Overview of SweetRules V2.0:

Tools for Semantic Web Rules and Ontologies, including Translation, Inferencing, Analysis, and Authoring

by Benjamin Grosof and Mike Dean***

**MIT Sloan School of Management, http://ebusiness.mit.edu/bgrosof*
***BBN Technologies, http://www.daml.org/people/mdean*

This is a section of the presentation “DAML Rules Report for PI Mtg. Nov.-Dec. 2004”

by Benjamin Grosof and Mike Dean***
(DAML Rules Co-Chairs)
**MIT Sloan School of Management, http://ebusiness.mit.edu/bgrosof*
***BBN Technologies, http://www.daml.org/people/mdean*

Presented at DARPA Agent Markup Languages program (DAML)
Principal Investigators Meeting (held Nov. 30 – Dec. 2),
Dec. 1, 2004, San Antonio, Texas, USA
http://www.daml.org

Announcing…

- SweetRules V2.0 Initial Release was Monday Nov. 29 2004.
- Open-source on SemWebCentral.org
  - http://sweetrules.projects.semwebcentral.org
- *You're the first to hear* ☺

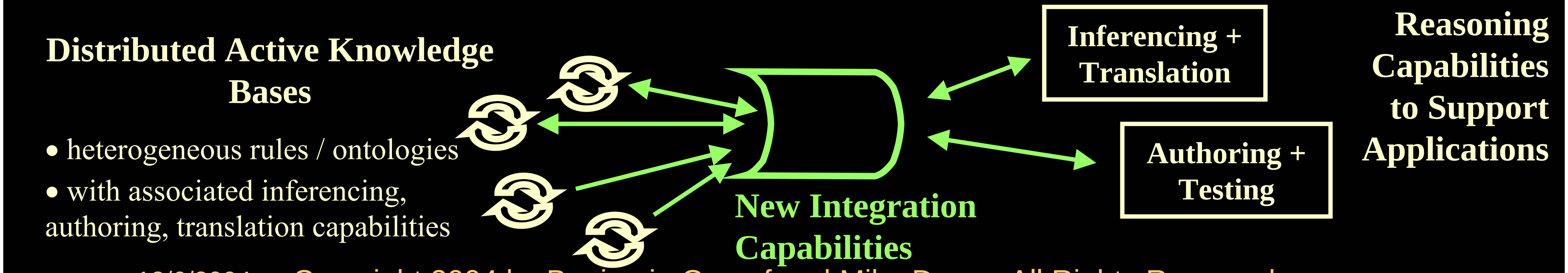
SweetRules V2 Overview

Key Ideas:

- Unite the commercially most important kinds of rule and ontology languages via a a new, common knowledge representation (SCLP) in a new standardized syntax (RuleML), including to cope with *heterogeneity* and resolve contradictory *conflicts*.
  - Capture most of the useful expressiveness, interoperably and scalably.
- Combine a large *distributed* set of rule and ontology knowledge bases that each are *active:* each has a different *associated engine* for reasoning capabilities (inferencing, authoring, and/or translation).
- Based on recent fundamental KR theory advances, esp. Situated Courteous Logic Programs (SCLP) and Description Logic Programs.
  - Including semantics-preserving translations between different rule languages/systems/families, e.g., Situated LP ↔ production rules

Application Areas (prototyped scenarios):

- Policies and authorizations; contracting, supply chain management; retailing, customer relationship management; business process automation and e-services; financial reporting and information; etc.



Distributed Active Knowledge Bases

- heterogeneous rules / ontologies
- with associated inferencing, authoring, translation capabilities

New Integration Capabilities

Inferencing + Translation

Authoring + Testing

Reasoning Capabilities to Support Applications

SweetRules Concept and Architecture

- Concept and Architecture: Tools suite for Rules and RuleML
  - Translation and interoperability between heterogeneous rule systems (forward- and backward-chaining) and their rule languages/representations
  - Inferencing including via translation between rule systems
  - Authoring, Analysis, and testing of rulebases
  - Open, lightweight, extensible, pluggable architecture overall

  - Merge knowledge bases
    - Combine rules with ontologies, incl. OWL
  - SWRL rules as special case of RuleML
  - Focus on kinds of rule systems that are commercially important

SweetRules Goals

- Research vehicle: embody ideas, implement application scenarios (e.g., contracting, policies)
  - Situated Courteous Logic Programs (SCLP) KR
  - Description Logic Programs (DLP) KR which is a subset of SCLP KR
  - RuleML/SWRL

- Proof of concept for feasibility, including of KR algorithms and translations between heterogenous families of rule systems
  - Encourage others: researchers; industry esp. vendors

- Catalyze/nucleate SW Rules communal efforts on:
  - Tools, esp. open-source
  - Application scenarios / use cases, esp. in services

SweetRules Website

- See http://sweetrules.projects.semwebcentral.org
  - – Downloadable
  - – Open-source code
  - – Documentation
    - Javadoc
    - ISWC-2004 Tutorial on Rules+Ontologies+Ebiz
    - Overview, README, Rule Formats, ...

SweetRules Context and Players

- Part of SWEET = “Semantic WEb Enabling Tools” (2001 –)
  - Other parts: … these use SweetRules …
    - SweetDeal for e-contracting
    - SweetPH for Process Handbook ontologies
- Cross-institutional. Collaborators invited!
  - Originated and coordinated by MIT Sloan since 2001
  - Code base: Java, XSLT; convenience shell scripts (for testing drivers)
  - Code by MIT, UMBC, BBN, Stanford, U. Zurich
  - Cooperating other institutions: U. Karlsruhe, IBM, NRC/UNB, SUNY Stonybrook, HP, Sandia Natl. Labs; RuleML Initiative
    - Collaboration on design of code by Stanford, U. Karlsruhe
  - Uses code by IBM, SUNY Stonybrook, Sandia Natl. Labs, HP, Stanford, Helsinki
  - Many more are good targets: subsets of Flora-2, cwm, KAON, JTP, SWI Prolog, Hoolet, Triple, DRS, ROWL, ...

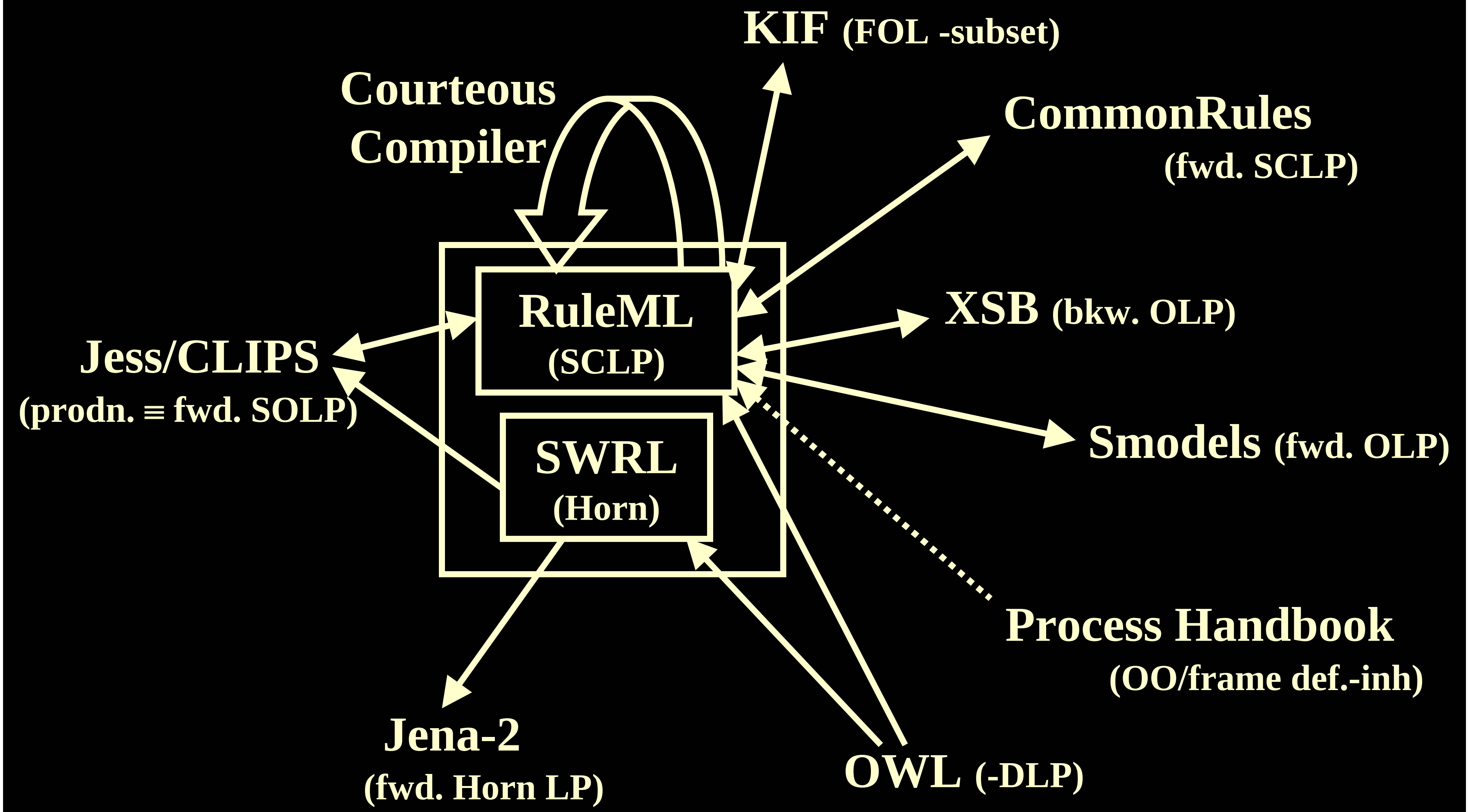
SweetRules V2.0 Fundamental KR Today

- Fundamental KR: Situated Courteous Logic Programs (SCLP)
  - Horn
  - + Negation-As-Failure (NAF) = Ordinary LP
  - + Courteous prioritized conflict handling
    - overrides relation on rule labels, classical negation, mutex integrity constraints
  - + Situated sensing & effecting
    - Invoke external procedural attachments
    - Sensing = tests/queries; e.g., built-ins
    - Effecting = side-effectful actions, triggered by conclusions

SweetRules V2.0 KR Languages Supported

- RuleML (SCLP)
- SWRL rules (named-classes-only)
- OWL
  - Esp. Description Logic Programs subset
- Prolog (pure, plus informational built-ins) – bkw. OLP
  - XSB
- Production Rules -- fwd. ~ SOLP
  - Jess/CLIPS; Jena
- Other:
  - KIF (FOL subset), IBM CommonRules (fwd. SCLP), Smodels (fwd. Prolog)
  - *Soon to be integrated:* Process Handbook (OO/frame ontologies with default inheritance)

SweetRules Today: Translators Graph

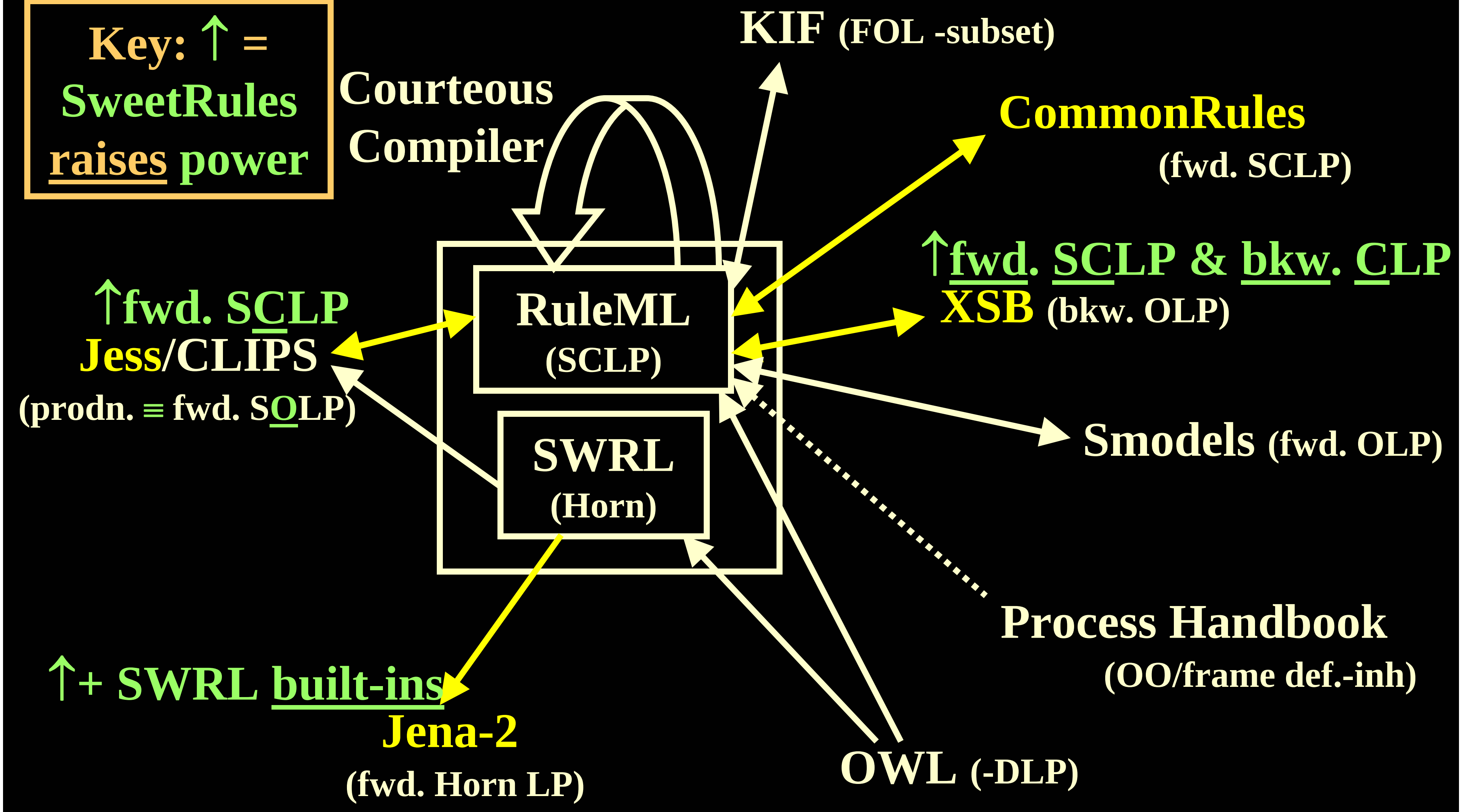


SweetRules Inferencing Capabilities Today: Overview

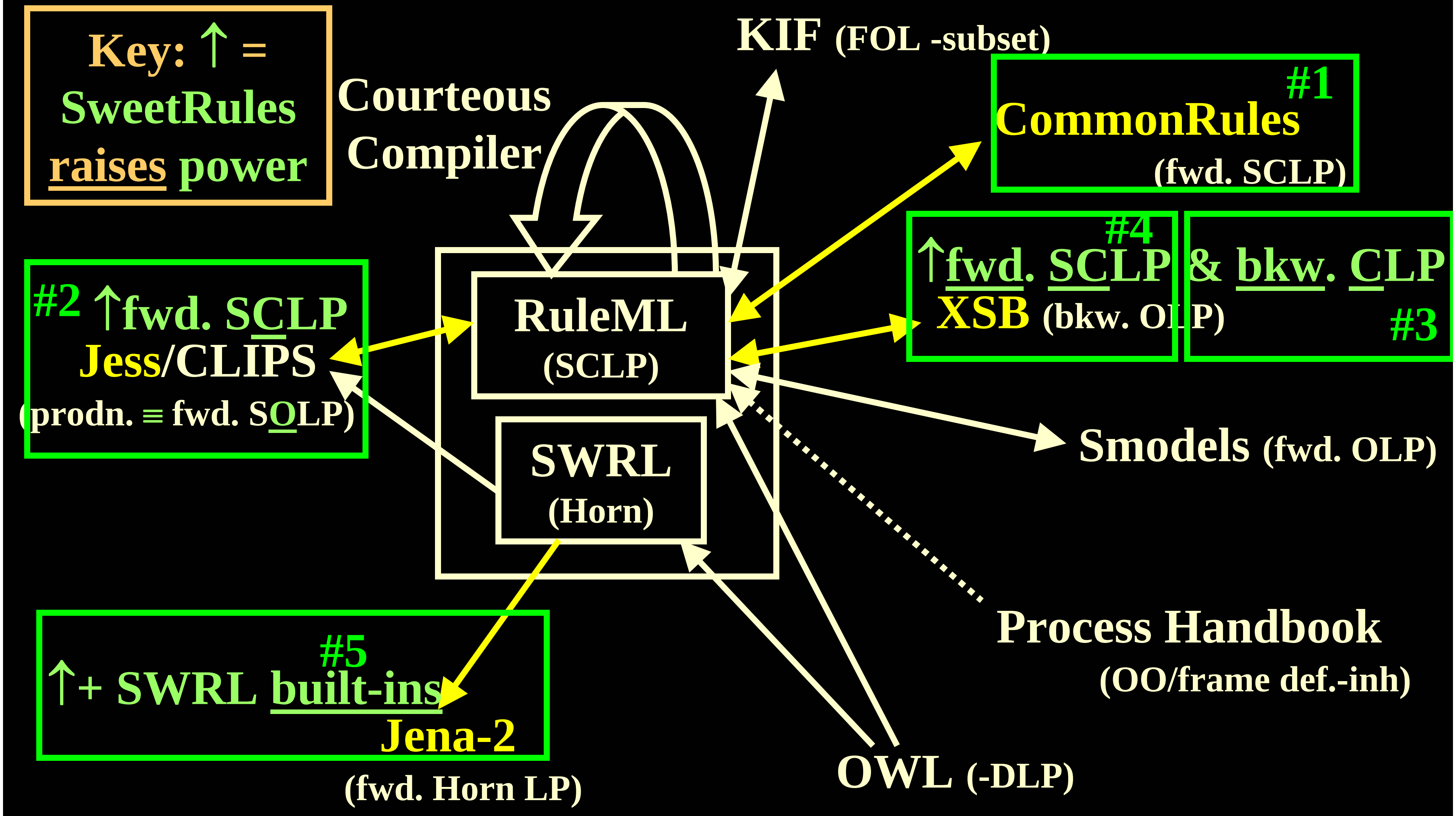
- Inferencing engines in RuleML/SWRL via translation:
  - Indirect inferencing:
    1. translate to another rule system, e.g., {XSB, Jess, CommonRules, or Jena}
    2. run inferencing in that system’s engine
    3. translate back
  - Can use composite translators

SweetRules V2.0: Indirect Inferencing Engines

Key: ↑ = SweetRules <u>raises</u> power



SweetRules V2.0 New Inferencing Engines



Key: ↑ = SweetRules raises power

KIF (FOL -subset)

Courteous Compiler

#1 CommonRules (fwd. SCLP)

#2 ↑fwd. SCLP Jess/CLIPS (prodn. ≡ fwd. SOLP)

RuleML (SCLP)

SWRL (Horn)

#4 ↑fwd. SCLP XSB (bkw. OLP)

& bkw. CLP #3

Smodels (fwd. OLP)

Process Handbook (OO/frame def.-inh)

#5 ↑+ SWRL built-ins Jena-2 (fwd. Horn LP)

OWL (-DLP)

SweetRules Capabilities Today Cont.'d

- Authoring and Testing front-end: *currently less mature, more partial*
  - Command-line UI
    - *Future: Dashboard GUI with set of windows*
  - Edit rulebases. Run translations. Run inferencing. Compare.
  - Edit in RuleML. Edit in other rule systems' syntaxes. Compare.
  - View human-oriented presentation syntax. View XML/RDF markup syntax.
  - Protégé OWL Plug-in Enhancement
    - SWRL Rule Editor (separate component from SweetRules)

- Analyzers incl. Validators: *currently less mature, more partial*
  - Detect violations of expressive restrictions, e.g., required syntax
  - Misc. other kinds of analyzers
    - e.g., DiffFacts for incremental reasoning
  - Some validators & analyzers as part of various translator & inferencing components
    - e.g., in SweetOnto, SweetXSB, SweetJess

SweetRules Components Today

- Some components have distinct names (for packaging or historical reasons): E.g.,
  - SweetCR translation & inferencing RuleML ↔ CommonRules
  - SweetXSB translation & inferencing RuleML ↔ XSB
  - SweetJess translation & inferencing RuleML ↔ Jess
  - SweetOnto translation {RuleML, SWRL} ← OWL + RDF-facts
  - SweetJena translation & inferencing SWRL → Jena-2
- Other Project Components: (separate codebases for licensing or other reasons)
  - SWRL Built-Ins library *Currently:* for Jena-2
  - SweetPH translation RuleML ← Process Handbook (OO/frame ontologies)
    - *Currently V1.2 is running. Separately downloadable V2 is in progress.*
  - Protégé OWL Plug-in authoring SWRL rules (Horn, referencing OWL)
    - Enhancement providing SWRL Rules authoring is part of the Plug-In.
  - SWRL Validator

Novel NAF Capability in Production Rules I

- Newly Supports Correct Negation-As-Failure in Production Rules
  - Problem: Jess does not correctly implement Negation-As-Failure
    - Conjecture: this problem is shared by all current production rule systems (OPS5-heritage family, based on Rete)
      - *Currently investigating this conjecture.*
  - Solution: We have developed two new techniques with associated KR proof/model theory
    - Stratified case of NAF: declare stratification-based salience in the production rules, when translating from RuleML
      - *Is implemented in SweetRules V2.0 (SweetJess component). Works correctly in all initial phase tests. More testing is in progress.*

Novel NAF Capability in Production Rules II

- General non-stratified case of NAF: new bottom-up algorithm for well founded semantics of OLP
  - *Currently detailed algorithm has been designed and is being implemented.*

- Observation on Additional Value-add: This eliminates the need for agenda meta-rules hacking to get NAF right in production rules, which is frequent in existing production rule applications (and is part of training/methodology)
  - *Interesting Question: How big a percentage of overall agenda meta-rules in typical applications are thus eliminated? Most?*

More Novel Capabilities

- Newly Uses Courteous Compiler to support Courteous feature (prioritized conflict handling) even in systems that don’t directly support it, as long as they support negation-as-failure
  - E.g., XSB Prolog, Jess, Smodels
  - Uses Courteous Compiler component from IBM CommonRules
- New Include-a-KB mechanism, similar to owl:imports Has Include-a-KB mechanism, similar to owl:imports (prelim. RuleML V0.9)
  - Include a remote KB that is translatable to RuleML
- Uses New Action Launcher component to support Situated effecting feature (actions triggered by conclusions) even in systems that don’t directly support it. Facts input, actions output.

Additional Firsts in Implementation

- SWRL/RuleML Built-Ins: (which are based largely on XML-Schema operations)
  - In SweetJena *(in progress: also in rest of SweetRules)*
- Forward Situated Courteous LP inferencing+action with intrinsically highly scaleable run-time performance
    - Both XSB/Prolog and Jess/Rete/production-rules reportedly scale very well to very large rulebases (~100K+ non-fact rules, many Millions facts)
  - Restrictions: Stratified NAF, function-free
  - SweetXSB forward-direction engine
    - Uses Query-All-Predicates, Action Launcher techniques
    - *Currently:* Restriction from XSB: sensing limited to built-ins
  - SweetJess engine
    - *Currently:* Restriction from Jess: all-bound-sensors (includes built-ins)
- Backward Courteous LP inferencing for general non-stratified NAF, and scaleably in above sense
  - SweetXSB backward-direction engine
    - *Currently:* Restriction from XSB: sensing limited to built-ins

Novel KB Merging of Rules + Ontologies

- Combine:
  - Multiple SCLP RuleML (/ SWRL) rulebases
    - Or any knowledge base that is translatable into RuleML
  - Heterogeneous kinds of rules
    - E.g., originally XSB rules + Jess facts
    - These get translated and union’d into a single RuleML rulebase (possibly virtual)
  - OWL ontologies
    - Translate Description Logic Programs (DLP) subset of OWL into RuleML
    - Hybrid reasoning via DLP-fusion, i.e., LP inferencing after translate
  - OO/Frame ontologies with default inheritance
    - E.g., Process Handbook ontologies
    - … which get translated to (S)CLP rules

Novel Integration Framework

- Pluggability & Composition Framework Architecture with detailed interfaces
  - Add your own translator/inferencing-engine/authoring/testing tools
    - We've used this to integrate previous existing translators, and some of our new translators
      - Found it to be easy! How about you?
  - Compose tools automatically, e.g.:
    - translator1 ⊗ translator2
    - translator ⊗ inferencing-engine
  - Search for tools

Object Models for Rules/Ontologies

- SweetRules uses popular API’s & Tools Underneath to manipulate SW markup object models

API/Tool	Kind of Object Model
Jena	OWL, RDF
Protégé (API)	SWRL -RDF
JAXB	RuleML/SWRL -XML
XSLT	RuleML/SWRL -XML

E.g., the predicate-dependency graph and stratifier for SweetJess NAF handling was easily built out of the JAXB object model.

Measuring Power, Elegance and Reuse

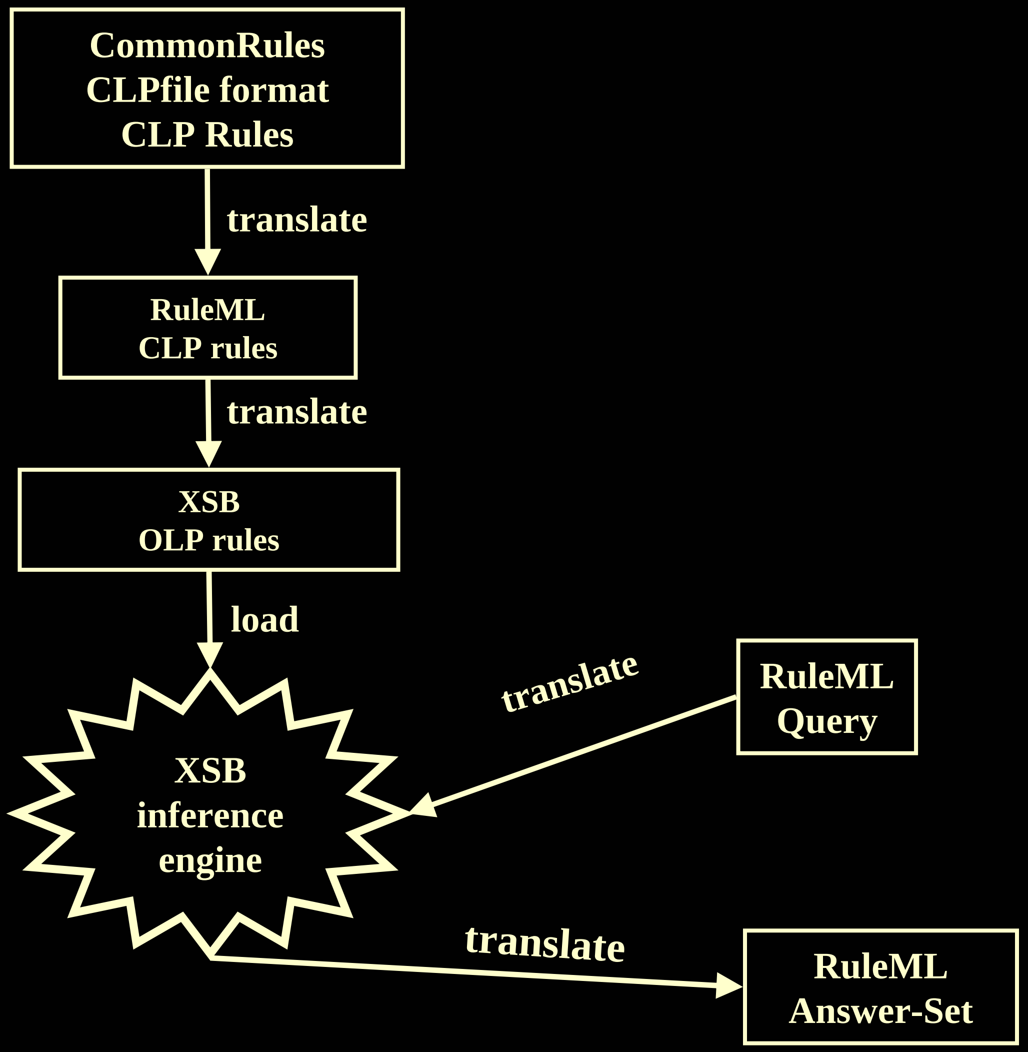
- Significant increases in KR expressiveness of (semantically correct) translation and inferencing relative to previous tools/approaches
  - Production rules join the party of SW and interoperability
  - Correct negation/nonmonotonicity in production rules without extensive agenda meta-rules hacking
  - Courteous extensions of commercial-grade inferencing engines for Prolog and production rules
- Significant increases in scaleability of forward and backward inferencing for (S)CLP
- Weighted coverage: Support the commercially most important kinds of rule systems (production rules, Prolog) for both translation and inferencing

- 10+ diverse KR languages/systems/formats supported
  - Half pre-SW, Half SW
- 20 simple translators; + composite translators
- 5 indirect inferencing engines
- All in code base of 23K Lines Of Code, built mostly in 6 months.
  - MUCH less than the total size of the interoperated systems

SweetRules V2   Demo Outline

- Pacifism (Quakers and Republicans)
  - Translation and CLP inferencing
  - SweetCR, SweetXSB backward (with RuleML answersets)
- Ordering Lead Time (e-commerce policies and notification)
  - KB Merging
  - Hybrid reasoning combining SCLP rules with DLP OWL ontologies
  - Effecting (actions)
  - SweetOnto, SweetJess, SweetXSB forward
- Search and compose translators within SweetRules repository
- Genealogy (family relationships, e.g., uncle-of)
  - Hybrid reasoning combining SWRL rules with DLP OWL ontologies, plus SWRL/RuleML built-ins and Protégé-created SWRL rules
  - SweetJena, Protégé SWRL editor, SWRL builtins, SweetOnto
- SweetDeal E-Contracting Application using SweetRules (supply chain)
  - SCLP RuleML rules that include DLP OWL ontologies

Quaker Example Demo Flow



CommonRules
CLPfile format
CLP Rules

translate

RuleML
CLP rules

translate

XSB
OLP rules

load

XSB
inference
engine

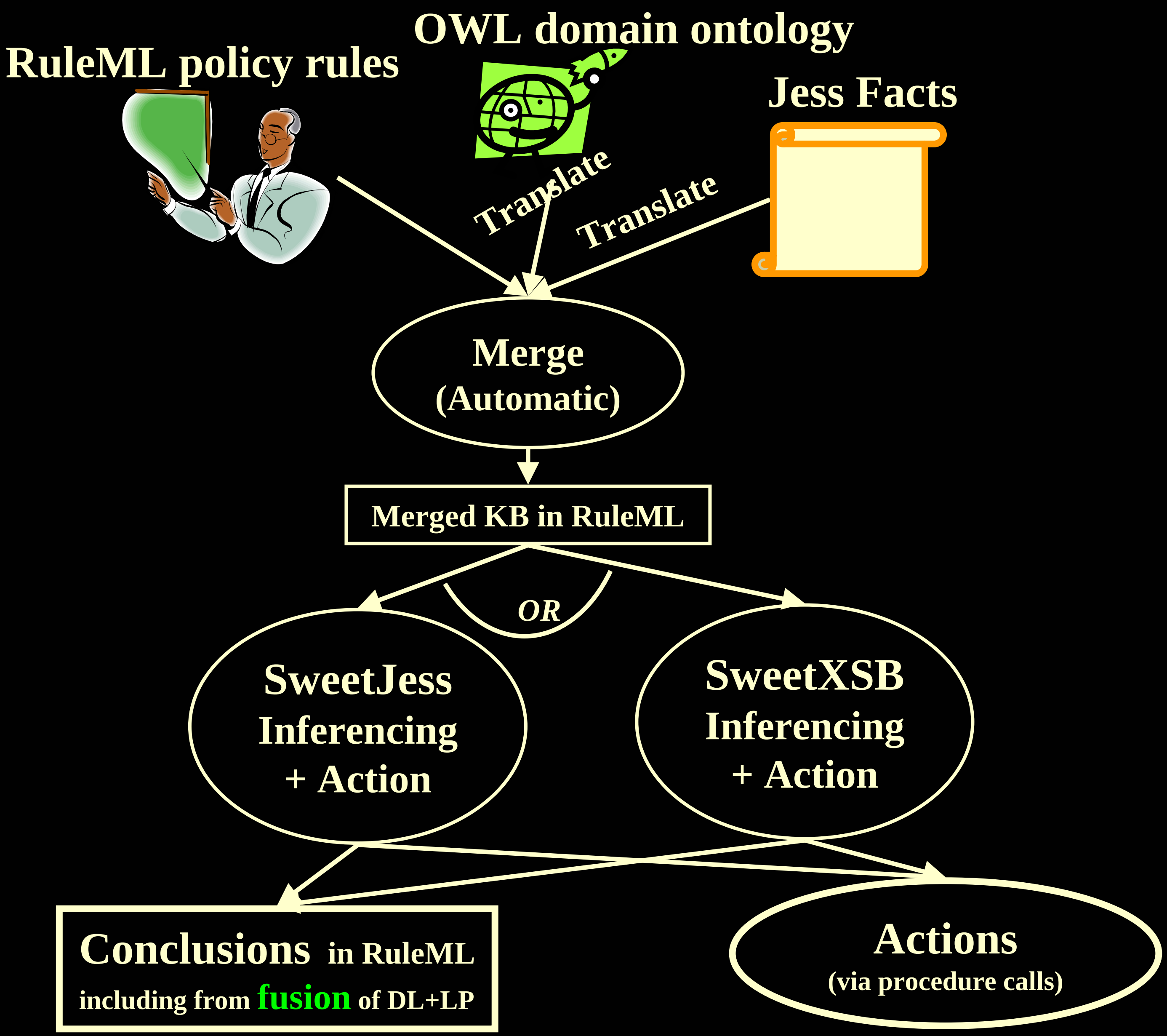
translate

RuleML
Query

translate

RuleML
Answer-Set

OrderingLeadTime Example Demo Flow



RuleML policy rules

OWL domain ontology

Jess Facts

Translate

Translate

Merge (Automatic)

Merged KB in RuleML

OR

SweetJess Inferencing + Action

SweetXSB Inferencing + Action

Conclusions in RuleML including from **fusion** of DL+LP

Actions (via procedure calls)

SweetDeal V2   Demo Outline

- SweetDeal E-Contracting Application using SweetRules (supply chain)
  - SCLP RuleML that include DLP OWL ontologies
  - Contract proposals/final-agreements are SCLP RuleML rulebases that reference/include OWL ontologies
  - Humans edit & communicate, supported by automated agents
  - Proposal evaluation supported by inferencing
  - Agreed business process is executable via inferencing+action

SweetRules V2   Demo Examples

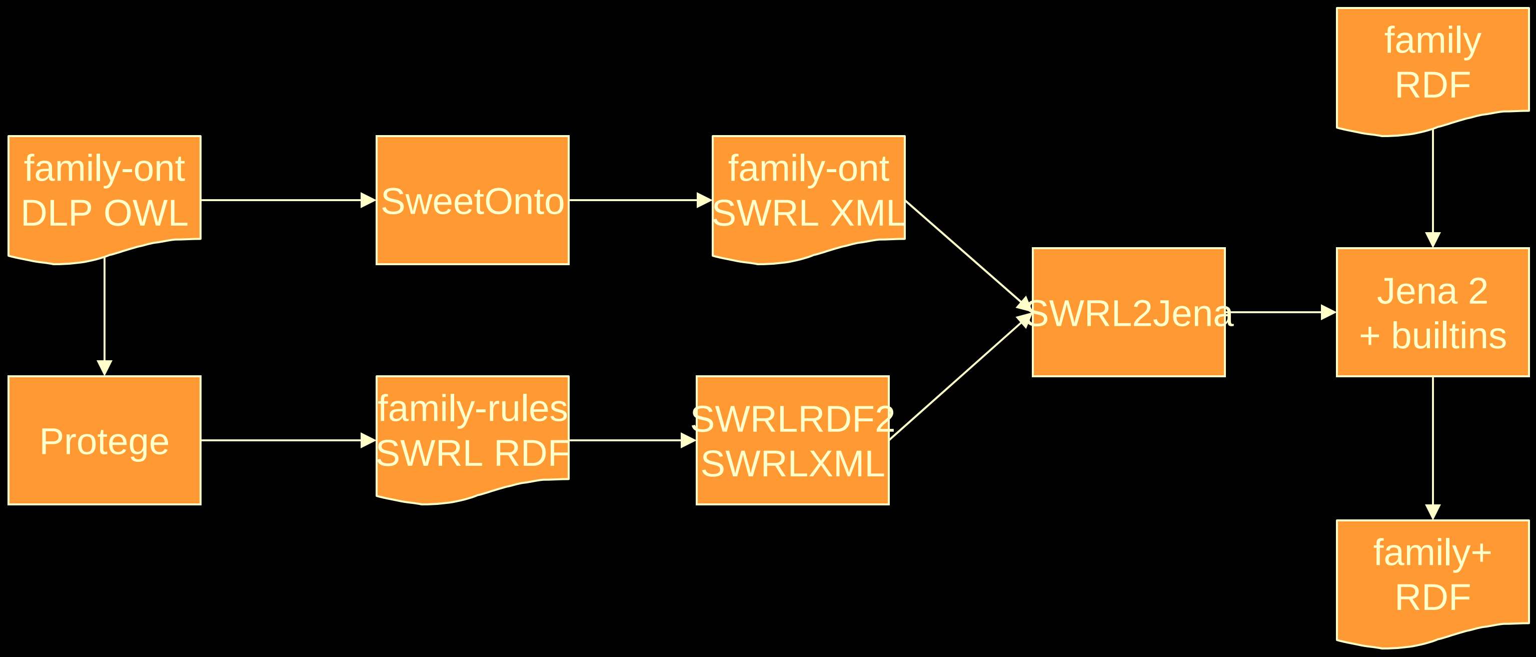
- See separate SweetRules V2   demo examples material.

SWRL-y SweetRules V2   Demo by Mike Dean

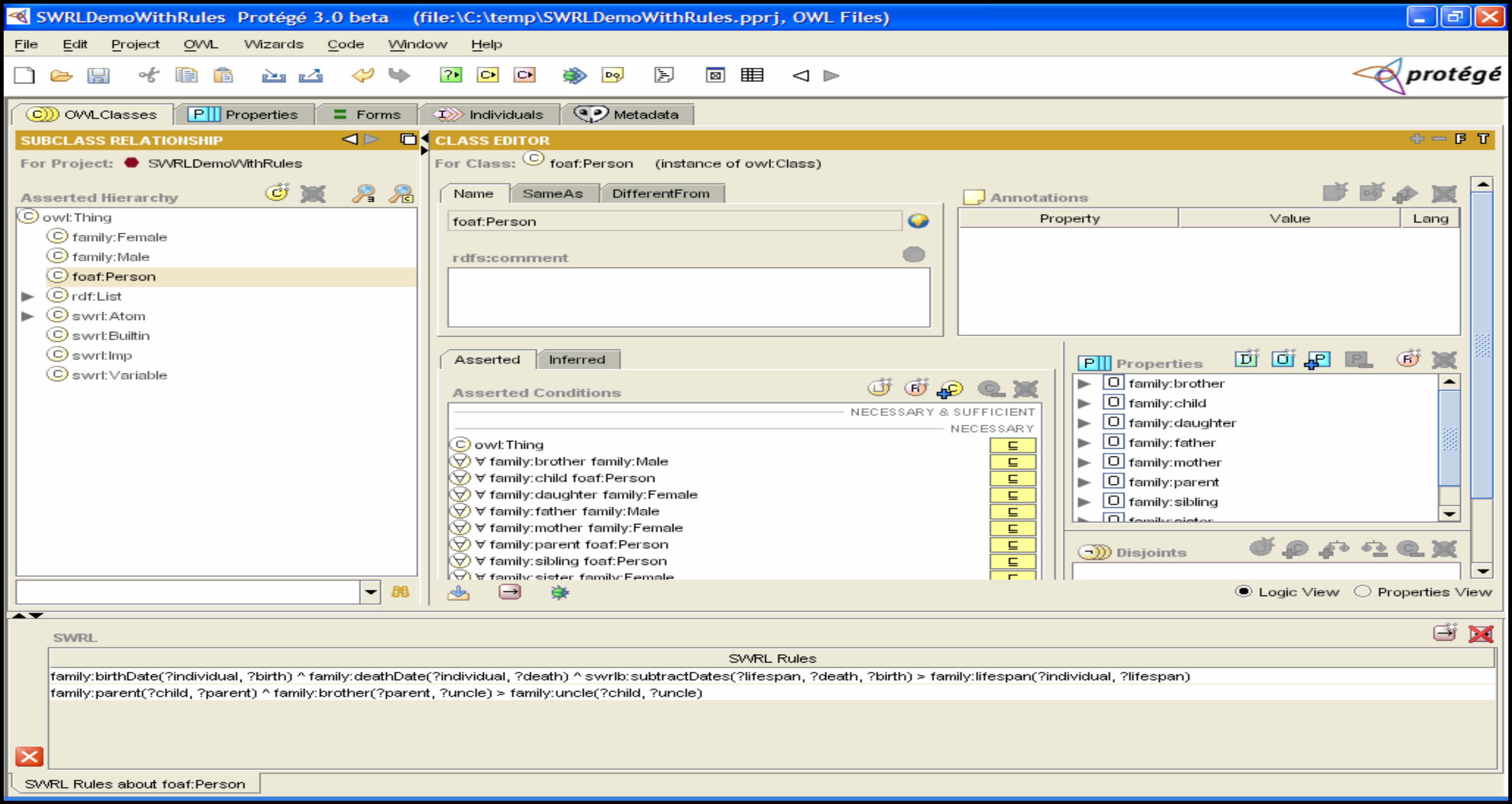
SLIDES FOLLOW

- And also see separate SweetRules V2   demo examples material.

Protégé/SWRL/Jena Demo



Protégé Ontology and Rules



SWRLDemoWithRules Protégé 3.0 beta (file:\C:\temp\SWRLDemoWithRules.pprj, OWL Files)

For Project: SWRLDemoWithRules

Asserted Hierarchy

- owl:Thing
  - family:Female
  - family:Male
  - foaf:Person
  - rdf:List
  - swrl:Atom
  - swrl:Builtin
  - swrl:Imp
  - swrl:Variable

For Class: foaf:Person (instance of owl:Class)

foaf:Person

rdfs:comment

Property	Value	Lang

Asserted Conditions

NECESSARY & SUFFICIENT

NECESSARY

owl:Thing	⊑
∀ family:brother family:Male	⊑
∀ family:child foaf:Person	⊑
∀ family:daughter family:Female	⊑
∀ family:father family:Male	⊑
∀ family:mother family:Female	⊑
∀ family:parent foaf:Person	⊑
∀ family:sibling foaf:Person	⊑
∀ family:sister family:Female	⊑

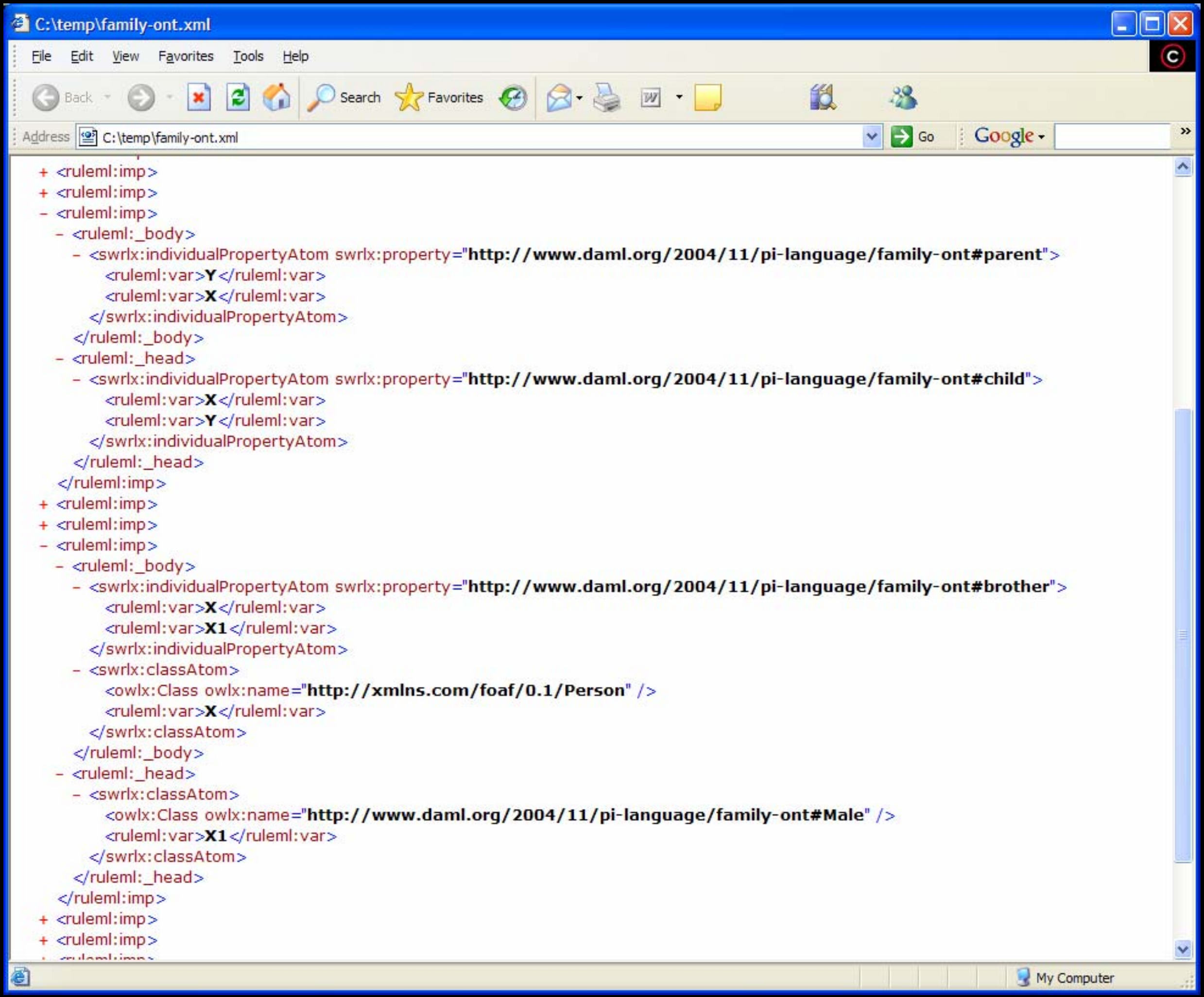
- family:brother
- family:child
- family:daughter
- family:father
- family:mother
- family:parent
- family:sibling
- family:sister

● Logic View ○ Properties View

SWRL Rules
family:birthDate(?individual, ?birth) ^ family:deathDate(?individual, ?death) ^ swrlb:subtractDates(?lifespan, ?death, ?birth) > family:lifespan(?individual, ?lifespan)
family:parent(?child, ?parent) ^ family:brother(?parent, ?uncle) > family:uncle(?child, ?uncle)

SWRL Rules about foaf:Person

family-ont rules from SweetOnto



```
+ <ruleml:imp>
+ <ruleml:imp>
- <ruleml:imp>
  - <ruleml:_body>
    - <swrlx:individualPropertyAtom swrlx:property="http://www.daml.org/2004/11/pi-language/family-ont#parent">
        <ruleml:var>Y</ruleml:var>
        <ruleml:var>X</ruleml:var>
      </swrlx:individualPropertyAtom>
    </ruleml:_body>
  - <ruleml:_head>
    - <swrlx:individualPropertyAtom swrlx:property="http://www.daml.org/2004/11/pi-language/family-ont#child">
        <ruleml:var>X</ruleml:var>
        <ruleml:var>Y</ruleml:var>
      </swrlx:individualPropertyAtom>
    </ruleml:_head>
  </ruleml:imp>
+ <ruleml:imp>
+ <ruleml:imp>
- <ruleml:imp>
  - <ruleml:_body>
    - <swrlx:individualPropertyAtom swrlx:property="http://www.daml.org/2004/11/pi-language/family-ont#brother">
        <ruleml:var>X</ruleml:var>
        <ruleml:var>X1</ruleml:var>
      </swrlx:individualPropertyAtom>
    - <swrlx:classAtom>
        <owlx:Class owlx:name="http://xmlns.com/foaf/0.1/Person" />
        <ruleml:var>X</ruleml:var>
      </swrlx:classAtom>
    </ruleml:_body>
  - <ruleml:_head>
    - <swrlx:classAtom>
        <owlx:Class owlx:name="http://www.daml.org/2004/11/pi-language/family-ont#Male" />
        <ruleml:var>X1</ruleml:var>
      </swrlx:classAtom>
    </ruleml:_head>
  </ruleml:imp>
+ <ruleml:imp>
+ <ruleml:imp>
```

family

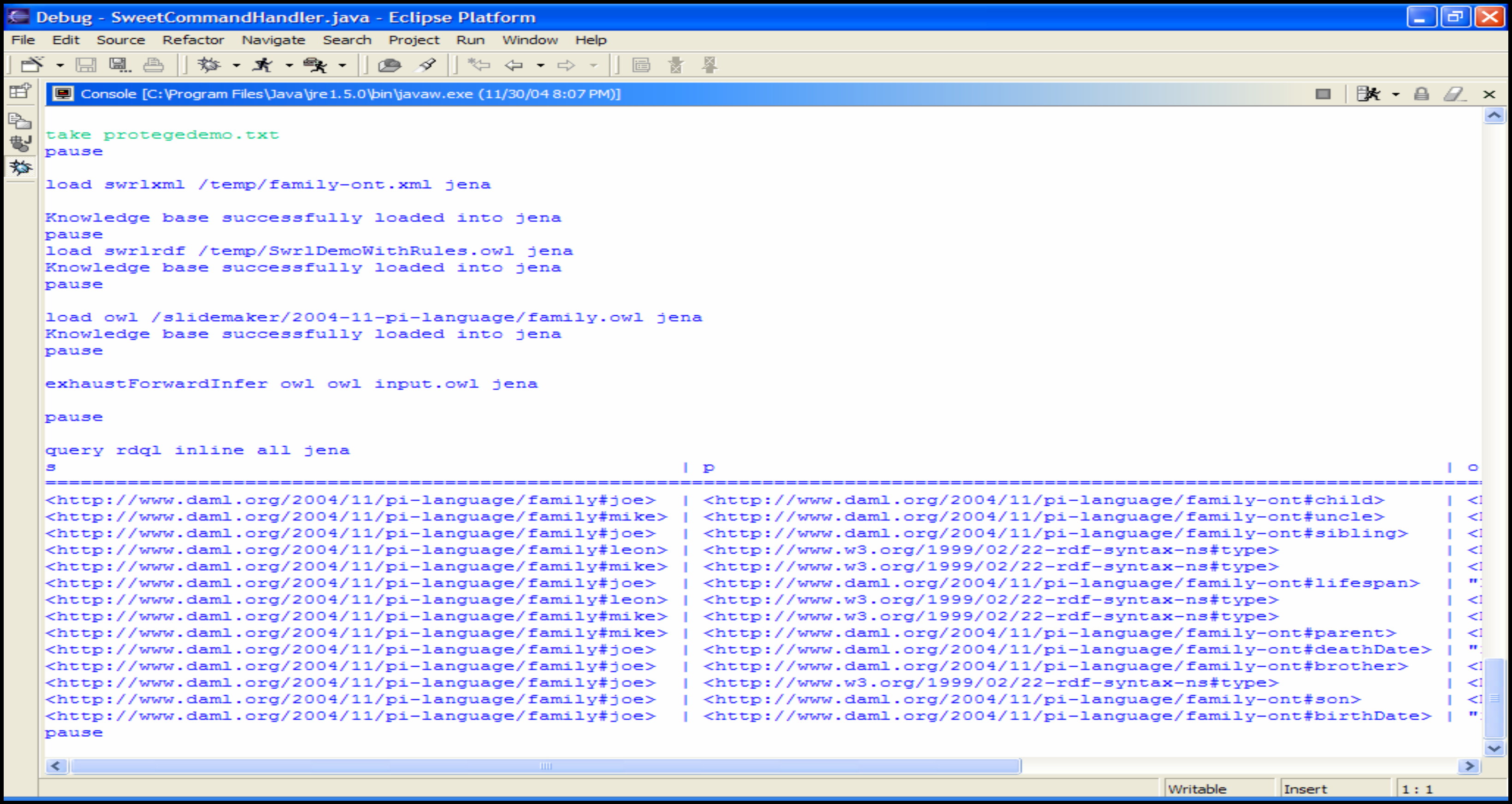
```
<?xml version='1.0' encoding='ISO-8859-1'?>
<!DOCTYPE rdf:RDF [
        <!ENTITY xsd 'http://www.w3.org/2001/XMLSchema#'>
]>

<rdf:RDF
  xmlns:rdf ="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:foaf="http://xmlns.com/foaf/0.1/"
  xmlns:family="http://www.daml.org/2004/11/pi-language/family-ont#"
  xml:base="http://www.daml.org/2004/11/pi-language/family">

<foaf:Person rdf:ID="joe">
  <family:birthDate rdf:datatype="&xsd;date">1923-10-23</family:birthDate>
  <family:deathDate rdf:datatype="&xsd;date">1999-03-17</family:deathDate>
  <family:son rdf:resource="#mike"/>
  <family:brother rdf:resource="#leon"/>
</foaf:Person>

</rdf:RDF>
```

SweetRules Execution


Debug - SweetCommandHandler.java - Eclipse Platform
File Edit Source Refactor Navigate Search Project Run Window Help
Writable
Insert
1 : 1

Console [C:\Program Files\Java\jre1.5.0\bin\javaw.exe (11/30/04 8:07 PM)]

```
take protegedemo.txt
pause

load swrlxml /temp/family-ont.xml jena

Knowledge base successfully loaded into jena
pause
load swrlrdf /temp/SwrlDemoWithRules.owl jena
Knowledge base successfully loaded into jena
pause

load owl /slidemaker/2004-11-pi-language/family.owl jena
Knowledge base successfully loaded into jena
pause

exhaustForwardInfer owl owl input.owl jena

pause

query rdql inline all jena
s                                                     | p                                                              | o
==========================================================================================================================
<http://www.daml.org/2004/11/pi-language/family#joe>  | <http://www.daml.org/2004/11/pi-language/family-ont#child>     | <
<http://www.daml.org/2004/11/pi-language/family#mike> | <http://www.daml.org/2004/11/pi-language/family-ont#uncle>     | <
<http://www.daml.org/2004/11/pi-language/family#joe>  | <http://www.daml.org/2004/11/pi-language/family-ont#sibling>   | <
<http://www.daml.org/2004/11/pi-language/family#leon> | <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>              | <
<http://www.daml.org/2004/11/pi-language/family#mike> | <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>              | <
<http://www.daml.org/2004/11/pi-language/family#joe>  | <http://www.daml.org/2004/11/pi-language/family-ont#lifespan>  | "
<http://www.daml.org/2004/11/pi-language/family#leon> | <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>              | <
<http://www.daml.org/2004/11/pi-language/family#mike> | <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>              | <
<http://www.daml.org/2004/11/pi-language/family#mike> | <http://www.daml.org/2004/11/pi-language/family-ont#parent>    | <
<http://www.daml.org/2004/11/pi-language/family#joe>  | <http://www.daml.org/2004/11/pi-language/family-ont#deathDate> | "
<http://www.daml.org/2004/11/pi-language/family#joe>  | <http://www.daml.org/2004/11/pi-language/family-ont#brother>   | <
<http://www.daml.org/2004/11/pi-language/family#joe>  | <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>              | <
<http://www.daml.org/2004/11/pi-language/family#joe>  | <http://www.daml.org/2004/11/pi-language/family-ont#son>       | <
<http://www.daml.org/2004/11/pi-language/family#joe>  | <http://www.daml.org/2004/11/pi-language/family-ont#birthDate> | "
pause
```

family+

Subject	Predicate	Object	Rationale
family:joe	rdf:type	foaf:Person	
family:joe	family-ont:birthDate	"1923-10-23"^^xsd:date	
family:joe	family-ont:deathDate	"1999-03-17"^^xsd:date	
family:joe	family-ont:son	family:mike	
family:joe	family-ont:brother	family:leon	
family:joe	family-ont:child	family:mike	superproperty
family:joe	family-ont:sibling	family:leon	superproperty
family:joe	family-ont:lifespan	"P27539D"^^xsd:duration	rule
family:mike	rdf:type	family-ont:Male	allValuesFrom
family:mike	rdf:type	foaf:Person	allValuesFrom
family:mike	family-ont:parent	family:joe	inverse
family:mike	family-ont:uncle	family:leon	rule
family:leon	rdf:type	family-ont:Male	allValuesFrom
family:leon	rdf:type	foaf:Person	allValuesFrom

Demonstrated

- Hybrid reasoning with ontologies and rules
- SWRL editing with Protégé
- Transparent chained SweetRules translation
  - OWL DLP to SWRL
  - SWRL RDF to SWRL XML
  - SWRL XML to Jena 2
- Rule execution using Jena 2 with builtins

SweetDeal V2 Demo: Novelty Highlights

1. SweetDeal is the first e-contracting application scenario, and first real e-business application scenario, combining RuleML with OWL. It uses DLP-fusion combining the OWL with RuleML to do combined hybrid inferencing. It combines contract rulesets in RuleML with business process/contract ontologies in OWL.
2. Moreover, SweetDeal is the first to have such contracts contain rules that employ procedural attachments to perform actions (side-effectful) as part of the business processes that the contracts specify.
3. SweetDeal is the first previous application to be refitted to use SweetRules V2 – and the first to be refitted to use DLP-fusion.

- Deltas wrt the previous SweetDeal V1 prototype (of 2002):
  - Uses OWL (previous DAML+OIL); DLP-fusion; procedural attachments for actions; SweetRules as infrastructure

SweetRules: Use Cases Overview

- Trust Policies: authorization, privacy, security, access control
  - E.g., financial services, health care
  - Extensive analysis of business case/value

- Semantic mediation: rule-based ontology translation, context-based information integration

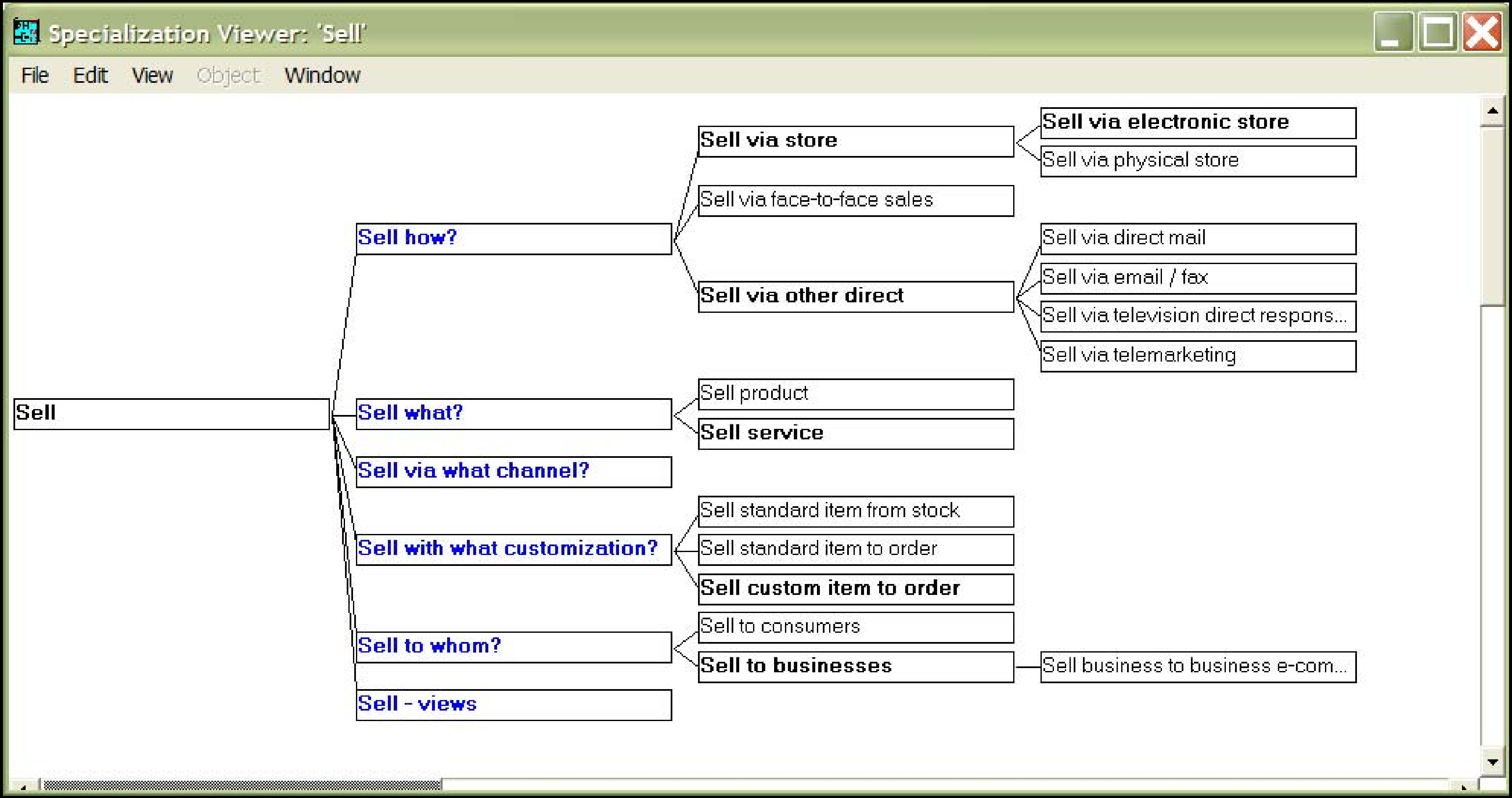
- Contracts/negotiation, advertising/discovery
  - E-procurement, E-selling
  - Pricing, terms & conditions, supply chain, …

- Monitoring:
  - Exception handling, e.g., of contract violations
    - Late delivery, refunds, cancellation, notifications
  - Personal messaging and workflow

Opportunity for Process Handbook in SWS

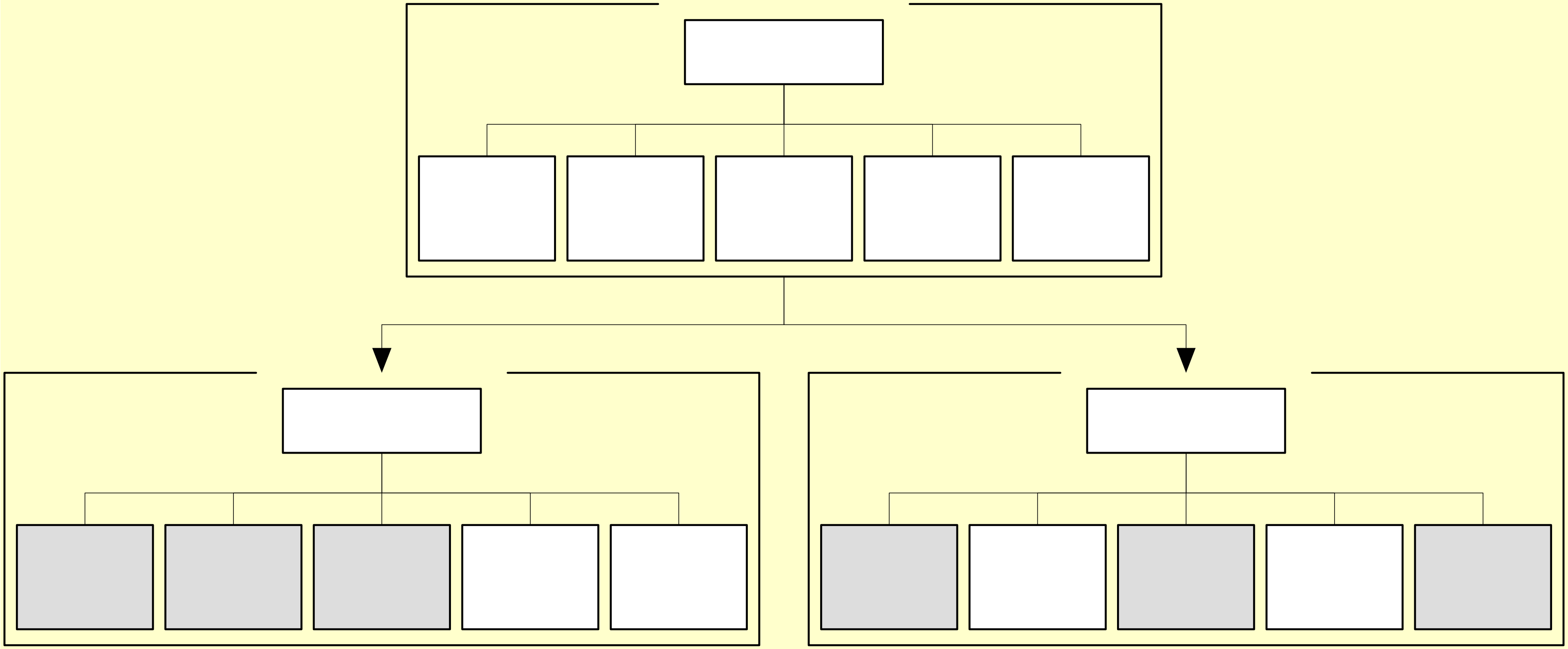
- Need for Shared Knowledge Bases about Web Services / Business Processes
  - For Semantic Web Services, etc.
- Want to leverage legacy process knowledge content
  - Go where the knowledge already is
- Process Handbook (PH) as candidate nucleus for shared business process ontology for SWS
  - 5000+ business processes, + associated class/property concepts, as structured knowledge (http://ccs.mit.edu/ph)
  - E.g., used in SweetDeal E-Contracting prototype
- Concept: Use Semantic Web KR and standards to represent Object-Oriented framework knowledge:
  - class hierarchy, types, generalization-specialization, domain & range, properties/methods' association with classes

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



- Sell
  - Sell how?
    - Sell via store
      - Sell via electronic store
      - Sell via physical store
    - Sell via face-to-face sales
    - Sell via other direct
      - Sell via direct mail
      - Sell via email / fax
      - Sell via television direct respons...
      - Sell via telemarketing
  - Sell what?
    - Sell product
    - Sell service
  - Sell via what channel?
  - Sell with what customization?
    - Sell standard item from stock
    - Sell standard item to order
    - Sell custom item to order
  - Sell to whom?
    - Sell to consumers
    - Sell to businesses
      - Sell business to business e-com...
  - Sell - views

PH Example: Selling Processes



An activity (e.g., SellProduct) has sub-activities (steps).

Its specializations (e.g., SellByMailOrder) inherit its sub-activities by default.

Key: gray = modified (overridden). X = deleted (canceled).

ent

poten

SweetPH's New Technical Approach: Courteous Inheritance for PH & OO

- Surprise: use SW *rule* language not the main SW *ontology* language! I.e., use (SCLP) RuleML not OWL.
  - OO inheritance is *default* ⇒ more reuse in ontologies
  - OWL/FOL cannot represent default inheritance
  - RuleML/nonmon-LP can
- Courteous Inheritance approach translates PH to SCLP KR
  - A few dozen background axioms. Linear-size translation. Inferencing is tractable computationally.
- PH becomes a SWS OO process ontology repository
- *In progress: open source version of PH content*
- *In progress: extend approach to OO ontologies generally*

SweetRules: Plans within DAML program

- Polishing, generally, of doc and code

- SweetPH release
- Non-stratified NAF (WFS) in SweetJess
- More tightly integrate SWRL with RuleML: spec, code

- More application scenarios, esp. services
  - Policies, contracts, mediation, …

SweetRules: Directions *beyond DAML program*

- Hook up to Web Services
  - Importing knowledge bases / modules, procedural attachments, translation/inferencing, events, …
- More on authoring, UI, editors
- Support increased expressiveness of DLP
  - *Later in session:* new theory; services uses
- Support more rule/ontology engines/systems:
  - Tasks: translation, inferencing
  - Flora, cwm, Triple, Hoolet, DRS, ROWL, KAON, JTP, SWI Prolog, …
  - Systems of new/various kinds: ECA, RDF-Query/XQuery, …

SweetRules: Directions beyond the DAML program, cont.'d

- More support of SWSL-Rules, incl. for Hilog, frame syntax features
- More support of FOL
  - FOL RuleML / SWRL FOL / KIF / SCL
- More conflict analysis
- Incremental reasoning, events
- Scaleability performance testing/benchmarking

- *More Collaborators invited!*

SweetRules V2 Team

- Core Team:
  - B. Grosof (MIT Sloan), M. Dean (BBN), S. Ganjugunte (UMBC student), S. Tabet (MIT Sloan), C. Neogy (MIT Sloan)

- Project Lead: B. Grosof.   Project Co-Lead: M. Dean.
- Lead designer of core including SCLP RuleML and DLP OWL aspects: B. Grosof
- Lead implementer of core: S. Ganjugunte
- Lead designer and implementer of SweetJena & several SWRL tools: M. Dean
- Lead implementer of SWRL built-ins: D. Kolas (BBN)
- Lead designers of Protégé Rules Editor enhancement: M. Musen (Stanford), M. O'Connor (Stanford); Project Lead: M. Musen; Lead Implementer: M. O'Connor.
- Lead designers of SweetPH: B. Grosof, A. Bernstein (U. Zurich)
- Lead implementer of SweetPH: A. Bernstein
- Lead designer of SweetDeal application scenario prototype: B. Grosof
- Lead implementer of SweetDeal: S. Bhansali (MIT Sloan student)

- Other Contributors: B. Motik (U. Karlsruhe student), R. Studer (U. Karlsruhe), R. Volz (U. Karlsruhe student); T. Finin (UMBC), A. Joshi (UMBC); J. Bonin (U. Zurich student); T. Poon (MIT student); H. Chan (IBM); H. Boley (NRC/UNB)
- * (This is a preliminary list, we may have forgotten to include someone; if so, apologies!)

ADDITIONAL LONG-VERSION SWEETRULES SLIDES FOLLOW

- These omitted, due to limited time, from the Rules Plenary Session presentation
  of the Nov.-Dec. 2004 DAML PI Meeting.

Rule and Ontology Languages/Systems That Interoperate via SweetRules and RuleML, Today I

1. RuleML
   - Situated Courteous LP extension, V0.8+
2. XSB (the pure subset of it = whole Ordinary LP)
   - Backward. Prolog. Fast, scalable, popular. Good support of SQL DB's (e.g., Oracle) via ODBC backend. Full well-founded-semantics for OLP. Implemented in C. By SUNY Stonybrook. Open source on sourceforge. Well documented and supported. Papers.
3. Jess (a pure subset of it = a large subset of Situated Ordinary LP)
   - Forward. Production Rules (OPS5 heritage). Flexible, fast, popular. Implemented in Java. By Sandia National Labs. Semi-open source, free for research use. Well documented and supported. Book.
   - *Uses recent novel theory for translation between SOLP and Production Rules.*

Rule and Ontology Languages/Systems That Interoperate via SweetRules and RuleML, Today II

4. IBM CommonRules (whole = large subset of stratified SCLP)
   - Forward. SCLP. Implemented in Java. Expressive. By IBM Research. Free trial license, on IBM AlphaWorks (since 1999). Considerable documentation. Papers. Piloted.
   - Implements the Courteous Compiler (CC) KR technique.
     - which reduces (S)CLP to equivalent (S)OLP, tractably.
   - Includes bidirectional translators for XSB, KIF, Smodels.
   - Its overall concept and design was point of departure for several aspects of SweetRules

5. Knowledge Interchange Format (KIF) (a subset of it = an extension of Horn LP)
   - First Order Logic (FOL). Semi-standard, morphing into Simplified Common Logic ISO standard. Several tools support, e.g., JTP. Research language to date.
     - Note: FOL is superset of DLP and of SWRL's fundamental KR.

Rule and Ontology Languages/Systems That Interoperate via SweetRules and RuleML, Today III

6. OWL (the Description Logic Programs subset)
   - Description Logic ontologies. W3C standard. Several tools support, e.g., FACT, RACER, Jena, Hoolet, etc.
   - *Uses recent novel DLP theory for translation between Description Logic and Horn LP.*
7. Process Handbook (large subset = subset of SCLP)
   - Frame-style object-oriented ontologies for business processes design, i.e., for services descriptions. By MIT and Phios Corp. (spinoff). Large (5000 business processes). Practical, commercial. Good GUI. Open source license in progress. Available free for research use upon request. Includes extensive textual information too. Well documented and supported. Papers. Book. Dozens of research users.
   - *Uses recent novel SCLP representation of Frames with multiple default inheritance.*
8. Smodels (NB: somewhat old version; large subset = finite OLP)
   - Forward. Ordinary LP. Full well-founded-semantics or stable semantics. Implemented in C. By Helsinki univ. Open source. Research system.

Rule and Ontology Languages/Systems That Interoperate via SweetRules and RuleML, Today IV

9. Jena-2 *currently only with SWRL, DLP OWL*
   - Forward and backward. Subset of Datalog Horn LP. Plus builtins. Plus RDF & (subset) OWL support. Implemented in Java. By HP. Open source. Popular SW toolkit.
10. SWRL V0.6 *currently only with DLP OWL, Jena-2, Jess/CLIPS*
    - XML syntax (initially). Named-classes-only subset – i.e., Datalog unary/binary Horn FOL. Essentially a subset of RuleML *(in progress: tight convergence).*

SweetRules Translator Capabilities Today

- Translators in and out of RuleML:
  - RuleML ↔ {XSB, Jess, CommonRules, KIF, Smodels}
  - RuleML ← {OWL, Process Handbook} (one-direction only)
  - SOLP RuleML ← SCLP RuleML (Courteous Compiler)
- Translators in and out of SWRL Rules (NB: SWRL Rules is essentially subset of RuleML):
  - SWRL ← OWL (one-direction only)
  - Jena-2 ← SWRL (one-direction only)
  - Jess/CLIPS ← SWRL (one-direction only)
  - *More to come – tighter integration between RuleML and SWRL*
- Composite Translators, e.g.,
  - {XSB, Jess, Jena-2, CommonRules, KIF, Smodels} ← OWL ;
  - Jess ↔ {XSB, CommonRules} ; …

SweetRules Inferencing Components Today I

- Inferencing engines in SCLP RuleML / SWRL via translation:

1. SweetCR: Forward Situated Courteous LP
    - Restrictions from CommonRules: stratified NAF; *currently (due to CR bug)* limited sensing (built-ins only); slow performance
2. SweetXSB: Backward Courteous LP (+ built-ins)
    - Uses Courteous Compiler technique
    - Supports general non-stratified Negation-As-Failure (Well Founded Semantics), using XSB capability
    - Intrinsically highly scaleable run-time performance
        - XSB reportedly scales very well to very large rulebases (~100K+ non-fact rules, many Millions facts)

SweetRules Inferencing Components Today II

3. SweetXSB: Forward Situated Courteous LP
   - Uses Query-All-Predicates technique to support forward-direction. Uses backward SweetXSB engine.
     - Restriction from being forward: limited recursion through functions
       - *Currently*: function-free
   - Uses Action Launcher technique for effecting (actions)
   - *Currently:* Restriction from XSB: limited sensing (built-ins only)
   - As in backward SweetXSB: uses Courteous Compiler; supports general NAF (WFS); intrinsically highly scaleable run-time performance

SweetRules Inferencing Components Today III

4. SweetJess: Forward Situated Courteous LP

- Uses our recent novel theory on translating between Situated Ordinary LP and production rules
  - Uses novel technique for NAF to remedy Jess/Rete limitations
- Uses Courteous Compiler technique
- *Currently:* Restriction: stratified NAF.
  - *In progress: general non-stratified NAF (WFS)*
- Restrictions from production rules: function-free; all-bound-sensors
- Intrinsically highly scaleable run-time performance
  - Jess/Rete reportedly scales very well to very large rulebases (~100K+ non-fact rules, many Millions facts)

SweetRules   Inferencing Components Today   IV

5. SweetJena: Forward Horn LP (+ built-ins)
    - SWRL/RuleML rules, using Jena forward engine
    - Supports SWRL/RuleML built-ins
        - Uses recent SWRL/RuleML built-ins syntax (which are based largely on XML-Schema datatype operations)
        - Uses new implemented library of built-ins
    - Restrictions from Jena: unary/binary predicates, function-free, Horn (NAF-free)
        - *In progress: general non-stratified NAF (WFS)*
    - Direct access to RDF fact store, using Jena capability

More about Combining Rules with Ontologies

There are several ways to use SweetRules to combine rules with ontologies:

1. By reference: via URI as name for predicate
2. Translate DLP subset of OWL into RuleML (or SWRL)
    - Then can add SCLP rules
        - E.g., add Horn LP rules and built-in sensors
          ⇒ interesting subset of the SWRL V0.6 KR
        - E.g., add default rules or procedural attachments
3. Translate non-OWL ontologies into RuleML
    - E.g., object-oriented style with default inheritance
        - E.g., Courteous Inheritance for Process Handbook ontologies
4. Use RuleML/SWRL Rules to map between ontologies
    - E.g., a number of SWRL use cases
    - E.g., in the spirit of the Extended COntext Interchange (ECOIN) approach/system.

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 was available free via Web/email
- SweetJess V2 implementation available Nov. 2004 open source on SemWebCentral as part of SweetRules V2

SweetJess: Translating an Effector Statement

```
<effe>
  <_opr>
   <:rel>giveDiscount</damlRuleML:rel>
  </_opr>
  <_aproc>
   <jproc>
     <meth>setCustomerDiscount</meth>
     <clas>orderMgmt.dynamicPricing</clas>
     <path>com.widgetsRUs.orderMgmt</path>
    </jproc>
  </_aproc>
</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 B. Grosof paper
  - “Representing E-Commerce Rules Via Situated Courteous Logic Programs in RuleML”, in *Electronic Commerce Research and Applications* journal, 2004
  - Available at http://ebusiness.mit.edu/bgrosof

Objectives for Integrating Distributed SW Rules and Ontologies, Motivating SweetRules I

Address “the 5 D’s” of real-world reasoning ⇒ *desired improvements*:

1. **Diversity** – Existing/emerging kinds of ontologies and rules have heterogeneous KR's. *Handle more heterogeneous systems.*
2. **Distributedness** - of ownership/control of ontology/rule active KB's. *Handle more source active KB’s.*
3. **Disagreement** - Conflict (contradiction) will arise when merging knowledge. *Handle more conflicts.*
4. **Dynamism** - Updates to knowledge occur frequently, overturning previous beliefs. *Handle higher rate of revisions.*
5. **Delay** - Computational scaleability is vital to achieve the promise of knowledge integration. *Achieve Polynomial-time (~ databases).*

Objectives for Integrating Distributed SW Rules and Ontologies,

Motivating SweetRules II

BEFORE

AFTER

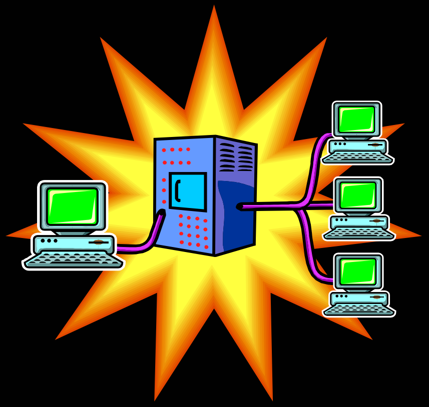
Contradictory conflict is globally contagious, invalidates all results.

⇒

Contradictory conflict is contained locally, indeed tamed to aid modularity.

Knowledge integration tackling the 5 D's (esp. diversity and distributedness) is labor-intensive, slow, costly.

⇒



Knowledge integration is highly automated, faster, cheaper.

OPTIONAL SWEETRULES SLIDES FOLLOW

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.,
  - Jess, ILOG, Blaze, 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.)*

Open Source pre-SW Rule Tools: Popular, Mature

- XSB Prolog [SUNY Stonybrook]
  - Supports Well Founded Semantics for general, non-stratified case
  - Scales well
  - C, with Java front-end available (InterProlog)

- Jess production rules [Sandia Natl. Lab USA]
  - Semi-open source
  - Java
  - Successor to: CLIPS in C [NASA]

- SWI Prolog [Netherlands]

Overview of SW Rule Tool Generations

Analysis: 3 Generations of SW rule tools to date

1. Rudimentary Interoperability and XML/RDF Support
   - CommonRules, SweetRules V1, OWLJessKB

2. Rule Systems within RDF/OWL/SW Toolkits
   - cwm, Jena-2, and others – incl. SWRL tools

3. SW Rule Integration and Life Cycle
   - SweetRules V2

END OF:

ADDITIONAL LONG-VERSION SWEETRULES SLIDES FOLLOW