Rules + Ontologies for Semantic Web Services

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

Slides presented at U. Maryland Computer Science Dept. Seminar, 12/06/2002
Hosted by Jim Hendler

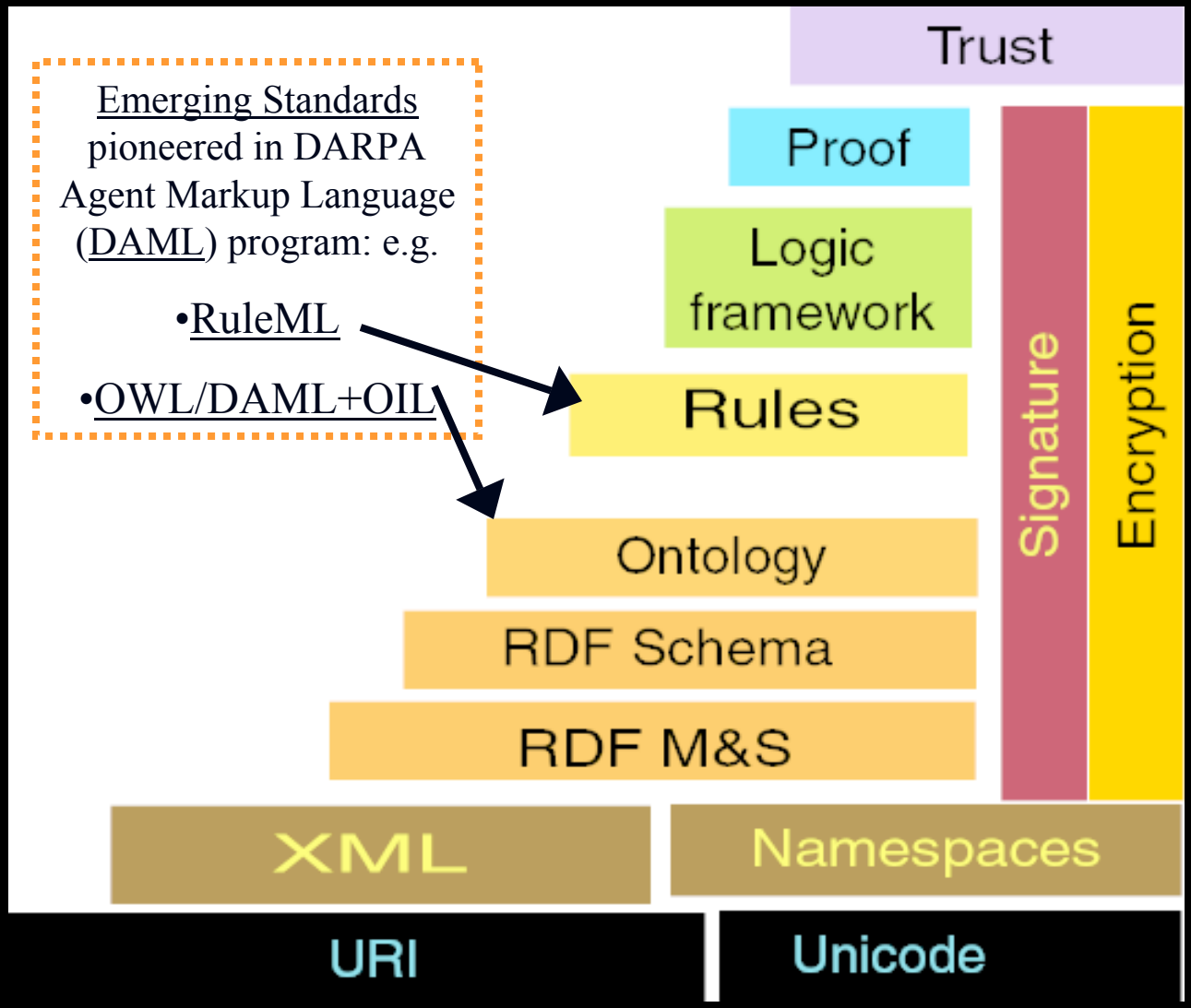
Outline of Talk

- Rules and Semantic Web Services: Overview
  - KR for Agents in E-Business
  - Semantic Web Services
  - RuleML
  - Uses of Rules in SWS
- SweetDeal e-contracting as scenario
  - Rules + Ontologies + Process Descriptions
  - Exception handling
- Description Logic Programs: Overview
  - KR to combine RuleML + OWL

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

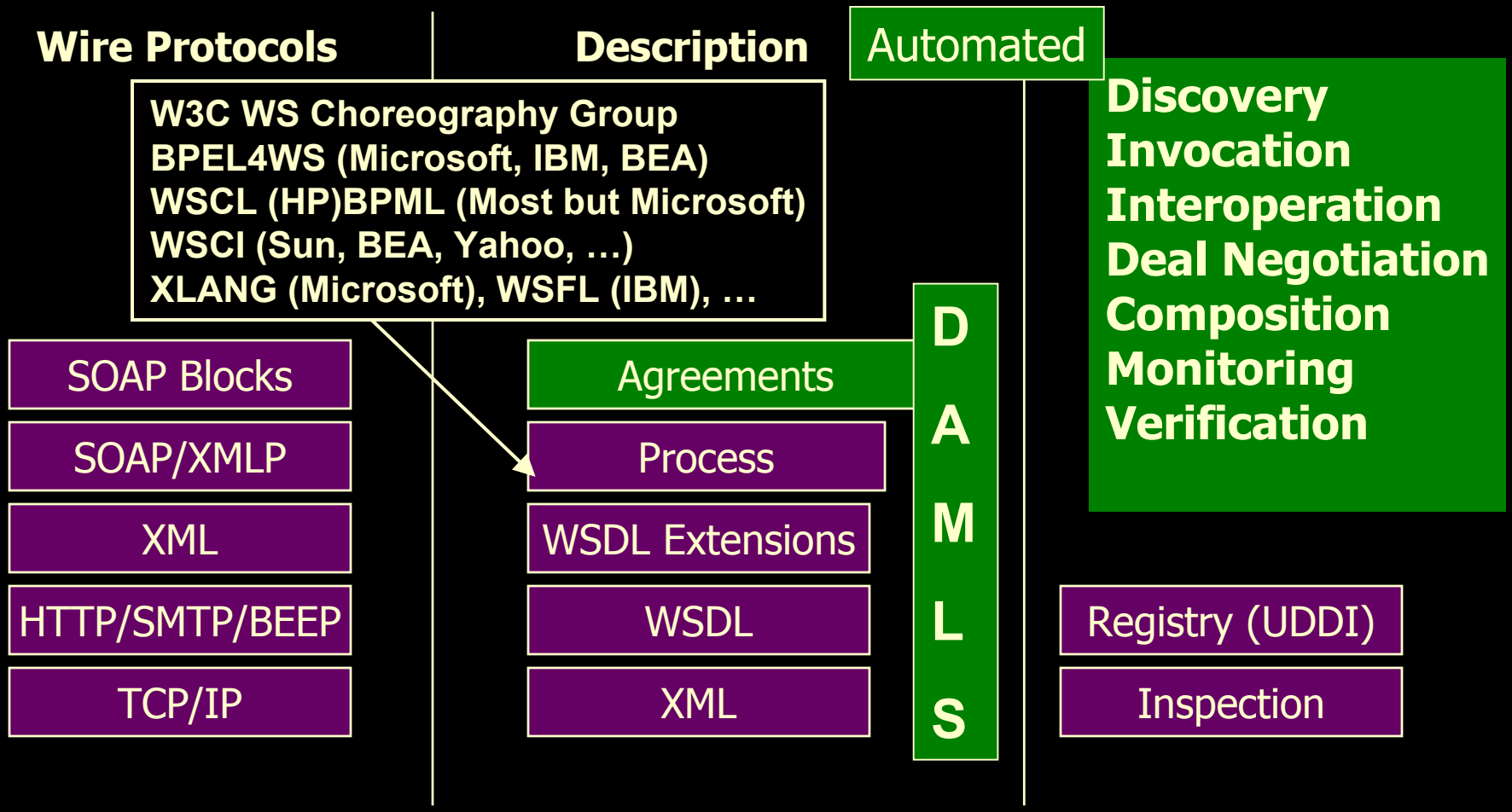
- Bids in auctions and reverse auctions
- Orders in supply chain or B2C
- Contracts/Deals/Proposals/RequestsForProposals
  - prices; product/service descriptions; refunds, contingencies
- Buyer/Seller interests, preferences, capabilities, profiles
  - recommender systems; yellow pages; catalogs
- Ratings, 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, ...*

Semantic Web "Stack": Standardization Steps



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

Current Web Services Standards Stack; Context for Semantic Web Services



[Modification of slide by James Snell (IBM)]

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
  - Integrated knowledge
- (Semantic Web) Services: e.g., infrastructural
  - Knowledge/info/DB integration
  - Inferencing and translation

SWS Tasks at higher layers of WS stack

Automation of:

- Web service discovery

  Find me a shipping service that will transport frozen vegetables from San Francisco to Tuktoyuktuk.

- Web service invocation

  Buy me "Harry Potter and the Philosopher's Stone" at www.amazon.com

- Web service deals, i.e., contracts, and their negotiation

  Propose a price with shipping details for used Dell laptops to Sue Smith.

- Web service selection, composition and interoperation

  Make the travel arrangements for my WWW11 conference.

[Modification of slide also by Sheila McIlraith (Stanford) and David Martin (SRI International)]

SWS Tasks at higher layers of WS stack, continued

- Web service execution monitoring and problem resolution
  Has my book been shipped yet? … [NO!] Obtain recourse.
- Web service simulation and verification
  Suppose we had to cancel the order after 2 days?
- Web service executably specified at "knowledge level"
  The service is performed by running the contract ruleset through a rule engine.

[Modification of slide also by Sheila McIlraith (Stanford) and David Martin (SRI International)]

SWS: Research Players

- DAML Services (DAML-S)
  - service descriptions using ontologies and now rules
- Web Services Mediator Framework (WSMF)
  - EU, Oracle
  - early phase; list of many companies
- @ MIT: Sloan IT group:
  - SweetDeal: e-contracting, policies
  - Extended COIN: financial info integration

Outline of Talk

- Rules and Semantic Web Services: Overview
  - KR for Agents in E-Business
  - Semantic Web Services
  - RuleML
  - Uses of Rules in SWS
- SweetDeal e-contracting as scenario
  - Rules + Ontologies + Process Descriptions
  - Exception handling
- Description Logic Programs: Overview
  - KR to combine RuleML + OWL

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.,
  - Blaze, ILOG, 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.)*

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.

Overview of RuleML Today

- RuleML Initiative (2000--)
  - Dozens of institutions (~35), researchers; esp. in US, EU
  - Mission: Enable semantic exchange of rules/facts between most commercially important rule systems
  - Standards specification: 1st version 2001; basic now fairly stable
  - A number of tools (~12 engines, translators, editors), demo applications
  - Successful Workshop on Rules at ISWC was mostly about RuleML / LP
  - Has now a "home" institutionally in DAML and Joint Committee
    - Discussions well underway to launch W3C, Oasis efforts
- Initial Core: Horn Logic Programs KR

  …Webized (in markup)… and with expressive extensions

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

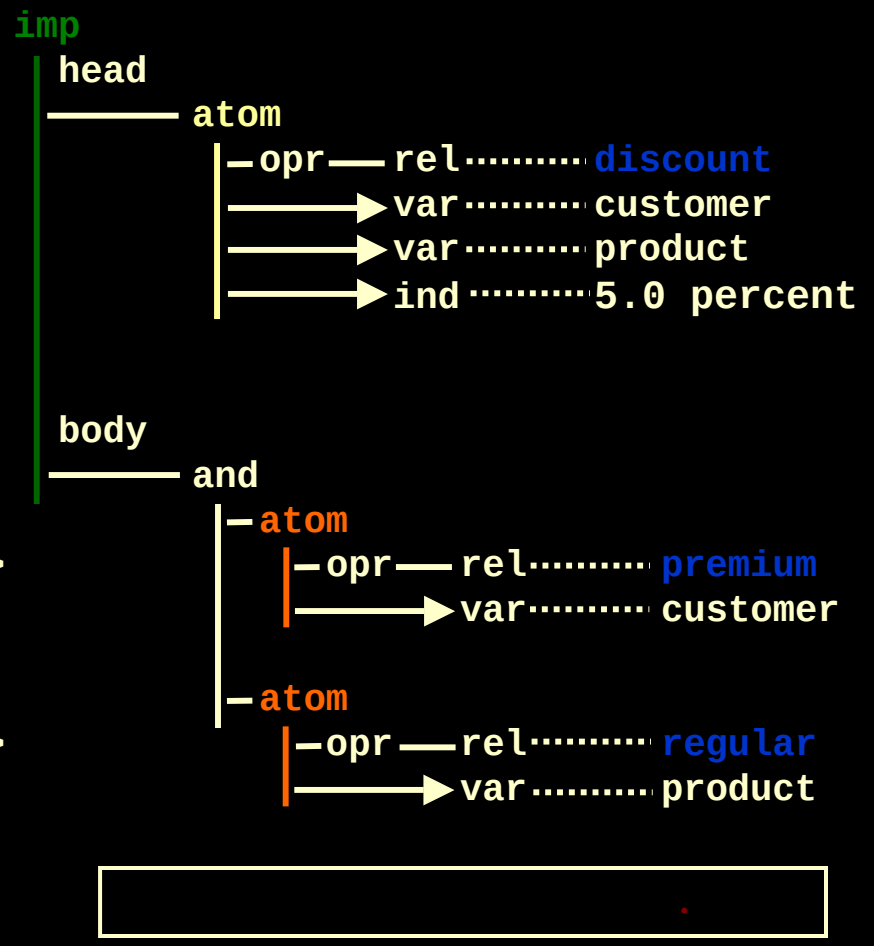
RuleML Example: Markup and Tree

"The **discount** for a *customer* buying a *product* is **5.0 percent**
if the *customer* is **premium** and the *product* is **regular**.',

discount(?customer,?product,"5.0 percent") ← premium(?customer) /\ regular(?product);

Markup Syntax

```
<imp>
  <_head>
    <atom>
     <_opr><rel>discount</rel></_opr>
     <tup><var>customer</var>
          <var>product</var>
          <ind>5.0 percent</ind></tup>
    </atom>
  </_head>
  <_body>
    <and>
      <atom>
        <_opr><rel>premium</rel></_opr>
        <tup><var>customer</var></tup>
      </atom>
      <atom>
        <_opr><rel>regular</rel></_opr>
        <tup><var>product</var></tup>
      </atom>
    </and>
  </_body>
</imp>
```



Technical Approach of RuleML: I

- 1. Expressively: Start with: Datalog Logic Programs *as kernel*
  - Rule := H ← B1 /\ … /\ Bk ; k ≥ 0, H and Bi's are atoms.
    head if body ;
- Declarative LP with model-theoretic semantics
  - forward ("derivation"/ "transformation") and backward ("query") inferencing
- Rationale: captures well a simple shared core among CCI rule sys.
  - Tractable! (if bounded # of logical variables per rule)
- Horn LP -- differences from Horn FOL:
  - Conclusions are a set of ground atoms.
  - Consider Herbrand models only, *in typical usage.*
    - Can extend to permit equalities in rules/conclusions.
  - Rule has non-empty head, *in typical usage.*

Technical Approach of RuleML: II

- 2. Syntax: Permit rules to be labeled -- need names on the Web!
- 3. Syntax: Permit URI's as predicates, functions, etc. (names)
  - namespaces too
- 4. Expressively: Add: extensions cf. established research
  - negation-as-failure (well-founded semantics) -- in body *(stays tractable!)*
    - "Ordinary" LP (cf. declarative pure Prolog)
  - classical negation: limited to head or body atom – syntactic sugar
  - prioritized conflict handling cf. Courteous LP *(stays tractable!)*
    - modular rulesets; modular compiler to Ordinary LP
  - procedural attachments: actions, queries ; cf. Situated LP
  - 1st-order logic type expressiveness cf. Lloyd LP's – syntactic sugar
    - $\vee,\forall,\exists$ in body; $\wedge,\forall$ in head *(stays tractable!)*
  - logical functions (arity > 0)

Technical Approach of RuleML: III

- 5. Expressively: Add: restrictions cf. established R&D
  - E.g., for particular rule systems, e.g., Prolog, Jess, …
    - Also "pass-thru" some info without declarative semantics (pragmatic meta-data)
- 6. Syntax for XML:
  - Family of DTD's/Schemas:
    - a generalization-specialization hierarchy (lattice)
    - define DTD's modularly, using XML entities (~macros)
    - optional header to describe expressive-class using "meta-"ontology
- 7. Syntax: abstract unordered graph syntax (data model)
  - Support RDF as well as XML (avoid reliance on sequence in XML)
  - "Roles" 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
- 8. Syntax: module inclusion: merge rulesets ; import/export
  - URI's name/label knowledge subsets

Tools: SweetRules, including SweetJess

- SweetRules V1 '01: RuleML inferencing and bi-directional translation with equivalent semantics via RuleML, between:
  - XSB Prolog: backward Ordinary Logic Programs (OLP)
  - Smodels: forward OLP
  - IBM CommonRules: forward Situated Courteous LP (SCLP)
  - Knowledge Interchange Format (KIF): First Order Logic interlingua
  - + *Design in principle for:* SQL
    - well-understood in theory literature: as OLP
  - + *Design in principle for:* production (OPS5), ECA
    - Based on Situated extension of LP, piloted in IBM Agent Building Environment '96 for info-workflow applications. Also piloted in EECOMS.
    - BUT: not much other literature/theory to support
    - HENCE motivation to "bring them to the party" … resulting in:
- …V2 '02: adds SweetJess as component:
  - Jess: production (OPS5) , close to ECA
    - popular, open-source, Java: it's useful in particular
    - expressive restriction: "all bound sensors"

SWEET =

Semantic WEb

Enabling Tools

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 [Grosof & 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, Grosof, & Feigenbaum 2000]
  - Financial knowledge integration (ECOIN) [Firat, Madnick, & Grosof 2002]
    - Rule-based translation among contexts / ontologies
    - Equational ontologies
  - Business policies, more generally, e.g., privacy (P3P)

Outline of Talk

- Rules and Semantic Web Services: Overview
  - KR for Agents in E-Business
  - Semantic Web Services
  - RuleML
  - Uses of Rules in SWS
- SweetDeal e-contracting as scenario
  - Rules + Ontologies + Process Descriptions
  - Exception handling
- Description Logic Programs: Overview
  - KR to combine RuleML + OWL

Representing Agent Contracts with Exceptions using XML Rules, Ontologies, and Process Descriptions

Presentation of Paper at the International Workshop on Rule Markup Languages for Business Rules on the Semantic Web, held in conjunction with the 1st International Semantic Web Conference, June 14, 2002, Sardinia, Italy

Benjamin Grosof

MIT Sloan School of Management

bgrosof@mit.edu http://www.mit.edu/~bgrosof/

Terrence Poon

MIT Computer Science

tpoon@alum.mit.edu

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*

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
  - ⇔ 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)*

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.

EECOMS Supply Chain Project: Overview

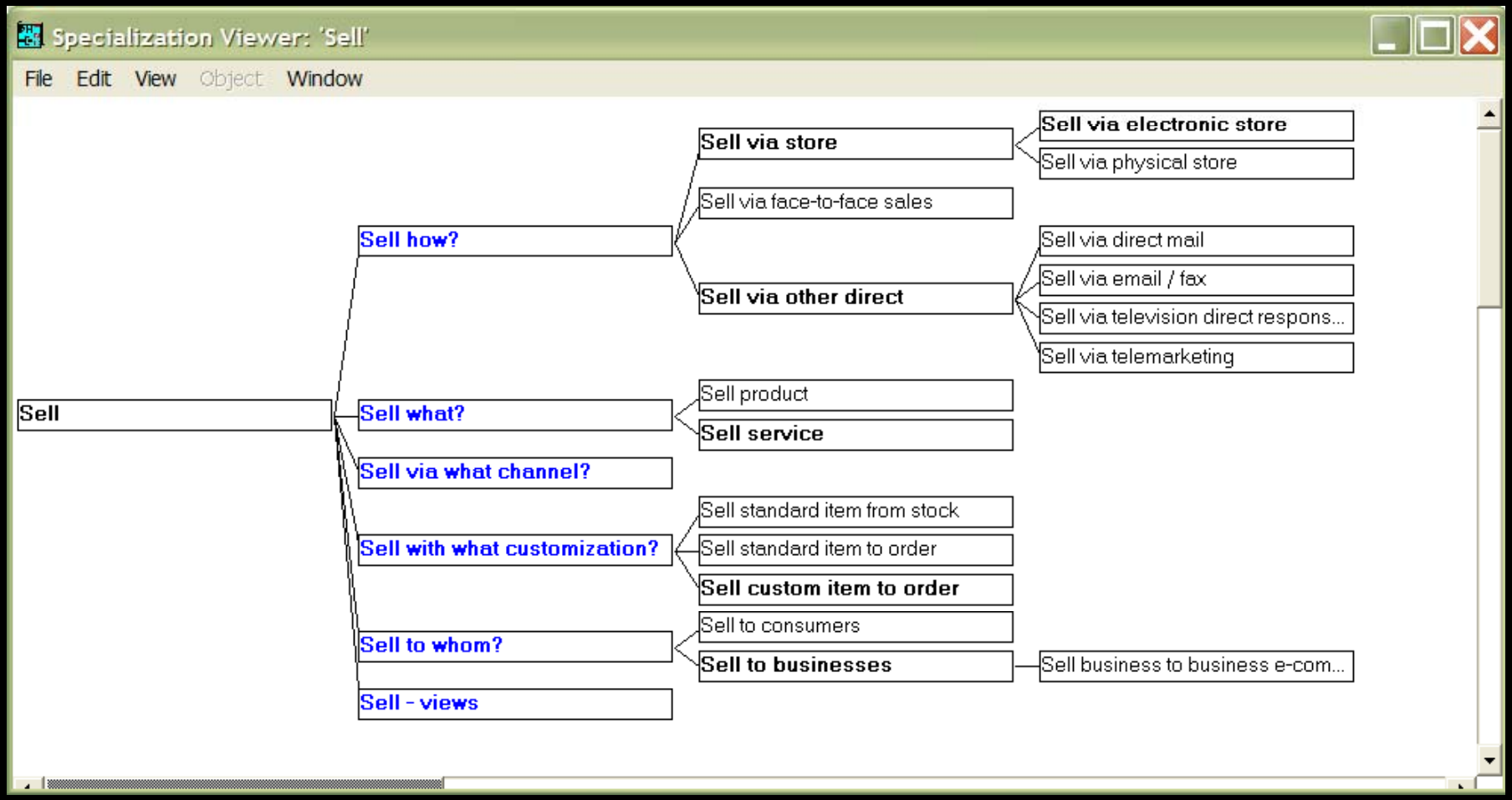
- EECOMS = Extended Enterprise Consortium for Integrated Collaborative Manufacturing Systems. Completed Project: 3/1998 - 2/2001
- Inter-enterprise supply chain integration/collaboration, in manufacturing.
- IBM-led consortium includes Boeing, TRW Consulting, Baan, Vitria, smaller rules & tools co.'s, 3 universities.
- 50%-funded by US government's NIST Advanced Technology Program. $29Million over 3 years (3/98 - 2/01).
- Business Focus: improve **"agility"**: late delivery, plant line breakdown, larger than expected order. React quickly, including modify plans, schedules.
- Technical Focus: **rules and conflict handling** for automated collaboration: contracts, negotiation, authorization, workflow; virtual situation room for human collaborative workflow.
- Follow-on to CIIMPLEX (IBM-led NIST ATP $22M) & challenges it identified. Shares: consortium, scenarios, agent-based approach.

Overview I: *SweetDeal, Exception Handlers, Web Services*

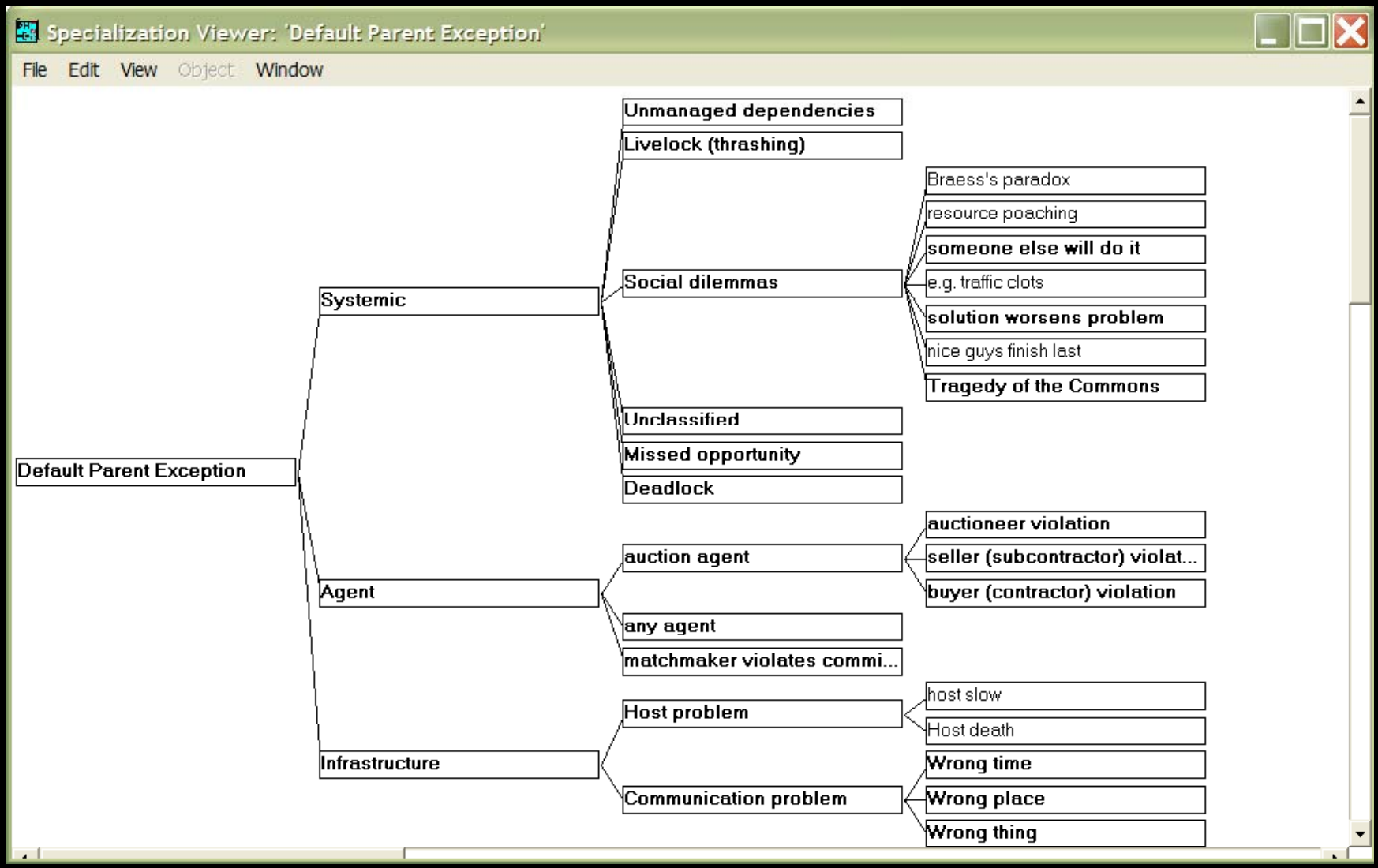
- This work is part of **SweetDeal**: rule-based approach for e-contracting
- Advantages of rule-based: (use Situated Courteous LP KR in RuleML)
  - high level of conceptual abstraction to specify; modularly modifiable; reusable; executable
  - esp. good for specifying *contingent* provisions

- Here, newly extend to include exception handlers:
  - = violations of commitments → invoke business processes
  - more complex behavior
  - good for services, e.g., **deals about Web services**
  - process descriptions whose ontologies are in DAML+OIL
    - drawn from MIT Process Handbook, a previous repository
      - uniquely large & well-used (by industry biz process designers)
  - partially or fully specified by rules (executably)

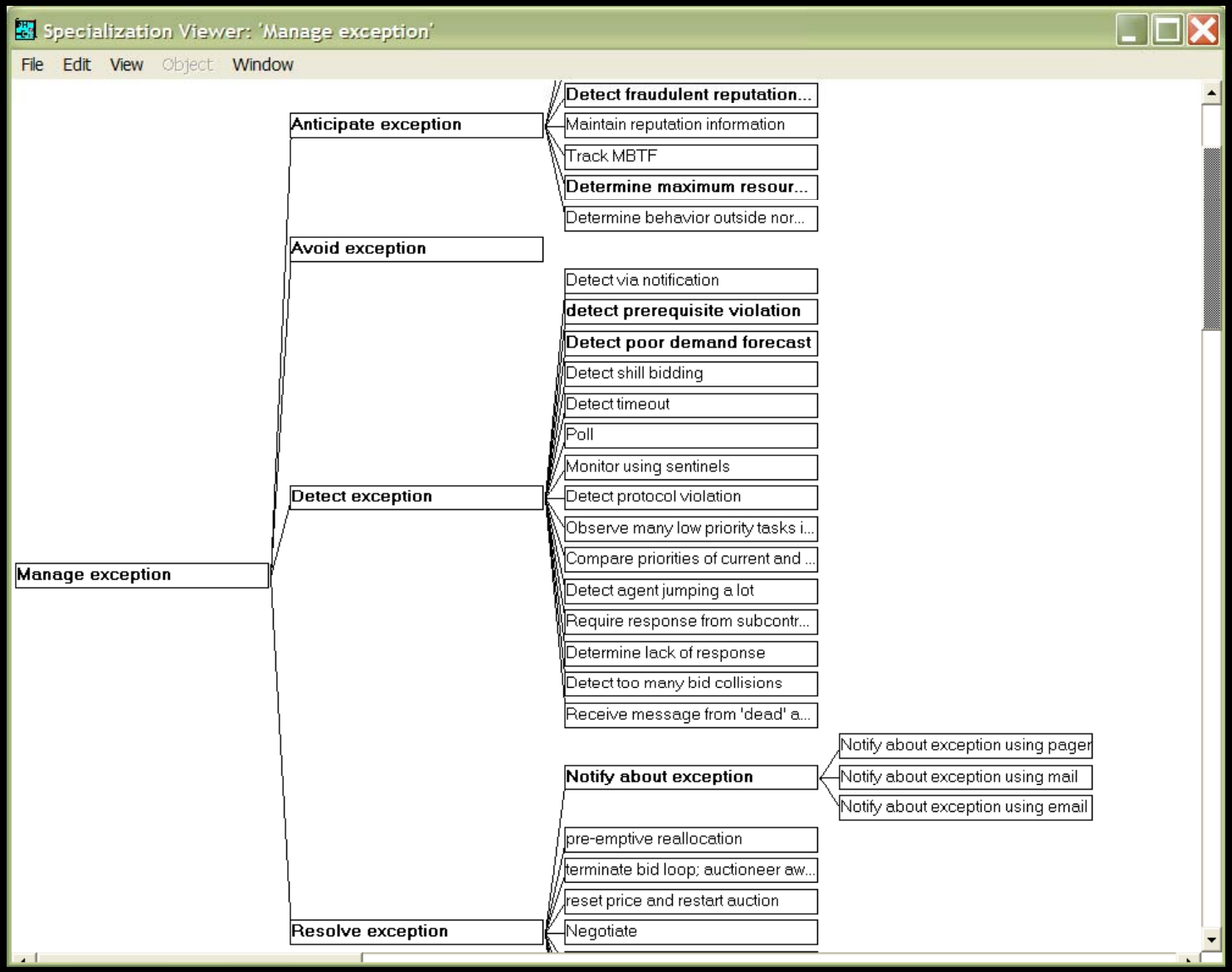
Some Specializations of "Sell" in the MIT Process Handbook (PH)



Some Exceptions in the MIT Process Handbook



Some exception handlers in the MIT Process Handbook



Representing PH Process Ontology in DAML+OIL: Some Main Concepts

```
<daml:Class rdf:ID="Process">
  <rdfs:comment>A process</rdfs:comment>
</daml:Class>
```

Define pr.daml

```
<daml:Class rdf:ID="CoordinationMechanism">
  <rdfs:comment>A process that manages activities between multiple
agents</rdfs:comment>
</daml:Class>

<daml:Class rdf:ID="Exception">
  <rdfs:comment>A violation of an inter-agent commitment</rdfs:comment>
</daml:Class>

<daml:Class rdf:ID="ExceptionHandler">
  <rdfs:subClassOf rdf:resource="#Process"/>
  <rdfs:comment>A process that helps to manage a particular
exception</rdfs:comment>
</daml:Class>
```

Representing PH Process Ontology in DAML+OIL: More

```
<daml:ObjectProperty rdf:ID="hasException">
  <rdfs:comment>Has a potential exception</rdfs:comment>
  <rdfs:domain rdf:resource="#Process" />
  <rdfs:range rdf:resource="#Exception" />
</daml:ObjectProperty>

<daml:ObjectProperty rdf:ID="isHandledBy">
  <rdfs:comment>Can potentially be handled by, in some way </rdfs:comment>
  <rdfs:domain rdf:resource="#Exception" />
  <rdfs:range rdf:resource="#ExceptionHandler" />
</daml:ObjectProperty>

...
<daml:Class rdf:ID="ContractorDoesNotPay">
  <rdfs:subClassOf rdf:resource="#ContractorViolation"/>
  <rdfs:subClassOf>
   <daml:Restriction>
      <daml:onProperty rdf:resource="#isHandledBy"/>
      <daml:hasClass rdf:resource="#ProvideSafeExchangeProtocols"/>
    </daml:Restriction>
  </rdfs:subClassOf>
</daml:Class>
```

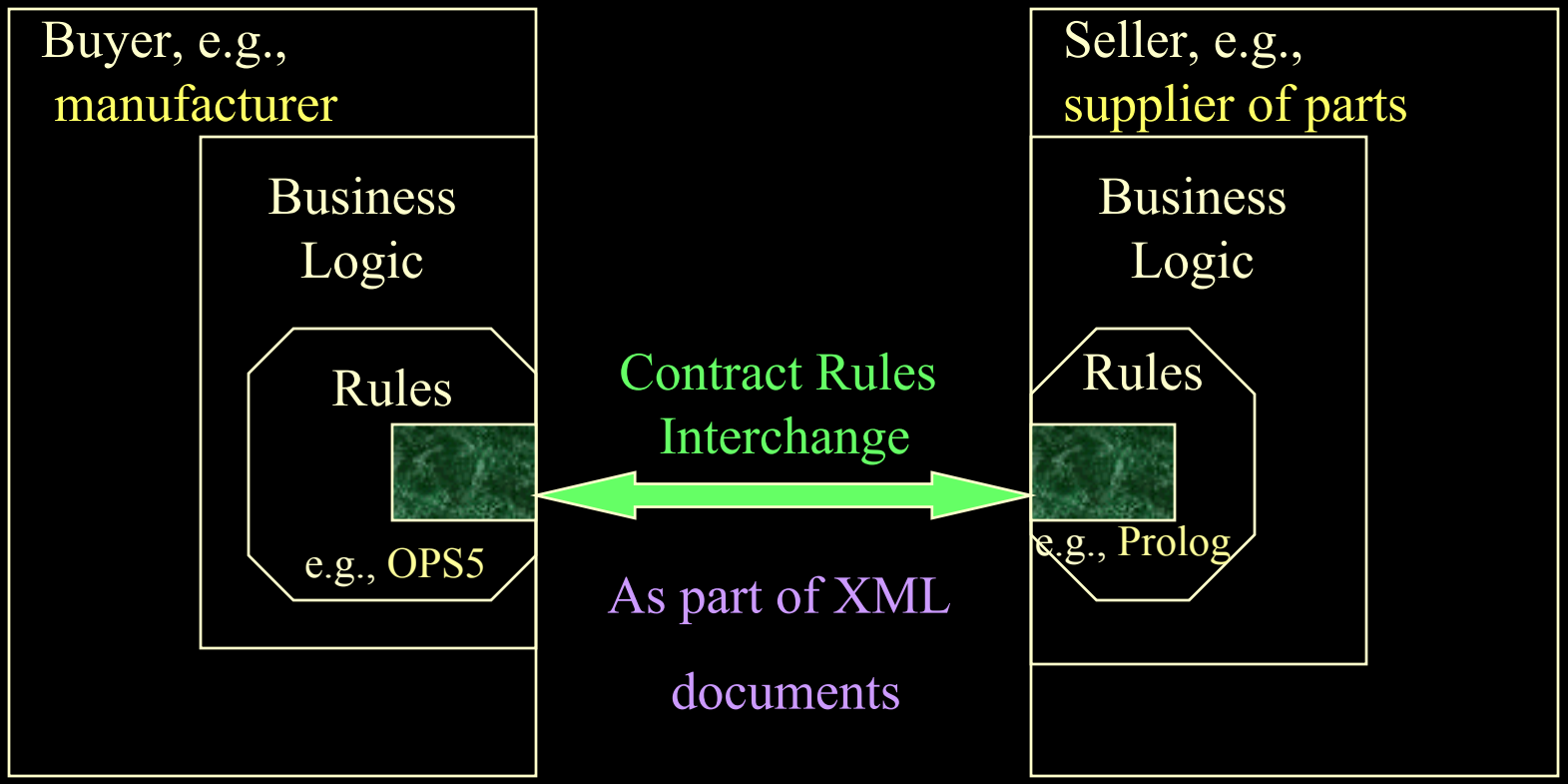
Representing New Contract Ontology in DAML+OIL

```
<daml:Class rdf:ID="Contract">
  <rdfs:subClassOf>
    <daml:Restriction daml:minCardinality="1">
      <daml:onProperty rdf:resource="#specFor"/>
    </daml:Restriction>
  </rdfs:subClassOf>
</daml:Class>

<daml:ObjectProperty rdf:ID="specFor">
  <rdfs:domain rdf:resource="#Contract" />
  <rdfs:range rdf:resource="http://xmlcontracting.org/pr.daml#Process"/>
</daml:ObjectProperty>

<daml:Class rdf:ID="ContractResult"/>

<daml:ObjectProperty rdf:ID="result">
  <rdfs:domain rdf:resource="#Contract" />
  <rdfs:range rdf:resource="#ContractResult" />
</daml:ObjectProperty>
```

Define sd.daml

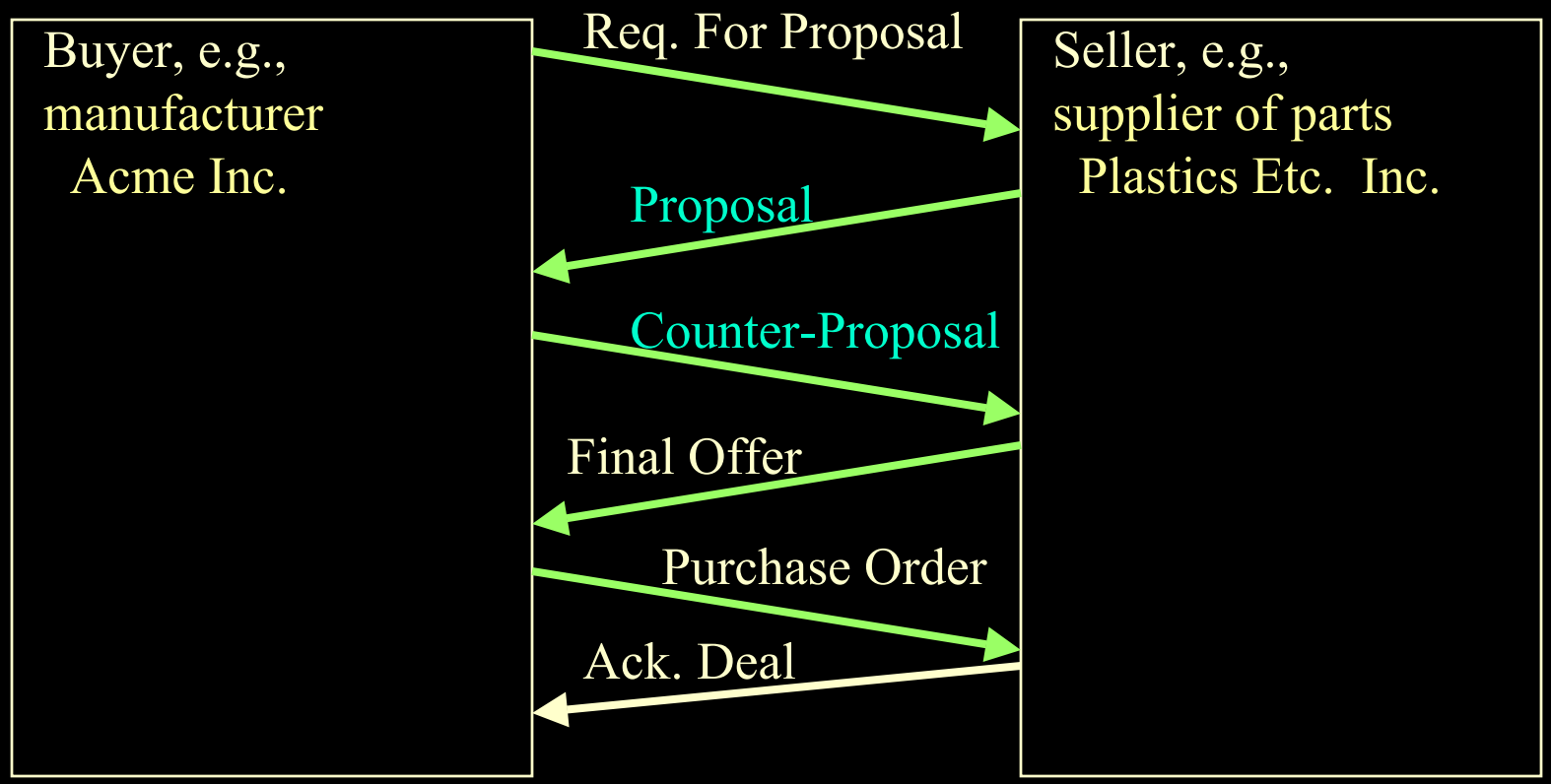
(imports pr.daml)

Contract Rules during Negotiation



Contracting parties NEGOTIATE via shared rules.

Exchange of Rules Content during Negotiation: example



Example Contract Proposal with Exception Handling

Represented using RuleML & DAML+OIL, Process Descriptions

Using concise text syntax

(SCLP textfile format)

for concise human reading

```
buyer(co123,acme);
seller(co123,plastics_etc);
product(co123,plastic425);
price(co123,50);
quantity(co123,100);
http://xmlcontracting.org/sd.daml#Contract(co123);
http://xmlcontracting.org/sd.daml#specFor(co123,co123_process);
http://xmlcontracting.org/sd.daml#BuyWithBilateralNegotiation(co123_process);
http://xmlcontracting.org/sd.daml#result(co123,co123_res);
shippingDate(co123,3); // i.e. 3 days after order placed
// base payment = price * quantity
payment(?R,base,?Payment) <-
  http://xmlcontracting.org/sd.daml#result(co123,?R) AND
  price(co123,?P) AND quantity(co123,?Q) AND
  multiply(?P,?Q,?Payment) ;
```

SCLP TextFile Format for RuleML

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

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

... </drm:andb> </drm:_body>  </drm:imp>
```

drm = namespace for RuleML

Example Contract Proposal, Continued

- 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 Proposal, Continued: lateDeliveryPenalty exception handler module

```
lateDeliveryPenalty_module {
// lateDeliveryPenalty is an instance of PenalizeForContingency
//   (and thus of AvoidException, ExceptionHandler, and Process)
http://xmlcontracting.org/pr.daml#PenalizeForContingency(lateDeliveryPenalty) ;
// lateDeliveryPenalty is intended to avoid exceptions of class
// LateDelivery.
http://xmlcontracting.org/sd.daml#avoidsException(lateDeliveryPenalty,
  http://xmlcontracting.org/pr.daml#LateDelivery);
// penalty = - overdueDays * 200 ; (negative payment by buyer)
<lateDeliveryPenalty_def> payment(?R, contingentPenalty, ?Penalty) <-
  http://xmlcontracting.org/sd.daml#specFor(?CO,?PI) AND
  http://xmlcontracting.org/pr.daml#hasException(?PI,?EI) AND
  http://xmlcontracting.org/pr.daml#isHandledBy(?EI,lateDeliveryPenalty) AND
  http://xmlcontracting.org/sd.daml#result(?CO,?R) AND
  http://xmlcontracting.org/sd.daml#exceptionOccurred(?R,?EI) AND
  shippingDate(?CO,?CODate) AND shippingDate(?R,?RDate) AND
  subtract(?RDate,?CODate,?OverdueDays) AND
  multiply(?OverdueDays, 200, ?Res1) AND multiply(?Res1, -1, ?Penalty) ;
}
<lateDeliveryPenaltyHandlesIt(e1)> // specify lateDeliveryPenalty as a handler for e1
 http://xmlcontracting.org/pr.daml#isHandledBy(e1,lateDeliveryPenalty);
```

Example, Continued: Counter-Proposal

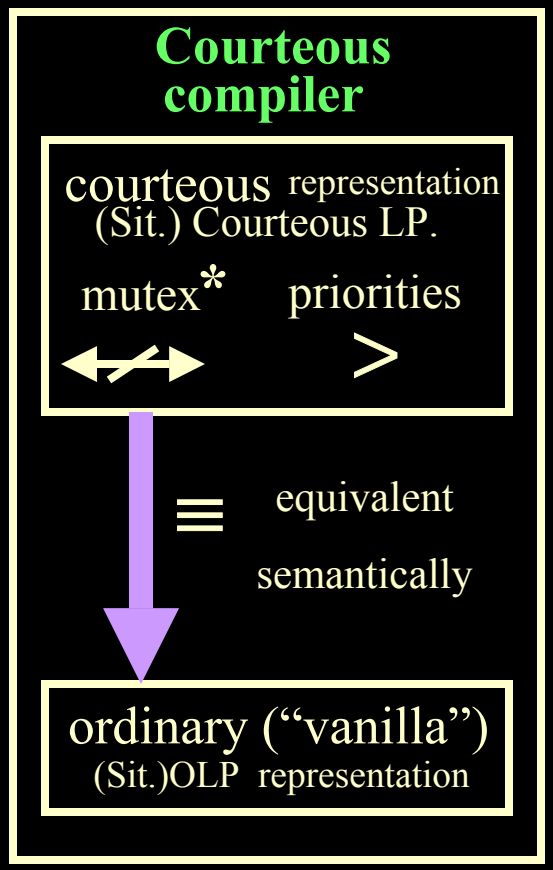
- Seller modifies the draft contract (it's a *negotiation*!)
- Simply adds* another rule module to specify:
  - lateDeliveryRiskPayment as exception handler
    - lump-sum in advance, based on average lateness
      - instead of proportional to actual lateness
  - higher-priority for that module than 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

Example Counter-Proposal's ruleset's prioritized conflict handling

```
// priority specified via syntactically reserved "overrides" predicate

 overrides(lateDeliveryRiskPaymentHandlesIt(e1),
           lateDeliveryPenaltyHandlesIt(e1) ) ;



// There is at most one avoid handler for a given exception instance.
// Consistency is enforced wrt this "mutex" integrity constraint.

  MUTEX
    http://xmlcontracting.org/pr.daml#isHandledBy(?EI, ?EHandler1) AND
    http://xmlcontracting.org/pr.daml#isHandledBy(?EI, ?Ehandler2)
GIVEN
    http://xmlcontracting.org/sd.daml#AvoidException(?Ehandler1) AND
    http://xmlcontracting.org/sd.daml#AvoidException(?Ehandler2) ;
```

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 $\rightarrow$ (v+2)

Preserves ontology.

Plus extra predicates for

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

* classical negation too

Overview II: More New Contributions

- 1. Combine Situated Courteous Logic Programs (SCLP) case of RuleML with DAML+OIL; i.e., SCLP + Description Logic (DL)
  - rules "on top of" ontologies
  - show how and why to do as representational style (KR, syntax)
    - DAML+OIL class or property used as predicate in RuleML
      - heavily exploit feature of RuleML that predicate can be a URI
    - in progress: deeper semantics of the combination
  - more generally, 1st combo of nonmon RuleML / SCLP with DL
  - 1st combo of nonmon rules + DL (also Antoniou, independently)
- 2. Combine further with process descriptions
- 1st substantial practical e-business application domain scenario for 1., 2.
- Point of convergence between Semantic Web and Web Services
- 1st: approach to automate MIT Process Handbook using: a) XML ; b) powerful KR (but encoded only small fraction of its content so far!)
  - underline incapacity of DAML+OIL to represent default inheritance

Related Work: Ours & Theirs

- Previous Work on SweetDeal
  - Rule-based Approach; Requirements analysis for SW rule KR for e-contracting & e-business
  - ContractBot + AuctionBot: negotiation, auction configuration
  - EECOMS $29Million industry pilot on manufacturing supply chain: negotiation
- Recent Work on SweetDeal:
  - Contract fragments, with queryable repository
    - modules inclusion & naming: new technical aspects for RuleML
  - Contract-proposer "market" agent: GUI, with rule-based backend; semi-automated creation, modification, communication, inferencing
- Prototype running; publicly available soon
- Future Directions: Larger Projects:
  - Rule KR Technologies, esp. for Semantic Web Services
    - Current work: theory of {Description Logic ∪ LP}
  - Business Applications of Semantic Web Services
    - Deal Level of SW/Services; B2B, policies, supply chain, finance

DAML-S, WSMF

Antoniou '02

Outline of Talk

- Rules and Semantic Web Services: Overview
  - KR for Agents in E-Business
  - Semantic Web Services
  - RuleML
  - Uses of Rules in SWS
- SweetDeal e-contracting as scenario
  - Rules + Ontologies + Process Descriptions
  - Exception handling
- Description Logic Programs: Overview
  - KR to combine RuleML + OWL

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

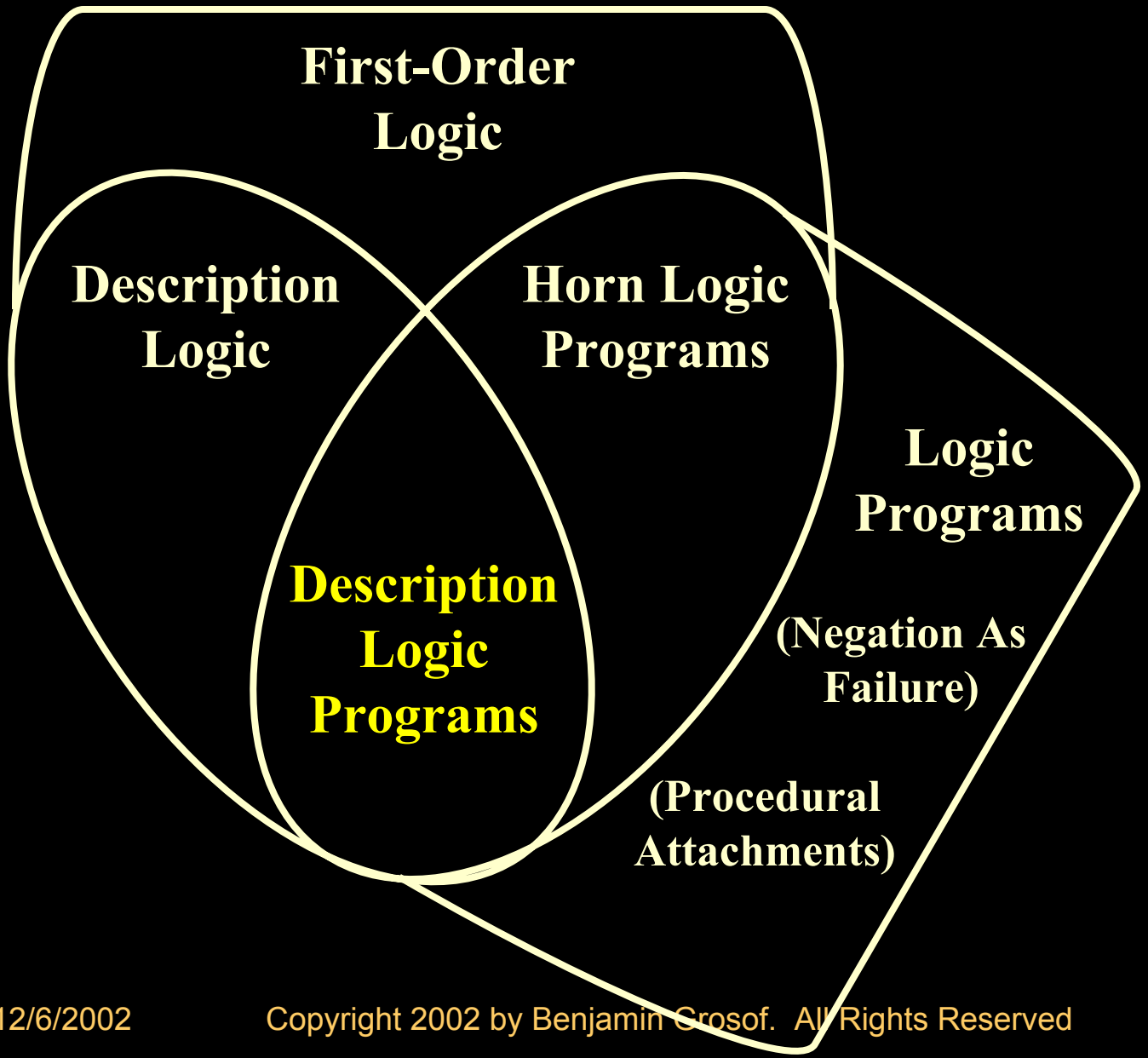
- Rules often good to executably specify service process models
  - e.g., business process automation using procedural attachments to perform side-effectful/state-changing actions ("effectors" triggered by drawing of conclusions)
  - e.g., rules obtain info via procedural attachments ("sensors" test 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

Challenges in combining SW rules with ontologies

- Logical KR for combining RuleML with OWL?
- Completeness?
- Consistency?
- Tractability?

- Goal: rules on top of ontologies
- Goal: specify ontologies via rules
- Goal: scaleability wrt |rules|, |ontologies|

Venn Diagram: Expressive Overlaps among KR's



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 Features

- Essentially, DLP captures RDFS subset of DL -- plus a bit more.
- RDFS subset of DL permits the following statements:
  - Class C is Subclass of class D.
  - Domain of property P is class C.
  - Range restriction on property P is class D.
  - Property P is Subproperty of property Q.
  - a is an instance of class C.
  - (a,b) is an instance of property P.
- DLP also captures:
  - Using the Intersection connective (conjunction) in class descriptions
  - Stating that a property P is Transitive.
  - Stating that a property P is Symmetric.
- DLP can *partially* capture: most other DL features.
- Relevant technical issues in LP:
  - treatment of equality, e.g., uniqueness of names.

Overview of DLP Features, cont'd

- More details on other DL features:
  - Universal in superclass part of subclassof statement
  - Existential in subclass part of subclassof statement
  - Union (disjunction) in subclass part of subclassof statement
- More via skolemization, negation, integrity constraints
- Essentially, DLP captures RDFS subset of DL -- plus significantly more.

Benefits: What DLP Enables, in Principle

- LP rules "on top of" DL ontologies.
  - E.g., LP imports DLP ontologies, with completeness & consistency
- 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

- Translation of LP conclusions to DL.
- Translation of DL conclusions to LP.

- Facilitate rule-based mapping between ontologies / "contexts"

Outline of Talk

- Rules and Semantic Web Services: Overview
  - KR for Agents in E-Business
  - Semantic Web Services
  - RuleML
  - Uses of Rules in SWS
- SweetDeal e-contracting as scenario
  - Rules + Ontologies + Process Descriptions
  - Exception handling
- Description Logic Programs: Overview
  - KR to combine RuleML + OWL

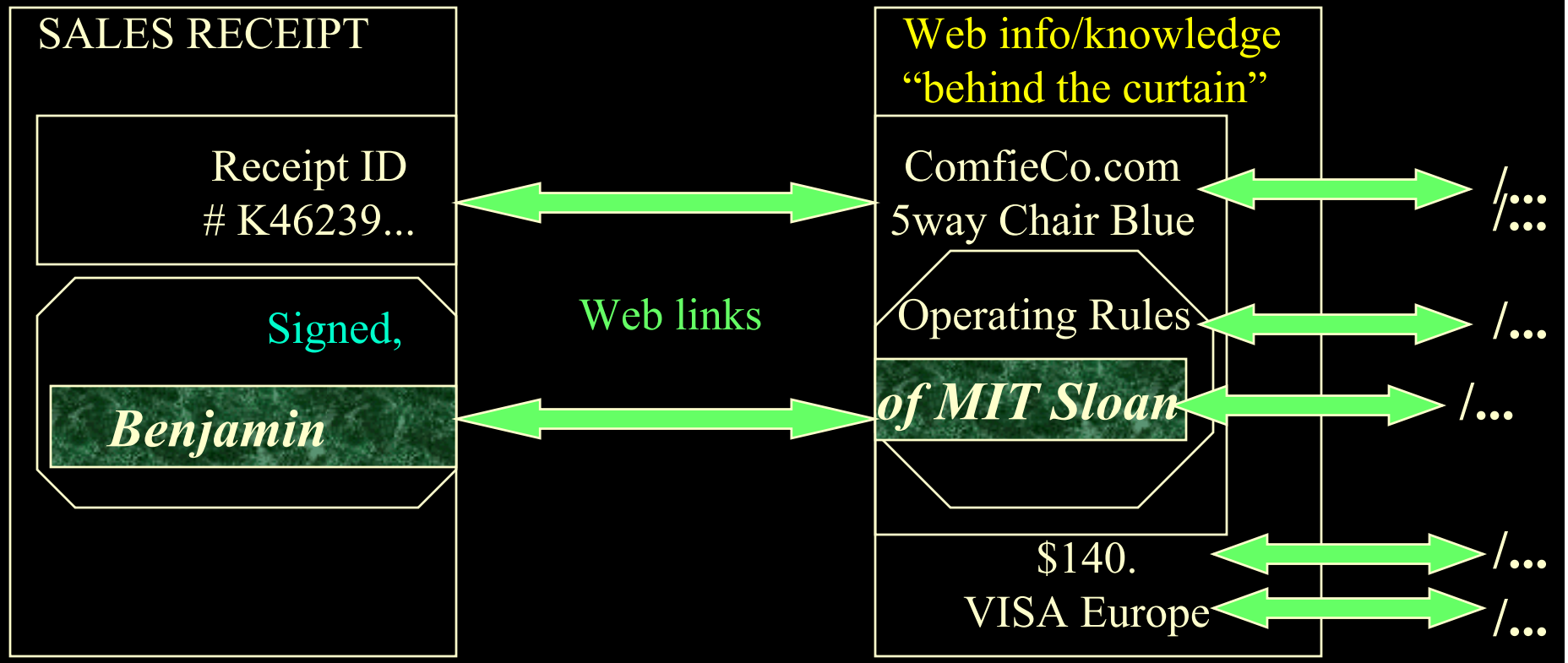
Acknowledgements: Other Contributors

- RuleML design of syntax and Webizing:
  - Harold Boley (DFKI/NRC) and Said Tabet (Nisus)
- SweetRules and SweetJess:
  - Mahesh Gandhe (student UMBC, now IBM), Tim Finin (UMBC), Youssef Kabbaj (student MIT)
- SweetDeal:
  - Terrence Poon (student MIT, now Oracle)
- Description Logic Programs:
  - Ian Horrocks (U. Manchester)
- Semantic Web Services overview:
  - Sheila McIlraith (Stanford), David Martin (SRI)
- ECOIN:
  - Aykut Firat (student MIT), Stuart Madnick (MIT)

OPTIONAL SLIDES FOLLOW

Looks Simple To Start... then Gets Interestingly Precise

A Vision/Approach of what Web & Agents enable



Web is becoming XML → the Semantic Web

- XML (vs. HTML) offers much greater capabilities for structured detailed descriptions that can be processed automatically.
  - Eases application development effort for assimilation of data in inter-enterprise interchange
  - A suite of open standards both current and emerging
  - … including for knowledge-level SEMANTICS
- *Soon, Agents will Talk according to these standards…*
  - ∴ potential to revolutionize *interactivity* in Web marketplaces
    - B2B, …
- HTML itself is becoming XHTML: just a special case of XML

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*

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

Technology Research Directions: KR for Agent Communication

- Aims:
  - deeper reasoning intra-agent
    - "understanding" 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

SW Stack: Acronym Expansion

- W3C = World Wide Web Consortium: umbrella standards body
- XML-S: XML Schema, i.e., basic XML spec
- RDF: Resource Description Framework:
  - W3C Working Group
  - Labelled directed graph syntax
  - Good for building knowledge representation on top of: simpler, more powerful than basic XML
  - M&S = Model and Syntax
  - RDF Schema = extension: simple class hierarchies
- Ontology = formally defined vocabulary & axioms esp. about class hierarchy, generalizes Entity-Relationship models
  - OWL = W3C Web Ontologies Working Language
  - … based closely on DAML+OIL (uses Description Logic KR)

SW: Research Players

- US: DARPA Agent Markup Language Program (DAML) program
- EU: OntoWeb program
- @MIT:
  - Sloan IT group: Grosof, Madnick, Firat, Klein, *et al*
  - LCS / W3C advanced-dev.: Berners-Lee, *et al*
- Number of companies:
  - HP, IBM, Adobe, Oracle, …

Why Standardize Rules Now?

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

Going Beyond KIF

- KIF is KR Ag. Comm. Lang.'s point of departure:
  - Intent: general-knowledge interlingua.
  - Emerging standard, in ANSI commmittee.
  - 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.

Overview of RuleML Today, Continued

- Fully Declarative KR (not simply Prolog!)
  - Well-established logic with model theory
  - Available algorithms, implementations
  - Close connection to relational DB's; core SQL is Horn LP
  - *See [Baral & Gelfond '94] for good survey on declarative LP.*
- Abstract graph syntax
  - 1st encoded in XML…
  - … then RDF (draft), … then DAML+OIL (draft)
- Expressive Extensions incrementally, esp. already:
  - Non-monotonicity: Negation as failure; Courteous priorities
  - Procedural Attachments: Situated actions/effecting, tests/sensing
  - *In-progress:* Events cf. OPS5/Event-Condition-Action

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 binding-signature.
- Enable dynamic loading and remote 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.

Criteria for Rule Representation, e.g., for Contracts

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. } OLP
  - essential feature in commercially important rule systems.
- Prioritized conflict handling. → Courteous
- Ease of parsing.
- Integration into Web-world software engineering. } → XML
- Procedural attachments. → Situated

More Current & Future Work

- Representing Default Inheritance in Ontologies
- Relating to Semantic Web Services elements:
  - SOAP, UDDI, WSDL
  - DAML-S, WSMF; WSFL/Xlang, …
  - E-Business/Agent Messaging, e.g., ebXML, UBL
- Relation to Legal aspects of Contracting ; Legal XML

MORE OPTIONAL SLIDES FOLLOW

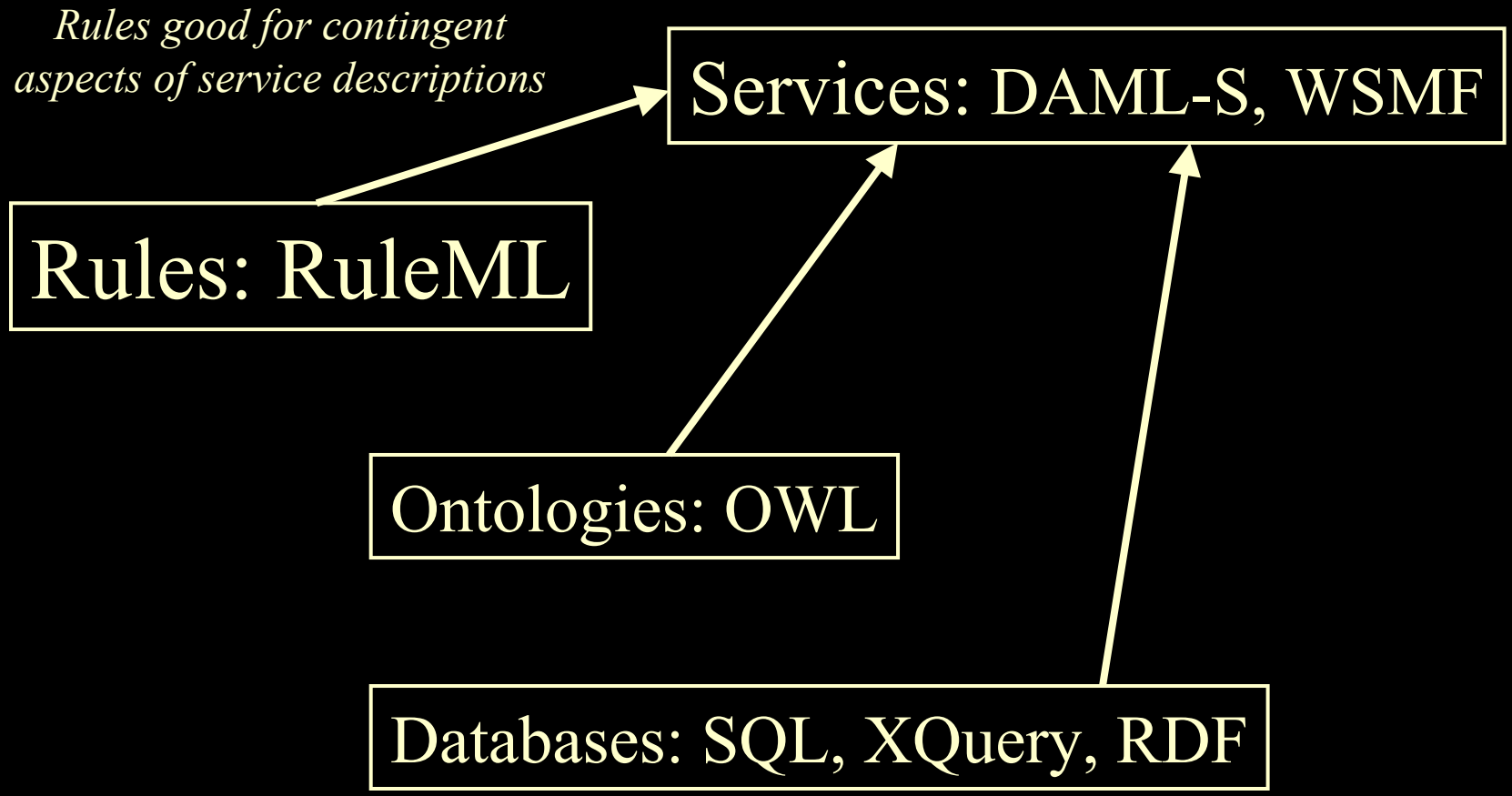
SW: Standards Players

- US-EU Joint Committee:
  - Early standards drafting
  - 1st focus: ontologies: DAML+OIL → W3C OWL
  - 2nd focus (current): rules: RuleML
- W3C: Semantic Web Activity
- Oasis: various incl. Security
- New efforts (currently in formation):
  - US-EU Joint Committee on Semantic Web Services
  - ISO: CommonLogic first-order logic (formerly KIF)

SW-Related: XML Query Languages

- Goals
  - a data model for generic "natively" XML documents,
  - a set of query operators on that data model,
  - and a query language based on these query operators
  - Queries operate on single documents or fixed collections of documents.
- What SQL is for relational databases, XML Query languages are for collections of XML docs.
- There is a standard: W3C's XML Query Working Group
  - (W3C = World Wide Web Consortium)
- Oracle, IBM, Microsoft, etc. already support some
  - Not taking off quickly – complex spec

Vision: Semantic Web and Web Services Use DB's, Ontologies, and Rule Systems



Web Service -- definition

- *(For purposes of this talk:)*

- A procedure/method that is invoked through a Web protocol interface, typically with XML inputs and outputs

WS Stack: some Acronym Expansion

- SOAP = simple protocol for XML messaging
- WSDL = protocol for basic invocation of Web Services, their input and output types in XML
- Choreography = higher-level application interaction protocols in terms of sequences of exchanged message types, contingent branching
  - Currently morphing into a W3C activity

- *Overall: lots of proprietary jockeying and de-facto mode testing/pressuring of the open-consortial standards bodies (e.g., of W3C) "riding the tiger"*

WS Players

- Basically, all the major software vendors
  - Biggies: Microsoft, IBM, Oracle, Sun, SAP, …
  - Webserver/XML ebiz space: BEA, CommerceOne, Ariba, …
  - Niche offerings, e.g., travel agent services, weather, …
- Standards bodies: W3C; Oasis incl. Security
- Overall: lots of proprietary jockeying and *de-facto* mode testing/pressuring of the open-consortial standards bodies (e.g., of W3C) "riding the tiger"
- Still low-level in terms of application abstractions

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

SW Early Adopters: Areas by Industry or Task

- Early SW techniques already in use:
  - e-contracting, supply chain incl. procurement
    - manufacturing, e.g. computer/electronics (RosettaNet), automotive (Covisint),
    - EECOMS pilot (Boeing, IBM, TRW, Baan)
    - office supplies (OBI)
    - retailing: shopbots and salesbots: comparisons, recommendations
    - extensive standards activity: Oasis ebXML, XML eContracts, UN UBL, EDI

SW Early Adopters: Areas by Industry or Task

- *Continued:* Early SW techniques already in use:
  - cyber goods:
    - financial services (rules; onto translation)
    - travel "agency", i.e.: tickets, packages (AI smarts for scheduling)
  - military intelligence (e.g., funded DAML)

SW Early Adopters: Areas by Industry or Task

- Still in research or early standardization, mainly:
  - e-contracting:
    - auctions
    - construction
    - insurance, risk management
    - SME's, spontaneity
    - international
    - distribution
  - authorization and security policies
  - business policies

SW Early Adopters: Areas by Industry or Task

- *Continued:* Still in research or early standardization, mainly:
  - reputations, ratings
  - legal/regulatory: forms, dispute resolution ; Oasis Legal XML
  - computer games: massive multi-player
  - question-answering
  - news filtering, e.g., financial
  - knowledge management
  - advertising
  - bioinformatics, scientific Grid
- Others? ? ? ? ? ?

FOR MORE INFO -- on author's webpage

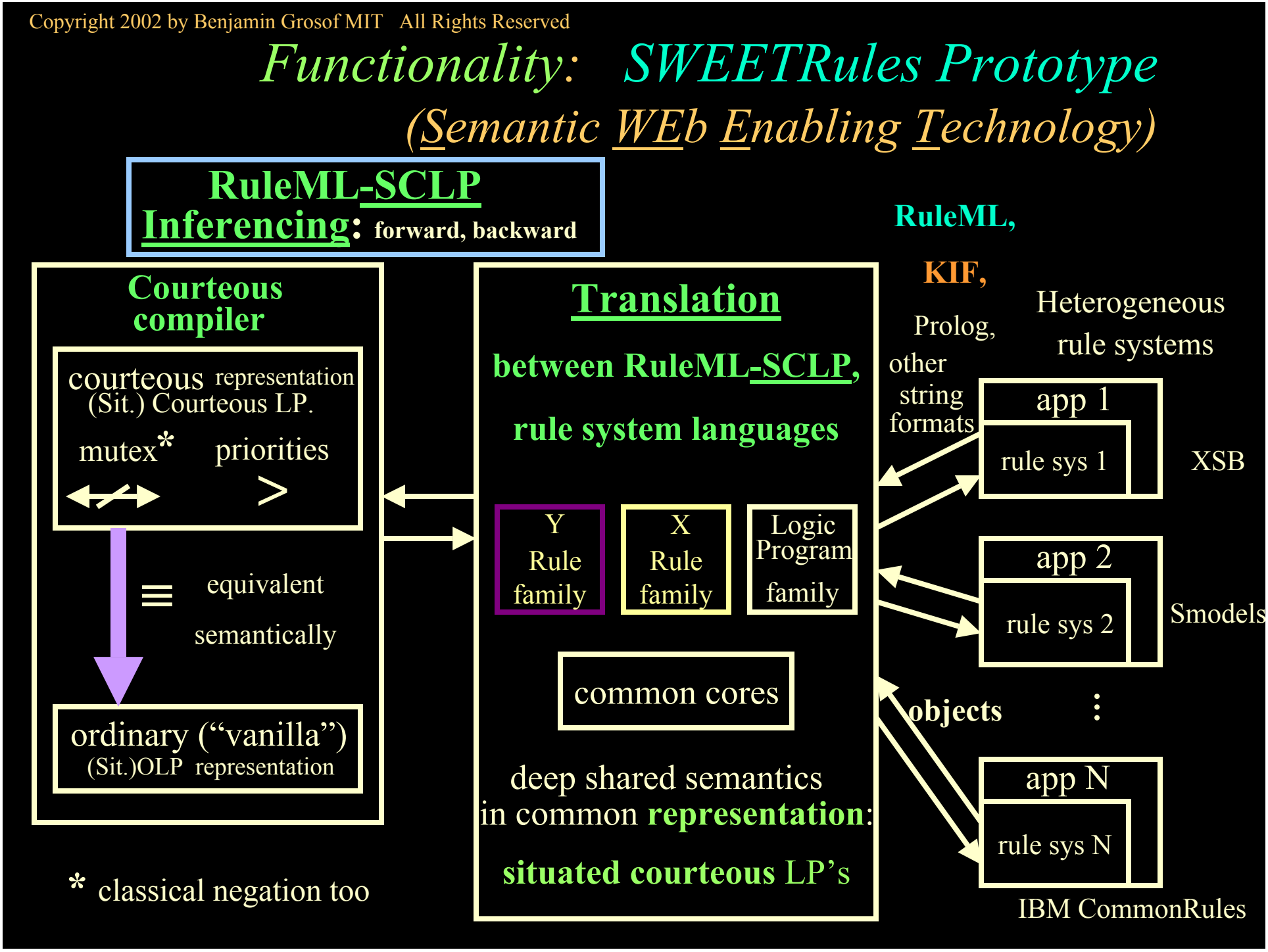
- At http://ebusiness.mit.edu/bgrosof:
  - Recent SweetDeal paper and talk, from Intl. Sem. Web. Conf. (2002) Workshop on Rules; and earlier papers
    - …/#SweetDealExceptions
  - RuleML Overviews
    - …/#RuleML, esp. 10/29/02 Joint Committee intro talk
  - Description Logic Programs paper and talk (discusses deeper technical approach to combining rules and ontologies)
    - …/#DLP
  - SWS Project overviews
    - …/#Overview and …/#Projects

FOR MORE INFO - resources on SW,WS, SWS

- SWS overview: http://ebusiness.mit.edu/#SWS
- DAML http://www.daml.org ; esp. DAML-S …/services
- WSMF http://informatik.uibk.ac.at/users/c70385/wese/publications.html
- W3C SW: http://www.w3.org/2001/sw -> charter, RDF, WebOnt
- Also at W3C: WSDL, Xquery, …
- Web Services – Interoperability http://www.ws-i.org
- Oasis XML standards body http://www.oasis-open.org
- RuleML main site (major editing in progress): http://www.ruleml.org
- And:
  - XML world: the Cover pages http://xml.coverpages.org
  - A SW community portal http://www.semanticweb.org

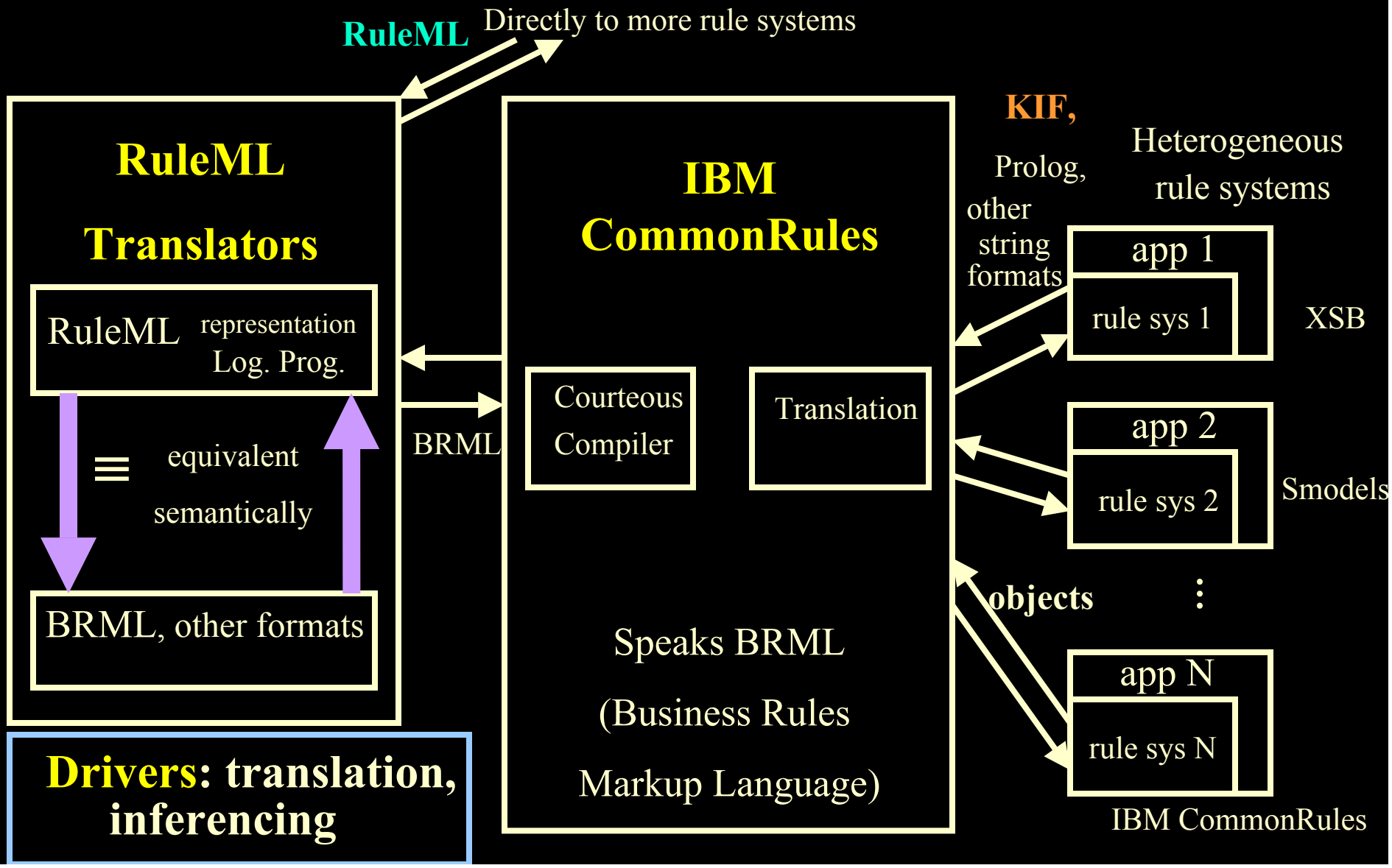
Functionality: SWEETRules Prototype

(Semantic WEb Enabling Technology)



Dec.-2001 Architecture: SWEETRules Prototype

(Semantic WEb Enabling Technology)



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., $\bot \leftarrow$ discount(?product,5%) $\wedge$ discount(?product,10%) .
  - E.g., $\bot \leftarrow$ loyalCustomer(?c,?s) $\wedge$ premiereCustomer(?c,?s) .
  - Permit classical-negation of atoms: $\neg$p means p has truth value *false*
    - implicitly, $\bot \leftarrow p \wedge \neg p$ for every atom p.
- **Priorities** between rules: partially-ordered.
  - Represent priorities via reserved predicate that compares rule labels:
    - overrides(rule1,rule2) means rule1 is higher-priority than rule2.
    - Each rule optionally has a rule label whose form is a functional term.
    - overrides can be reasoned about, just like any other predicate.

Priorities are available and useful

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

Courteous Compiler

- Transformer compiles a courteous LP into an ordinary LP.
- A radically innovative approach in rules representation.
- "Compiles away" conflict, as modular add-on to rule system X's
  - inferencing
  - specification
- Enables courteous features to be added to, or implemented in, a variety of rule systems.
- Tractable: O(n^3). Incremental.

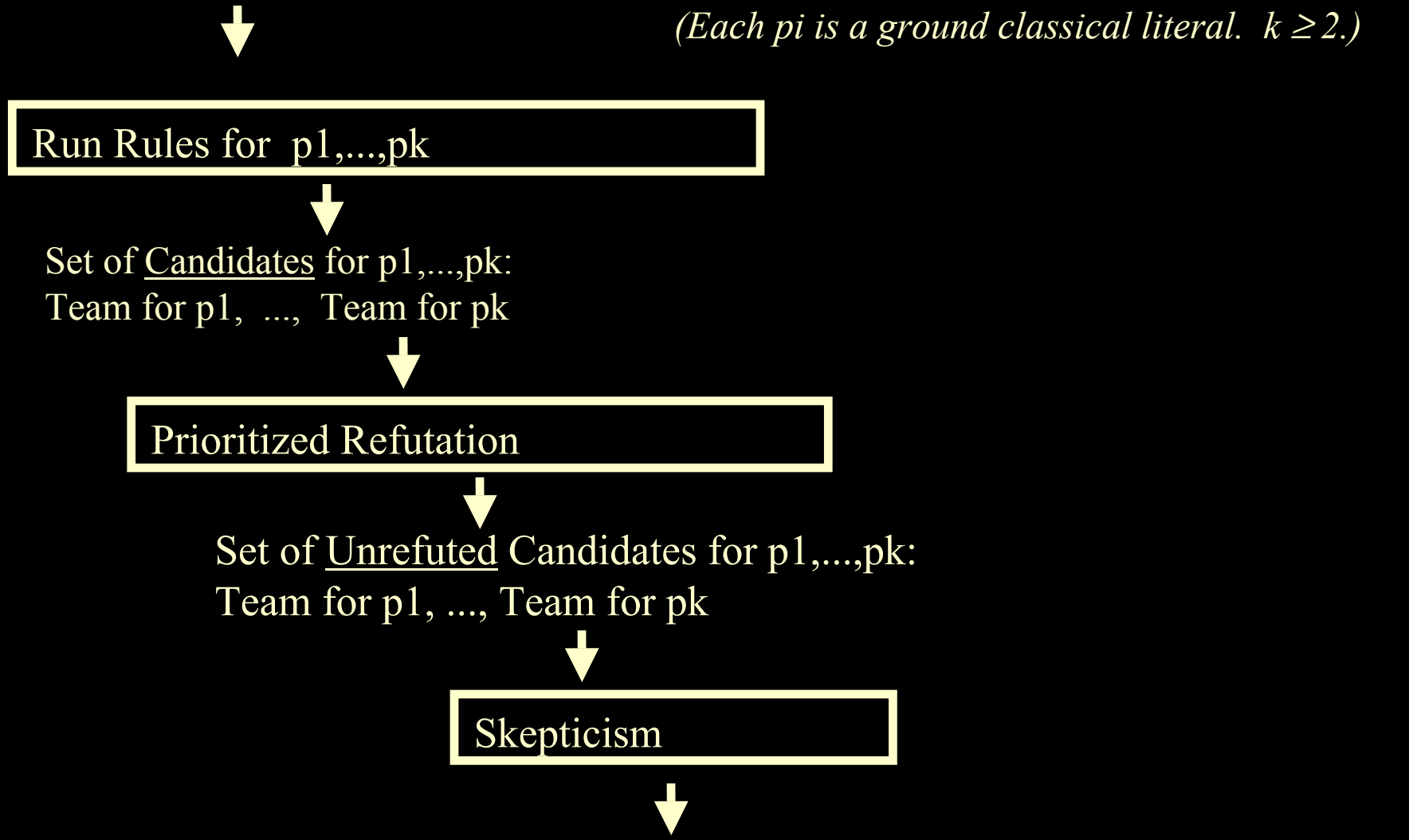
Courteous LP's: more details

- Optionally, insert here:
  - 3 phases of argumentation in
    - courteous semantics
    - post-compilation rules
  - sample post-compilation rule set

Prioritized argumentation in an opposition-locale.

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

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



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

Courteous LP's: 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 both p & ¬p.
- Efficient: low computational overhead beyond ordinary LP's.
  - Tractable given reasonable restrictions (Datalog, bound v on #var's/rule):
    - extra cost is equivalent to increasing v to (v+2) in ordinary LP's.
  - By contrast, more expressive prioritized rule representations (e.g., Prioritized Default Logic) add NP-hard overhead.
- Modular software engineering: via courteous compiler: CLP → OLP.
  - A radical innovation. Add-on to variety of OLP rule systems. O(n^3).

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.

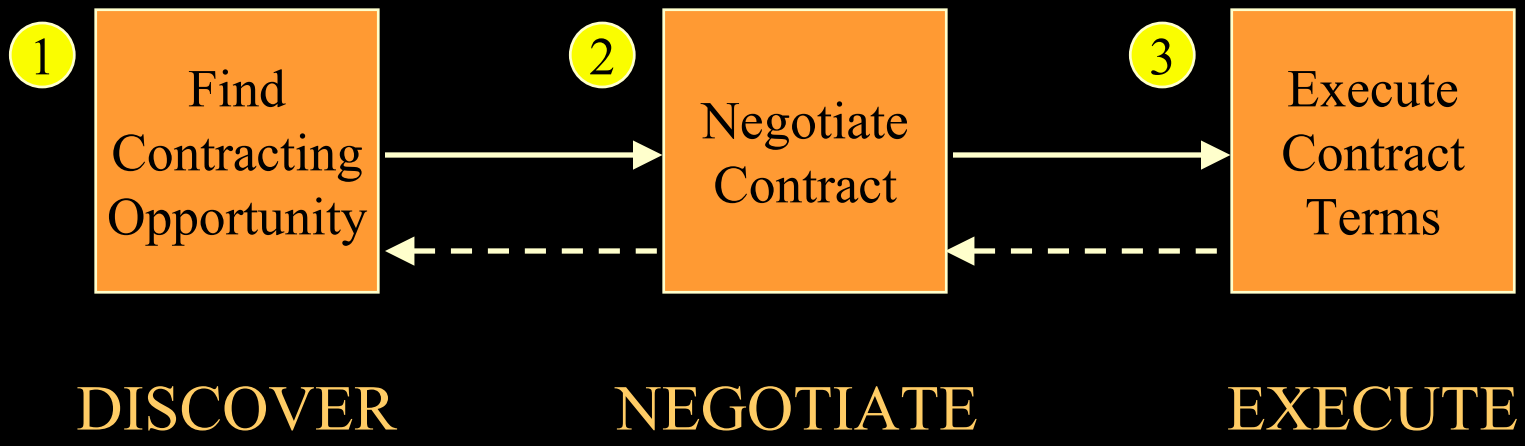
Summary: Courteous LP's in XML as Core KR

- 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.
  - courteous compiler → modularity in software engineering.
  - mutex's & conflict locales → keep tractability. (Compiler is O(n^3).)
- Novelty: do it in XML → ease of parsing, integration in Web engineering.

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.

Contracting 1-2-3



- Applies to any contracting, electronic or not.
- May iterate or interleave these steps.
- Boundaries not necessarily sharp.