

*Overview of SweetRules V2.1:
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/bgrosoref>*

***BBN Technologies, <http://www.daml.org/people/mdean>*

*This is updated from an earlier version
which was a section of the presentation:
“DAML Rules Report for PI Mtg. Nov.-Dec. 2004”*

by Benjamin Grosf and Mike Dean***

(DAML Rules Co-Chairs)

**MIT Sloan School of Management, <http://ebusiness.mit.edu/bgrosf>*

***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 V2.1 Release was Monday Apr. 25 2005.
 - V2.0 release was Monday Nov. 29 2004.
- Open-source on SemWebCentral.org
 - <http://sweetrules.projects.semwebcentral.org>
- Press release available there
 - full 2-page or short 1-page versions

*For V2.1 Delta Beyond
V2.0:
SEE later section of this
presentation*

- Approximately slides 45-51

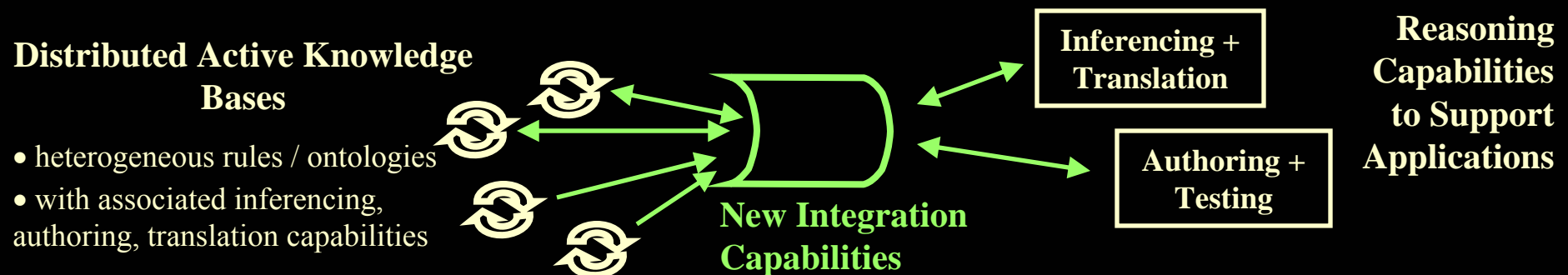
SweetRules V2 Overview

Key Ideas:

- Unite the commercially most important kinds of rule and ontology languages via 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 \leftrightarrow 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.



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, ...

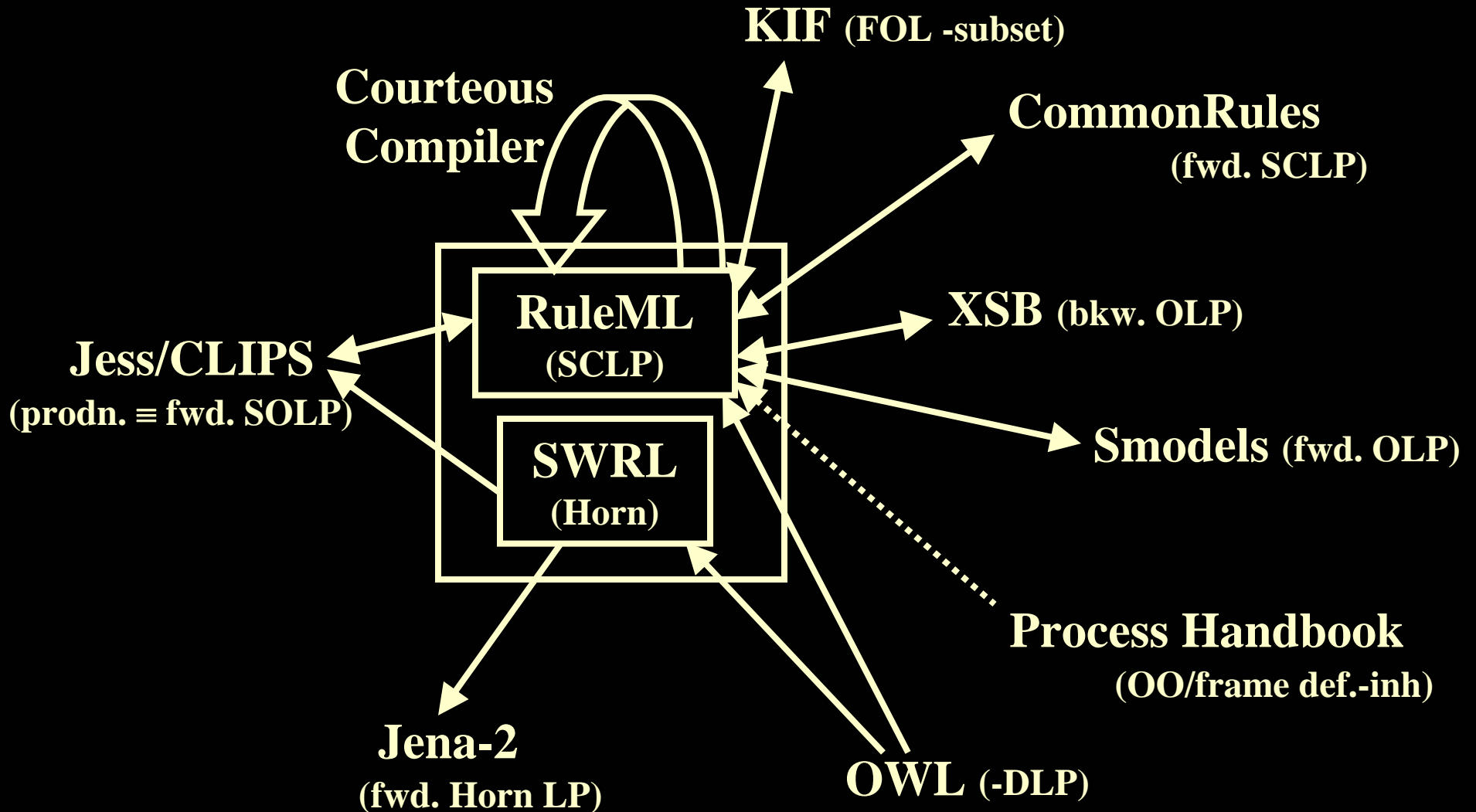
SweetRules V2.0+ Fundamental KR Today

- 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)
 - Running SweetPH prototype tool already implemented. Being tested. Will be separate codebase for licensing reasons.

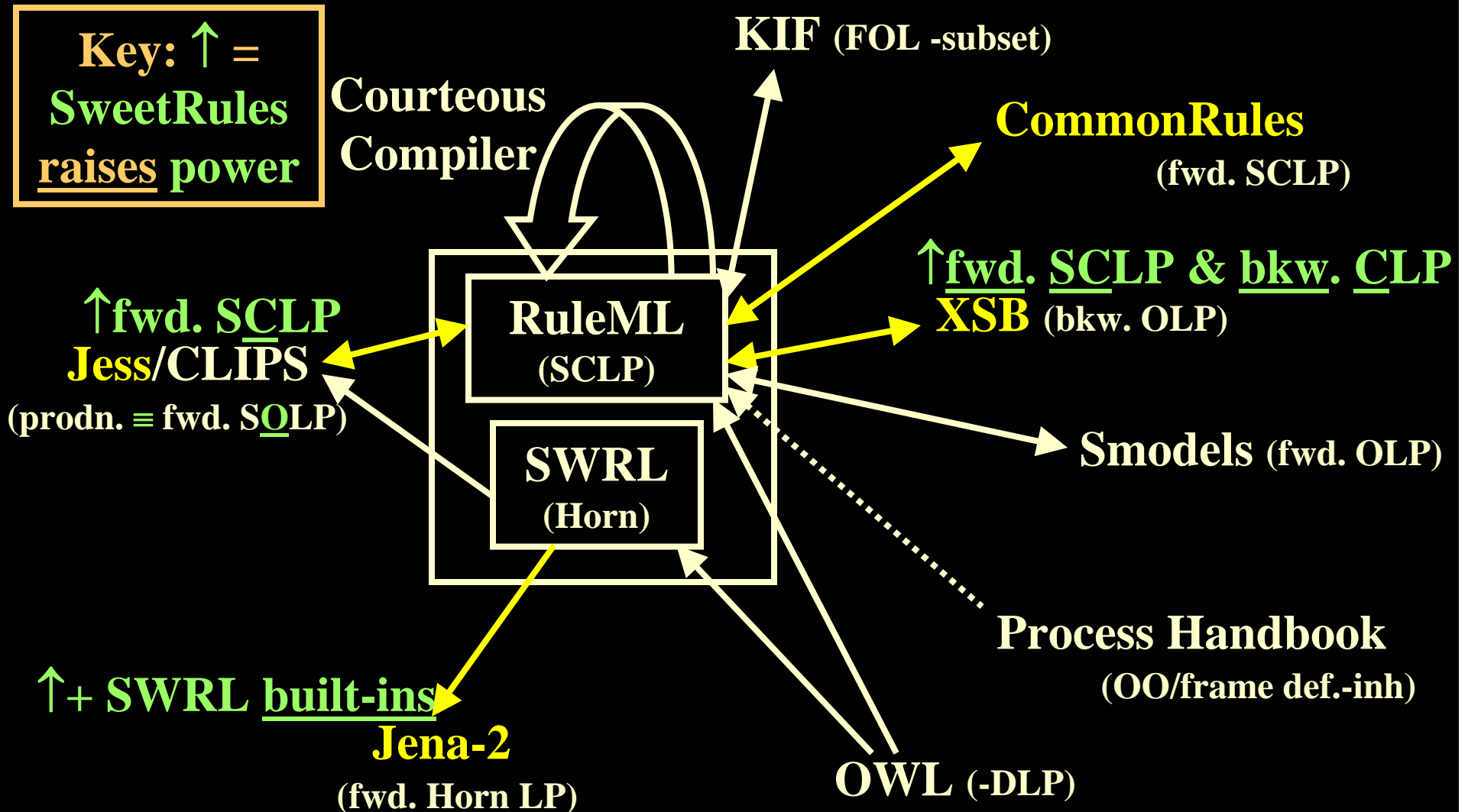
SweetRules Today: Translators Graph



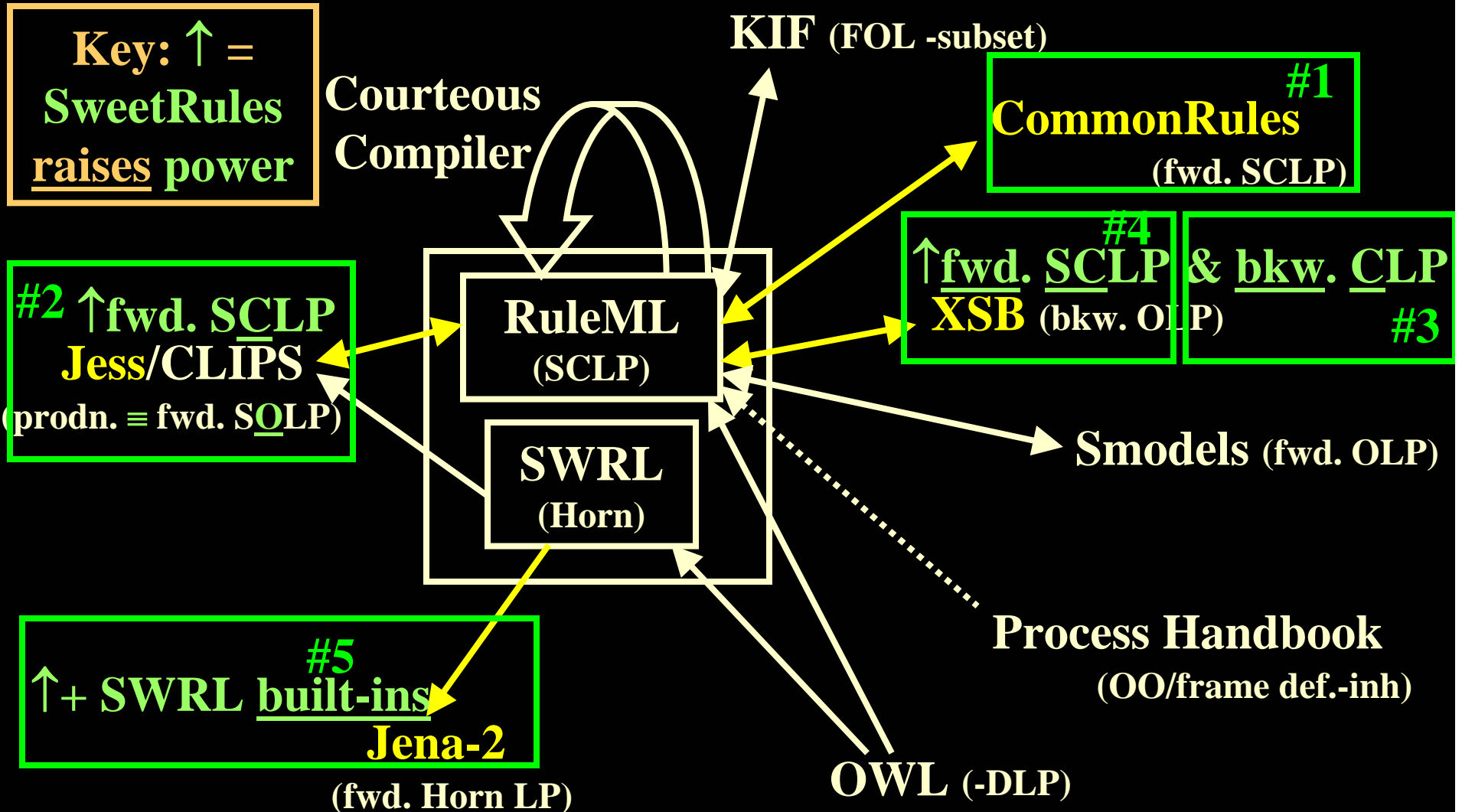
SweetRules Inferencing Capabilities Today: Overview

- **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



SweetRules V2.0+ *New Inferencing Engines*



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 \leftrightarrow CommonRules
 - **SweetXSB** translation & inferencing RuleML \leftrightarrow XSB
 - **SweetJess** translation & inferencing RuleML \leftrightarrow Jess
 - **SweetOnto** translation {RuleML, SWRL} \leftarrow OWL + RDF-facts
 - **SweetJena** translation & inferencing SWRL \rightarrow Jena-2
 - **SweetCourteousCompiler** translation Courteous LP \rightarrow Ordinary LP
- Other Project Components: (separate codebases for licensing or other reasons)
 - **SWRL Built-Ins** library *Currently:* for Jena-2
 - **SweetPH** translation RuleML \leftarrow 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
 - *Is implemented in SweetRules V2.1+ (SweetJess component)*
 - *Exploits capabilities of, and has optimizations for, production rules*
- 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., in XSB Prolog, Jess, Smodels
 - Uses Native Open-Source Courteous Compiler (CC) or CC from IBM CommonRules
- **New 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.
 - E.g., in SweetXSB forward inferencing

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

Additional Firsts in Implementation

- RuleML Presentation Syntax Support:
 - First implementation of the presentation syntax
 - Initially generator not yet parser
 - Extends to Situated feature as well
- New Open-Source Courteous Compiler
 - First open-source implementation of Courteous Compiler
 - Serves as reference implementation and algorithm for the Courteous Compiler technique
 - *Current Work (coding done, being tested):*
 - An incremental Courteous Compiler that reuses previous CC work when a (premise) Courteous LP is updated
 - Uses novel theory and algorithms

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

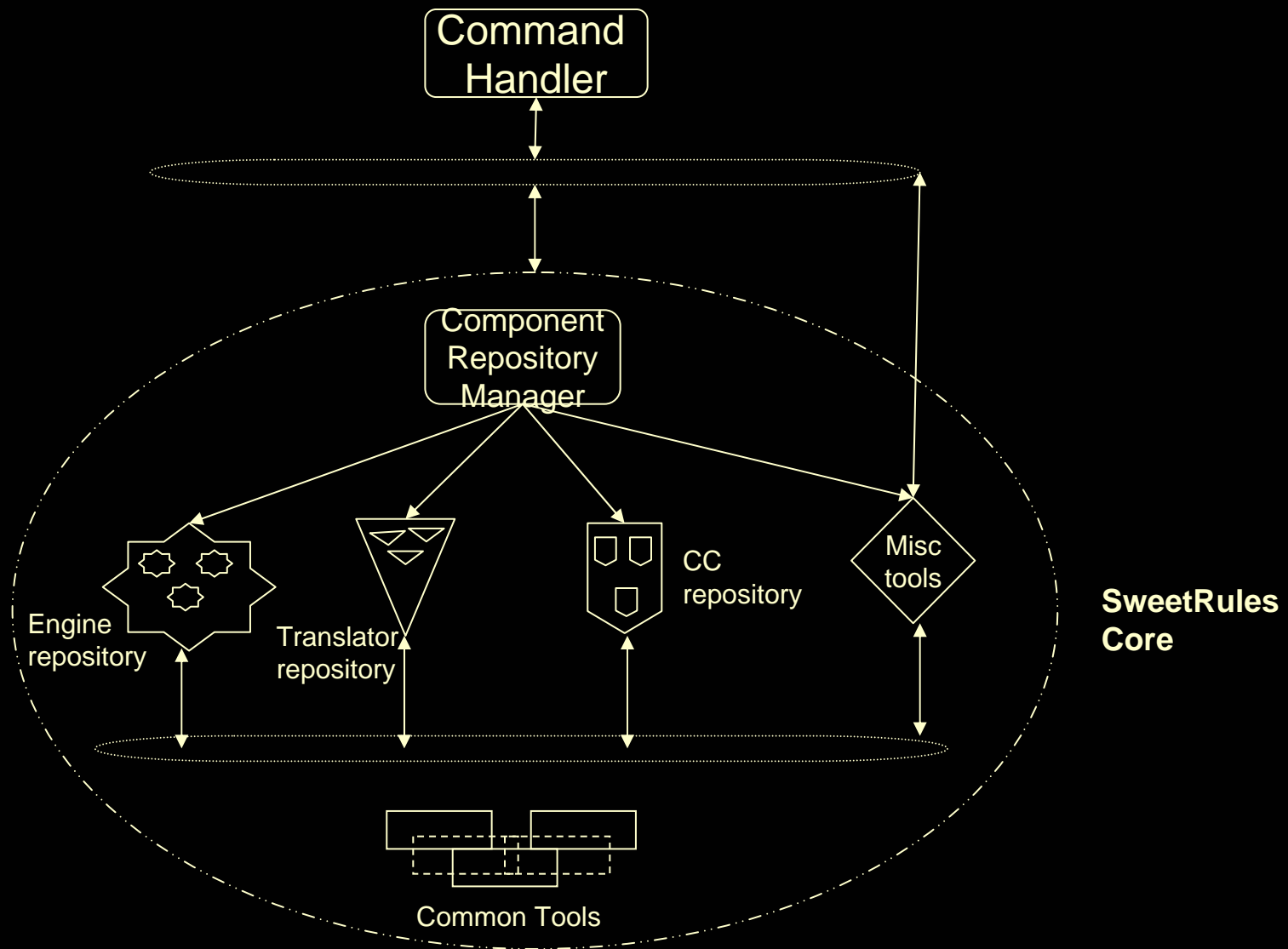
Novel Capability: Combine above with WSDL Web Services Support

- WSDL Web Services permitted as procedural attachments
 - Initially, only for effecting not yet sensing
 - Rules can directly trigger WSDL Web Services as actions
 - Experimental extension of RuleML syntax to specify a WSDL web service
 - Supports dynamic creation/invocation of the WSDL Web Service client (via DII)
 - In Action Launcher, thus available in SweetXSB, SweetJess, SweetCR

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 \otimes translator2
 - translator \otimes inferencing-engine
 - Search for tools

SweetRules architecture



Object Models for Rules/Ontologies

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

<u>API/Tool</u>	<u>Kind of Object Model</u>
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.

SweetRules platform – tool categories

- Translators
 - Simple
 - Composite
- Inference engines
 - Direct
 - Indirect
- Courteous compiler
 - Static
 - Incremental
- Misc. tools based on RuleML object model
 - Diff for Factsets
 - Predicate dependency graph builder
 - Predicate Stratifier
- Action Launcher
 - Web service support

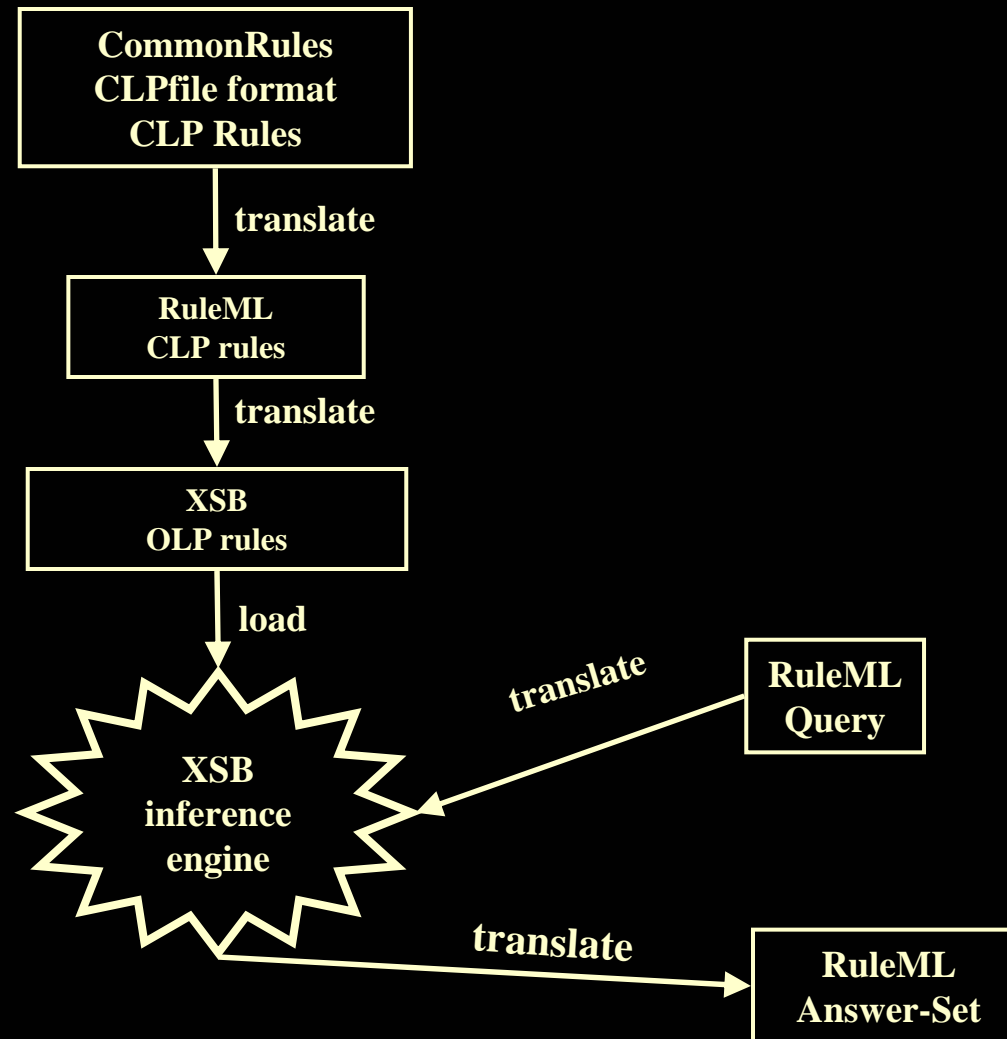
Measuring Power, Elegance and Reuse

- 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

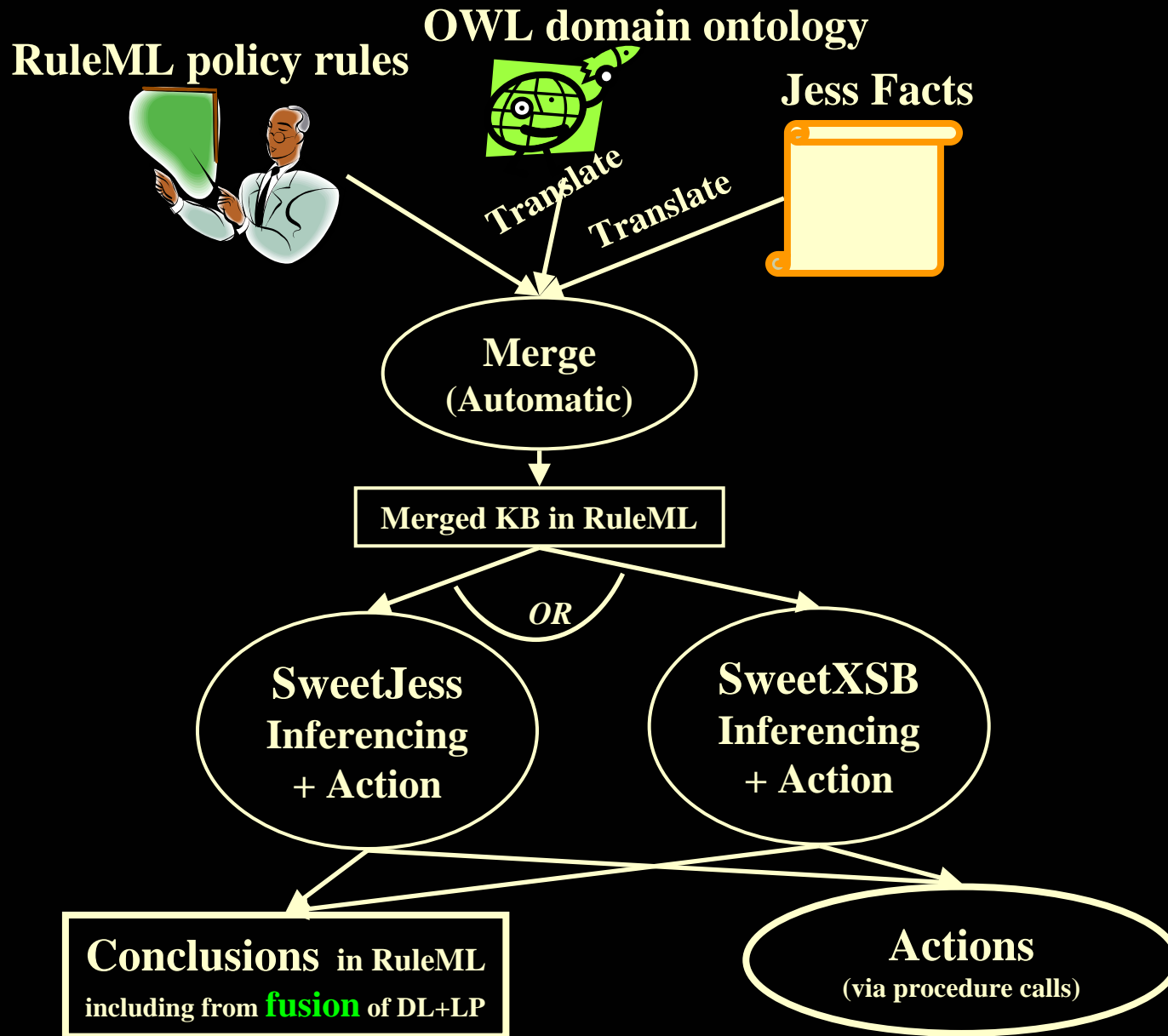
SweetRules V2 Demo Outline

- 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



OrderingLeadTime Example Demo Flow



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

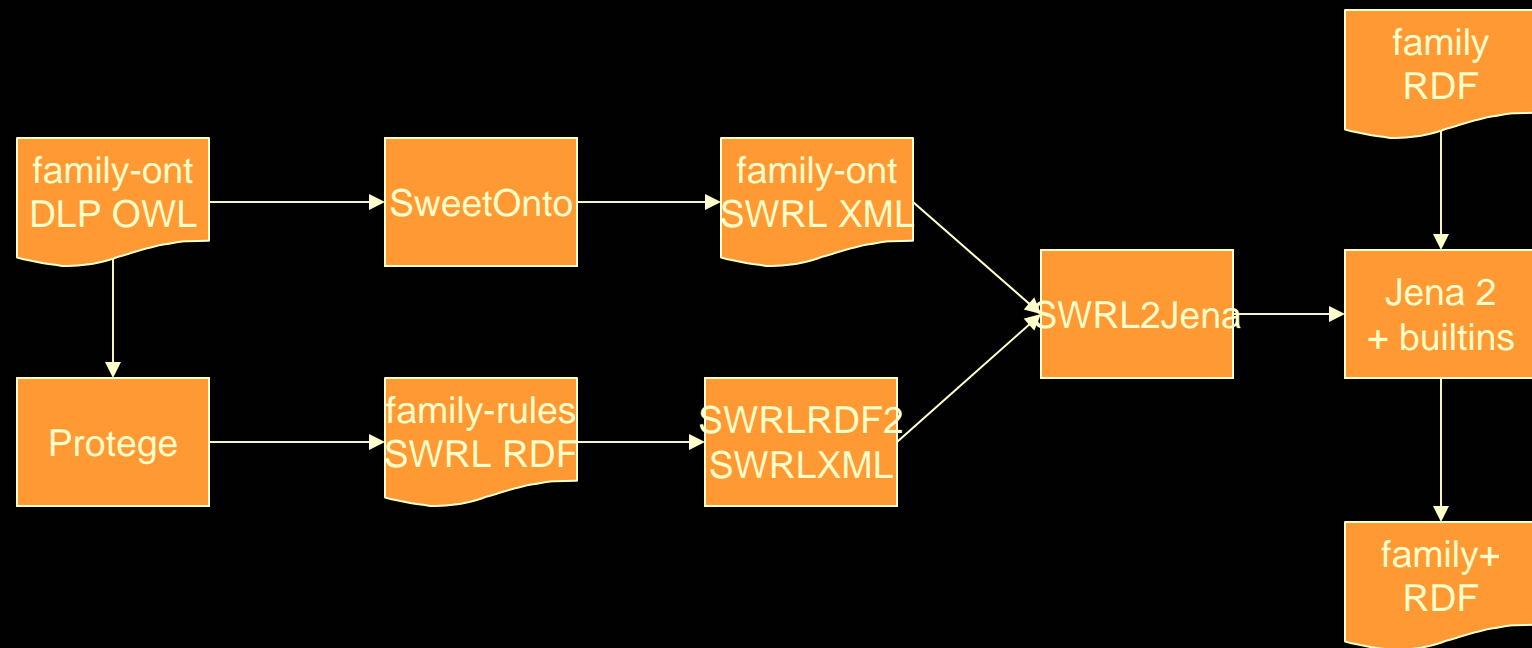
- See separate SweetRules V2 demo examples material.

SWRL-y SweetRules V2 Demo
by Mike Dean

SLIDES FOLLOW

- And also see separate SweetRules V2 demo examples material.

Protégé/SWRL/Jena Demo

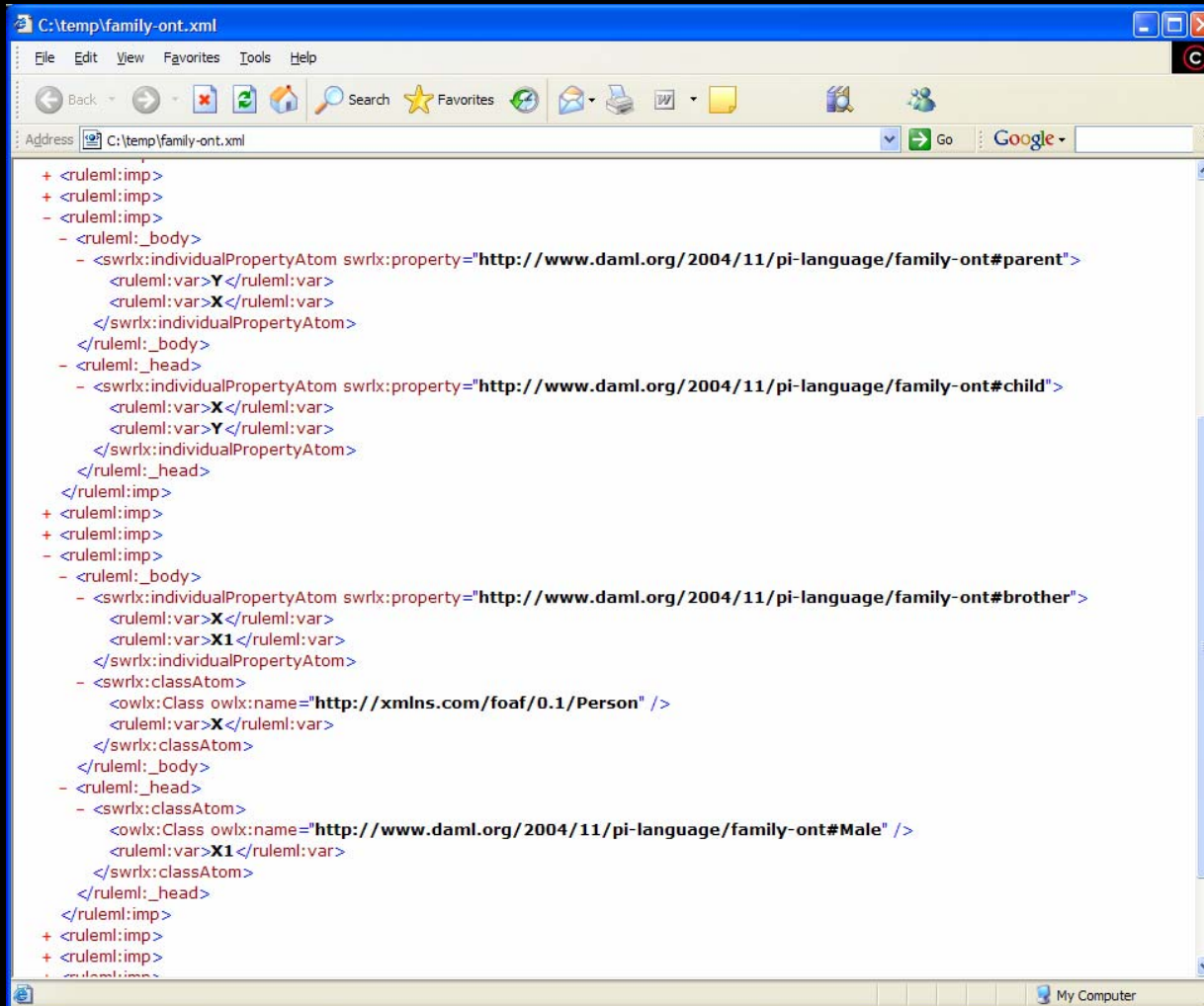


Protégé Ontology and Rules

The screenshot displays the Protégé 3.0 beta interface for editing an ontology. The main window is titled "SWRLDemoWithRules Protégé 3.0 beta (file:\C:\temp\SWRLDemoWithRules.pprj, OWL Files)". The interface is divided into several panes:

- Subclass Relationship:** Shows the asserted hierarchy for the project "SWRLDemoWithRules". The hierarchy includes: owl:Thing, family:Female, family:Male, foaf:Person (selected), rdf:List, swrt:Atom, swrt:Builtin, swrt:Imp, and swrt:Variable.
- Class Editor:** Shows the editor for the class "foaf:Person". It includes tabs for "Name", "SameAs", and "DifferentFrom". The "Name" tab is active, showing "foaf:Person" and "rdfs:comment". There is also an "Annotations" table with columns for Property, Value, and Lang.
- Asserted Conditions:** Shows a list of asserted conditions for the class "foaf:Person". The conditions are listed under "NECESSARY & SUFFICIENT" and "NECESSARY". The conditions include: 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, and family:sister family:Female.
- Properties:** Shows a list of properties: family:brother, family:child, family:daughter, family:father, family:mother, family:parent, family:sibling, and family:sister.
- SWRL:** Shows a list of SWRL rules. The rules are: family:birthDate(?individual, ?birth) ^ family:deathDate(?individual, ?death) ^ swrlb:subtractDates(?lifespan, ?death, ?birth) > family:lifespan(?individual, ?lifespan) and family:parent(?child, ?parent) ^ family:brother(?parent, ?uncle) > family:uncle(?child, ?uncle).

family-ont rules from SweetOnto



The screenshot shows a web browser window with the address bar set to `C:\temp\family-ont.xml`. The browser displays the XML content of the file, which consists of several rules. The rules are structured as follows:

```
+ <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>
+ <ruleml:imp>
```

family

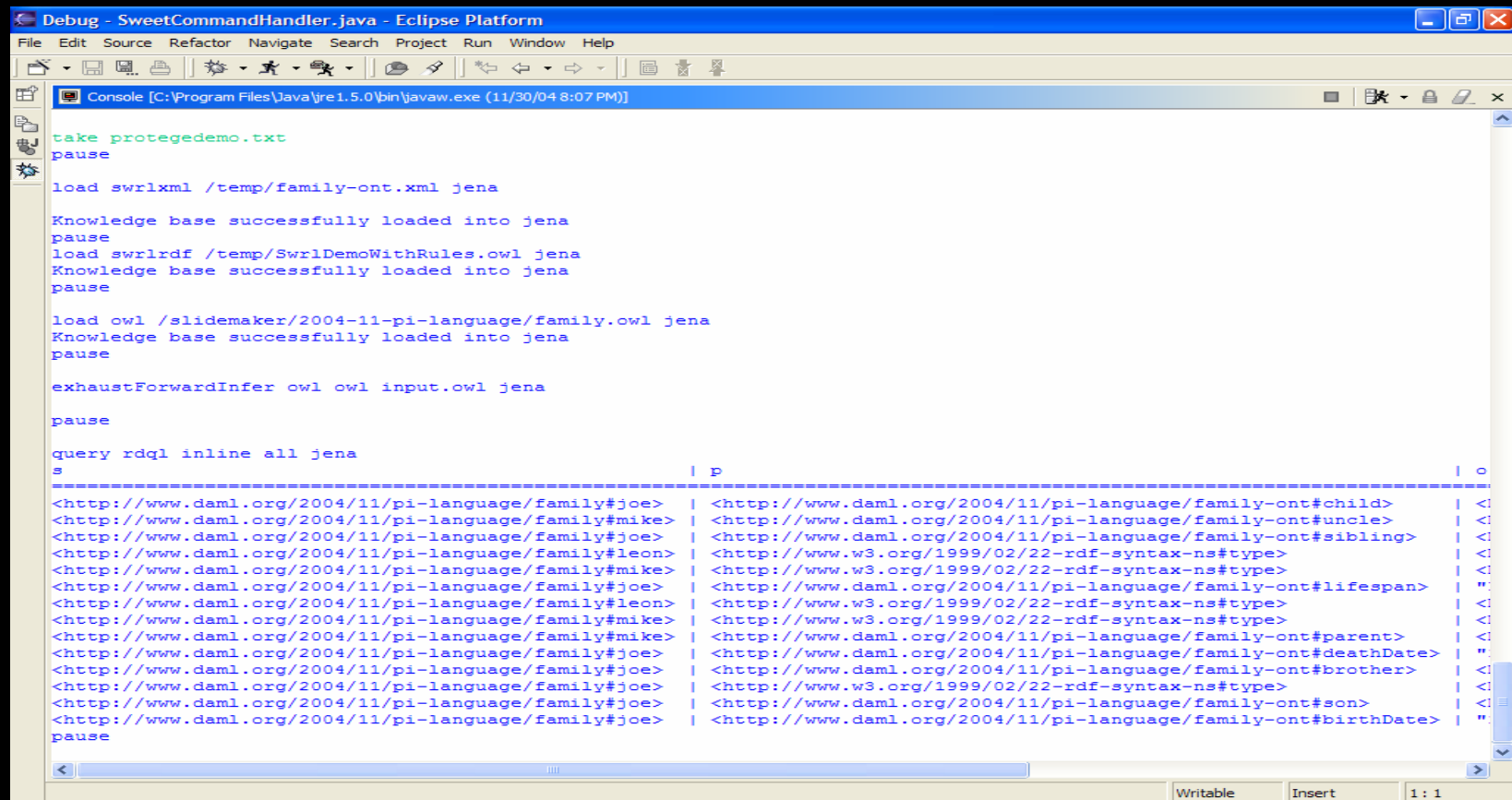
```
<?xml version='1.0' encoding='ISO-8859-1'?>
<!DOCTYPE rdf:RDF [
  <!ENTITY xsd 'http://www.w3.org/2001/XMLSchema#'>
]>

<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
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#joe> | <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
Writable Insert 1: 1
```


family+

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

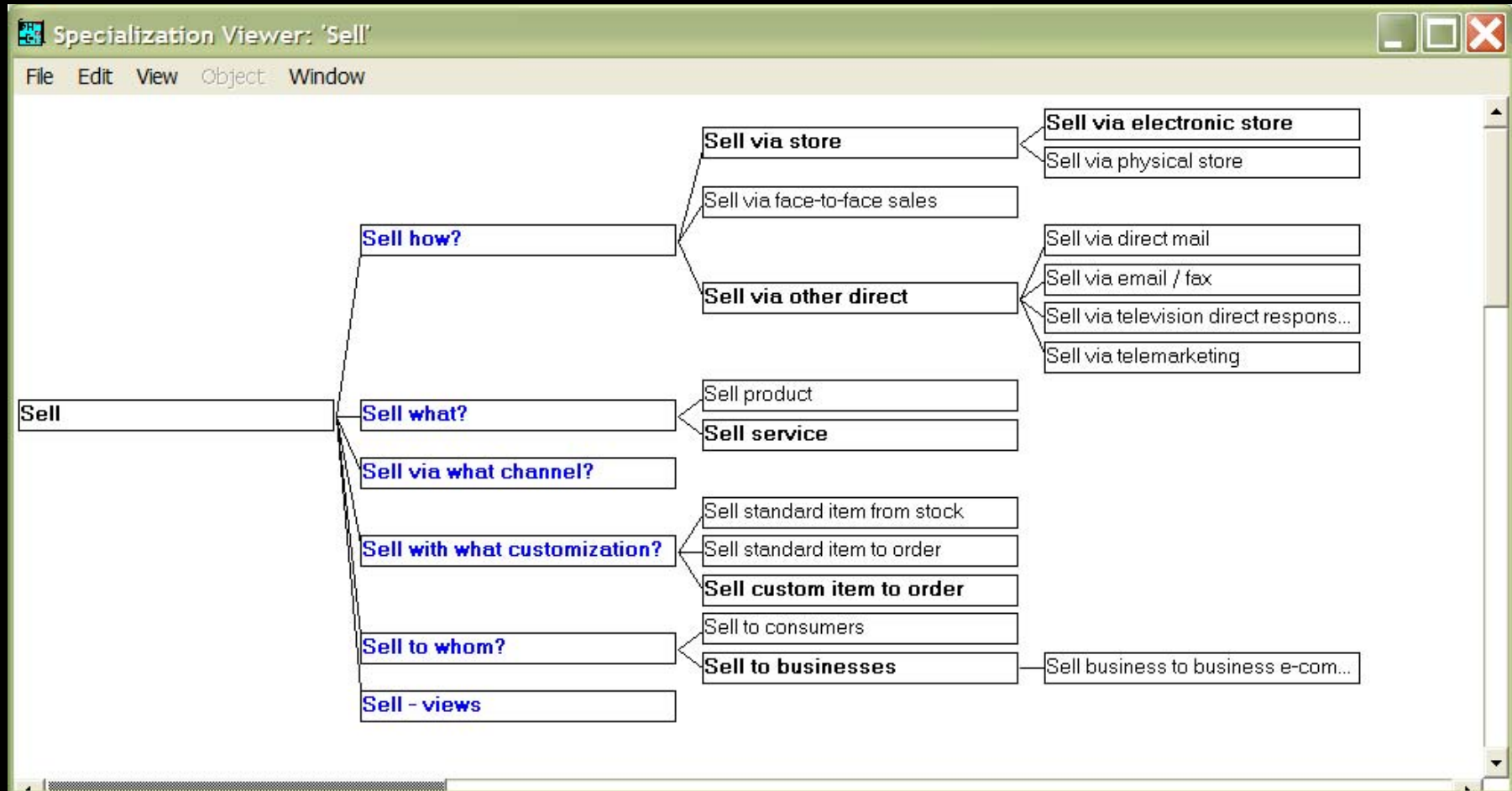
SweetRules: Use Cases Overview

- 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

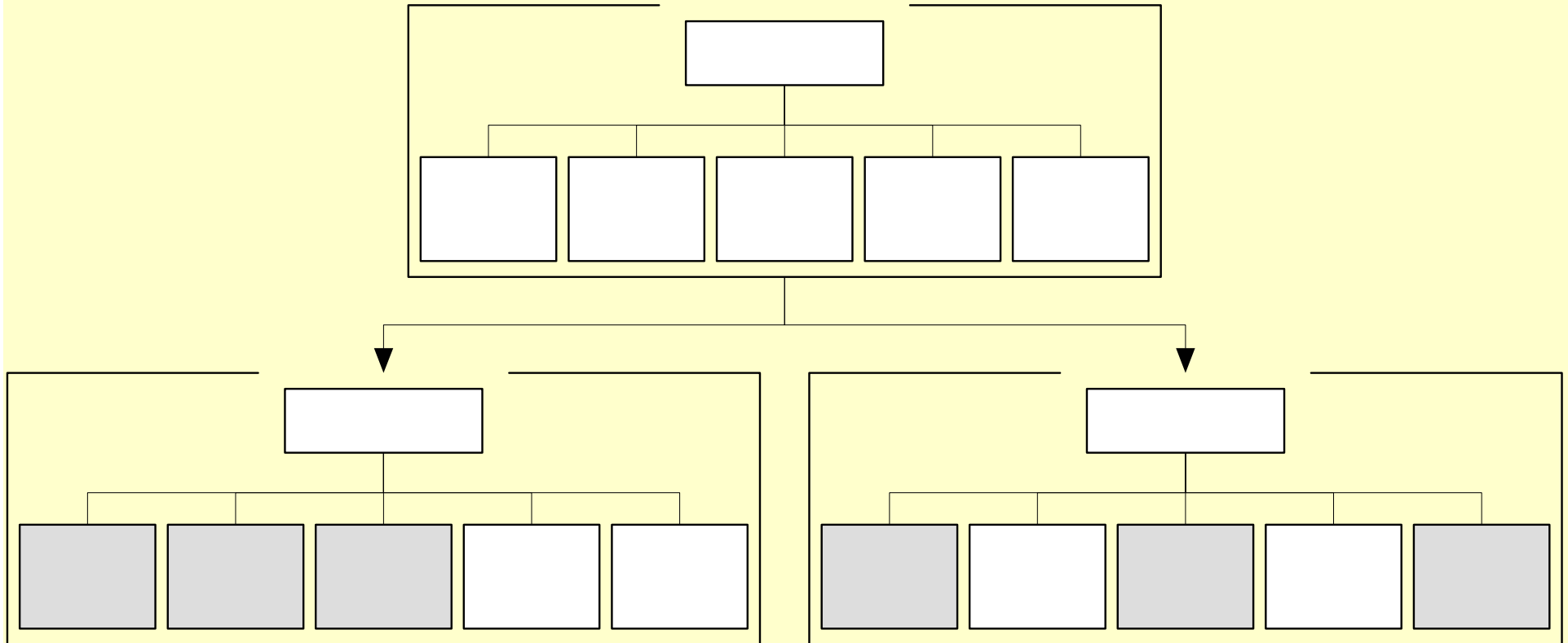
Opportunity for Process Handbook in SWS

- **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)



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).

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 \Rightarrow 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*

V2.1 Delta beyond V2.0: Major New Technical Features

- WSDL Web Services Support (initially effecting only)
 - Rules can directly trigger WSDL Web Services as actions
- Installation wizard, with selective configuration
- New open-source courteous compiler
- Full non-stratified NAF in SweetJess
- RuleML/SWSL presentation syntax support (initially generator only)

V2.1 Delta: Other Enhancements

- More application scenario examples.
 - E.g., multi-agent e-commerce security/trust authorization and policy management (merchant and bank credit card authorization)
- General polishing, including augmented documentation

V2.1 WSDL Web Services Support

- WSDL Web Services permitted as procedural attachments
 - Initially, only for effecting not yet sensing
 - Rules can directly trigger WSDL Web Services as actions
 - Experimental extension of RuleML syntax to specify a WSDL web service
 - Supports dynamic creation/invocation of the WSDL Web Service client (via DII)
 - In Action Launcher, thus available in SweetXSB, SweetJess, SweetCR

V2.1 Installation Wizard

- Installation wizard, with selective configuration
 - Guides user through process of installation, including of 3rd-party tools
 - Allows installation to omit selected components, including unneeded 3rd-party tools
 - Omit any subset of {SweetXSB, SweetJess, SweetOnto, SweetJena}
- Reduces installation effort by 5-10X

V2.1 New Courteous Compiler, in Open Source

- New Courteous Compiler, in open source
 - First-ever in open source
 - Provides reference implementation/algorithm for this technique
 - Removes dependency on IBM CommonRules' Courteous Compiler
 - *Background:* Courteous Compiler enables the Courteous prioritized conflict handling expressive feature to be added to any LP rule system that supports negation-as-failure
 - E.g. Production rules or Prolog system

V2.1 Full Non-Stratified NAF in SweetJess

- Full non-stratified Negation-As-Failure in SweetJess is supported in (translation and) inferencing via Jess
 - The general case of well-founded semantics for declarative LP
 - Implements new bottom-up algorithm [Grosf]
- First-of-a-kind capability for any inferencing engine based on a production rule system
 - Production rule systems do not support negation-as-failure with usual declarative LP semantics
 - But they can with a little help from their friends ☺
- *Background:*
 - V2.0 supported only stratified case of NAF

V2.1 RuleML Presentation Syntax Support

- RuleML / SWSL presentation syntax support
 - Initially, only a generator not yet a parser
 - Concise ASCII syntax
 - User-friendlier for editing
 - Experimental extension of RuleML presentation syntax for Situated feature of LP:
 - Effector statements, sensor statements
 - Attached procedures
- First-ever implementation

SweetRules: Future Directions I

- Polishing, generally, of doc and code
- Application Scenarios, esp. services – More
 - Policies, contracts, mediation, ...
- SweetPH release

- Tighter integration of SWRL with RuleML:
 - Specification (collaborating with RuleML Initiative)
 - Code

- Parser for RuleML presentation syntax
- Authoring, UI, Editors – More
 - Utilize RuleML presentation syntax

Future Directions II

- Hook up to Web Services more comprehensively
 - Sensing, querying of Web Services
 - Exposing functionality via/as Web Services
 - Importing knowledge bases / modules
 - Events
 - WS Protocols other than WSDL
- More Rule/Ontology Engines/Systems of Research kind:
 - Tasks: translation, inferencing
 - Flora-2 – esp. interesting for expressiveness incl. Hilog, Frame syntax
 - cwm, Rei, SWI Prolog, Triple, Hoolet, KAON, JTP, SNARK, DRS, ROWL, ...
 - Systems of new/various kinds: ECA, RDF-Query/XQuery, ...

Future Directions III

- Commercial Rule/Ontology Engines/Systems – More:
 - Production rules
 - Prolog: e.g., SWI Prolog
 - Relational databases
 - Event-Condition
- Production Rules – More Expressiveness
 - Use Production Logic Programs KR
 - Semantic formulation of production rules, equivalent to fragment of SCLP
 - Actions, tests directly in rules
 - Use latest theory [Grosf]

Future Directions IV

- Increase Expressiveness cf. RuleML and SWSL-Rules
 - Slotted syntax
 - Hilog (Use latest theory [Grosf & Kifer])
 - Frame syntax
 - Skolemization
 - XML-Schema Datatypes
 - Lists
 - User-defined Equality (Use latest theory)
 - Fuller Lloyd-Topor
 - Reification (Use latest theory [Kifer])
 - FOL
- FOL Support – More
 - FOL RuleML
 - Common Logic / KIF
 - perhaps SWRL FOL

Future Directions V

- Increase expressiveness of DLP (Description Logic Programs)
 - On translating Description Logic \Rightarrow LP
 - Use latest theory [Motik, Grosf, & Horrocks]
- Object Oriented Ontologies – More
 - With default inheritance, e.g., cf. SweetPH
 - Use latest theory [Grosf & Bernstein, Yang & Kifer]
- Hypermonotonic reasoning
 - Translation and merging between:
 - (Nonmon LP with Courteous/NAF) \leftrightarrow (FOL/DL)
 - Use latest theory [Grosf]

Future Directions VI

- Incremental reasoning, Events
 - Handle frequent updates
 - Incremental Courteous compilation and inferencing
 - Events cf. production rules, Event-Condition-Action rules
- Analyzers – More
 - Detect violations of expressive restrictions needed for translation or inferencing
 - Aid editing,
 - Conflict Analysis – More
 - find potential conflicts and suggest where prioritization needed
- Scaleability performance testing/benchmarking

Strategy on Future Directions

- *More Collaborators invited!*

SweetRules V2 Project Team

- **Core** Project Team members:
 - Benjamin Grosf (MIT Sloan), Project Lead, Lead Designer
 - Mike Dean (BBN Technologies), Project Co-Lead for V2.0
 - Shashidhara Ganjugunte (UMBC student), Lead Developer
 - Said Tabet (MIT Sloan)
 - Chitravanu Neogy (MIT Sloan)
- Other Project Team members:
 - Sumit Bhansali (MIT Sloan student)
 - Mark Musen (Stanford U.)
 - Martin O'Connor (Stanford U.)
 - Abraham Bernstein (U. Zurich)
 - Dave Kolas (BBN Technologies)
 - Timothy Finin (UMBC)
 - Anupam Joshi (UMBC)

SweetRules V2 Cooperating Efforts Team

- Cooperating Researchers Contributors:
 - Boris Motik (as U. Karlsruhe student)
 - Rudi Studer (U. Karlsruhe)
 - Raphael Volz (as U. Karlsruhe student)
 - June von Bonin (as U. Zurich student)
 - Terrence Poon (as MIT student)
 - Hoi Chan (IBM)
 - Harold Boley (NRC/UNB)
 - Dave Reynolds (HP)
 - Ernest Friedman-Hill (Sandia National Labs)
 - Michael Kifer (SUNY Stonybrook)
 - * (We may have forgotten to include someone that we should have; if so, apologies!)
- Cooperating Efforts:
 - RuleML Initiative (close collaboration)
 - SWSL: Semantic Web Services Initiative's Language Committee (SWSL)
 - WSML: Web Services Mediation Ontology's Language effort
 - W3C, Oasis, OMG standards bodies – via RuleML, SWSI, and WSMO

SweetRules V2 Team Roles

- Project Lead: B. Grosf.
 - Project Co-Lead for V2.0: M. Dean.
- Lead designer of core including SCLP RuleML and DLP OWL aspects: B. Grosf
- Lead developer 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. Grosf, A. Bernstein (U. Zurich)
- Lead implementer of SweetPH: A. Bernstein
- Lead designer of SweetDeal application scenario prototype: B. Grosf
- Lead developer of SweetDeal: S. Bhansali (MIT Sloan student)
- Leads on coordinating documentation & support: S. Ganjugunte (UMBC), C. Neogy (MIT Sloan)

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

Supporting Query Sessions in SweetXSB

- XSB is a backward reasoner meant for query-answering
- Indirect inferencing through RuleML query
- SweetRules has a notion of session, to facilitate loading KB and querying XSB
- RuleML schema had to be extended to include query and answerSet elements

```
<!ELEMENT query ( (_rlab,_body) | (_body,_rlab?) )>
```

```
<!ELEMENT answerSet (answer*)>
```

```
<!ELEMENT answer (binding*)>
```

```
<!ELEMENT binding ((BVar, BSubstitution)|(BSubstitution, BVar))>
```

```
<!ELEMENT BVar var>
```

```
<!ELEMENT BSubstitution cterm>
```

More Details about SweetXSB (cont...)

- Courteous features supported via invocation of courteous compiler
- Simulate forward reasoning by querying every predicate
 - XSB's tabling capability makes this tolerably efficient
- Handled XSB awkwardness problem with the undefined (u) truth value
 - Even though XSB is based on WFS, it returns 't' for an undefined answer

SweetJess [Grosf, 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. Grosf 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/bgrosf>

Problem with negation in Jess

- Consider the following Rulebase:
 - (assert (f c))
(defrule rk1 (and (f ?x) (not (h ?x))) => (assert (k ?x))) (defrule rh2 (and (f ?x) (not (g ?x))) => (assert (h ?x)))
 - Expected conclusions (including premises):
 - (f c)
 - (h c)
 - Jess incorrectly also give (k c)
- If one re-order the rules in the rulebase one gets the proper conclusions (cf. LP semantics of NAF)

Stratified case – easy fix

- Stratified means the rulebase does not have cycles which involve NAF
- Stratification is grouping predicates into strata such that
 - If p depends on naf q, then $\text{strata}(p) > \text{strata}(q)$
 - If p depends on q, then $\text{strata}(p) \geq \text{strata}(q)$
 - So, evaluate the rule base for low strata and proceed to higher strata to get conclusions (**low** → **high**)
- Jess has a notion of rule salience which gives a sequencing “priority” to rules, i.e. **higher salience** rule is executed first (**high** → **low**)

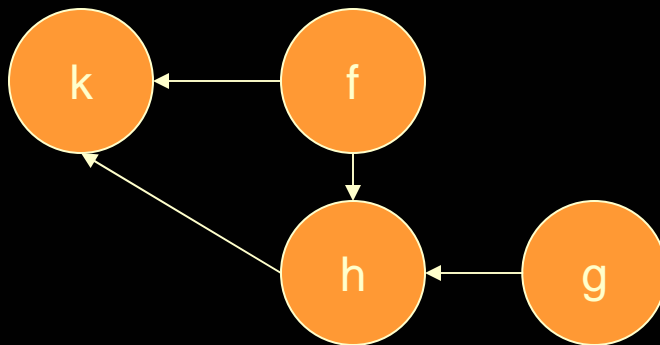
Stratified case – easy fix (cont...)

- Salience for a rule with head predicate whose strata is s , is calculated as
 - $\text{Salience} = (T + 1 - s) * w$
 - T is total number of strata and W is the scaling factor
 - $(1, w$ are not absolutely essential, but they are present for robustness)
- For a rule with a conjunctive head,
 - $\text{Salience} = \text{maximum salience value from among all head conjuncts}$

Example

- `(assert (f c))`
`(defrule rk1 (and (f ?x) (not (h ?x))) => (assert (k ?x)))`
`(defrule rh2 (and (f ?x) (not (g ?x))) => (assert (h ?x)))`
- PDG will look like:

Stratification



Saliency for rk1 is $(4 - 3) * 1000 = 1000$ and for rh2 is $(4 - 2) * 1000 = 2000$

g	h	f	k
1	2	1	3

Non-stratified case

- The fix is non-trivial as the rulebase does not have an order in which it can be evaluated (so as to compute the proper conclusions)
- We give a new algorithm for computing the Well-Founded Model, that exploits the capabilities available in production rule systems (specimen Jess)
 - Horn LP forward inferencing, with Rete network
 - Simple negation (lookup in working fact set)
 - “Modules” (manage inferencing in multiple rulebases that can share facts across multiple modules’ working fact sets)
- We extend this new approach to handle sensing/tests and effecting/actions, cf. production rules and Situated feature of LP/RuleML.

This exploits the capabilities available in production rule systems (specimen Jess) for:

 - Sensing/tests
 - Effecting/actions

Challenges in Adapting WFS to Production Rules

- The previous proof theory by Przymusinski [1994] covering exhaustive forward inferencing is given in terms of instantiated LP
- Instantiation is expensive
- Instantiating might not result in good use of the Rete network
- So, we need a way to compute for a non-ground LP and iterate until quiescence.
- Also, we need to incorporate sensing and effecting.

Incremental forward inferencing for Courteous LP

- Goal: Support incremental inferencing for Courteous LP.
 - Backward inferencing: Fairly simple: just run incremental CC, and query the resulting OLP (using also inverse ECN)
 - Forward inferencing: More complicated. want to store and reuse conclusions, in addition to running the incremental CC.
- Approach: Maintain a Predicate Dependency Graph (PDG), and its Strong Connected Components (SCC's) for the post-CC OLP
 1. Recompute first the conclusions for the predicate locales whose premises were modified in the post-CC OLP (“directly updated predicates”)
 2. Use the PDG/SCC's to determine which other predicate locales are “indirectly affected” since they are dependent on predicates that are directly updated. Recompute the conclusions for those too.
 - a. All predicates in the same SCC as a directly updated predicate.
 - b. All predicates in SCC's that are dependent upon a directly updated SCC.

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

Address “the 5 D’s” of real-world reasoning \Rightarrow *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 scalability is vital to achieve the promise of knowledge integration. *Achieve Polynomial-time (\sim databases).*

Objectives for Integrating Distributed SW Rules and Ontologies,

Motivating SweetRules II

BEFORE

Contradictory conflict is globally contagious, invalidates all results.



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

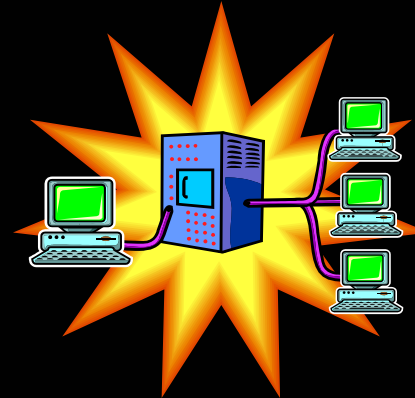


AFTER

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



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