SweetPH: Using the Process Handbook for Semantic Web Services

Benjamin Grosof* and Abraham Bernstein**

*MIT Sloan School of Management, Information Technologies group, http://ebusiness.mit.edu/bgrosof

**U. Zurich, Dept. of Informatics, http://www.ifi.unizh.ch/~bernstein

Slides presented at SWSL F2F (Semantic Web Services Initiative's Language Committee Face-to-Face Meeting), Hawthorne, New York, Dec. 9-10, 2004

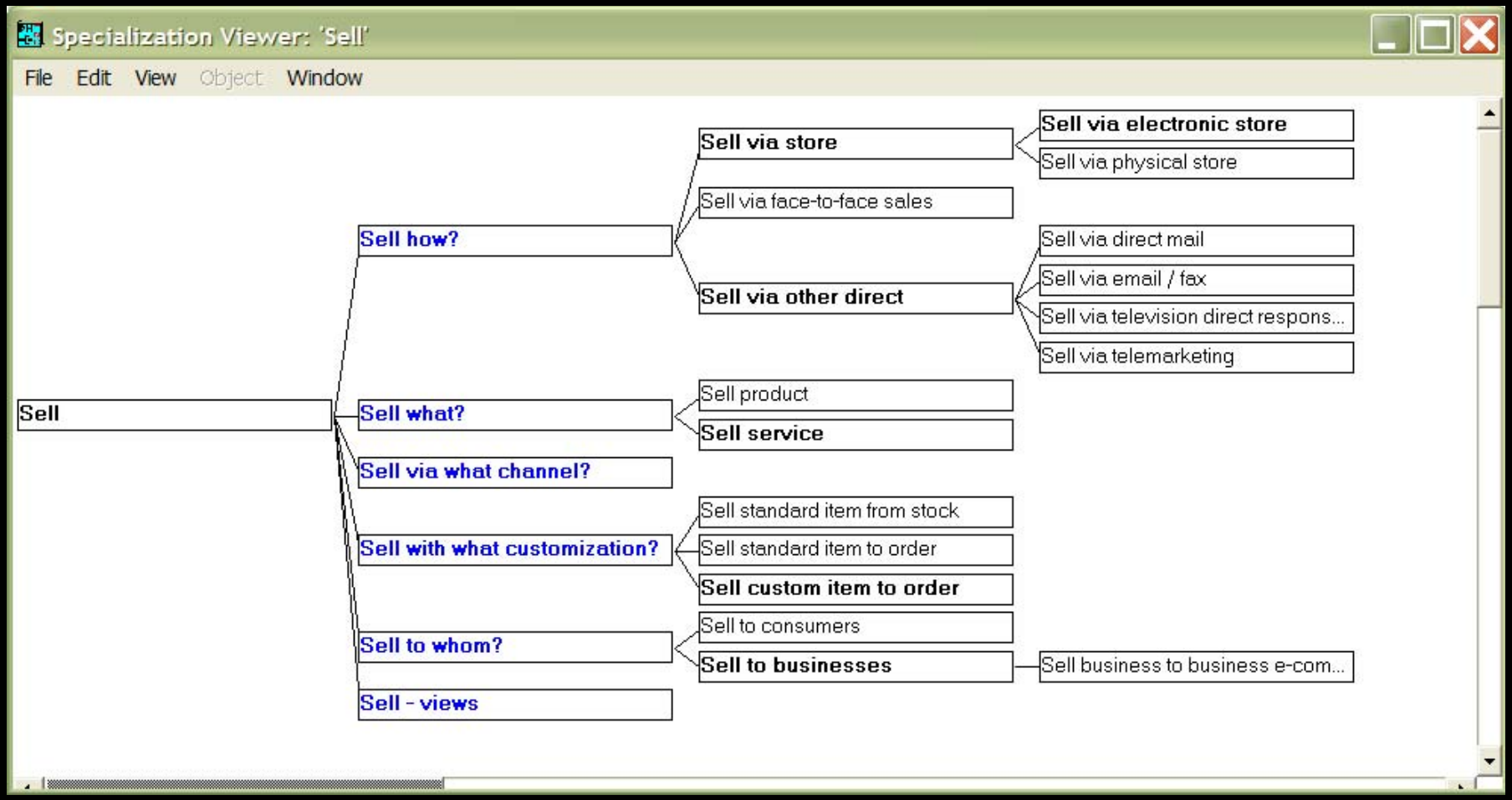
http://www.swsi.org

Opportunity for Process Handbook in SWS

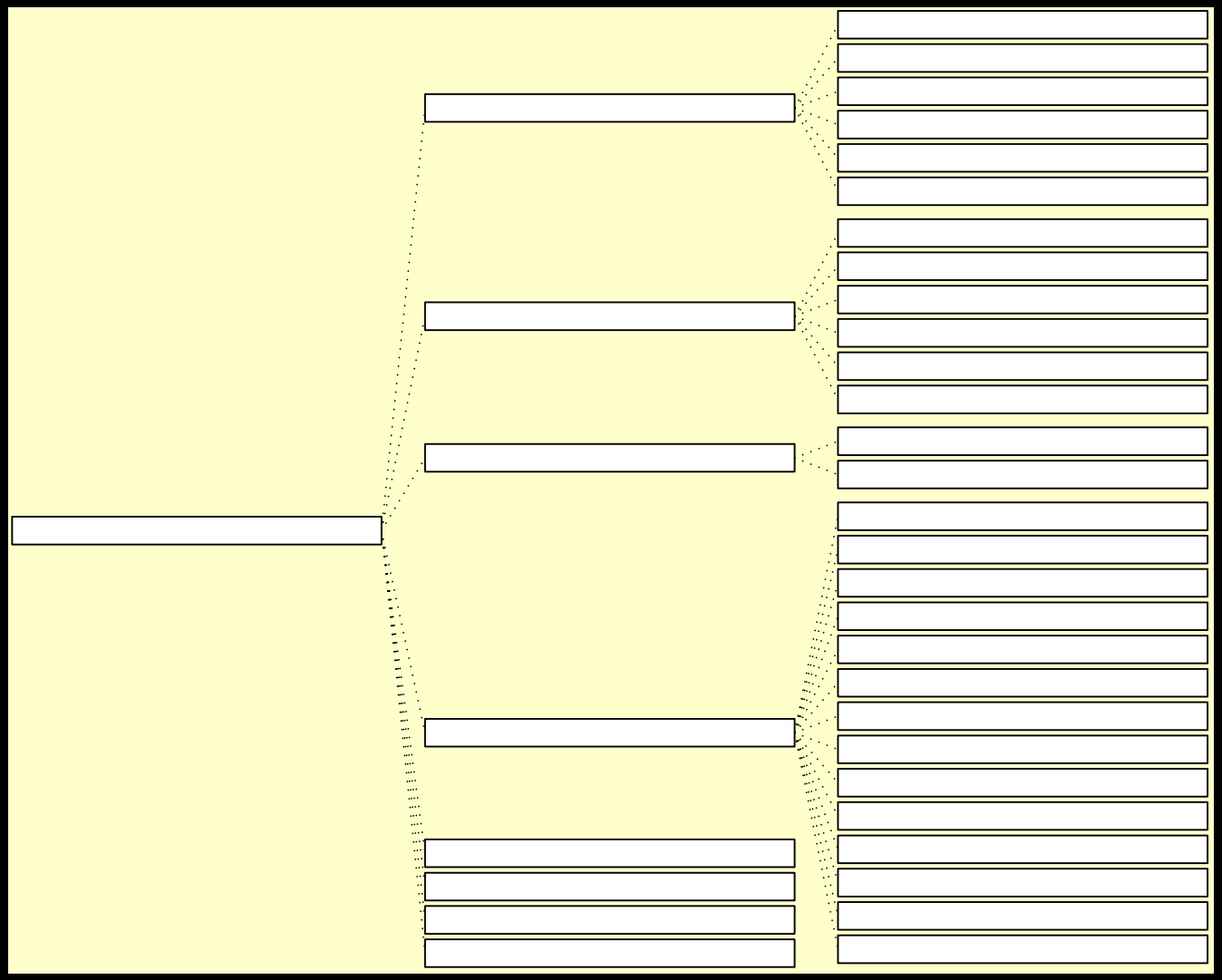
- Need for Shared Knowledge Bases about Web Services / Business Processes
  - For Semantic Web Services, etc.
- Want to leverage legacy process knowledge content
  - Go where the knowledge already is

- 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

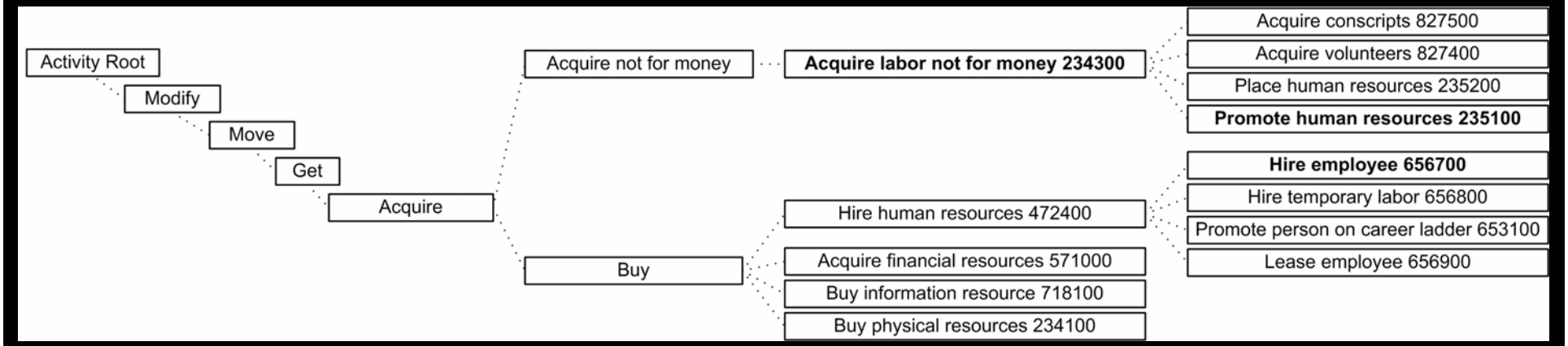
Some Specializations of "Sell" in the Process Handbook (PH)



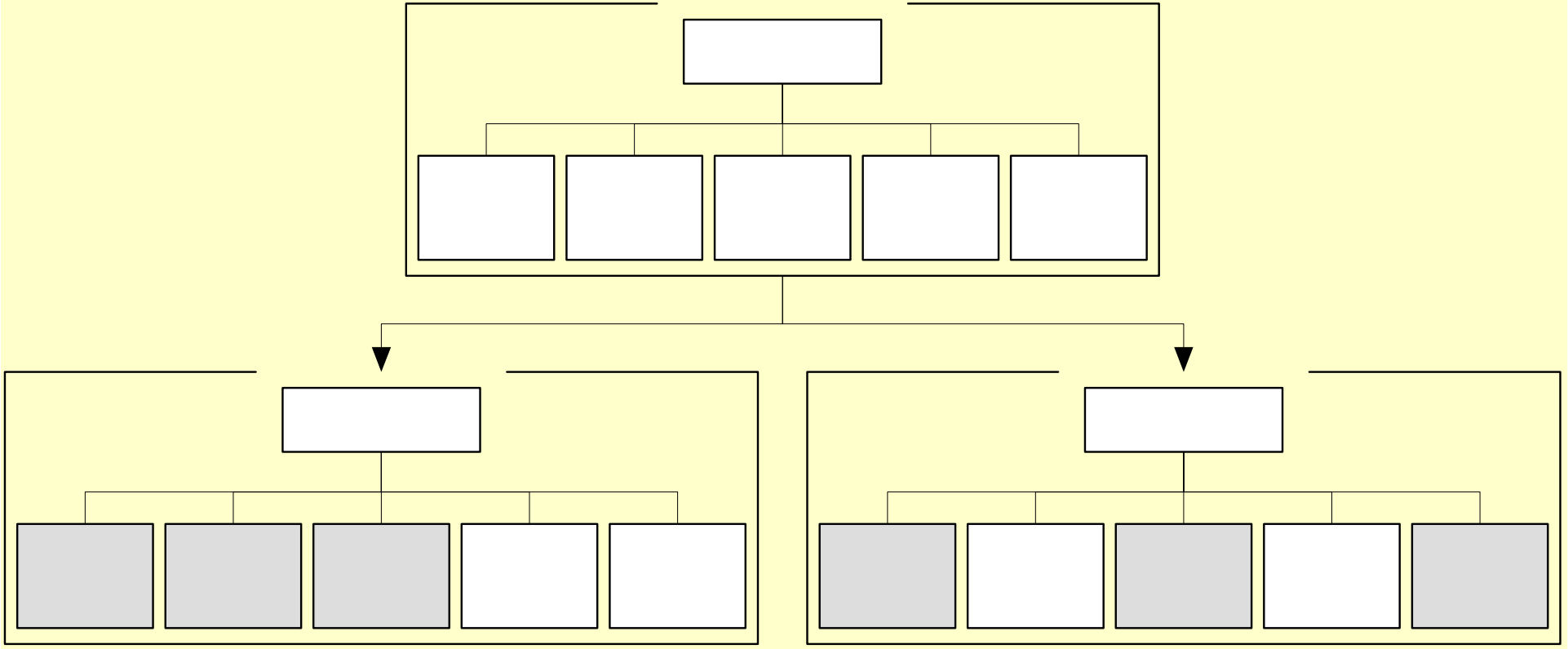
Some Process Handbook Ontology



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).

ent

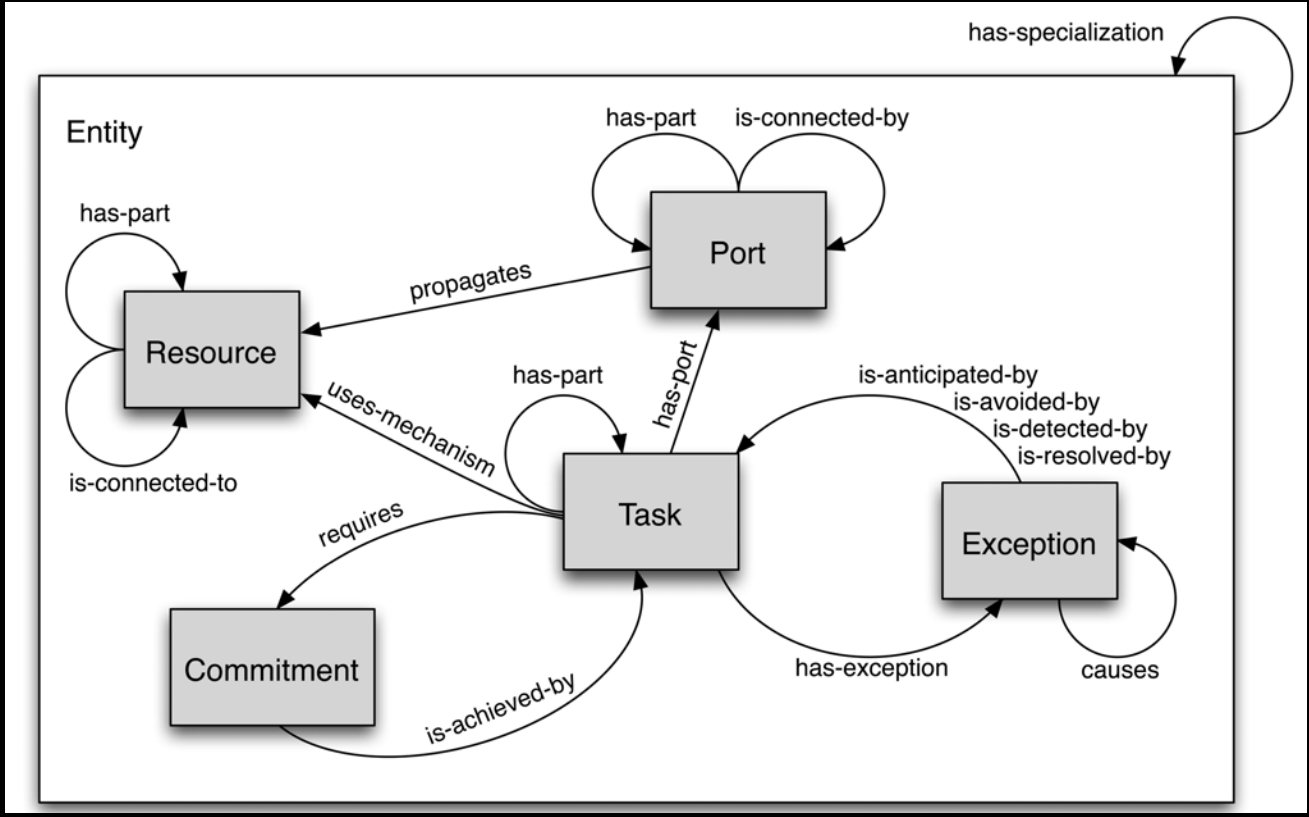
poten

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

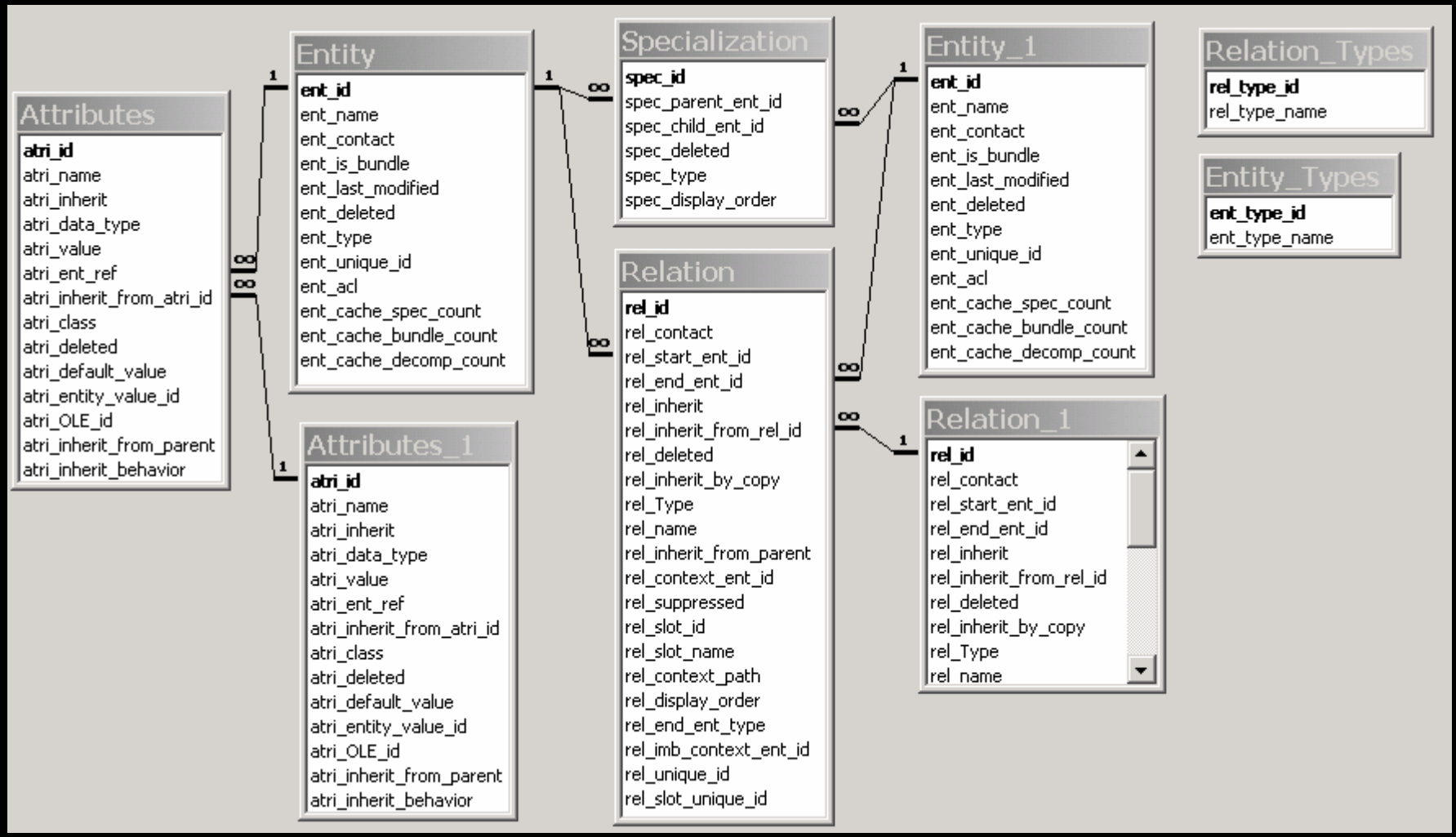
- Surprise: use SW *rule* language not the main SW *ontology* language! I.e., use (SCLP) RuleML not OWL.
  - OO inheritance is *default* ⇒ more reuse in ontologies
  - OWL/FOL cannot represent default inheritance
  - RuleML/nonmon-LP can
- Courteous Inheritance approach translates PH to SCLP KR
  - A 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*

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



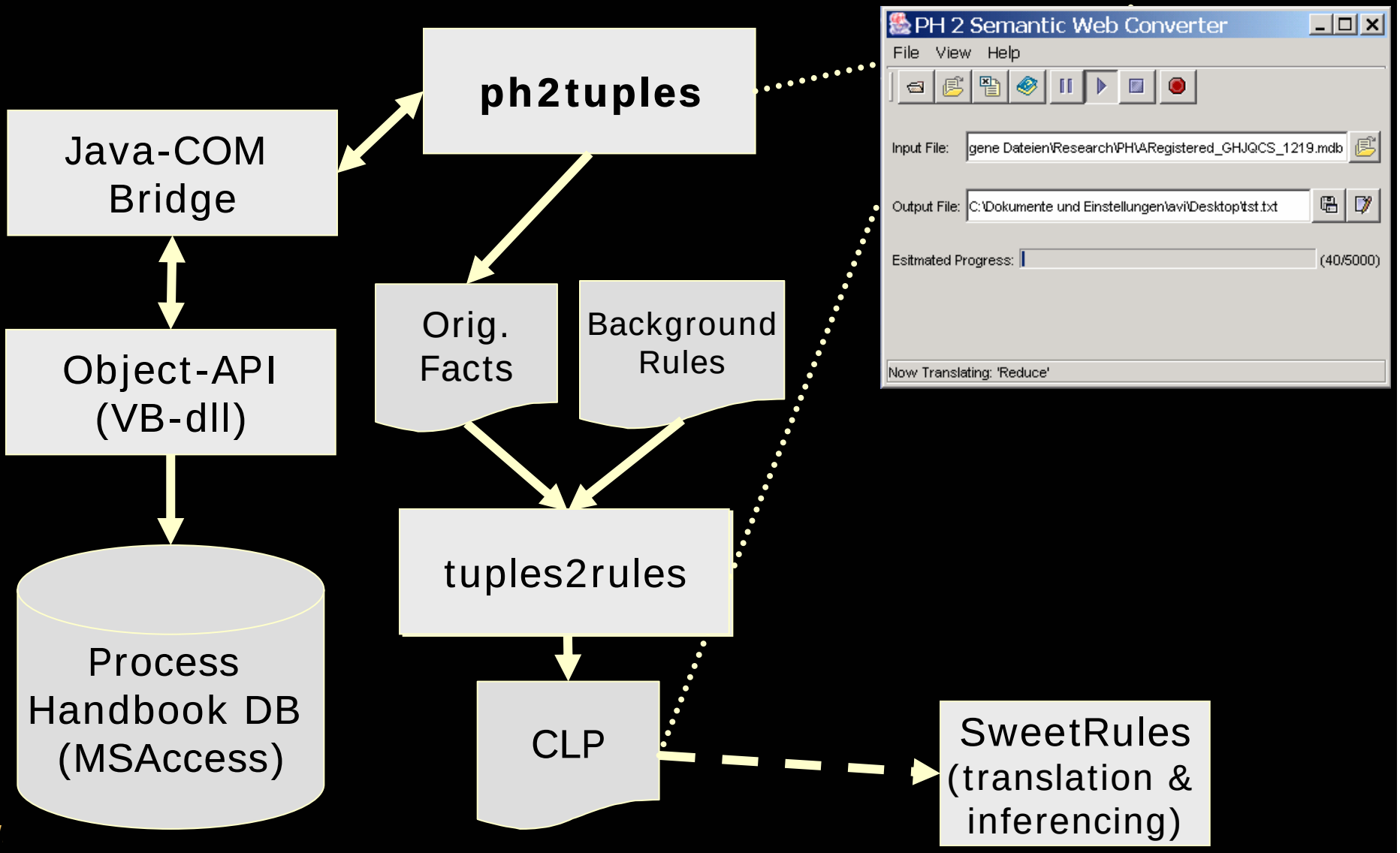
Original PH Data Base E-R Model



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(la4987, '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(la4997, 'Sell_263900, 'has_attribute, 'ph_Name, "Sell");

/* New value for 'has_attribute' at entity: Sell_263900 and slot: ph_PIFID */
<lb5003>
        pr(la5003, '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);
```

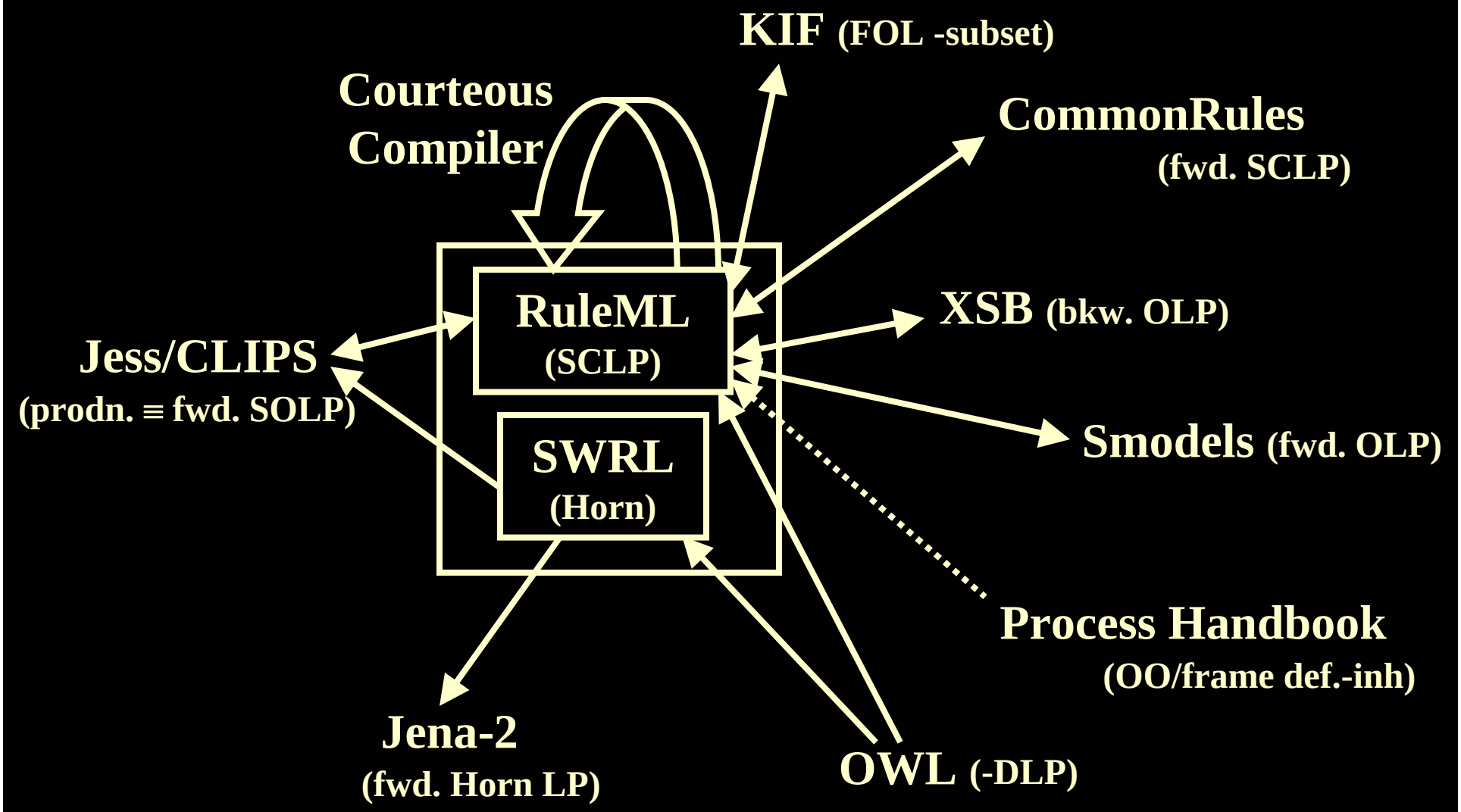
Sample Conclusion

/* 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).

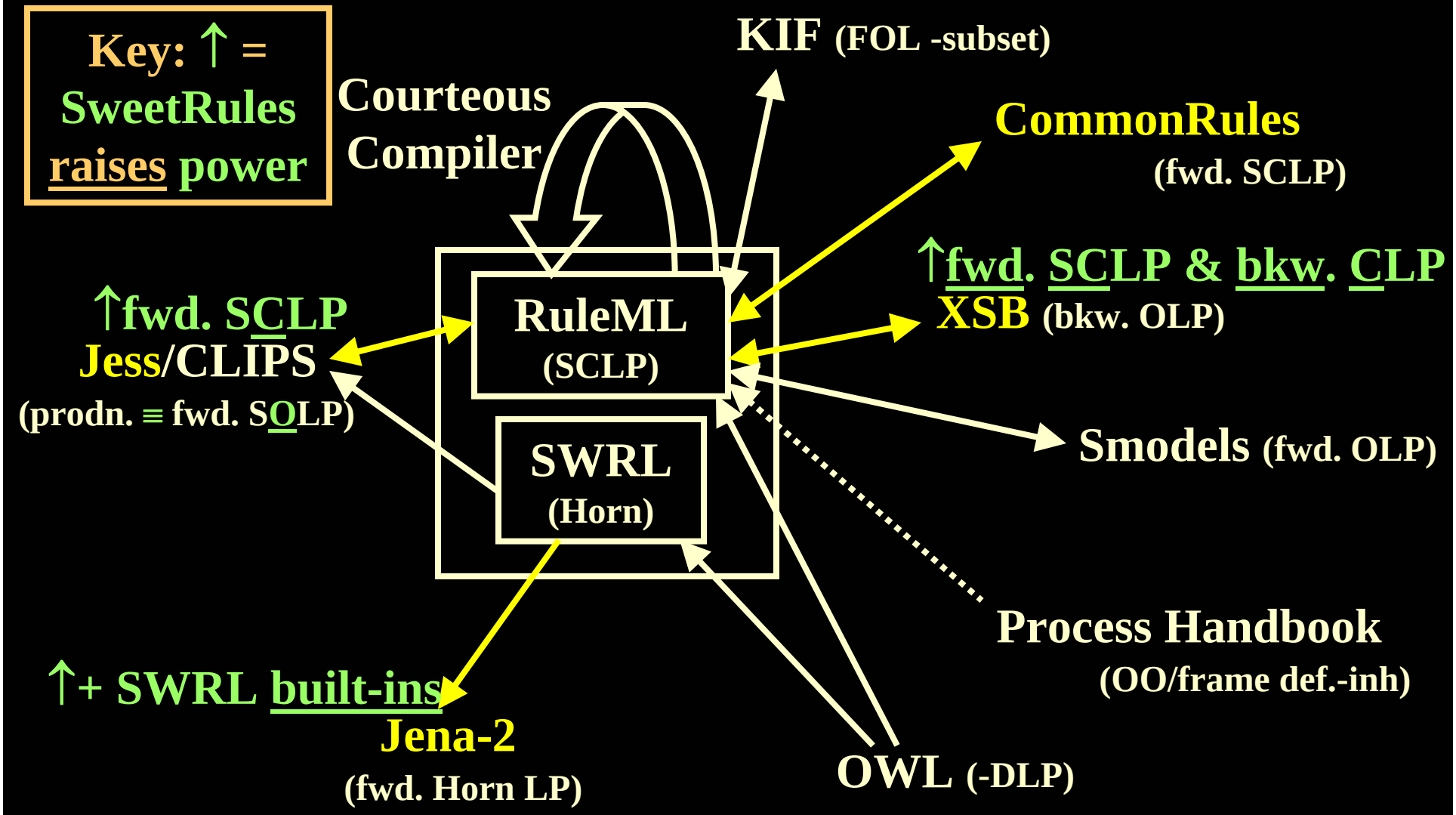
SweetRules Today: Translators Graph



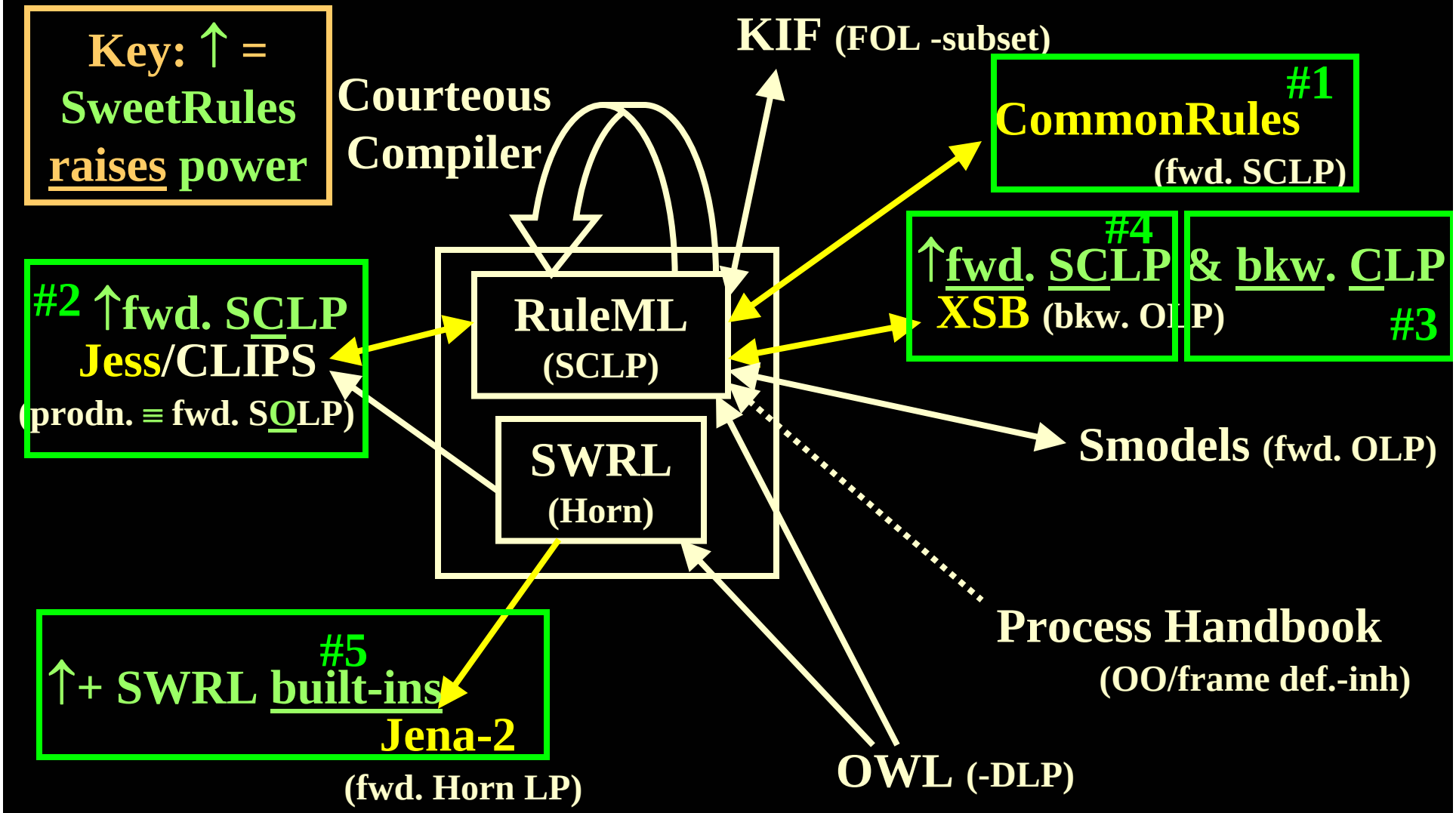
SweetRules Inferencing Capabilities Today: Overview

- 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 ↔ 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