantics of NL
  - KRR was central to first wave of AI success. KRR + ML = core of AI.

Open Research Topics: Logical (I)

- Reactive: semantics, event handling/dispatching

    - Good candidate to integrate into Rulelog: Production LP approach (see our AAAI-13 rules tutorial)

    - Relate to Reaction RuleML, Prova, production/ECA rules, Transaction Logic

- Probabilistic: distribution semantics – optimization of restricted cases, hookups to ML approaches

- Reasoning by cases: theory/semantics, algorithms

    - Do selectively. Soundness/relationship to: FOL, ASP, MKNF.

- Hypothetical reasoning, abduction

- Distributed reasoning: algorithms and testbeds

    - Finely parallelized too. Leverage persistent stores.

Open Research Topics: Logical (II)

- Explanation methods – improve further

- Tools for debugging knowledge – improve further

- Argumentation rules to work better with omnis

- Equality: axiomatic semantics, efficient algorithms

- Aggregates – handle indefiniteness, unstratified cases

- "Constraints" – cf. constraint LP: theory, algorithms

- Optimizations: e.g.,
    - subgoal re-ordering for efficiency
    - leverage subsumptive tabling (cf. LP)
    - external query plans

Research Directions – Other Aspects

❖ Combine closely with deep semantic NL interpretation

❖ Complement ML: feed inferences to ML, query ML
    ❖ Rulelog deduction can also directly be inductive ML

❖ Applications
- NLP and HCI, e.g., for cognitive, IoT
- Legal. Medical.
- Defense. Education. Social media. *Many more.*

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

Outline of Tutorial

A. Overview

  ➢ Practical logic. Applications. Features. Software. Textual.

  ➢ Case Study Demo and Features Tour

    ➢ Financial regulatory/policy compliance

B. Concepts and Foundations: Drill-downs

  ➢ Expressive features, semantics, algorithms;
     relationships to natural language and machine learning

C. Conclusions and Future Work

– Background Assumed: basic knowledge of first-order logic,
  databases. Helpful: XML, RDF and semantic web concepts

Thank You

Disclaimer: All brands, logos and products are trademarks or registered trademarks of their respective companies.