

WELCOME! to the AAAI-17 Tutorial

Rulelog: Deep KRR for Cognitive Computing

*presented by Benjamin Groszof**

*also authored by Michael Kifer**, and Paul Fodor***

* **Accenture** ; work primarily done while formerly CTO at Coherent Knowledge

<http://benjamingroszof.com>

** *Sony Brook University, and Coherent Knowledge*

<http://www.cs.stonybrook.edu/~kifer>

<http://www.cs.stonybrook.edu/~pfodor>

Note on Copyright and Permissions:

Slides with Coherent Knowledge logo are, in whole or part, courtesy of Coherent Knowledge; and several other slides as well are, in whole or part, courtesy of Coherent Knowledge.

Preface ; For More Info

- Previous tutorials on Rulelog and its forerunners were presented at IJCAI-16, RuleML-2016, RuleML-2015, AAI-13, and ~9 other times at conferences since 2004.
- This slideset is available on the web, at <http://benjamingrosof.com/misc-publications/#AAAI17RulelogTutorial> , and via the AAI-17 Tutorials page
- For more info beyond the slideset: see the authors' websites and <http://coherentknowledge.com/publications>
- References are in the short proceedings paper of the RuleML-2015 tutorial, and in the AAI-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, entrepreneur, and executive
- Principal Director & Research Fellow 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. Helpful: XML, RDF and semantic web concepts

Concept of logical Knowledge Representation (KR)

- A KR S is defined as a triple $(LA, LC, |=)$, where:
 - LA is a formal language of sets of assertions (i.e., premise expressions)
 - LC is a formal language of sets of conclusions (i.e., conclusion expressions)
 - LC is not necessarily even a subset of LA . E.g., in LP and Rulelog.
 - $|=$ is the entailment relation.
 - $\text{Conc}(A, S)$ stands for the set of conclusions that are entailed in KR S by a set of premises A
 - We assume here that Conc is a functional relation.
- Typically, e.g., in FOL and declarative Logic Programs, entailment is defined formally in terms of models, i.e., truth assignments that satisfy the premises and meet other criteria.

Practical Logic, vs. Classical Logic

- Support IT, not mathematics
- Databases
- Rules
- Scalable computationally
- Robust in face of human errors and miscommunications
- Thus: Humble -- avoid general reasoning by cases and general proof by contradiction

What is “reasoning by cases”: (background)

Assertions: if A then C. if B then C. A or B.

Conclude: C.

Main Kinds of Practical Logic

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

More Practical Logic Context for Rulelog

- Also subsets of LP and thus of Rulelog:
 - Databases
 - Production rules, ECA rules, Prolog (their logical subsets)
 - OWL-RL (Rules profile)
- “Smart Data” is hot in industry:
 - Graph / linked database, with explicit schemas and a bit of semantics. Also as input to machine learning.
- Next step: “Smart Rules”, using Rulelog and subsets
 - Rules that chain, deeper reasoning and semantics, meta flexibility with scalability
 - Enterprises leverage investments in smart data
- Keys: semantics, agile schema, meta data (incl. linking), meta knowledge; simplicity, flexibility, reusability

Semantic

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

- Relational DB was 1st successful semantic tech
- LP theory was invented to formalize it and unify it with the pure subset of Prolog

Semantic Rules: Differences from Rules in the 1980's / Expert Systems Era

- Get the KR right (knowledge representation)
 - More mature research understanding
 - Semantics independent of algorithm/implementation
 - Cleaner; avoid general programming/scripting language capabilities
 - Highly scaleable performance; better algorithms; choice for interoperability
 - Highly modular wrt updating; use prioritization
 - → Higher practical expressiveness
 - → Highly dynamic, scaleable rulebase authoring: distributed, integration, partnering
- Leverage Web, esp. XML
 - Interoperable syntax (e.g., RuleML, RIF)
 - Merge knowledge bases
- Embeddable
 - Into mainstream software development environments (Java, C++, C#); not its own programming language/system (cf. Prolog)
- Knowledge Sharing: intra- or inter- enterprise
- Broader set of Applications

Value of Rules as form of KR

- Rules as a form of KR (*knowledge representation*) are especially useful
 - relatively mature from basic research viewpoint
 - good for prescriptive specifications (vs. descriptive)
 - a restricted programming mechanism
 - integrate well into commercially mainstream software engineering, e.g., OO and DB
 - easily embeddable; familiar
 - vendors interested already: Webizing, application development tools
- ⇒ Identified as part of mission of the W3C Semantic Web Activity, in about 2001

Declarative Logic Programs (LP) is the Core KR in today's world ... including the Semantic Web

- **LP is the core KR of structured knowledge management today**

- **Databases**

- Relational, semi-structured, RDF, XML, object-oriented
- SQL, SPARQL, XQuery
- Each fact, query, and view is essentially a rule



- **Semantic Rules**

- RuleML standards design, including ...
- Rule Interchange Format (RIF) from W3C
 - Framework for Logic Dialects (FLD); also RIF-BLD, RIF-Core, SWRL
- RIF-Rulelog draft
- Legal RuleML draft



- **Semantic Ontologies**

- RDF-Schema and OWL-RL (= the Rules subset). E.g., Oracle's implementation.



Overview of Rulelog: Highly Expressive

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

Overview of Rulelog (II)

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

Overview of Rulelog (III)

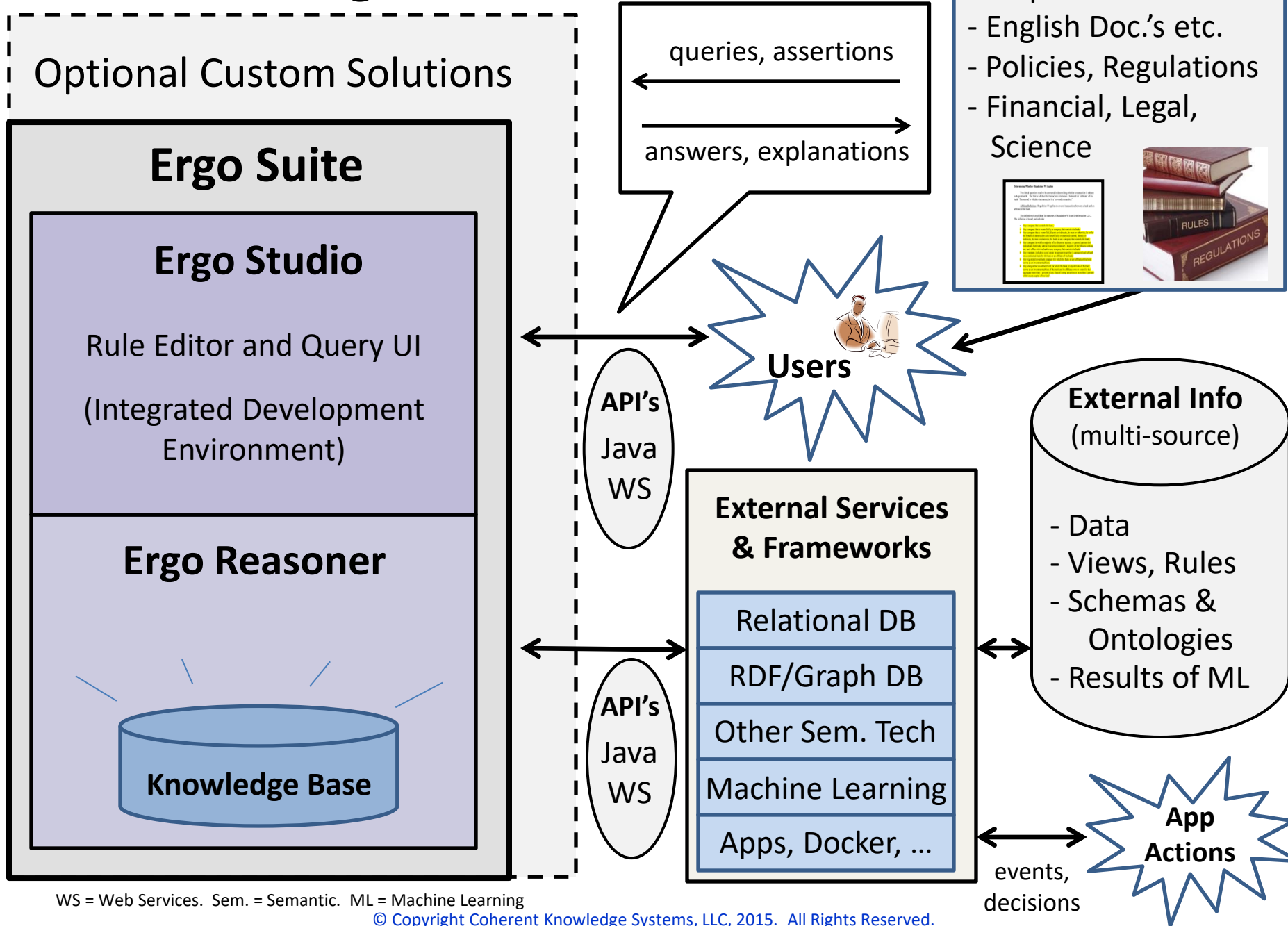
- Closely integrates with OWL-DL ontologies
- Closely integrates with natural language processing
 - Text interpretation: map text to logic
 - Text generation: map logic to text
- Closely integrates with machine learning (ML)
 - Import ML results as probabilistic knowledge
 - Export conclusions to ML

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

Rulelog: Software Tools

- ❖ Lots of Rulelog expressiveness:
 - Ergo Lite: Large subset of Rulelog. Open source.
 - A.k.a. originally as Flora-2
 - Ergo (from Coherent): Most of Rulelog. Has IDE.
 - Available [free for academic research use](#) ; and [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 Architecture



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

KRR Features Comparison: Rulelog Shines

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

Ergo is based on Textual Rulelog

- *Rulelog* is a kind of knowledge representation and reasoning (KRR)
 - A major research advance in KRR theory & algorithms, which culminated in 2012
- **Ergo is the most complete & highly optimized implementation available**
- Rulelog features very high/flexible expressiveness: logical chaining, higher-order, general quantified formulas, defeasibility/exceptions, provenance, probabilistic, restraint bounded rationality, and more
- Yet Rulelog scales well: reasoning is polynomial time (as in databases)
- *Textual* Rulelog extends Rulelog with natural language processing (NLP)
- Logic itself is utilized to map between logic syntax and English syntax
- ErgoText templates aid knowledge authoring and explanation generation
- *More background:* Rulelog adds “full meta” expressiveness to *Datalog*
 - Datalog is the logic of databases, business rule systems (production/ECA/Prolog), semantic web ontologies, and earlier-generation semantic web rules (e.g., SWRL and RIF-BLD)
 - Rulelog extends also declarative logic programs (LP)

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)

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

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,...

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 Cost Effective – less labor; subject matter experts in closer loop
- More Agile – faster to update
- More Overall Effectiveness: less exposure to risk of non-compliance



Demo of Ergo Suite for Compliance Automation: US Federal Reserve Regulation W

- EDM Council Financial Industry Consortium
Proof of Concept – **successful and touted pilot**
 - Enterprise Data Management Council (Trade Assoc.)
 - Coherent Knowledge Systems (USA, Technology)
 - SRI International (USA, Technology)
 - Wells Fargo (Financial Services)
 - Governance, Risk and Compliance Technology Centre (Ireland, Technology)
- Reg W regulates and limits \$ amount of transactions that can occur between banks and their affiliates. Designed to limit risks to each bank and to financial system.
- Must answer 3 key aspects:
 1. *Is the transaction's counterparty an affiliate of the bank?*
 2. *Is the transaction contemplated a covered transaction?*
 3. *Is the amount of the transaction permitted ?*

Determining Whether Regulation W Applies

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

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

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

- Any company that controls the bank;
- Any company that is controlled by a company that controls the bank;
- Any company that is controlled, directly or indirectly, by trust or otherwise, by or for the benefit of shareholders who beneficially or otherwise control, directly or indirectly, by trust or otherwise, the bank or any company that controls the bank;
- Any company in which a majority of its directors, trustees, or general partners (or individuals exercising similar functions) constitute a majority of the persons holding any such office with the bank or any company that controls the bank;
- Any company, including a real estate investment trust, that is sponsored and advised on a contractual basis by the bank or an affiliate of the bank;
- Any registered investment company for which the bank or any affiliate of the bank serves as an investment adviser;
- Any unregistered investment fund for which the bank or any affiliate of the bank serves as an investment adviser, if the bank and its affiliates own or control in the aggregate more than 5 percent of any class of voting securities or more than 5 percent of the equity capital of the fund¹;

The Starting Point - Text of Regulation W

Demo goes here

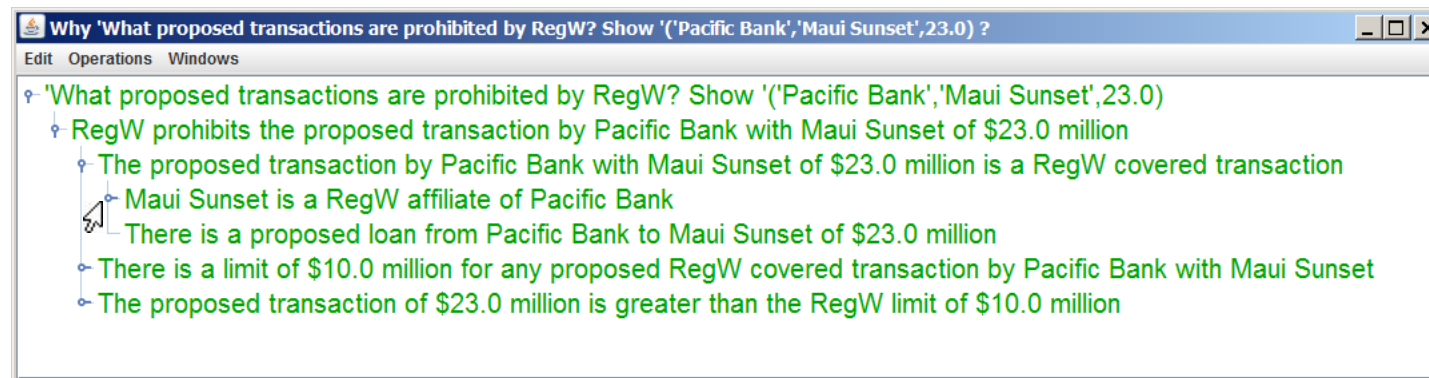
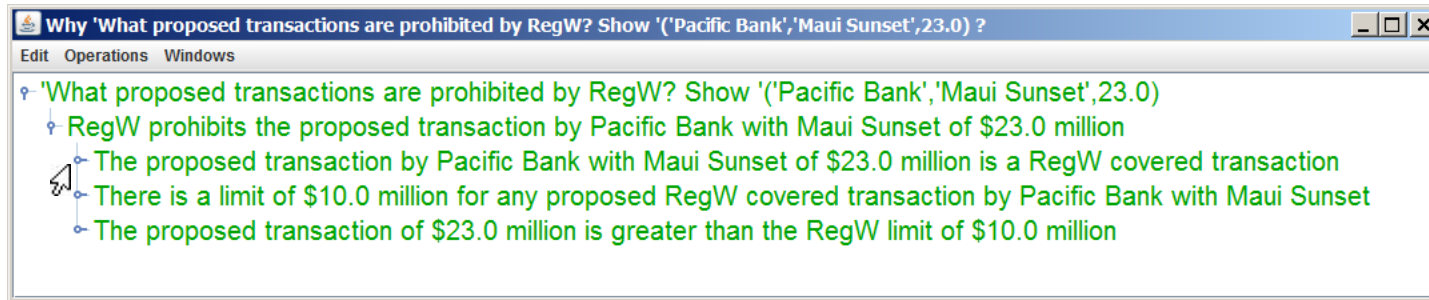
- The next 3 slides show screenshots from it

Query is asked in English

The screenshot shows a software window titled "Coherent Query". The menu bar includes "Edit", "Restraint", "View", "Explain", "History", and "Windows". On the right side of the window, there are three buttons: "Execute", "Pause", and "Stop". The main text area contains the query: "What proposed transactions are prohibited by RegW? Show ' (?Bank,?Company,?Amount)." Below the query is a table with three columns: "?Bank", "?Company", and "?Amount". The first row of data shows "'Pacific Bank'", "'Maui Sunset'", and "23.0". A red circle highlights the "Why?" button, which has opened a dropdown menu with the options "Explanation Game" and "See answer term as tree".

?Bank	?Company	?Amount
'Pacific Bank'	'Maui Sunset'	23.0

User Clicks the handles to expand the Explanations



Why is the proposed transaction prohibited by Regulation W?

1. *Is the transaction's counterparty an "affiliate" of the bank?*

YES.

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

Edit Operations

- 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
 - Hawaii Bank is a RegW affiliate of Pacific Bank
 - There is common control of Hawaii Bank and Pacific Bank
 - Hawaii Bank is controlled by Americas Bank
 - Hawaii Bank is a subsidiary of Americas Bank
 - Pacific Bank is controlled by Americas Bank
 - Pacific Bank is a subsidiary of Americas Bank
 - Maui Sunset is advised by Hawaii 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

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).
```



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/EC A 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: Humagic Knowledge

- Humagic = human-machine logic
- 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
- Logic-syntax sentence may have 1 or more NL-syntax sentences associated with it
 - E.g., that result from text generation on it
- Other sentences can be in a mix of NL-syntax and logic-syntax
 - ErgoText: templates used for text interpretation and text generation

Textual extension of Rulelog (I)

- Leverage Rulelog to much more simply and closely map between natural language (NL) and logic
- Rulelog's high expressiveness is much closer to conceptual abstraction level used in NL
- English sentence \leftrightarrow Rulelog sentence (rule)
- Textual terminology:
 - English phrase \leftrightarrow logical term in Rulelog
 - English word \leftrightarrow logical functor in Rulelog
 - Basis for textual templates

Textual Rulelog (TR) – approach (II)

- TR text interpretation:
Rulelog rules map from NL to logic
- TR text generation:
Rulelog rules map from logic to NL
- TR terminology mapping:
Rulelog rules map between phrasings and ontologies – in NL or logic
 - “moving a bomb” implies “transporting weaponized material”
 - `isBomb(?x)` implies `rdftriple(?x,rdftype,bomb)`

Ergo Makes Sentences Executable

- If *something* is true then *something else* must be true.

Written as:

something_else :- *something*

Diagram illustrating the Ergo sentence structure: *something_else* :- *something*. The word "then" is shown in a box above the colon, and the word "if" is shown in a box above the right-hand side of the sentence.

- Example of executable Ergo sentence:

\(The individual affiliate threshold for transaction under Regulation W
by ?Bank with ?Counterparty is ?Amount\) :-

\(?Counterparty is deemed an affiliate of ?Bank under Regulation W\) \and
\(?Bank has capital stock and surplus ?Capital\) \and
\(the threshold percentage for an individual affiliate is ?Percentage\) \and
?Amount \is ?Capital * ?Percentage/100.

ErgoText

- ErgoText:

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

- ErgoText Template:

```
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

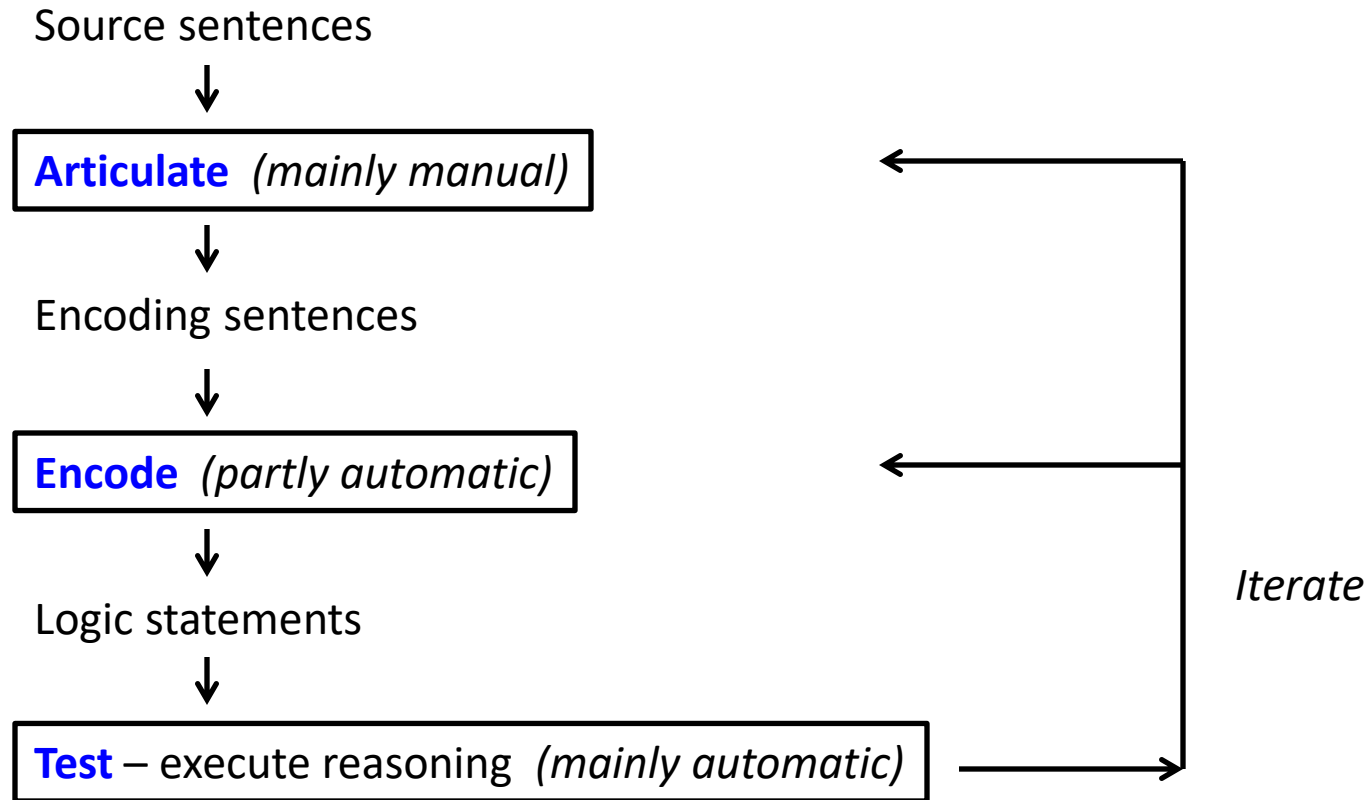
Textual Rulelog (III)

- Almost any NL sentence can be represented as a logical sentence
 - Leverages the logical quantifiers feature of Rulelog
 - Ex.: “each large company has some talented CEO”
 - forall(?x)^((?x \isa \large company\)) ==>
exists(?y)^((\?x has ?y\) \and
(?y \isa \talented CEO\))).

Knowledge Authoring Process using Textual Rulelog

- Start with source text in English – e.g., textbook or policy guide
 - A sentence/statement can be an assertion or a query
- Articulate: create encoding sentences (text) in English.
As necessary:
 - Clarify & simplify – be prosaic and grammatical, explicit and self-contained
 - State relevant background knowledge – that's not stated directly in the source text
- Encode: create executable logic statements
 - Each encoding text sentence results in one executable logic statement (“rules”)
 - Use IDE tools and methodology
- Test and debug, iteratively
 - Execute reasoning to answer queries, get explanations, perform other actions
 - Find and enter missing knowledge
 - Find and fix incorrect knowledge
 - Optionally: further optimize reasoning performance, where critical

Knowledge Authoring Steps using Textual Rulelog



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

Health Care Case Study: Task Domain

- Task: Treatment Guidance for
 - Delivery of care, e.g., by medical staff or self-service
 - Insurance
 - Oversight of quality (e.g., “care measurement”)
- Guidance takes form of policies
 - Portions are often based closely on clinical studies
 - Top ~100 diseases have “protocols” written up in considerable detail

Kinds of Domain Knowledge & Reasoning

- Knowledge & reasoning about:
 - Patient characteristics and history
 - Symptoms
 - Diseases and diagnoses
 - Drug treatments
 - Non-drug treatments
 - Medical tests
 - Intended effects
 - Side effects
 - Interactions between treatments, e.g., drug-drug
 - Risks
 - Interactions between risks; aggravation and severity of risks

Challenges & Requirements

- Challenge: personalization
- Patients undergo multiple diseases and treatments, but protocols are developed, based on clinical studies, for
 - One disease (e.g., diagnosis) at a time
 - One treatment (e.g., drug) at a time
- Requirements, both beforehand and post-play, for
 - Correctness / competence
 - Maximize benefit to patient
 - Minimize harm to patient, incl. avoid potential treatment errors
 - Minimize costs
 - Verifiability, therefore
 - Explainability to:
 - Medical staff performing care delivery – e.g., combat “alert fatigue”
 - Patients – e.g., improve compliance by knowing why to avoid an easy-to-obtain drug
 - Insurers
 - Oversight staff, incl. for audits

Treatment Scenario

- A busy intern encounters an elderly woman in a rehabilitation facility complaining of knee pain.
- What treatment should be given -- or not given – and why?
- EHR records show:
 - The elderly woman is currently taking Coumadin to treat the pre-existing condition of atrial fibrillation which increases the risk of blood clot and stroke.
- *Automatically gives both alerts and educates.*



DEMO GOES HERE

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: Virtual Rulelog

- Rulelog orchestrates overall federated reasoning by sub-goaling dynamically
- A variety of other structured information systems are virtualized as Rulelog via Ergo federation connectors, which import/query and translate

Kinds of Virtual 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
- Web services: via XML connector (JSON is under dev)
- 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.

Ergo RDF/OWL

Help

Ergo RDF&OWL Import Tool

Original RDF/OWL file: WorldBank.ttl Ergo file: WorldBank.ttl.ergo

Status: Done translating WorldBank.ttl

Select input:

- ☐ Import RDF/OWL N-triples or N-quads file (.nq, .nt)
- ☐ Import RDF/OWL N-triples or N-quads directory
- ☐ Import RDF/OWL XML file (.rdf, .owl, .xml)
- ☐ Import RDF/OWL XML directory
- ☐ Import JSON-LD file (.jsonld)
- ☐ Import JSON-LD directory
- ☒ Import RDF/OWL Turtle file (.ttl)
- ☐ Import RDF/OWL Turtle directory

Input file: WorldBank.ttl

Output predicate arity (n-quads or n-triples):

- ☒ n-triples
- ☐ n-quads

Output quad's graph name ('main' is the default)

Output format (fastload .P or .ergo):

- ☒ fastload format
- ☐ predicate syntax: p(s,o) or p(s,o,g)
- ☐ frame syntax: s[p->o]

Manage IRIs:

xsd = http://www.w3.org/2001/XMLSchema#
rdf = http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs = http://www.w3.org/2000/01/rdf-schema#
owl = http://www.w3.org/2002/07/owl#

Import RDF/OWL

```
@prefix void: <http://rdfs.org/ns/void#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix qb: <http://purl.org/linked-data/cube/> .
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .
@prefix sd: <http://www.w3.org/ns/sparql-service-description#> .
@prefix : <http://worldbank.270a.info/void.ttl#> .
@prefix worldbank-graph: <http://worldbank.270a.info/graph/> .
@prefix oecd-dataset: <http://oecd.270a.info/dataset/> .
@prefix bfs-dataset: <http://bfs.270a.info/dataset/> .
@prefix fao-dataset: <http://fao.270a.info/dataset/> .
@prefix ecb-dataset: <http://ecb.270a.info/dataset/> .
@prefix imf-dataset: <http://imf.270a.info/dataset/> .
@prefix uis-dataset: <http://uis.270a.info/dataset/> .
@prefix frb-dataset: <http://frb.270a.info/dataset/> .
@prefix worldbank-dataset: <http://worldbank.270a.info/dataset/> .
@prefix transparency-dataset: <http://transparency.270a.info/dataset/> .

<http://csarven.ca/#>
  rdfs:label "Sarven Capadisli"@en ;
  .

<http://creativecommons.org/publicdomain/zero/1.0/>
  rdfs:label "CC0 1.0 Universal"@en ;
  .

<http://worldbank.270a.info/void.ttl#>
  a void:DatasetDescription ;
  dcterms:title "A Void Description of the worldbank.270a.info Dataset" ;
```

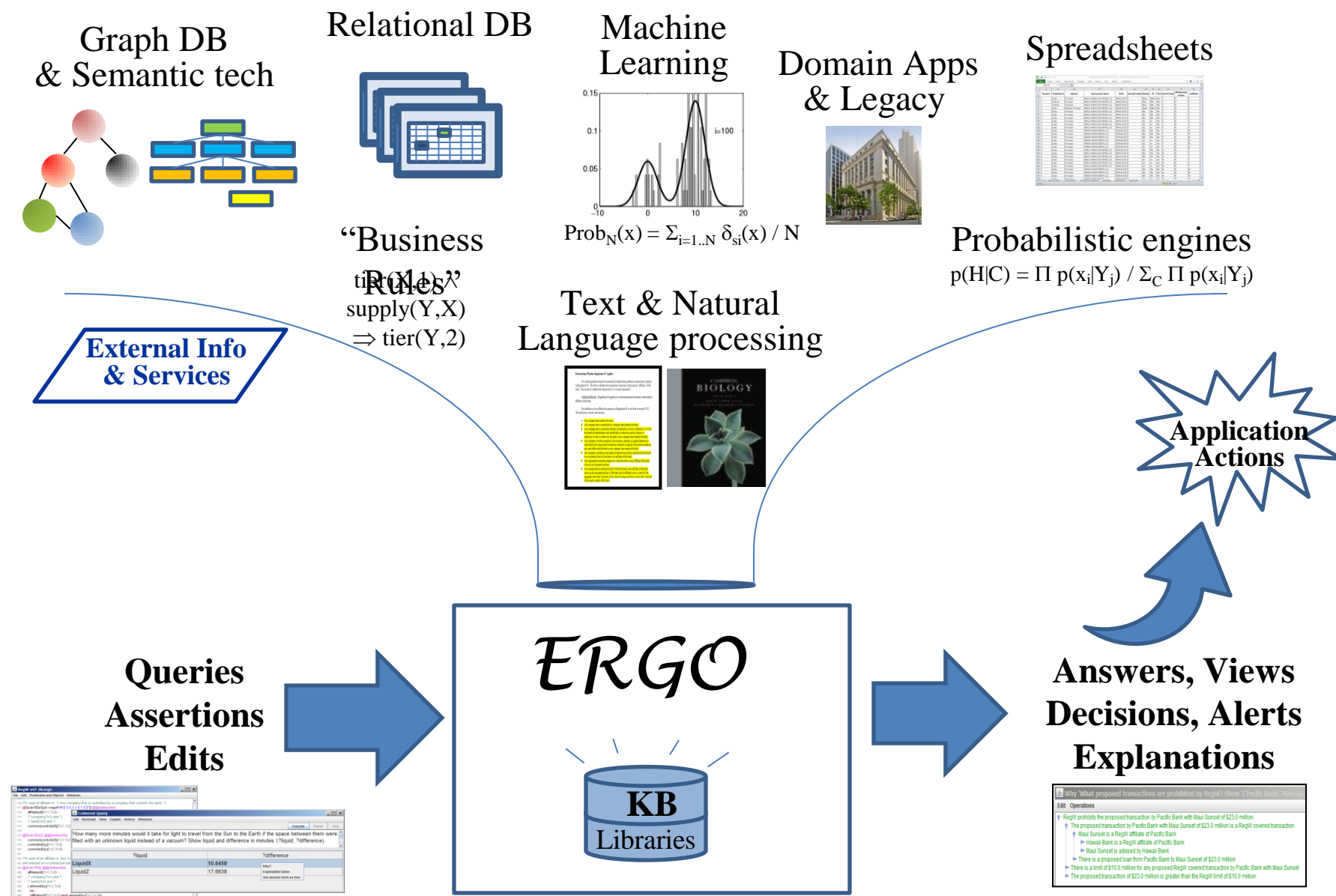
#deffast xsd http://www.w3.org/2001/XMLSchema#
#deffast rdf http://www.w3.org/1999/02/22-rdf-syntax-ns#
#deffast rdfs http://www.w3.org/2000/01/rdf-schema#
#deffast owl http://www.w3.org/2002/07/owl#

% imported OWL axioms

```
'http://rdfs.org/ns/void#entities'('Bb38eba1f27de68147b4ed800deeca630')
'http://rdfs.org/ns/void#class'('Bb38eba1f27de68147b4ed800deeca630')
'http://rdfs.org/ns/void#triples'('Bd43452bbb1eb87dc80d56d1c001f106')
'http://rdfs.org/ns/void#property'('Bd43452bbb1eb87dc80d56d1c001f1')
'http://rdfs.org/ns/void#distinctSubjects'('Bd43452bbb1eb87dc80d56d1')
'http://rdfs.org/ns/void#distinctObjects'('Bd43452bbb1eb87dc80d56d1')
'http://rdfs.org/ns/void#triples'('Bf7753516cd20cb7f77df061010915387')
'http://rdfs.org/ns/void#property'('Bf7753516cd20cb7f77df061010915387')
'http://rdfs.org/ns/void#distinctSubjects'('Bf7753516cd20cb7f77df06101')
'http://rdfs.org/ns/void#distinctObjects'('Bf7753516cd20cb7f77df06101')
'http://rdfs.org/ns/void#triples'('Bcf6bcafdc90833f622e5bb10c95d4d14')
'http://rdfs.org/ns/void#property'('Bcf6bcafdc90833f622e5bb10c95d4d14')
'http://rdfs.org/ns/void#distinctSubjects'('Bcf6bcafdc90833f622e5bb10c')
'http://rdfs.org/ns/void#distinctObjects'('Bcf6bcafdc90833f622e5bb10c')
'http://rdfs.org/ns/void#triples'('B3eef943acdd45aecbebacdd21158b10')
'http://rdfs.org/ns/void#property'('B3eef943acdd45aecbebacdd21158b10')
'http://rdfs.org/ns/void#distinctSubjects'('B3eef943acdd45aecbebacdd2')
'http://rdfs.org/ns/void#distinctObjects'('B3eef943acdd45aecbebacdd2')
'http://rdfs.org/ns/void#triples'('Be8f34857a86f0bce3671e0fb6acb0f7d')
'http://rdfs.org/ns/void#property'('Be8f34857a86f0bce3671e0fb6acb0f7d')
'http://rdfs.org/ns/void#distinctSubjects'('Be8f34857a86f0bce3671e0fb6')
'http://rdfs.org/ns/void#distinctObjects'('Be8f34857a86f0bce3671e0fb6')
'http://rdfs.org/ns/void#triples'('Bd81143ffc178de642750be48bdfa8ad3')
'http://rdfs.org/ns/void#property'('Bd81143ffc178de642750be48bdfa8a')
'http://rdfs.org/ns/void#distinctSubjects'('Bd81143ffc178de642750be48')
'http://rdfs.org/ns/void#distinctObjects'('Bd81143ffc178de642750be48')
'http://rdfs.org/ns/void#triples'('B71e4380c4b52f4b64169b767fbcaf48')
```

@Copyright 2015, Coherent Knowledge Systems, Ergo/OWL translator version 0.7.19 (July 19, 2015)

Actively Reason over Today's Gamut of Knowledge



Other Case Studies

- Financial regulatory compliance decisions:
with databases/ontologies
- 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

Lessons Learned from Case Studies

Customers 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

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

Key features

1. Horn LP, with Functions
2. Well-Founded Negation
3. Tabling Algorithms for LP
4. **Restraint**: semantic bounded rationality
5. Frame syntax (a.k.a. F-Logic), Object Oriented style
6. **Higher-Order** Syntax via Hilog. Reification.
7. Rule ID's
8. **Defeasibility** via Argumentation Rules. Remediating FOL's Fragility.
9. **General Formulas**, Existentials and Skolems, Omni-directional Disjunction
 - Representing Text. Importing full OWL. FOL-Soundness.
10. **Probabilistic** knowledge and reasoning
11. External Querying
12. Reactiveness
13. Misc. Lesser Features: Datatypes, Aggregation, Integrity Constraints, Inheritance, Equality, "Constraints"
14. Terminology/Ontology Mapping
15. Justification/Explanation

Horn FOL

- The Horn subset of FOL is defined relative to clausal form of FOL
- A Horn clause is one in which there is at most one positive literal.

It takes one of the two forms:

1. $H \vee \neg B_1 \vee \dots \vee \neg B_m$. A.k.a. a definite clause / rule

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

2. $\neg B_1 \vee \dots \vee \neg B_m$. A.k.a. an integrity constraint

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

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

- Rule := $H \Leftarrow B_1 \wedge \dots \wedge B_m$. where $m \geq 0$, H and B_i 's are atoms
head if body ;

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

- $\perp \Leftarrow B_1 \wedge \dots \wedge B_m$. A.k.a. empty-head rule (\perp is often omitted).

For refutation theorem-proving, represent a negated goal as (2.).

*Horn **LP** Syntax and Semantics*

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

Example of Horn LP vs. Horn FOL

- Let P be:
 - $\text{DangerousTo}(\text{?x}, \text{?y}) \text{ :- } \text{PredatorAnimal}(\text{?x}) \wedge \text{Human}(\text{?y}).$
 - $\text{PredatorAnimal}(\text{?x}) \text{ :- } \text{Lion}(\text{?x}).$
 - $\text{Lion}(\text{Simba}).$
 - $\text{Human}(\text{Joey}).$
- $I1 = \{\text{Lion}(\text{Simba}), \text{Human}(\text{Joey})\}$
- $I2 = \{\text{PredatorAnimal}(\text{Simba}), \text{Lion}(\text{Simba}), \text{Human}(\text{Joey})\}$
- $I3 = \{\text{DangerousTo}(\text{Simba}, \text{Joey}), \text{PredatorAnimal}(\text{Simba}), \text{Lion}(\text{Simba}), \text{Human}(\text{Joey})\}$
- $I4 = I3$. Thus $M(P) = I3$.
- Let P' be the Horn FOL rulebase version of P above, where \Leftarrow replaces :- .
- Then the ground atomic conclusions of P' are exactly those in M(P) above.
- P' also entails various non-ground-atom conclusions, including:
 1. Non-unit derived clauses, e.g., $\text{DangerousTo}(\text{Simba}, \text{?y}) \Leftarrow \text{Human}(\text{?y}).$
 2. All tautologies of FOL, e.g., $\text{Human}(\text{?z}) \vee \neg \text{Human}(\text{?z}).$
 3. Combinations of (1.) and (2.), e.g., $\neg \text{Human}(\text{?y}) \Leftarrow \neg \text{DangerousTo}(\text{Simba}, \text{?y}).$

Horn LP Compared to Horn FOL

- Fundamental Theorem connects Horn LP to Horn FOL:
 - $M(P) = \{\text{all ground atoms entailed by } P \text{ in Horn FOL}\}$
- Horn FOL has additional non-ground-atom conclusions, notably:
 - non-unit derived clauses; tautologies
- Can thus view Horn LP as the f-weakening of Horn FOL.
 - “f-” here stands for “fact-form conclusions only”
 - A restriction on form of conclusions (not of premises).
- Horn LP – differences from Horn FOL:
 - Conclusions $\text{Conc}(P)$ = essentially a set of ground atoms.
 - Can extend to permit more complex-form queries/conclusions.
 - Consider Herbrand models only, *in typical formulation and usage*.
 - P can then be replaced equivalently by $\{\text{all ground instantiations of each rule in } P\}$
 - But can extend to permit: extra unnamed individuals, beyond Herbrand universe
 - Rule has non-empty head, *in typical formulation and usage*.
 - Can extend to detect violation of integrity constraints

The “Spirit” of LP

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

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

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

- Nonmonotonicity – defaults, incl. NAF
- Procedural attachments, esp. external actions

Requirements Analysis for Logical Functions

- Function-free is a commonly adopted restriction in practical LP/Web rules today
 - DB query languages: SQL, SPARQL, XQuery
 - RDFS
 - Production rules, and similar Event-Condition-Action rules
 - OWL
- BUT functions are often needed for Web (and other) applications. Uses include:
 - HiLog and reification – higher-order syntax
 - For meta-reasoning, e.g., in knowledge exchange or introspection
 - Ontology mappings, provenance, KB translation/import, multi-agent belief, context
 - KR macros, modals, reasoning control, KB modularization, navigation in KA
 - Meta-data is important on the Web
 - Skolemization – to represent existential quantifiers
 - E.g., RDF blank nodes
 - Convenient naming abstraction, generally
 - `steering_wheel(my_car)`

Functions in LP Lead to Undecidability; but Restraint Solves this

- Functions lead to undecidability, due to potentially infinite number of conclusions
- Example:
 - Assert: `num(succ(?x)) :- num(?x). num(0).`
 - Conclusions: `num(0), num(succ(0)), num(succ(succ(0))), ...`
- In Rulelog, restraint bounded rationality solves this
 - Specify radial restraint with radius of 3, for example
 - Then `num(succ(succ(succ(succ(0))))), ...` all have truth value u
- For more info on restraint, see
 - AAI-13 paper “Radial Restraint: A Semantically Clean Approach to Bounded Rationality” by B. Grosz and T. Swift
 - RuleML-2013 paper “Advanced Knowledge Debugging for Rulelog” by C. Andersen et al.
 - Both are available at <http://coherentknowledge.com/publications/>

Well Founded Semantics for LP

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

Tabling Algorithms for LP & Rulelog

- Builds and maintains a forest of saved subgoal attempts and results
- Thus heavily caches. Is mixed-direction, not just backward-direction.
- Efficient indexing and low level data structures
- Hilog (higher-order syntax) is a challenge, e.g., for indexing
- Nonmonotonicity of naf and defeasibility is a challenge
- Incremental tabling adds more dependency-awareness
 - Enables fast updating
 - E.g., for interactive rule authoring edit-test loop
- Highly sophisticated, optimized over last two decades

Hilog: Higher-Order Syntax

- Permit predicate or function to be a variable
- Permit predicate or function to be a complex functional term
- Elegant transformation defines the semantics, and is used to implement
- Intuition: $?pred(?arg1,?arg2) \rightarrow \rightarrow believe(?pred,?arg1,?arg2)$

HiLog

- 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
- 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]

Examples of Hilog (I)

Hilog permits variables over predicates and function symbols:

$p(?X, ?Y) : - ?X(a, ?Z) \text{ and } ?Y(?Z(b)).$

*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:

$p(q(a)).$

$r(?X) : - p(?X) \text{ and } ?X;$

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]

HiLog Transformation

- HiLog semantics is defined via a transformation
- A simplified version of that, which gives intuition:
 - Rewrite each atom $p(a,b) \rightarrow \text{holds_2}(p,a,b)$
 - Generic predicate constants $\text{holds_1}, \text{holds_2}, \dots$
 - Treat each term in similar manner
 - $f(a,b) \rightarrow \text{apply_2}(f,a,b)$
 - Generic function constants $\text{apply_1}, \text{apply_2}, \dots$

Representational Uses of HiLog

- There are many!
- E.g.:
 - Modalities, e.g., believe, permit
 - Adverbial modification in natural language
 - Principles of debate/argumentation
 - Restructuring in schema mapping
 - Probabilistic range and confidence about a sentence
 - Transitive closure of a relation

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

Outline of Part B. Concepts & Foundations

1. Horn LP, with Functions
2. Well-Founded Negation
3. Tabling Algorithms for LP
4. **Restraint**: semantic bounded rationality
5. Frame syntax (a.k.a. F-Logic), Object Oriented style
6. **Higher-Order** Syntax via Hilog. Reification.
7. Rule ID's
8. **Defeasibility** via Argumentation Rules. Remediating FOL's Fragility.
9. **General Formulas**, Existentials and Skolems, Omni-directional Disjunction
 - Representing Text. Importing full OWL. FOL-Soundness.
10. **Probabilistic** knowledge and reasoning
11. External Querying
12. Reactiveness
13. Misc. Lesser Features: Datatypes, Aggregation, Integrity Constraints, Inheritance, Equality, "Constraints"
14. Terminology/Ontology Mapping
15. Justification/Explanation

Concept of Logical Monotonicity

- A KR S is said to be logically monotonic when in it:
$$P1 \subseteq P2 \quad \Rightarrow \quad \text{Conc}(P1,S) \subseteq \text{Conc}(P2,S)$$
- Where P1, P2 are each a set of premises in S
- I.e., whenever one adds to the set of premises, the set of conclusions non-strictly grows (one does not retract conclusions).
- *Monotonicity is good for pure mathematics.*
 - *“Proving a theorem means never having to say you are sorry.”*

Nonmonotonicity – its Pragmatic 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 some regards
- Most commercially important rule systems/applications use nonmon
 - A basic expressive construct is ubiquitous there:
 - Default Negation a.k.a. Negation-As-Failure (NAF)
 - BUT with varying semantics – often not fully declarative cf. LP
 - Primarily due to historical hangovers and lack of familiarity with modern algorithms
 - Another expressive construct, almost as ubiquitous there, is:
 - Priorities between rules
- Such nonmonotonicity enables:
 - Modularity and locality in revision/updating/merging

Default Negation: Intro

- Default negation is the most common form of negation in commercially important rule and knowledge-based systems.
- Concept/Intuition for $\sim q$; \sim stands for **default** negation
 - q is not derivable from the available premise info
 - fail to believe q
 - ... but might also not believe q to be false
 - A.k.a. “*weak*” negation, or NAF. In Ergo: “\naf”
- Contrast with: $\neg q$; \neg stands for **strong** negation
 - q is believed to be false
 - A.k.a. “*classical*” negation. In Ergo: “\neg”

LP with Negation As Failure

- Normal LP (NLP), a.k.a. *Ordinary* LP (OLP)
 - Adds NAF to Horn LP
- Syntax: Rule generalized to permit NAF'd body literals:
 - $H :- B_1 \text{ \texttt{\textbackslash and} } \dots \text{ \texttt{\textbackslash and} } B_k$
 $\text{ \texttt{\textbackslash and} } \text{ \texttt{\textbackslash naf} } B_{k+1} \text{ \texttt{\textbackslash and} } \dots \text{ \texttt{\textbackslash and} } \text{ \texttt{\textbackslash naf} } B_m .$

where $m \geq 0$, H and B_i 's are atoms
- Semantics has subtleties for the fully general case.
 - Difficulty is interaction of NAF with “recursion”, i.e., cyclic dependencies (thru the rules) of predicates/atoms.
 - Lots of theory developed during 1984-1994
 - Well-understood theoretically since mid-1990's

Semantics for LP with Default Negation

- For fully general case, there are two major alternative semantics
- Both agree for a broad restricted case: stratified ordinary LP
- Well Founded Semantics (WFS): popular, widely used
 - Tractable for the propositional case. Often linear, worst-case quadratic.
 - Major commercial focus. E.g., XSB, OntoBroker.
 - Employs a 3rd truth value u (“undefined”), when non-stratified (“unstratified”)
 - Definition uses iterated minimality: Horn-case then close-off; repeat til done.
 - Major limitation: cannot reason by cases
- Answer Set Programs (ASP): popular as research topic
 - Enables a limited kind of disjunction in heads, conclusions
 - Good for combinatorial KR problems requiring nonmonotonicity
 - Only 2 truth values \Rightarrow sometimes ill-defined: no set of conclusions
 - Generalizes earlier “*stable model semantics*”
 - Can reason by cases! \Rightarrow Intractable for propositional case

Basic Example of LP with NAF

(NB: this example is purely fictional.)

• RB1:

- price(Amazon, Sony5401, ?day, ?cust, 49.99) :-
inUSA(?cust) \and inMonth(?day, 2004_10) \and \naf onSale(?day).
- price(Amazon, Sony5401, ?day, ?cust, 39.99) :-
inUSA(?cust) \and inMonth(?day, 2004_10) \and onSale(?day).
- inMonth(2004_10_12, 2004_10).
- inMonth(2004_10_30, 2004_10).
- inUSA(BarbaraJones).
- inUSA(SalimBirza).
- onSale(2004_10_30).

• RB1 entails: (among other conclusions)

1. Price(Amazon, Sony5401, 2004_10_12, BarbaraJones, 49.99)
2. Price(Amazon, Sony5401, 2004_10_30, SalimBirza, 39.99)

• RB2 = RB1 updated to add: onSale(2004_10_12).

• RB2 does NOT entail (1.). Instead (nonmonotonically) it entails:

3. Price(Amazon, Sony5401, 2004_10_12, BarbaraJones, 39.99)

Brief Examples of Non-Stratified Normal LP

- RB3:
 - a.
 - $c :- a \text{ \texttt{\textbackslash and} \texttt{\textbackslash naf} } b.$
 - $p :- \texttt{\textbackslash naf } p.$
- **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.
 - $c :- a \text{ \texttt{\textbackslash and} \texttt{\textbackslash naf} } b.$
 - $p :- \texttt{\textbackslash naf } q.$
 - $q :- \texttt{\textbackslash naf } p.$
- **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.

(Review:) Semantics of Horn LP

- Declarative LP with model-theoretic semantics
 - Same for forward-direction (“derivation” / “bottom-up”) and backward-direction (“query” / “top-down”) inferencing
- Model $M(P)$ = a set of concluded ground atoms
 - Where P = the set of premise rules

Semantics is defined via the least fixed point of an operator T_P . T_P outputs conclusions that are immediately derivable (through some rule in P) from an input set of intermediate conclusions I_j .

- $I_{j+1} = T_P(I_j)$; $I_0 = \emptyset$ (empty set)
 - $I_{j+1} = \{\text{all head atoms of rules whose bodies are satisfied by } I_j\}$
- $M(P) = \text{LeastFixedPoint}(T_P)$; where LFP = the I_m such that $I_{m+1} = I_m$
- Simple algorithm: DO {run each rule once} UNTIL {quiescence}

Well Founded Semantics: Least Model

P : a rulebase over language L

M : a partial Herbrand interpretation

- a set of literals (atoms and *naf* atoms) in the Herbrand Base
- all other atoms/literals have truth value u which means “undefined”

Consider ground cases.

- M is a model of P when it satisfies every rule in P
- A model M is a least model of P
if it is minimal with respect to \leq
 - $M1 \leq M2$ iff $M1^+ \subseteq M2^+$ and $M1^- \supseteq M2^-$
 - $M^+ =$ the set of *naf*-free literals in M ; $M^- =$ the set of *naf* literals in M
 - I.e., the usual notion of “minimal” for LP models
 - If P is Horn, i.e., *naf*-free, then M is said to be the minimal model.
 - In this case, M is simply the least fixed point of T_P (last slide)
 - ... and is straightforwardly computed via an iteration

Well-Founded Model: Quotient

- The well-founded semantics for LP, i.e., for NAF, is defined as a least model obtained by an iterative process (follows general outline of [*Przymusinski 94]’s WFS definition).
- Quotient of a rulebase w.r.t. an interpretation:
 - Let Q be a set of rules, and J a partial Herbrand interpretation for Q
 - The quotient $\frac{Q}{J}$ is obtained by:
 - In the body of each rule in Q , replace $\sim L$ by $J(\sim L)$

The resulting quotient LP is *almost* a set of plain Horn rules.

Because J is a partial, not total, interpretation, it’s a bit more complicated.

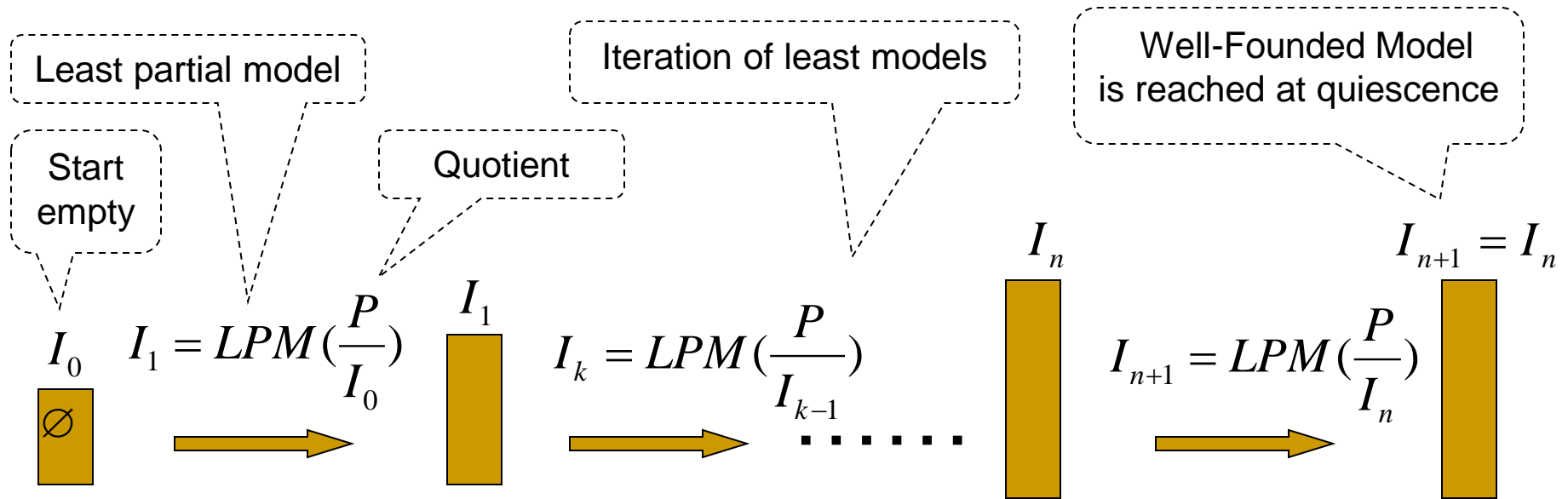
The quotient includes appearances of u . It is said to be *semi-positive*.

A semi-positive LP can be viewed as a *pair* of Horn LPs:

- a lower-bound LP (in which u is replaced by f)
- an upper-bound LP (in which u is replaced by t)

A semi-positive LP’s least partial model (LPM) is simple to compute, by taking the least fixed points of the lower-bound and upper-bound.

Well-Founded Model of LP



- The WFM of P = the iteration until quiescence of:
 - a) Take the quotient of P w.r.t. the previous iteration's interp.
 - b) Find the least partial model (LPM) of that quotient rulebase.
- ❖ Observation: The above is an “outer loop” iteration that contains an “inner loop” iteration of least fixed point (LFP), within LPM

Computing Well Founded Semantics for LP

- Always exactly one set of conclusions (entailed ground atoms)
- Tractable to compute all conclusions, for broad cases:
 - $O(n^2)$ for Propositional case of Normal LP
 - $O(n^{2v+2})$ for VB Datalog case ($v = \max \# \text{ vars per rule}$)
 - NAF only moderately increases computational complexity compared to Horn (frequently linear, at worst quadratic)
- *By contrast, for Stable Semantics:*
 - *There may be zero, or one, or a few, or very many alternative conclusion sets*
 - *Intractable even for Propositional case*
- Proof procedures are known that handle the non-stratified general case
 - backward-direction: notably, SLS-resolution
 - Fairly mature wrt performance, e.g., tabling refinements
 - forward-direction
 - Reuse insights from backward-direction. Restrict to function-free.
 - Fairly mature wrt performance. Room to improve: esp. for updating.

Some Implementations of Unstratified LP

- Well Founded:
 - XSB (research / commercial; open source)
 - Ontobroker (commercial)
 - Intellidimension (commercial)
 - SweetRules (research; open source)
 - SILK (research / commercial)
 - Flora-2 / Ergo Lite (research / commercial ; open source)
 - Ergo (commercial)
- Answer Set Programs:
 - Smodels (research)
 - DLV (research / commercial)
 - Clasp (research)
- *There are a number of others, esp. research*

Negation-As-Failure Implementations: Current Limitations in Many Systems

- Practice in Prolog and other currently commercially important (CCI) rule systems is **often “sloppy”** (incomplete / cut-corners) relative to canonical semantics for NAF
 - in cases of recursive rules, WFS algorithms required are more complex
 - ongoing diffusion of WFS theory & algorithms, beginning in Prologs
- Current implemented OLP inferencing systems often do not handle the fully general case in a semantically clean and complete fashion.
 - Many are still based on older algorithms that preceded WFS theory/algorithms
- Other CCI rule systems’ implementations of NAF are often “ad hoc”
 - Lacked understanding/attention to semantics, when developed

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 (Grosz, 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



Classical Logic is a “Bubble”

- The semantic web demands logical reasoning
- Classical logic is the basis for most of today’s semantic web reasoning
 - W3C OWL, W3C RIF-BLD
 - OMG SBVR, ISO Common Logic
- In classical logic, unlike Rulelog, any contradiction makes everything garbage
 - Total brittleness
 - The odds of consistency drop almost exponentially with the # of axioms





Above:

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

Defeasibility is Indicated When...

- **Useful generalities – and potential exceptions – coexist**
 - Specify knowledge in detail/precision appropriate 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”

Example of Confidentiality Policy

- Rules may accumulate over time or from different sources, and conflicts may arise. Priorities can resolve the conflicts. There can be exceptions to exceptions.
- **@{r1} permit(?request) :- *Condition1* .**
- **@{r2} neg permit(?request) :- *Condition2* .**
- ***Condition1(case58) \and Condition2(case58) .***
- **\overrides(r2, r1) .**
- **@{r3} permit(?request) :- *Condition3* .**

Ex.'s: Causal Chains & Change in Biology

- The change of state effected by process causality requires defeasibility in KR
 - A cause's effect is an exception to the persistence of previous state
 - When two causes interfere, one's effect is an exception to the other's effect
- Causal process reasoning is a large portion of AP Biology, often requiring multi-step causal chains and/or multiple grain sizes of description to answer a question
- E.g., Rulelog was piloted on such causal process reasoning in biology using SILK
- Hypothetical question about causal interference in an experiment:
 1. "A researcher treats cells with a chemical that prevents DNA synthesis from starting.
 2. This treatment traps the cells in which part of the cell cycle?"

Answer: G1 [which is a sub-phase of interphase]
- Counterfactual hypothetical question:
 1. " Suppose the typical number of chromosomes in a human liver cell was 12. [It's actually 46.]
 2. How many chromosomes would there be in a human sperm cell?"

Answer: 6. [I.e., half the number in the liver and most organs.]

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, Grosz RR-2010 / Semantic Web Journal 2015]

Courteous LP: the What

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

Priorities are available and useful

- Priority information is naturally available and useful. E.g.,
 - recency: higher priority for more recent updates
 - specificity: higher priority for more specific cases (e.g., exceptional cases, sub-cases, inheritance)
 - causality: higher priority for causal effects (direct or indirect) of actions than for inertial persistence of state (“frame problem”)
 - authority: higher priority for more authoritative sources (e.g., legal regulations, organizational imperatives)
 - reliability: higher priority for more reliable sources (e.g., security certificates, via-delegation, assumptions, observational data).
 - closed world: lowest priority for catch-cases
- Many practical rule systems employ priorities of some kind, often implicit. E.g.,
 - rule sequencing in Prolog and production rules
 - Courteous LP subsumes this as a special case (totally-ordered priorities)
 - Also Courteous LP enables: merging, more flexible & principled treatment

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 $\neg p$.
- Scalable & Efficient: low computational overhead beyond ordinary LPs.
 - Tractable given reasonable restrictions (VB Datalog):
 - 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 simple “argumentation theory” approach.
 - Add-on to variety of OLP rule systems, with modest effort.

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 exclusion 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 Courteous defaults 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 \neq .

Conclusions from opposition-locales previous 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 Candidates for $p1, p2$:
Team for $p1$, ..., Team for p_k



Prioritized Refutation



Set of Unrefuted Candidates for $p1, p2$:
Team for $p1$, Team for $p2$



Skepticism



Conclude Winning Side if any: at most one of $\{p1, p2\}$

Argumentation Theories approach to Defaults in LP

- **Combines Courteous + HiLog, and generalizes**
- **New approach to defaults: “argumentation theories”**
 - Meta-rules, in the LP itself, that specify when rules ought to be defeated
 - [Wan, Grosz, 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*

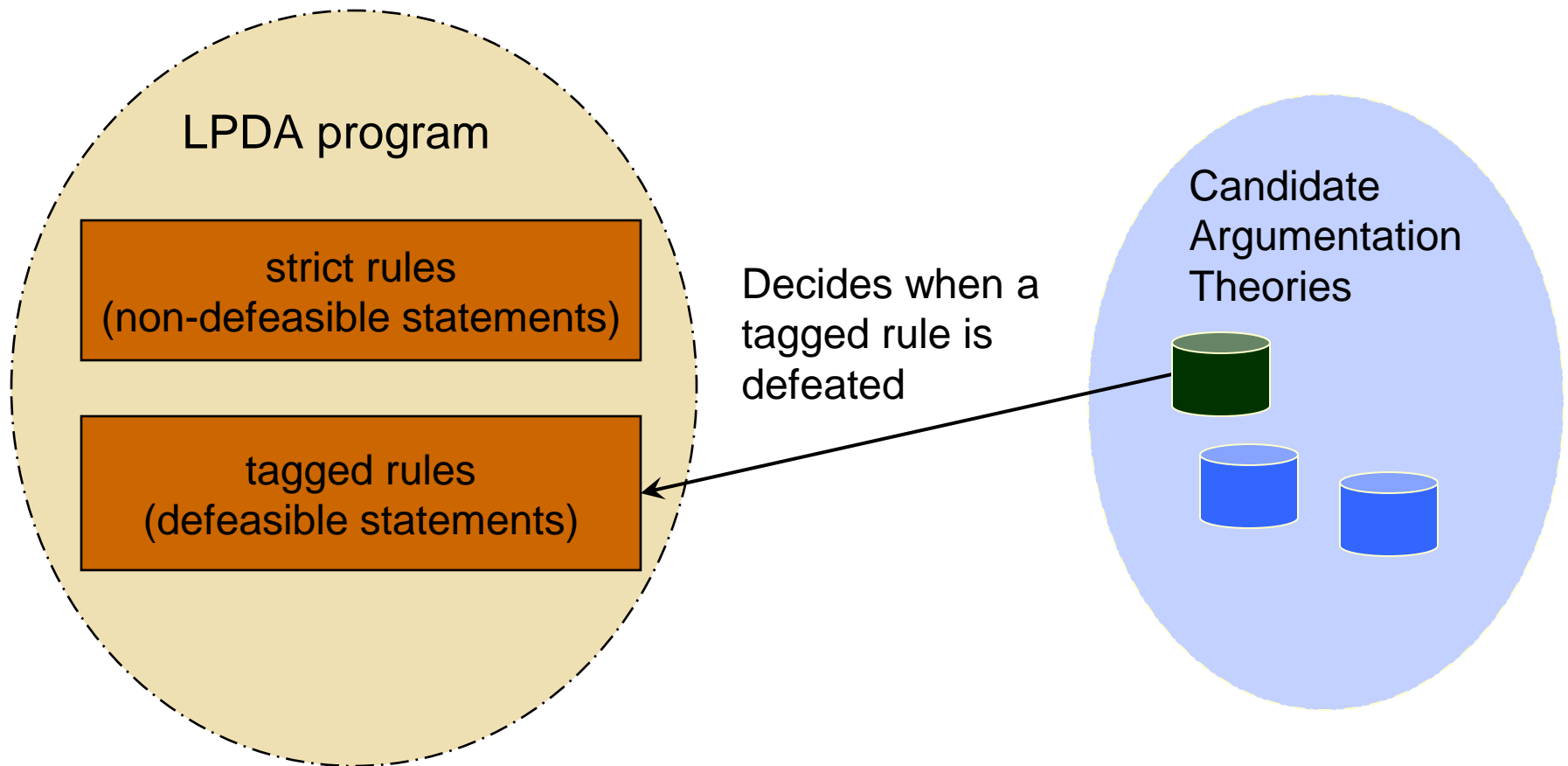
- **More Advantages**

- 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 Grosz 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

- **Logic Programs with Defaults and Argumentation theories**



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

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

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

Prioritization (user specified)

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

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

Default negation (NAF)

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

Example: E-Commerce Pricing Offer from SupplierCo to Buyer

```
@{usualPrice} price(per_unit, ?PO, $60) :-  
    purchaseOrder(?PO, supplierCo, ?AnyBuyer) and  
    quantity_ordered( ?PO, ?Q) and (?Q ≥ 5) and (?Q ≤ 1000) and  
    shipping_date(?PO, ?D) and (?D ≥ “2000-04-24”) and (?D ≤ “2000-05-12”) .  
  
@{volumeDiscount} price(per_unit, ?PO, $51) :-  
    purchaseOrder(?PO, supplierCo, ?AnyBuyer) and  
    quantity_ordered( ?PO, ?Q) and (?Q ≥ 100) and (?Q ≤ 1000) and  
    shipping_date(?PO, ?D) and (?D ≥ “2000-04-28”) and (?D ≤ “2000-05-12”) .  
  
\overrides(volumeDiscount , usualPrice) ; // volumeDiscount rule has higher priority  
// The below exclusion constraint says the value of price is unique for a given PO  
  
\opposes(price(per_unit, ?PO, ?X), price(per_unit, ?PO, ?Y)) :- ?X != ?Y .
```

...

- Notation:
“@foo” is an annotation preamble to a rule that specifies the rule’s tag. “?” prefixes a logical variable.
The “overrides” predicate specifies prioritization ordering.
An exclusion constraint specifies what constitutes a conflict.
“!=” means \neq .

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 \neq . \opposes indicates an exclusion constraint between two literals, which means “it’s a conflict if”.

E-Commerce Ex. of Causal Process Reasoning

/ E-commerce delivery logistics. */*

/ Initial fact, and prevention constraint that location is unique */*

`loc(0,PlasmaTV46,WH_LasVegasNV).`

`\opposes(loc(?s,?item,?posn1), loc(?s,?item,?posn2)) :- ?posn1 != ?posn2.`

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

`@{mov1} loc(?s+1,?item,?addr) \and \neg loc(?s+1,?item,?warehouse)`

`:- shipment(?s,?item,?warehouse,?addr) \and loc(?s,?item,?warehouse).`

/ Persistence (“frame”) axioms about location */*

`@{peloc1} loc(?s+1,?item,?posn) :- loc(?s,?item,?posn).`

`@{peloc2} \neg loc(?s+1,?item,?posn) :- \neg loc(?s,?item,?posn).`

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

`\overrides(mov1,peloc1).`

`\overrides(mov1,peloc2).`

/ An action instance occurs */*

`@{de7} shipment(1, PlasmaTV46, WH_LasVegasNV, 9_Fog_St_SeattleWA).`

As desired: \models `loc(2, PlasmaTV46, 9_Fog_St_SeattleWA)`

$\not\models$ `loc(2, PlasmaTV46, WH_LasVegasNV);`

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 on Mars.*

*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->3.7, time->7.35] // 7.35 = (2*100/3.7)^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 \neq . `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

1. Horn LP, with Functions
2. Well-Founded Negation
3. Tabling Algorithms for LP
4. **Restraint**: semantic bounded rationality
5. Frame syntax (a.k.a. F-Logic), Object Oriented style
6. **Higher-Order** Syntax via Hilog. Reification.
7. Rule ID's
8. **Defeasibility** via Argumentation Rules. Remediating FOL's Fragility.
9. **General Formulas**, Existentials and Skolems, Omni-directional Disjunction
 - Representing Text. Importing full OWL. FOL-Soundness.
10. **Probabilistic** knowledge and reasoning
11. External Querying
12. Reactiveness
13. Misc. Lesser Features: Datatypes, Aggregation, Integrity Constraints, Inheritance, Equality, "Constraints"
14. Terminology/Ontology Mapping
15. Justification/Explanation

Additional Material Follows to cover as/if time allows

- **General Formulas**, Existentials and Skolems, Omni-directional Disjunction
 - Representing Text. Importing full OWL. FOL-Soundness.
- **Probabilistic** knowledge and reasoning
- External Querying
- Reactiveness
- Misc. Lesser Features: Datatypes, Aggregation, Integrity Constraints, Inheritance, Equality, “Constraints”

Omniform Rules: Clausal case

- Rulelog introduces the concept of an omniform (“*omni*”) rule.
- Basic case is clausal. Here, the clause is treated *omni-directionally*.
 - $@\{G\} \text{ FC}$. where FC has the syntactic form of a FOL clause
 - The prioritization tag ($@\{G\}$) is optional. Outer universal quantification is implicit.
 - E.g., $@\{hi\} \text{ wet(lawn, nextMorning(?night)) } \backslash\text{or } \backslash\text{neg occur(rain, ?night)} ;$
- A clausal omni rule is transformed, i.e., directionalized, from $@\{G\} L_1 \backslash\text{or } L_2 \backslash\text{or } \dots \backslash\text{or } L_k ;$ where each L_i is an atom or the neg of an atom into a set of k directed rules, one for each choice of head literal:
 - $@\{G\} L_1 :- \backslash\text{neg } L_2 \backslash\text{and } \backslash\text{neg } L_3 \backslash\text{and } \dots \backslash\text{and } \backslash\text{neg } L_k .$
 - $@\{G\} L_2 :- \backslash\text{neg } L_1 \backslash\text{and } \backslash\text{neg } L_3 \backslash\text{and } \dots \backslash\text{and } \backslash\text{neg } L_k .$
 - ...
 - $@\{G\} L_k :- \backslash\text{neg } L_1 \backslash\text{and } \backslash\text{neg } L_2 \backslash\text{and } \dots \backslash\text{and } \backslash\text{neg } L_{k-1} .$
- This is called the set of *directional variant* rules.
- Avoids unrestricted reasoning by cases!!!
 - Cf. unit/linear resolution strategy in FOL

$\backslash\text{naf-free} !$

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**
 - $@G \quad F :- B .$
 - 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 ($\backslash\text{naf}$)
 - Use instead: $\backslash\text{neg}$ (strong negation), $\leq==$ (strong/material implication), and defeasibility (courteous)

Defeasible Existentials in Rulelog

- Existentials are needed, they arise often in natural language.
- Omniformity employs a transformation to normal LP that is based on Tight Normal Form, which differs somewhat from the Skolem Normal Form used in FOL clausification
 - Intuition: skolemize “after” directionalizing
 - Desirable to reduce the set of skolem terms that appear in bodies (post-transform)
 - See *[Grosz, RuleML-2013, invited talk summary paper]* for details
- Also extended to treat existentials are:
 - The defeasible argumentation theory (AT)
 - A new family (“ATCO”) of AT’s. Implemented in Flora.
 - NAF’s semantics and proof theory, when variables are unbound (i.e., NAF-unsafe)
 - Extension of “involuntary” restraint. Implemented in Flora.
 - Underlying technical issue: body universals arise from omni-directionality
 - *In-preparation: forthcoming papers that describe more details*

Hypermon Mapping between Rulelog and FOL

- Rulelog has a tight relationship to FOL, akin to that for Horn LP
- We can define this relationship via a *hypermonotonic* mapping T
 - Consists of a pair of mappings $(T1, T2)$, one for each interchange direction
- $T1$ maps an omni rule into a universal FOL axiom:
 - Replace :- by <== , and ignore the tag
 - E.g., $\text{@}\{G\} \text{ } F \text{ :- } B ; \quad \rightarrow\rightarrow\rightarrow\rightarrow \quad F \text{ <== } B .$
 - NB: Some non-onerous expressive restrictions apply (*current work*)
- $T1$ maps a (true) Rulelog conclusion into a FOL axiom with same formula
- $T2$ maps a universal FOL axiom into an omni rule with same formula
- Then from FOL viewpoint, entailment in Rulelog is sound and incomplete
- ... Even though Rulelog is **nonmonotonic!!!**
- Thus (restricted) Rulelog is FOL-Sound relative to interchange mapping T
- **The incompleteness is desirable when there is conflict**
 - Conflict-free case: Sound Rulelog reasoning is sound w.r.t. FOL
 - But incomplete – lacks reasoning by cases
 - Conflict-ful case: Rulelog reasoning is usefully selective unlike FOL

Interchange of Rulelog \leftrightarrow FOL

- **Omnis are a natural source/target for interchange with FOL**
- There is a (bi-)mapping T that's useful for such interchange. Its essence is:

<u>Rulelog</u>		<u>FOL</u>
@{G} E .	\leftarrow	E .
@{G} F :- B .	\rightarrow	F <== B .

(E, F, and B are formulas.
Certain restrictions apply.
The prioritization tag G is a term.)

- **W.r.t. T : Rulelog is sound and incomplete from FOL viewpoint**
- **When there is conflict, Rulelog reasoning is usefully selective unlike FOL**
- **Usage 1: Import clausal/universal FOL into Rulelog**
 - Can give prioritization to the imported rules
 - E.g., based on source authority, recency, reliability
- **Usage 2: Import Rulelog conclusions into FOL**
 - E.g., in conflict-free case. Rulelog there lacks “reasoning by cases”
- **Greatly generalizes well-known special case for definite Horn LP**
 - Handles negation (neg) and attendant conflicts
 - Can cover “nearly full”* FOL, OWL, Common Logic, SBVR

* via skolemization

Remedying FOL Semantics' Lack of Scalability

- Rulelog handles conflict robustly – get consistent conclusions
 - Whereas FOL is a “Bubble” – it’s perfectly 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 zero KB/KA conflicts from:

- Human knowledge entry/editing
 - Implicit context, cross-source ontology interpretation
 - Updating cross-source
 - Source trustworthiness
- Rulelog’s approach provides a critical advantage for KB scalability
 - semantically, as well as computationally

FOL: A Bubble

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



Left:

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

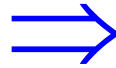
Above:

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

KR Conflict Handling – A Key to Scalability

BEFORE

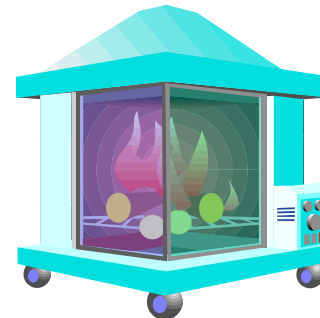
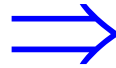
KR: Classical Logic (FOL, OWL)



AFTER

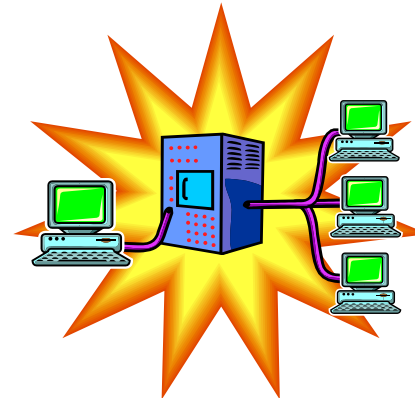
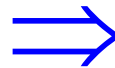
KR: LP with Defaults (Courteous-style)

Contradictory conflict is globally contagious, invalidates all results.



Contradictory conflict is contained locally, indeed tamed to aid modularity.

Knowledge integration involving conflict is labor-intensive, slow, costly.



Knowledge integration involving conflict is highly automated, faster, cheaper.

Probabilistic Knowledge & Reasoning, in Rulelog

- Overall approach: Leverage Hilog and restraint.
- Status in Ergo / Ergo Lite today: mostly “roll-your-own”.
- Probabilistic knowledge has tuple of parameters
 - $\text{Prob}(\langle \text{formula-term} \rangle, \langle \text{parameters} \rangle)$
 - Flexible in regard to what are the $\langle \text{parameters} \rangle$:
 - Point value
 - Interval
 - Mean, standard-deviation
 - Interval, confidence-level, sample-size, statistical-technique
- Evidential reasoning: weighted or prioritized combination
- Distribution semantics: semantics/foundation of Probabilistic LP

Example of Probabilistic Uncertainty in Rulelog

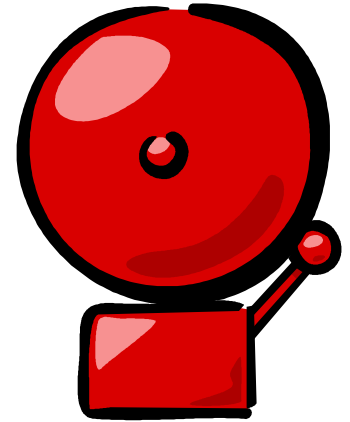
- `pr(\ (user desires a rest stop in next hour\), p_range(0.5, 0.8)) :-
 \ (user is on long trip\)\ and
 \ (last stop was more than an hour ago\
 \ (user is not in a rush\).`
- `\ (last stop was more than an hour ago\)` :-
 ...
- `\ (user is not in a rush\)` :-
 ...
- `\ (user is on a long trip\)` :-
 ...

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 Rules & LP, e.g., Rulelog

- Explicit equality (and equivalence) reasoning
 - In head of non-fact rules, therefore derived
 - Interaction with nonmonotonicity
 - Key characteristic: substitutivity of equals for equals
 - Related to Herbrand aspect of LP semantics
- Existentials, skolemization
 - RDF blank-nodes, anonymous individuals [Yang & Kifer]
 - Related to Herbrand aspect of LP semantics
- Aggregation (operate on entailed lists): count, total, min, max, etc.
 - Depends on nonmonotonicity, stratification
- Datatypes – they are basic but fairly straightforward
- “Constraints” (e.g., equation/inequality systems)
 - Commonly: via external query/assert to specialized solver
- *Also*: Reasoning within the KR about the results of side-effectful actions
 - E.g., Transaction Logic [Kifer *et al*], Golog [Reiter, Lin, *et al*]
 - These are research-world, not commercial, today

Outline of Part B. Concepts & Foundations

1. Horn LP, with Functions
2. Well-Founded Negation
3. Tabling Algorithms for LP
4. **Restraint**: semantic bounded rationality
5. Frame syntax (a.k.a. F-Logic), Object Oriented style
6. **Higher-Order** Syntax via Hilog. Reification.
7. Rule ID's
8. **Defeasibility** via Argumentation Rules. Remediating FOL's Fragility.
9. **General Formulas**, Existentials and Skolems, Omni-directional Disjunction
 - Representing Text. Importing full OWL. FOL-Soundness.
10. **Probabilistic** knowledge and reasoning
11. External Querying
12. Reactiveness
13. Misc. Lesser Features: Datatypes, Aggregation, Integrity Constraints, Inheritance, Equality, "Constraints"
14. Terminology/Ontology Mapping
15. Justification/Explanation

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

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

Other Ergo Features

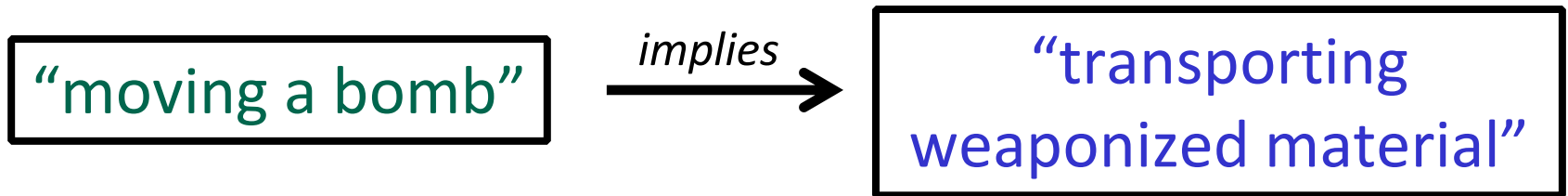
- Advanced knowledge base debugging: pause/resume, performance monitor, analysis of attempted infinite loops ("terminyzer")
- Probabilistic reasoning, e.g., evidential
- Longer-term directions, already under dev:
 - Semantically augmented NL parsing, for authoring
 - Optimize restricted cases of probabilistic
 - Complement ML via:
 - feed derived data to ML, query ML, supply features to ML

Why not Question-Answering 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?

Rulelog enriches Text Extraction

- Leverage Rulelog's high expressiveness and flexibility
- Mappings between multiple terminologies or ontologies



Internet of Things 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 AAI-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
 - 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)

- Tools for debugging of knowledge
- 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 Legal RuleML
 - 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.