SweetJess: Translating DamlRuleML To Jess

Presentation of Paper at the International Workshop on Rule Markup Languages for Business Rules on the Semantic Web, held at 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/

Mahesh Ghande, Timothy Finin

Inst. For Global Electronic Commerce, Computer Science Dept.

University of Maryland Baltimore County (UMBC)

{mgandh1, finin}@cs.umbc.edu

http://www.csee.umbc.edu/{~mgandh1,~finin}

Overall Problem Addressed, Previous Work

- Rules as widely deployed KR → SW Knowledge Integration for Business
- Challenge: inter-operability of heterogeneous intelligent applications ("agents") that use rules (incl. relational DB's).
  - E.g., rules represent e-business policies and workflows.
  - Heterogeneous rule systems: four important families:
    - Prolog, SQL; production (OPS5), ECA
- *History:*
  - Core requirements & design '99 (while at IBM Research)
    - Declarative Logic Programs in XML; + *extensions:*
      - Courteous LP: prioritized conflict handling; modularity; tractably
      - Situated LP: procedural attachments for actions, queries: cleanly
  - IBM CommonRules V1 '99 (V3 currently)
    - large-scale pilot (EECOMS $29Million, supply chain) '99-'00
  - Co-Lead RuleML: V0.7 '01 (V0.8 currently)

Problem and Previous Work continued

- SweetRules V1 '01: 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 for this work: "bring them to the party"
      - Jess: production (OPS5) , close to ECA
        - popular, open-source, Java: it's useful in particular

Projects Context at MIT Sloan since '01

- 1. Rules KR Technology, esp. for Semantic Web Services
  - fundamental theory, technology, support of standards
  - SweetRules prototype (Semantic WEb Enabling Technology)
    - translation, inferencing, merging
    - current work: + ontologies cf. OWL, database systems

- 2. Business Implications of the Semantic Web
  - applications & strategy
  - esp. B2B, e-contracting, finance, supply chain, policies
  - SweetDeal prototype for rule-based e-contracting
    - modular, reusable contract fragments: as SCLP RuleML rulesets

Outline

- 1. Intro: Why Care
  - "bring to the party" of SW e-business, RuleML, and SweetRules: production/OPS5 & ECA rules; inter-operate Jess via RuleML translator
- 2. Some Details of the Translation
  - Ordinary Logic Programs: facts, rules
  - Situated extension to LP: procedural attachments
    - effectors (actions); sensors (tests/queries)
  - Courteous extension to LP: prioritized conflict handling; mutex's, classical neg.
    - via tractable Courteous Compiler → OLP
- 3. Other Contributions related to the Translation
  - Inferencing in SCLP RuleML via: translate to Jess, run rules in Jess, go back
  - DamlRuleML: DAML+OIL ontology for RuleML's syntax
    - E.g., Rule, Atom, Predicate as classes. Nice, but not necessary, for translating.
- 4. Conclusions and Future Work
  - comparative insights: Jess limitations, e.g., all-bound-sensors
  - in progress: prototype; deeper theory

Translating a Fact from (Daml)RuleML to Jess

```
<damlRuleML:fact>
    <damlRuleML:_rlab>fact8962</damlRuleML:_rlab>
        <damlRuleML:_head>
          <damlRuleML:atom>
             <damlRuleML:_opr>
                <damlRuleML:rel>shopper<damlRuleML:rel>
             </damlRuleML:_opr>
             <damlRuleML:ind>Debbie</damlRuleML:ind>
          </damlRuleML:atom>
        </damlRuleML:_head>
</damlRuleML:fact>
```

```
equivalent in JESS:
  (assert (shopper Debbie) )
```


Translating a Rule from (Daml)RuleML to Jess

```
<damlRuleML:imp>
   <damlRuleML:_rlab>
      <damlRuleML:ind>steadySpender</damlRuleML:ind>
   </damlRuleML:_rlab>
   <damlRuleML:_body>
     <damlRuleML:andb>
        <damlRuleML:atom>
           <damlRuleML:_opr>
              <damlRuleML:rel>shopper<damlRuleML:rel>
           </damlRuleML:_opr>
           <damlRuleML:var>Cust</damlRuleML:var>
        </damlRuleML:atom>
        <damlRuleML:atom>
          <damlRuleML:_opr>
             <damlRuleML:rel>spendingHistory<damlRuleML:rel>
          </damlRuleML:_opr>
          <damlRuleML:tup>
            <damlRuleML:var>Cust</damlRuleML:var>
            <damlRuleML:ind>loyal</damlRuleML:ind>
          </damlRuleML:tup>
        </damlRuleML:atom>
      </damlRuleML:andb>
    </damlRuleML:_body>
```

Continued: Translating a Rule from (Daml)RuleML to Jess

```
<damlRuleML:_head>
    <damlRuleML:atom>
      <damlRuleML:_opr>
        <damlRuleML:rel>giveDiscount<damlRuleML:rel>
      </damlRuleML:_opr>
      <damlRuleML:tup>
       <damlRuleML:ind>percent5</damlRuleML:ind>
       <damlRuleML:var>Cust</damlRuleML:var>
      </damlRuleML:tup>
    </damlRuleML:atom>
   </damlRuleML:_head>
  </damlRuleML:imp>
```

```
Equivalent in  JESS:
(defrule steadySpender
    (shopper ?Cust)
    (spendingHistory ?Cust loyal)
    =>
    (assert (giveDiscount percent5 ?Cust) ) )
```


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) ) )
```


Translating a Sensor Statement

```
<damlRuleML:sens>
  <damlRuleML:_opr>
    <damlRuleML:rel>spendingHistory</damlRuleML:rel>
  </damlRuleML:_opr>
  <damlRuleML:_aproc>
    <damlRuleML:jproc>
        <damlRuleML:meth>getSpendingLevel</damlRuleML:meth>
        <damlRuleML:clas>transactions.customers.queries</damlRuleML:clas>
        <damlRuleML:path>com.widgetsRUs.transactionsDB.customers
          </damlRuleML:path> </damlRuleML:jproc>  </damlRuleML:_aproc>
  <damlRuleML:_modli>
     <damlRuleML:bmode val="bound"></damlRuleML:bmode>
     <damlRuleML:bmode val="bound"></damlRuleML:bmode>
   </damlRuleML:_modli>

</damlRuleML:sens>
```

Associates with predicate P : an attached procedure Q that is side-effect-*free*.

- Testing a rule condition about P results in a query to Q.

modli = the proc.'s **binding pattern**:

a list of, for each argument, a ...

bmode = binding mode (bound vs. free)

Simplistic view of Equivalent in JESS is:

```
(defrule sense_steadySpender_1
   (shopper ?Cust)
   (test (shopper_SF getSpendingLevel transaction.customer.queries
                    (create$ ?Cust   loyal) ) )
=> (assert (giveDiscount percent5 ?Cust) ) )
```

Translating a Sensor Statement continued

- Equivalent in JESS: More precisely, the presence of a sensor statement modifies the translation of every rule whose body mentions that sensor predicate:

```
• (defrule steadySpender
•    (shopper ?Cust)
•    (or (spendingHistory ?Cust ?loyal)
•        (test (sensor getSpendingLevel transaction.customer.queries
•                      (create$ ?Cust   loyal) ) ) )
• => (assert (giveDiscount percent5 ?Cust) ) )
```


Also in the Jess equivalent:

```
(deffunction effector                              /* generic effector */
  (?methodName ?className $?arglist)
    (bind ?classInstance (new ?className))
                                        /*create new instance of class */
    (return (call ?classInstance ?methodName $?arglist) )  )


(deffunction sensor                              /* generic sensor */
    (?methodName ?className $?arglist)
      (bind ?classInstance (new ?className))
                                        /*create new instance of class */
       (return (call ?classInstance ?methodName $?arglist) )  )
```

[& set the CLASSPATH, appropriately]

[similar for RMI, using hostname instead of classpath]

SweetRules & SweetJess: Translating Courteous features of SCLP RuleML

Courteous compiler

courteous representation
(Sit.) Courteous LP.

mutex* priorities

≡ equivalent semantically

ordinary ("vanilla")
(Sit.)OLP representation

Tractable:

O(n^3), often linear

Preserves ontology.

Plus extra predicates for

- phases of prioritized argumentation
- classical negations

Translation for Situated OLP

* classical negation too

Discussion, Conclusions, and Future Work

- Nature of contribution:
  - design for translation, and its use in inferencing
- In progress: implementation → testing/refinement of the design
- In progress: deeper theory → proof of correctness, hard limits of expressiveness that can handle

- Tricky/subtle: Jess "Functions"
  - used for procedures, logical functions, and system commands

- Expressive restrictions imposed on the translation (currently):
  - "All-bound-sensors": sensor arguments must all be bound (i.e., instantiated) before call.
  - "Datalog" (= no ctor's), stratified, misc. about naming

continued: Conclusions and Future Work

- Comparative insights:
  - Courteous more powerful & clean than control-sequencing
  - Situated more powerful and clean than Jess “functions”

- Implications → Future Work:
  - Can do translation and RuleML-based inter-operability for more systems in production/reactive/ECA category
    - Current Work: more closely represent Events cf. ECA
  - Enables merging, knowledge sharing/integration
  - Helps achieve business intelligence on the Semantic Web
- Broad Future Direction:
  - Represent and reason over RDF and DAML+OIL content

- For More Info:
  - http://www.mit.edu/~bgrosof/
- Download Site:
  - http://daml.umbc.edu/sweetjess

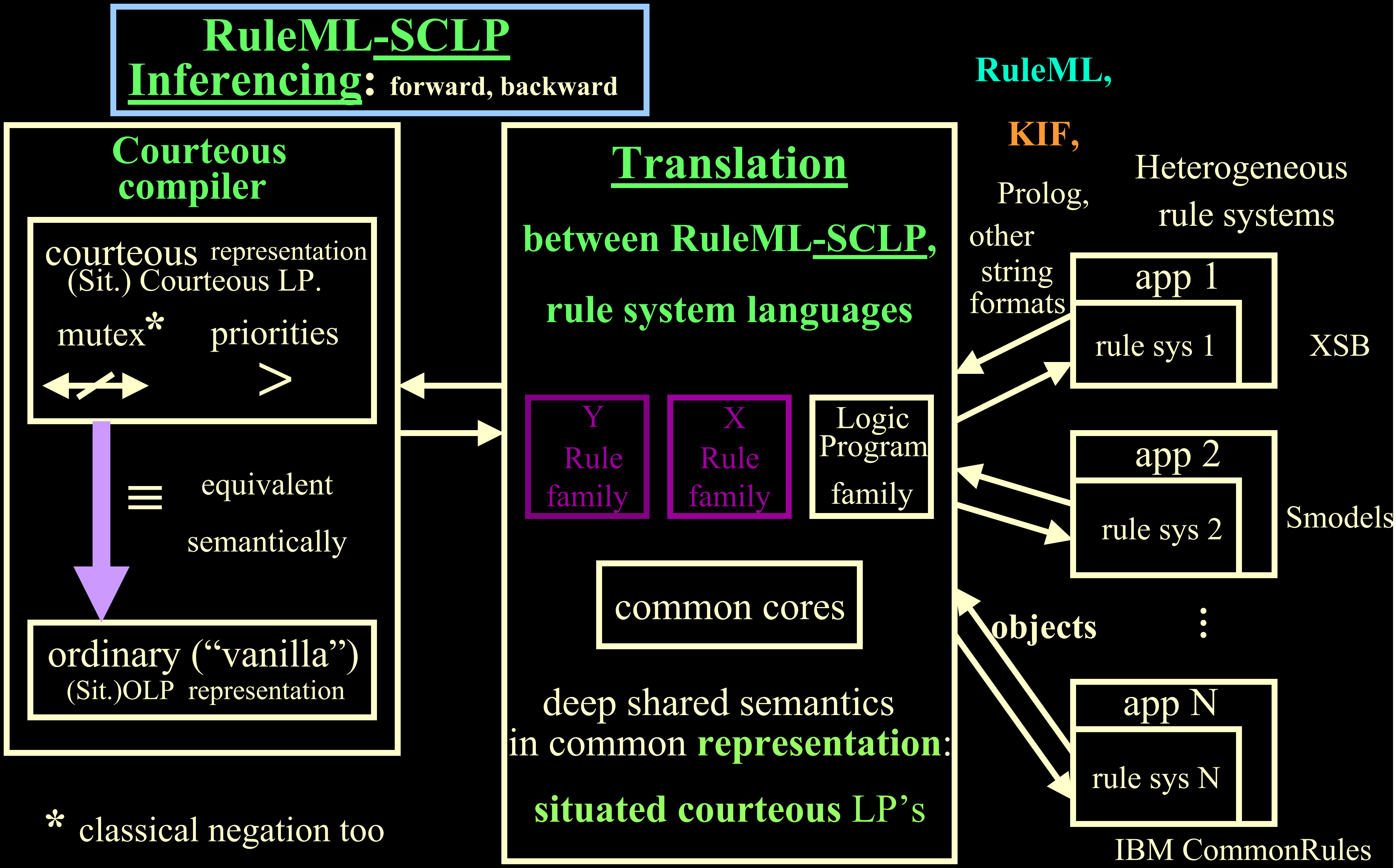
OPTIONAL SLIDES FOLLOW

“RuleML:

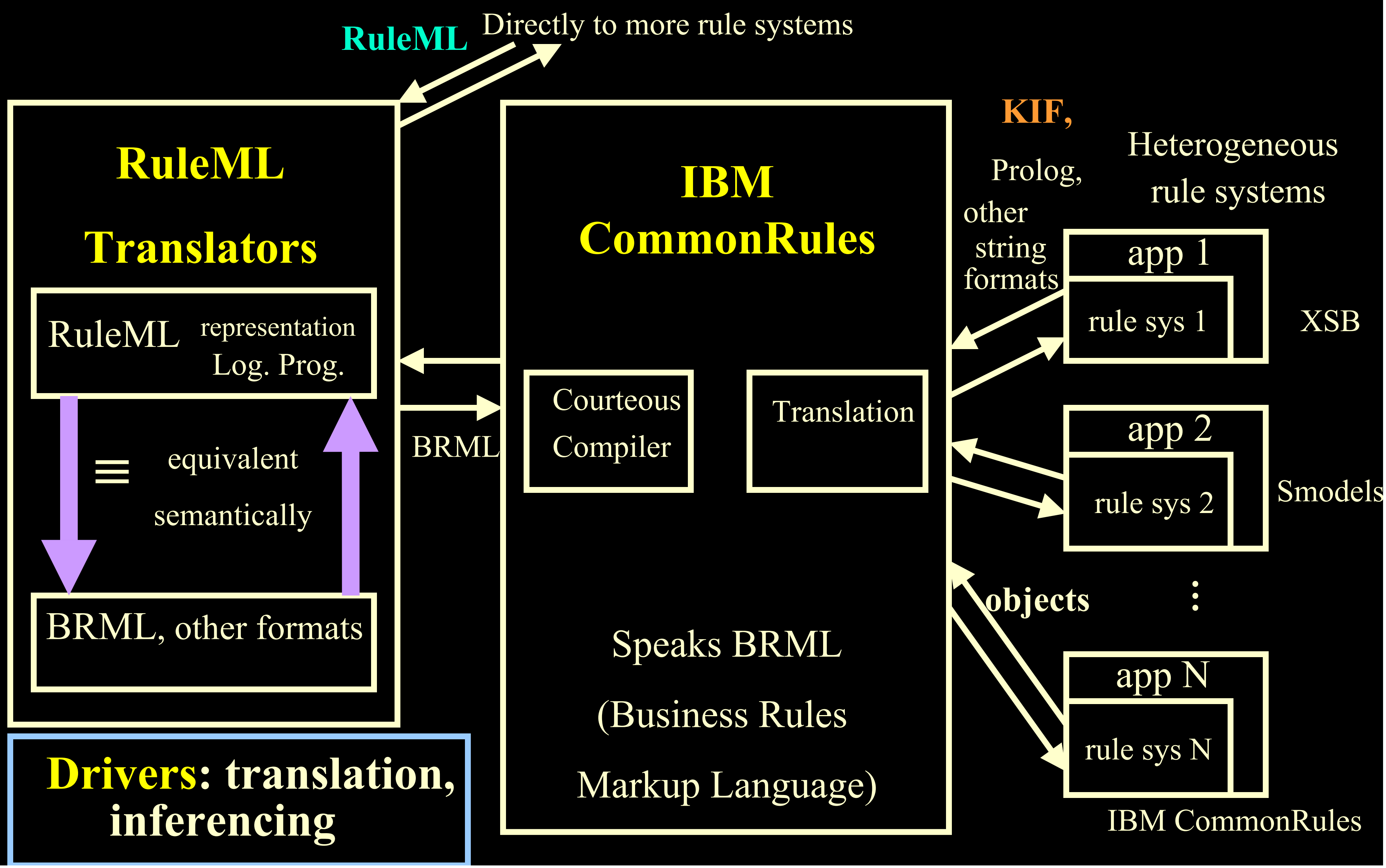
Semantic Web Rules!”

Functionality: SWEETRules Prototype

(Semantic WEb Enabling Technology)



Dec.-2001 Architecture: SWEETRules Prototype (Semantic WEb Enabling Technology)



Criteria for Contract Rule Representation

1

- *High-level:* Agents reach common understanding; contract 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.

  } OLP
- Prioritized conflict handling. → Courteous
- Ease of parsing.
- Integration into Web-world software engineering.

  } → XML
- Procedural attachments. → Situated

