Introduction to RuleML

Benjamin Grosof
MIT Sloan School of Management
Information Technologies group
http://www.mit.edu/~bgrosof

Part 2 of 2:

ADDITIONAL OPTIONAL SLIDES

to accompany the

Slides presented at 10/29/2002 Teleconference Meeting of Joint US/EU ad hoc Markup Language Committee

http://www.daml.org/committee

With thanks to Steve Ross-Talbot, Bruce Spenser, Said Tabet, and Gerd Wagner

MORE-DETAILS SLIDES FOLLOW

RuleML News

- Overall: more tools, more participants.
- Situated courteous LP (SCLP) as extension of spec.
  - Implemented in SweetRules [Grosof 2001] inferencing and translation.
- DAMLRuleML draft spec.: DAML+OIL spec. for RuleML's syntax.
  - Implemented in SweetJess [Grosof, Gandhe, and Finin 2002].
- SweetJess translator of SCLP RuleML to/from Jess, inferencing via Jess.
  - 1st bridge between Prolog/RDBMS and OPS5/ECA.
- Reactive rules subgroup effort launching.
- Applications:
  - Configurable reusable e-contracts (SweetDeal).
  - Ontology-based financial knowledge integration (ECOIN).
- Oasis interest in “Policy RuleML” (tentative name) as possible TC.
  - RuleML for interchange between policy languages.
- Engaging on W3C front, as well.
- Events aimed for in 2003: W3C Plenary, WWW Conf., ISWC.
- More news is on the RuleML main site http://www.dfki.de/ruleml

Standardizing XML Rules: Overall Goals

- Provide a basis for a standardized rule markup language, with declarative KR semantics
  - interoperability of heterogeneous rule systems and applications
  - information integration of heterogeneous rule KB's/services
- Start with commercially important flavors of rules
- Start simple with a kernel KR, then add extensions incrementally.

Standardizing XML Rules: More Goals

- Add extensions incrementally to:
  - raise KR expressiveness and syntactic convenience
  - connect cleanly to procedural mechanisms
  - pass-thru/bundle-in system-specific (meta-)info
  - exploit Web-world functionality, standards
- Synergize with other KR aspects of Semantic Web:
  - RDF;   Ontologies: DAML+OIL/Description-Logic
    - rules in/for ontologies, ontologies for/of rules
- Complement XML non-SW ontologies already evolving
- Synergize with other Web standards: P3P APPEL, XML Query, Web Services, ...

Incremental Strategy of Standards Development

- *Initial Step*: *Keep It Simple,* focus primarily on:
  - Currently Commercially Important (CCI) kinds of rules
  - with XML syntax
  - with shared semantics and interoperability
  - *BUT: foresee to max. smooth evolution, back-compatibility*
- *Later*: get *fancier* in regard to:
  - Web-izing: features, synergy with other standards
  - KR expressiveness
  - incorporate new fundamental research results & consensus
- *Rationale: speed acceptance & deployment; avoid "bleeding edge"*

Technical Challenge #1: which initial core KR semantics?

- *Analytic Insight [many]:*
  - Horn FOL is a shared KRsem. E.g., KIF conformance level
- *Analytic Insight [Grosof 99]:*
  - *!!Can do better -- closer, more expressive!!*
  - Start with Horn Logic Program (LP), esp. Datalog
    - closer correspondence to what CCI rule systems actually do
    - generate ground-literal conclusions only, no other "tautologies" (e.g., OR's)
    - Unique Names Assumption (UNA) is typical; opt.: explicitly add equalities
    - {Datalog + {bounded # logical variables per rule} } is frequent, tractable
  - Extend LP to negation, priorities, procedures
    - needed in CCI rule systems, fairly well-understood fundamentally

Technical Challenge #2: how to handle CCI non-monotonicity?

- CCI non-monotonicity is heavily used, includes:
  - negation
  - priorities (Prolog, OPS5, DB updates, inheritance exceptions)
    - Common CCI Theme: enable modularity in specification

- *Analytic Insight [many]:*
  - negation-as-failure (NAF), not classical negation, is the form of negation typically used in CCI
    - more natural/easy to implement, more flexible

Semantics of Negation As Failure in CCI

- canonical semantics of NAF <u>in LP</u> is well-understood theoretically since 1990's:
  - <u>Well-Founded Semantics</u> (WFS); nuanced for unrestrictedly recursive rules
  - consensus has formed in fundamental research community
  - only modestly increases computational complexity compared to Horn (frequently linear, at worst quadratic)

- ...but practice in Prolog and other CCI is often "sloppy" (incomplete / cut-corners) relative to canonical semantics
  - in cases of recursive rules, WFS algorithms required are more complex
  - ongoing diffusion of WFS theory & algorithms, beginning in Prolog's

Ordinary Logic Programs as Shared KR

- {Horn LP} + NAF = "Ordinary" LP (OLP)
  - a.k.a. "general", "normal", …
  - e.g., "pure" Prolog is backward-direction OLP

Ordinary Logic Programs as basic representation: Definition

- A LP is a set of (premise) rules; semantically, it specifies a set of conclusions.
- replyInterval(?msg,CustomerRep,quick)
- $\leftarrow$ from(?msg,?s) $\wedge$ customer(?s) $\wedge$ ~urgency(?msg,low). *Example Rule*
-
- where the "?" prefix indicates a logical variable.
- Generally, a rule has the form of Head IF Body :
- $H \leftarrow B_1 \wedge \ldots \wedge B_j \wedge \sim B_{j+1} \wedge \ldots \wedge \sim B_m$.
- where $m \geq 0$; $\wedge$ stands for logical "AND"; $\leftarrow$ stands for logical "IF"; and $H, B_1, \ldots, B_m$ are each an <u>atom</u> with form: $\text{Predicate}(\text{Term}_1, \ldots, \text{Term}_k)$.
- A predicate = a relation. An atom semantically denotes a boolean.
- ~ stands for <u>negation-as-failure</u> (a.k.a. weak negation, default negation).
  - The negation-as-failure construct is logically non-monotonic.
  - Intuitively, ~p means p's truth value is either *false* OR *unknown*.

Ordinary Logic Programs: Definition (continued)

- Each argument Term_1, ..., Term_k is a <u>term</u>.
- A term is either a <u>logical constant</u> (e.g., “Joe”) OR a <u>logical variable</u> (e.g., “?msg”) OR a <u>functional expression</u> of the form:
-   LogicalFunction(Term_1, ..., Term_k)
- A functional expression semantically essentially denotes a logical constant.
- A term, atom, or rule is called “<u>ground</u>” when it has no logical variables.

- A <u>fact</u> is a ground rule with empty body.
- A primitive <u>conclusion</u> has the form of a ground atom (compound conclusions are built up from these via logical operators such as AND etc.).
- Semantically, a rule or LP stands for the set of all its ground instances.
- (Observe that a rule body can represent an expression in relational algebra cf. relational DB’s (e.g., SQL).)

Ordinary Logic Programs as basic representation: Advantages

- Declarative: semantics is independent of inferencing procedure implementation, e.g., forward vs. backward chaining, sequencing of executing rules or conditions within rules.
- Expressive: relational expressions cf. SQL, large fragment of first-order logic, chaining, basic logical non-monotonicity (unlike first-order logic / ANSI-draft Knowledge Interchange Format).
- Efficient: computationally tractable given two reasonable restrictions:
  - 1. Datalog = no logical functions of non-zero arity.
  - 2. Bounded number v of logical variables per rule.
  - m = O(n^(v+1)), where n = ||LP||, m = ||ground-instantiated LP||.
  - Inferencing time is O(m) for broad case (stratified), O(m^2) generally (for well-founded semantics).
  - By contrast, first-order-logic inferencing is NP-hard.

Ordinary Logic Programs: Advantages (continued)

- Widely deployed and familiar:
  - relational DB's, SQL
  - Prolog
  - knowledge-based systems and intelligent agents
    - (e.g., IBM's Agent Building Environment)
- Common core shared semantically by many rule systems: e.g.,
  - relational DB's, SQL
  - Prolog
  - production rules (OPS5 heritage)
  - Event-Condition-Action rules
  - first-order-logic

how to handle CCI non-monotonicity? continued

- *Synthetic Insight* [Grosof 97..99]:
  - "Courteous" LP (CLP) [Grosof 97..99] is able to represent the basic kinds of priorities used in CCI
    - static rule sequence, e.g., in Prolog
    - dynamically-computed rule sequence, e.g., in OPS5
    - inheritance with exceptions
    - DB updates
  - CLP only moderately increases computational complexity compared to OLP (frequently linear, worst-case cubic)
  - CLP modular for software engineering
    - compileable into OLP (preserving ontology)

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)
- $\leftarrow$ preferredCustomerOf(?Buyer,?Seller) $\wedge$
- purchaseOrder(?Order,?Buyer,?Seller) .
- <leadTimeRule2> orderModificationNotice(?Order,30days)
- $\leftarrow$ minorPart(?Buyer,?Seller,?Order) $\wedge$
- purchaseOrder(?Order,?Buyer,?Seller) .
- <leadTimeRule3> orderModificationNotice(?Order,2days)
- $\leftarrow$ preferredCustomerOf(?Buyer,?Seller) $\wedge$
- orderModificationType(?Order,reduce) $\wedge$
- orderItemIsInBacklog(?Order) $\wedge$
- purchaseOrder(?Order,?Buyer,?Seller) .
- overrides(leadTimeRule3 , leadTimeRule1) .
- $\perp \leftarrow$ orderModificationNotice(?Order,?X) $\wedge$
- orderModificationNotice(?Order,?Y); GIVEN ?X $\neq$?Y.

Technical Challenge #3: how to handle CCI procedural aspects?

- *Ignoring procedural control (cf. inferencing control strategies)…*
- CCI procedural aspects are heavily used, including:
  - Prolog: built-ins
  - OPS5/ECA: actions, some conditions
    - key to embeddability in mainstream software dev.
  - "triggers" and "active rules" in relational DB's

- *Analytic Insight* [Grosof 99]:
  - view as <u>procedural attachments</u> (cf. KR theory)

how to handle CCI procedural aspects? continued

- *Synthetic Insight* [Grosof 95..00]:
  - "Situated" LP (SLP) [Grosof 97..00] appears able to represent the basic kinds of procedural attachments used in CCI, though with more discipline(/restrictions)
    - "aproc" = external attached procedure
    - "effecting": drawing pure-belief conclusion triggers invocation of action aproc for sake of its side-effects
    - "sensing": test pure-belief antecedent condition by invoking purely-informational query to aproc
    - discipline: restrict state changes from external procedures
      - querying (sensor) attached procedures does not change state
      - performing effector associate predicates with external procedures

Situated LP's: Overview

- phoneNumberOfPredicate ::s:: BoeingBluePagesClass.getPhoneMethod .
  ex. Of sensor statement
- shouldSendPagePredicate ::e:: ATTPagerClass.goPageMethod .
  ex. effector statement
- Sensor procedure may require some arguments to be ground, i.e., bound; in general it has a specified <u>binding-signature</u>.
- Enable <u>dynamic loading</u> and <u>remote loading</u> of the attached procedures (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.

Going Beyond KIF/CommonLogic

- KIF/CL is KR Ag. Comm. Lang.'s point of departure:
  - Intent: general-knowledge interlingua.
  - Emerging standard, in ISO process
  - Main focus: classical logic, esp. first-order.
    - This is the declarative core, with deep semantics.
  - Has major limitations:
    - general-purpose-ness
    - logically monotonic
    - pure-belief
      - no invoking of procedures external to the inference engine.

Criteria for Agent-Communication Rule Representation

1
- *High-level:* Agents reach common understanding; ruleset is easily modifiable, communicatable, executable.

2
- Inter-operate: heterogeneous commercially important rule systems.
- Expressive power, convenience, natural-ness.
- ... but: computational tractability.
- Modularity and locality in revision.

3
- Declarative semantics.
- Logical non-monotonicity: default rules, negation-as-failure.
  - essential feature in commercially important rule systems.
- Prioritized conflict handling.
- Ease of parsing.
- Integration into Web-world software engineering.
- Procedural attachments.

OLP

Courteous

XML

Situated

MORE OPTIONAL SLIDES FOLLOW

Important KR's today in E-Business

- Rules, relational databases
  - emerging standard: RuleML
- Description Logic, frames, taxonomies
  - emerging standard: DAML+OIL
- (other) Classical Logic
  - emerging standard: Knowledge Interchange Format (KIF)
- Bayes Nets & Decision Theory: probabilities, dependencies, utilities
  - early, primarily for researchers: Bayes Net Interchange Format (BNIF)
- (other) Data Mining inductive predictive models: neural nets, associations, fuzzy, regressions, …   -- early: Predictive Model Markup Lang.
- *Arguably:* Semi-Structured Data: XML Query, RDF
- *Arguably:* UML

Applications of Agent Communication in Knowledge-Based E-Markets (KBEM)

- Bids in auctions and reverse auctions
- Orders in supply chain or B2C
- Contracts/Deals/Proposals/RequestsForProposals
  - prices; product/service descriptions; refunds, contingencies
- Buyer/Seller interests, preferences, capabilities, profiles
  - recommender systems; yellow pages; catalogs
- Ratings, reputations; customer feedback or problems
- Demand forecasts in manufacturing supply chain
- Constraints in travel planning
- Creditworthiness, trustworthiness, 3rd-party recommendations
- *Industry-verticals: computer parts, real estate, …*

Technology Research Directions: KR for Agent Communication

- Aims:
  - deeper reasoning intra-agent
    - "<u>understanding</u>" what receive
  - more modularity in:
    - content
    - software engineering
  - KR of the kind needed for e-market applications
    - catalogs, contracts, negotiation/auctions, trust, profiles/preferences/targeting, …
  - play with XML standards, capabilities, mentality

Technology Research Direction: KR on the Web

- Apply KR viewpoint and techniques to Web info
- "Web-ize" the KR's
  - exploit Web/XML hyper-links, interfaces, tools
  - think global, act global : as part of whole Web
- Radically raise the level of shared meaning
  - level = conceptual/abstraction level
  - meaning = sanctioned inferences / vocabularies
  - shared = tight correspondence
- "The Semantic Web", "The Web of Trust" [Tim B-L]
- Build: The Web Mark II

Current Uses of Rules in E-Business

- Inferencing in
  - business rules
  - workflow
  - database queries and triggers
  - intelligent agents, KB systems
- Transformation in (XML) document translation
- *Identified as a Design Issue of the W3C Semantic Web*

Automating Contracting

- "Contract" in broad sense: = offering or agreement.
- "Automate" in deep sense: =
  - 1. Communicatable automatically.
  - 2. Executable within appropriate context of contracting parties' business processes.
  - 3. Evaluable automatically by contracting parties.
    - "reason about it".
  - 4. Modifiable automatically by contracting parties.
    - negotiation, auctions.

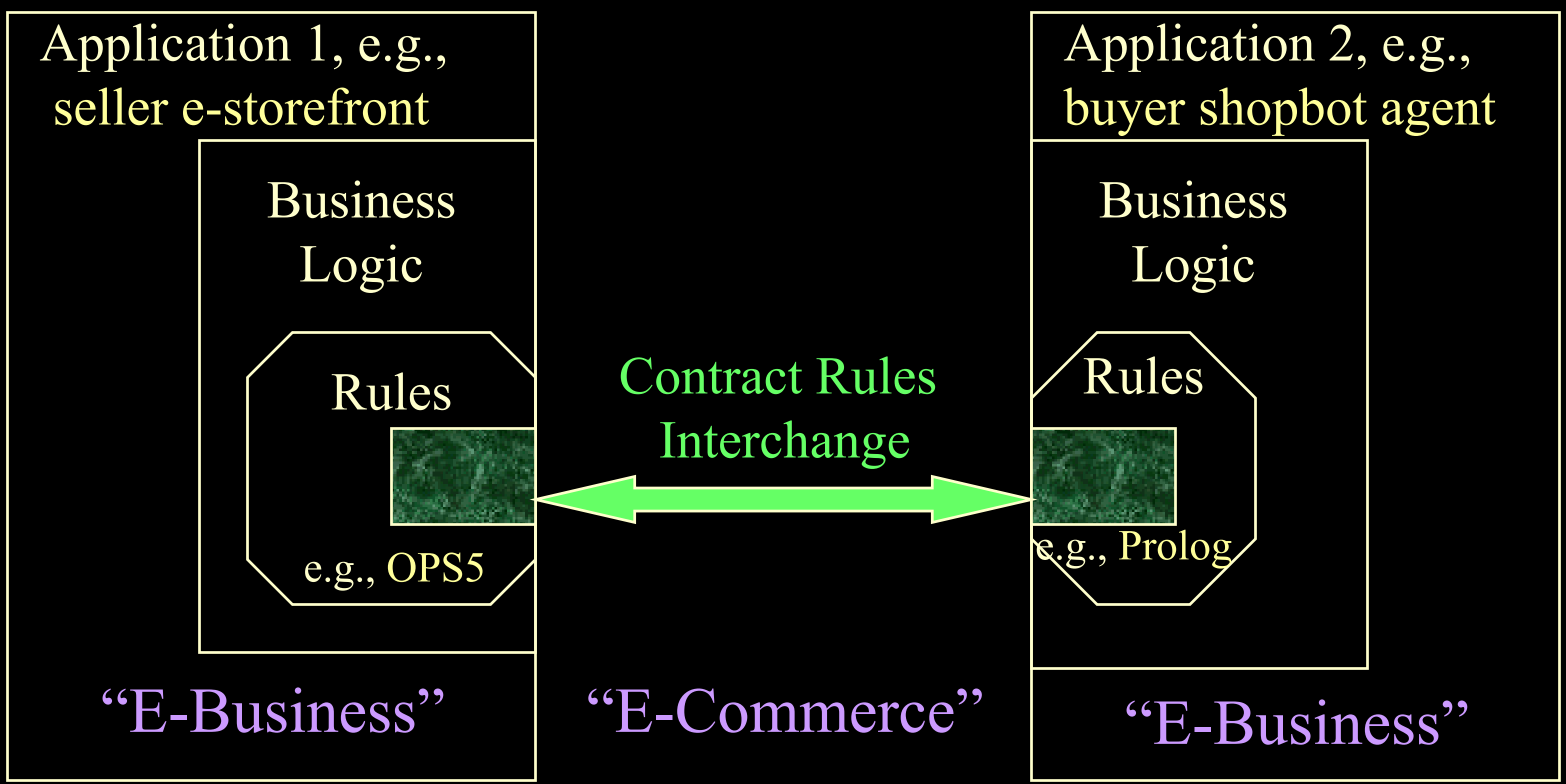
Idea/Vision #1: Rule-based Contracts for E-commerce

- 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.

Examples of Rules in Contracts

- Terms & conditions, e.g., price discounting.
- Service provisions, e.g., rules for refunds.
- Surrounding business processes, e.g., lead time to order.
- Price vs. quantity vs. delivery date.
- Cancellations.
- Discounting for groups.
- Product catalogs: properties, conditional on other properties.
- Creditworthiness, trustworthiness, authorization.

Contract Rules across Applications / Enterprises



Contracting parties integrate e-businesses via shared rules.

Courteous LP's: 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., $\perp \leftarrow$ discount(?product,5%) $\wedge$ discount(?product,10%) .
  - E.g., $\perp \leftarrow$ loyalCustomer(?c,?s) $\wedge$ premiereCustomer(?c,?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:
    - overrides(rule1,rule2) means rule1 is higher-priority than rule2.
    - Each rule optionally has a rule label whose form is a functional term.
    - overrides can be reasoned about, just like any other predicate.

Priorities are available and useful

- Priority information is naturally available and useful. E.g.,
  - recency: higher priority for more recent updates.
  - specificity: higher priority for more specific cases (e.g., exceptional cases, sub-cases, inheritance).
  - authority: higher priority for more authoritative sources (e.g., legal regulations, organizational imperatives).
  - reliability: higher priority for more reliable sources (e.g., security certificates, via-delegation, assumptions, observational data).
  - closed world: lowest priority for catch-cases.
- Many practical rule systems employ priorities of some kind, often implicit, e.g.,
  - rule sequencing in Prolog and production rules.
    - courteous subsumes this as special case (totally-ordered priorities), plus enables: merging, more flexible & principled treatment.

Prioritized argumentation in an opposition-locale.

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

↓

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

Run Rules for p1,...,pk

↓

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

↓

Prioritized Refutation

↓

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

↓

Skepticism

↓

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

Situated LP's: Overview

- Point of departure: LP's are <u>pure-belief</u> representation, but most practical rule systems want to invoke external procedures.
- <u>Situated</u> 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 **link** statement specifies an association from a predicate to a procedural call pattern, e.g., a method. A link is specified as part of the representation. I.e., a SLP is a <u>conduct set</u> that includes links as well as rules.

Summary: Courteous (Situated) LP's as Core KR

- Key Observations about Declarative OLP:
  - captures common core among commercially important rule systems.
  - is expressive, tractable, familiar.
  - advantages compared to classical logic / ANSI-draft KIF:
    - + + logical non-monotonicity, negation-as-failure.
    - – – disjunctive conclusions.
    - + + tractable.
    - + + procedural attachments: Situated LP's.
- Cleverness of Courteous extension to the OLP representation:
  - prioritized conflict handling → modularity in specification. And consistency.
  - courteous compiler → modularity in software engineering.
  - mutex's & conflict locales → keep tractability. (Compiler is O(n^3).)

Declarative Semantics at Core

- Desire: deep semantics (model-theoretic) to
  – understand and execute imported rules.
- Possible only for shared expressive subsets: "cores".
  – Rest translated with superficial semantics.
- Approach: declarativeness of core / rep'n (in sense of knowledge representation theory).
  – A given set of premises entails a set of sanctioned conclusions. Independent of implementation & inferencing control (bkw vs. fwd).
  – Maximizes overall advantages of rules:
    - Non-programmers understand & modify.
    - Dynamically (run-time) modify.

Technical Approach of RuleML

- Start with: Datalog Logic Programs with rules labeled *as kernel*
- Add: expressive extensions/restrictions, URI's
  - negation-as-failure (well-founded semantics); classical negation (limited)
  - prioritized conflict handling cf. Courteous Logic Programs *(stays tractable!)*
    - modular rulesets; modular compiler to Ordinary Logic Programs
  - procedural attachments: actions, queries ; cf. Situated Logic Programs
  - logical functions: standard built-ins, user-defined
  - 1st-order logic type expressiveness cf. Lloyd LP's, DAML+OIL, KIF
  - *more:* equivalence/rewriting rules; ... temporal, Bayesian, fuzzy, …
- Family of DTD's: a generalization-specialization hierarchy (lattice)
  - define DTD's modularly, using XML entities (~macros)
  - optional header to describe expressive-class using "meta-"ontology

Webizing Rule KR

- URIs for logical vocabulary and knowledge subsets
- labels for rules/rulebases, import/export
- headers: meta-data describes doc's expressive class
- procedural attachments using Web protocols;
  queries or actions via CGI/servlets/SOAP/…

- *Other practical mechanics:*
  – build on existing W3C standards: namespaces, …
  – share mechanisms with RDF/RDFS, DAML+OIL
  – use ontologies for rules, and rules for ontologies
    - ontology tags in: rulebase, predicate symbol, …

RuleML has some First Steps of Webizing Rule KR

- URIs for logical vocabulary and knowledge subsets
  - RuleML V0.8: predicates, functions, rules, rulebases
  - RuleML V0.8: labels for rules/rulebases
- Support RDF:
  - RuleML V0.8:
    - syntax: mostly <u>un</u>orderedness of graph
    - … with explicit orderedness
    - partial first drafts of alternative RDF syntax
- Support evolution and tight description of KR expressive classes:
  - RuleML Syntax defined as generalization-specialization <u>lattice</u> of DTD's
    - uses XML entity mechanism

RuleML's First Steps of Webizing Rule KR (continued)

- Exploratory features in RuleML 0.8 [FEEDBACK PLEASE!]:
  - meta "role" convention in DTD: to aid RDF-friendliness
  - argument "roles" for atom/term argument lists
    - step toward OO support and RDF support

- RuleML Tools beginning to appear
  - several links on website

More Bibliography

- Firat, Aykut and Madnick, Stuart, and Grosof, Benjamin. "Knowledge Integration to Overcome Ontological Heterogeneity: Challenges from Financial Information Systems". Proc. Intl. Conf. on Information Systems (ICIS), 12/02.
- Firat, Aykut and Madnick, Stuart, and Grosof, Benjamin. "Financial Information Integration in the Presence of Equational Ontological Conflicts". Proc. Wksh. on Information Technologies and Systems (WITS-02), held 12/02 at the Intl. Conf. on Information Systems (ICIS). *Describes ECOIN system.*
- Grosof, Benjamin. "Representing E-Business Rules for the Semantic Web: Situated Courteous Logic Programs in RuleML". Proc. Wksh on Information Technologies and Systems (WITS-01), held 2001 at the Intl. Conf. on Information Systems (ICIS). *Describes SweetRules tool as well as RuleML.*
- Li, Ninghui and Grosof, Benjamin and Feigenbaum, Joan. "Delegation Logic: A Logic-based Approach to Distributed Authorization". Forthcoming, ACM Trans. on Information Systems Security (TISSEC) journal.