

# Representing Ontologies and Integrating them with Rules, for Semantic Policies and Services on the Web

Benjamin Grosf

MIT Sloan School of Management

Information Technologies group

<http://ebusiness.mit.edu/bgrosf>

*Slides presented at SRI International, Artificial Intelligence Center,*

*Dec. 6, 2005, Menlo Park, CA, USA*

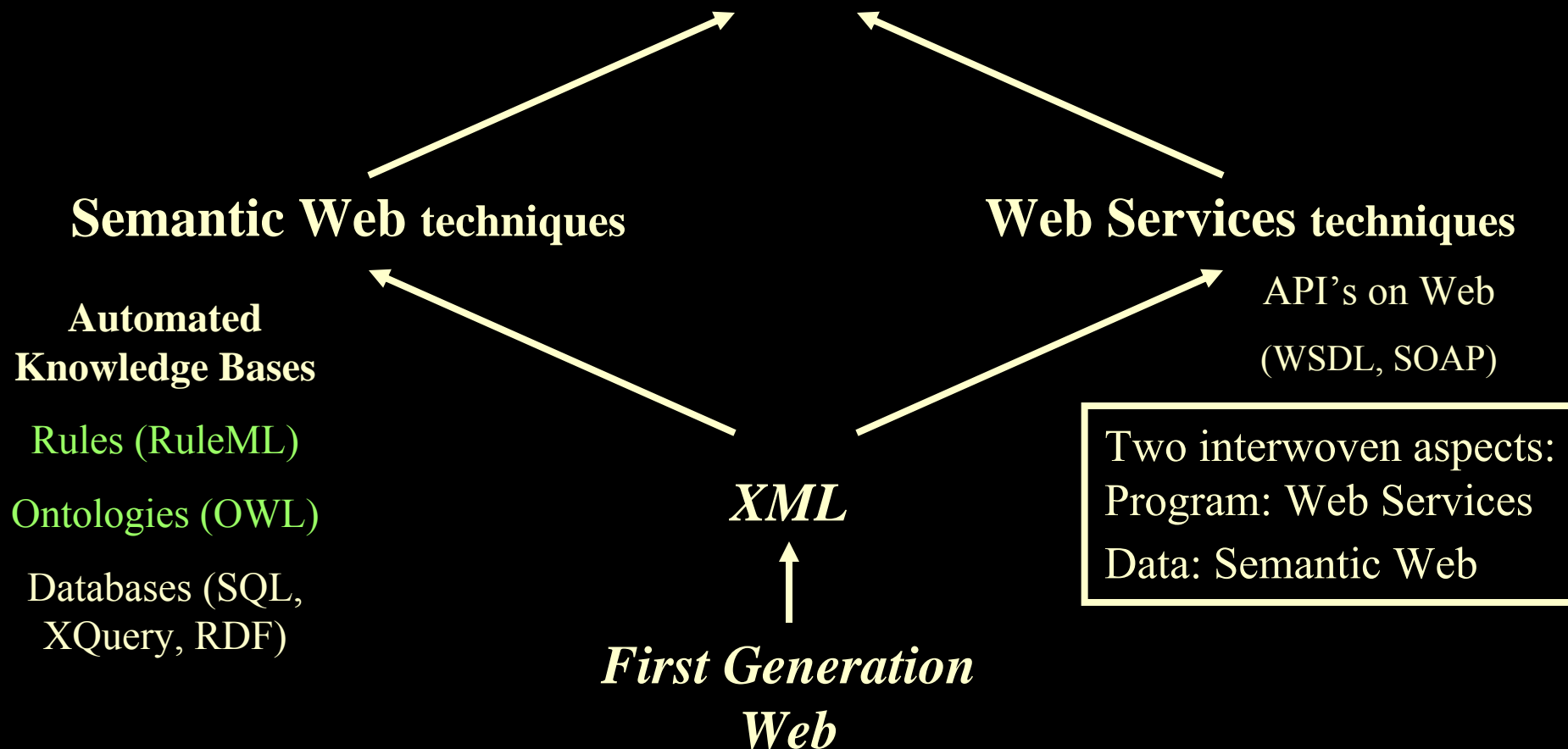
<http://www.ai.sri.com>

# Outline of Talk

- Intro: Research on Semantic Web Services (SWS), its Business Uses
  - Rules, contracting, trust, policies
  - Integration, knowledge representation, standards
- Problem: Reusable Ontological Knowledge to Describe Services
  - Technique: knowledge representation to standardize on
  - Content investment: how to leverage legacy business process K
- New Technical Approach to represent OO Frameworks using SW
  - Courteous Inheritance: default rules increases reuse in ontologies
- New Strategy: go where the knowledge already is, then work outwards
  - Begin with MIT Process Handbook – open-source version in development
    - Example: process knowledge about selling
  - Future: Transformational wrappers around various legacy OO frameworks
- Business Value Analysis and Market/Applications Roadmapping
- *If time: more details on integrating FOL / OWL ontologies into Logic Programs*

# Next Generation Web

## Semantic Web **Services**



# *Big Questions*

## *about the New Generation Web*

- What are the critical features/aspects of the new technology?
- What business problems does it help solve?
- What are the likely innovation evolution paths, and associated entrepreneurial opportunities?

# *Our Research Aspects/Questions about the Semantic Web*

- **Core technologies:** Requirements, concepts, theory, algorithms, standards?
  - Rules in combination with ontologies; probabilistic, decision-/game-theoretic
- **Business applications and implications:** concepts, requirements analysis, techniques, scenarios, prototypes; strategies, business models, market-level evolution?
  - End-to-end e-contracting, finance, trust; ...

*Some Answers to:*  
*“Why does SW Matter to Business?”*

- 1. “Death. Taxes. Integration.” - They’re always with us.
- 2. “Business processes require communication between organizations / applications.” - Data and programs cross org./app. boundaries, both intra- and inter- enterprise.
- 3. “It’s the *automated knowledge* economy, stupid!”
  - The world is moving towards a knowledge economy. And it’s moving towards deeper and broader automation of business processes. The first step is automating the use of structured knowledge.
  - Theme: *reuse* of knowledge across multiple tasks/app’s/org’s

# *Strategic Business Foci in our SW Research*

- Knowledge-based Services Engineering: intra- and inter- enterprise
- Target “killer app” known for 30 years: do better job of EDI
- Challenges:
  - Ease of development, deployment ↑
  - Reuse of knowledge ↑
  - ⇒ life cycle costs ↓ , agility ↑
- Starting with: Policies
  - Using recent theory breakthroughs in semantic rules
  - E.g., for end-to-end contracting and authorization (incl. security)
- Starting with: EAI as well as B2B

# *Vision: Uses of Rules in E-Business*

- Rules as an important aspect of coming world of Internet e-business: rule-based business policies & business processes, for B2B & B2C.
  - represent seller's offerings of products & services, capabilities, bids; map offerings from multiple suppliers to common catalog.
  - represent buyer's requests, interests, bids; → matchmaking.
  - represent sales help, customer help, procurement, authorization/trust, brokering, workflow.
- Known advantages of rules vs. general code
  - separable business logic, more reusable across app.'s, life cycle
  - good for loose coupling cf. workflow
  - good for representing contingent behavior of services/processes.
  - high level of conceptual abstraction; easier for non-programmers to understand, specify, dynamically modify & merge.
  - executable but can treat as data, separate from code
    - potentially ubiquitous; already wide: e.g., SQL views, queries.
- Rules in communicating applications, e.g., embedded intelligent agents.



# *Semantic Rules: Differences from Rules in the 1980's / Expert Systems Era*

- Get the KR right
  - More mature research understanding
  - Semantics independent of algorithm/implementation
  - Cleaner; avoid general programming/scripting language capabilities
  - Highly scaleable; high performance; better algorithms
  - Highly modular wrt updating; use prioritization
- Leverage Web, esp. XML
  - Interoperable syntax
  - Merge knowledge bases
- Embeddable
  - Into mainstream software development environments (Java, C++, C#); not its own programming language/system (cf. Prolog)
- Knowledge Sharing: intra- or inter- enterprise
- Broader set of Applications

# *New Fundamental Rule KR Theory that enables Key Technical Requirements for SWS*

## *In 1985-94:*

- Prolog interoperable with relational DB; LP extends core-SQL [many]
- Richer logical connectives, quantifiers [Lloyd & Topor]
- “Well Founded” Semantics for Negation-As-Failure [Van Gelder et al; Przymusiński]
- Hilog quasi-higher order expressiveness, meta-syntax flexibility [Kifer et al.]
- Frame syntax cf. F-Logic [Kifer et al.]

## *In 1995-2004:*

- **Courteous LP: prioritized conflict handling** [Groszof]
  - Robust, tractable, modular merging & updating
- **Situated LP: hook rules up to services** [Groszof]
- **Description LP: combine Description Logic ontologies** [Groszof et al.]
- **Courteous Inheritance: combine OO default ontologies** [Groszof et al.]
- **Production Rules as LP: interoperate** [Groszof et al.]
  - Declarative LP as interoperable core between commercial families [Groszof et al.]
- **Hypermonotonic Reasoning: combine with FOL** [Groszof (in-progress)]

# *Production Logic Programs:* *A New Fundamental Rule KR Approach*

*In 2005:*

- Production extension of LP:
  - actions and tests appear directly within rules (procedural attachments)
  - Generalizes Situated LP a bit, and reformulates it more familiarly
- Theory & algorithms achieving semantic interoperability of {core Production Rules}  $\leftrightarrow$  declarative LP
  - Handles negation correctly, by stratifying PR agenda control strategy
  - 1<sup>st</sup> declarative semantics for Production Rules
- Combines with all the other features Courteous, ...
- $\rightarrow$  “**Production LP**” as umbrella LP KR approach

## *SW Rules: Use Cases from our research*

- Contracts/negotiation, advertising/discovery
  - E-procurement, E-selling
  - Pricing, terms & conditions, supplier qualification, ...
- Monitoring:
  - Exception handling, e.g., of contract violations
    - Late delivery, refunds, cancellation, notifications
  - Notifications, personal messaging, and other workflow
- Trust Policies: authorization, confidentiality & 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

# *SWS and Rules Summary*

*\*\* SWS Tasks Form 2 Distinct Clusters,  
each with associated Central Kind of Service-description  
Knowledge and Main KR*

## 1. Security/Trust, Monitoring, Contracts, Advertising/Discovery, Ontology-mapping Mediation

- Central Kind of Knowledge: Policies
- Main KR: Nonmon LP (rules + ontologies)

## 2. Composition, Verification, Enactment

- Central Kind of Knowledge: Process Models
- Main KR: FOL (axioms + ontologies)
  - + Nonmon LP for ramifications (e.g., cf. Golog)
- Thus RuleML & SWSF specify both Rules, FOL
  - Fundamental KR Challenge: “Bridging” Nonmon LP with FOL
    - SWSF experimental approach based on hypermon. [Grosf & Martin]

# *Advantages of Standardized SW Rules for Policies, e.g., Authorization/Security*

- Easier Integration: with rest of business policies and applications, business partners, mergers & acquisitions
  - Enterprise integration, B2B
- Familiarity, training
- Easier to understand and modify by humansChange management
- Quality and Transparency of implementation in enforcement
  - Provable guarantees of behavior of implementation
- Reduced Vendor Lock-in
- Expressive power
  - Principled handling of conflict, negation, priorities
- ⇒ **Agility, change management** ↑

## *Advantages of SW Rules, cont'd:*

### *Loci of Business Value in Policy Management*

- Reduced system dev./maint./training costs
- Better/faster/cheaper policy admin.
- Interoperability, flexibility and re-use benefits
- Greater visibility into enterprise policy implementation ⇒ better compliance
- Centralized ownership and improved governance by Senior Management
- Rich, expressive policy management language allows better conflict handling in policy-driven decisions
- Strategic agility, incl. wrt business model

# *Future Work Directions*

- More scenarios, esp. in SWS policy/SCAMP task cluster
- Integration of more expressive ontologies from OWL, FOL (beyond DLP)
  - Extend DLP in various ways:
    - Use: skolems, integrity constraints, equality, sensing
  - Use hypermonotonic reasoning approach (new KR theory) [SWSF 2005]
    - Map FOL  $\leftrightarrow$  Courteous LP
    - View nonmon LP as weakened FOL: sound, incomplete
    - E.g., policy rules + background FOL/DL ontologies
- Integration of OO ontologies with default inheritance
- More integration into e-business communication and Web Services, following our SWS vision



# Outline of Talk

- Intro: Research on Semantic Web Services (SWS), its Business Uses
  - Rules, contracting, trust, policies
  - Integration, knowledge representation, standards
- Problem: Reusable Ontological Knowledge to Describe Services
  - Technique: knowledge representation to standardize on
  - Content investment: how to leverage legacy business process K
- New Technical Approach to represent OO Frameworks using SW
  - Courteous Inheritance: default rules increases reuse in ontologies
- New Strategy: go where the knowledge already is, then work outwards
  - Begin with MIT Process Handbook – open-source version in development
    - Example: process knowledge about selling
  - Future: Transformational wrappers around various legacy OO frameworks
- Business Value Analysis and Market/Applications Roadmapping
- *If time: more details on integrating FOL / OWL ontologies into Logic Programs*

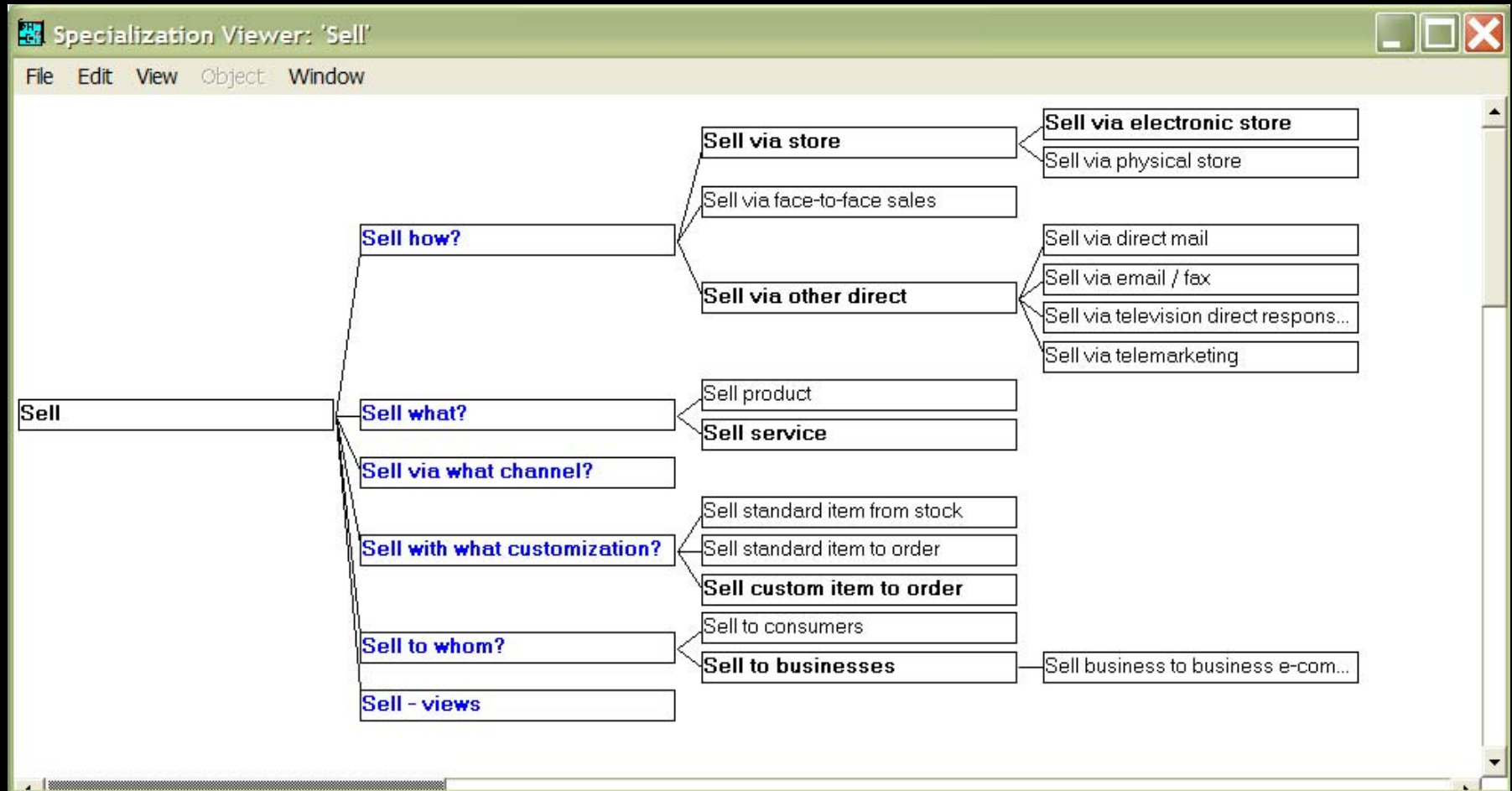
# *Problem: Reusable Knowledge to Describe Services*

- Has two aspects:
  1. **Technical/technique problem:** what form of knowledge? I.e., what knowledge representation to standardize on?
  2. **Content investment problem:** how to leverage to accomplish the reuse of legacy business process knowledge?

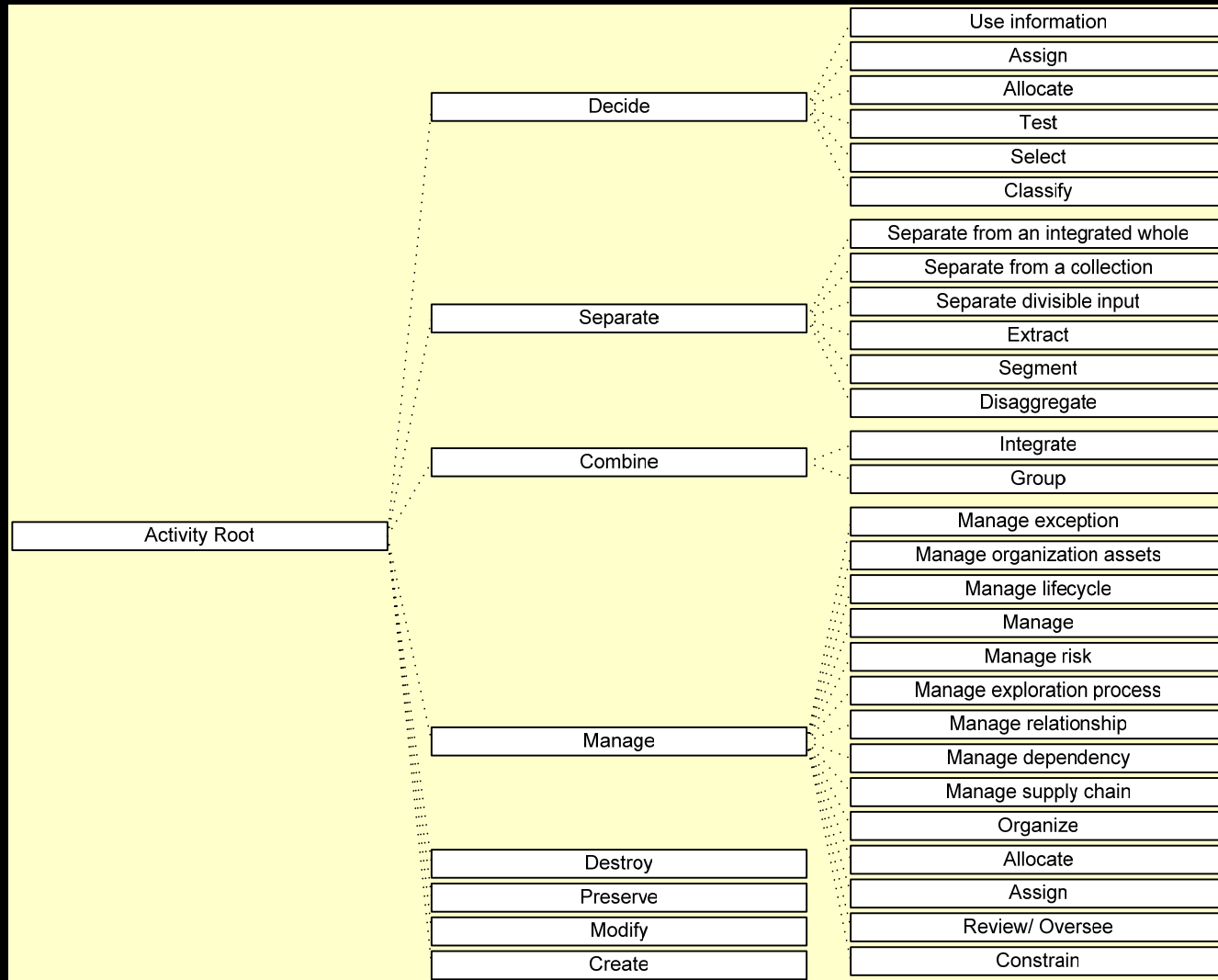
# *Opportunity for Process Handbook in SWS*

- **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)



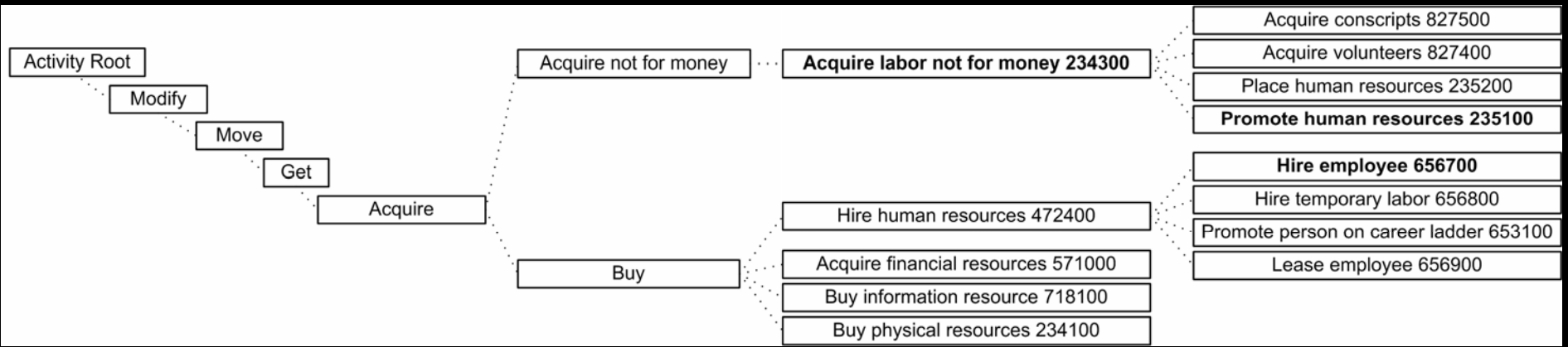
# Some Process Handbook Ontology



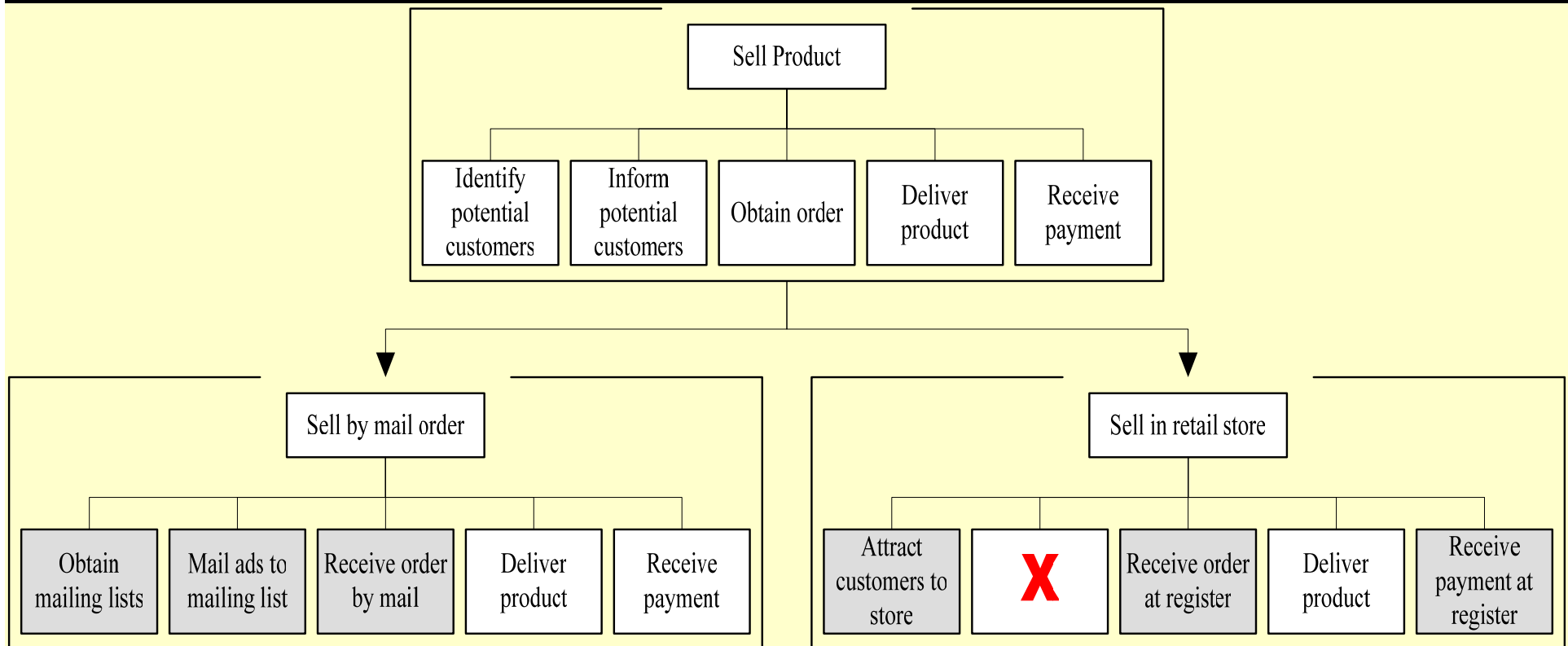
12/6/2005

Copyright 2002-2005 by Benjamin Grosf. All Rights Reserved.

# *Some Process Handbook Ontology*



# 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).

# *Represent Default-Inheritance Object-Oriented Ontologies Via Courteous LP*

- Default-inheritance object-oriented ontologies are ubiquitous in business process realm:
  - Java, C++ frameworks
  - Frame-based systems
- Override or cancel inheritance at subclass.
- OWL, Description Logic, FOL cannot represent default behavior: monotonic only.
- Nonmonotonic/default character increases reuse as compared to monotonic-only.
- Courteous LP can represent them nicely.
  - E.g., SweetPH represents Process Handbook OO business process ontology (5000 processes, 38000 axioms) [Grosf & Bernstein 2003]



*Example of Default-Inheritance OO  
Ontologies in Courteous LP:  
Via Direct Specification in CLP*

```
{buyRegular} paymentMode(?quoteID,invoice) :- Buy(?quoteID).
```

```
/* BuyWithCredit is a subclass of Buy */
```

```
Buy(?quoteID) :- BuyWithCredit(?quoteID).
```

```
{buyCredit} paymentMode(?quoteID,credit)  
                :- BuyWithCredit(?quoteID).
```

```
overrides(buyCredit, buyRegular).
```

# *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*

# *New Technical Approach: Courteous Inheritance in the Process Handbook*

- Use SW KR and standards to represent Object-Oriented framework knowledge: class hierarchy, types, generalization-specialization, domain & range, properties/methods' association with classes
- Surprise: use SW *rule* language not the main SW *ontology* language! I.e., use RuleML not OWL.
- Exploit RuleML's nonmonotonic ability to represent prioritized default reasoning as kind of knowledge representation (KR)

## *New Technical Approach, continued*

- Courteous Inheritance KR is built simply on top of the (Situating) Courteous Logic Programs KR of RuleML
  - A few dozen background axioms. Linear-size reformulation. Inferencing is tractable computationally.
- Particularly: represent PH's structured part
  - a scheme specific to PH's flavor of OO
- PH becomes a SWS process ontology repository
  - to be combined, fed, used with/by other SWS
- Kill two birds with one stone:
  - form of K that facilitates leveraging of legacy process K content including PH, OO

## *New Technical Approach, continued more*

- Example(s): selling, PO, price, shipping, delivery, payment, lateness.
- For details, see submitted paper “Beyond Monotonic Inheritance: Towards Semantic Web Process Ontologies” on webpage.
  - Example: selling process

# *Larger Approach: Transformation Wrappers for OO Frameworks*

- New Strategy: go where the knowledge already is, then work outwards
- Future: Transformational wrappers around various legacy OO frameworks
  - C++
  - Java, C#
  - UML
- Can use XSLT, SW tools, and/or XQuery engines to implement the transformations, guided by SWS ontology standardization practices

# *Market Evolution: Discussion Questions*

- Existing and prospective early adopters
- Importance of open source content: seems to be an assumption/axiom for many people
- Prospective sources of open source content

# Outline

- Introduction and Context: Semantic Web Services for E-Business; Policies
- Overview: SweetDeal Approach, New Extensions
- More Details: SweetDeal, SCLP, KB merging, SweetRules
- Procurement Scenario
- Fact-queries, as part of communicated KB's
- OO default inheritance ontologies, as Courteous LP
- Relationship to E-Business Messaging Standards / Platforms
- Business Value Analysis
- Conclusions & Future Work



# *Some Technical Directions for Research*

- Incremental Reasoning: Events, Updates
- LP KR other extensions:
  - Existentials via skolemization
  - Combine Hilog higher-order features reducible to first-order; OWL-Full, RDF-Full
  - Equality: user-defined, nonmonotonic
  - Reification
- Hypermonotonicity: analysis of LP, merging; new KR's incl. disjunctive
- Probabilistic, decision-theoretic, game-theoretic; Inductive, learning, data mining
- Constraints: satisfaction, optimization
  
- Trust policies for firewalls, confidentiality, security, privacy, access control
- E-Contracting end-to-end reuse, power: incl. business process monitoring
- Policy Ontology, Services Ontologies, Relationship to C++/Java/C# Inheritance
- Web Services “Policy Management”, “Contracts”
- Add semantics to existing standards: XBRL, XACML, ebXML, RosettaNet, EDI
- Biomedical: patient records privacy and workflow, drug discovery, treatment safety tracking
- Marketing, intelligence, supply chain, financial reporting, travel
- Business Value Analysis, Strategy, Roadmapping

# Outline of Talk

- Intro: Research on Semantic Web Services (SWS), its Business Uses
  - Rules, contracting, trust, policies
  - Integration, knowledge representation, standards
- Problem: Reusable Ontological Knowledge to Describe Services
  - Technique: knowledge representation to standardize on
  - Content investment: how to leverage legacy business process K
- New Technical Approach to represent OO Frameworks using SW
  - Courteous Inheritance: default rules increases reuse in ontologies
- New Strategy: go where the knowledge already is, then work outwards
  - Begin with MIT Process Handbook – open-source version in development
    - Example: process knowledge about selling
  - Future: Transformational wrappers around various legacy OO frameworks
- Business Value Analysis and Market/Applications Roadmapping
- *If time: more details on integrating FOL / OWL ontologies into Logic Programs*

# OPTIONAL SLIDES FOLLOW

- About techniques for integrating OWL and FOL ontologies, including Description Logic Programs

# *URI Ontological Reference Approach*

- A RuleML predicate (or individual / logical function) is specified as a URI, that refers to a predicate (or individual / logical function, respectively) specified in another KB, e.g., in OWL.
- Application pilot and first use case: in SweetDeal e-contracting system (design 2001, prototype early 2002).
- Approach was then soon incorporated into RuleML and adopted in SWRL design (which is based mainly on RuleML), and used heavily there.
- Issue: want to scope precisely which premises in an overall ontological KB are being referenced.
  - Approach in our current work: define a KB (e.g., a subset/module) and reference that KB.

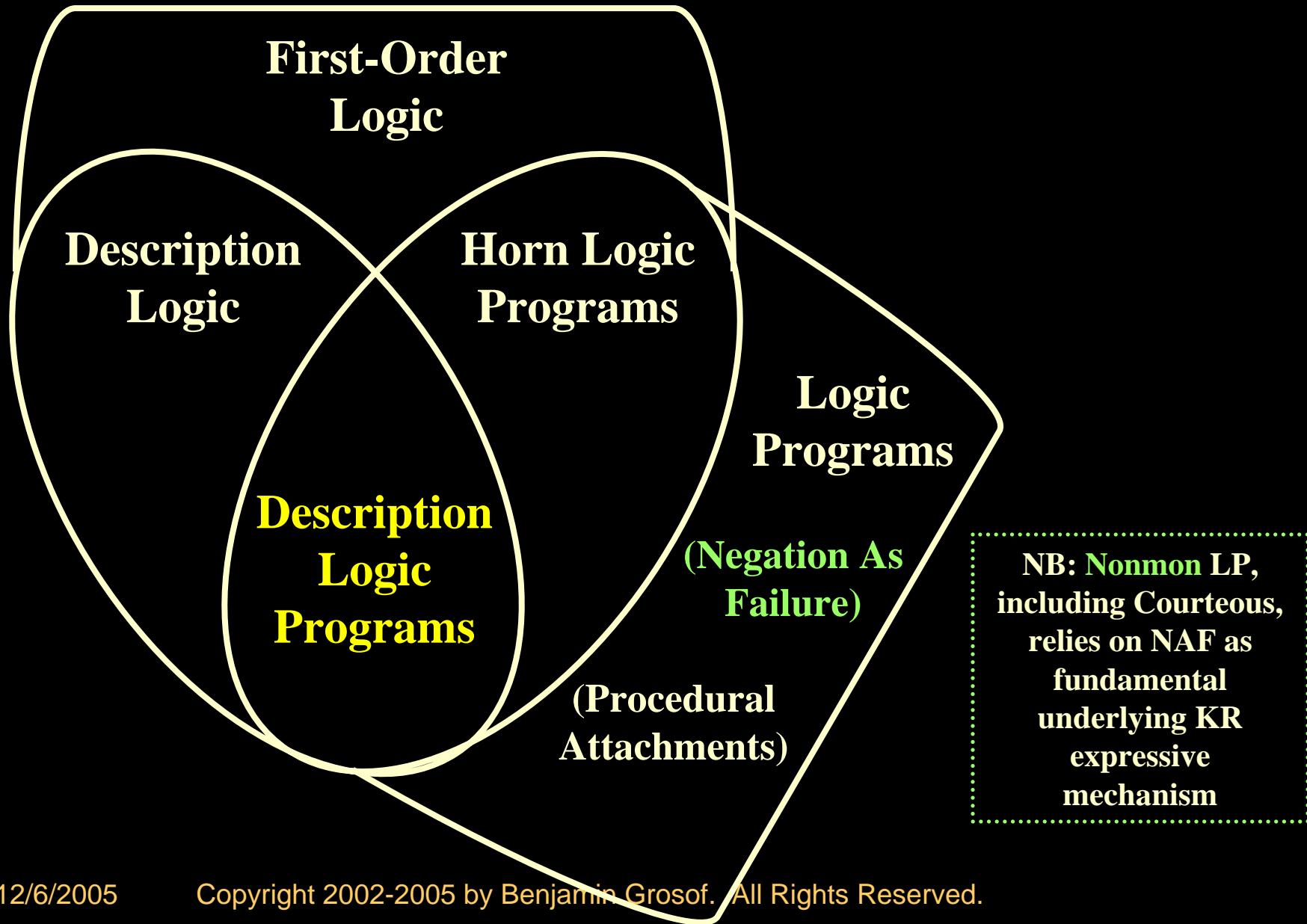
## URI Ontological Reference Approach Example, in RuleML

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

### SCLP TextFile Format for RuleML

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

# Venn Diagram: Expressive Overlaps among KR's



# Overview of DLP KR Features

- DLP captures **completely** a subset of DL, comprising RDFS & more
- **RDFS subset** of DL permits the following statements:
  - Subclass, Domain, Range, Subproperty (also SameClass, SameProperty)
  - instance of class, instance of property
- DLP also completely captures **more** DL statements beyond RDFS:
  - Using Intersection connective (conjunction) in class descriptions
  - Stating that a property (or inverse) is Transitive or Symmetric
  - Using Disjunction or Existential in a *subclass* expression
  - Using Universal in a *superclass* expression
  - ∴ “**OWL Feather**” – subset of OWL Lite
    - Update summer 2004: New Related Effort is “OWL Lite Minus” by WSMO
- DLP++: enhanced translation into LP can express even more of DL:
  - Using explicit equality, skolemization, integrity constraints
  - Using NAF, for T-box reasoning
  - (*Part still in progress.*)

## *DLP-Fusion:*

### *Technical Capabilities Enabled by DLP*

- LP rules "on top of" DL ontologies.
  - E.g., LP imports DLP ontologies, with completeness & consistency
  - Consistency via completeness. (Also, Courteous LP is always consistent.)
- Translation of LP rules to/from DL ontologies.
  - E.g., develop ontologies in LP (or rules in DL)
- Use of efficient LP rule/DBMS engines for DL fragment.
  - E.g., run larger-scale ontologies
  - $\Rightarrow$  Exploit: Scalability of LP/DB engines  $\gg$  DL engines , as  $|\text{instances}| \uparrow$  .
- Translation of LP conclusions to DL.
- Translation of DL conclusions to LP.
- Facilitate rule-based mapping between ontologies / “contexts”



# *Design Perspective*

Alternative points in design space:

1. partial LP + full DL = SWRL V0.6

*versus*

2. full LP + partial DL = SCLP RuleML V0.8+  
(with DLP OWL2RuleML)

(SCLP = Situated Courteous Logic Programs KR)

# *Need for Other Kinds of Ontologies besides OWL*

- Kinds of ontologies practically/commercially important in the world today\*:
  - SQL DB schemas, E-R, UML, OO inheritance hierarchies, LP/FOL predicate/function signatures; equations and conversion-mapping functions; XML-Schema
- OWL is still emerging.
- Overall relationship of OWL to the others is as yet largely unclear
  - There are efforts on some aspects, incl. UML
- OWL cannot represent the nonmon aspects of OO inheritance
- OWL does not yet represent, except quite awkwardly:
  - n-ary signatures
  - ordering aspects of XML-Schema
- (\*NB: Omitted here are statistically flavored ontologies that result from inductive learning and/or natural language analysis.)

## *Need for Other Kinds of Ontologies besides OWL, cont.'d*

- Particularly interesting:
  - OO-ish nonmon taxonomic/frames
  - Equations and context mappings cf. ECOIN – can be represented in FOL or often in LP
  - OWL DL beyond DLP
- Builtins (sensed) are a relatively simple kind of shared ontology
  - SWRL V0.6 and forthcoming RuleML V0.9

# Default Inheritance cf. OO

- Ubiquitous in object-oriented programming languages & applications
- Default nature increases reuse, modularity
- **OWL/DL fundamentally incapable of representing, since monotonic**
- Requirements of semantic web service process ontologies:
  - Need to jibe with mainstream web service development methodologies, based on Java/C#/C++
- Approach: Represent OO default-inheritance ontologies using nonmon LP rules
  1. [Grosf & Bernstein] Courteous Inheritance approach
    - Transforms inheritance into Courteous LP in RuleML
    - Represents MIT Process Handbook (ancestor of PSL)
      - 5,000 business process activities; 38,000 properties/values
      - Linear-size transform ( $n + \text{constant}$ ).
    - SweetPH prototype: extends SweetRules
  2. [Yang & Kifer] approach
    - Transform inheritance into essentially Ordinary LP
    - Extends Flora-2

*Fundamental KR Challenge in  
Combining Rules with Ontologies:  
Unify FOL/DL More Deeply with Nonmon LP*

- Motivations: Better support KB merging, SWSL, unify SW overall, more of DL/FOL in LP, handle conflicts between DL/FOL KB's, ...
- Approach: “Hypermonotonic” reasoning [Grosf]
  - Courteous LP mapped  $\iff$  clausal FOL
    - Courteous LP always sound wrt FOL
    - ... & incomplete wrt FOL
      - Enables: always consistent, robust in merging
    - Mapping is linear-size and local

# OPTIONAL SLIDES FOLLOW

- About Situated and Courteous extensions of LP

## *Review: Situated and Courteous extensions of LP*

- 1. **Situated Logic Programs:**
  - KR to hook rules (with ontologies) up to (web) services
  - Rules use services, e.g., to query, message, act with side-effects
  - Rules constitute services executably, e.g., workflow-y business processes
- 2. **Courteous Logic Programs:**
  - KR to combine rules from many sources, with:
    - Prioritized conflict handling to enable consistency, modularity; scaleably
    - Interoperable syntax and semantics
- These extensions combine essentially orthogonally.
  - Sensors may be the subject of prioritized conflict handling, so it is useful to give them labels.

# *EECOMS Example of Conflicting Rules: Ordering Lead Time*

- Vendor's rules that prescribe how buyer must place or modify an order:
  - A) 14 days ahead if the buyer is a qualified customer.
  - B) 30 days ahead if the ordered item is a minor part.
  - C) 2 days ahead if the ordered item's item-type is backlogged at the vendor, the order is a modification to reduce the quantity of the item, and the buyer is a qualified customer.
- Suppose more than one of the above applies to the current order? **Conflict!**
- Helpful Approach: **precedence** between the rules. Often only *partial* order of precedence is justified. E.g.,  $C > A$ .



# *Courteous LP's:*

## *Ordering Lead Time Example*

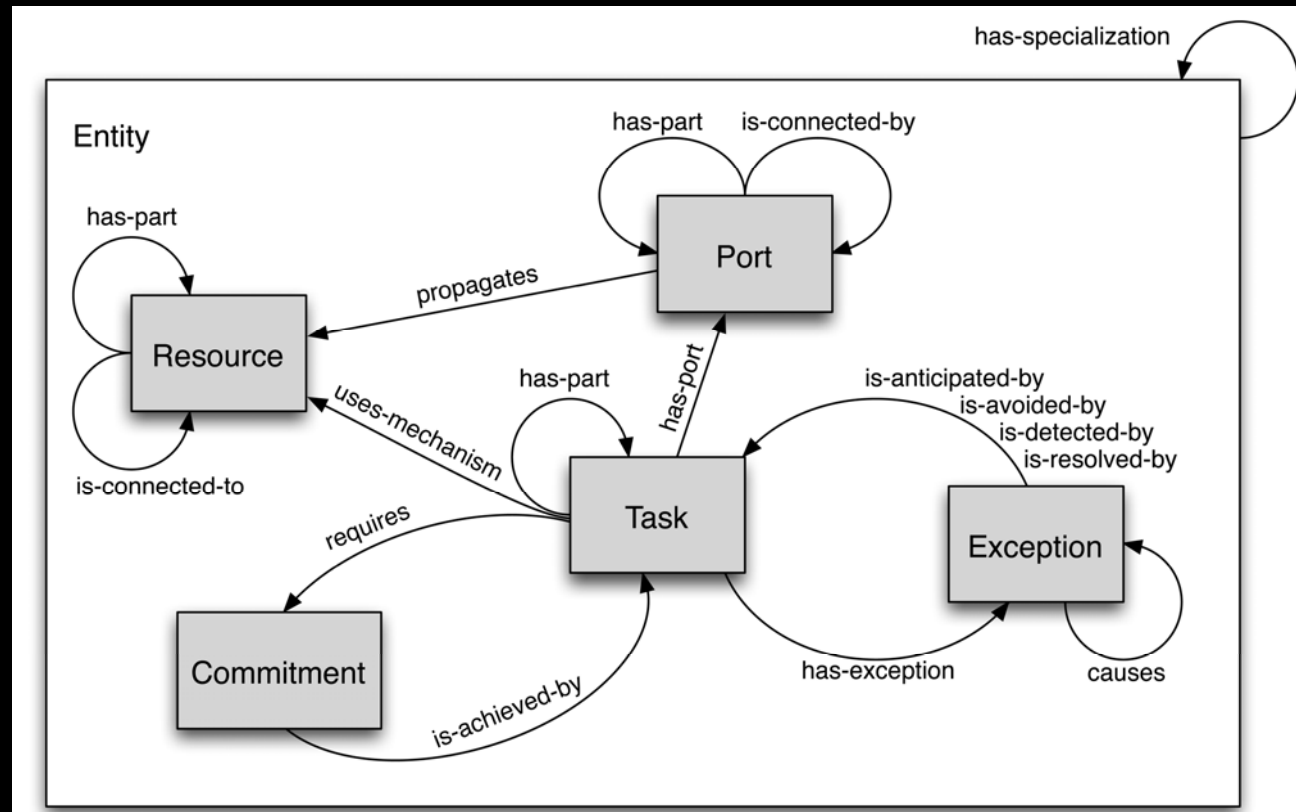
- `<leadTimeRule1> orderModificationNotice(?Order,14days)`
- `← preferredCustomerOf(?Buyer,?Seller) ∧`
- `purchaseOrder(?Order,?Buyer,?Seller) .`
- `<leadTimeRule2> orderModificationNotice(?Order,30days)`
- `← minorPart(?Buyer,?Seller,?Order) ∧`
- `purchaseOrder(?Order,?Buyer,?Seller) .`
- `<leadTimeRule3> orderModificationNotice(?Order,2days)`
- `← preferredCustomerOf(?Buyer,?Seller) ∧`
- `orderModificationType(?Order,reduce) ∧`
- `orderItemIsInBacklog(?Order) ∧`
- `purchaseOrder(?Order,?Buyer,?Seller) .`
- `overrides(leadTimeRule3 , leadTimeRule1) .`
- `(⊥ ← orderModificationNotice(?Order,?X) ∧`
- `orderModificationNotice(?Order,?Y)) ← (?X ≠?Y) .`

# OPTIONAL SLIDES FOLLOW

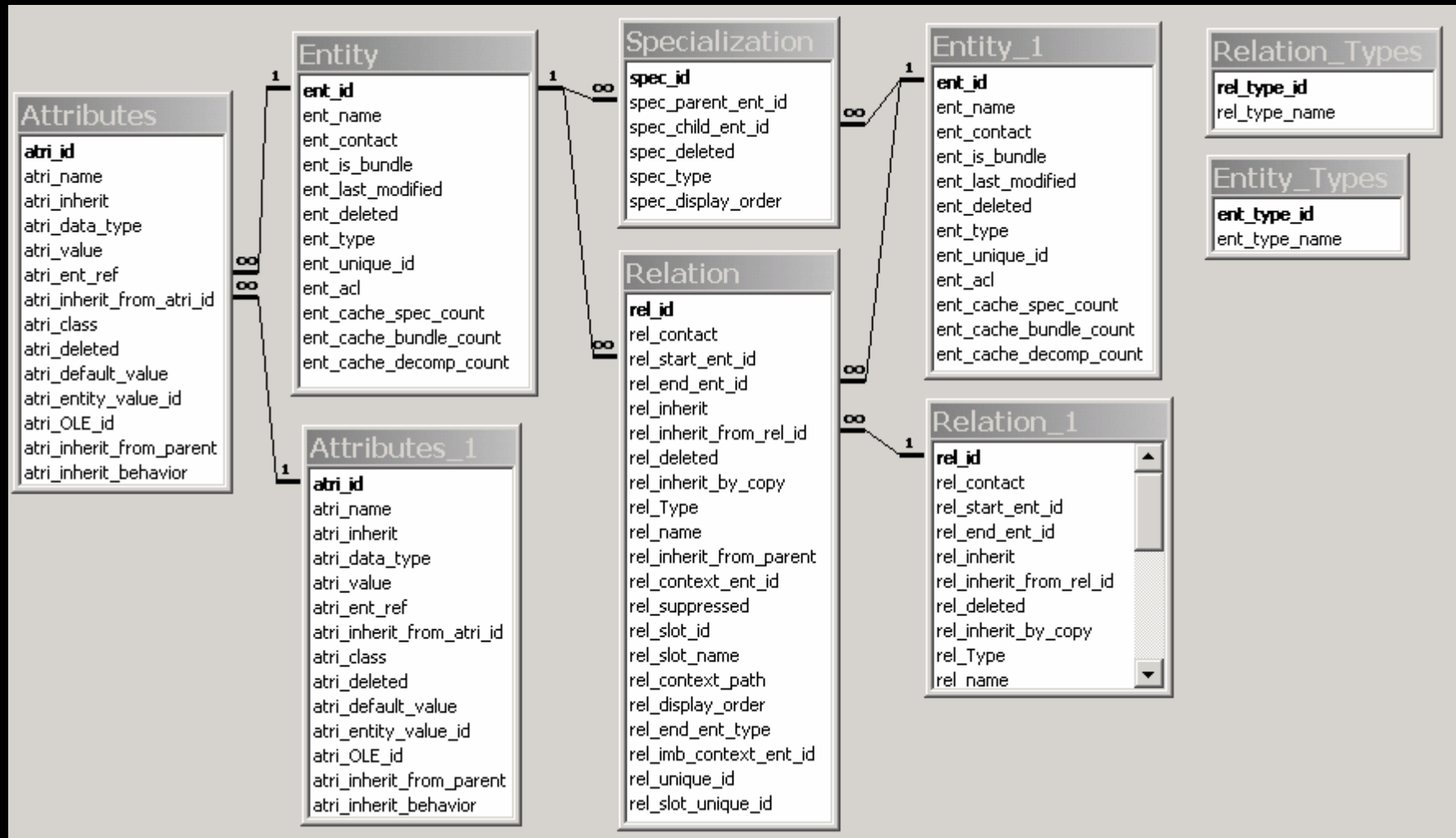
- About Process Handbook and SweetPH

# The MIT Process Handbook

- Process repository (built for human consumption)
- Over 5000 processes, ~ 50000 assertions
  - Taxonomy of generic activity types
  - Case examples, on-line discussion forums



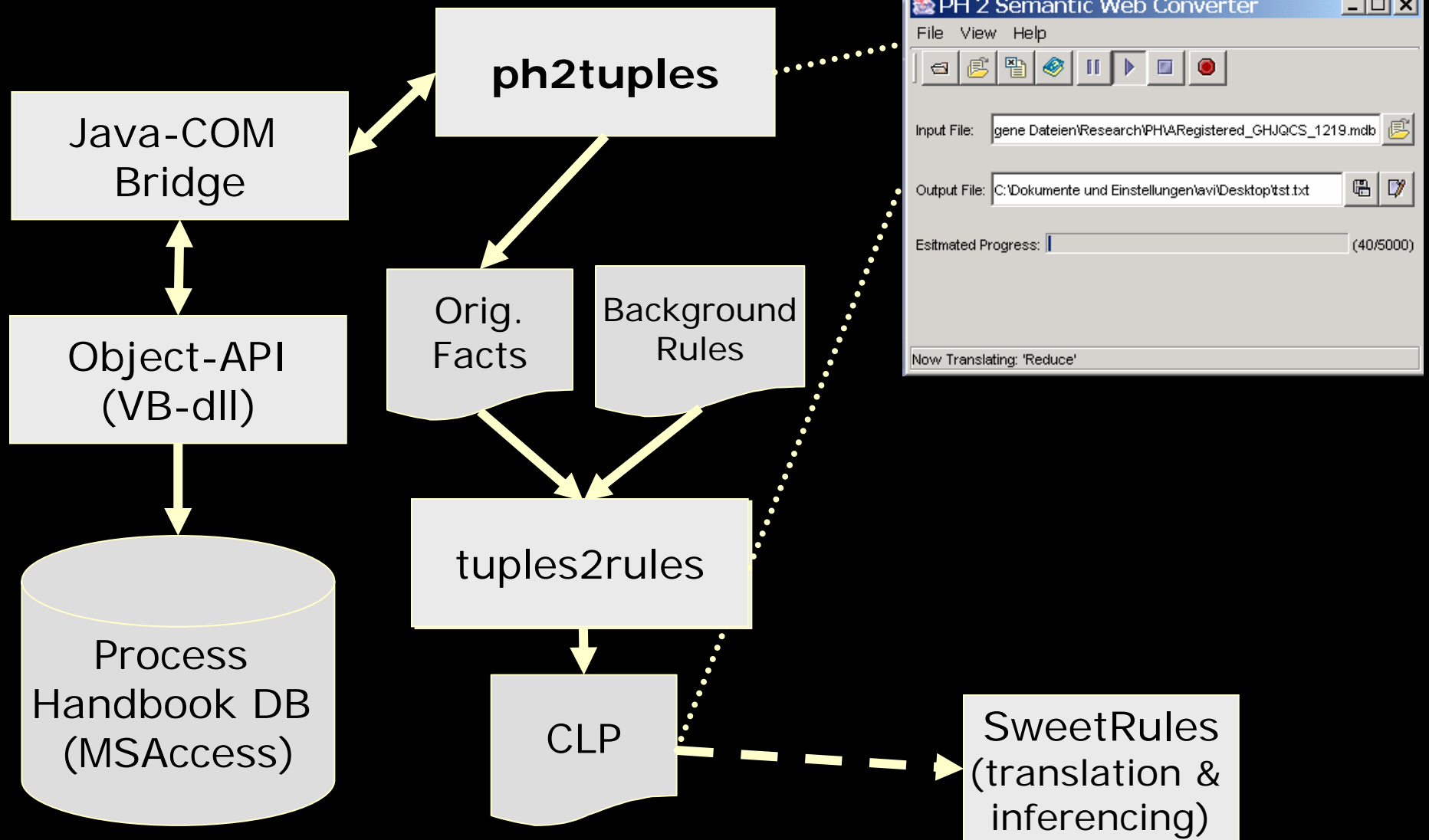
# Original PH Data Base E-R Model



# *Hurdles Encountered when Translating the Process Handbook*

- Nonmonotonic
  - FOL (including OWL) cannot represent
- Inheritance semantics hidden in code
  - Need to rationally reconstruct
- Only derived assertions are saved
  - Need to reconstruct premises
- Concept of slotted predicates
  - Use n-tuples
- Class as instance
  - *In-progress: combining with class as predicate*

# Translation Processing Architecture



# *Output Background rules*

- ~ 50 Background rules in CLP (~80 OLP)
- Transitivity of subclasses
- Domain and range for properties
- Partial functionality of slotted properties
- Axiomatization of inheritance prioritization partial order
- Default inheritance for properties

# Output

## Partial Output on Process “Sell” I

```
/* Declare subtype relationship 'Sell_263900' of 'Exchange_74000' */  
subclassof('Sell_263900', 'Exchange_74000');
```

```
/* Declare type 'Sell_263900' */  
class('Sell_263900');
```

```
/* Declare subtype relationship 'Sell_263900' of 'activity' */  
subclassof('Sell_263900', 'activity');
```

```
/* New value for 'has_attribute' at entity: Sell_263900 and slot: ph_Description */  
<lb4987>
```

```
pr(1a4987, 'Sell_263900', 'has_attribute', 'ph_Description', "Selling implies an exchange of  
value from the customer to the seller for a product and/or service. _cr_nl_cr_nlNote that the  
subactivities in 'sell' are the converse of 'buy'.");
```

```
/* New value for 'has_attribute' at entity: Sell_263900 and slot: ph_Name */  
<lb4997>
```

```
pr(1a4997, 'Sell_263900', 'has_attribute', 'ph_Name', "Sell");
```

```
/* New value for 'has_attribute' at entity: Sell_263900 and slot: ph_PIFID */  
<lb5003>
```

```
pr(1a5003, 'Sell_263900', 'has_attribute', 'ph_PIFID', "960823131555AB2639");
```



## *Output: Partial Output on Process “Sell” II*

```
/* New value for 'has_task' at entity: Sell_263900 and slot:  
960823131555AB2639SL1367 */
```

```
<lb5008>
```

```
pr(la5008, 'Sell_263900, 'has_task, '960823131555AB2639SL1367,  
'Identify_potential_customers_53400);
```

```
/* New value for 'has_task' at entity: Sell_263900 and slot:  
960823131555AB2639SL1369 */
```

```
<lb5009>
```

```
pr(la5009, 'Sell_263900, 'has_task, '960823131555AB2639SL1369,  
'Identify_potential_customers'_needs_328100);
```

```
/* New value for 'has_task' at entity: Sell_263900 and slot:  
960823131555AB2639SL1368 */
```

```
<lb5010>
```

```
pr(la5010, 'Sell_263900, 'has_task, '960823131555AB2639SL1368,  
'Inform_potential_customers_98400);
```

## *Output: Partial Output on Process “Sell” II*

```
/* New value for 'has_task' at entity: Sell_263900 and slot: 960823131555AB2639SL1366 */  
<lb5011>
```

```
    pr(la5011, 'Sell_263900, 'has_task, '960823131555AB2639SL1366,  
    'Obtain_order_280400);
```

```
/* New value for 'has_task' at entity: Sell_263900 and slot: 960823131555AB2639SL1371 */  
<lb5012>
```

```
    pr(la5012, 'Sell_263900, 'has_task, '960823131555AB2639SL1371,  
    'Deliver_product_or_service_262300);
```

```
/* New value for 'has_task' at entity: Sell_263900 and slot: 960823131555AB2639SL1370 */  
<lb5013>
```

```
    pr(la5013, 'Sell_263900, 'has_task, '960823131555AB2639SL1370,  
    'Receive_payment_53800);
```

```
/* New value for 'has_task' at entity: Sell_263900 and slot: 960823131555AB2639SL3867 */  
<lb5014>
```

```
    pr(la5014, 'Sell_263900, 'has_task, '960823131555AB2639SL3867,  
    'Manage_customer_relationships_267400);
```

## *Sample Conclusion*

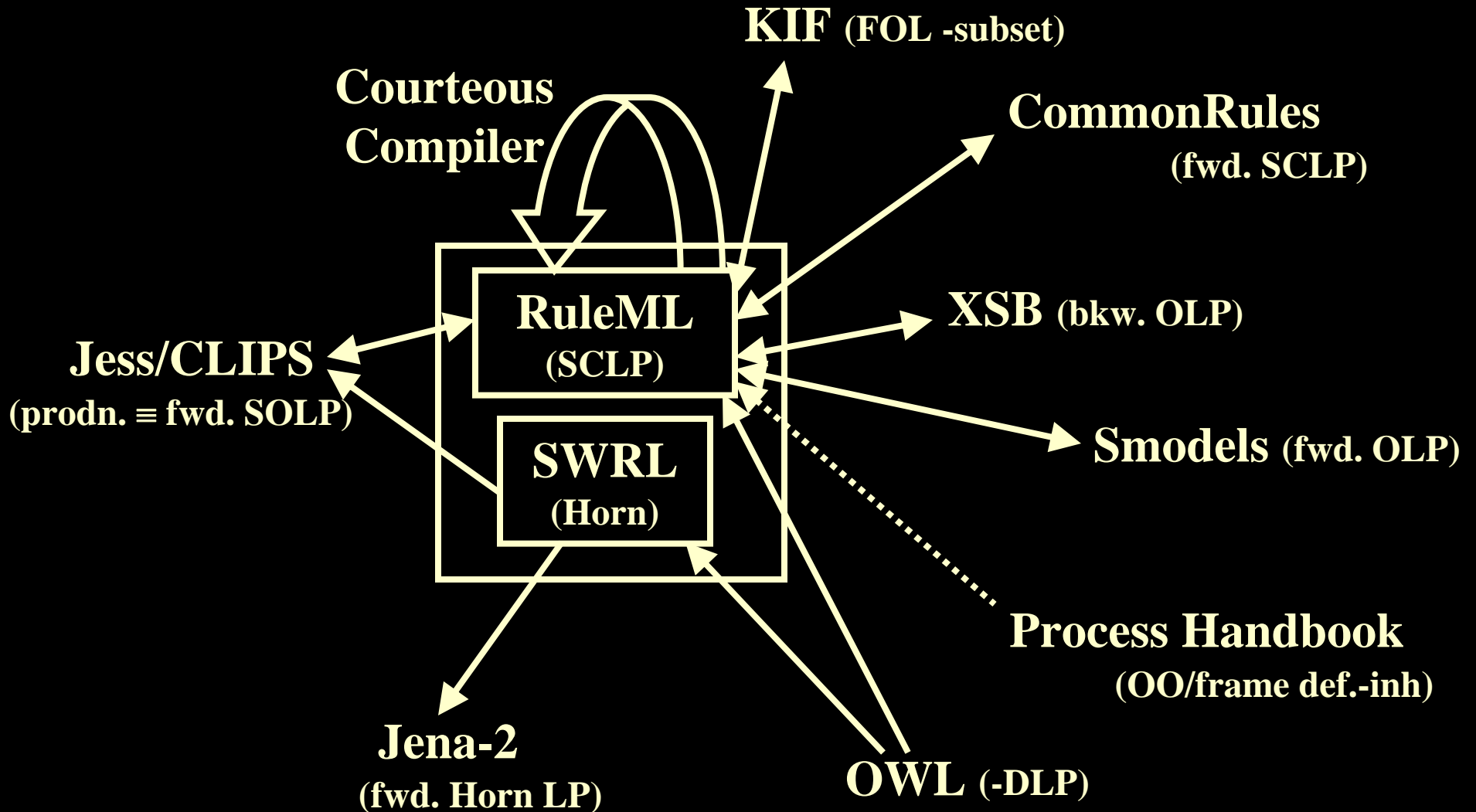
```
/* Sell_by_mail_order has subactivity  
   Deliver_product.
```

This is inherited by default from Sell\_Product.

```
*/
```

```
h('Sell_by_mail_order,  
   'has_task,  
   960823131555AB2639SL1371,  
   'Deliver_product).
```

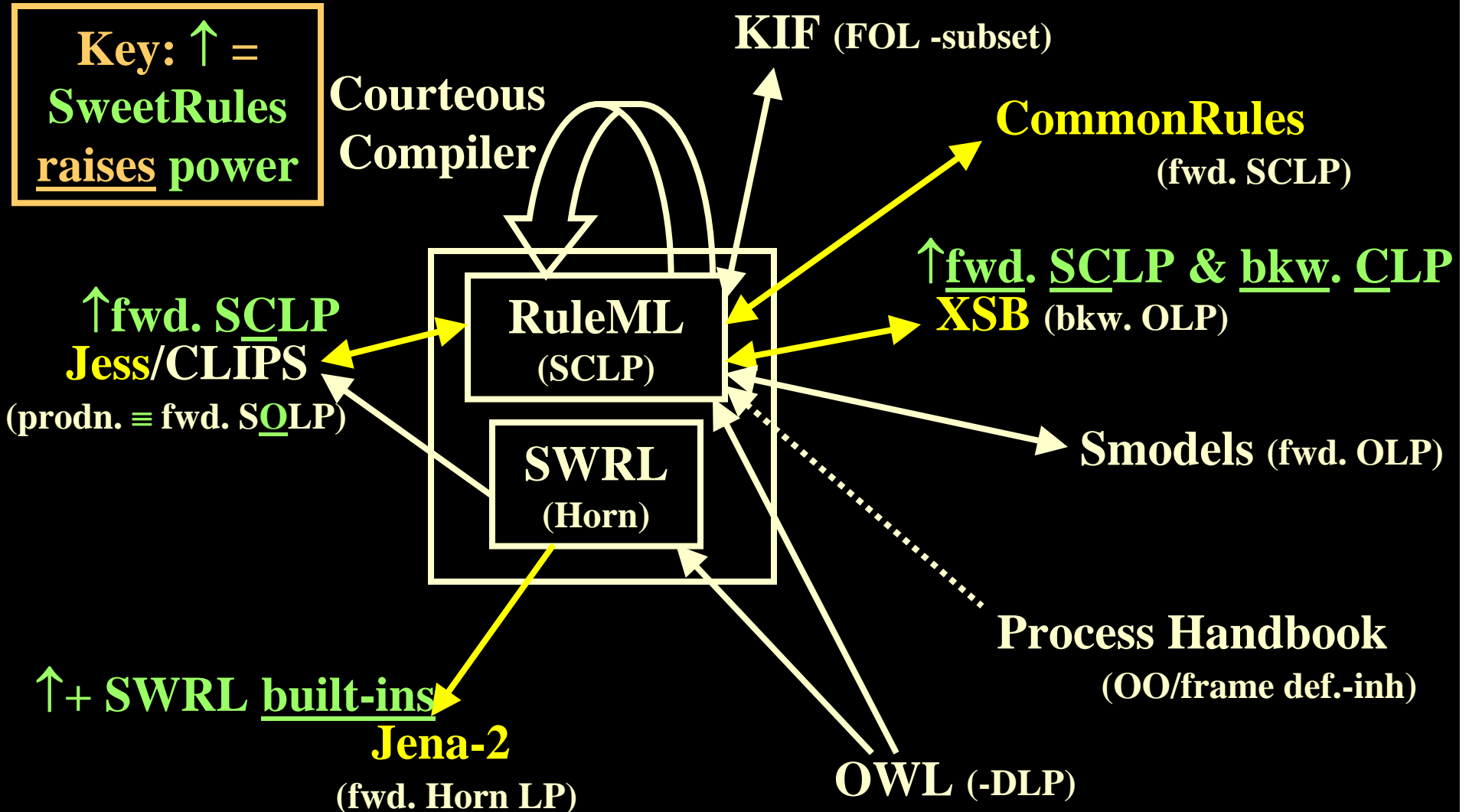
# *SweetRules Today: Translators Graph*



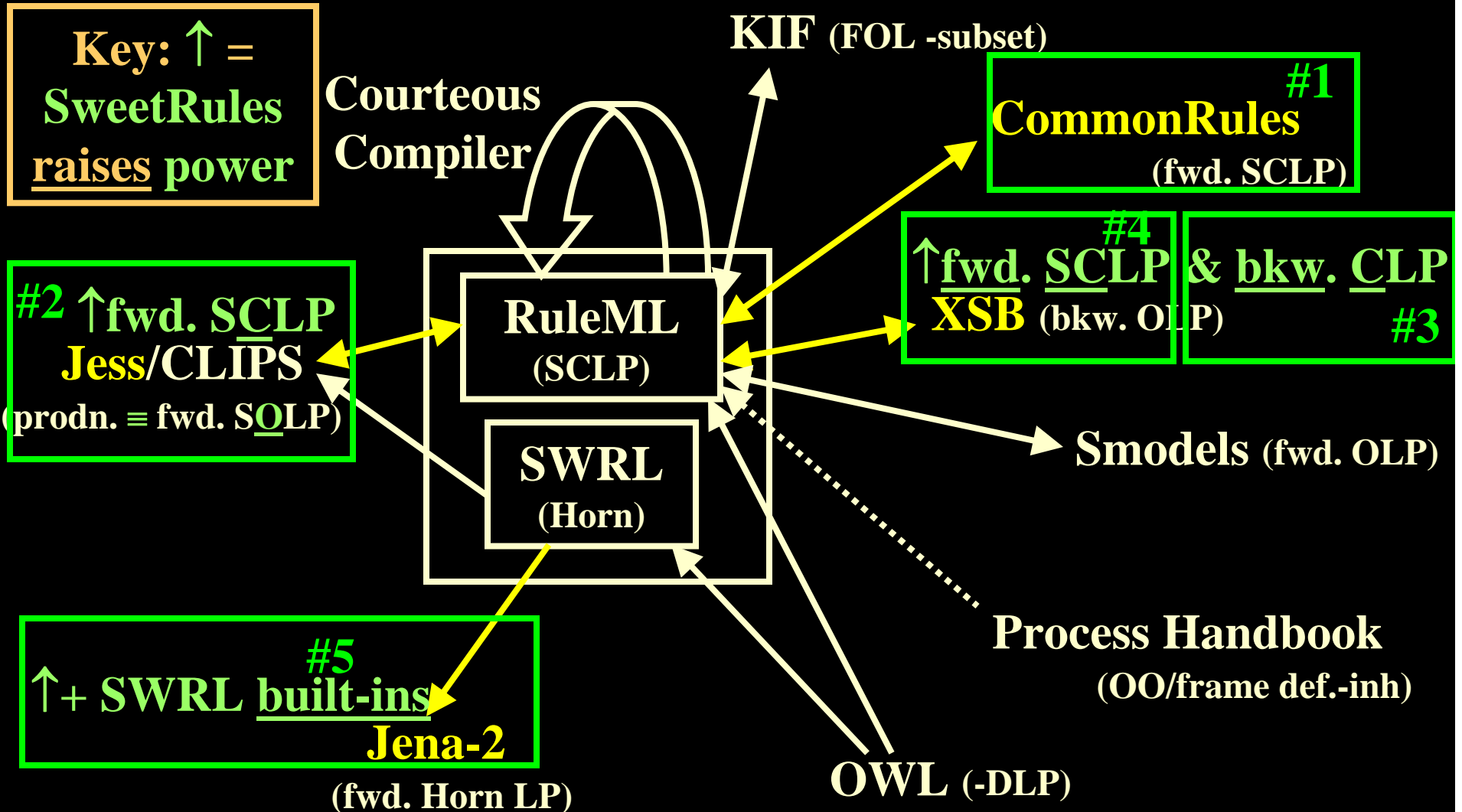
# *SweetRules Inferencing Capabilities Today: Overview*

- **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 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
- 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**