The RuleML Family of Web Rule Languages

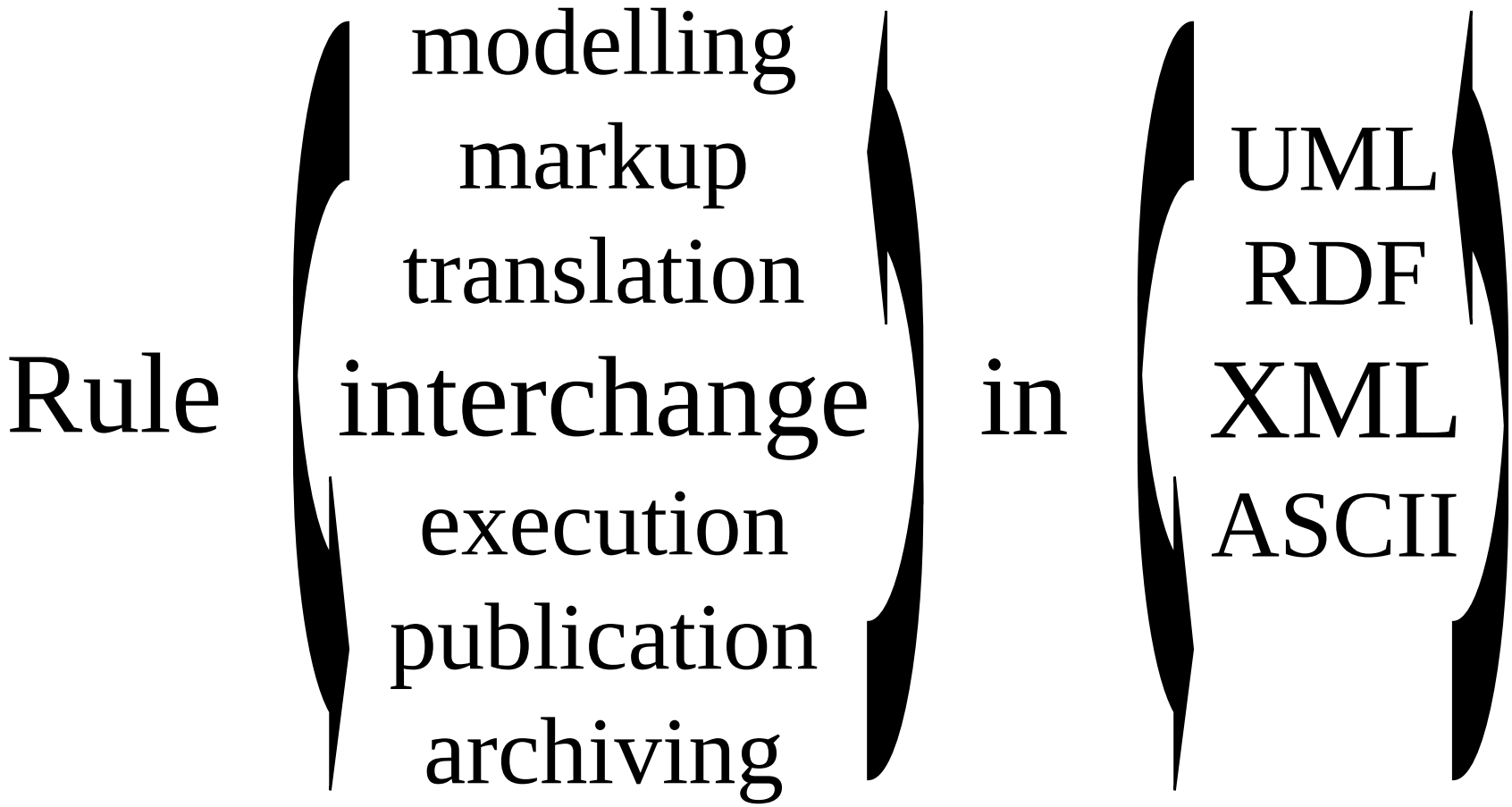
PPSWR'06, Budva, Montenegro, 10 June 2006
Revised, RuleML'06, Athens, GA, 11 Nov. 2006
Shortened, Vienna, SWT Course, 17 Nov. 2008

Harold Boley
University of New Brunswick, Canada
National Research Council of Canada

Introduction

- Rules are central to the Semantic Web
- Rule interchange in an open format is important for e-Business
- RuleML is the de facto open language standard for rule interchange/markup
- Collaborating with W3C (RIF), OMG (PRR, SBVR), OASIS, DARPA-DAML, EU-REWERSE, and other standards/gov'nt bodies

RuleML Enables ...



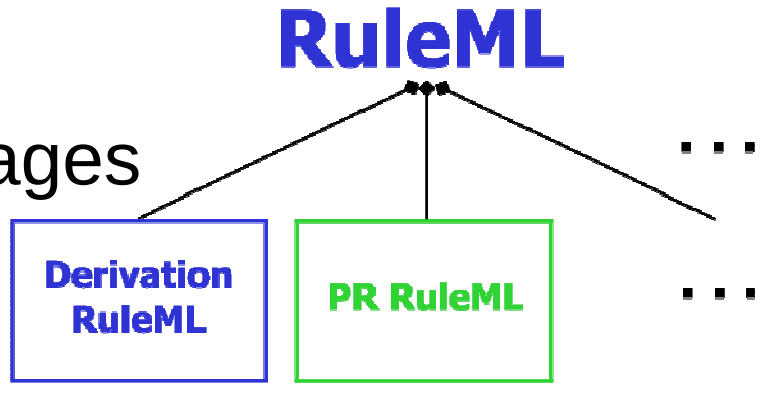
RuleML Identifies ...

- Expressive sublanguages
  - for Web rules
  - started with
    - *Derivation* rules: extend SQL views
    - *Reaction* rules: extend SQL triggers
  - to empower their subcommunities

RuleML Specifies ...

- Derivation rules via XML Schema:
  - All sublanguages:(OO) RuleML 0.91
  - First Order Logic: FOL RuleML 0.91
  - With Ontology language: SWRL 0.7
    - A Semantic Web Rule Language Combining OWL (W3C) and RuleML
  - With Web Services language: SWSL 0.9
- Translators in & out (e.g. Jess) via XSLT

Modular Schemas

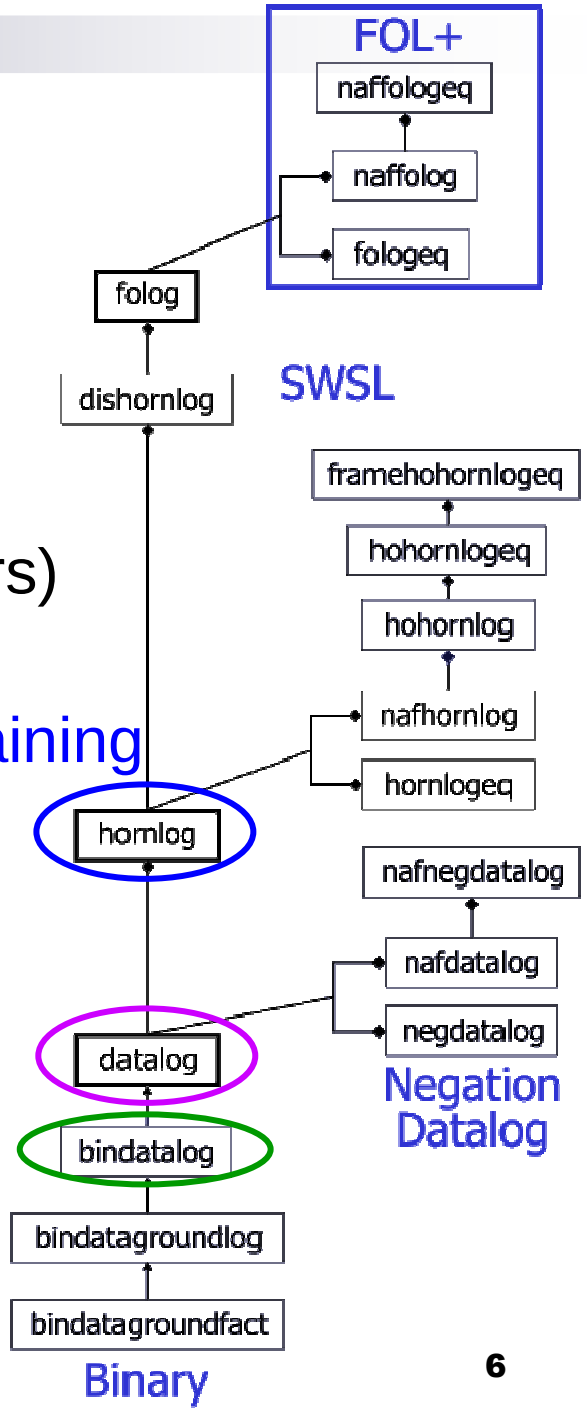


"RuleML is a **family** of sublanguages whose **root** allows access to the language as a whole and whose **members** allow to identify customized subsets of the language."

- RuleML: Rule Markup Language
  - RuleML derivation rules (shown here) and production rules defined in XML Schema Definition (XSD)
  - Each XSD of the family corresponds to the expressive class of a specific RuleML sublanguage
- The most recent schema specification of RuleML is always available at http://www.ruleml.org/spec
- Current release: RuleML 0.91
- Previews: http://wiki.ruleml.org/XSD_Workplan

Schema Modularization

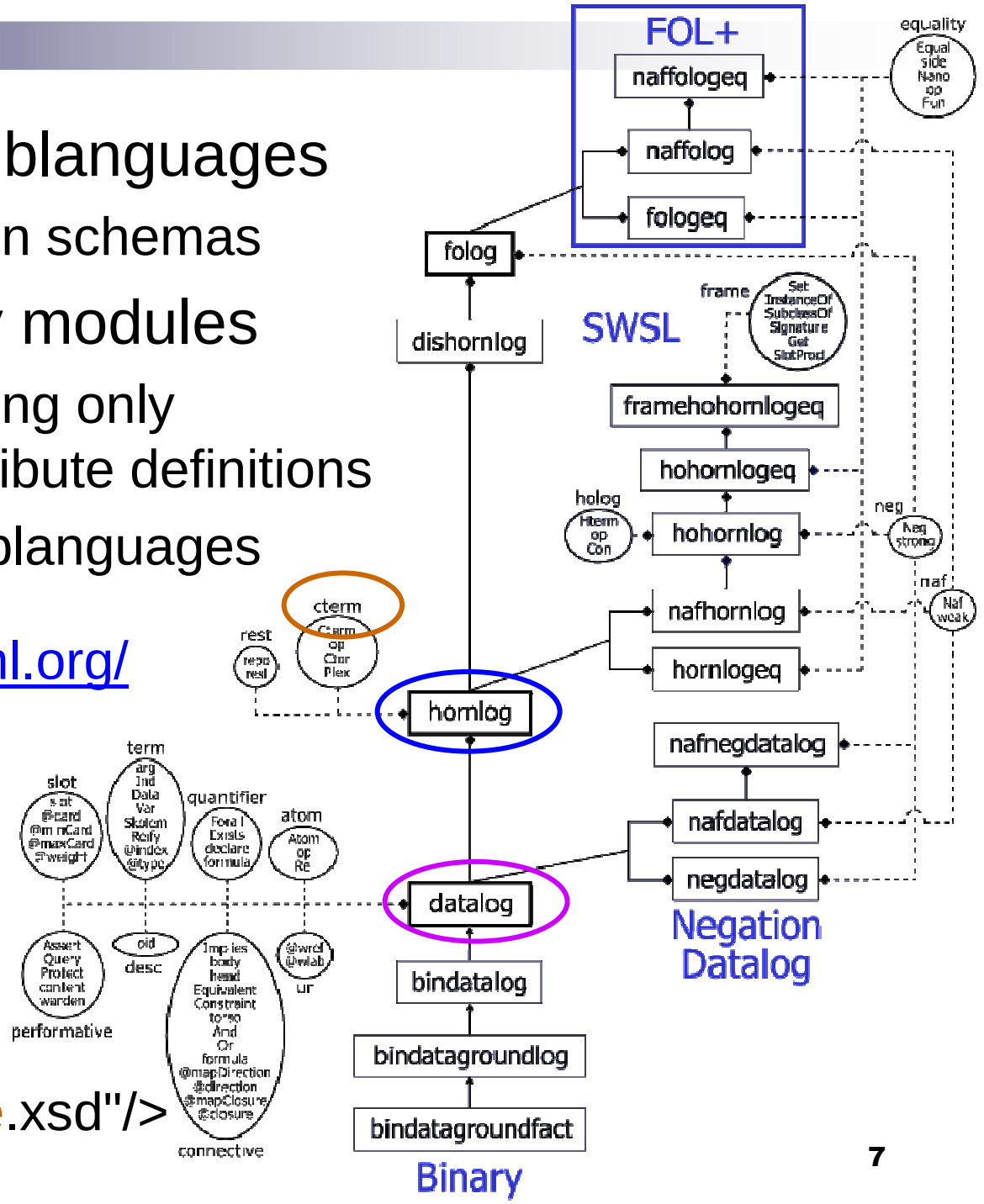
- XSD URIs identify expressive classes
  - Receivers of a rulebase can validate applicability of tools (such as Datalog vs. Hornlog interpreters)
  - Associated with semantic classes (such as function-free vs. function-containing Herbrand models)
- Modularization (Official Model)
  - Aggregation: e.g., Datalog *part of* Hornlog
  - Generalization: e.g., Bindatalog *is a* Datalog



- **Rectangles** are sublanguages
  - Inheritance between schemas
- **Ovals** are auxiliary modules
  - Elementary, including only element and/or attribute definitions
  - Become *part of* sublanguages

E.g., in http://www.ruleml.org/0.91/xsd/hornlog.xsd

```
<xs:redefine
schemaLocation=
"datalog.xsd">
<xs:include
schemaLocation=
"modules/cterm_module.xsd"/>
```

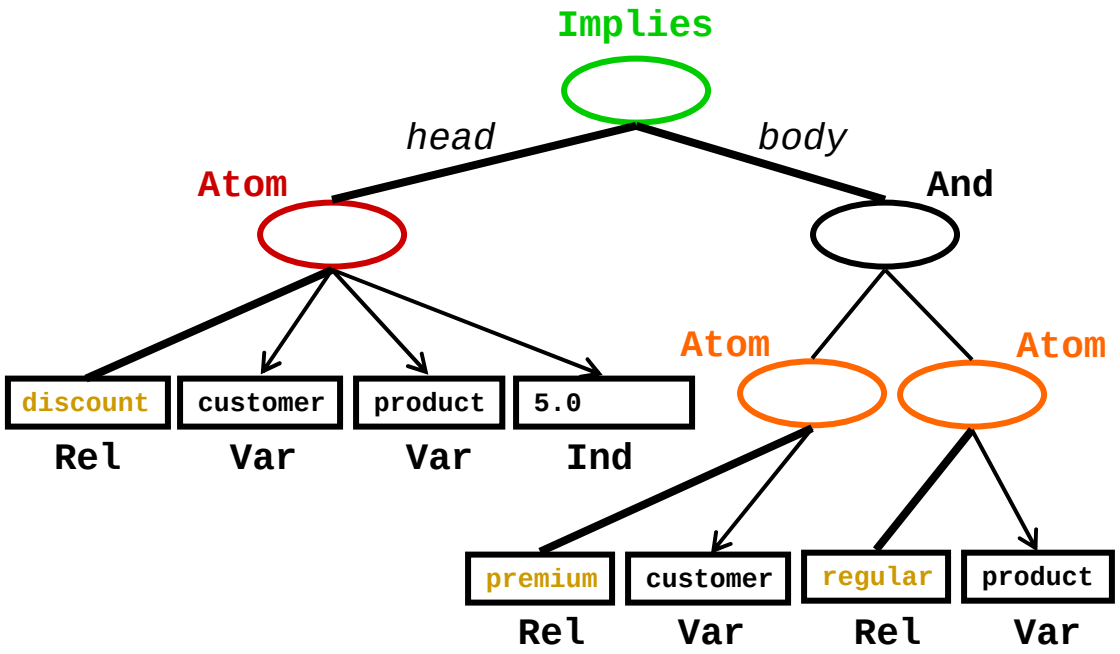


Bring Datalog to the Semantic Web

- Start with n-ary relations (not binary properties)
- Keep **`Var`**iable typing optional (reuse RDFS' subClassOf taxonomies as sort lattices)
- Allow signature declarations of arities and types
- Employ function-free facts as well as Horn rules (rather than 1st: RDF descriptions; 2nd: RDF rules)
- Use function-free Herbrand model semantics (querying stays decidable)
- Provide three syntactic levels:
  - User-oriented: Prolog-like, but with "?"-variables
  - Abstract: MOF/UML diagrams
  - XML serialization: Datalog RuleML

Business Rule: Positional

"The **discount** for a *customer* buying a *product* is 5 percent if the *customer* is **premium** and the *product* is **regular**."



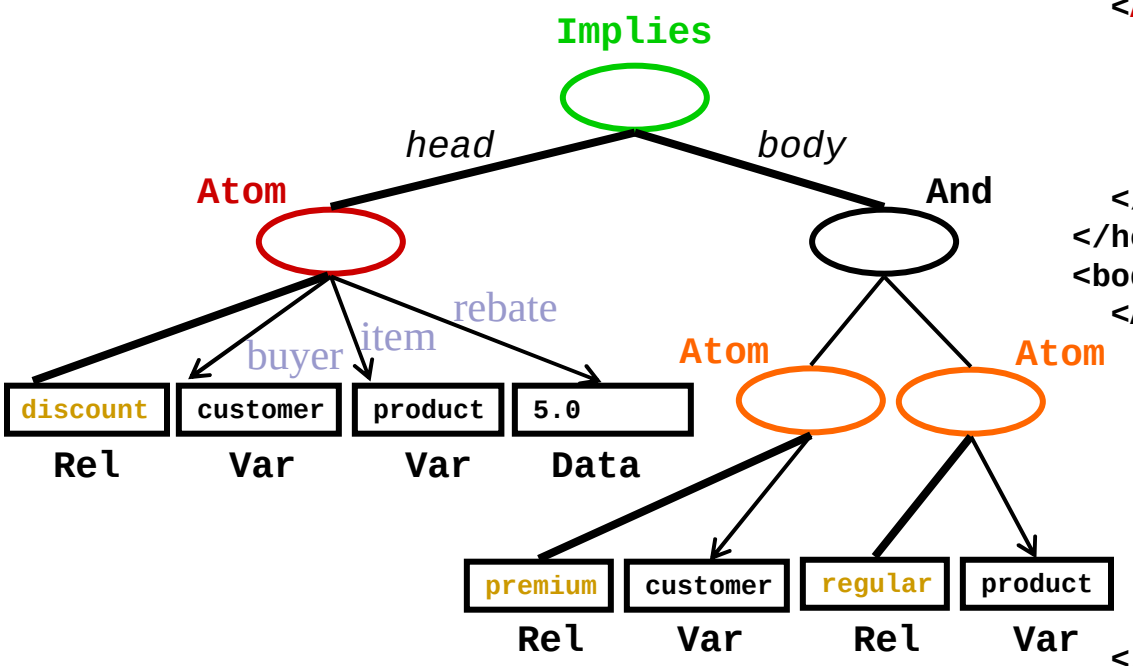

```
<Implies>
  <head>
    <Atom>
      <Rel>discount</Rel>
      <Var>customer</Var>
      <Var>product</Var>
      <Ind>5.0</Ind>
    </Atom>
  </head>
  <body>
    <And>
      <Atom>
        <Rel>premium</Rel>
        <Var>customer</Var>
      </Atom>
      <Atom>
        <Rel>regular</Rel>
        <Var>product</Var>
      </Atom>
    </And>
  </body>
</Implies>
```

Extend Datalog for the Semantic Web (I)

- Allow slots as name**->**filler pairs in **`Atom`**s (cf. F-logic's methods and RDF's properties)
- Extend optional types and signatures for slots
- Add optional object identifiers (**`oid`**s) to atoms
- Separate **`Data`** literals from **`Ind`**ividual constants

Business Rule: Slotted (for OO)

"The **discount** for a *customer* buying a *product* is 5 percent if the *customer* is **premium** and the *product* is **regular**."




```
<Implies>
  <head>
    <Atom>
      <Rel>discount</Rel>
      <slot><Ind>buyer</Ind><Var>customer</Var></slot>
      <slot><Ind>item</Ind><Var>product</Var></slot>
      <slot><Ind>rebate</Ind><Data>5.0</Data></slot>
    </Atom>
  </head>
  <body>
    <And>
      <Atom>
        <Rel>premium</Rel>
        <Var>customer</Var>
      </Atom>
      <Atom>
        <Rel>regular</Rel>
        <Var>product</Var>
      </Atom>
    </And>
  </body>
</Implies>
```

Extend Datalog for the Semantic Web (II)

- Permit IRI webizing for **`Data`** (XML Schema Part 2), **`Ind`**ividuals (RDF's **`resource`**s), **`Rel`**ations, **`slot`** names, types (RDFS' classes), and **`oid`**s (RDF's **`about`**)
- Introduce **`Module`** (scope) construct for clauses (cf. RDF's named graphs)
- Add scoped-default (**`Naf`**), strong (**`Neg`**), scoped-default-of-strong negation (unscoped: cf. ERDF)
- Integrate with Description Logics
  - Homogeneous (SWRL, Datalog RuleML + OWL-DL)
  - Hybrid (AL-log, Datalog^{DL}, DL+log, ...)

Bring Horn Logic to the Semantic Web

- Augment Datalog with uninterpreted **`Fun`**ctions and their **`Expr`**essions; also for extended Datalog
- Augment Datalog's Herbrand model semantics with such **`Fun`**ctions (querying becomes undecidable)
- Extend Datalog syntaxes
  - XML Schema of Hornlog RuleML inherits and augments XML Schema of Datalog RuleML
- Add **`Equal`**ity and **`in`**terpreted **`Fun`**ctions (XML serialization: attribute **`in="yes"`**)
- Reuse XQuery/XPath functions and operators as built-ins

Specify a First-Order Logic Web Language

- Layer on top of either
  - Disjunctive Datalog: **`Or`** in the head generalizing Datalog
  - Disjunctive Horn Logic: **`Or`** in head of near-Horn clauses
- Alternatively, layer on top of either
  - Disjunctive Datalog with restricted strong **`Neg`**ation
  - Disjunctive Horn Logic with restricted strong **`Neg`**
- Permit unrestricted **`Or`**, **`And`**, strong **`Neg`**, and quantifiers **`Forall`** and **`Exists`** to obtain FOL
- Use semantics of classical FOL model theory
- Extend Hornlog RuleML syntax to FOL RuleML

Approach Production and Reaction Rules

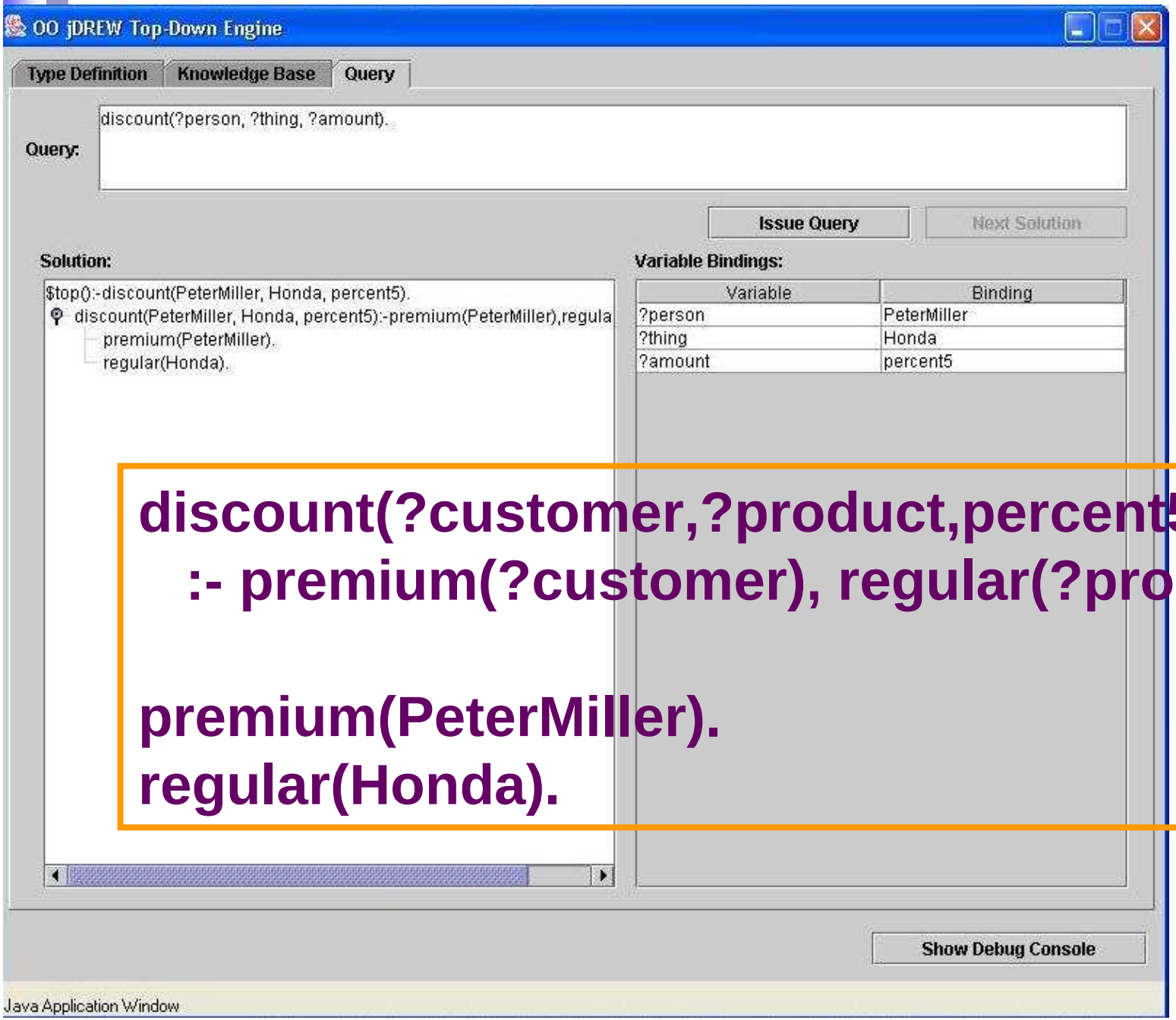
- Share Condition (C) part with earlier languages as proposed for the RIF Condition Language
- Develop Action (A) part of Production Rules via a taxonomy of actions on KBs (Assert, Retract, ...), on local or remote hosts, or on the surroundings
- Develop Event (E) part of Reaction Rules via a corresponding taxonomy
- Create CA and ECA families bottom-up and map to relevant languages for Semantic Web Services
- Serialized: <**Reaction**> E C A </**Reaction**>
- See http://ibis.in.tum.de/research/ReactionRuleML TG

Bidirectional Interpreters in Java

- Two varieties of reasoning engines
  - **T**op-**D**own: backward chaining
  - **B**ottom-**U**p: forward chaining
- jDREW: *Java Deductive Reasoning Engine for the Web* includes both TD and BU
  http://www.jdrew.org
- OO jDREW: *Object-Oriented* extension to jDREW
  http://www.jdrew.org/oojdrew
- Java Web Start online demo available at
  http://www.jdrew.org/oojdrew/demo.html

OO jDREW Slots

- Normalized atoms and complex terms
  - **`oid`**s (object identifier)
  - Positional parameters (in their original order)
  - Positional rest terms
  - Slotted parameters (in the order encountered)
  - Slotted rest terms
- Efficient unification algorithm
  - Linear O(m+n): instead of O(m*n)
    - No need for positional order
    - Slots internally sorted
  - Steps:
    - Scan two lists of parameters
      - Matching up roles and positions for positional parameters
      - Unifying those parameters
    - Add unmatched roles to list of rest terms
    - Generate dynamically a Plex (RuleML's closest equivalent to a list) for a collection of rest terms

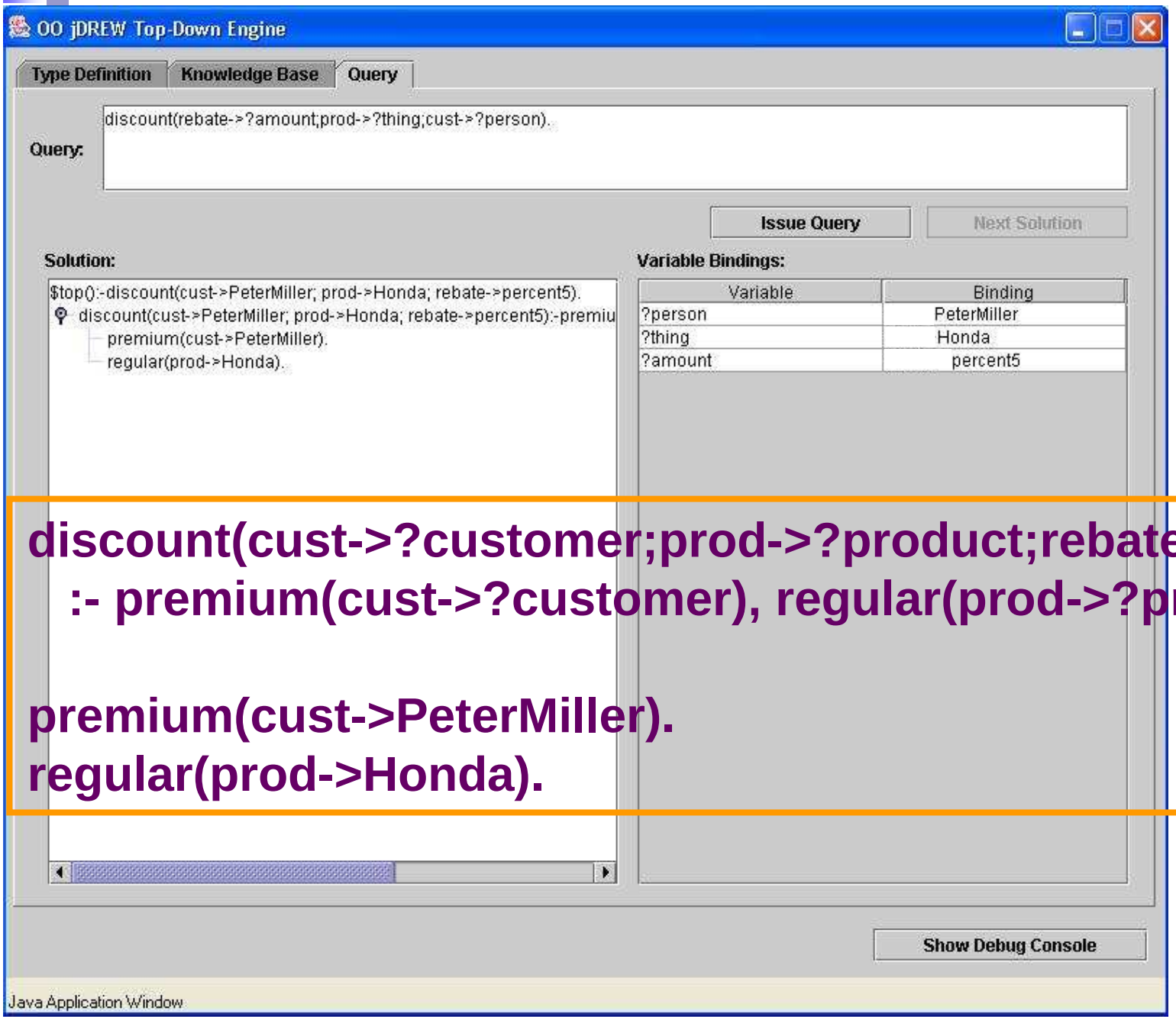


positional

POSL syntax

```
discount(?customer,?product,percent5)
   :- premium(?customer), regular(?product).

premium(PeterMiller).
regular(Honda).
```



slotted

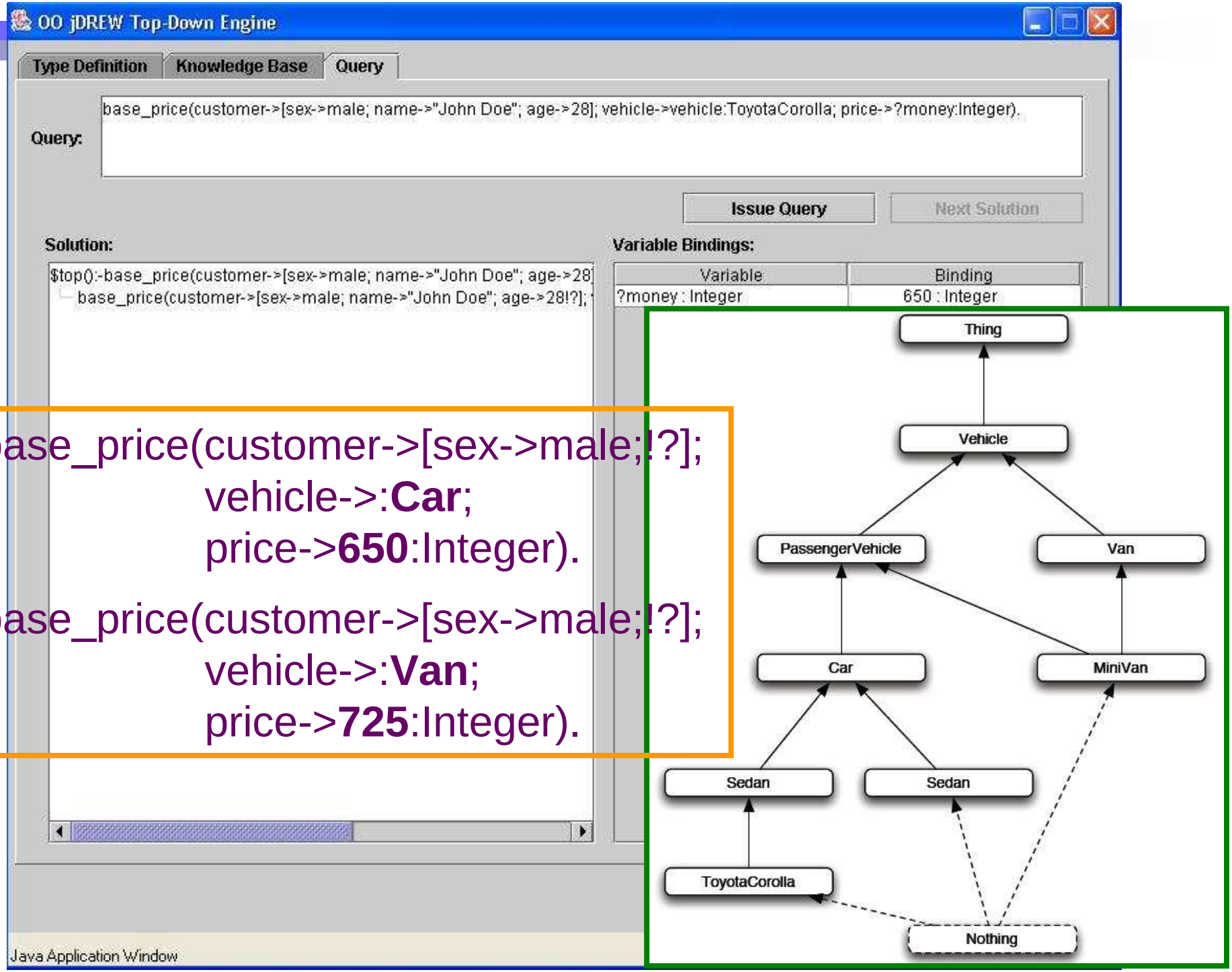
POSL syntax

```
discount(cust->?customer;prod->?product;rebate->percent5)
   :- premium(cust->?customer), regular(prod->?product).

premium(cust->PeterMiller).
regular(prod->Honda).
```

OO jDREW Types

- Order-sorted type system
  - RDF Schema: lightweight taxonomies of the Semantic Web
  - To specify a partial order for a set of classes in RDFS
- Advantages
  - Having the appropriate types specified for the parameters
  - To restrict the search space
  - Faster and more robust system than when reducing types to unary predicate calls in the body
- Limitations
  - Only modeling the taxonomic relationships between classes
  - Not modeling properties with domain and range restrictions



base_price(customer->[sex->male;!?];
vehicle->:**Car**;
price->**650**:Integer).

base_price(customer->[sex->male;!?];
vehicle->:**Van**;
price->**725**:Integer).

Conclusions

- RuleML is modular family, whose root allows to access the language as a whole and whose members allow customized subsets
- New members joining, e.g. Fuzzy RuleML
- Concrete & abstract syntax of RuleML
  - Specified by modular XSD (shown here) & MOF
- Formal semantics of OO Hornlog RuleML
  - Implemented by OO jDREW BU & TD
- Interoperability/Interchange of/with RuleML
  - Realized by translators, primarily via XSLT