PRELIMINARY PARTIAL VERSION of:

Rules on the Web

WWW-2009 Conference Tutorial

Half-day (3 hours + break), 21 April 2009,
18th International World Wide Web Conference, Madrid, Spain

*by Benjamin Grosof *(presenter), Mike Dean**, and Michael Kifer****

** Vulcan Inc. http://www.mit.edu/~bgrosof*
*** BBN Technologies http://www.daml.org/people/mdean*
**** Stony Brook University http://www.cs.sunysb.edu/~kifer*

Acknowledgements: Thanks for outline help from: Raphael Volz, Innovation Consulting GmbH http://raphaelvolz.de

WELCOME! to the WWW-2009 Tutorial

"Rules on the Web"

by Benjamin Grosof (presenter), Mike Dean, and Michael Kifer

INSTRUCTIONS! All participants, please:

- Download the final-version tutorial slideset (updated since the preliminary web-posted version) at http://www.mit.edu/~bgrosof/#WWW2009RulesTutorial
- Sign in on the participants list (hard copy sheet) with your name, organization, email; optionally also add your interests, homepage URL

Top-Level Outline of Tutorial

A. Introduction

B. Formal Foundations (longest part)

C. Languages & Standards

D. Tools

E. Wrap-up and more discussion

Outline of Part A. Introduction

1. Overview and get acquainted
   - What are Rules on the Web
2. Uses and Kinds of rules in current commercial and web systems
   - Database Queries, Production style Rules, Prolog
   - Current web rule languages, standards, tools, applications
3. Desired/envisioned uses of web rules, including services
4. High-level Requirements for web rules
5. Value of rules in IT systems lifecycle and integration
6. Use Cases
   - Info integration, ontology mapping
   - Pricing, ordering, supply chain
   - Trust/security and other policies
7. Strategic Roadmapping of Business Value and Adoption

Outline of Part B. Formal Foundations

1. Overview of Logical Knowledge Representations
   - First Order Logic (FOL), Logic Programs (LP)
2. Horn Case
3. Nonmonotonicity, Defaults, Negation, Priorities
   - Courteous LP, Argumentation Theories
4. More Connectives and Quantifiers: Lloyd-Topor, Skolemization
5. Additional Features: Datatypes, Integrity Constraints, Equality, Aggregation
6. Procedural Attachments to Actions, Queries, Built-ins, and Events
   - Production LP, Production Rules
7. HiLog, Higher-Order Syntax, Reification, Meta-Reasoning
8. F-Logic, Frame Syntax, Object Oriented Style
9. Combining / Relating LP and FOL; LP Rules with FOL Ontologies
   - Description LP, DL-Safe
   - Weakened FOL in LP, Hypermonotonic Mapping

Outline of Part C. Languages & Standards

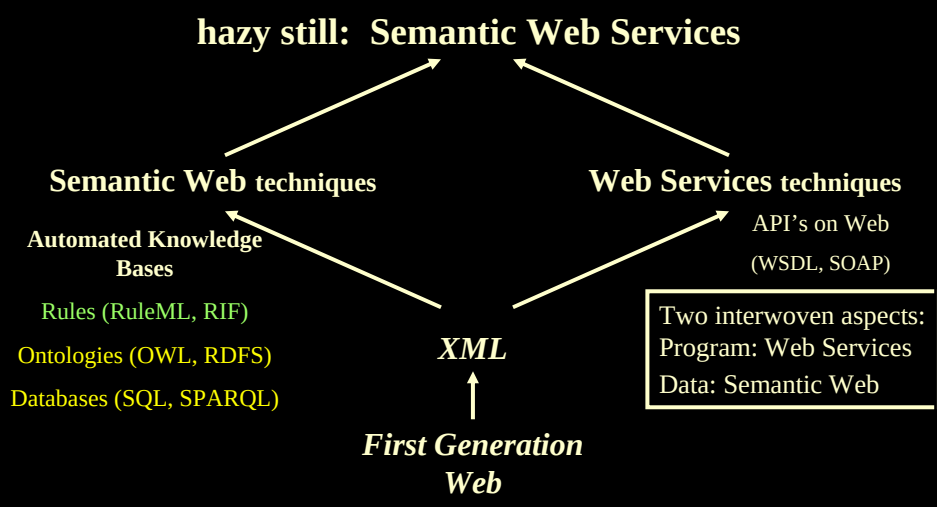
1. Database Queries are Rules
   - SQL, SPARQL, XQuery
2. Rule Markup/Modeling Language (RuleML) and related standards designs
   - Web Services modeling: SWSL, WSML
   - FOL: SWRL, Common Logic
3. W3C Rule Interchange Format (RIF)
   - Basic + Framework
4. SILK
5. Rules in, and for, W3C OWL (and RDFS) ontologies
6. OMG Production Rule Representation (PRR)
7. OMG Semantics of Business Vocabulary and Business Rules (SBVR)
8. JSR94 Rule Management API's

Outline of Part D. Tools

1. Rule systems designed to work with RDF/OWL
   - Commercial-world: Jena; Oracle; others
   - Research-world: SweetRules; cwm; others
2. Prolog and Production Rule systems
   - XSB; Jess; others
3. Advanced Expressiveness
   - Flora-2 and SILK; IBM CommonRules
4. Rules in Semantic Wikis
   - Semantic MediaWiki
5. Some Available Large Rule Bases
   - OpenCyc, Process Handbook, OpenMind

SOME PART A SLIDES FOLLOW

Emerging Next Generation Web



Flavors of Rules Commercially Most Important today in E-Business

- E.g., in OO app's, DB's, workflows.

- Relational databases, SQL: Views, queries, facts are all rules.
  - SQL99 even has recursive rules.
- Production rules (OPS5 heritage): e.g.,
  - Jess, ILOG, Blaze, Haley: rule-based Java/C++ objects.
- Event-Condition-Action rules (loose family), cf.:
  - business process automation / workflow tools.
  - active databases; publish-subscribe.
- Prolog. *"logic programs" as a full programming language.*
- *(Lesser: other knowledge-based systems.)*

Commercial Applications of Rules today in E-Business

- There are many. An established area since the 1980's.
  - Expert systems, policy management, workflow, systems management, etc.
  - Far more applications to date than of Description Logic.
- Advantages in systems specification, maintenance, integration.
- Market momentum: moderately fast growing
  - Fast in early-mid 1980's.
  - Slow late 1980's-mid-1990's.
  - Picked up again in late 1990's. (Embeddable methodologies.)
  - Accelerating in 2000's.

Vision: Uses of Rules in E-Business

- Rules as an important aspect of coming world of Internet e-business: rule-based business policies & business processes, for B2B & B2C.
  - represent seller's offerings of products & services, capabilities, bids; map offerings from multiple suppliers to common catalog.
  - represent buyer's requests, interests, bids; → matchmaking.
  - represent sales help, customer help, procurement, authorization/trust, brokering, workflow.
  - high level of conceptual abstraction; easier for non-programmers to understand, specify, dynamically modify & merge.
  - executable but can treat as data, separate from code
    - potentially ubiquitous; already wide: e.g., SQL views, queries.
- Rules in communicating applications, e.g., embedded intelligent agents.

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 from interoperability
  - Highly modular wrt updating; use prioritization
  - → Highly dynamic, scaleable rulebase authoring: distributed, integration, partnering
- Leverage Web, esp. XML
  - Interoperable syntax
  - 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

Why Standardize Rules Now?

- 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, app. dev. tools
- *⇒⇒ Identified as part of mission of the W3C Semantic Web Activity, for example*

Courteous LP Example: E-Contract Proposal from supplierCo to manufCo

- …
  <usualPrice> price(per_unit, ?PO, $60) ←
- purchaseOrder(?PO, supplierCo, ?AnyBuyer) ∧
- quantity_ordered(?PO, ?Q) ∧ (?Q ≥ 5) ∧ (?Q ≤ 1000) ∧
- shipping_date(?PO, ?D) ∧ (?D ≥ 24Apr00) ∧ (?D ≤ 12May00).
- <volumeDiscount> price(per_unit, ?PO, $51) ←
- purchaseOrder(?PO, supplierCo, ?AnyBuyer) ∧
- quantity_ordered(?PO, ?Q) ∧ (?Q ≥ 100) ∧ (?Q ≤ 1000) ∧
- shipping_date(?PO, ?D) ∧ (?D ≥ 28Apr00) ∧ (?D ≤ 12May00) .
  overrides(volumeDiscount , usualPrice) .
- ⊥ ← price(per_unit, ?PO, ?X) ∧ price(per_unit, ?PO, ?Y) GIVEN (?X ≠ ?Y).
- ...

Negotiation Example --

XML Encoding of Rules in RuleML

- <rulebase>
- <imp>
- <_rlab>usualPrice</_rlab>
- <_head>
- <cslit>
- <_opr><rel>price</rel></_opr>
- <ind>per_unit</ind>
- <var>PO</var>
- <ind>$60</ind>
- </cslit>
- </_head>
- <_body> … *(see next page)* </_body>
- </imp>
- …
- </rulebase>

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.

- Suppose more than one of the above applies to the current order? **Conflict!**

- Helpful Approach: **precedence** between the rules. Often only *partial* order of precedence is justified. E.g., C > A.

Courteous LP's: Ordering Lead Time Example

- <leadTimeRule1> orderModificationNotice(?Order,14days)
- ← preferredCustomerOf(?Buyer,?Seller) ∧
- purchaseOrder(?Order,?Buyer,?Seller) .
- <leadTimeRule2> orderModificationNotice(?Order,30days)
- ← minorPart(?Buyer,?Seller,?Order) ∧
- purchaseOrder(?Order,?Buyer,?Seller) .
- <leadTimeRule3> orderModificationNotice(?Order,2days)
- ← preferredCustomerOf(?Buyer,?Seller) ∧
- orderModificationType(?Order,reduce) ∧
- orderItemIsInBacklog(?Order) ∧
- purchaseOrder(?Order,?Buyer,?Seller) .
- overrides(leadTimeRule3 , leadTimeRule1) .
- ⊥ ← orderModificationNotice(?Order,?X) ∧
- orderModificationNotice(?Order,?Y); GIVEN ?X ≠?Y.

Policies for Compliance and Trust Mgmt.: Role for Semantic Web Rules

- Trust Policies usually well represented as rules
  - Enforcement of policies via rule inferencing engine
  - E.g., Role-based Access Control
    - This is the most frequent kind of trust policy in practical deployment today.
  - W3C P3P privacy standard, Oasis XACML XML access control emerging standard, …
- Ditto for Many Business Policies beyond trust arena, too
  - "Gray" areas about whether a policy is about trust vs. not: compliance, regulation, risk management, contracts, governance, pricing, CRM, SCM, etc.
  - Often, authorization/trust policy is really a part of overall contract or business policy, at application-level. Unlike authentication.
  - Valuable to reuse policy infrastructure

Advantages of Standardized SW Rules

- Easier Integration: with rest of business policies and applications, business partners, mergers & acquisitions
- Familiarity, training
- Easier to understand and modify by humans
- Quality and Transparency of implementation in enforcement
  - Provable guarantees of behavior of implementation
- Reduced Vendor Lock-in
- Expressive power
  - Principled handling of conflict, negation, priorities

Advantages of SW Rules, cont'd:
Loci of Business Value

- Reduced system dev./maint./training costs
- Better/faster/cheaper policy admin.
- Interoperability, flexibility and re-use benefits
- Greater visibility into enterprise policy implementation => better compliance
- Centralized ownership and improved governance by Senior Management
- Rich, expressive trust management language allows better conflict handling in policy-driven decisions

Some Answers to:
"Why does SWS Matter to Business?"

- 1. "Death. Taxes. Integration." - They're always with us.
- 2. "Business processes require communication between organizations / applications." - Data and programs cross org./app. boundaries, both intra- and inter- enterprise.
- 3. "It's the *automated knowledge* economy, stupid!" - The world is moving towards a knowledge economy. And it's moving towards deeper and broader automation of business processes. The first step is automating the use of structured knowledge.
  - Theme: *reuse* of knowledge across multiple tasks/app's/org's

Rule-based Semantic Web Services

- Rules/LP in appropriate combination with DL as KR, for RSWS
  - DL good for categorizing: a service overall, its inputs, its outputs
- Rules to describe service process models
  - rules good for representing:
    - preconditions and postconditions, their contingent relationships
    - contingent behavior/features of the service more generally,
      - e.g., exceptions/problems
  - familiarity and naturalness of rules to software/knowledge engineers
- Rules to specify deals about services: cf. e-contracting.

SOME PART B SLIDES FOLLOW

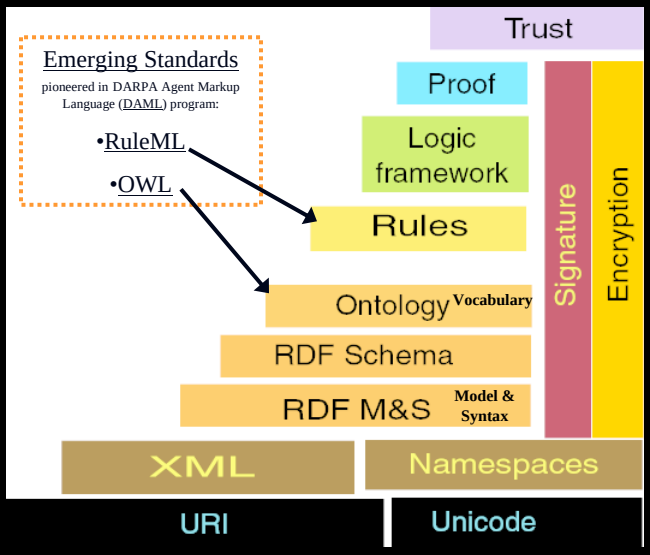
Concept of KR

- A KR S is defined as a triple (LP, LC, |=), where:
  - LP is a formal language of sets of premises (i.e., premise expressions)
  - LC is a formal language of sets of conclusions (i.e., conclusion expressions)
    - *Remark: In declarative logic programs KR, LC is a subset of LP*
  - |= is the entailment relation.
    - Conc(P,S) stands for the set of conclusions that are entailed in KR S by a set of premises P
    - We assume here that |= is a functional relation.

Knowledge Representation: What's the Game?

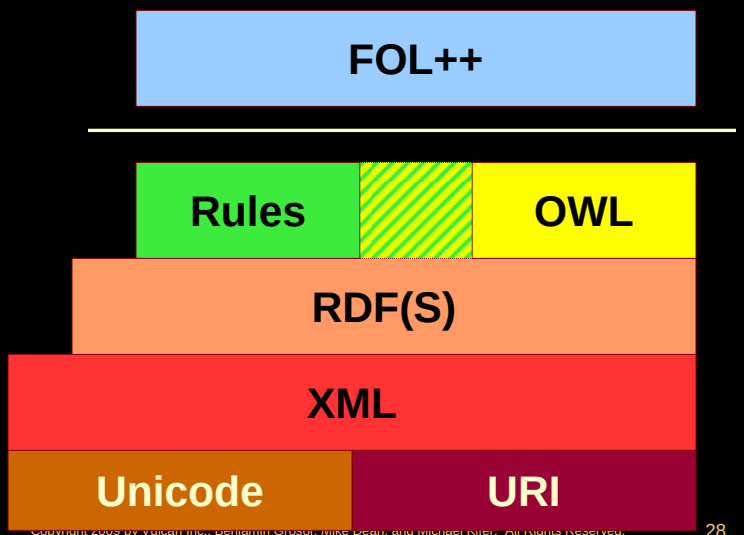
- Expressiveness: useful, natural, complex enough
- Reasoning algorithms
- Syntax: encoding data format -- here, in XML
- Semantics: principles of sanctioned inference, independent of reasoning algorithms
- Computational Tractability (esp. worst-case): scale up in a manner qualitatively similar to relational databases: computation cycles go up as a polynomial function of input size

W3C Semantic Web "Stack": Standardization Steps [2002]

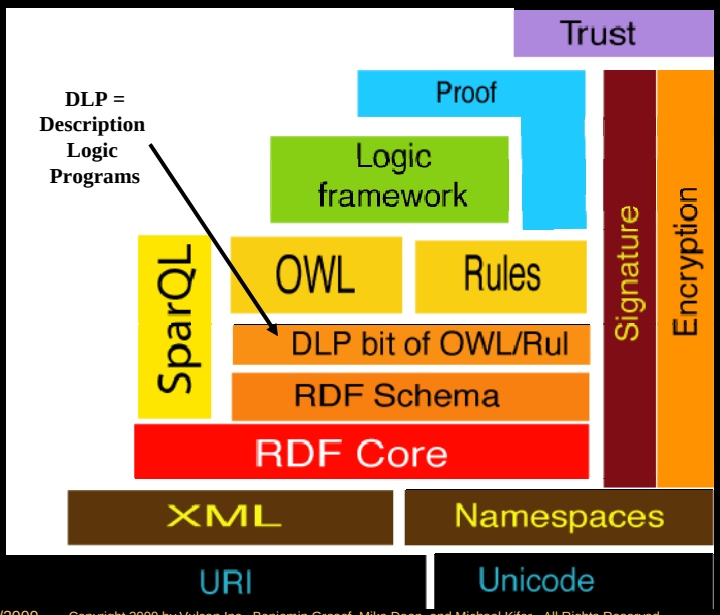


[Diagram http://www.w3.org/DesignIssues/diagrams/sw-stack-2002.png is courtesy Tim Berners-Lee]

The Web Rule Language in its Context [by RuleML & SWSI & WSMO 04-2005]



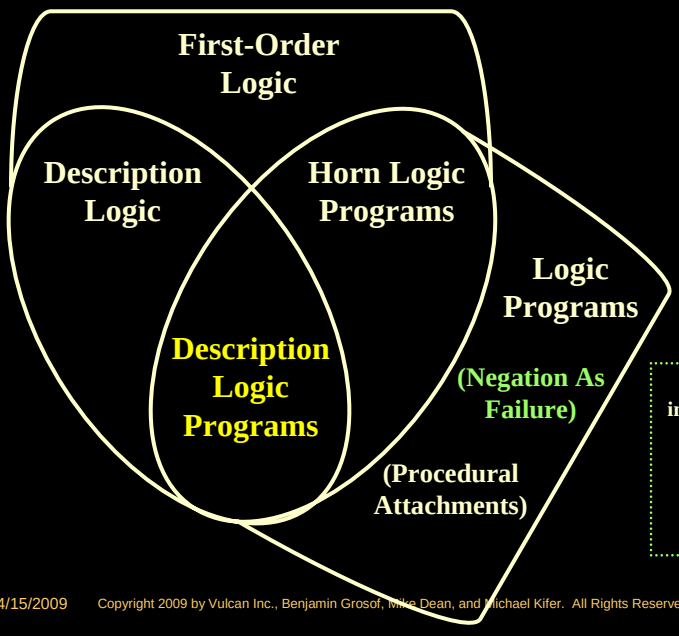
08-2005 W3C Semantic Web "Stack": Standardization Steps



Overview of Logic Knowledge Representations (KR's) and Markup Standards

- First Order Logic (FOL)
  - Standards efforts:
    - ISO Common Logic (CL) (formerly Knowledge Interchange Format)
    - FOL-RuleML (sublanguage of RuleML) & the closely related SWRL-FOL
  - Restriction: Horn FOL
  - Restriction: Description Logic (DL)
    - Standard: W3C OWL-DL & the closely related RDF-Schema (subset)
  - Extension: Higher Order Logic (HOL)
- Logic Programs (LP)
  - (Here: in the *declarative* sense.)
  - Standards efforts: RuleML & the closely related SWRL (subset)
  - Extension features:
    - Nonmonotonicity: Negation-As-Failure (NAF) ; Priorities (cf. Courteous)
    - Procedural Attachments (aproc's) for tests and actions (cf. Situated)
  - Restriction: Horn LP
  - Restriction: Description Logic Programs (DLP): overlaps with DL

Venn Diagram: Expressive Overlaps among KR's



NB: Nonmon LP, including Courteous, relies on NAF as fundamental underlying KR expressive mechanism

Description Logic: KR Expressiveness, in brief

- Restriction of First Order Logic (FOL)
  - Most essentially on the patterns of variable appearances
  - Class predicates of arity 1
  - Property predicates of arity 2
  - Complex class expressions
  - Membership axioms: foo instance-of BarClass
  - Inclusion axioms between classes (possibly complex)
    - C1 subsumed-by C2
    - I.e., x instance-of C1 $\Rightarrow$ x instance-of C2
- No logical functions
- Cannot directly represent n-ary predicates, but can indirectly

- Good for representing:
  - Many kinds of ontological schemas, including taxonomies
  - Taxonomic/category subsumptions (with strict inheritance)
  - Some kinds of categorization/classification tasks
  - Some kinds of configuration tasks

Summary of Computational Complexity of KR's

- *For task of inferencing, i.e., computing entailment of a given query.*
  - *Tractable = time is polynomial in n ; where n = |premises|*
- First Order Logic (FOL)
  - Intractable for restriction to Description Logic, or to Propositional
  - Undecidable, in general
- Logic Programs (LP) with extensions for NAF, Courteous, Test/Action Aproc's
  - Tractable, under common restrictions; complexity similar to Relational DB's
  - $O(n^2)$, for restriction to Propositional with NAF
  - Undecidable, in general

Overview of Computational Complexity of KR's

- *For task of inferencing, i.e., computing entailment of a given query.*
  - *Tractable = time is polynomial in n = |premises|*
- First Order Logic (FOL):
  - Intractable (co-NP-complete) but decidable, for restriction to Propositional
  - Intractable but decidable, for restriction to Description Logic cf. OWL-DL
  - Undecidable, in general; e.g., for restriction to SWRL
- Logic Programs (LP) with extensions for NAF, Courteous, Test/Action Aproc's:
  - Tractable, for restriction VB Datalog: (Similar to Relational DB's)
    1. Datalog* = no logical functions of arity > 0 *; and*
    2. VB = constant-bounded number of distinct variables per rule
  - … Can actually tractably compute all atomic conclusions
  - … (Under well-founded-semantics definition of NAF, tractable aproc call)
  - Tractable, therefore, for restriction to Description Logic Programs
  - $O(n^2)$, for restriction to Propositional with NAF
  - Undecidable, in general – due to recursion through logical functions
  - * Can relax to: no recursion through logical functions (ensures tractable Herbrand universe)

Horn LP as Foundation Core KR

- Horn LP provides the foundation core KR and conceptual intuitions for Rules
  - pre- Semantic Web
  - Semantic Web – including RuleML

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. $H \vee \neg B1 \vee \ldots \vee \neg Bm$. A.k.a. a definite clause / rule
     - Fact H . is special case of rule (H ground, m=0)
  2. $\neg B1 \vee \ldots \vee \neg Bm$. A.k.a. an integrity constraint

  where $m \geq 0$, H and Bi's are atoms.

  (An atom = pred(term_1,…,term_k) where pred has arity k.)
- A definite clause (1.) can be written equivalently as an implication:
  - Rule := $H \Leftarrow B1 \wedge \ldots \wedge Bm$. where $m \geq 0$, H and Bi's are atoms

    head if body ;
- An integrity constraint (2.) can likewise be written as:
  - $\bot \Leftarrow B1 \wedge \ldots \wedge Bm$. A.k.a. empty-head rule ($\bot$ is often omitted).

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

Advantage of Horn: Reduced Complexity

- Horn is less complex computationally -- and algorithmically
- Propositional FOL is co-NP-complete (recall 3-SAT is NP-complete…)
- Propositional Horn FOL is O(n)
- *(For task of inferencing, i.e., computing entailment of a given query.*
  - *n = |Premise KB|)*

Horn LP Syntax and Semantics

- Horn LP syntax is similar to implication form of Horn FOL.
  - The implication connective's semantics are a bit weaker however. We will write it as $\leftarrow$ 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
    - *(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
    - I_{j+1} = all head atoms of rules whose bodies are satisfied by I_j.
  - M(P) = LeastFixedPoint(T_P) *(LFP = I_m such that $I_{m+1} = I_m$)*

Example of Horn LP vs. Horn FOL

- Let P be:
  - DangerousTo(?x,?y) ← PredatorAnimal(?x) and Human(?y).
  - PredatorAnimal(?x) ← Lion(?x).
  - Lion(Simba).
  - Human(Joey).
- I1 = {Lion(Simba), Human(Joey)}
- I2 = {PredatorAnimal(Simba),Lion(Simba), Human(Joey)}
- I3 = {DangerousTo(Simba,Joey), PredatorAnimal(Simba),Lion(Simba), Human(Joey)}
- I4 = I3. Thus M(P) = I3.

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

Horn LP Compared to Horn FOL

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

- Horn FOL has additional non-ground-atom conclusions, notably:
  - non-unit derived clauses; tautologies
- Can thus view Horn LP as the f-weakening of Horn FOL.
  - "f-" here stands for "fact-form conclusions only"
  - A restriction on form of conclusions (not of premises).
- Horn LP -- differences from Horn FOL:
  - Conclusions Conc(P) = essentially a set of ground atoms.
    - Can extend to permit more complex-form queries/conclusions.
  - Consider Herbrand models only, *in typical formulation and usage.*
    - *P can then be replaced equivalently by {all ground instantiations of each rule in P}*
    - Can extend to permit: equalities in rules/conclusions. (Also: universal queries.)
  - Rule has non-empty head, *in typical formulation and usage.*
    - Can extend to detect violation of integrity constraints.

Summary: The "Spirit" of LP

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

- "Avoid Disjunction"
  - Avoid disjunctive expressions as premises or conclusions.
  - I.e., disjunctions of positive literals are not permitted as premises, nor as intermediate or final conclusions. Permitting such disjunctions is what creates exponential blowup of computational complexity in propositional FOL (3-SAT NP-hard).
  - No "reasoning by cases", therefore.
- "Stay Grounded"
  - Avoid non-ground conclusions.

Straightforwardly extensible, therefore, to:

- Non-monotonicity (negation-by-failure, then prioritized defaults)
- Procedural attachments (external actions, external premise facts)

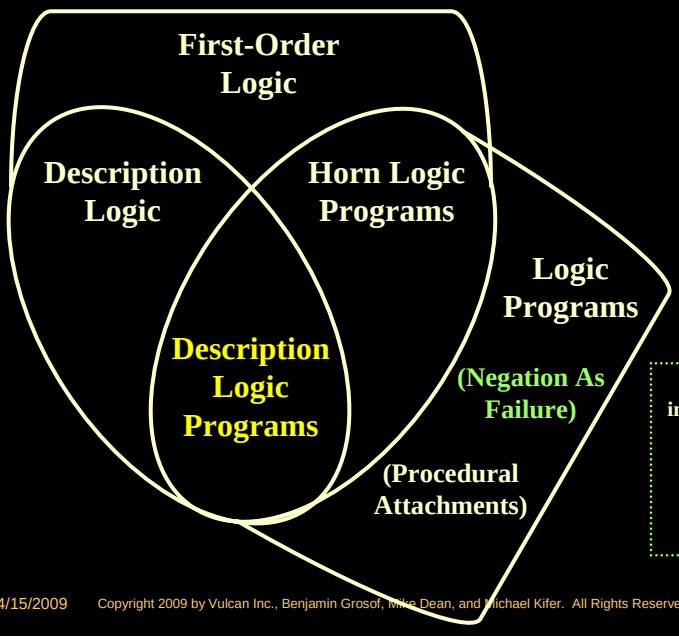
Horn LP Computational Complexity

- *For task of inferencing, i.e., computing entailment of a given query.*
  - *n = |Premise KB| i.e., |P|*

- Tractable, for restriction VB Datalog*: (Similar to Relational DB's)
  1. Datalog = no logical functions of arity > 0 *; and*
  2. VB = constant-bounded number of distinct variables per rule
  - … Can actually tractably compute all atomic conclusions
  - $O(n^{v+1})$ where v is the bound in VB
  - Tractable, therefore, for restriction to Description Logic Programs
    - In DL form of DLP, VB ≡ constant-bounded number of distinct DL quantifiers (incl. min/max cardinality) in class descriptions per inclusion axiom
  - O(n), for restriction to Propositional

  - * Can relax to: no recursion through logical functions (ensures tractable Herbrand universe)

Venn Diagram: Expressive Overlaps among KR's



NB: Nonmon LP, including Courteous, relies on NAF as fundamental underlying KR expressive mechanism

Concept of Logical Monotonicity

- A KR S is said to be <u>logically monotonic</u> when in it:

  $P1 \subseteq P2 \quad \Rightarrow \quad Conc(P1,S) \subseteq Conc(P2,S)$

- 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're sorry."*

Nonmonotonicity Motivations

- Pragmatic reasoning is, in general, nonmonotonic.
  - E.g., policies for taking actions, exception handling, legal argumentation, Bayesian/statistical/inductive, etc.
  - Monotonic is a special case – simpler wrt updating/merging, good for pure mathematics.
- Most commercially important rule systems and applications use nonmonotonicity
- A basic expressive construct is ubiquitous there:
  - Negation-As-Failure (NAF) a.k.a. Default Negation
- Another kind of expressive construct, almost as ubiquitous there, is:
  - Priorities between rules
- Such nonmonotonicity enables:
  - Modularity and locality in revision/updating/merging

Negation As Failure: Intro

- NAF is the most common form of negation in commercially important rule and knowledge-based systems.
- Concept/Intuition for ~q (~ stands for NAF)
  - 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. *default* negation, *weak* negation
- Contrast with: ¬q (¬ stands for classical negation)
  - q is believed to be false
  - A.k.a. *strong* negation

LP with Negation As Failure

- Ordinary LP (OLP), a.k.a. *Normal* LP (a.k.a. "general" LP)
  - Adds NAF to Horn LP
- Syntax: Rule generalized to permit NAF'd body literals:
  - $H \leftarrow B_1 \wedge \ldots \wedge B_k \wedge \sim B_{k+1} \wedge \ldots \wedge \sim B_m$.

  where $m \geq 0$, H and Bi'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 Negation As Failure

- For fully general case, there are multiple proposed semantics.
  - They all agree for a broad restricted case: *stratified* OLP
  - The *Well Founded* Semantics (WFS) is the most popular among commercial system implementers (e.g., XSB) and probably also among researchers
  - A previous *Stable* Semantics is also still popular among some researchers

Basic Example of LP with NAF

- RB1: (NB: this example is purely fictional.)
  - price(Amazon,Sony5401,?day,?cust,49.99)
    ← inUSA(?cust) /\ inMonth(?day,2004-10) /\ ~onSale(?day).
  - price(Amazon,Sony5401,?day,?cust,39.99)
    ← inUSA(?cust) /\ inMonth(?day,2004-10) /\ 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 OLP

- RB3:
  - a.
  - c ← a /\ ~b.
  - p ← ~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).
- Stable Semantics for RB3: there *does not exist* a set of conclusions.
  (NOT: there is a set of conclusions that is empty.)

- RB4:
  - a.
  - c ← a /\ ~b.
  - p ← ~q.
  - q ← ~p.
- WFS for RB4 entails conclusions {a,c}. p,q have truth value u.
- Stable 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.

Computing Well Founded Semantics for OLP

- Always exactly one set of conclusions (entailed ground atoms).
- Tractable to compute all conclusions:
  - $O(n^2)$ for Propositional case
  - $O(n^{2v+2})$ for VB Datalog case
  - 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
    - Not very mature yet, esp. wrt performance, for fully general case.
    - (Fairly mature wrt performance for broad restricted cases, e.g., magic sets.)

Negation As Failure Implementations: Current Limitations

- 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 Prolog's
- 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

Well Founded Semantics: Implementations of non-stratified general case

- Commercial implementations that handle non-stratified general case:
  - XSB Prolog (backward inferencing) is the currently most important and mature
  - Not many others (?none)
- There are a few other research implementations that handle non-stratified general case:
  - Smodels (exhaustive forward inferencing) is the currently most important

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

Semantical 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 CommonRules, since 1999
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

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 *pairwise* mutual exclusion constraints.
  - E.g., $\bot \leftarrow$ discount(?product,5%) $\wedge$ discount(?product,10%) .
  - E.g., $\bot \leftarrow$ loyalCustomer(?c,?s) $\wedge$ premiereCustomer(?c,?s) .
  - Permit classical-negation of atoms: ¬p means p has truth value *false*
    - implicitly, $\bot \leftarrow p \wedge \neg p$ for every atom p.
- **Priorities** between rules: partially-ordered.
  - Represent priorities via reserved predicate that compares rule labels:
    - overrides(rule1,rule2) means rule1 is higher-priority than rule2.
    - Each rule optionally has a rule label 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).
  - 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 special case (totally-ordered priorities), plus enables: merging, more flexible & principled treatment.

Courteous LP: Advantages

- Facilitate updating and merging, modularity and locality in specification.
- Expressive: classical negation, mutual exclusions, partially-ordered prioritization, reasoning to infer prioritization.
- Guarantee consistent, unique set of conclusions.
  - **Mutual exclusion is enforced**. E.g., never conclude discount is both 5% and that it is 10%, nor conclude both p and ¬p.
- Scaleable & Efficient: low computational overhead beyond ordinary LP's.
  - 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:
  - via courteous compiler: CLP → OLP.
    - A radical innovation. Add-on to variety of OLP rule systems. $O(n^3)$.

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.

- Suppose more than one of the above applies to the current order? **Conflict!**

- Helpful Approach: **precedence** between the rules. Often only *partial* order of precedence is justified. E.g., C > A.

Courteous LP's: Ordering Lead Time Example

- <leadTimeRule1> orderModificationNotice(?Order,14days)
- ← preferredCustomerOf(?Buyer,?Seller) ∧
- purchaseOrder(?Order,?Buyer,?Seller) .
- <leadTimeRule2> orderModificationNotice(?Order,30days)
- ← minorPart(?Buyer,?Seller,?Order) ∧
- purchaseOrder(?Order,?Buyer,?Seller) .
- <leadTimeRule3> orderModificationNotice(?Order,2days)
- ← preferredCustomerOf(?Buyer,?Seller) ∧
- orderModificationType(?Order,reduce) ∧
- orderItemIsInBacklog(?Order) ∧
- purchaseOrder(?Order,?Buyer,?Seller) .
- overrides(leadTimeRule3 , leadTimeRule1) .
- (⊥ ← orderModificationNotice(?Order,?X) ∧
- orderModificationNotice(?Order,?Y)) ← (?X ≠?Y) .

Courteous LP Semantics: *Prioritized argumentation in an opposition-locale.*

Conclusions from opposition-locales previous to this opposition-locale {p1,...,pk}

(Each pi is a ground classical literal. k ≥ 2.)

↓

Run Rules for p1,...,pk

↓

Set of Candidates for p1,...,pk:
Team for p1, ..., Team for pk

↓

Prioritized Refutation

↓

Set of Unrefuted Candidates for p1,...,pk:
Team for p1, ..., Team for pk

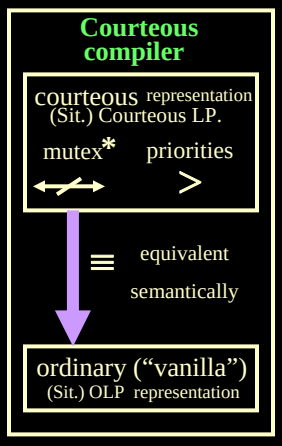
↓

Skepticism

↓

Conclude Winning Side if any: at most one of {p1,...,pk}

Courteous feature: compileable, tractable



Sit. = Situated

* classical negation too

Tractable compilation:

O(n^3), often linear

Tractable inference: e.g., worst-case

when no logical functions ("Datalog")

& bounded v = |var's per rule|

is equivalent to OLP with v → (v+2)

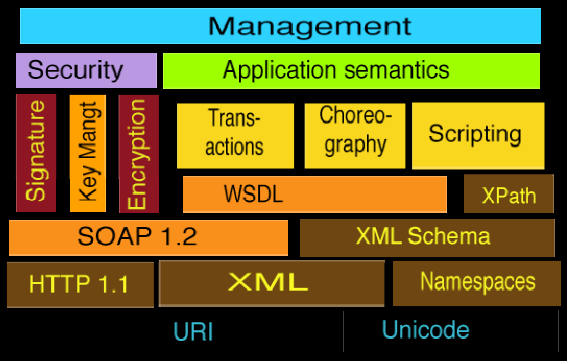
Preserves ontology.

Plus extra predicates for

- phases of prioritized argumentation (refutation, skepticism)
- classical negations

SOME PART C SLIDES FOLLOW

Web Services Stack outline

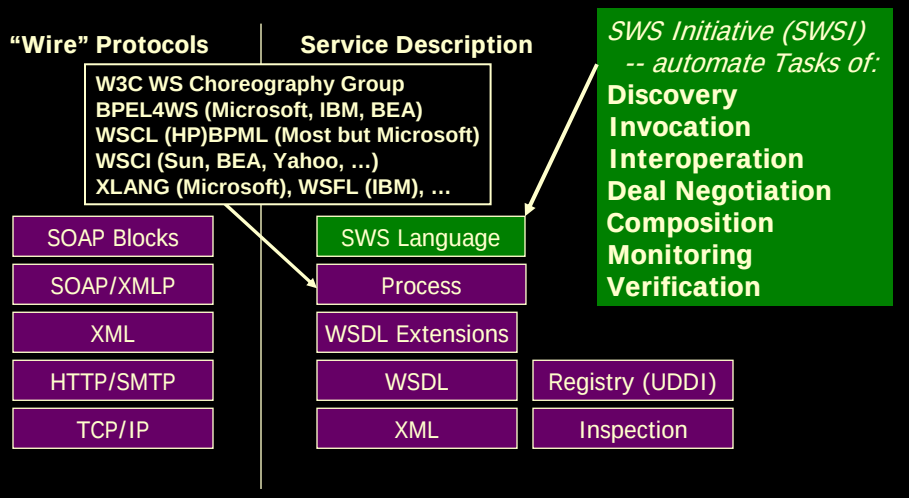


NOTES:

WSDL is a Modular Interface spec
SOAP is Messaging and Runtime
Also:
- UDDI is for Discovery
- BPEL4WS, WSCI, …
  are for transactions
- Routing, concurrency, …

Diagram courtesy Tim Berners-Lee: http://www.w3.org/2004/Talks/0309-ws-sw-tbl/slide6-0.html

SWS Language effort, on top of Current WS Standards Stack



[Slide authors: Benjamin Grosof (MIT Sloan), Sheila McIlraith (Stanford) , David Martin (SRI International), James Snell (IBM)]

Semantic Web Services Framework (SWSF)

- By Semantic Web Services Initiative (SWSI) http://www.swsi.org
  - Coordinates global research and early-phase standardization in SWS
  - http://www.swsi.org
  - Researchers from universities, companies, government
  - Industrial partners; DAML and WSMO backing
  - Collaborators: OWL-S, WSMO, RuleML, DAML
- Designed SWSF: http://www.daml.org/services/swsf/1.0/
  - Rules & FOL language (SWSL/RuleML)
  - Ontology for SWS (SWSO)
    - Drawn largely from OWL-S and PSL
  - Application Scenarios
  - *Also: requirements analysis*

SWS Tasks Form 2 Distinct Clusters, each with associated Central Kind of Service-description Knowledge and Main KR

1. Security/Trust, Monitoring, Contracts, Advertising/Discovery, Ontology-mapping Mediation
   - Central Kind of Knowledge: Policies
   - Main KR: Nonmon LP (rules + ontologies)

2. Composition, Verification, Enactment
   - Central Kind of Knowledge: Process Models
   - Main KR: FOL (axioms + ontologies)
     - + Nonmon LP for ramifications (e.g., cf. Golog)

END OF ALL SLIDES

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

