Slideset 7 – Optionals – of "E-Commerce Applications of Semantic Web Services"

Benjamin Grosof

MIT Sloan School of Management,
http://ebusiness.mit.edu/bgrosof

EC '04 Conference Tutorial

5th ACM Conference on Electronic Commerce, May 17, 2004, New York City

Version Date: April 29, 2004

OPTIONAL SLIDES FOLLOW

OPTIONAL SLIDES on RDF FOLLOW

RDF vs. XML

- RDF original goals were: (1) to represent open meta-data created by multiple authors over the Web to describe and annotate arbitrary Web resources, e.g., whole websites, e.g., for use by digital librarians; and (2) to facilitate logical KR / SW. Lots of emphasis on an abstract data model.
- XML original goals largely to facilitate human-read document processing, then later to accommodate structured data cf. databases. More initial focus on syntax than on its own data model.

RDF vs. XML, continued

- XML document is a: labelled directed graph that is also:
  - Ordered (sequence of children matters)
  - Tree (a restriction!)
- RDF analogue of an XML document is a set of arcs ("triples"); this is a labelled directed graph that is:
  - Unordered (but can declare explicit order where need)
    - Good for general data modeling, e.g., in mainstream software engineering
  - NB: Cycles permitted (no tree restriction)
- RDF also permits:
  - "Reification", i.e., naming of an RDF triple so that it can be a node in another RDF triple.
- RDF encourages the nodes and arc labels to be URI's themselves. XML less general and open in this regard – it's clumsier, is one way to view it.

RDF vs. XML, continued more

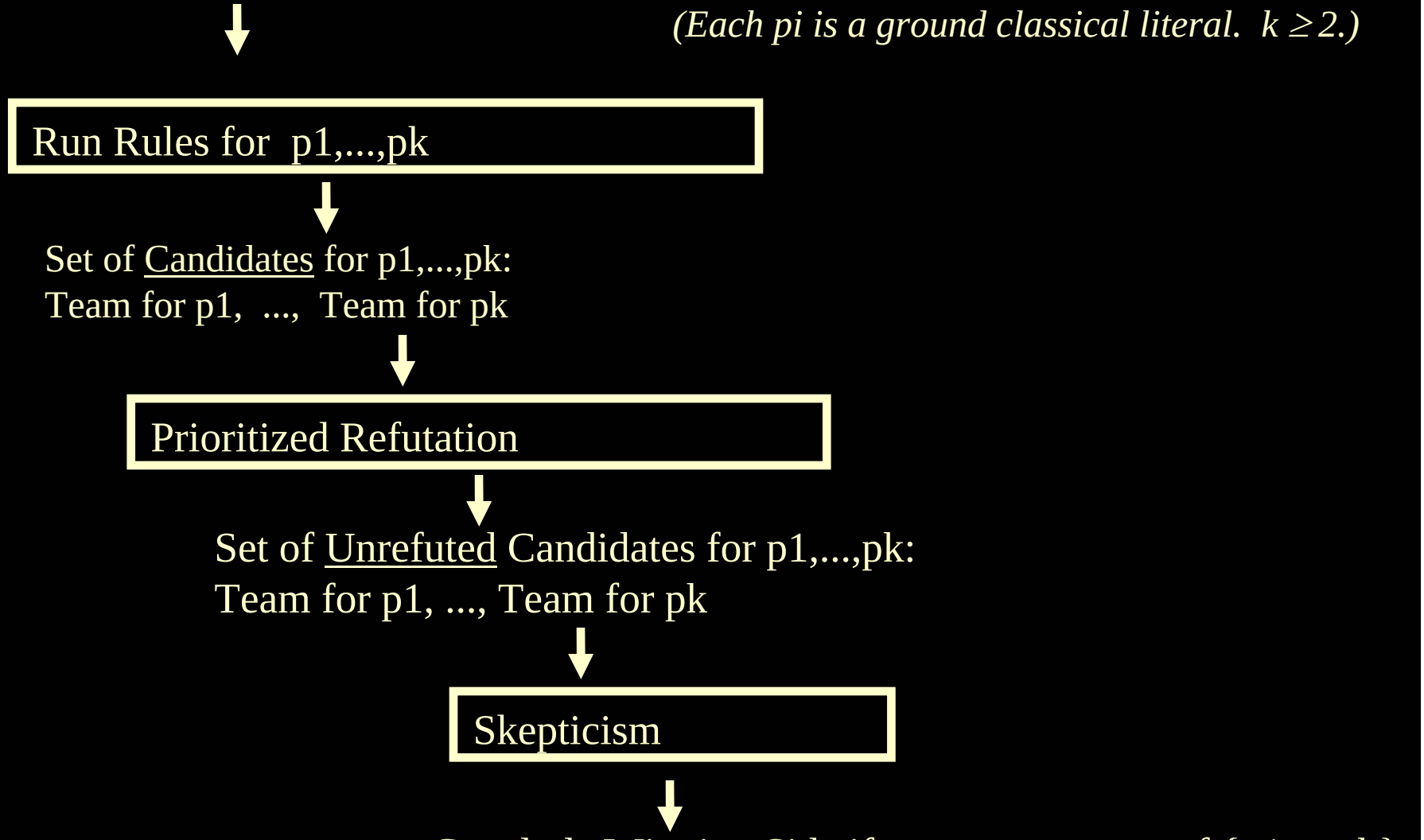
- RDF (vs. XML) facilitates fine-grain sharing and annotation
- RDF adoption is much much less (yet) than XML.
- RDF is usually used with a particular XML syntax, but there are several for it.
- The theoretical understanding that underpins RDF is not quite finished yet.
- RDF/RDF-Schema also includes some treatment of types and classes; XML Schema does too in a more practical manner.
- RDF and XML will probably be converged in the next several years – their data models are fairly close already. But XML has lots of inertia so far.

OPTIONAL SLIDES on Courteous LP FOLLOW

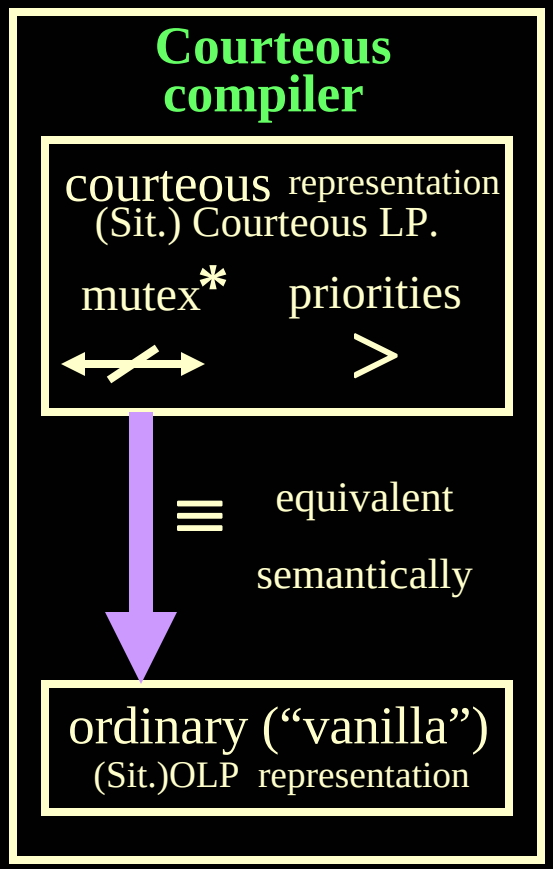
Prioritized argumentation in an opposition-locale.

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

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



Courteous feature: compileable, tractable



Tractable compilation:

O(n^3), often linear

Tractable inference: e.g., worst-case

when no ctor's ("Datalog")

& bounded v = |var's per rule|

is equivalent to OLP with v → (v+2)

Preserves ontology.

Plus extra predicates for

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

* classical negation too

Courteous LP's: Keys to Tractability

- Overall: mutex's & conflict locales $\rightarrow$ keep tractability.
- LP's: disallow disjunctive conclusions, essentially. Classical allows $\Rightarrow$ NP-hard.
- LP's: disallow contraposition (= $\{\neg a \leftarrow . \ , \ a \leftarrow b \wedge c.\} \Rightarrow (\neg b \vee \neg c)\}$) which requires disjunctive conclusions. "Directional". Classical allows $\Rightarrow$ NP-hard.
- Highly expressive prioritized rule representations (e.g., Prioritized Default Logic, Prioritized Circumscription) allow minimal conflict sets of arbitrary size
  $\Rightarrow$ NP-hard overhead for conflict handling.
- Courteous conflict handling involves essentially only pairwise conflicts, i.e., minimal conflict sets of size 2. (Current work: possibly generalize to size k.)
  - Novelty: generalize to pairwise mutex's beyond $\bot \leftarrow p \wedge \neg p$, e.g., partial-functional, thus avoid need for contraposition and larger conflict sets.
- Courteous conflict handling is local within an opposition locale: a set of rules whose heads oppose each other through mutex's. Refutation and Skepticism are applied within each locale.

OPTIONAL SLIDES on Situated LP FOLLOW

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.

Situated LP's: Overview

- 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 **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 conduct set that includes links as well as rules.

Situated LP's: Overview (cont.'d)

- phoneNumberOfPredicate ::s:: BoeingBluePagesClass.getPhoneMethod . *ex. sensor link*
- shouldSendPagePredicate ::e:: ATTPagerClass.goPageMethod . *ex. effector link*
- Sensor procedure may require some arguments to be ground, i.e., bound; in general it has a specified binding-signature.
- Enable dynamic or remote invocation/loading 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. (Declarative = Independent of inferencing control.)

Overview: Semantics of Situated Logic Programs

- 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.
    - But often intuitively less appropriate if only doing backward inferencing.
  - Separates pure-belief conclusion from action.

Overview: Semantics of Situated LP, continued

- 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”.

Overview of Semantics of Situated LP, continued

- Conditions:
  - Effectors have only *side* effects: they do not affect operation of the (episode's) inferencing+action engine itself, nor change the (episode's) knowledge base.
  - Sensors are purely informational: they do not have side effects (i.e., any such can be ignored).
  - Timelessness of sensor and effector calls: their results are not dependent on when they are invoked, during a given inferencing episode.
  - "Sensor-safeness": Each rule ensures sufficient (variable) bindings are available to satisfy the binding signature of each sensor associated with any of its body literals – such bindings come from the other, non-sensor literals in the rule body. During overall "testing" of a rule body, sensors needing such bindings can be viewed as invoked after the other literals have been "tested".

Overview: Semantics of Situated LP, Continued More

- Generalizations possible:
  - permit multiple sensors or effectors per predicate.
  - sense functions (or terms) not just predicates.
  - permit sensor priority – i.e, specify the prioritization of the facts that result from a particular sensor .
  - associate sensing with atoms/literals (or terms), but this is reducible to sensing predicates (or functions) – by rewriting of the rules.
- Challenge: error handling info returned from attached procedures

SweetJess [Grosof, 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 on Web

SweetJess: Translating an Effector Statement

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


</damlRuleML: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 extended version of B. Grosof WITS-2001 conference paper
  - "Representing E-Business Rules on the Semantic Web: Situated Courteous Logic Programs in RuleML"
  - Available at http://ebusiness.mit.edu/bgrosof

OPTIONAL DLP SLIDES FOLLOW

Outline/Overview

- Intro and Motivations
  - Semantic Web rules "on top of" ontologies, for Semantic Web Services
  - Need for unified semantics with completeness, consistency $\Rightarrow$ new KR Theory
- A New KR Expressive Class; Mapping between KR's
  - Define DLP $\subseteq$ LP $\cap$ DL $\Rightarrow\Rightarrow$ Enable LP $\cup$ DL
  - Detailed Mapping from DL to LP ; via Horn FOL ; invertible
  - DLP Fragment of DL is an "ontology sub-language" of LP
  - Expressive features completely captured: RDF-Schema plus much more
- Technical Capabilities and Task Scenarios Enabled
  - Primary and secondary Goals achieved for large expressive class
  - Bi-directionality enables efficiency & options in inferencing & authoring
- More Details on the mapping; Examples
- Conclusions, Related Work, Current/Future Directions

Goal: Hybridize KR's for Rules & Ontologies

- Goal: hybridize two important knowledge representations (KR's):
- 1. Description Logic (DL) ontologies cf. OWL
- 2. Logic Program (LP) rules cf. RuleML

- Primary Task Requirement identified in Semantic Web generally, e.g., by RuleML, DAML, W3C efforts:
  - LP rules use DL ontologies: rules "on top of" ontologies
    - Rules mention predicates defined in the DL ontology KB
    - Rules mention individuals that are DL ontology instances

- Secondary task objective identified in DAML:
  - Extend DL with extra LP expressiveness, to define ontologies

Challenges in combining LP rules with DL ontologies for SW

- What Logical KR for combining LP with DL? , with:
  - Power in inferencing? Completeness?
  - Consistency? (needs Completeness/Power)
  - Scaleability in inferencing? Tractability?
  - … Tools? Familiarity by developers for specification?
- Requirement: rules on top of ontologies
- Objective: specify ontologies via rules
- Requirement: scaleability wrt |rules|, |ontologies|

Candidate: First Order Logic

- FOL has practical and expressive drawbacks for union of DL and Rules:
  - Undecidable/Intractable
  - Lacks non-monotonicity and procedural attachments
  - Unfamiliar to mainstream software engineers

Enter… Description Logic Programs (DLP)

Goal: understand relationship between DL and LP/HornFOL as KR's

Insight: the expressive *intersection* is also
a key to the expressive *combination/union*

Analyze this intersection: define DLP

Enable "DLP-Fusion" as approach:
use DLP as bridge to combine knowledge from DL with knowledge from LP

LP as a superset of DLP

- "Full" LP, including with non-monotonicity and procedural attachments, can thus be viewed as including an "ontology sub-language", namely the DLP subset of DL.

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

Overview of DLP KR Features, Continued

- DLP can *largely but partially* capture: most other DL features:
  - Cardinality, functionality of property (or inverse), existential in superclass, universal in subclass.
  - But NOT: (general) negation, disjunction in superclass
- Map also to Relational DBMS (SQL) – which is LP-based.
- *Current Work:* Extend mapping (and inferencing power) via explicit equality, skolemization, integrity constraints.
  - Explicit equality for: cardinality, functionality
  - Skolemization for: existential in superclass, universal in subclass, cardinality
  - Integrity constraints for: negation

More about the Mapping between DL and LP

- Translation simpler to define from DL $\Rightarrow$ LP than DL $\Leftarrow$ LP.

- Translation is actually via Description Horn Logic (DHL), a subset of Datalog Horn FOL (and of DL) (Datalog = no logical functions of arity > 0)
  - Horn LP is a "*f*-weakening" of Horn FOL wrt power in inferencing
    - Conclude only ground facts (– or what's reducible to that).
  - DLP (subset of Horn LP) similarly is f-weakening of DHL
  - Then show formally that DLP is adequate for various DL / LP inferencing tasks that are of most common practical interest
    - (just as Horn LP is adequate wrt most practical inferencing tasks in Horn FOL)
    - Via expressive reduction of various inferencing tasks to other inferencing tasks
  - Additional restriction: equality-free (relaxed in Current Work)

Hybrid DL+LP Task Scenarios/Use-Cases

- 1. Service descriptions combining LP rules and DL ontologies
- 2. Rules for knowledge translation: e.g.,
  - translating/merging ontologies (or rules)

OPTIONAL SLIDES on Misc. FOLLOW

Benjamin Grosof's Contributions to Early Standards Efforts: RuleML, SWSI

- RuleML Initiative
  - Co-Lead, Co-Founder
  - RuleML based largely on IBM CommonRules
  - Designed most key RuleML features
  - RuleML already has basic support for Description LP, Situated LP, Courteous LP
- Active in SWSI, esp. on Rules
  - Member of SWS Language committee
  - Co-chair Industrial Partners forum: several dozen companies
  - Technical challenge: representing service pre- / post-conditions, coherently on top of evolving messiness of WS process models (e.g., BPEL4WS)

The Semantic Web

The 1st generation, the Internet, enabled disparate machines to exchange data.

•The 2nd generation, the World Wide Web, enabled new applications on top of the growing Internet, making enormous amounts of information available, in human-readable form, and allowing a revolution in new applications, environments, and B2C e-commerce.

•The next generation of the net is an "agent-enabled" resource (the "Semantic Web") which makes a huge amount of information available in machine-readable form creating a revolution in new applications, environments, and B2B e-commerce.

…by enabling "agent" communication at a Web-wide scale.