

Slideset 1 of

“Semantic Web Rules with Ontologies, and their E-Business Applications”

by Benjamin Grosfof and Mike Dean***

**MIT Sloan School of Management, <http://ebusiness.mit.edu/bgrosfof>*

***BBN Technologies, <http://www.daml.org/people/mdean>*

ISWC-2004 Conference Tutorial (half-day),

at 3rd International Semantic Web Conference, Nov. 7, 2004, Hiroshima, Japan

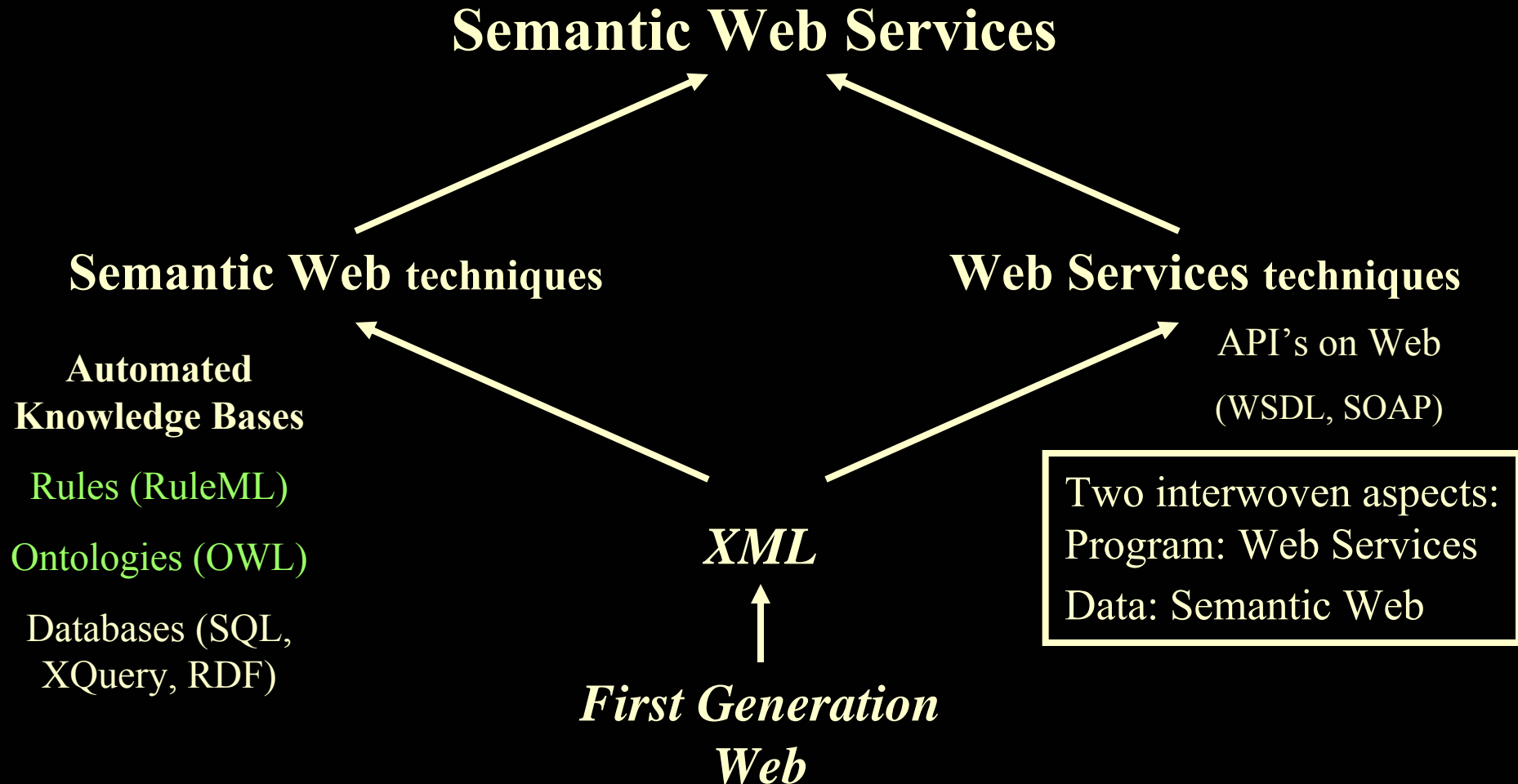
Version Date: Oct. 15, 2004

11/26/2004 Copyright 2004 by Benjamin Grosfof and Mike Dean. All Rights Reserved

Top-Level Outline of Tutorial

- *Overview and Get Acquainted*
 - A. **Core** -- KR Languages and Standards
 - B. **Tools** -- SweetRules, Jena, cwm, and More
(BREAK in middle)
 - C. **Applications** -- Policies, Services, and Semantic Integration
- *Windup*

Next Generation Web



Big Questions Addressed

- What are the critical features/aspects of the new technology for SW rules, in combination with ontologies?
- What business problems does it help solve?
- ... *from a researcher perspective...*

Outline of Part A.

A. Core -- KR Languages and Standards

1. Intro
2. Overview of Logic Knowledge Representations and Standards
3. Horn Logic / Horn LP
4. Nonmonotonic LP
5. Procedural Attachments
6. RuleML
7. Combining Rules with Ontologies
8. Datatypes
9. Review of OWL and RDF
10. SWRL
11. Additional Aspects and Approaches

Outline of Part B.

B. Tools -- SweetRules, Jena, cwm, and More

(BREAK in middle)

1. Commercially Important pre-SW Rule Systems
 - Prolog, production rules, DBMS
2. Overview of SW Rule Generations
3. 1st Gen.: Rudimentary Interoperability and XML/RDF Support
 - CommonRules, SweetRules V1, OWLJessKB
4. 2nd Gen.: Rule Systems within RDF/OWL/SW Toolkits
 - cwm, Jena-2, and others
5. 3rd Gen.: SW Rule Integration and Life Cycle
 - SweetRules V2

Outline of Part C.

C. Applications -- Policies, Services, and Semantic Integration

1. Ontology Translation and Semantic Integration
 - SWRL uses, ECOIN, financial services
2. End-to-End E-Contracting and Business Process Automation
 - supply chain, e-tailing, auctions, SweetDeal, Process Handbook
3. Business Policies including Trust
 - credit, health, RBAC, XACML, P3P, justifications
4. Semantic Web Services
 - SWSL tasks
5. Prospective Early Adopter areas, strategy, and market evolution
6. Windup and Discussion

Let's Get Acquainted

- ... We'll go around the room ...
- Please BRIEFLY (15sec max) tell the group your name, institution, particular interest/background

Quickie Bio of Presenter Benjamin Grosf

- MIT Sloan professor since 2000
- 12 years at IBM T.J. Watson Research; 2 years at startups
- PhD Comp Sci, Stanford; BA Applied Math Econ/Mgmt, Harvard
- Semantic web services is main research area:
 - Rules as core technology
 - Business Applications, Implications, Strategy:
 - e-contracting/supply-chain; finance; trust; ...
 - Overall knowledge representation, e-commerce, intelligent agents
- Co-Founder, Rule Markup Language Initiative – the leading emerging standards body in semantic web rules (<http://www.ruleml.org>)
 - Co-Lead, DAML Rules
 - Co-Lead on Rules, Joint US-EU ad hoc Agent Markup Language Committee
- Core participant in Semantic Web Services Initiative – which coordinates world-wide SWS research and early standards (<http://www.swsi.org>)
 - Area Editor for Contracts & Negotiation, Language Committee
 - Co-Chair, Industrial Partners program (SWSIP)

Quickie Bio of Presenter Mike Dean

- Principal Engineer, BBN Technologies
- B.S. in Computer Engineering from Stanford University.

- Principal Investigator, DAML Integration and Transition effort
- Chair, Joint US/EU ad hoc Markup Language Committee
 - responsible for DAML+OIL and SWRL
- Editor, OWL Web Ontology Language Reference
- Developer of several Semantic Web tools and reference data sets
- Actively using SWRL in a variety of Semantic Web applications
- Member, W3C RDF Core and Web Ontology Working Groups
- Member, RuleML Steering Committee
- Member, Architecture Committee, Semantic Web Services Initiative

Slideset 2 of

“Semantic Web Rules with Ontologies, and their E-Business Applications”

by Benjamin Grosfof and Mike Dean***

**MIT Sloan School of Management, <http://ebusiness.mit.edu/bgrosfof>*

***BBN Technologies, <http://www.daml.org/people/mdean>*

*ISWC-2004 Conference Tutorial (half-day),
at 3rd International Semantic Web Conference, Nov. 7, 2004, Hiroshima, Japan*

Version Date: Oct. 15, 2004

11/26/2004 Copyright 2004 by Benjamin Grosfof and Mike Dean. All Rights Reserved

Outline of Part A.

A. Core -- KR Languages and Standards

1. Intro
2. Overview of Logic Knowledge Representations and Standards
3. Horn Logic / Horn LP
4. Nonmonotonic LP
5. Procedural Attachments
6. RuleML
7. Combining Rules with Ontologies
8. Datatypes
9. Review of OWL and RDF
10. SWRL
11. Additional Aspects and Approaches

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.

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.

Rule-based Semantic Web Services

- Rules often good to executably specify service process models
 - e.g., business process automation using procedural attachments to perform side-effectful/state-changing actions ("effectors" triggered by drawing of conclusions)
 - e.g., rules obtain info via procedural attachments ("sensors" test rule conditions)
 - e.g., rules for knowledge translation or inferencing
 - e.g., info services exposing relational DBs
- Infrastructural: rule system functionality as services:
 - e.g., inferencing, translation

Application Scenarios for Rule-based Semantic Web Services

- SweetDeal [Grosf & Poon 2002] configurable reusable e-contracts:
 - LP rules about agent contracts with exception handling
 - ... on top of DL ontologies about business processes;
 - *a scenario motivating DLP*
- Other:
 - Trust management / authorization (Delegation Logic) [Li, Grosf, & Feigenbaum 2000]
 - Financial knowledge integration (ECOIN) [Firat, Madnick, & Grosf 2002]
 - Rule-based translation among contexts / ontologies
 - Equational ontologies
 - Business policies, more generally, e.g., privacy (P3P)

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
- $\Rightarrow\Rightarrow$ *Identified as part of mission of the W3C Semantic Web Activity*

Outline of Part A.

A. Core -- KR Languages and Standards

1. Intro
2. Overview of Logic Knowledge Representations and Standards
3. Horn Logic / Horn LP
4. Nonmonotonic LP
5. Procedural Attachments
6. RuleML
7. Combining Rules with Ontologies
8. Datatypes
9. Review of OWL and RDF
10. SWRL
11. Additional Aspects and Approaches

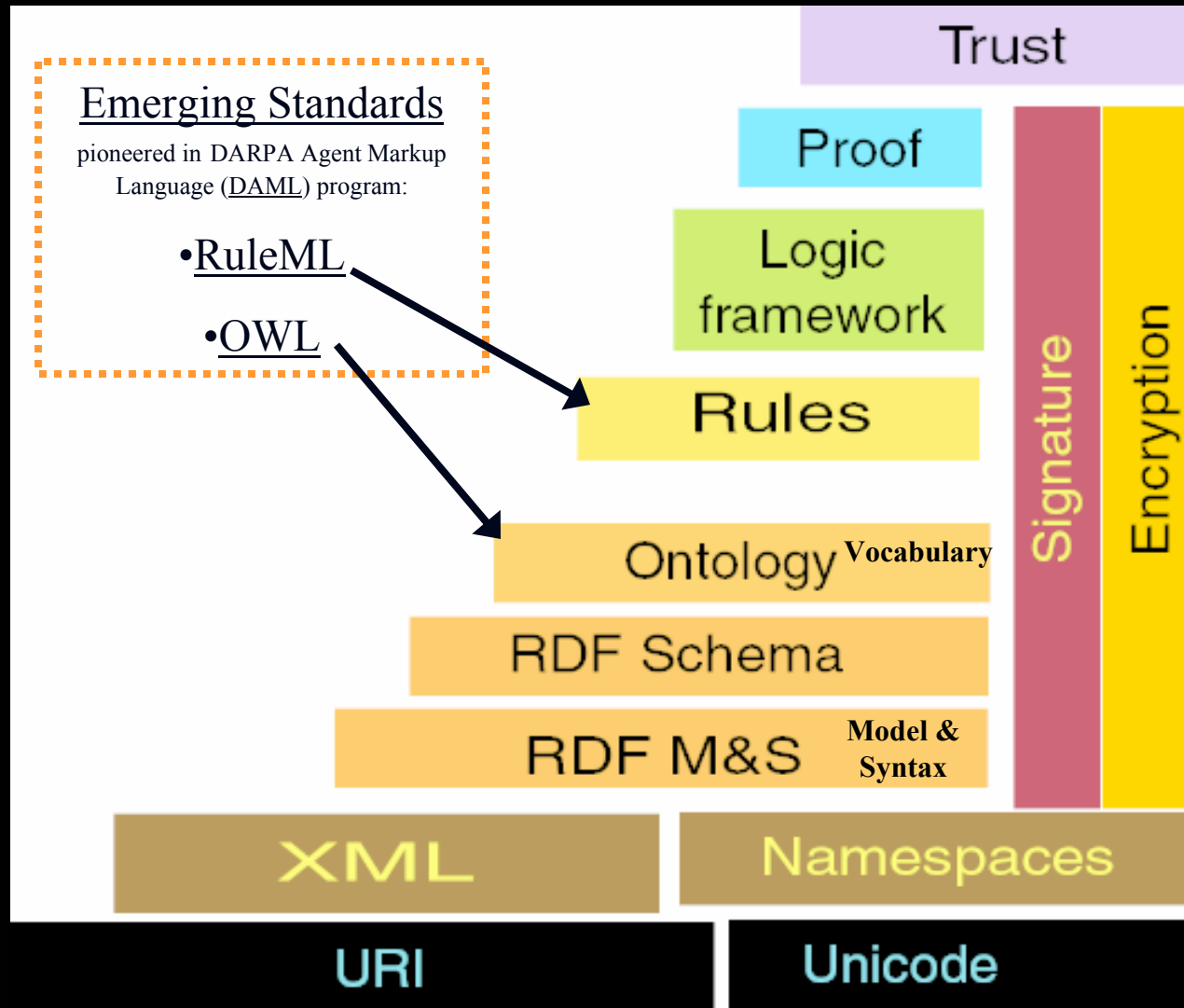
Concept of KR

- A KR S is defined as a triple (LP, LC, \models) , 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)
 - \models is the entailment relation.
- $\text{Conc}(P, S)$ stands for the set of conclusions that are entailed in KR S by a set of premises P
- We assume here that \models 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

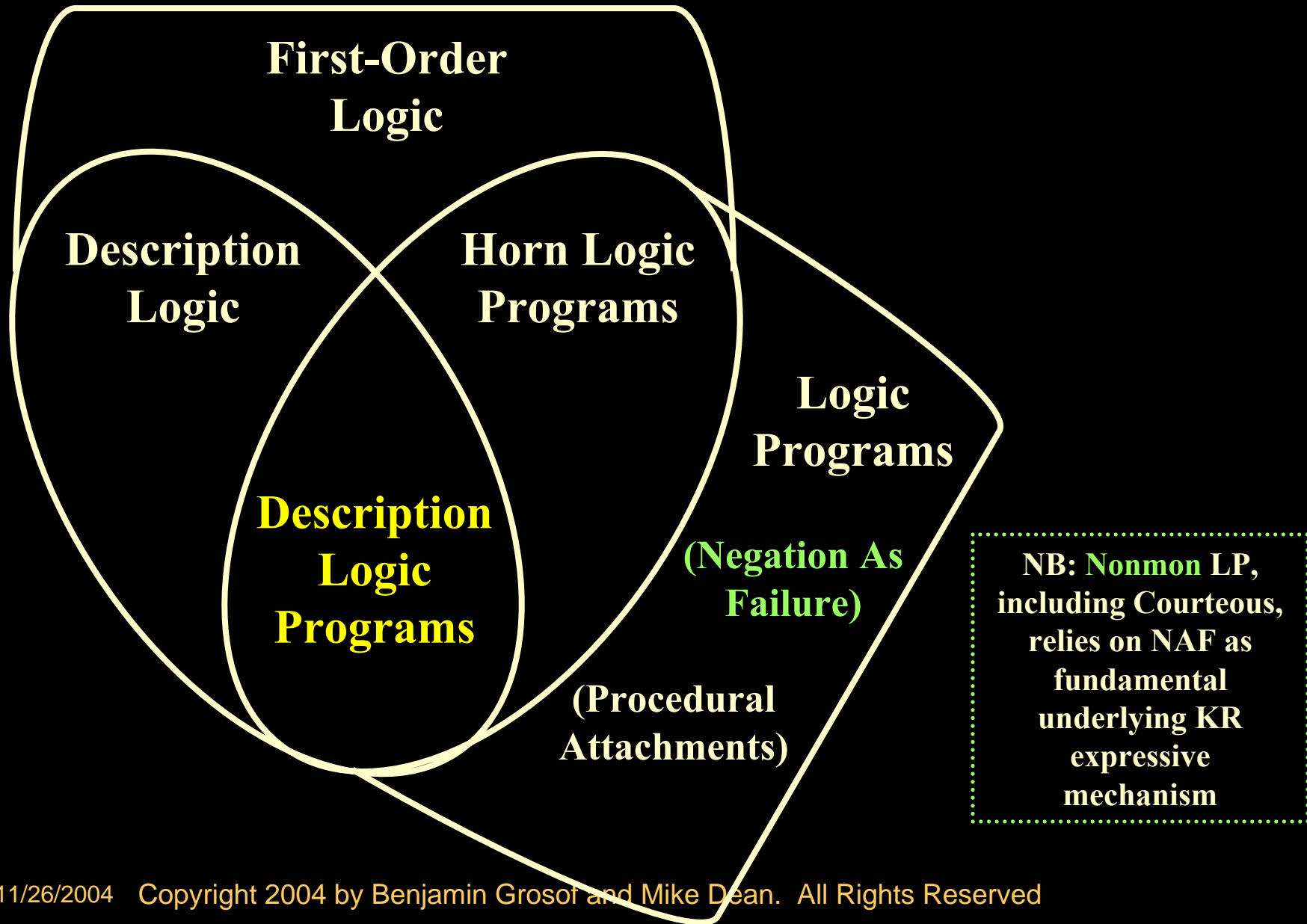


[Diagram <http://www.w3.org/DesignIssues/diagrams/sw-stack-2002.png> is courtesy Tim Berners-Lee]

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

- **First Order Logic (FOL)**
 - Standards efforts:
 - ISO Simplified Common Logic (SCL) (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



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 $n = |\text{premises}|$
- 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
 - Intractable, 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 $n = |\text{premises}|$
- 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
 - Intractable but decidable, in general
 - * Can relax to: no recursion through logical functions (ensures tractable Herbrand universe)

Outline of Part A.

A. Core -- KR Languages and Standards

1. Intro
2. Overview of Logic Knowledge Representations and Standards
3. Horn Logic / Horn LP
4. Nonmonotonic LP
5. Procedural Attachments
6. RuleML
7. Combining Rules with Ontologies
8. Datatypes
9. Review of OWL and RDF
10. SWRL
11. Additional Aspects and Approaches

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. $H \vee \neg B_1 \vee \dots \vee \neg B_m$. 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 .)

- 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.).

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.*
 - $n = |P|$)

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
 - ($P = \text{the set of premise rules}$)
- Semantics is defined via the least fixed point of an operator T_P . T_P outputs conclusions that are immediately derivable (through some rule in P) from an input set of intermediate conclusions I_j .
 - $I_{j+1} = T_P(I_j)$; $I_0 = \text{emptyset}$
 - I_{j+1} = all head atoms of rules whose bodies are satisfied by I_j .
 - $M(P) = \text{LeastFixedPoint}(T_P)$ ($LFP = I_m$ such that $I_{m+1} = I_m$)

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 {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.

Example of Horn LP vs. Horn FOL

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

- Let P' be the Horn FOL rulebase version of P above, where \Leftarrow replaces \leftarrow .
- 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 Computational Complexity

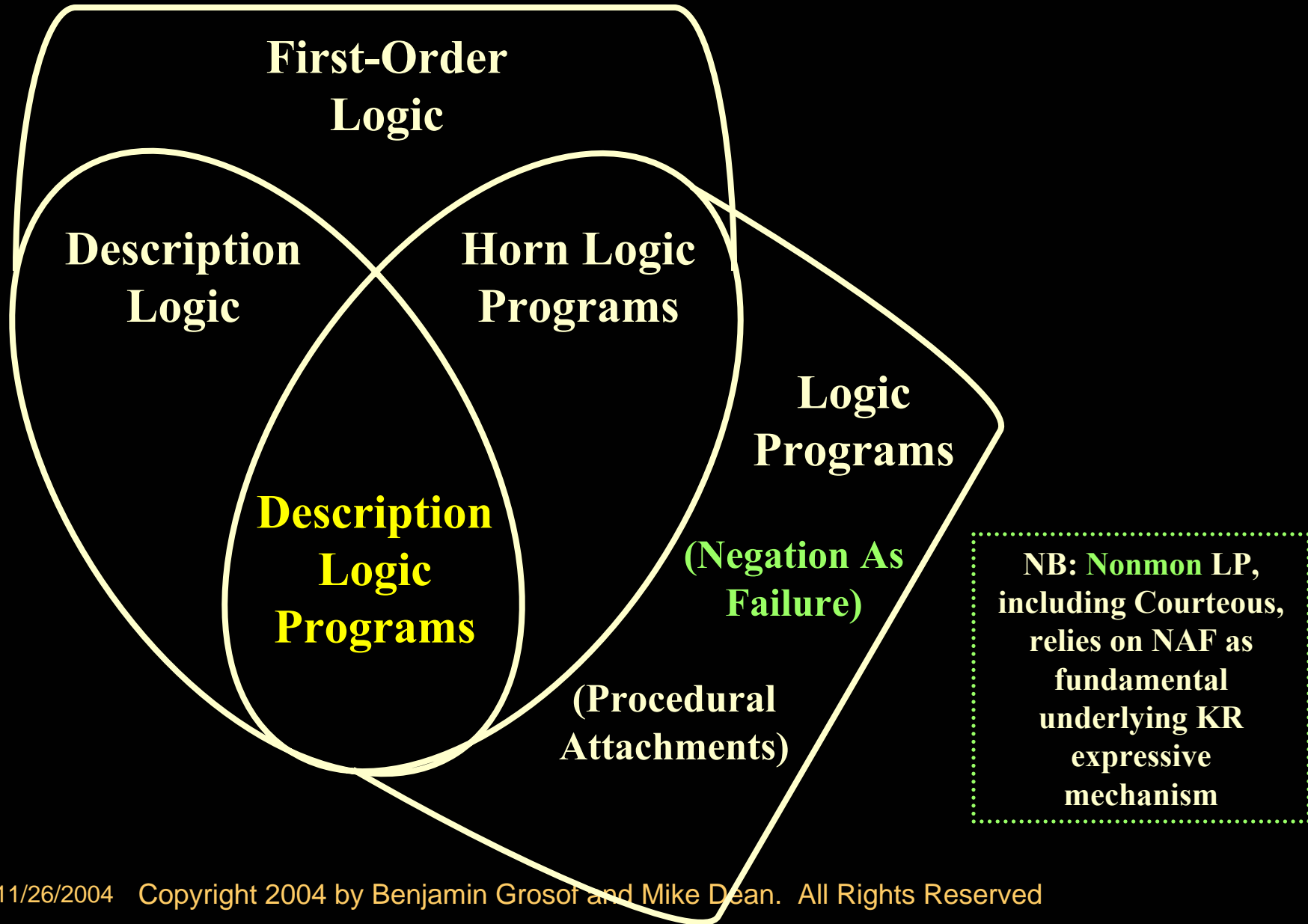
- For task of inferencing, i.e., computing entailment of a given query.
 - $n = |\text{Premise KB}|$ i.e., $|P|$
- **Tractable**, for restriction **VB Datalog***: (Similar to Relational DB's)
 1. **Datalog** = no logical functions of arity > 0 ; *and*
 2. **VB** = constant-bounded number of distinct 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 \equiv 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)

Outline of Part A.

A. Core -- KR Languages and Standards

1. Intro
2. Overview of Logic Knowledge Representations and Standards
3. Horn Logic / Horn LP
4. Nonmonotonic LP
5. Procedural Attachments
6. RuleML
7. Combining Rules with Ontologies
8. Datatypes
9. Review of OWL and RDF
10. SWRL
11. Additional Aspects and Approaches

Venn Diagram: Expressive Overlaps among KR's



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)
- 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 $\sim q$ (\sim 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: $\neg q$ (\neg 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:
 - $H \leftarrow B_1 \wedge \dots \wedge B_k \wedge \sim B_{k+1} \wedge \dots \wedge \sim B_m .$
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 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.
 - $c \leftarrow a \wedge \sim b$.
 - $p \leftarrow \sim p$.
- **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.
 - $c \leftarrow a \wedge \sim b$.
 - $p \leftarrow \sim q$.
 - $q \leftarrow \sim p$.
- **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., $\perp \leftarrow \text{discount}(\text{?product}, 5\%) \wedge \text{discount}(\text{?product}, 10\%)$.
 - E.g., $\perp \leftarrow \text{loyalCustomer}(\text{?c}, \text{?s}) \wedge \text{premiereCustomer}(\text{?c}, \text{?s})$.
 - Permit classical-negation of atoms: $\neg p$ means p has truth value *false*
 - implicitly, $\perp \leftarrow p \wedge \neg p$ for every atom p .
- Priorities between rules: partially-ordered.
 - Represent priorities via reserved predicate that compares rule labels:
 - $\text{overrides}(\text{rule1}, \text{rule2})$ 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 $\neg p$.
- 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 \rightarrow 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

- $\langle \text{leadTimeRule1} \rangle$ $\text{orderModificationNotice}(\text{?Order}, 14\text{days})$
- $\leftarrow \text{preferredCustomerOf}(\text{?Buyer}, \text{?Seller}) \wedge$
- $\text{purchaseOrder}(\text{?Order}, \text{?Buyer}, \text{?Seller}) .$
- $\langle \text{leadTimeRule2} \rangle$ $\text{orderModificationNotice}(\text{?Order}, 30\text{days})$
- $\leftarrow \text{minorPart}(\text{?Buyer}, \text{?Seller}, \text{?Order}) \wedge$
- $\text{purchaseOrder}(\text{?Order}, \text{?Buyer}, \text{?Seller}) .$
- $\langle \text{leadTimeRule3} \rangle$ $\text{orderModificationNotice}(\text{?Order}, 2\text{days})$
- $\leftarrow \text{preferredCustomerOf}(\text{?Buyer}, \text{?Seller}) \wedge$
- $\text{orderModificationType}(\text{?Order}, \text{reduce}) \wedge$
- $\text{orderItemIsInBacklog}(\text{?Order}) \wedge$
- $\text{purchaseOrder}(\text{?Order}, \text{?Buyer}, \text{?Seller}) .$
- $\text{overrides}(\text{leadTimeRule3} , \text{leadTimeRule1}) .$
- $(\perp \leftarrow \text{orderModificationNotice}(\text{?Order}, \text{?X}) \wedge$
- $\text{orderModificationNotice}(\text{?Order}, \text{?Y})) \leftarrow (\text{?X} \neq \text{?Y}) .$

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

Conclusions from opposition-locales previous to this opposition-locale $\{p_1, \dots, p_k\}$

(Each p_i is a ground classical literal. $k \geq 2$.)

Run Rules for p_1, \dots, p_k

Set of Candidates for p_1, \dots, p_k :
Team for p_1 , ..., Team for p_k

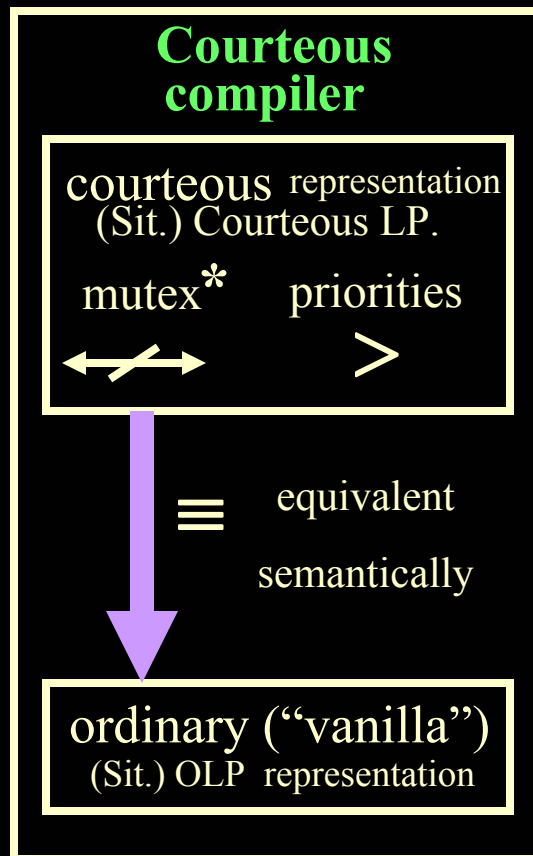
Prioritized Refutation

Set of Unrefuted Candidates for p_1, \dots, p_k :
Team for p_1 , ..., Team for p_k

Skepticism

Conclude Winning Side if any: at most one of $\{p_1, \dots, p_k\}$

Courteous feature: compileable, tractable



Tractable compilation:
 $O(n^3)$, often linear

Tractable inference: e.g., worst-case when no logical functions ("Datalog") & bounded $v = |\text{var's per rule}|$ is equivalent to OLP with $v \rightarrow (v+2)$

Preserves ontology.

Plus extra predicates for

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

Sit. = Situated

* classical negation too

Outline of Part A.

A. Core -- KR Languages and Standards

1. Intro
2. Overview of Logic Knowledge Representations and Standards
3. Horn Logic / Horn LP
4. Nonmonotonic LP
5. Procedural Attachments
6. RuleML
7. Combining Rules with Ontologies
8. Datatypes
9. Review of OWL and RDF
10. SWRL
11. Additional Aspects and Approaches

Heavy Reliance on Procedural Attachments in Currently Commercially Important Rule Families

- E.g., in OO app's, DB's, workflows.
- Relational databases, SQL: **Built-in sensors**, e.g., for arithmetic, comparisons, aggregations. **Sometimes effectors**: active rules / triggers.
- Production rules (OPS5 heritage): e.g., Jess
 - **Pluggable** (and built-in) sensors and effectors.
- Event-Condition-Action rules:
 - **Pluggable** (and built-in) sensors and effectors.
- Prolog: e.g., XSB.
 - **Built-in sensors and effectors**. More recent systems: more pluggability of the built-in attached procedures.

Additional Motivations in Semantic Web for Procedural Attachments

- Query over the web
- Represent services
- Shared ontology of basic built-in purely-informational operations on XML Schema datatypes,
 - E.g., addition, concatenation
 - E.g., in RuleML & SWRL, N3.
- Hook rules to web services, generally

Providing Declarative Semantics for Procedural Attachments

- Procedural attachments historically viewed in KR theory as ... well... *procedural* ;-)) ... rather than declarative.
 - Not much theoretical attention altogether.
- Needed for Semantic Web: a *declarative* KR approach to them
- Situated LP is currently probably the most important approach
 - In RuleML, since 2001
 - Provides disciplined expressive abstraction for two broad, often-used categories of procedural attachments:
 - Purely-informational Tests
 - Side-effectful Actions
 - Makes restrictions / assumptions become explicit
 - Declarative semantic guarantees, interoperability
 - Embodies primarily analytical insight, initially
 - Provides also: expressive generalizations, algorithms/techniques

Situated LP: Overview II

- Point of departure: LP's are pure-belief representation, but most practical rule systems want to invoke external procedures.
- Situated LP's feature a semantically-**clean** kind of **procedural attachments**. I.e., they hook beliefs to drive procedural API's outside the rule engine.
- Procedural attachments for **sensing** (queries) when testing an antecedent condition or for **effecting** (actions) upon concluding a consequent condition. Attached procedure is invoked when testing or concluding in inferencing.
- **Sensor or effector statement** specifies an association from a predicate to a procedural call pattern, e.g., a method. Such statements are specified as **part of the extended KR**.

Situated LP: Overview III

- `phoneNumberOfPredicate ::s:: BoeingBluePagesClass.getPhoneMethod .`
example sensor statement
- `shouldSendPagePredicate ::e:: ATTPagerClass.goPageMethod .`
example effector statement
- Sensor procedure may require some arguments to be ground, i.e., bound; in general it has a specified binding-signature which specifies bound vs. free for each argument.
- Enable dynamic or remote invocation/loading of the attached procedures (e.g., exploit Java goodness).
- Overall: cleanly separate out the procedural semantics as a declarative extension of the pure-belief declarative semantics. Easily separate chaining from action. (Declarative = Independent of inferencing control.)

Situated LP: Overview IV

- SLP is KR for Hooking Rules to Services
 - With ontologies
 - Esp. Web services
 - Declaratively
- Rules use services
 - E.g., to query, message, act with side-effects
- Rules constitute services executably
 - E.g., workflow-y business processes

Semantics of Situated LP I

- Definitional: complete inferencing+action occurs during an “episode” – intuitively, run all the rules (including invoking effectors and sensors as go), then done.
- Effectors can be viewed as all operating/invoked after complete inferencing has been performed.
 - **Independent of inferencing control.**
 - Separates pure-belief conclusion from action.

Semantics of Situated LP II

- Sensors can be viewed as accessing a virtual knowledge base (of facts). Their results simply augment the local set of facts. These can be saved (i.e., cached) during the episode.
 - **Independent of inferencing control.**
- The sensor attached procedure could be a remote powerful DB or KB system, a web service, or simply some humble procedure.
- Likewise, an effector attached procedure could be a remote web service, or some humble procedure. An interesting case for SW is when it performs updating of a DB or KB, e.g., “delivers an event”.
- Terminology:
 - *Situated Inferencing* = inferencing with sensing and effecting, i.e., inferencing+action

Situated Courteous LP (SCLP)

- The Situated and Courteous extensions combine essentially orthogonally.
 - Sensors may be the subject of prioritized conflict handling, so it is useful to give (optional) labels to sensor statements.

Outline of Part A.

A. Core -- KR Languages and Standards

1. Intro
2. Overview of Logic Knowledge Representations and Standards
3. Horn Logic / Horn LP
4. Nonmonotonic LP
5. Procedural Attachments
6. RuleML
7. Combining Rules with Ontologies
8. Datatypes
9. Review of OWL and RDF
10. SWRL
11. Additional Aspects and Approaches

Overview of RuleML Today I

- RuleML Initiative (2000--)
 - Dozens of institutions (~35), researchers; esp. in US+Canada, EU
 - Mission priorities:
 1. Enable semantic exchange of rules/facts between most commercially important rule systems
 2. Synergize with RDF, OWL (& other relevant web standards as arrive)
 3. Enable rule-based semantic web services, e.g., policies
 - Standards specification: current version V0.8+
 - 1st version 2001; basic now fairly stable
 - A number of tools (~12 engines, translators, editors), demo applications

Overview of RuleML Today II

- Annual RuleML Workshop at ISWC since 2002 on RuleML & SW Rules
- Has now a “home” institutionally in DAML and Joint Committee
- Discussions well underway to launch W3C, Oasis efforts
- Collaborating with Semantic Web Services Initiative (SWSL)
- Close relationship with REWERSE (EU Network of Excellence on SW Rules)
- Collaborating with WSMO (early phase)
- Initial Core: Horn Logic Programs KR
 - ... Webized (in markup)... and with expressive extensions

URI's, XML, RDF, ... non-mon, actions, ...

Overview of RuleML Today III

- Fully Declarative KR (not simply Prolog!)
 - Well-established logic with model theory
 - Available algorithms, implementations
 - Close connection to relational DB's
 - core SQL is Datalog Horn LP
- Abstract graph syntax
 - 1st encoded in XML...
 - ... then RDF
- Expressive Extensions incrementally, esp. already:
 - Non-monotonicity: Negation as failure; Courteous priorities
 - Procedural Attachments: Situated actions/effecting, tests/sensing
 - *In-progress*:
 - Hilog, frame syntax, reification cf. F-Logic Programs, SWSL
 - Events cf. Event-Condition-Action

Technical Approach of RuleML: I

1. Family of sub-languages, each a Webized KR expressive class.

With various expressive and syntactic extension features / restrictions.

Two major sub-families:

a. Declarative LP: mainly Situated Courteous LP and restrictions

b. FOL (in collaboration with Joint Committee)

2. Expressively: Start with: Datalog Horn LP *as kernel*

Rationale: captures well a simple shared core among CCI rule sys.

- Tractable! (if bounded # of logical variables per rule)

3. Syntax: Permit URI's as predicates, functions, etc. (names)

- namespaces too

4. Expressively: Permit rules to be labeled

Need names on the Web: best within the KR, e.g., prioritizing, meta-rules

Technical Approach of RuleML: II

- 5. Expressively: Add: extensions to LP KR cf. established research
 - negation-as-failure (well founded semantics) -- in body (*stays tractable!*)
 - classical negation: limited to head or body atom – syntactic sugar
 - prioritized conflict handling cf. Courteous LP (*stays tractable!*)
 - procedural attachments: actions, queries ; cf. Situated LP (*stays declarative!*)
 - logical functions (arity > 0)
 - datatypes cf. XML-Schema, RDF, OWL
 - 1st-order logic type expressiveness cf. Lloyd-Topor – syntactic sugar
 - $\forall, \exists, \nabla, \leftarrow$ in body; $\wedge, \forall, \leftarrow$ in head (*stays tractable!*) (*part still in progress*)
 - Equality (explicit): in body; in facts, in rule head (*part still in progress*)
 - Hilog (*in progress*)
 - frame syntax cf. F-Logic Programs – syntactic sugar (*in progress*)
 - reification (*in progress*)
 - integrity constraints (*in progress*)

Technical Approach of RuleML: III

- 6. Expressively: Add: restrictions cf. established R&D
 - E.g., for particular flavors of rule systems
 - E.g., Prolog, production rules, SQL, ...
 - Also “pass-thru” some info without declarative semantics (pragmatic meta-data)
- 7. Syntax for XML:
 - Family of XML-Schemas:
 - a generalization-specialization hierarchy (lattice)
 - define Schemas modularly, using XML entities (~macros)
 - optional header to describe expressive-class using “meta-”ontology
- 8. Syntax: abstract unordered graph syntax (data model)
 - Support RDF as well as XML (avoid reliance on sequence in XML)
 - “Slots” name each child, e.g., in collection of arguments of an atom
 - Orderedness as optional special case, e.g., for tuple of arguments of an atom
- 9. Syntax: module inclusion: merge rulesets ; import/export
 - URI’s name/label knowledge subsets

Technical Approach of RuleML: IV

10. Expressively and syntactically:

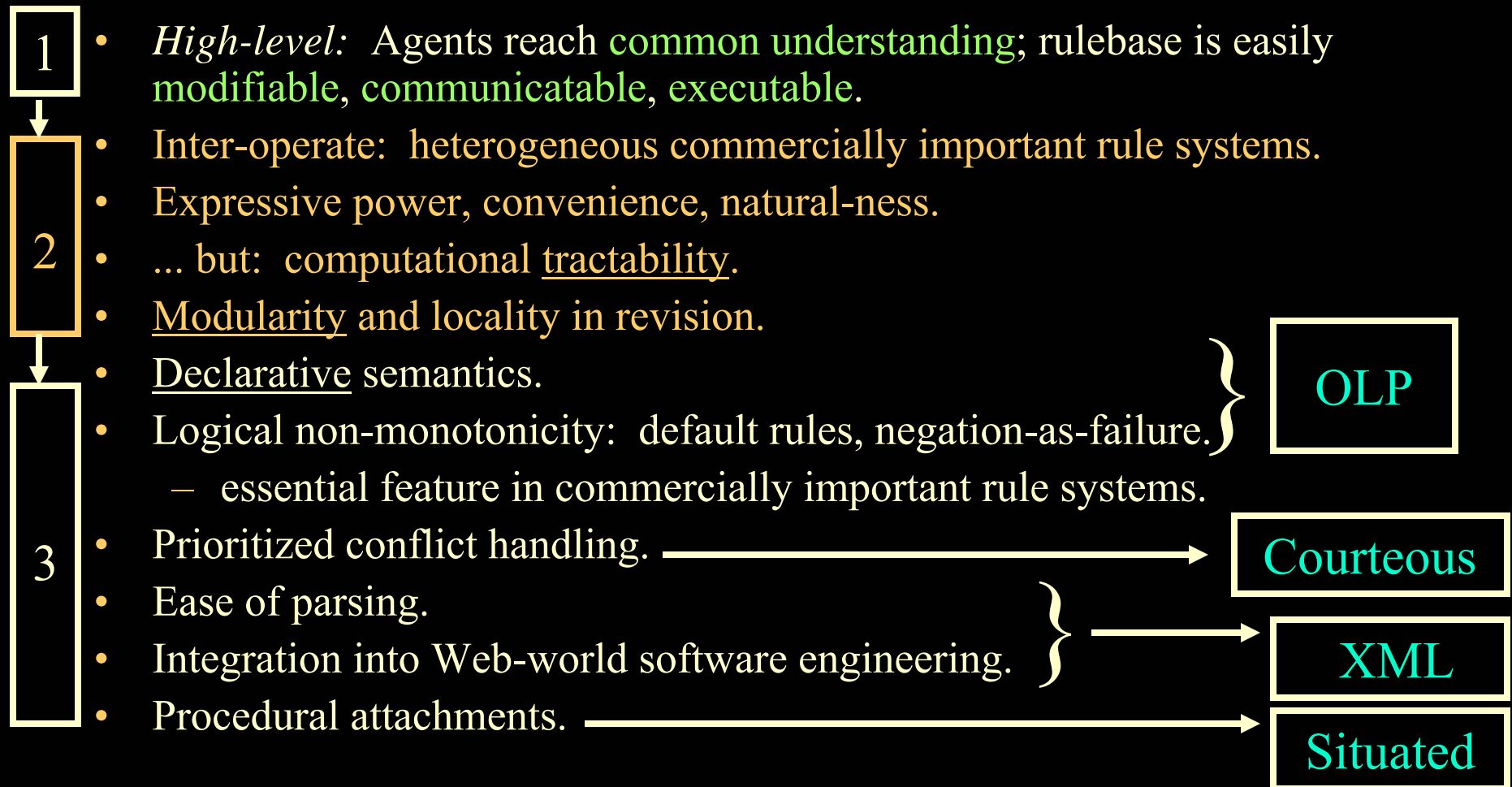
Supports *referencing OWL (or other)* ontologies:

URI predicate name	refers to	class or property
(in RuleML rule)		(in OWL axioms)

This was pioneered in SweetDeal using SweetRules.

The same approach was then taken in SWRL V0.5+.

Criteria for SW Rule Representation



Outline of Part A.

A. Core -- KR Languages and Standards

1. Intro
2. Overview of Logic Knowledge Representations and Standards
3. Horn Logic / Horn LP
4. Nonmonotonic LP
5. Procedural Attachments
6. RuleML
7. Combining Rules with Ontologies
8. Datatypes
9. Review of OWL and RDF
10. SWRL
11. Additional Aspects and Approaches

URI Ontological Reference Approach

- A RuleML predicate (or individual / logical function) is specified as a URI, that refers to a predicate (or individual / logical function, respectively) specified in another KB, e.g., in OWL.
- Application pilot and first use case: in SweetDeal e-contracting system (design 2001, prototype early 2002).
- Approach was then soon incorporated into RuleML and adopted in SWRL design (which is based mainly on RuleML), and used heavily there.
- Issue: want to scope precisely which premises in an overall ontological KB are being referenced.
 - Approach in our current work: define a KB (e.g., a subset/module) and reference that KB.

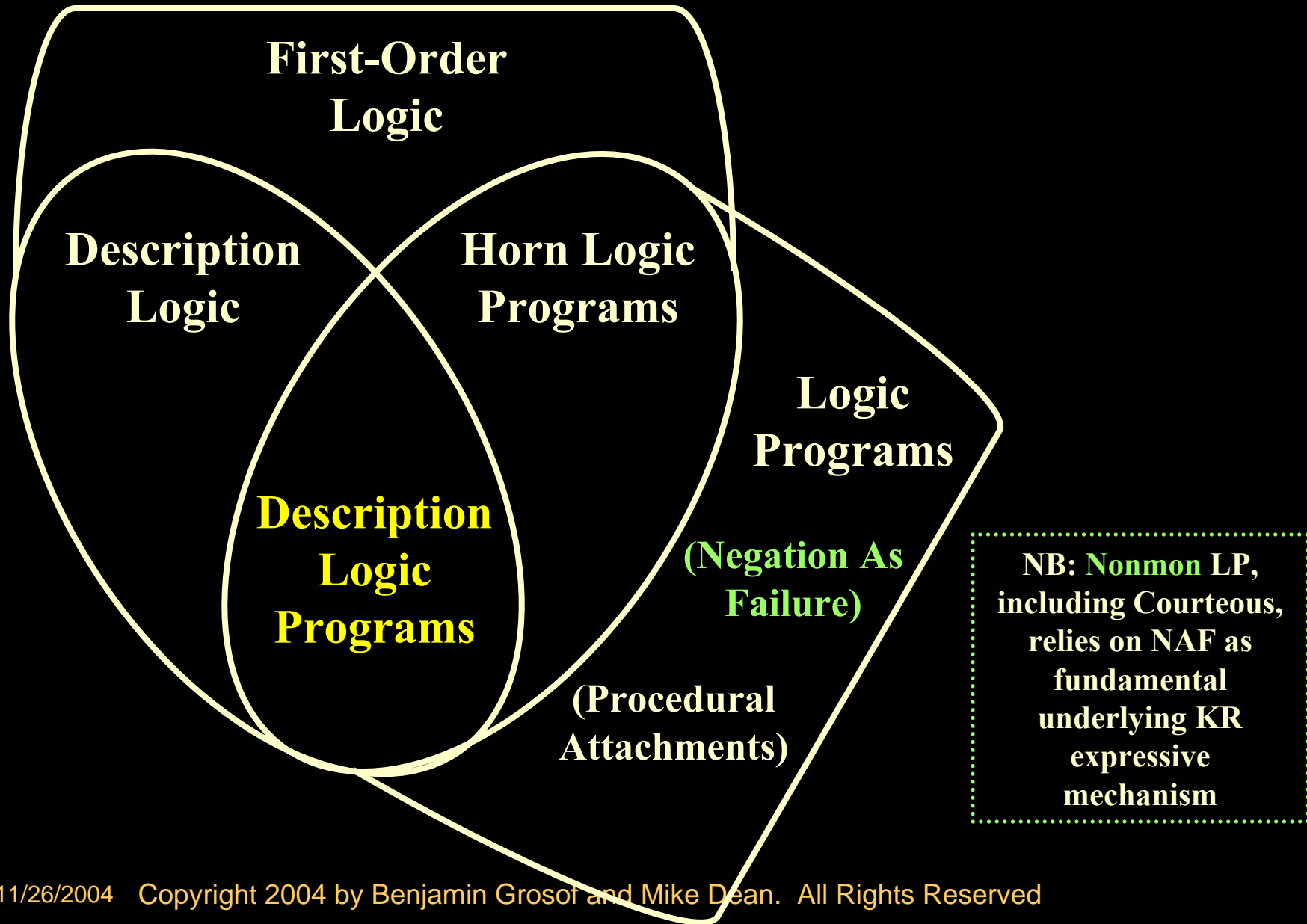
URI Ontological Reference Approach Example, in RuleML

```
payment(?R,base,?Payment) <-  
http://xmlcontracting.org/sd.owl#result(co123,?R) AND  
price(co123,?P) AND quantity(co123,?Q) AND  
multiply(?P,?Q,?Payment) ;
```

SCLP TextFile Format for RuleML

```
<imp>  
  <_head> <atom>  
    <_opr><rel>payment</_opr></rel>    <tup>  
      <var>R</var> <ind>base</ind> <var>Payment</var>  
    </tup></atom> </_head>  
  <_body>  
    <andb>  
      <atom> <_opr>  
        <rel href= "http://xmlcontracting.org/sd.owl#result" />  
      </_opr> <tup>  
        <ind>co123</ind> <var>Cust</var>  
      </tup> </atom>  
    .. </andb> </_body> </imp>
```

Venn Diagram: Expressive Overlaps among KR's



Overview of DLP KR Features

- DLP captures **completely** a subset of DL, comprising RDFS & more
- **RDFS subset** of DL permits the following statements:
 - Subclass, Domain, Range, Subproperty (also SameClass, SameProperty)
 - instance of class, instance of property
- DLP also completely captures **more** DL statements beyond RDFS:
 - Using Intersection connective (conjunction) in class descriptions
 - Stating that a property (or inverse) is Transitive or Symmetric
 - Using Disjunction or Existential in a *subclass* expression
 - Using Universal in a *superclass* expression
 - ∴ “**OWL Feather**” – subset of OWL Lite
 - Update summer 2004: New Related Effort is “OWL Lite Minus” by WSMO
- DLP++: enhanced translation into LP can express even more of DL:
 - Using explicit equality, skolemization, integrity constraints
 - Using NAF, for T-box reasoning
 - (*Part still in progress.*)

DLP-Fusion:

Technical Capabilities Enabled by DLP

- LP rules "on top of" DL ontologies.
 - E.g., LP imports DLP ontologies, with completeness & consistency
 - Consistency via completeness. (Also, Courteous LP is always consistent.)
- Translation of LP rules to/from DL ontologies.
 - E.g., develop ontologies in LP (or rules in DL)
- Use of efficient LP rule/DBMS engines for DL fragment.
 - E.g., run larger-scale ontologies
 - \Rightarrow Exploit: Scalability of LP/DB engines \gg DL engines , as $|\text{instances}| \uparrow$.
- Translation of LP conclusions to DL.
- Translation of DL conclusions to LP.
- Facilitate rule-based mapping between ontologies / “contexts”

Expressiveness of SWRL (V0.6)

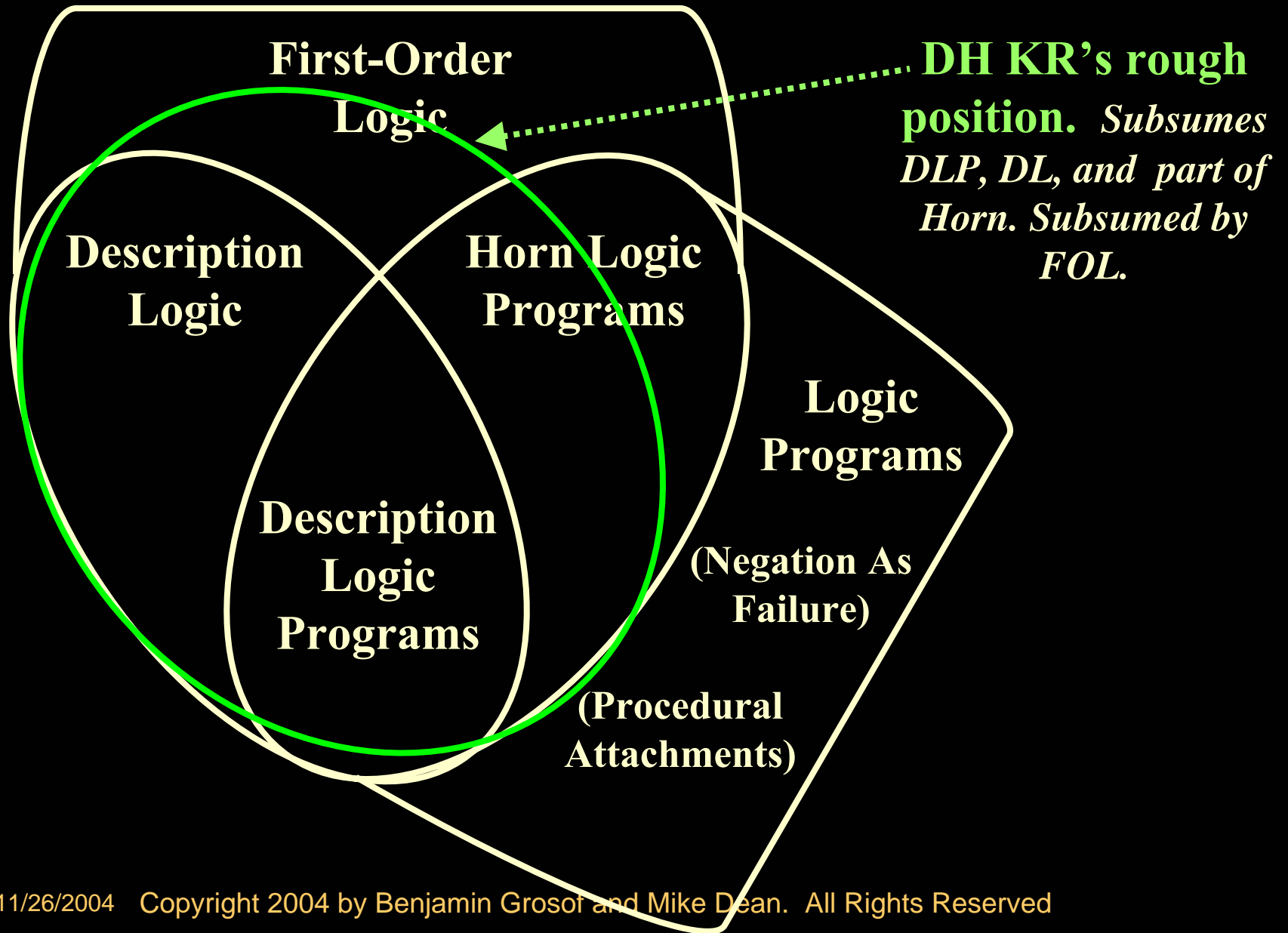
SWRL expressiveness =

1. OWL-DL (i.e., SHOIQ Description Logic (DL) which is an expressive subset of FOL)
 2. + Datalog Horn FOL rules, syntactically RuleML, where each predicate may be:
 - OWL named class (thus arity 1)
 - More generally, may use a complex class, but this is expressively inessential – can just replace by a named class and define that named class as equivalent to the complex class.
 - OWL property (thus arity 2)
 - OWL data range (thus arity 1)
 - RDF datatype
 - set of literal values, e.g., {3} or {1,2,3,4,5} or {"Fred","Sue"}
 3. + some built-ins (mainly XML-Schema datatypes and operations on them)
 - This is new with V0.6
 - Plan: the set of built-ins is extensible
- The fundamental KR is an expressive subset of FOL
 - We'll call it "DH" here. (It doesn't have a real name yet.)
 - Its expressiveness is equivalent to: DL + Datalog Horn.

“Warning Label” for SWRL

1. The Theory of DH is Little Explored Territory as a KR.
 - In its full generality, DH is a relatively unstudied fragment of FOL.
 - Its worst-case computational complexity is undecidable and is not known to be better than that of full FOL (e.g., for the propositional case).
 - There are not yet efficient algorithms known for inferencing on it “natively” as a KR.
2. To ensure extensibility of SWRL rulebases to include LP features that go beyond Horn expressiveness, restrict the OWL ontologies used within SWRL to be in the DLP subset of OWL-DL. E.g.:
 - If you want to use nonmonotonicity / negation-as-failure / priorities in your rules
 - If you want to use procedural attachments that go beyond the SWRL built-ins
 - E.g., effectors/actions with side effects

Venn Diagram: Expressive Overlaps among KR's



Design Perspective

Alternative points in design space:

1. partial LP + full DL = SWRL V0.6

versus

2. full LP + partial DL = SCLP RuleML V0.8+
(with DLP OWL2RuleML)

(SCLP = Situated Courteous Logic Programs KR)

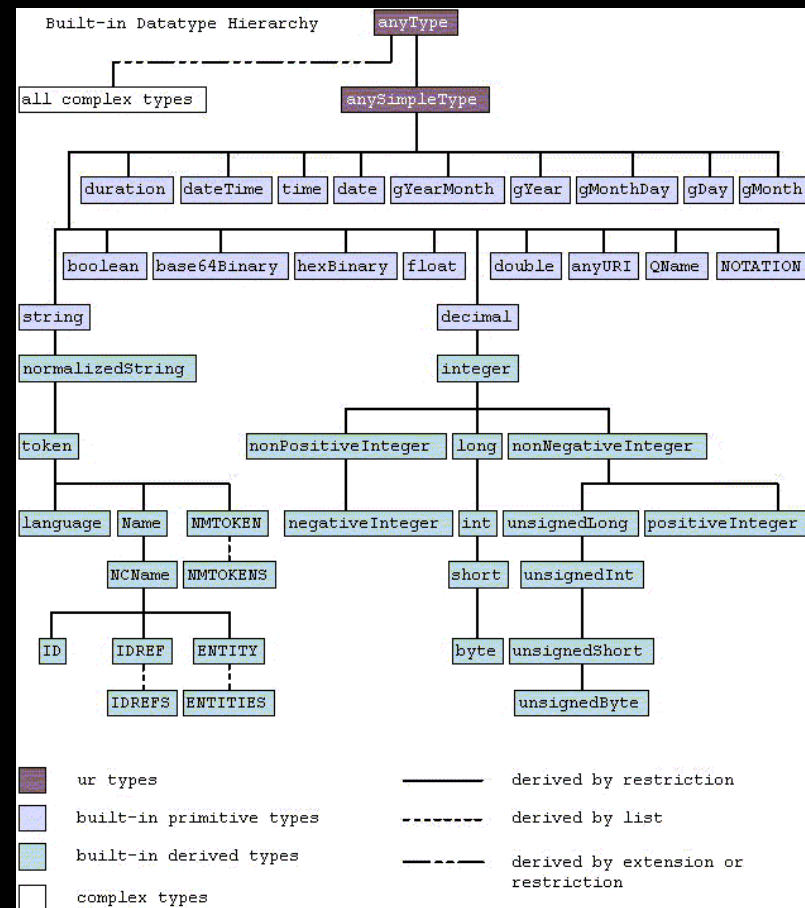
Outline of Part A.

A. Core -- KR Languages and Standards

1. Intro
2. Overview of Logic Knowledge Representations and Standards
3. Horn Logic / Horn LP
4. Nonmonotonic LP
5. Procedural Attachments
6. RuleML
7. Combining Rules with Ontologies
8. Datatypes
9. Review of OWL and RDF
10. SWRL
11. Additional Aspects and Approaches

XML Schema Datatypes

- Supports validation of XML character data
- W3C Recommendation
 - <http://www.w3.org/TR/xmlschema-2/>



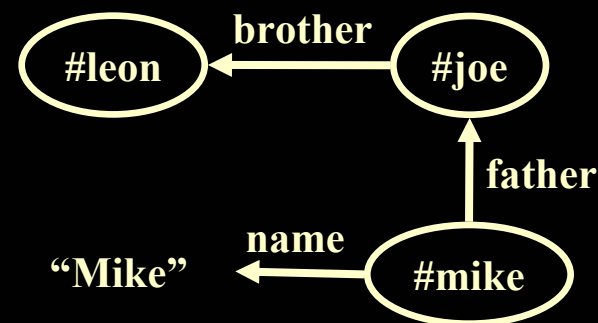
Outline of Part A.

A. Core -- KR Languages and Standards

1. Intro
2. Overview of Logic Knowledge Representations and Standards
3. Horn Logic / Horn LP
4. Nonmonotonic LP
5. Procedural Attachments
6. RuleML
7. Combining Rules with Ontologies
8. Datatypes
9. Review of OWL and RDF
10. SWRL
11. Additional Aspects and Approaches

Resource Description Framework (RDF)

- Defines graph data model
- RDF/XML provides the serialization syntax for the Semantic Web
- RDF Schema adds
 - Classes
 - Person is a subclass of Mammal
 - Properties
 - father is a subproperty of parent
- Datatype support recently added
 - Uses XML Schema Datatypes
- W3C Recommendation
 - <http://www.w3.org/TR/rdf-primer/>



```
<rdf:Description rdf:about="#mike">
  <name>Mike</name>
  <father>
    <rdf:Description rdf:about="#joe">
      <brother rdf:resource="#leon"/>
    </rdf:Description>
  </father>
</rdf:Description>
```

OWL Web Ontology Language I

- Adds expressive power beyond RDF Schema
 - Restrictions
 - Every Person has 1 father
 - The parent of a Person is a Person
 - Class expressions
 - Man is the intersection of Person and Male
 - A Father is a Person with at least one child
 - Equivalence
 - #mike is the same individual as #michael
 - ont1:Car is the same class as ont2:Automobile
 - Properties of properties
 - parent is the inverse of child
 - ancestor is transitive
 - spouse is symmetric
 - A Person can be uniquely identified by his homepage

OWL Web Ontology Language II

- Multiple dialects
 - OWL Lite: basic capability
 - OWL DL: maximum decidable subset
 - OWL Full: compatibility with arbitrary RDF
- W3C Recommendation
 - <http://www.w3.org/TR/owl-features/>

Outline of Part A.

A. Core -- KR Languages and Standards

1. Intro
2. Overview of Logic Knowledge Representations and Standards
3. Horn Logic / Horn LP
4. Nonmonotonic LP
5. Procedural Attachments
6. RuleML
7. Combining Rules with Ontologies
8. Datatypes
9. Review of OWL and RDF
10. SWRL
11. Additional Aspects and Approaches

Semantic Web Rule Language (SWRL)

- Motivation:
 - Extend expressiveness of OWL
- Combines
 - OWL (DL and Lite)
 - Unary/Binary Datalog Horn RuleML
- Developed by the Joint US/EU ad hoc Agent Markup Language Committee (JC), in collaboration with RuleML Initiative
 - JC developed DAML+OIL
- Acknowledged as a W3C Member Submission
 - Allows use by a future W3C Semantic Web Rules Working Group
- Multiple syntaxes
 - Abstract Syntax (extends the OWL Abstract Syntax)
 - XML Concrete Syntax (extends the OWL XML Presentation Syntax)
 - RDF Concrete Syntax

SWRL Expressiveness

- Recall earlier slides (section A.7.) on SWRL's expressiveness, computational complexity, and “warning label”.

SWRL Ontology

- Extends owl:Ontology
- ```
<swrlx:Ontology swrlx:name = xsd:anyURI >
 Content: (owlx:VersionInfo |
 owlx:PriorVersion |
 owlx:BackwardCompatibleWith |
 owlx:IncompatibleWith |
 owlx:Imports |
 owlx:Annotation |
 owlx:Class |
 owlx:EnumeratedClass |
 owlx:SubClassOf |
 owlx:EquivalentClasses |
 owlx:DisjointClasses |
 owlx:DatatypeProperty |
 owlx:ObjectProperty |
 owlx:SubPropertyOf |
 owlx:EquivalentProperties |
 owlx:Individual |
 owlx:SameIndividual |
 owlx:DifferentIndividuals |
 ruleml:imp |
 ruleml:var)*
</swrlx:Ontology>
```

## *SWRL Rule*

- `<ruleml:imp>`  
*Content:* ( `_rlab?`,  
`owlx:Annotation*`,  
`_body`,  
`_head` )  
`</ruleml:imp>`



## *\_body*

- Specifies the “if” part of the rule
- `<ruleml:_body>`  
*Content: ( swrlx:atom\* )*  
`</ruleml:_body>`

## *\_head*

- Specifies the “then” part of the rule
- `<ruleml:_head>`  
*Content: ( swrlx:atom\* )*  
`</ruleml:_head>`

# *SWRL Atoms*

- The rule head and body consist of sets of SWRL atoms
  - `swrlx:classAtom`
  - `swrlx:datarangeAtom`
  - `swrlx:individualPropertyAtom`
  - `swrlx:datavaluedPropertyAtom`
  - `swrlx:sameIndividualAtom`
  - `swrlx:differentIndividualsAtom`
  - `swrlx:builtinAtom`

# *classAtom*

- Tests or asserts that the instance is of the specified class
- Can use a named class or class expression
- `<swrlx:classAtom>`  
*Content:* ( *owlx:description*,  
*swrlx:iObject* )  
`</swrlx:classAtom>`
- `<swrlx:classAtom>`  
`<owlx:Class`  
`owlx:name="&foaf;Person" />`  
`<ruleml:var>person</ruleml:var>`  
`</swrlx:classAtom>`

# *datarangeAtom*

- Tests or asserts that the literal value or variable is of the specified datatype
- `<swrlx:datarangeAtom>`  
*Content:* ( *owlx:datarange*,  
*swrlx:dObject* )  
`</swrlx:datarangeAtom>`
- `<swrlx:datarangeAtom>`  
    `<owlx:Datatype`  
        `owlx:name="xsd:int" />`  
    `<ruleml:var>age</ruleml:var>`  
`</swrlx:datarangeAtom>`

# *individualPropertyAtom*

- Tests or asserts the value of an owl:ObjectProperty
- ```
<swrlx:individualPropertyAtom
  swrlx:property = xsd:anyURI {required}
>
Content: ( swrlx:iObject,
           swrlx:iObject )
</swrlx:individualPropertyAtom>
```
- ```
<swrlx:individualPropertyAtom
 swrlx:property="&foaf;organization">
 <ruleml:var>person</ruleml:var>
 <ruleml:var>company</ruleml:var>
</swrlx:individualPropertyAtom>
```

# *datavaluedPropertyAtom*

- Tests or asserts the value of an owl:DatatypeProperty
- ```
<swrlx:datavaluedPropertyAtom
  swrlx:property = xsd:anyURI {required}
>
Content: ( swrlx:iObject,
           swrlx:dObject )
</swrlx:datavaluedPropertyAtom>
```
- ```
<swrlx:datavaluedPropertyAtom
 swrlx:property="&foaf:name">
 <ruleml:var>person</ruleml:var>
 <ruleml:var>name</ruleml:var>
</swrlx:datavaluedPropertyAtom>
```

# *Example SWRL Rule*

```
<ruleml:imp>
 <ruleml:_rlab ruleml:href="#uncle" />
 <owlx:Annotation>
 <owlx:Documentation>parent's brother
 </owlx:Documentation>
</owlx:Annotation>
<ruleml:_body>
 <swrlx:individualPropertyAtom
 swrlx:property="&family;parent">
 <ruleml:var>child</ruleml:var>
 <ruleml:var>parent</ruleml:var>
 </swrlx:individualPropertyAtom>
 <swrlx:individualPropertyAtom
 swrlx:property="&family;brother">
 <ruleml:var>parent</ruleml:var>
 <ruleml:var>uncle</ruleml:var>
 </swrlx:individualPropertyAtom>
</ruleml:_body>
<ruleml:_head>
 <swrlx:individualPropertyAtom
 swrlx:property="&family;uncle">
 <ruleml:var>child</ruleml:var>
 <ruleml:var>uncle</ruleml:var>
 </swrlx:individualPropertyAtom>
</ruleml:_head>
</ruleml:imp>
```



## *sameIndividualAtom*

- Explicitly test for equality
- `<swrlx:sameIndividualAtom>`  
*Content: ( swrlx:iObject\* )*  
`</swrlx:sameIndividualAtom>`
- `<swrlx:sameIndividualAtom>`  
    `<ruleml:var>person1</ruleml:var>`  
    `<ruleml:var>person2</ruleml:var>`  
`</swrlx:sameIndividualAtom>`

# *differentIndividualsAtom*

- Explicitly test for inequality
- `<swrlx:differentIndividualsAtom>`  
*Content: ( swrlx:iObject\* )*  
`</swrlx:differentIndividualsAtom>`
- `<swrlx:differentIndividualsAtom>`  
    `<ruleml:var>person1</ruleml:var>`  
    `<ruleml:var>person2</ruleml:var>`  
`</swrlx:differentIndividualsAtom>`

# *builtinAtom*

- Provides access to builtin functions
- ```
<swrlx:builtinAtom
  swrlx:builtin = xsd:anyURI {required}
>
Content: ( swrlx:dObject* )
</swrlx:builtinAtom>
```
- ```
<swrlx:builtinAtom
 swrlx:builtin="&swrlb;multiply">
 <ruleml:var>inches</ruleml:var>
 <ruleml:var>feet</ruleml:var>
 <owlx:DataValue
 owlx:datatype="&xsd;int">12</owlx:DataValue>
</swrlx:builtinAtom>
```

# *SWRL Builtins*

- Motivation
  - Ontology translation
    - Unit conversion (inches = feet \* 12)
  - Defining OWL classes in terms of datatype values
    - An Adult is a Person with age > 17
- Added in SWRL 0.6
  - Limited to side-effect free builtins
- Collected from multiple sources
  - XQuery
  - Other rule systems
  - Programming language libraries

# SWRL Builtins

## Comparison

equal  
notEqual  
lessThan  
lessThanOrEqual  
greaterThan  
greaterThanOrEqual

## Math

add  
subtract  
multiply  
divide  
integerDivide  
mod  
pow  
unaryPlus  
unaryMinus  
abs  
ceiling  
floor  
round  
roundHalfToEven  
sin  
cos  
tan

## Booleans

booleanNot

## Strings

stringEqualIgnoreCase  
stringConcat  
substring  
stringLength  
normalizeSpace  
upperCase  
lowerCase  
translate  
contains  
containsIgnoreCase  
startsWith  
endsWith  
substringBefore  
substringAfter  
matches  
replace  
tokenize

## Lists

listConcat  
listIntersection  
listSubtraction  
member  
length  
first  
rest  
sublist  
empty

## Date, Time, and Duration

yearMonthDuration  
dayTimeDuration  
dateTime  
date  
time  
addYearMonthDurations  
subtractYearMonthDurations  
multiplyYearMonthDuration  
divideYearMonthDurations  
addDayTimeDurations  
subtractDayTimeDurations  
multiplyDayTimeDurations  
divideDayTimeDurations  
subtractDates  
subtractTimes  
addYearMonthDurationToDateTime  
addDayTimeDurationToDateTime  
subtractYearMonthDurationFromDateTime  
subtractDayTimeDurationFromDateTime  
addYearMonthDurationToDate  
subtractYearMonthDurationFromDate  
addDayTimeDurationToTime  
subtractDayTimeDurationFromTime  
subtractDateTimesYieldingYearMonthDuration  
subtractDateTimesYieldingDayTimeDuration

## URIs

resolveURI  
anyURI

See <http://www.daml.org/rules/proposal/builtins> for details

# *Rule Using a Builtin*

```
<ruleml:imp>
 <ruleml:_body>
 <swrlx:datavaluedPropertyAtom swrlx:property="&data:length">
 <ruleml:var>instance</ruleml:var>
 <ruleml:var>feet</ruleml:var>
 </swrlx:datavaluedPropertyAtom>
 <swrlx:builtinAtom swrlx:builtin="&swrlb:multiply">
 <ruleml:var>inches</ruleml:var>
 <ruleml:var>feet</ruleml:var>
 <owlx:DataValue
owlx:datatype="&xsd:int">12</owlx:DataValue>
 </swrlx:builtinAtom>
 </ruleml:_body>
 <ruleml:_head>
 <swrlx:datavaluedPropertyAtom
swrlx:property="&domain:length">
 <ruleml:var>instance</ruleml:var>
 <ruleml:var>inches</ruleml:var>
 </swrlx:datavaluedPropertyAtom>
 </ruleml:_head>
</ruleml:imp>
```

# *SWRL Implementations Today*

- swrl2clips
  - Translates rules for use with CLIPS or JESS
- Hoolet
  - Translates rules for use with the Vampire FOL reasoner
- SweetJena
  - Translates rules for use with Jena
- Solanki, et al
  - Augments Semantic Web Service descriptions with SWRL rules
- Christine Golbreich
  - Uses SWRL with Protégé, JESS, and Racer
- ...

# *Outline of Part A.*

## A. Core -- KR Languages and Standards

1. Intro
2. Overview of Logic Knowledge Representations and Standards
3. Horn Logic / Horn LP
4. Nonmonotonic LP
5. Procedural Attachments
6. RuleML
7. Combining Rules with Ontologies
8. Datatypes
9. Review of OWL and RDF
10. SWRL
11. Additional Aspects and Approaches



# *SWRL-FOL*

- See separate SWRL-FOL material.

# *FOL RuleML*

- See separate FOL RuleML material.

# *Need for Other Kinds of Ontologies besides OWL*

- Kinds of ontologies practically/commercially important in the world today\*:
  - SQL DB schemas, E-R, UML, OO inheritance hierarchies, LP/FOL predicate/function signatures; equations and conversion-mapping functions; XML-Schema
- OWL is still emerging.
- Overall relationship of OWL to the others is as yet largely unclear
  - There are efforts on some aspects, incl. UML
- OWL cannot represent the nonmon aspects of OO inheritance
- OWL does not yet represent, except quite awkwardly:
  - n-ary signatures
  - ordering aspects of XML-Schema
- (\*NB: Omitted here are statistically flavored ontologies that result from inductive learning and/or natural language analysis.)

## *Need for Other Kinds of Ontologies besides OWL, cont.'d*

- Particularly interesting:
  - OO-ish nonmon taxonomic/frames
  - Equations and context mappings cf. ECOIN – can be represented in FOL or often in LP
  - OWL DL beyond DLP
- Builtins (sensed) are a relatively simple kind of shared ontology
  - SWRL V0.6 and forthcoming RuleML V0.9

# Default Inheritance cf. OO

- Ubiquitous in object-oriented programming languages & applications
- Default nature increases reuse, modularity
- **OWL/DL fundamentally incapable of representing, since monotonic**
- Requirements of semantic web service process ontologies:
  - Need to jibe with mainstream web service development methodologies, based on Java/C#/C++
- Approach: Represent OO default-inheritance ontologies using nonmon LP rules
  1. [Grosf & Bernstein] Courteous Inheritance approach
    - Transforms inheritance into Courteous LP in RuleML
    - Represents MIT Process Handbook (ancestor of PSL)
      - 5,000 business process activities; 38,000 properties/values
      - Linear-size transform ( $n + \text{constant}$ ).
    - SweetPH prototype: extends SweetRules
  2. [Yang & Kifer] approach
    - Transform inheritance into essentially Ordinary LP
    - Extends Flora-2

# *“Object Oriented Syntax” for Rules*

- RuleML slots for arguments
- SWRL RDF-triple style
- F-Logic, TRIPLE: frame syntax
  - *In progress: Add as feature to RuleML*

# More Aspects and Approaches

- Relationship of rules to RDF query/access languages and tools
  - XQuery too
- Explicit equality (and equivalence) reasoning
  - In head of non-fact rules
  - Interaction with nonmonotonicity
  - Related to Herbrand aspect of LP semantics
- Existentials, skolemization
  - RDF blank-nodes, anonymous individuals [Yang & Kifer]
  - Related to Herbrand aspect of LP semantics
- Higher-order syntax: RDF, OWL-Full, Hilog, F-Logic
  - Hilog approach appears promising for OWL-Full and RDF-Full [Kifer; Hayes]
- Reasoning within the KR/language about the results of side-effectful actions:
  - E.g., Golog [Reiter, Lin, *et al*]; Transaction Logic [Kifer *et al*]

*Fundamental KR Challenge in  
Combining Rules with Ontologies:  
Unify FOL/DL More Deeply with Nonmon LP*

- Motivations: Better support KB merging, SWSL, unify SW overall, more of DL/FOL in LP, handle conflicts between DL/FOL KB's, ...
- Approach: “Hypermonotonic” reasoning [Grosf]
  - Courteous LP mapped  $\Leftrightarrow$  clausal FOL
    - Courteous LP always sound wrt FOL
    - ... & incomplete wrt FOL
      - Enables: always consistent, robust in merging
    - Mapping is linear-size and local



# *Slideset 3 of*

## *“Semantic Web Rules with Ontologies, and their E-Business Applications”*

*by Benjamin Grosf\* and Mike Dean\*\**

*\*MIT Sloan School of Management, <http://ebusiness.mit.edu/bgrosf>*

*\*\*BBN Technologies, <http://www.daml.org/people/mdean>*

*ISWC-2004 Conference Tutorial (half-day),*

*at 3rd International Semantic Web Conference, Nov. 7, 2004, Hiroshima, Japan*

*Version Date: Oct. 15, 2004*

11/26/2004 Copyright 2004 by Benjamin Grosf and Mike Dean. All Rights Reserved

# *Outline of Part B.*

## B. Tools -- SweetRules, Jena, cwm, and More

*(BREAK in middle)*

1. Commercially Important pre-SW Rule Systems
  - Prolog, production rules, DBMS
2. Overview of SW Rule Generations
3. 1st Gen.: Rudimentary Interoperability and XML/RDF Support
  - CommonRules, SweetRules V1, OWLJessKB
4. 2nd Gen.: Rule Systems within RDF/OWL/SW Toolkits
  - cwm, Jena-2, and others
5. 3rd Gen.: SW Rule Integration and Life Cycle
  - SweetRules V2

# *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.*)

# *Open Source pre-SW Rule Tools: Popular, Mature*

- XSB Prolog [SUNY Stonybrook]
  - Supports Well Founded Semantics for general, non-stratified case
  - Scales well
  - C, with Java front-end available (InterProlog)
- Jess production rules [Sandia Natl. Lab USA]
  - Semi-open source
  - Java
  - Successor to: CLIPS in C [NASA]
- SWI Prolog [Netherlands]

# *Overview of SW Rule Tool Generations*

Analysis: 3 Generations of SW rule tools to date

1. Rudimentary Interoperability and XML/RDF Support
  - CommonRules, SweetRules V1, OWLJessKB
2. Rule Systems within RDF/OWL/SW Toolkits
  - cwm, Jena-2, and others – incl. SWRL tools
3. SW Rule Integration and Life Cycle
  - SweetRules V2

# *Outline of Part B.*

## B. Tools -- SweetRules, Jena, cwm, and More

*(BREAK in middle)*

1. Commercially Important pre-SW Rule Systems
  - Prolog, production rules, DBMS
2. Overview of SW Rule Generations
3. 1st Gen.: Rudimentary Interoperability and XML/RDF Support
  - CommonRules, SweetRules V1, OWLJessKB
4. 2nd Gen.: Rule Systems within RDF/OWL/SW Toolkits
  - cwm, Jena-2, and others
5. 3rd Gen.: SW Rule Integration and Life Cycle
  - SweetRules V2

# *IBM CommonRules I*

- Java library. V3.3 is current version. (V1.0 was 1999.)
- Available for researchers under trial license on IBM AlphaWorks
- Supports Situated Courteous LP
- Defined own markup language – BRML
  - Plan: migrate to RuleML in V4.0
- Defined own presentation (string) language
- Courteous Compiler component: transforms CLP → OLP
- Native forward-direction SCLP inferencing engine
  - Does not scale up well (was not intended to)
  - Stratified-only case of NAF

# *IBM CommonRules II*

- Translation  $\leftrightarrow$  several other rule systems:
  - XSB Prolog
  - Smodels (forward OLP, in Prolog syntax)
  - KIF(Translation enables true semantic interoperability.)
- Support for adding new/user aproc's is fairly rudimentary
  - Has basic built-ins
- Sensing aspect of core inferencing procedure is sophisticated
  - Lacks conflict handling for sensors, however
- Forerunner to RuleML
- Forerunner to SweetRules



# SweetRules V1

- 2001. [MIT Sloan: Grosf, Poon, & Kabbaj]
- SCLP RuleML Translation and Inferencing
  - Enhance functionality of IBM CommonRules
- Concept prototype
  - Part of SWEET = Semantic Web Enabling Toolkit
- Java, XSLT, command shell script drivers
- Translation ↔ several other rule systems:
  - IBM CommonRules
  - XSB Prolog
  - Smodels (forward OLP, in Prolog syntax)
  - KIF
- No native inferencing engine
  - All inferencing indirect via translation
- Used in SweetDeal V1
  - **e-contracting application** prototype

# SweetOnto V1

- 2003. [U. Karlsruhe *et al*: Motik, Volz, Bechhofer, Grosf; also Horrocks, Decker]
- Translates DLP OWL → RuleML
- A.k.a. DLP component of KAON
- Java

# OWLJessKB

- Translates OWL ontologies and instances for use with the Java Expert System Shell (Jess), a popular semi-open-source production rule system
  - Supports some DLP reasoning
  - Can be augmented with JESS Rules
- Sample rule
  - ```
(defrule uncle
  "a parent's brother is an uncle"
  (triple (predicate "http://example.org/family#parent")
    (subject ?child)
    (object ?parent))
  (triple (predicate "http://example.org/family#brother")
    (subject ?parent)
    (object ?uncle))
  =>
  (assert (triple
    (predicate "http://example.org/family#uncle")
    (subject ?child)
    (object ?uncle))))
```
- More information
 - <http://edge.cs.drexel.edu/assemblies/software/owljesskb/>

Outline of Part B.

B. Tools -- SweetRules, Jena, cwm, and More

(BREAK in middle)

1. Commercially Important pre-SW Rule Systems
 - Prolog, production rules, DBMS
2. Overview of SW Rule Generations
3. 1st Gen.: Rudimentary Interoperability and XML/RDF Support
 - CommonRules, SweetRules V1, OWLJessKB
4. 2nd Gen.: Rule Systems within RDF/OWL/SW Toolkits
 - cwm, Jena-2, and others
5. 3rd Gen.: SW Rule Integration and Life Cycle
 - SweetRules V2

- Python-based open source Semantic Web toolkit from W3C/MIT
 - Supports Notation 3 as well as RDF-XML
 - Includes a forward-chaining reasoner
 - Supports a variety of rule builtins
- Sample N3 rules:

```
1. { ?child family:parent ?parent .  
    ?parent family:brother ?uncle }  
=>  
{ ?child family:uncle ?uncle }
```

```
2. { ?instance ont1:length ?feet .  
    ( ?feet "12" ) math:product ?inches }  
=>  
{ ?instance ont2:length ?inches }
```

- Semantic Web Tutorial using N3
 - <http://www.w3.org/2000/10/swap/doc/>

Jena 2

- Java-based open source Semantic Web toolkit from HP Labs
 - Parser
 - Serializer
 - Persistence
 - Query
 - Reasoner
- Jena 2 includes a general purpose rule engine
 - Forward-chaining RETE (cf. subset of production rules)
 - Backward-chaining LP with tabling
 - Hybrid forward/backward rules
 - Used primarily to implement OWL Lite reasoner
 - Available for general use
 - Supports a basic set of builtins
 - Limited expressively in various ways, however (e.g., nonmon, logical functions, procedural attachments).

Jena 2, cont.'d

- Important because
 - Most Java Semantic Web developers are already using Jena
 - Rules work directly on RDF graph – no need to copy in/out of rule working memory
- Sample rules:
 - [uncle: (?child family:parent ?parent)
(?parent family:brother ?uncle)
-> (?child family:uncle ?uncle)]
 - [convert: (?instance ont1:length ?feet)
product(?feet 12 ?inches)
-> (?instance ont2:length ?inches)]
- More information
 - <http://jena.sourceforge.net/inference/>

SWRL Implementations Today

- Swrl2clips (NB: now also part of SweetRules)
 - Translates rules for use with CLIPS or JESS
- Hoolet
 - Translates rules for use with the Vampire FOL reasoner
- SweetJena (NB: part of SweetRules)
 - Translates rules for use with Jena
- Solanki, et al
 - Augments Semantic Web Service descriptions with SWRL rules
- Christine Golbreich
 - Uses SWRL with Protégé, JESS, and Racer
- ...

Other Tools

- Several other tools were also presented at the WWW-2004 Developer Day Rules on the Web Track
 - OO JDrew: RuleML inferencing
 - Flora-2: extends XSB with Hilog, F-Logic frame syntax
 - Triple: LP rules for RDF manipulation
 - ROWL: rule-based privacy policy markup lang., on top of Jess

Outline of Part B.

B. Tools -- SweetRules, Jena, cwm, and More

(BREAK in middle)

1. Commercially Important pre-SW Rule Systems
 - Prolog, production rules, DBMS
2. Overview of SW Rule Generations
3. 1st Gen.: Rudimentary Interoperability and XML/RDF Support
 - CommonRules, SweetRules V1, OWLJessKB
4. 2nd Gen.: Rule Systems within RDF/OWL/SW Toolkits
 - cwm, Jena-2, and others
5. 3rd Gen.: SW Rule Integration and Life Cycle
 - SweetRules V2

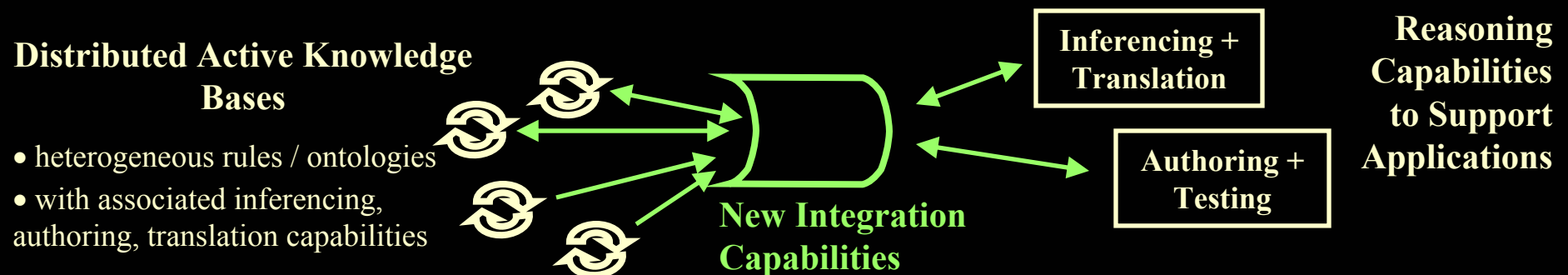
SweetRules V2 Overview

Key Ideas:

- Unite the commercially most important kinds of rule and ontology languages via a new, common knowledge representation (SCLP) in a new standardized syntax (RuleML), including to cope with *heterogeneity* and resolve contradictory *conflicts*.
 - Capture most of the useful expressiveness, interoperably and scalably.
- Combine a large *distributed* set of rule and ontology knowledge bases that each are *active*: each has a different *associated engine* for reasoning capabilities (inferencing, authoring, and/or translation).
- Based on recent fundamental KR theory advances, esp. Situated Courteous Logic Programs (SCLP) and Description Logic Programs.
 - Including semantics-preserving translations between different rule languages/systems/families, e.g., Situated LP \leftrightarrow production rules

Application Areas (prototyped scenarios):

- Policies and authorizations; contracting, supply chain management; retailing, customer relationship management; business process automation and e-services; financial reporting and information; etc.



SweetRules *Concept, Architecture, and Goals*

- **Concept and Architecture: Tools suite** for Rules and RuleML
 - **Translation and interoperability** between heterogeneous rule systems (forward- and backward-chaining) and their rule languages/representations
 - **Inferencing** including **via translation** between rule systems
 - **Authoring** and testing of rulebases
 - **Open, lightweight, extensible, pluggable** architecture overall
- **Goals:**
 - **Research vehicle: embody ideas, implement application scenarios** (e.g., contracting, policies)
 - Situated Courteous Logic Programs (SCLP) KR
 - Description Logic Programs (DLP) KR which is a subset of SCLP KR
 - **Proof of concept for feasibility, including of translations between heterogenous families of rule systems**
 - Encourage others: researchers; industry esp. vendors
 - **Catalyze open source communal toolset efforts**
 - *Initial open-source release on SemWebCentral.org Nov. 2004*

SweetRules *Context and Players*

- Part of SWEET = “Semantic Web Enabling Tools” (2001 –)
 - Other parts:
 - SweetDeal for e-contracting
 - Which uses SweetRules
 - SweetPH for Process Handbook ontologies
 - Which uses SweetRules
- Cross-institutional. Collaborators invited!
 - Originated and coordinated by MIT since 2001
 - Code by MIT, UMBC, U. Karlsruhe, U. Zurich, BBN
 - Uses code by IBM, SUNY Stonybrook, Sandia Natl. Labs, Helsinki, HP
 - More loosely, several other institutions cooperating: BBN, NRC/UNB, Stanford
 - Many more are good targets: subsets of Flora, cwm, Triple, Hoolet, DRS, ?ROWL, KAON (main), JTP, SWI Prolog, ...

Rule and Ontology Languages/Systems That Interoperate via SweetRules and RuleML, Today I

1. RuleML

- Situated Courteous LP extension, V0.8+

2. XSB (the pure subset of it = whole Ordinary LP)

- Backward. Prolog. Fast, scalable, popular. Good support of SQL DB's (e.g., Oracle) via ODBC backend. Full well-founded-semantics for OLP. Implemented in C. By SUNY Stonybrook. Open source on sourceforge. Well documented and supported. Papers.

3. Jess (a pure subset of it = a large subset of Situated Ordinary LP)

- Forward. Production Rules (OPS5 heritage). Flexible, fast, popular. Implemented in Java. By Sandia National Labs. Semi-open source, free for research use. Well documented and supported. Book.
- *Uses recent novel theory for translation between SOLP and Production Rules.*

Rule and Ontology Languages/Systems That Interoperate via SweetRules and RuleML, Today II

4. **IBM CommonRules** (whole = large subset of stratified SCLP)
 - Forward. SCLP. Implemented in Java. Expressive. By IBM Research. Free trial license, on IBM AlphaWorks (since 1999). Considerable documentation. Papers. Piloted.
 - Implements the Courteous Compiler (CC) KR technique.
 - which reduces (S)CLP to equivalent (S)OLP, tractably.
 - Includes bidirectional translators for XSB, KIF, Smodels.
 - Its overall concept and design was point of departure for several aspects of SweetRules

5. **Knowledge Interchange Format (KIF)** (a subset of it = an extension of Horn LP)
 - First Order Logic (FOL). Semi-standard, morphing into Simplified Common Logic ISO standard. Several tools support, e.g., JTP. Research language to date.
 - Note: FOL is superset of DLP and of SWRL's fundamental KR.

Rule and Ontology Languages/Systems That Interoperate via SweetRules and RuleML, Today III

6. **OWL** (the Description Logic Programs subset)
 - Description Logic ontologies. W3C standard. Several tools support, e.g., FACT, RACER, Jena, Hoolet, etc.
 - *Uses recent novel DLP theory for translation between Description Logic and Horn LP.*
7. **Process Handbook** (large subset = subset of SCLP)
 - Frame-style object-oriented ontologies for business processes design, i.e., for services descriptions. By MIT and Phios Corp. (spinoff). Large (5000 business processes). Practical, commercial. Good GUI. Open source license in progress. Available free for research use upon request. Includes extensive textual information too. Well documented and supported. Papers. Book. Dozens of research users.
 - *Uses recent novel SCLP representation of Frames with multiple default inheritance.*
8. **Smodels** (NB: somewhat old version; large subset = finite OLP)
 - Forward. Ordinary LP. Full well-founded-semantics or stable semantics. Implemented in C. By Helsinki univ. Open source. Research system.

Rule and Ontology Languages/Systems That Interoperate via SweetRules and RuleML, Today IV

9. **Jena-2** *currently only with SWRL*

- Forward and backward. Subset of Datalog Horn LP. Plus builtins. Plus RDF & (subset) OWL support. Implemented in Java. By HP. Open source. Popular SW toolkit.

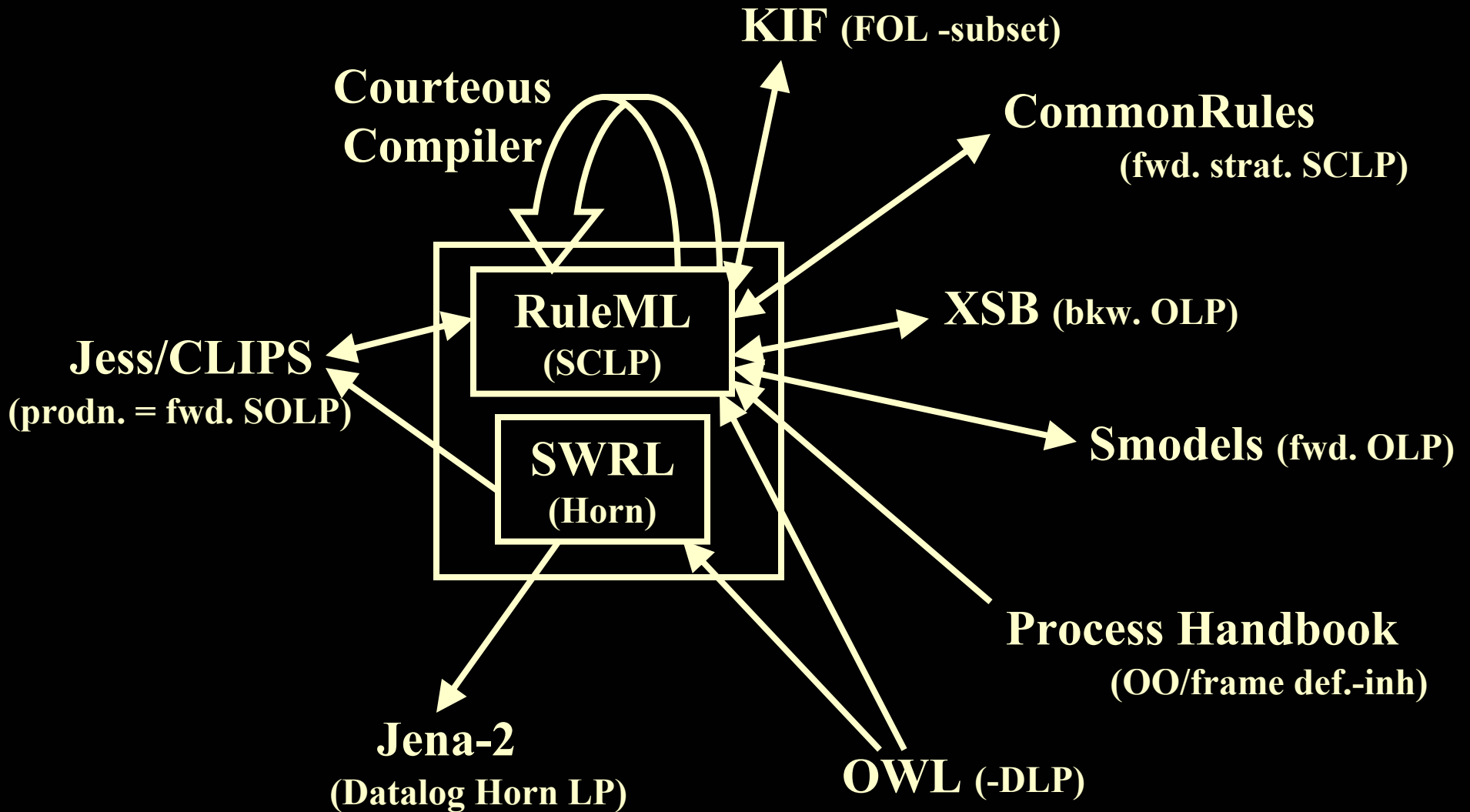
10. **SWRL V0.6** *currently only with DLP OWL, Jena-2, Jess/CLIPS*

- XML syntax (initially). Named-classes-only subset – i.e., Datalog unary/binary Horn FOL. Essentially a subset of RuleML (*in progress: tight convergence*).

SweetRules Capabilities & Components Today I

- **Translators** in and out of RuleML:
 - RuleML \leftrightarrow {XSB, Jess, CommonRules, KIF, Smodels}
 - RuleML \leftarrow {OWL, Process Handbook} (one-direction only)
 - SOLP RuleML \leftarrow SCLP RuleML (Courteous Compiler)
- **Translators** in and out of SWRL (essentially subset of RuleML):
 - SWRL \leftarrow OWL (one-direction only)
 - Jena-2 \leftarrow SWRL (one-direction only)
 - Jess/CLIPS \leftarrow SWRL (one-direction only)
 - *More to come – tighter integration between RuleML and SWRL*
- **Inferencing engines** in RuleML via translation:
 - Simple drivers translate to another rule system, e.g., CommonRules, Jess, or XSB, then run inferencing in that system's engine, then translate back.
 - Observation: Can easily combine components to do other kinds of inferencing, in similar indirect style, by combining various translations and engines.

SweetRules Today: Translators Graph



SweetRules Capabilities & Components Today II

- **Uses Courteous Compiler** to support Courteous feature (prioritized conflict handling) even in systems that don't directly support it, as long as they support negation-as-failure
 - E.g., XSB Prolog, Jess, Smodels
 - Uses Courteous Compiler component from IBM CommonRules
- **Has Include-a-KB** mechanism, similar to owl:imports (prelim. RuleML V0.9)
 - Include a remote KB that is translatable to RuleML
- Uses IBM CommonRules translators: CommonRules \leftrightarrow {XSB, KIF, Smodels}
- Some components have distinct names (for packaging or historical reasons):
 - **SweetCR** translation & inferencing RuleML \leftrightarrow IBM CommonRules
 - **SweetXSB** translation & inferencing RuleML \leftrightarrow XSB
 - **SweetJess** translation & inferencing RuleML \leftrightarrow Jess
 - **SweetOnto** translation {RuleML, SWRL} \leftarrow OWL + RDF-facts
 - **SweetPH** translation RuleML \leftarrow Process Handbook
 - **SweetJena** translation & inferencing Jena-2 \leftarrow SWRL

SweetRules Capabilities & Components Today III

- Code base: Java, XSLT; convenience shell scripts (for testing drivers)
- **Pluggability & Composition Architecture** with detailed interfaces
 - Add your own translator/inferencing-engine/authoring/testing tools
 - Compose tools automatically, e.g.:
 - translator1 \otimes translator2
 - translator \otimes inferencing-engine
 - Search for tools

SweetRules Capabilities & Components Today IV

- **Authoring and Testing front-end:** *currently rudimentary, partial*
 - Command-line UI + Dashboard GUI with set of windows
 - Edit rulebases. Run translations. Run inferencing. Compare.
 - Edit in RuleML. Edit in other rule systems' syntaxes. Compare.
 - View human-oriented presentation syntax. View XML syntax. (Future: RDF.)
- **Validators:** *currently rudimentary, partial*
 - Detect violations of expressive restrictions, required syntax

SweetRules V2 API's Design

- See separate SweetRules V2 javadoc material.

SweetRules V2 Demo Examples

- See separate SweetRules V2 demo examples material.

SweetRules: More Goals

- Additional Goals:
 - More meat to pluggable composition architecture
 - More authoring/UI capabilities
 - More SWRL support, more tightly integrated with RuleML overall
 - More wrt additional kinds of rule systems:
 - ECA rules, SQL (needs some theory work, e.g., events for ECA)
 - RDF-Query and XQuery
 - More wrt connections-to / support-of web services:
 - Importing knowledge bases / modules, procedural attachments, translation/inferencing, events, ...
 - Explore applications in services, e.g., policies, contracts
- More Collaborators Invited!
 - Many more rule/ontology systems are good targets for interoperation/translation:
 - Flora, cwm, Triple, Hoolet, DRS, ROWL, KAON, JTP, SWI Prolog, ...

More about Combining Rules with Ontologies

There are several ways to use SweetRules to combine rules with ontologies:

1. **By reference:** via URI as name for predicate
2. **Translate DLP** subset of OWL into RuleML (or SWRL)
 - Then can **add SCLP** rules
 - E.g., add Horn LP rules and built-in sensors
⇒ interesting subset of the SWRL V0.6 KR
 - E.g., add default rules or procedural attachments
3. **Translate non-OWL ontologies** into RuleML
 - E.g., object-oriented style with default inheritance
 - E.g., Courteous Inheritance for Process Handbook ontologies
4. Use RuleML (or SWRL) **Rules to map between ontologies**
 - E.g., in the spirit of the Extended COntext Interchange (ECOIN) approach/system.
 - SWRL V0.6 good start for mapping between non-DLP OWL ontologies.

SweetJess [Grosf, Gandhe, & Finin 2002]:

First-of-a-kind Translation Mapping/Tool between LP and OPS5 Production Rules

- Requirement for rules interoperability:
Bridge between multiple families of commercially important rule systems: SQL DB, Prolog, OPS5-heritage production rules, event-condition rules.
- Previously known: SQL DB and Prolog are LP.
- Theory and Tool Challenge: bring production rules and event-condition-action rules to the SW party
- Previously not known how to do even theoretically.
- Situated LP is the KR theory underpinning SweetJess, which:
 - Translates between RuleML and Jess production rules system
- SweetJess V1 implementation available free via Web/email
- SweetJess V2 implementation forthcoming Nov. or Dec. 2004
open source on SemWebCentral as part of SweetRules V2

SweetJess: Translating an Effector Statement

```
<effe>
  <_opr>
    <:rel>giveDiscount</damlRuleML:rel>
  </_opr>
  <_aproc>
    <jproc>
      <meth>setCustomerDiscount</meth>
      <clas>orderMgmt.dynamicPricing</clas>
      <path>com.widgetsRUs.orderMgmt</path>
    </jproc>
  </_aproc>
</effe>
```

Associates with predicate P : an attached procedure A that is side-effectful.

- Drawing a conclusion about P triggers an action performed by A.

jproc = Java attached procedure.

meth, *clas*, *path* = its methodname,
 classname, pathname.

Equivalent in JESS: key portion is:

```
(defrule effect_giveDiscount_1
  (giveDiscount ?percentage ?customer)
  =>
  (effector setCustomerDiscount orderMgmt.dynamicPricing
            (create$ ?percentage ?customer) ) )
```

Example: Notifying a Customer when their Order is Modified

- See B. Grosf paper
 - “Representing E-Commerce Rules Via Situated Courteous Logic Programs in RuleML”, in *Electronic Commerce Research and Applications* journal, 2004
 - Available at <http://ebusiness.mit.edu/bgrosf>

Objectives for Integrating Distributed SW Rules and Ontologies, Motivating SweetRules I

Address “the 5 D’s” of real-world reasoning \Rightarrow *desired improvements*:

- 1. Diversity** – Existing/emerging kinds of ontologies and rules have heterogeneous KR's. *Handle more heterogeneous systems.*
- 2. Distributedness** - of ownership/control of ontology/rule active KB's. *Handle more source active KB's.*
- 3. Disagreement** - Conflict (contradiction) will arise when merging knowledge. *Handle more conflicts.*
- 4. Dynamism** - Updates to knowledge occur frequently, overturning previous beliefs. *Handle higher rate of revisions.*
- 5. Delay** - Computational scalability is vital to achieve the promise of knowledge integration. *Achieve Polynomial-time (\sim databases).*

Objectives for Integrating Distributed SW Rules and Ontologies,

Motivating SweetRules II

BEFORE

Contradictory conflict is globally contagious, invalidates all results.

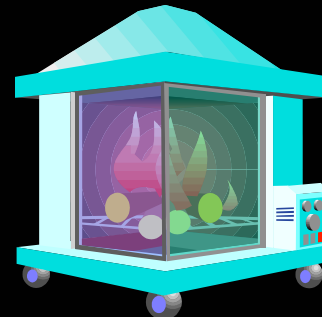


Knowledge integration tackling the 5 D's (esp. diversity and distributedness) is labor-intensive, slow, costly.

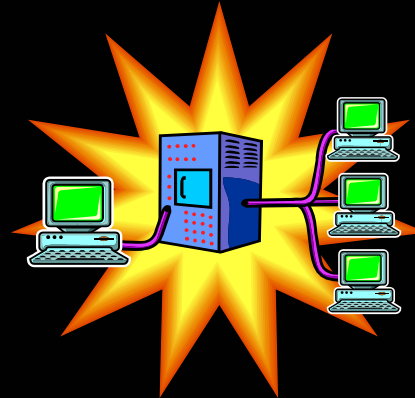


AFTER

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



Knowledge integration is highly automated, faster, cheaper.



Slideset 4 of

“Semantic Web Rules with Ontologies, and their E-Business Applications”

by Benjamin Grosfof and Mike Dean***

**MIT Sloan School of Management, <http://ebusiness.mit.edu/bgrosfof>*

***BBN Technologies, <http://www.daml.org/people/mdean>*

*ISWC-2004 Conference Tutorial (half-day),
at 3rd International Semantic Web Conference, Nov. 7, 2004, Hiroshima, Japan*

Version Date: Oct. 15, 2004

11/26/2004 Copyright 2004 by Benjamin Grosfof and Mike Dean. All Rights Reserved

Top-Level Outline of Tutorial

- *Overview and Get Acquainted*
 - A. **Core** -- KR Languages and Standards
 - B. **Tools** -- SweetRules, Jena, cwm, and More
(BREAK in middle)
 - C. **Applications** -- Policies, Services, and Semantic Integration
- *Windup*

Outline of Part C.

C. Applications -- Policies, Services, and Semantic Integration

1. Ontology Translation and Semantic Integration
 - SWRL uses, ECOIN, financial services
2. End-to-End E-Contracting and Business Process Automation
 - supply chain, e-tailing, auctions, SweetDeal, Process Handbook
3. Business Policies including Trust
 - credit, health, RBAC, XACML, P3P, justifications
4. Semantic Web Services
 - SWSL tasks
5. Prospective Early Adopter areas, strategy, and market evolution
6. Windup and Discussion

Enhancing OWL Expressiveness with Rules to represent ontologies

- Use rules to express things that can't be represented in OWL
 - An uncle is the brother of a parent
 - 2 siblings have the same father
 - An InternationalFlight involves airports located in different countries
 - An Adult is a Person with age > 17

Ontology Translation Via Rules

- Use rules to represent mappings from data source to domain ontologies
 - Rules can be automatically or manually generated
 - Can support unit of measure conversion and structural transformation
- Example using SWRL
 - <http://www.daml.org/2004/05/swrl-translation/Overview.html>

Translation Coverage Matrix

- Standardized rule representation allows us to easily analyze the ontology translation coverage
- Table represents mappings from data ontology properties (rows) to domain ontology properties (columns)
 - Empty columns reflect information gaps
 - Columns > 1 reflect potential conflicts
 - Empty rows reflect unused information

Translation Coverage Matrix

		5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
	0	3	1	4	1	1	3	3	4	4	1	3	2	1	2	
V 6	V			V				V	V			V	V			
1 5	R							R	C		C	R				
1 1		R													R	
1 1							R	R								
1 1							R									
1 1				R												
1 1								R								
1 0																
1 3									R	C		C				
1 2									R				R			
1 1				R												
2 1	R															
2 1				R												
2 1								R								
2 1								R								
3 1				R												
3 1								R								
3 1								R		R	C		C			
4 4								R		R	C		C			

Legend

R = rule (rule without operationAtoms)
 C = calculated (rule with operationAtoms)
 V = rule with constant value
 S = owl:equivalentProperty

Example

domain:latitude = data:position.lat
 domain:valueInInches = data:valueInFeet * 12
 domain:priority = 5
 domain:id = data:idNumber

Matching across Datasets via Rules

- Use rules to match items between multiple data sets
- Example:
 - Match credit card transactions, expense report fields, and reimbursements
 - Imprecise dates
 - Aggregation
 - <http://www.daml.org/2001/06/expenses/>

Expansion via Rules

- Use rules to convert from
 - Compact representation easy to generate
 - \rightarrow Expanded representation easy to use
- Example
 - Represent subway lines with ordered lists of stations
 - Use rules to associate adjacent stations and stations with lines
 - <http://www.daml.org/2003/05/subway/>

Equational Ontological Conflicts in Financial Reporting

of customers = # of
end_customers + # of distributors

Gross Profit = Net Sales – Cost of
Goods

P/E Ratio = Price / Earnings(**last 4**
Qtr)

Price = Nominal Price + Shipping

of customers = # of end_customers
+ # of prospective customers

Gross Profit = Net Sales – Cost of
Goods – **Depreciation**

P/E Ratio = Price/ [Earnings(**last 3**
Qtr) +Earnings(**next** quarter)]

Price = Nominal Price + Shipping +
Tax

“ heterogeneity in the way data items are *calculated* from other
data items *in terms of definitional equations*”

EOC in Primark Databases

Key Concepts

Top 25 US Co. by Net Sales (Disclosure)

Rank	Company	Net Sales (000's)	Date
1	General Motors Corp	168,828,600	12/31/95
2	Ford Motor Co	137,137,000	12/31/95
3	Exxon Corp	121,804,000	12/31/95
4	Wal Mart Stores Inc	93,627,000	01/31/96
5	AT&T	79,609,000	12/31/95
6	Mobil Corp	73,413,000	12/31/95
7	International Business M	71,904,000	12/31/95
8	General Electric Co	70,028	
...

Top 25 International Co. by Net Sales (Worldscope)

Rank	Company	Net Sales (000's)	Date
1	Mitsubishi Corporation	165,848,468	03/31/96
2	General Motors Corp	163,861,100	12/31/95
...
8	Exxon Corp	107,893,000	12/31/95
...
16	International Business M	71,940,000	12/31/95
17	General Electric Co	69,948,000	12/31/95
20	Mobil Corp	64,767,000	12/31/95
...

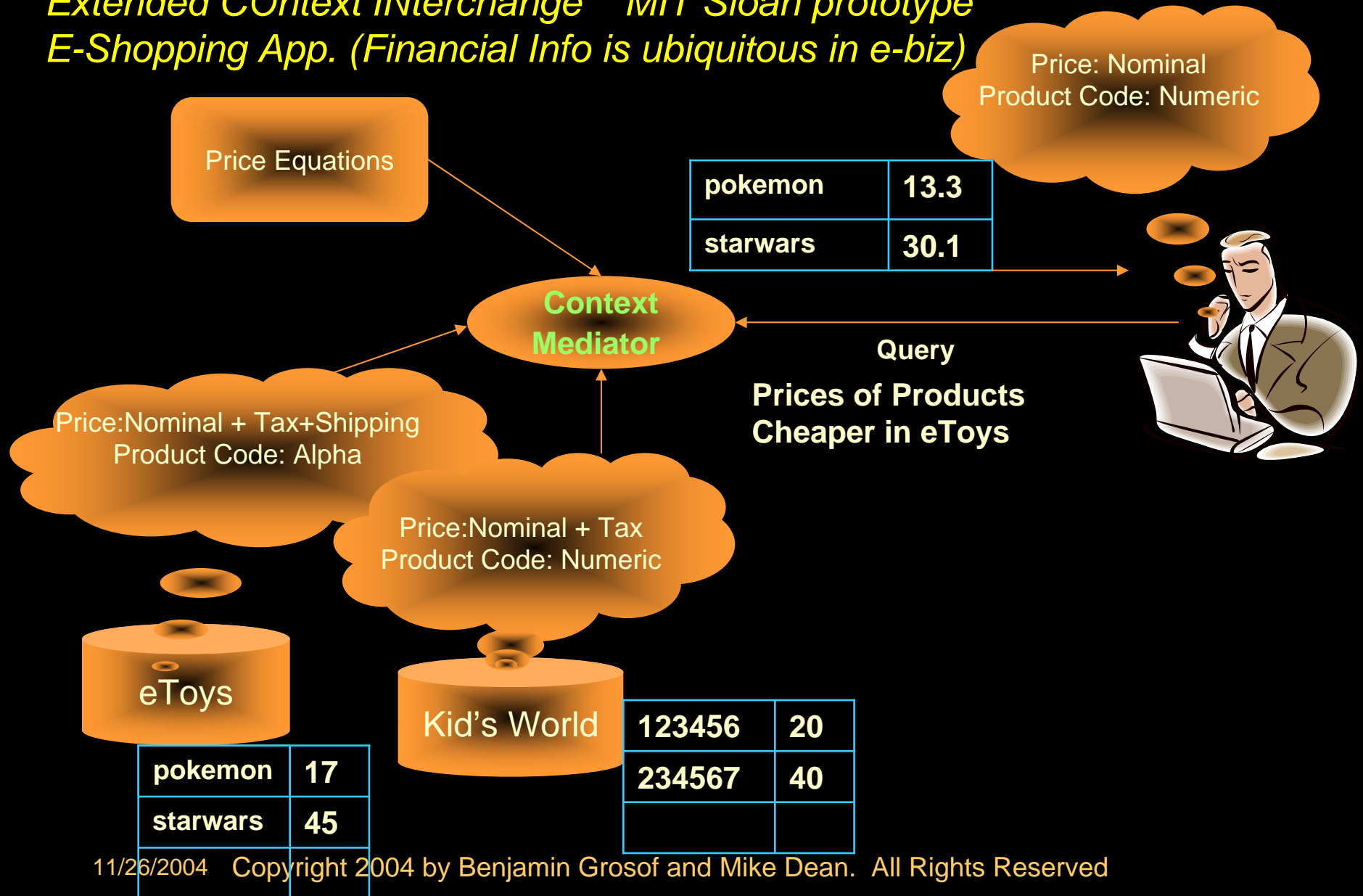
Primark was a company that owned:

- Disclosure
- Worldscope
- DataStream

Information services

Solution Approach: ECOIN

Extended COntext INterchange MIT Sloan prototype
E-Shopping App. (Financial Info is ubiquitous in e-biz)



Approach: ECOIN

Solution Methodology

- Context-based loosely-coupled integration
 - Extends the Context Interchange (COIN) framework developed at MIT
- Symbolic Equation Solving using Constraint Logic Programming
 - Integrates symbolic equation solving techniques with abductive logic programming
- *In-progress:* Utilizing RuleML and OWL in ECOIN
 1. OWL formulation of COIN ontologies: see [Bhansali, Madnick, & Grosf ISWC-2004 poster]



PRESS ROOM

EVENTS

CONTACT

- WHAT IS XBRL
- NEWS ABOUT XBRL
- THE SHOWCASE
- XBRL IN ACTION
- XBRL AND BUSINESS
- TECHNICAL INFORMATION
- EDUCATION AND TRAINING
- ABOUT THE ORGANISATION
- HOW TO JOIN
- MEMBERS' AREA**
- NEWS FOR MEMBERS
- TECHNICAL DEVELOPMENT
- WORKING GROUPS
- SUPPLY CHAINS

Welcome to XBRL International



Financial Reporting Goes Global - XBRL and IFRS Working Together

For more information, please visit the [Conference Website](#) [register today](#).

XBRL is a language for the electronic communication of business financial data which is set to revolutionise business reporting around the world. It provides major benefits in the preparation, analysis and communication of business information. It offers cost savings, greater efficiency and improved accuracy and reliability to all those involved in supplying or using financial data.

XBRL stands for eXtensible Business Reporting Language. It is one of a family of "XML" languages which is becoming a standard means of communicating information between businesses and on the internet.

XBRL is being developed by an international non-profit consortium of approximately 250 major companies, organisations and governmental agencies. **It is an open standard, free of licence fees.** It is already being put to practical use in a number of countries and implementations of XBRL are growing rapidly around the world.

This site provides information about the nature, uses and benefits of XBRL. It explains how individuals and companies can join the effort to move forward and make use of the language.

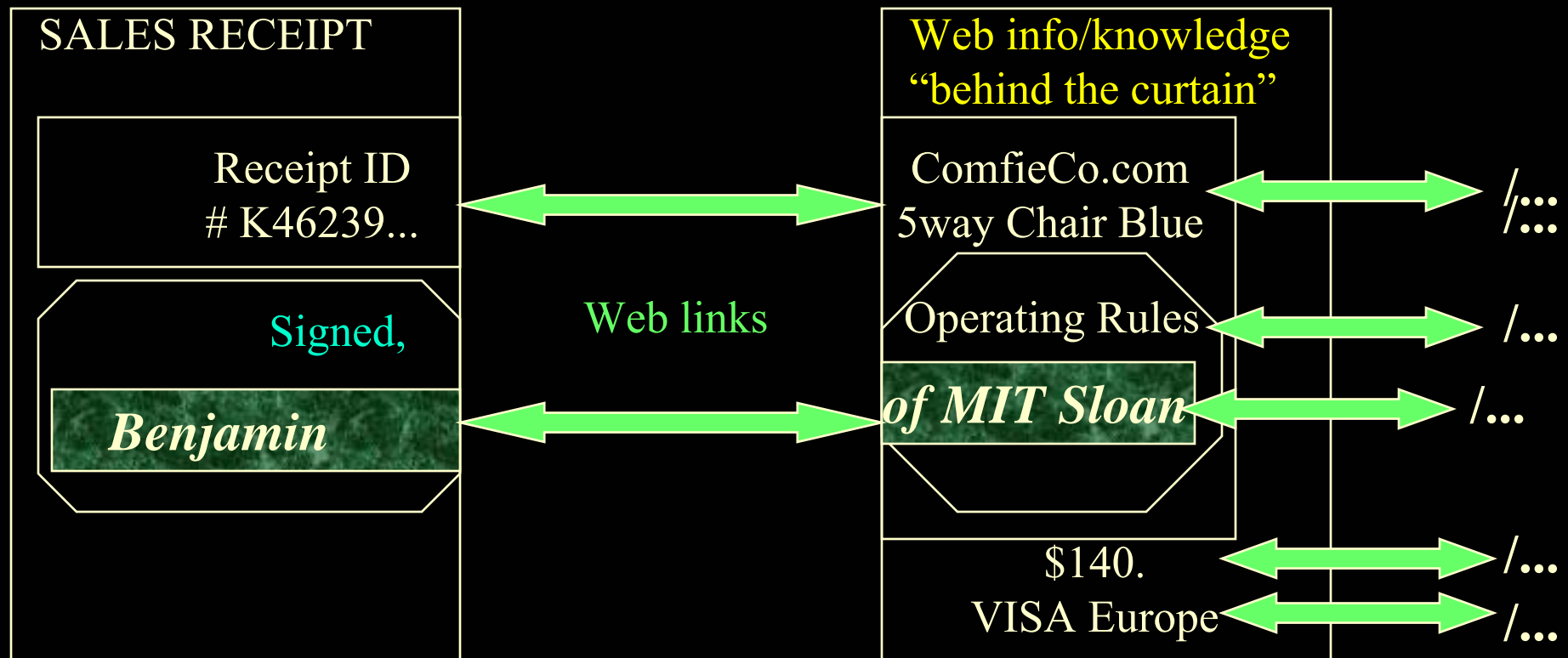
Outline of Part C.

C. Applications -- Policies, Services, and Semantic Integration

1. Ontology Translation and Semantic Integration
 - SWRL uses, ECOIN, financial services
2. End-to-End E-Contracting and Business Process Automation
 - supply chain, e-tailing, auctions, SweetDeal, Process Handbook
3. Business Policies including Trust
 - credit, health, RBAC, XACML, P3P, justifications
4. Semantic Web Services
 - SWSL tasks
5. Prospective Early Adopter areas, strategy, and market evolution
6. Windup and Discussion

Looks Simple To Start... then Gets Interestingly Precise

A Vision/Approach of what Web & Agents enable



End-to-End E-Contracting Tasks

- Discovery, advertising, matchmaking
 - Search, sourcing, qualification/credit checking
- Negotiation, bargaining, auctions, selection, forming agreements, committing
 - Hypothetical reasoning, what-if'ing, valuation
- Performance/execution of agreement
 - Delivery, payment, shipping, receiving, notification
- Problem Resolution, Monitoring
 - Exception handling

Approach:

Rule-based Contracts for E-commerce

- Rules as way to specify (part of) business processes, policies, products: as (part of) contract terms.
- Complete or partial contract.
 - As **default rules**. **Update**, e.g., in negotiation.
- Rules provide high level of conceptual abstraction.
 - **easier for non-programmers** to understand, specify, **dynamically modify & merge**. E.g.,
 - by multiple authors, cross-enterprise, cross-application.
- Executable. Integrate with other rule-based business processes.

SweetDeal Approach

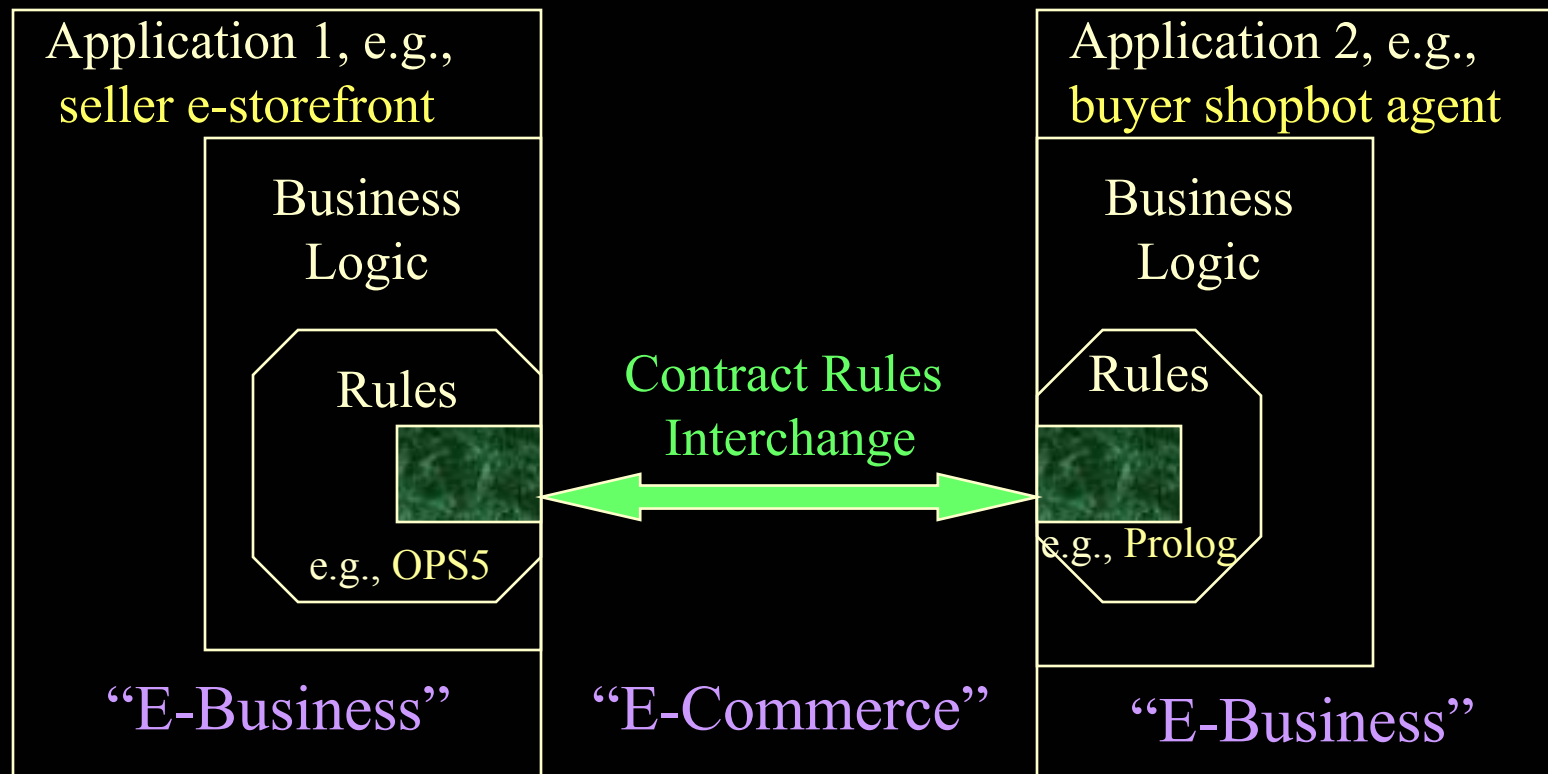
[Grosf, Labrou, & Chan EC-99; Wellman, Reeves, & Grosf Computational Intelligence 2002; Grosf & Poon Intl. J. of Electronic Commerce 2004]

- SWEET = Semantic Web Enabling Technology
 - software components, theory, approach
 - pilot application scenarios, incl. **contracting** (SweetDeal)
- Uses/contributes *emerging standards* for XML and knowledge representation:
 - RuleML semantic web rules
 - OWL ontologies (W3C)
- Uses *repositories* of business processes and contracts
 - MIT Process Handbook (Sloan IT)
 - legal/regulatory sources: law firms, ABA, CommonAccord, ... **Suggestions welcome!!**

What Can Be Done with the Rules in contracting, & negotiation, based on our SweetDeal approach to rule representation

- **Communicate:** with deep shared semantics
 - via RuleML, inter-operable with same sanctioned inferences
 - \Leftrightarrow heterogeneous rule/DB systems / rule-based applications (“agents”)
- **Execute** contract provisions:
 - infer; ebiz actions; authorize; ...
- **Modify** easily: contingent provisions
 - default rules; modularity; exceptions, overriding
- **Reason** about the contract/proposal
 - hypotheticals, test, evaluate; tractably
 - *(also need “solo” decision making/support by each agent)*

Contract Rules across Applications / Enterprises

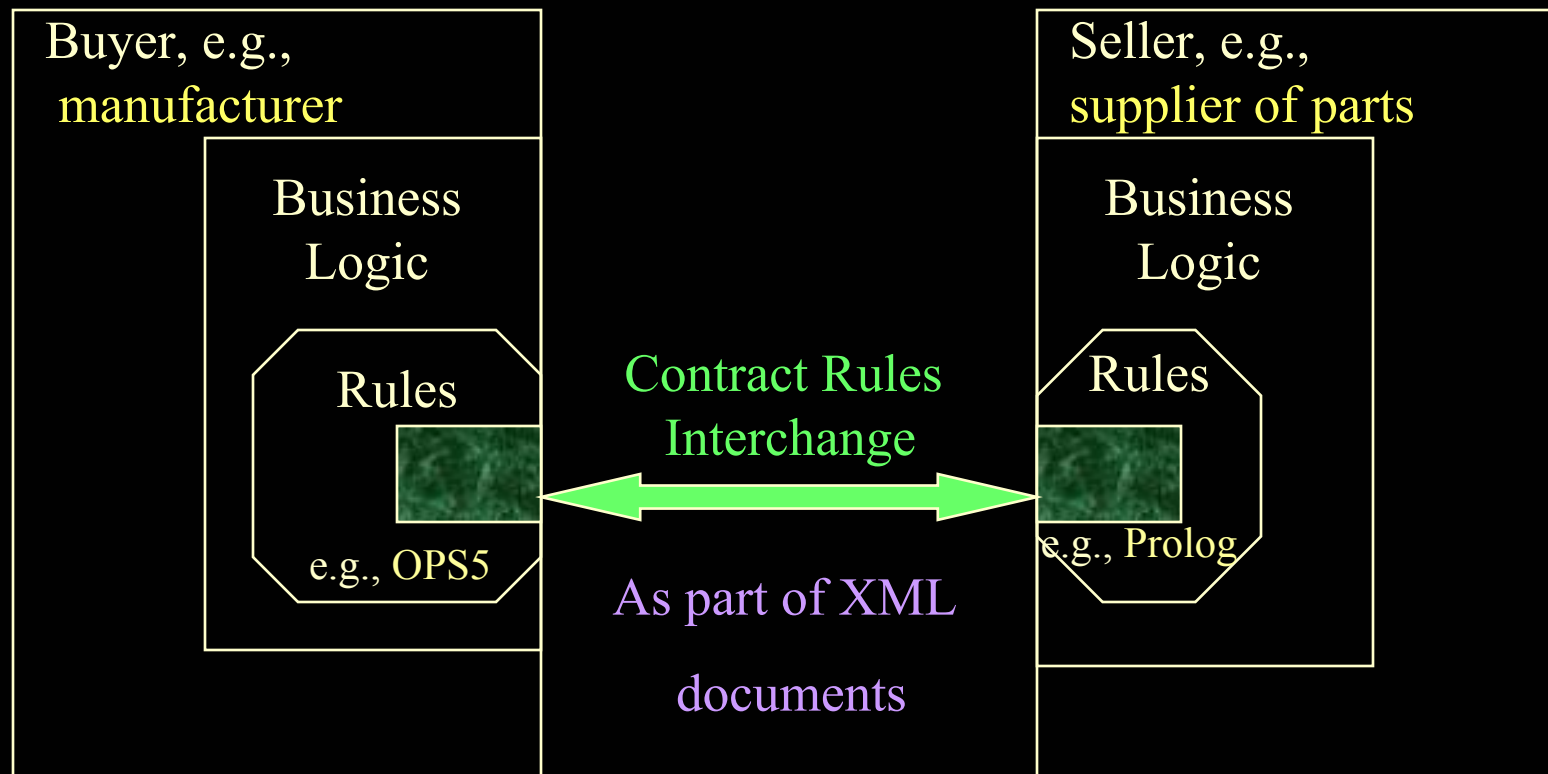


Contracting parties integrate e-businesses via shared rules.

Examples of Contract Provisions Well-Represented by Rules in Automated Deal Making

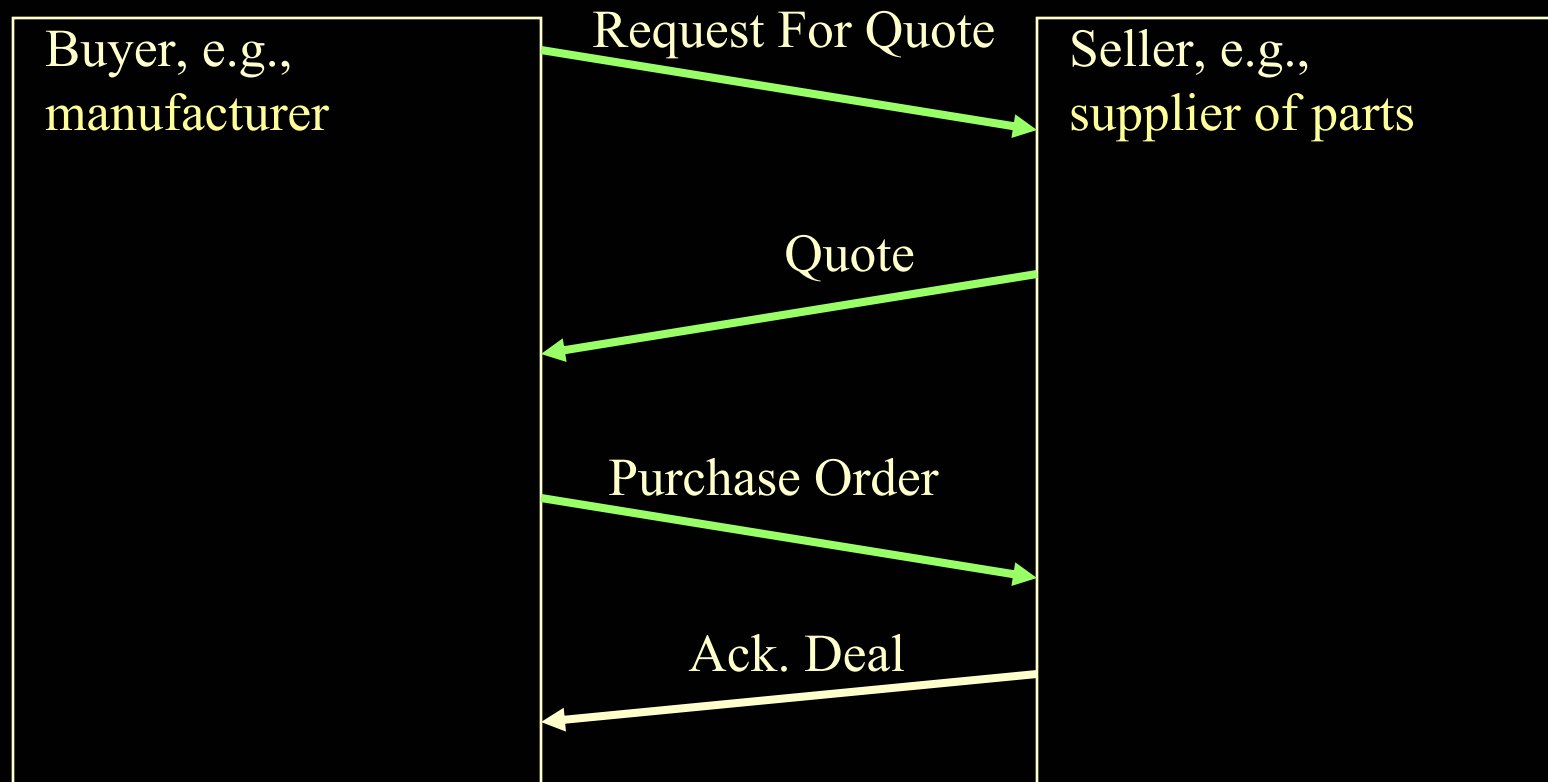
- **Product descriptions**
 - Product catalogs: properties, conditional on other properties.
- **Pricing dependent upon:** delivery-date, quantity, group memberships, umbrella contract provisions
- **Terms & conditions:** refund/cancellation timelines/deposits, lateness/quality penalties, ordering lead time, shipping, creditworthiness, biz-partner qualification, service provisions
- **Trust**
 - Creditworthiness, authorization, required signatures
- *Buyer Requirements (RFQ, RFP) wrt the above*
- *Seller Capabilities (Sourcing, Qualification) wrt the above*

Contract Rules during Negotiation

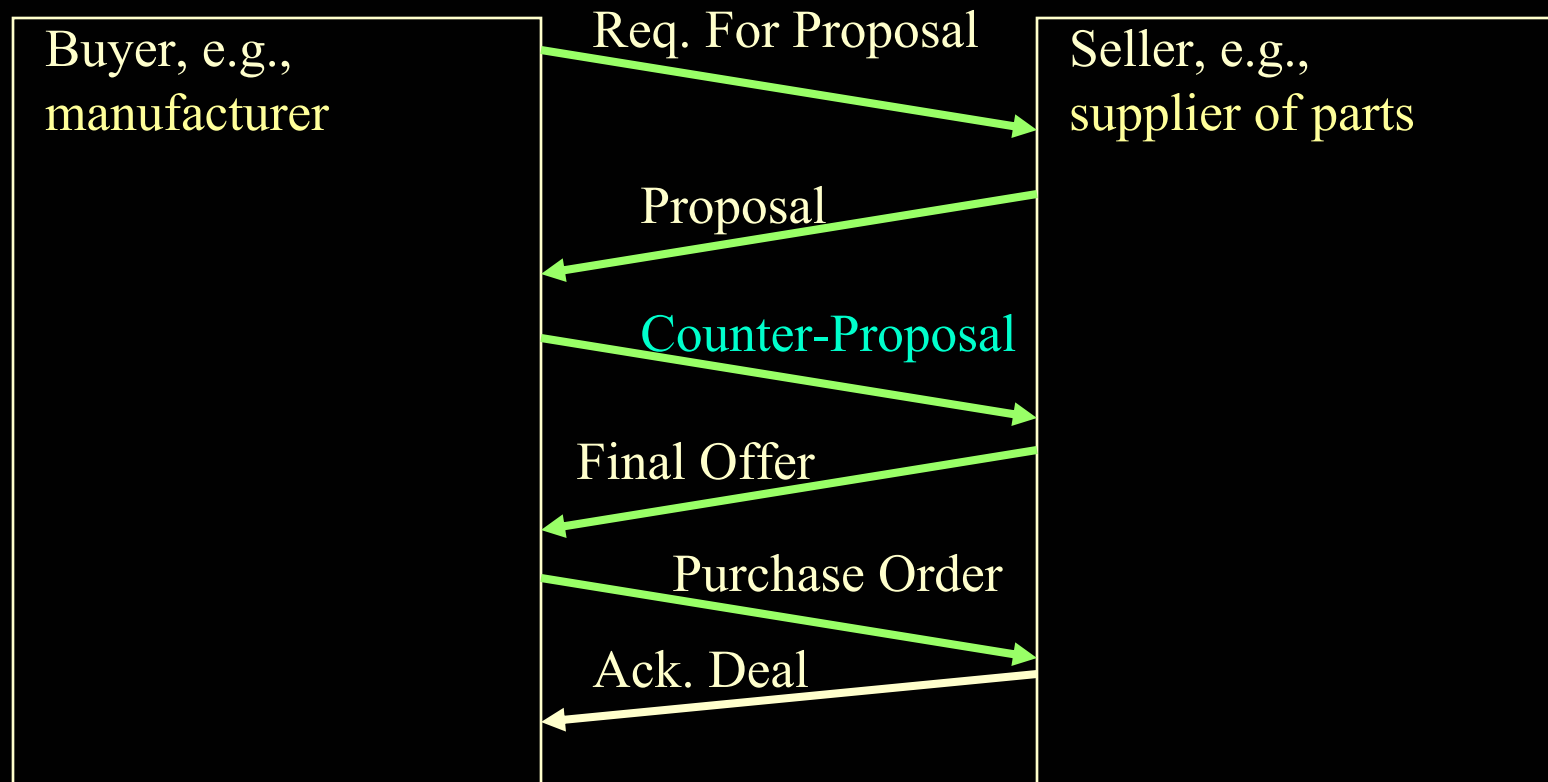


Contracting parties NEGOTIATE via shared rules.

Exchange of Rules Content during Negotiation: example



Exchange of Rules Content during Negotiation: example



Negotiation Example XML Document: Proposal from supplierCo to manufCo

- `<negotiation_message>`
- `<message_header>`
- `<proposal/>`
- `<from> supplierCo </from>`
- `<to> ManufCo </to>`
- `</message_header>`
- `<rules_content>`
- `...[see next slide]`
- `</rules_content>`
- `...`
- `</negotiation_message>`

- Example of similar message document format:

- FIPA Agent Communication Markup Language (draft industry standard).

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) .
- \perp ← price(per_unit, ?PO, ?X) ∧ price(per_unit, ?PO, ?Y) GIVEN (?X ≠ ?Y).
- ...

Negotiation Ex. Doc. Rules:

Counter-Proposal from *manufCo* to *supplierCo*

- ...
- $\langle \text{usualPrice} \rangle$ price(per_unit, ?PO, \$60) \leftarrow ...
- $\langle \text{volumeDiscount} \rangle$ price(per_unit, ?PO, \$51) \leftarrow
- purchaseOrder(?PO, supplierCo, ?AnyBuyer) \wedge
- quantity_ordered(?PO, ?Q) \wedge (?Q \geq 5) \wedge (?Q \leq 1000) \wedge
- shipping_date(?PO, ?D) \wedge (?D \geq 28Apr00) \wedge (?D \leq 12May00) .
- overrides(volumeDiscount , usualPrice) .
- $\perp \leftarrow$ price(per_unit, ?PO, ?X) \wedge price(per_unit, ?PO, ?Y) GIVEN (?X \neq ?Y).
- $\langle \text{aSpecialDeal} \rangle$ price(per_unit, ?PO, \$48) \leftarrow
- purchaseOrder(?PO, supplierCo, **manufCo**) \wedge
- quantity_ordered(?PO, ?Q) \wedge (?Q \geq **400**) \wedge (?Q \leq 1000) \wedge
- shipping_date(?PO, ?D) \wedge (?D \geq **02May00**) \wedge (?D \leq 12May00) .
- overrides(aSpecialDeal, volumeDiscount) .
- overrides(aSpecialDeal , usualPrice) .
- ...

**Simply
added
rules!**

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>`

Negotiation Example --

XML Encoding of Rules in RuleML, Continued

- `<_body>`
- `<andb>`
- `<fclit>`
- `<_opr><rel>purchaseOrder</rel></_opr>`
- `<var>PO</var>`
- `<ind>supplierCo</ind>`
- `<var>AnyBuyer</var>`
- `</fclit>`
- `<fclit>`
- `...`
- `</fclit>`
- `...`
- `</andb>`
- `</_body>`

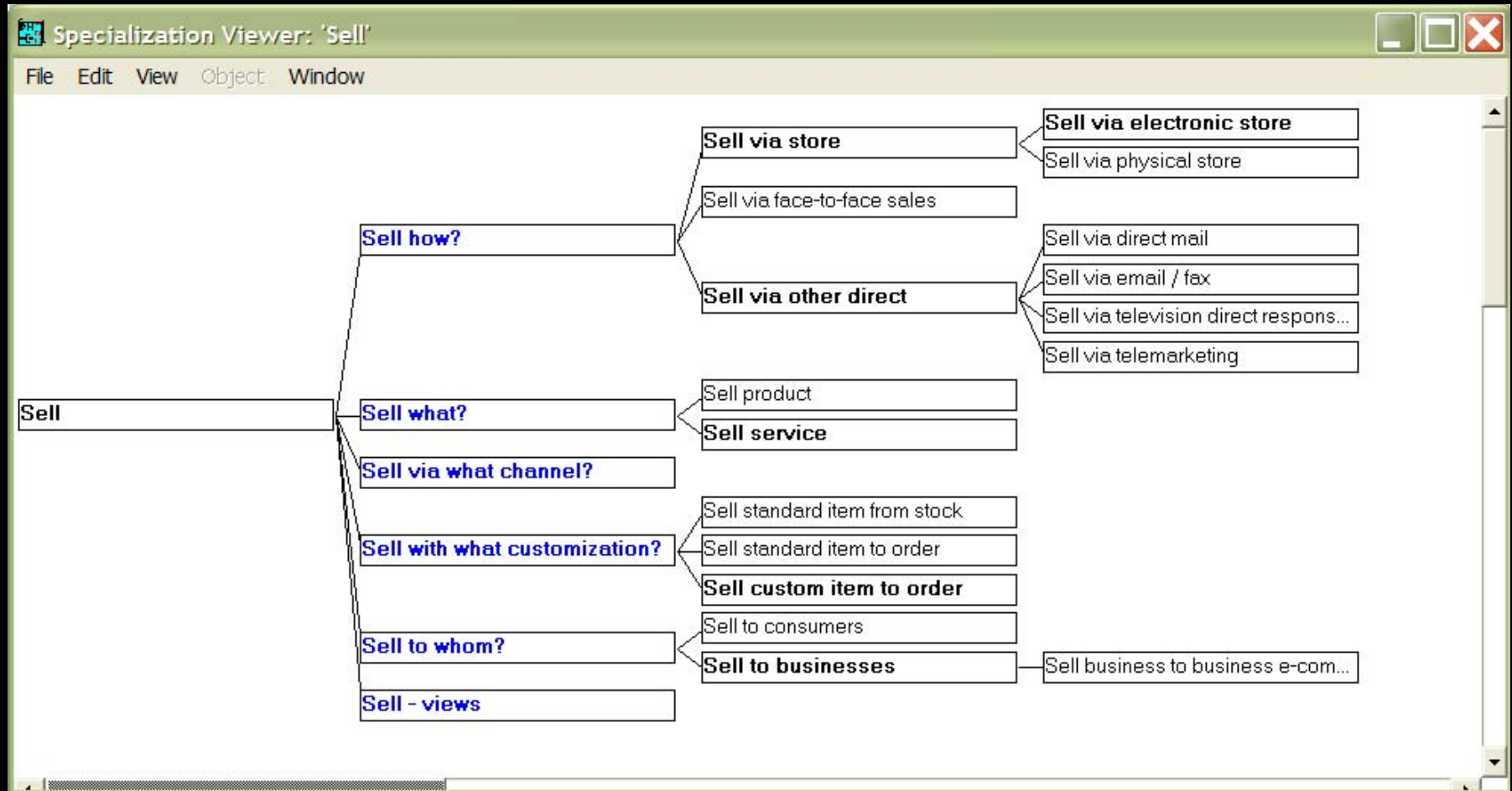
URI Ontological Reference Approach Example, in RuleML

```
payment(?R,base,?Payment) <-  
http://xmlcontracting.org/sd.owl#result(co123,?R) AND  
price(co123,?P) AND quantity(co123,?Q) AND  
multiply(?P,?Q,?Payment) ;
```

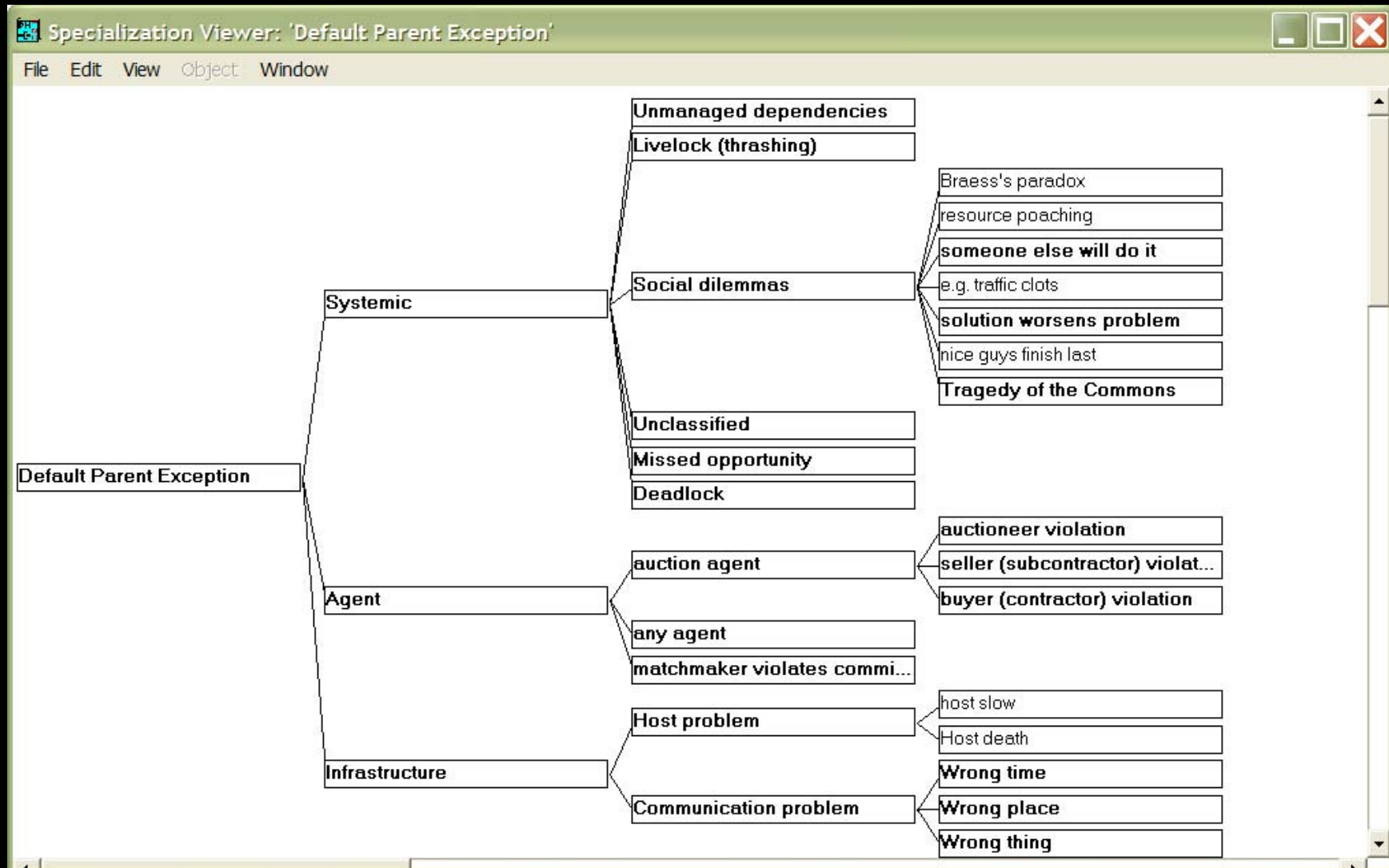
SCLP TextFile Format for RuleML

```
<imp>  
  <_head> <atom>  
    <_opr><rel>payment</_opr></rel>    <tup>  
      <var>R</var> <ind>base</ind> <var>Payment</var>  
    </tup></atom> </_head>  
  <_body>  
    <andb>  
      <atom> <_opr>  
        <rel href= "http://xmlcontracting.org/sd.owl#result" />  
      </_opr> <tup>  
        <ind>co123</ind> <var>Cust</var>  
      </tup> </atom>  
    .. </andb> </_body> </imp>
```

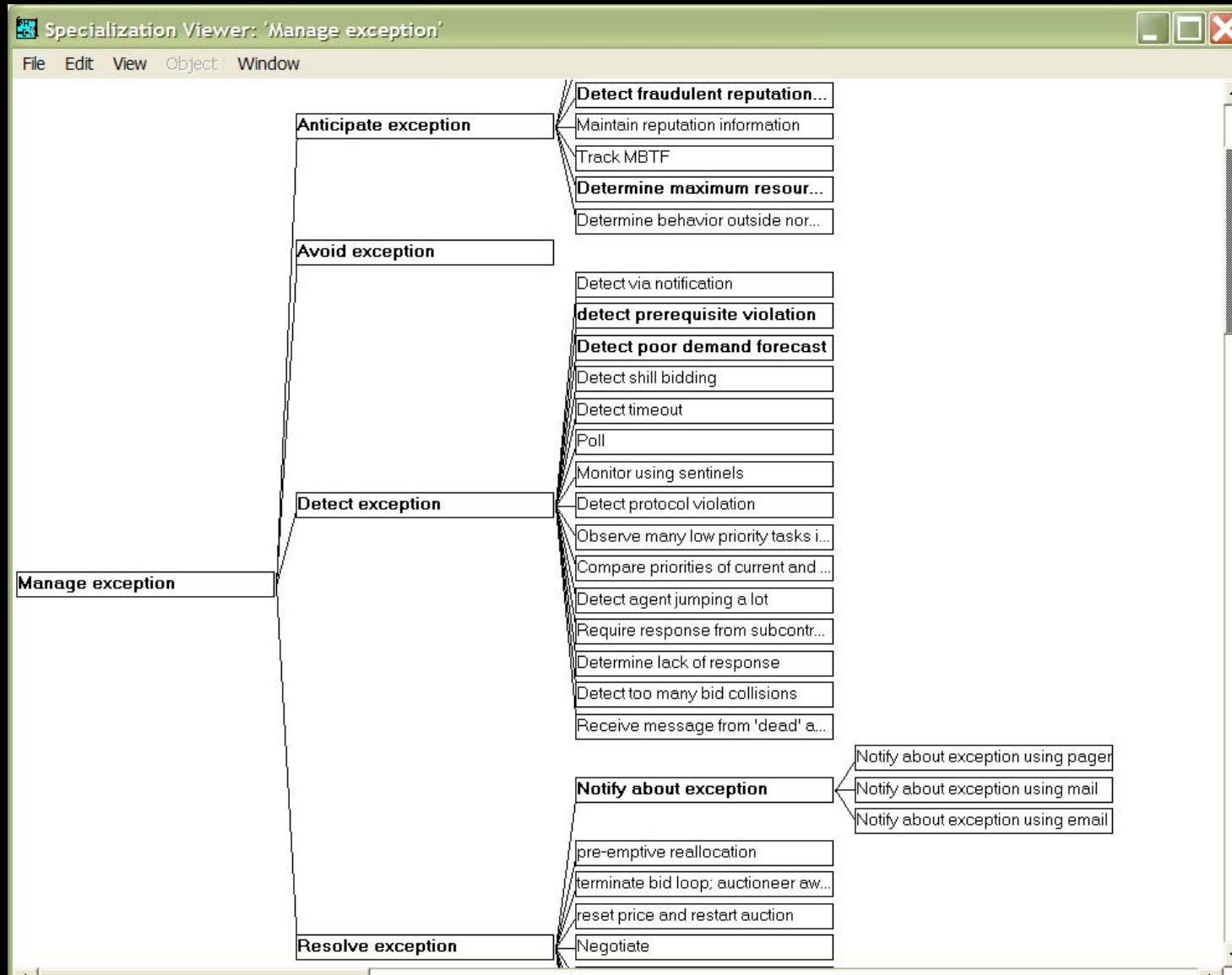
Some Specializations of “Sell” in the MIT Process Handbook (PH)



Some Exceptions in the MIT Process Handbook



Some exception handlers in the MIT Process Handbook



Example Contract Proposal with Rule-based Exception Provisions

- Buyer adds rule modules to the contract proposal to specify:
 - 1. **detection** of an exception
 - **LateDelivery** as a potential exception of the contract's process
 - **detectLateDelivery** as exception handler: recognize occurrence
 - 2. **avoidance** of an exception (and perhaps also resolution of the exception)
 - **lateDeliveryPenalty** as exception handler: penalize per day
- Rule module = a nameable ruleset → a subset of overall rulebase
 - can be included directly and/or imported via link; nestable
 - similar to legal contracts' "incorporation by reference"
 - an extension to RuleML; in spirit of "Webizing"

Example Contract Counter-Proposal with Rule-based Exception Provisions

- Seller modifies the draft contract (it's a *negotiation!*)
- Simply adds* another rule module to specify:
 - **lateDeliveryRiskPayment** as exception handler
 - lump-sum in advance, based on average lateness
 - instead of proportional to actual lateness
 - higher-priority for that module than for the previous proposal, e.g., higher than lateDeliveryPenalty's rule module
- Courteous LP's **prioritized conflict handling** feature is used
- ***NO change** to previous proposal's rules needed!
 - similar to legal contracts' accumulation of provisions

EECOMS Supply Chain Early Commercial Implementation & Piloting

- EECOMS agile supply chain collaboration industry consortium including Boeing, Baan, TRW, Vitria, IBM, universities, small companies
 - \$29Million 1998-2000; 50% funded by NIST ATP
 - application piloted IBM CommonRules and early approaches which led to SweetDeal, RuleML, and SweetRules
 - contracting & negotiation; authorization & trust

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.`

Outline of Part C.

C. Applications -- Policies, Services, and Semantic Integration

1. Ontology Translation and Semantic Integration
 - SWRL uses, ECOIN, financial services
2. End-to-End E-Contracting and Business Process Automation
 - supply chain, e-tailing, auctions, SweetDeal, Process Handbook
3. Business Policies including Trust
 - credit, health, RBAC, XACML, P3P, justifications
4. Semantic Web Services
 - SWSL tasks
5. Prospective Early Adopter areas, strategy, and market evolution
6. Windup and Discussion

Challenge: Capturing Semantics around Policies

- Deep challenge is to capture the semantics of data and processes, so that can:
 - Represent, monitor, and enforce policies – e.g., trust and contracts
 - Map between definitions of policy entities, e.g., in financial reporting
 - Integrate policy-relevant information powerfully

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

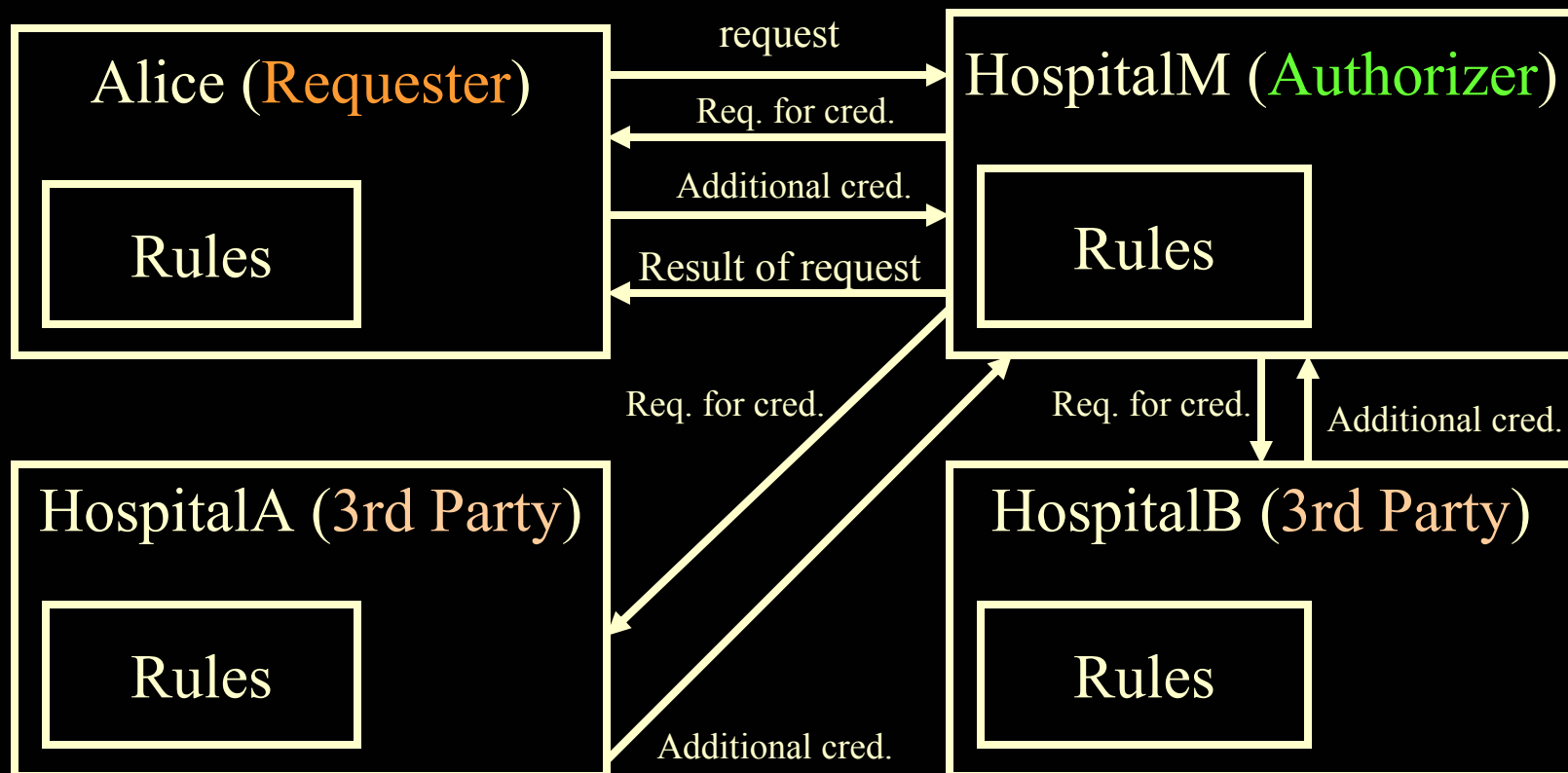
Delegation Logic (D1LP) Example: accessing medical records

[N. Li, B. Grosf, J. Feigenbaum ACM TISSEC 2003]

- **Problem:** Hospital HM to decide: requester Alice authorized for patient Peter?
- **Policies:** HM will authorize only the patient's physician. HM trusts any hospital it knows to certify the physician relationship. Two hospitals together can vouch for a 3rd hospital.
 - HM says `authorized(?X, read(medRec(?Y)))` if HM says `inRole(?X, physic(?Y))`.
 - HM delegates `inRole(?X, physic(?Y))^1` to `threshold(1, ?Z, HM says inRole(?Z, hosp))`.
 - HM delegates `inRole(?H, hosp)^1` to `threshold(2 , ?Z, HM says inRole(?Z, hosp))`.
- **Facts:** HC certifies Alice is Peter's physician. HM knows two hospitals HA and HB. HA and HB each certify HC as a hospital.
 - HC says `inRole(Alice, physic(Peter))`. HA says `inRole(Joe, physic(Sue))`.
 - HM says `inRole(HA, hosp)`. HM says `inRole(HB, hosp)`.
 - HA says `inRole(HC, hosp)`. HB says `inRole(HC, hosp)`.
- **Conclusion:** HM says `authorized(Alice, read(medRec(Peter)))`. *Joe NOT authorized.*

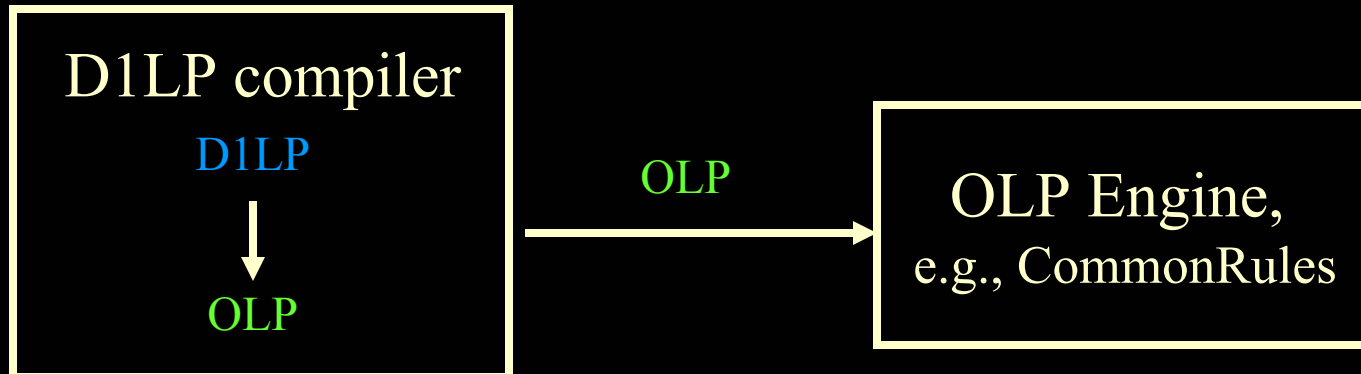
Slide also by Ninghui Li and Joan Feigenbaum

Example Scenario Information Flow



D1LP Compiler (Architecture)

- Java Implementation (part of CommonRules research prototype)



- Prolog Implementation:
 - The compiler is written in Prolog
 - The compiler dynamically asserts OLP rules into Prolog engine
 - Uses Prolog engine to do inference

Trust Policies and Compliance in US Financial Industry Today

- Ubiquitous high-stakes Regulatory Compliance requirements
 - Sarbanes Oxley, SEC (also in medical domain: HIPAA), etc.
- Internal company policies about access, confidentiality, transactions
 - For security, risk management, business processes, governance
- Complexities guiding who can do what on certain business data
- Often implemented using rule techniques
- Often misunderstood or poorly implemented leading to vulnerabilities
- Typically embedded redundantly in legacy silo applications, requiring high maintenance
- Policy/Rule engines lack interoperability

Example Financial Authorization Rules

Classification	Application	Rule
Merchant	Purchase Approval	If credit card has fraud reported on it, or is over limit, do not approve.
Mutual Funds	Rep trading	<i>Blue Sky</i> : State restrictions for rep's customers.
Mortgage Company	Credit Application	TRW upon receiving credit application must have a way of securely identifying the request.
Brokerage	Margin trading	Must compute current balances and margin rules before allowing trade.
Insurance	File Claims	Policy States and Policy type must match for claims to be processed.
Bank	Online Banking	User can look at own account.
All	House holding	For purposes of silo (e.g., statements or discounts), aggregate accounts of all family members.

Example 1 – Credit Card Verification System

- Typical for eCommerce websites accepting credit cards – Visa, MC, Discover, Amex
- Rules for transaction authorization
 - Bank performs account limit, expiration, address and card code verification
 - A fraud alert service may flag a card
 - Service provider may blacklist customer
- Overrides, e.g., alert service > bank rules

Example II – Brokerage Access Control

- Need protection of customer accounts of retail (own) and many client correspondents from unauthorized access by traders (reps)
- Many Complex Rules for access control
 - Retail reps can look at any retail account but not correspondent accounts
 - A correspondent user may look at accounts for their organization but...
 - Only from those branches over which rep's branch has fiduciary responsibility
 - For certain branches, customer accounts are explicitly owned by certain reps and cannot be divulged even to his partner!
- More rules, with several overrides

CommonRules Implementation for Credit Card Verification Example

Sample Rule Listing

```
<bankResp>
  if checkTran(?Requester)
  then
    transactionValid(self,?Requester);
<cardRules2>
  if    checkCardDet(?Requester, ?accountLimit, ?exp_flag, ?cardholderAddr,
    ?cardholderCVC) and
    checkTranDet(?Requester, ?tranAddr, ?tranCVC) and
    notEquals(?tranCVC, ?cardholderCVC)
  then
    CNEG transactionValid(self,?Requester);
...
overrides(cardRules2, bankResp);
checkTran(Joe);
checkCardDet(Joe, 50, "false", 13, 702);
checkTranDet(Joe, 13, 702);
cardGood(Fraudscreen.net,Joe,good);
customerRating(Amazon.com, Joe, good);
```

**CommonRules translates
straightforwardly ↔ RuleML.**

**We show its human-oriented
syntax as a presentation syntax for
RuleML.**

Runtime Results for Credit Card Verification

Sample Output

SCLP Engine: Adorned Derived Conclusions:

```
CNEG transactionValid_c_3(self, Mary);
transactionValid_c_2(self, Joe);
transactionValid_c_2(self, Mary);
transactionValid_r_2(self, Mary);
transactionValid_u(self, Joe);
CNEG transactionValid_u(self, Mary);
```

```
transactionValid(self, Joe);
CNEG transactionValid(self, Mary);
```

Adorned conclusions represent intermediate phases of prioritized conflict handling in Courteous Logic Programs

CNEG = limited classical negation
(which is permitted in Courteous LP)
CNEG p means *p* is (believed to be) false

Self = the agent making the authorization decision, i.e., the viewpoint of this local rulebase.
(This is as usual in trust management.)



PRESS ROOM

EVENTS

CONTACT

WHAT IS XBRL

NEWS ABOUT XBRL

THE SHOWCASE

XBRL IN ACTION

XBRL AND BUSINESS

TECHNICAL INFORMATION

EDUCATION AND TRAINING

ABOUT THE ORGANISATION

HOW TO JOIN

MEMBERS' AREA

NEWS FOR MEMBERS

TECHNICAL DEVELOPMENT

WORKING GROUPS

SUPPLY CHAINS

Welcome to XBRL International



Financial Reporting Goes Global - XBRL and IFRS Working Together

For more information, please visit the [Conference Website](#) [register today](#).

XBRL is a language for the electronic communication of business financial data which is set to revolutionise business reporting around the world. It provides major benefits in the preparation, analysis and communication of business information. It offers cost savings, greater efficiency and improved accuracy and reliability to all those involved in supplying or using financial data.

XBRL stands for eXtensible Business Reporting Language. It is one of a family of "XML" languages which is becoming a standard means of communicating information between businesses and on the internet.

XBRL is being developed by an international non-profit consortium of approximately 250 major companies, organisations and governmental agencies. **It is an open standard, free of licence fees.** It is already being put to practical use in a number of countries and implementations of XBRL are growing rapidly around the world.

This site provides information about the nature, uses and benefits of XBRL. It explains how individuals and companies can join the effort to move forward and make use of the language.

More Strategic Opportunities in Compliance

- XBRL (eXtensible Business Reporting Language):
 - SWS rules + ontologies can reduce degree of industry consensus required to enable interoperability
 - Difficult to get agreement on single definition of “earnings”; easier to agree on “long-term capital gains realized from sale of real estate assets”.
 - Translate between different use contexts’ ontologies
- SEC and other regulatory agencies:
 - They can accelerate compliance
 - via providing automated SWS specifications of regulations and reporting forms (+ the instructions)
 - e.g., RuleML regulatory rulebases accessible via Web Services interfaces

eXtensible Access Control Markup Language (XACML)

- Oasis XACML is leading technical standard for access control policies in XML
 - Access to XML info
 - Policies in XML
- Uses a rule-based approach
 - Including for prioritized combination of policies
- Status: Emerging
- **Needs a formal semantics -- and a more principled and standardized approach to rules KR, generally.**
 - **Research opportunity!**

Platform for Privacy Preferences (P3P)

- W3C P3P is leading technical standard for privacy policies representation and enforcement
- Client privacy policies specified in a simple rule language (APPEL, part of P3P)
- Has not achieved great usage yet
 - Microsoft dominance of browsers a strategic issue
- Needs a formal semantics -- and a more principled and standardized approach to rules KR, generally.
 - Research opportunity!

Web Services Trust Policy Management

- Web Services (WS) area is evolving quickly
- Emerging hot area: WS policy management, including for security/trust -- which includes privacy
 - Defined as next-phase agenda in standards efforts, major vendor white papers/proposals (e.g., Microsoft, IBM)
 - Semantic Web Services research in this is growing, e.g., DAML-Security effort, Rei, SWSL
- **Research opportunity!**

Other Aspects and Approaches: Web Trust and Policies

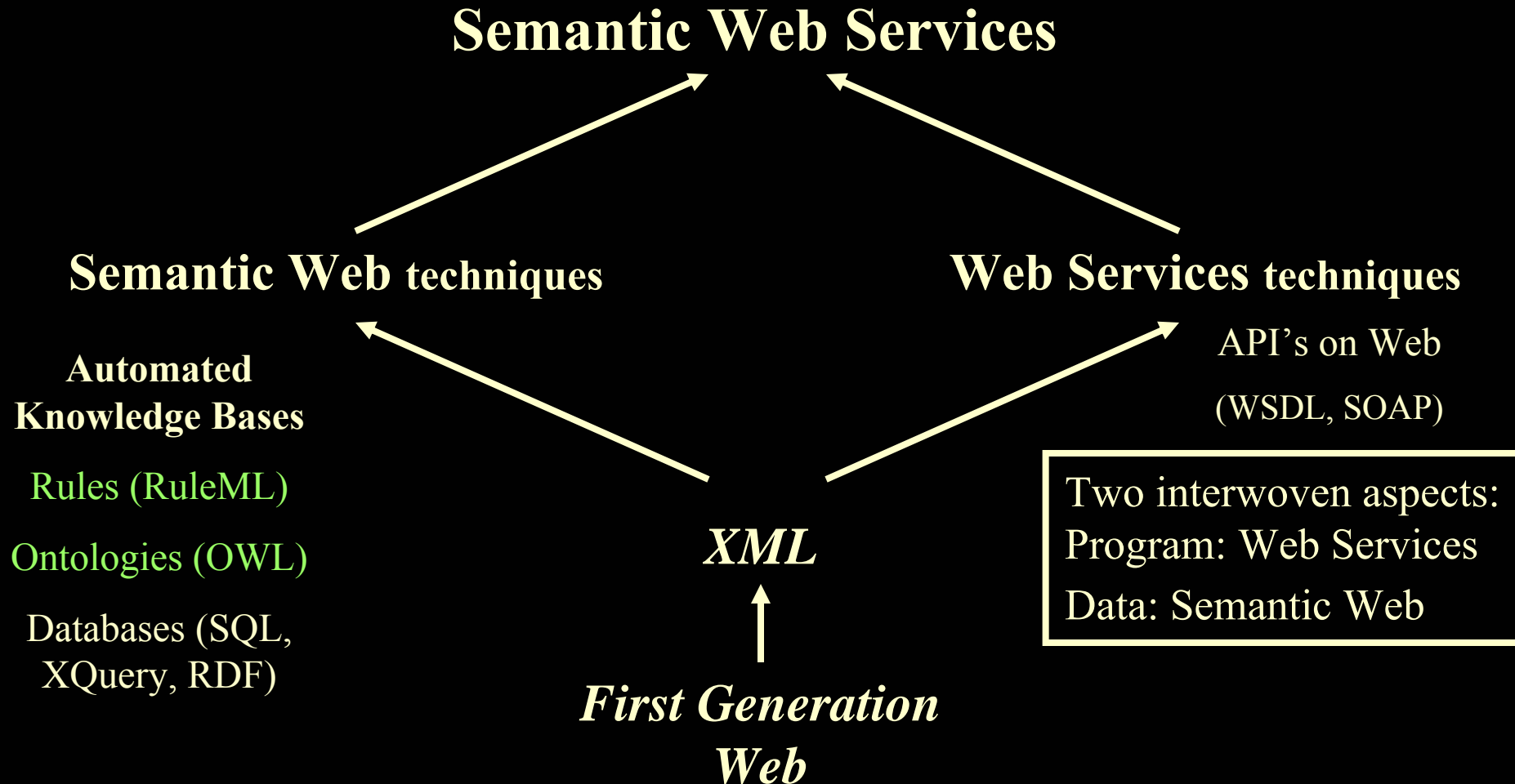
- Rei rule-based policy language [L. Kagal *et al*]
 - Builds upon SCLP, OWL, Delegation Logic approach
- DAML-Security effort [Denker *et al*]
- PeerTrust rule-based trust negotiation [Nejdl *et al*]
 - Builds upon OLP, Delegation Logic approach; protocols
- Justifications and proofs on the Semantic Web:
 - InferenceWeb approach [D. McGuinness *et al*]

Outline of Part C.

C. Applications -- Policies, Services, and Semantic Integration

1. Ontology Translation and Semantic Integration
 - SWRL uses, ECOIN, financial services
2. End-to-End E-Contracting and Business Process Automation
 - supply chain, e-tailing, auctions, SweetDeal, Process Handbook
3. Business Policies including Trust
 - credit, health, RBAC, XACML, P3P, justifications
4. Semantic Web Services
 - SWSL tasks
5. Prospective Early Adopter areas, strategy, and market evolution
6. Windup and Discussion

Next Generation Web



Semantic Web Services

- Convergence of Semantic Web and Web Services
- Consensus definition and conceptualization still forming
- Semantic (Web Services):
 - Knowledge-based service descriptions, deals
 - Discovery/search, invocation, negotiation, selection, composition, execution, monitoring, verification
 - Advantage: **reuse** of knowledge across app's, these tasks
 - Integrated knowledge
- (Semantic Web) Services: e.g., infrastructural
 - Knowledge/info/DB integration
 - Inferencing and translation

Monitoring

- task of enforcing policies (e.g., for trust or contracts)
- policies to handle exceptions & non-compliance (compare results to promises)

Rules in Semantic Web Services

- *We discussed earlier:*
 - Vision of rules in e-business
 - Concept and advantages of rule-based SWS
 - at high level
 - Various applications
- SWS provides a framework
 - *For perspective to view applications*
 - *A target for impact of applications*

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.

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.

Rule-based Semantic Web Services

- Rules often good to executably specify service process models
 - e.g., business process automation using procedural attachments to perform side-effectful/state-changing actions ("effectors" triggered by drawing of conclusions)
 - e.g., rules obtain info via procedural attachments ("sensors" test rule conditions)
 - e.g., rules for knowledge translation or inferencing
 - e.g., info services exposing relational DBs
- Infrastructural: rule system functionality as services:
 - e.g., inferencing, translation

Rules in Semantic Web Services: SWSL Strategic Requirements Analysis I

[Grosf, Kifer, Martin et al – SWSI Language Committee May 2004]

- The opportunity for near-term impact of SWS is mostly: ...
- Use of LP Rules in: the “SCAMP” group of tasks:
 - **SCAMP** = Security, Contracts, Adsvertising, access, authorization, mappings/mediation for semantic interoperability, Monitoring, privacy, and Policies

Rules in Semantic Web Services:

SWSL Strategic Requirements Analysis II

*[Grosz, Kifer, Martin et al – SWSL Language Committee
May 2004]*

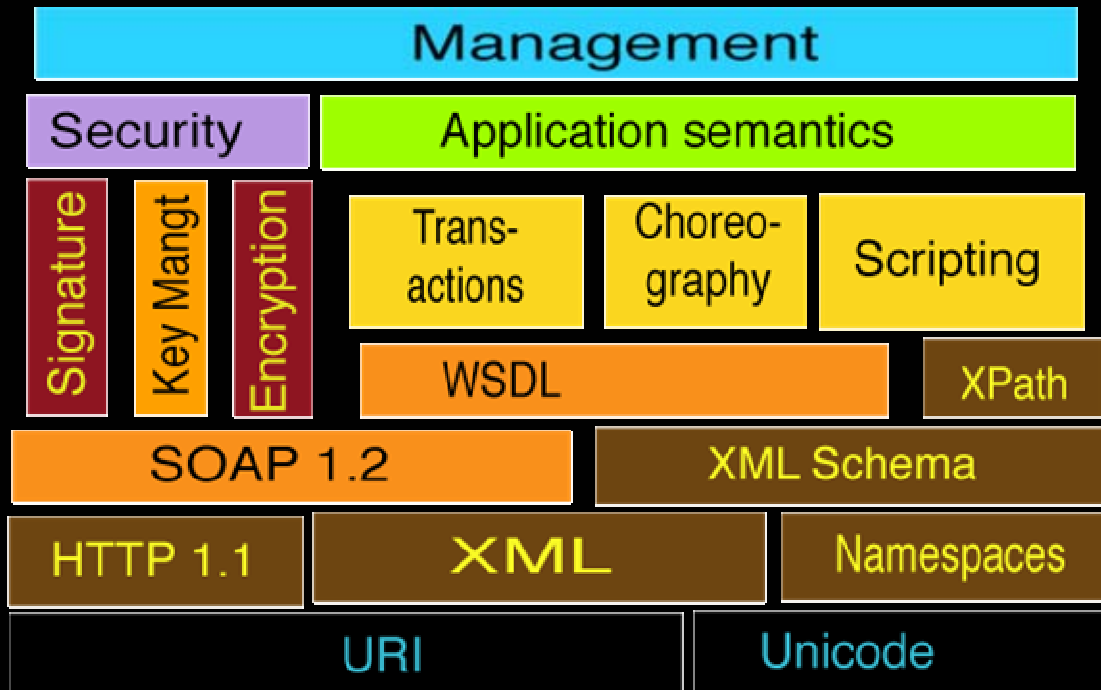
- Especially:
 - Policies in Security/Trust, Contracts, simple Advertising, Monitoring
 - Combine rules + ontologies in LP
 - Extend OWL-S profile
 - Note that wrt task analysis:
 - Advertising uses partial contracts. Advertising is part of a cluster with discovery and matchmaking.
 - Monitoring revolves around policies and contracts

Rules in Semantic Web Services: SWSL Strategic Requirements Analysis III

[Grosz, Kifer, Martin et al – SWSL Language Committee May 2004]

- Starting points technically:
 - (nonmon) LP RuleML (and Horn SWRL)
 - + OWL-S profile
 - + DLP OWL core service ontologies
 - + DLP OWL domain ontologies
 - Then later add: + additional PSL service ontologies
+ HiLog/F-Logic extensions to LP RuleML

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

"Wire" Protocols

Service Description

W3C WS Choreography Group
 BPEL4WS (Microsoft, IBM, BEA)
 WSCL (HP)BPML (Most but Microsoft)
 WSCI (Sun, BEA, Yahoo, ...)
 XLANG (Microsoft), WSFL (IBM), ...

SOAP Blocks

SOAP/XMLP

XML

HTTP/SMTP

TCP/IP

SWS Language

Process

WSDL Extensions

WSDL

XML

Registry (UDDI)

Inspection

SWS Initiative (SWSI)
 -- automate Tasks of:
Discovery
Invocation
Interoperation
Deal Negotiation
Composition
Monitoring
Verification

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

Outline of Part C.

C. Applications -- Policies, Services, and Semantic Integration

1. Ontology Translation and Semantic Integration
 - SWRL uses, ECOIN, financial services
2. End-to-End E-Contracting and Business Process Automation
 - supply chain, e-tailing, auctions, SweetDeal, Process Handbook
3. Business Policies including Trust
 - credit, health, RBAC, XACML, P3P, justifications
4. Semantic Web Services
 - SWSL tasks
5. Prospective Early Adopter areas, strategy, and market evolution
6. Windup and Discussion

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

Opportunity from Semantic Web Services

-- the New Generation Web Platform

- New technologies for Rules (RuleML standard, based on Situated Courteous Description Logic Programs knowledge representation)
 - + New technologies for Ontologies* (OWL standard)
 - + Databases (SQL, XQuery, RDF)
 - + Web Services (WSDL, SOAP, J2EE, .Net)
- Status today:
 - Technologies: emerging, strong research theory underneath
 - Standards activities: intense (W3C, Oasis, ...)
 - Commercialization: early-phase (majors in alpha, startups)

(* *Ontology = structured vocabulary, e.g., with subclass-superclass, domain, range, datatypes. E.g., database schemas.*)

B2B Tasks: Communication for Business Processes with Partners

- B2B business processes involving significant Communication with customers/suppliers/other-partners is overall a natural locus for future first impact of SWS.
- Customer Relationship Management (CRM)
 - sales leads and status
 - customer service info and support
- Supply Chain Management (SCM):
 - source selection
 - inventories and forecasts
 - problem resolution
 - transportation and shipping, distribution and logistics
- orders; payments, bill presentation

Some B2B Tasks (continued)

- bids, quotes, pricing, **CONTRACTING; AUCTIONS**; procurement
- authorization (vs. authentication) for credit or trust
- database-y: e.g.,
 - catalogs & their merging
 - policies
- inquiries and answers; live feedback
- notifications
- trails of biz processes and interactions
- ratings, 3rd party reviews, recommendations
- knowledge management with partners/mkt/society

Vision of Evolution: Agents in Knowledge-Based E-Markets

Coming soon to a world near you:...

- billions/trillions of agents (= k-b applications)
- ...with smarts: knowledge gathering, reasoning, economic optimization
- ...doing our **bidding**
 - but with some autonomy
- *A 1st step: ability to communicate with sufficiently precise shared meaning... via the SEMANTIC WEB*

Some Semantic Web Advantages for Biz

- Builds upon XML's much greater capabilities (vs. HTML*) for structured detailed descriptions that can be processed automatically.
 - Eases application development effort for **assimilation of data in inter-enterprise interchange**
- **Knowledge-Based E-Markets -- where Agents Communicate**
(Agent = knowledge-based application)
 - ∴ potential to revolutionize interactivity in Web marketplaces: B2B, ...
- Reuse same **knowledge for multiple purposes/tasks/app's**
 - Exploit declarative KR; Schemas
- * new version of HTML itself is now just a special case of XML

SW Early Adoption Candidates: High-Level View

- “Death. Taxes. Integration.”
- Application/Info Integration:
 - Intra-enterprise
 - EAI, M&A; XML infrastructure trend
 - Inter-enterprise
 - E-Commerce: procurement, SCM
 - Combo
 - Business partners, extranet trend

SWS Adoption Roadmap: Strategy Considerations

- Expect see beginning in a lot of B2B interoperability or heterogeneous-info-integration intensive (e.g., finance, travel)
 - Actually, probably 1st intra-enterprise, e.g., EAI
- Reduce costs of communication in procurement, operations, customer service, supply chain ordering and logistics
 - increase speed, creates value, increases dynamism
 - macro effects create
 - stability sometimes (e.g., supply chain reactions due to lag; other negative feedbacks)
 - volatility sometimes (e.g., perhaps financial market swings)
 - increase flexibility, decrease lock-in
- Agility in business processes, supply chains

Prospective SW Early Adopters: Areas by Industry or Task

- *We discussed earlier a number of industry or task areas:*
 - Manufacturing supply chain, procurement, pricing, selling, e-tailing, financial/business reporting, authorization/security/access/privacy policies, health records, credit checking, banking, brokerage, contracts, advertising, ...
- Others:
 - travel "agency", i.e.: tickets, packages
 - See Trading Agent Competition, [M.Y. Kabbaj thesis]
 - military intelligence (e.g., funded DAML)

Discussion: Early Adoption Application Prospects for SWS

- What business applications do you think are likely or interesting?
 - By vertical industry domain, e.g., health care or security
 - By task, e.g., authorization
 - By kind of shared information, e.g., patient records
 - By aspect of business relationships, e.g., provider network
- What do you think are entrepreneurial opportunity areas?

Outline of Part C.

C. Applications -- Policies, Services, and Semantic Integration

1. Ontology Translation and Semantic Integration
 - SWRL uses, ECOIN, financial services
2. End-to-End E-Contracting and Business Process Automation
 - supply chain, e-tailing, auctions, SweetDeal, Process Handbook
3. Business Policies including Trust
 - credit, health, RBAC, XACML, P3P, justifications
4. Semantic Web Services
 - SWSL tasks
5. Prospective Early Adopter areas, strategy, and market evolution
6. Windup and Discussion

Outline of Part A.

A. Core -- KR Languages and Standards

1. Intro
2. Overview of Logic Knowledge Representations and Standards
3. Horn Logic / Horn LP
4. Nonmonotonic LP
5. Procedural Attachments
6. RuleML
7. Combining Rules with Ontologies
8. Datatypes
9. Review of OWL and RDF
10. SWRL
11. Additional Aspects and Approaches

Outline of Part B.

B. Tools -- SweetRules, Jena, cwm, and More

(BREAK in middle)

1. Commercially Important pre-SW Rule Systems
 - Prolog, production rules, DBMS
2. Overview of SW Rule Generations
3. 1st Gen.: Rudimentary Interoperability and XML/RDF Support
 - CommonRules, SweetRules V1, OWLJessKB
4. 2nd Gen.: Rule Systems within RDF/OWL/SW Toolkits
 - cwm, Jena-2, and others
5. 3rd Gen.: SW Rule Integration and Life Cycle
 - SweetRules V2

Outline of Part C.

C. Applications -- Policies, Services, and Semantic Integration

1. Ontology Translation and Semantic Integration
 - SWRL uses, ECOIN, financial services
2. End-to-End E-Contracting and Business Process Automation
 - supply chain, e-tailing, auctions, SweetDeal, Process Handbook
3. Business Policies including Trust
 - credit, health, RBAC, XACML, P3P, justifications
4. Semantic Web Services
 - SWSL tasks
5. Prospective Early Adopter areas, strategy, and market evolution
6. Windup and Discussion

Slideset 5: ADDITIONAL (post-printing) Slides for

*“Semantic Web Rules with Ontologies, and
their E-Business Applications”*

by Benjamin Grosof and Mike Dean***

**MIT Sloan School of Management, <http://ebusiness.mit.edu/bgrosfof>*

***BBN Technologies, <http://www.daml.org/people/mdean>*

ISWC-2004 Conference Tutorial (half-day),

at 3rd International Semantic Web Conference, Nov. 7, 2004, Hiroshima, Japan

Version Date: Nov. 6, 2004

11/26/2004 Copyright 2004 by Benjamin Grosfof and Mike Dean. All Rights Reserved

*WRAP-UP:
RESEARCH DIRECTIONS*

Wrap-Up: Big-Picture Research Directions

- Core technologies: Requirements, concepts, theory, algorithms, standards?
 - Rules in combination with ontologies; probabilistic, decision-/game-theoretic
- Business applications and implications: concepts, requirements analysis, techniques, scenarios, prototypes; strategies, business models, market-level evolution?
 - End-to-end e-contracting, finance, trust; ...

Analysis:

High-Level Requirements for SWS

- Support Biz-Process Communication
 - E.g., B2B SCM, CRM
 - E.g., e-contracts, financial info, trust management.
- Support SWS Tasks above current WS layers:
 - Discovery/search, invocation, deal negotiation, selection, composition, execution, monitoring, verification

New Analysis: Key Technical Requirements for SWS

- 1. Combine rules with ontologies, from many web sources, with:
 - Rules on top of ontologies
 - Interoperability of heterogeneous rule and ontology systems
 - Power in inferencing
 - Consistency wrt inferencing
 - Scalability of inferencing
- 2. Hook rules (with ontologies) up to web services
 - Ex. web services: enterprise applications, databases
 - Rules use services, e.g., to query, message, act with side-effects
 - Rules constitute services executably, e.g., workflow-y business processes
 - Rules describe services non-executably, e.g., for discovery, deal negotiation
 - On top of web service process models, coherently despite evolving messiness

Core SW/KR Research Challenges on Rules and Ontologies

- Integrating rules with ontologies
 - Rules refer to ontologies (e.g., in RuleML)
 - Rules to specify ontologies (e.g., Description Logic Programs)
 - Rules to map between ontologies (e.g., ECOIN)
 - Combined rules + ontologies knowledge bases (e.g., RuleML + OWL)
- Describing business processes & web services via rules + ontologies
 - Capture object-oriented process ontologies
 - Default inheritance via rules (e.g., Courteous Inheritance)
 - Wrapper/transform to legacy C++, Java, UML
 - Develop open source knowledge bases (e.g., MIT Open Process Handbook Initiative)
 - Also:
 - Rules query web services (e.g., in RuleML Situated feature)
 - Rules trigger actions that are web services (e.g., ditto)
 - Event triggering of rules (e.g., capture ECA rules in RuleML)
 - Rules in process models, e.g., cf. OWL-S, PSL

*ADDITIONAL
REFERENCES &
RESOURCES
FOLLOW*

*N.B.: some references & resources
were given on various earlier slides*

References & Resources I:

Standards on Rules and Ontologies

- <http://www.ruleml.org> RuleML *Includes links to some tools and examples.*
- <http://www.w3.org/Submission/2004/SUBM-SWRL-20010521> SWRL
 - <http://www.daml.org/committee> Joint Committee. Besides SWRL (above) this includes:
 - <http://www.daml.org/2004/11/fo1/> SWRL-FOL
 - <http://www.ruleml.org/fo1> FOL RuleML (also see RuleML above)
 - <http://www.daml.org/rules> DAML Rules
- <http://www.swsi.org> Semantic Web Services Initiative. Especially:
 - Semantic Web Services Language (SWSL), incl. SWSL-Rules and SWSL-FOL and overall requirements/tasks addressed
- <http://cl.tamu.edu> Simple Common Logic (successor to Knowledge Interchange Format)
- Also: Object Management Group (OMG) has efforts on rules and ontologies (cooperating with RuleML and OWL)
- Also: JSR94 Java API effort on Rules (cooperating with RuleML)

References & Resources II: Standards on Rules and Ontologies

- <http://www.w3.org> World Wide Web Consortium, esp.:
 - .../2001/sw/ Semantic Web Activity, incl. OWL and RDF
 - .../2002/ws/ Web Services Activity, incl. SOAP and WSDL
 - www-rdf-rules@w3.org Rules discussion mailing list
 - www-sws-ig@w3.org Semantic Web Services discussion mailing list
 - P3P privacy policies
 - XQuery XML database query
- <http://www.oasis-open.org> Oasis, esp. on web policy & web services:
 - XACML XML access control policies
 - ebXML e-business communication in XML
 - Legal XML
 - BPEL4WS Business Processes as Web Services
 - Web Services Security

References & Resources III: LP with NAF

- Przymusiński, T., “Well Founded and Stationary Models of Logic Programs”, *Annals of Artificial Intelligence and Mathematics (journal)*, 1994. *Constructive model theory, and proof theory, of well founded semantics for LP.*
- Van Gelder, A., Schlipf, J.S., and Ross, K.A., “The Well-Founded Semantics for General Logic Programs”, *Journal of the ACM* 38(3):620-650, 1991. *Original theory of well founded semantics for LP.*
- Gelfond, M. and Lifschitz, V., *The Stable Model Semantics for Logic Programming*, Proc. 5th Intl. Conf. on Logic Programming, pp. 1070-1080, 1988, MIT Press. *Original theory of stable semantics for LP.*
- Lloyd, J.W., “Foundations of Logic Programming” (book), 2nd ed., Springer-Verlag, 1987. *Includes Lloyd-Topor transformation, and correspondence of semantics to FOL in definite Horn case. Reviews theory of declarative LP. Somewhat dated in its treatment of theory of NAF since it preceded well founded and stable semantics.*
- Baral, C., and Gelfond, M., “Logic Programming and Knowledge Representation”, *J. Logic Programming*, 1994. *First and last parts review theory of declarative LP. Stronger on stable semantics than on well founded semantics.*

References & Resources IV: Misc. on Rules and Ontologies

- <http://ccs.mit.edu/ph> MIT Process Handbook, incl. Open Process Handbook Initiative
- Grosz, B., Horrocks, I., Volz, R., and Decker, S., “Description Logic Programs: Combining Logic Programs with Description Logic”, Proc. 12th Intl. Conf. on the World Wide Web., 2003. *On DLP KR and how to use it.*
- Grosz, B., “Representing E-Commerce Rules Via Situated Courteous Logic Programs in RuleML”, Electronic Commerce Research and Applications (journal) 3(1):2-20, 2004. *On situated courteous LP KR, RuleML overview, and e-commerce applications of them.*
- Grosz, B. and Poon, T., “SweetDeal: Representing Agent Contracts with Exceptions using Semantic Web Rules, Ontologies, and Process Descriptions”, Intl. Journal of Electronic Commerce 8(4), 2004. *On SweetDeal e-contracting app.*
- Firat, A., Madnick, S., and Grosz, B., “Financial Information Integration in the Presence of Equational Ontological Conflicts”, Proc. Workshop on Information Technologies and Systems, 2002. *On ECOIN. Also see A. Firat's PhD thesis, 2003.*

References & Resources V: Misc. on Rules and Ontologies

- Grosz, B., Gandhe, M., and Finin, T., “SweetJess: Translating DamlRuleML To Jess”. Proc. Intl. Wksh. On Rule Markup Languages for Business Rules on the Semantic Web, 2002 (the 1st RuleML Workshop, held at ISWC-2002). See extended and revised working paper version, 2003. *On SweetJess translation/interoperability between RuleML and production rules.*
- Forgy, C.L., “Rete: A Fast Algorithm for the Many Pattern / Many Object Pattern Match Problem”. Artificial Intelligence 19(1):17-27, 1982. *On the key Rete algorithm for production rules inferencing.*
- Friedman-Hill, E., “Jess in Action” (book), 2003. *On Jess and production rules.*
- Ullman, J., “Principles of Knowledge Base and Database Systems Vol. I” (book), 1988. *See esp. the chapter on Logic Programs, incl. algorithm for stratification.*
- <http://xsb.sourceforge.net> XSB Prolog. See papers by D. Warren *et al.* for theory, algorithms, citations to standard Prolog literature (also via <http://www.sunysb.edu/~sbprolog>)
- (*ff. needs tweaking:*) Horrocks, I., and Patel-Schneider, P., paper on OWL Rules and SWRL, Proc. WWW-2004 Conf., 2004. *On SWRL theory incl. undecidability.*
- (*ff. needs tweaking:*) Horrocks, I., and Bechhofer, S., paper on Hoolet approach to SWRL inferencing via FOL theorem-prover, Proc. WWW-2004 Conf., 2004. *On SWRL inferencing.*

References & Resources VI: More on Courteous and Situated

- Grosf, B., Labrou, Y., and Chan, H., “A Declarative Approach to Business Rules in Contracts”, Proc. 1st ACM Conf. on Electronic Commerce, 1999, ACM Press. *On courteous LP KR with mutex's, and its e-contracts applications.*
- Grosf, B., “Courteous Logic Programs: Prioritized Conflict Handling for Rules”, Proc. Intl. Logic Programming Symposium., 1997. See extended version: IBM Research Report RC 20836, 1997. *Basic version courteous LP (since generalized).*
- Grosf, B., “A Courteous Compiler from Generalized Courteous Logic Programs To Ordinary Logic Programs”, (IBM) research report extension to “Compiling Courteous Logic Programs Into Ordinary Logic Programs”, 1999. Available via <http://ebusiness.mit.edu/bgrosf> or IBM incl. in CommonRules documentation. *Details on courteous compiler/transform.*
- Grosf, B., Levine, D.W., Chan, H.Y., Parris, C.J., and Auerbach, J.S., “Reusable Architecture for Embedding Rule-based Intelligence in Information Agents”, Proc. Wksh. on Intelligent Information Agents, at ACM Conf. on Information and Knowledge Management, ed. T. Finin and J. Mayfield, 1995. Available also as IBM Research Report RC 20305. *Basic situated LP paper. Also see 1998 patent.*
- Grosf, B., “Building Commercial Agents: An IBM Research Perspective (Invited Talk). Proc. 2nd Intl. Conf. on the Practical Applications of Intelligent Agents and Multi-Agent Technology (PAAM97), pub. The Practical Applications Company, 1997. Also available as IBM Research Report RC 20835. *Overview of situated LP.*

Resources VII: Web Services Applications

- <http://zdnet.com.com/2100-1106-975870.html> Fidelity's web services for EAI
- http://www.amazon.com/gp/browse.html/ref=smm_sn_aws/002-8992958-7364050?node=3435361 Amazon's web services – 1000's of developers