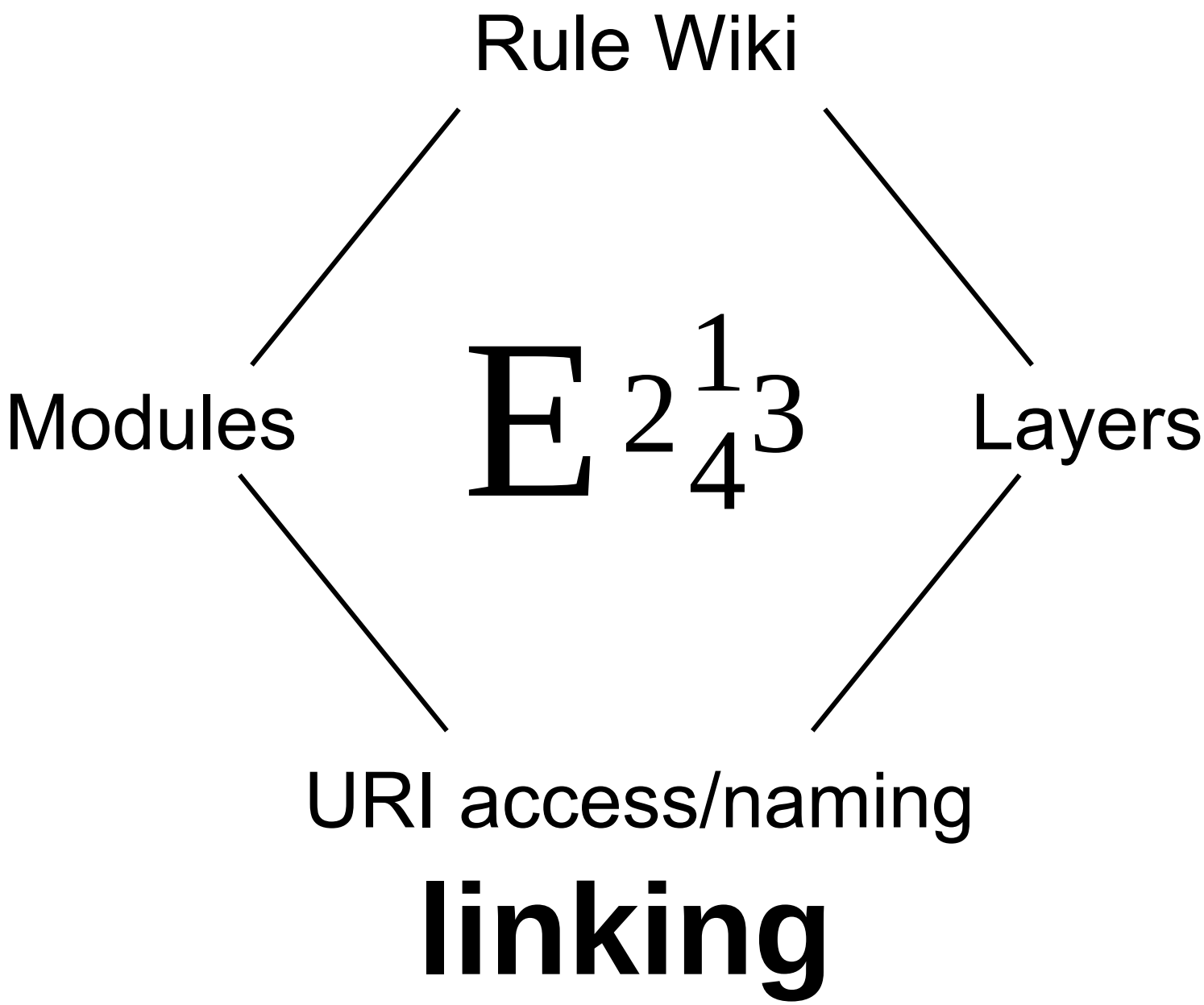
From Linked Data to Linked Rules: Web Rule Essentials

Talk at Semantic Days 2011
Oslo, Norway, 7-9 June 2011

Harold Boley
National Research Council of Canada
University of New Brunswick, Canada



linking

Introduction

- Web rules permit novel Web sites
  - with machine-interpretable rule representations
  - for automated reasoning
- Research builds on our previous work in
  - Web rule foundations (e.g., POSL, Datalog^{DL}, ALC$^{u}_{P}$)
  - Standards (e.g., RuleML, SWRL, RIF)
  - Engines (e.g., OO jDREW)
  - Use cases (e.g., AgentMatcher, FindXpRT, Rule Responder, Ontology Integration)

Objective

- Devise complementary techniques of
  - rule representation & reasoning
- for
  - Business Rules
  - the Semantic Web
  - Web Services
  - other Web (and Web 2.0) areas

Four Principal Web Rule Issues

Previous research led to following four principal Web rule issues used here as starting points ...

I1: Formal Knowledge as Content or Metadata

- Web increasingly has ‘Semantic Subwebs’ containing knowledge documents (knowledge bases, schemas, etc.)
- Formal knowledge representation can act as
  - content that is queried and retrieved in its own right
  - metadata that helps to retrieve other formal or informal content
  - a combination of both

I2: Global Inconsistency vs. Local Consistency

- Open Web as well as closed ‘intranets’ contain knowledge documents:
  - Open Web knowledge in expressively rich representations is typically inconsistent
  - Closed intranet knowledge is typically *paraconsistent*, i.e. the documents in each intranet are maintained to be (locally) consistent, although the intranet union may be (globally) inconsistent

→

I2: Global Inconsistency vs. Local Consistency (Cont'd)

- While classical 2-valued logic
  - cannot be directly used for open Web reasoning
  - it *can* be exploited locally for closed intranet reasoning
- Locality of documents creates an implicit module notion
- Locality can also be achieved via an explicit module construct

I3: Rule Layering on Top of RDF?

- Trade-off between representation expressiveness and reasoning tractability
- $\rightarrow$ Scalability of reasoning to the open Web is still unresolved for higher expressive classes
- $\rightarrow$ Representation layering on top of quite inexpressive languages:
- RDF is W3C's fundamental knowledge layer, although its XML syntax is somewhat complicated, and its semantics is rather complicated

I3: Rule Layering on Top of RDF? (Cont'd)

- Yet, simple RDF statements without blank nodes in assertions, queried without property variables, are a candidate for the least expressive (binary-)fact layer: Use RuleML & RIF's slotted syntax (F-logic semantics)
- Binary Datalog rules, similar to relational views over 2-column tables, can then be added to derive new facts from conjunctions of other facts, much like relational joins
- Finally, an irreflexive subClassOf fragment of RDF Schema can be employed to define order-sorted types for constants and rule variables

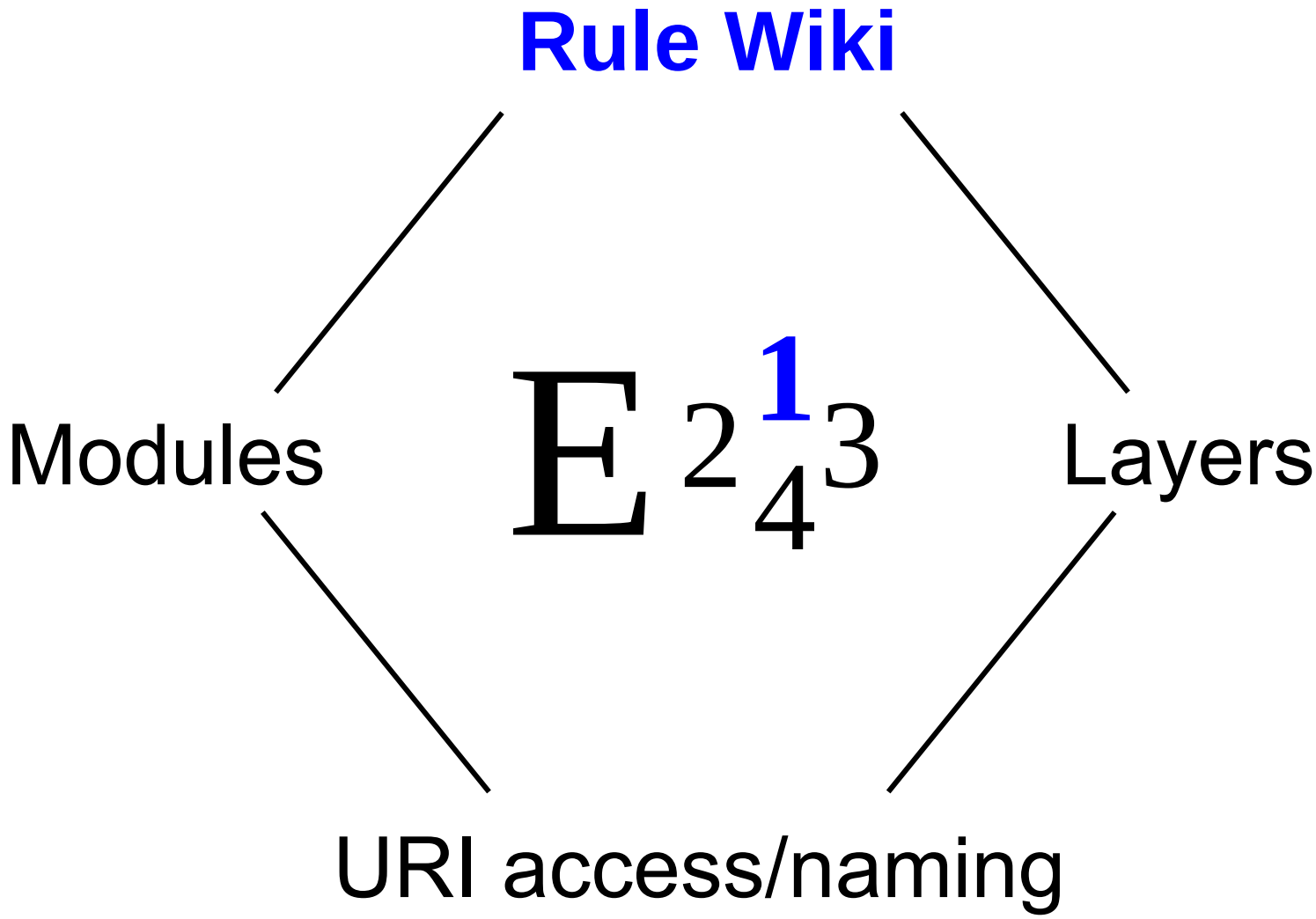
I4: Web Standards Compatibility / Webizing

- Web rules layered on, and side-by-side with, other Web languages
- → Represent rules so that compatibility with relevant Web standards (e.g., XML, RDF, OWL) is preserved
- Selecting Web standards can be hard, e.g. which, if any, query and transformation languages should be included (e.g., XQuery, XSLT, SPARQL, OWL-QL)
- Levels and degrees of compatibility with each selected language need to be determined
- → Focus on what is unique to Web languages, namely 'webizing', basically permitting Web rule language to use URIs for global constants, in ways compatible with URIs in existing (Semantic) Web languages

Four Essentials of Web Rules

To address issues I1-I4, we will consider four corresponding essentials of Web rules, E1-E4

Taken together, the Web rule essentials will constitute a diamond-like system, $\mathrm{E}^{1}_{2\ 4}{}^{3}$, with URIs (E4) at the bottom, modules (E2) and assertional-terminological layers (E3) on the same level in the middle, and a Controlled English Wiki (E1) at the top


Rule Wiki
Modules
E 2 1 3 4
Layers
URI access/naming

E1: Combining Logic Rules with Controlled English

- Combine formal and informal knowledge in a Rule Wiki, where clauses (here, rules and facts) are given dual representations, in natural language (e.g., English) and in logic
- Formal parts can be taken as code (or as metadata) for the informal parts, and the informal parts as documentation (or as content) for the formal parts

E1: Combining Logic Rules with Controlled English (Cont'd)

- This combination is analogous to Knuth's Literate Programming and to Javadoc
- Supported by tools mapping Controlled English into rules and back
  - English-to-rule tools based on Attempto: TRANSLATOR, and AceRules
  - Related tools have also been developed for AceWiki and "Semantics of Business Vocabulary and Business Rules" (SBVR)

E1: Rule(ML) Wiki

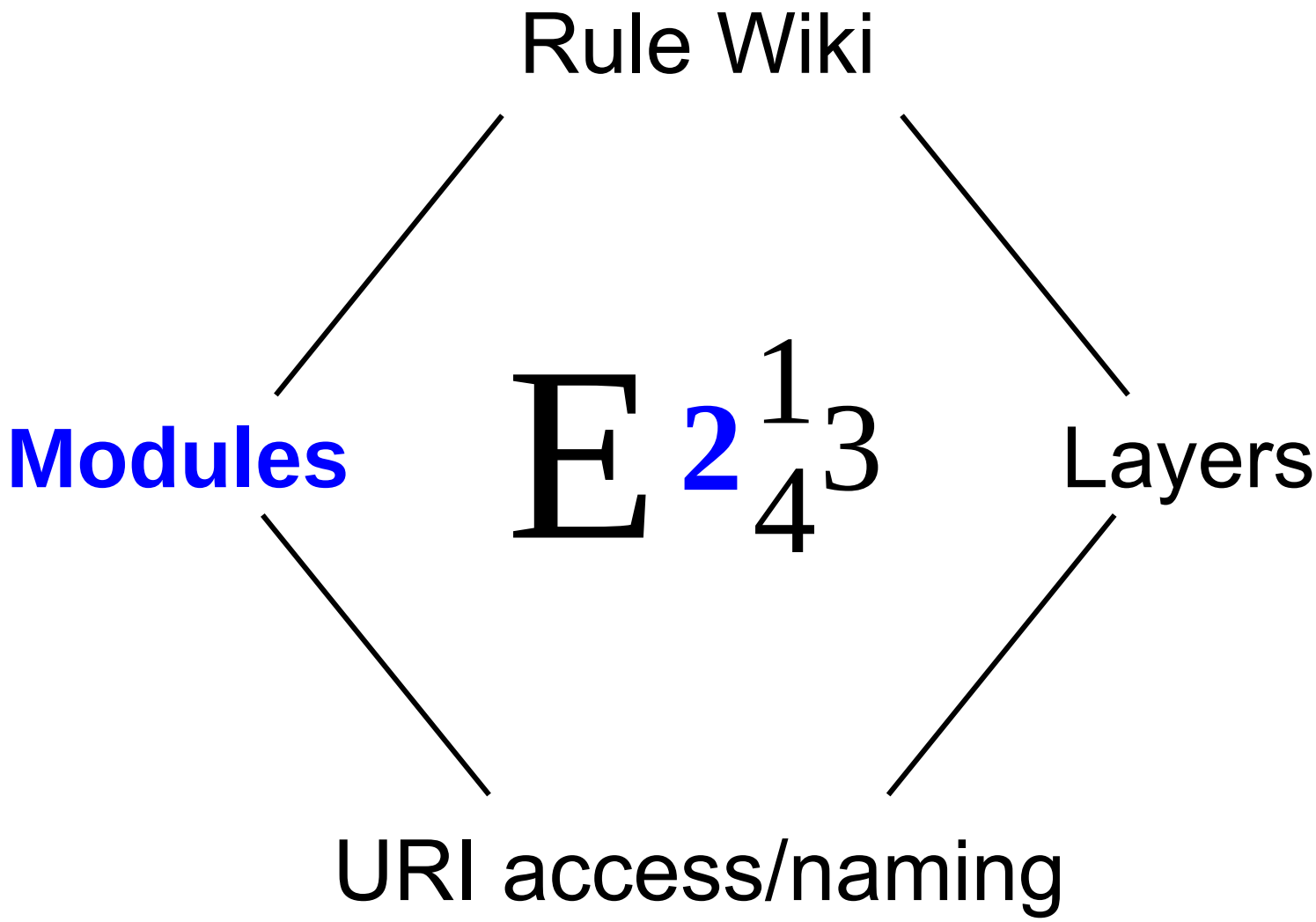
- Classical Wiki permits authoring of informal-knowledge documents using natural-language-enriching markup simpler than (but mapped to) HTML
- Extending this concept, a Rule Wiki permits formal-knowledge authoring using logic-language-enriching markup simpler than (but mapped to) XML, combining this with informal-knowledge authoring
- Formal-knowledge language can employ a human-readable syntax such as POSL, integrating the Prolog and F-logic syntaxes

E1: Rule(ML) Wiki (Cont'd)

A reciprocal shipping, **reciship**, at total amount, *cost*, takes place between sites *A* and *B* if

- a **shipment** has a **cargo**, a **price** of *cost1*, a **source** of *A*, and a destination, **dest**, of *B* and
- a **shipment** has a **cargo**, a **price** of *cost2*, a **source** of *B*, and a destination, **dest**, of *A* and
- *cost* is the addition, **add**, of *cost1* and *cost2*.

```
reciship(?cost,?A,?B) :-
  shipment(cargo->?;price->?cost1;source->?A;dest->?B),
  shipment(cargo->?;price->?cost2;source->?B;dest->?A),
  add(?cost,?cost1,?cost2).
```


Rule Wiki
Modules
E 2 1 3 4
Layers
URI access/naming

E2: A Distributed Rule Module Construct

- Beneficial to represent distributed knowledge via a module construct,
  - supporting local consistency
  - reducing the search space of scoped (module-restricted) queries
  - permitting scoped negation as failure (over closed worlds)
- Such Web modules may be
  - written and used ‘in place' or
  - defined at one place (URL) and accessed from other places
- The semantics of modules should not depend on any needed URL-dereferencing

E2: Modules in RuleML

- RuleML 0.91 embeds modules (Rulebases) into an Entails element, which serves to prove whether a query or module is entailed by another module
- Can be extended to nested (cycle-free) inheritance system of modules
- We only need a simplified kind of module inheritance, since by default we don’t assume Prolog-like textual order in a module's set of assertional (fact and rule) or terminological (subclass-ontological) clauses
- → Don’t need to merge clauses but can just take their union

E2: Module Example: `loyalty`

```
{
  discount(?customer,?product,percent[5]) :-
    premium(?customer),
    regular(?product).

  discount(?customer,?product,percent[10]) :-
    premium(?customer),
    luxury(?product).
}
```

E2: Module Example: `legality`

```
{
  -discount(?customer,?product1,?percent) :-
     payment(?customer,?product2,?amount,?method,?time),
     fraudulent(?customer,?method,?time).

  -discount(?customer,?product1,?percent) :-
     delivery(?customer,?product2,?amount,date[?y1,?m1,?d1]),
     payment(?customer,?product2,?amount,?method,date[?y2,?m2,?d2]),
     datediff(days[?delta],date[?y2,?m2,?d2],date[?y1,?m1,?d1]),
     greaterThan(?delta,45).
}
```

Using "-" prefix as POSL syntax for RuleML's (strong) Neg(ation) element

E2: Module Prioritization: `legality` vs. `loyalty`

Example modules locally consistent, but their union is inconsistent:

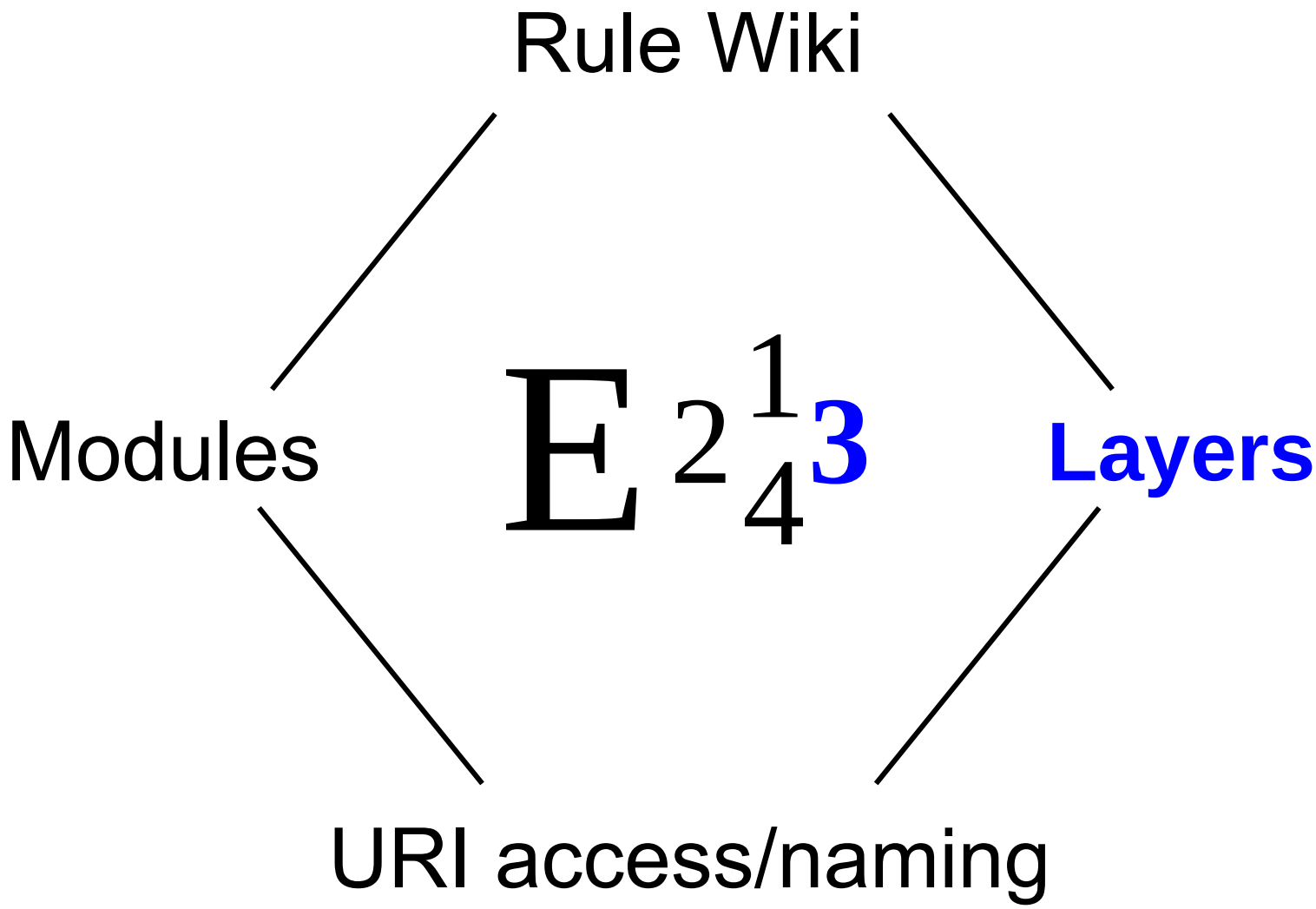
According to first rule of `loyalty` module, premium customers would be granted 5 percent discount for a regular product, but, according to the first rule of `legality` module, would be denied discount on any product if they used a fraudulent payment method on a product

To deal with this, prioritization (cf. Courteous Logic Programs and Defeasible Logic) can be employed on the module level to let all rules of the `legality` module *override* all `loyalty` rules

E2: Module-Scoped Queries — Example Based on Local Modules: `customer` and `product`

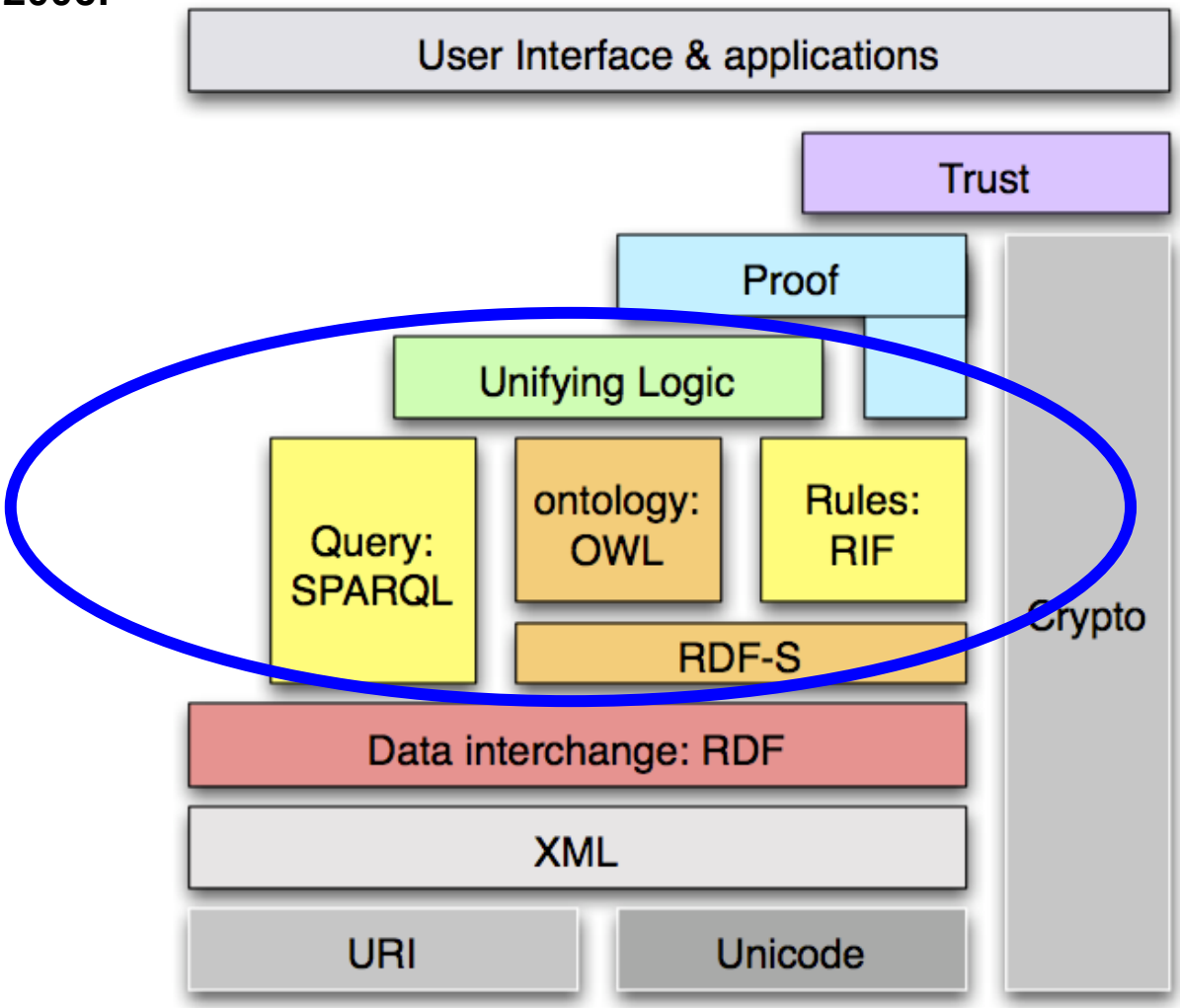
```
{
  discount(?customer,?product,percent[5]) :-
    customer |- premium(?customer),
    product |- regular(?product).

  discount(?customer,?product,percent[10]) :-
    customer |- premium(?customer),
    product |- luxury(?product).
}
```

Using “|-” infix as POSL syntax for RuleML’s Entails element


Rule Wiki
Modules
E 2 1 4 3
Layers
URI access/naming

E3: Assertional-Terminological Expressiveness Layering – Zooming into the 'Cake'

Tim Berners-Lee 2006:



E3: Assertional-Terminological Expressiveness Layering

- Various efforts towards dual expressiveness layering of assertional and terminological knowledge as well as their blends
- *Assertional bottom layer* usually consists of Datalog (function-free) assertions, perhaps restricted to unary/binary predicates
- *Terminological bottom layer* can employ irreflexive version of RDF Schema's subClassOf, which can later be extended towards the ρDF fragment of RDF

E3: Assertional-Terminological Expressiveness – Bottom Layers

Bottom layers can be blended through a

- **hybrid combination**: ρDF classes used as types for Datalog constants and variables, and subClassOf defined with order-sorted semantics

or

- **homogeneous integration**: ρDF classes used as unary predicates in the body of Datalog rules, and subClassOf defined as special rules with Herbrand-model semantics

E3: Assertional-Terminological Expressiveness – Higher Layers

Higher layers can develop

- Datalog into Horn and FOL (First-Order Logic) assertions
- ρDF into ALC and SHIQ terminologies with classes and properties
- appropriate blends, e.g. as advancements of our hybrid Datalog^{DL} or homogeneous ALC^{u}_{P}

Assertional layers can move even beyond FOL, including towards higher-order and modal logics, as started as part of the RuleML family

E3: Layering Example: Vehicle Registration

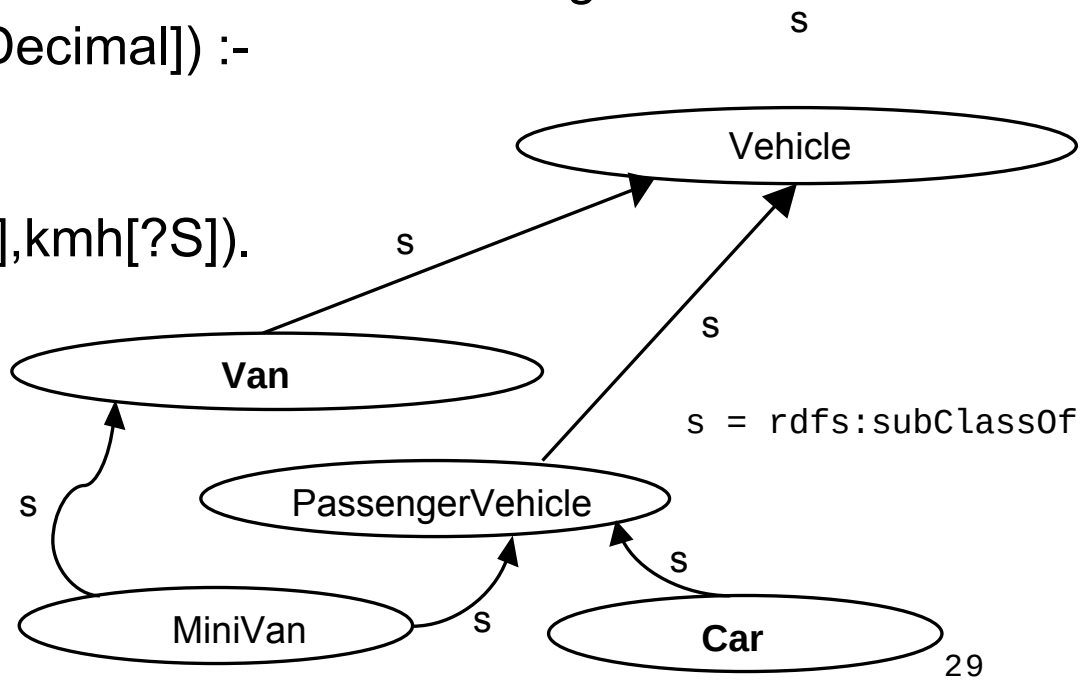
Assertions:

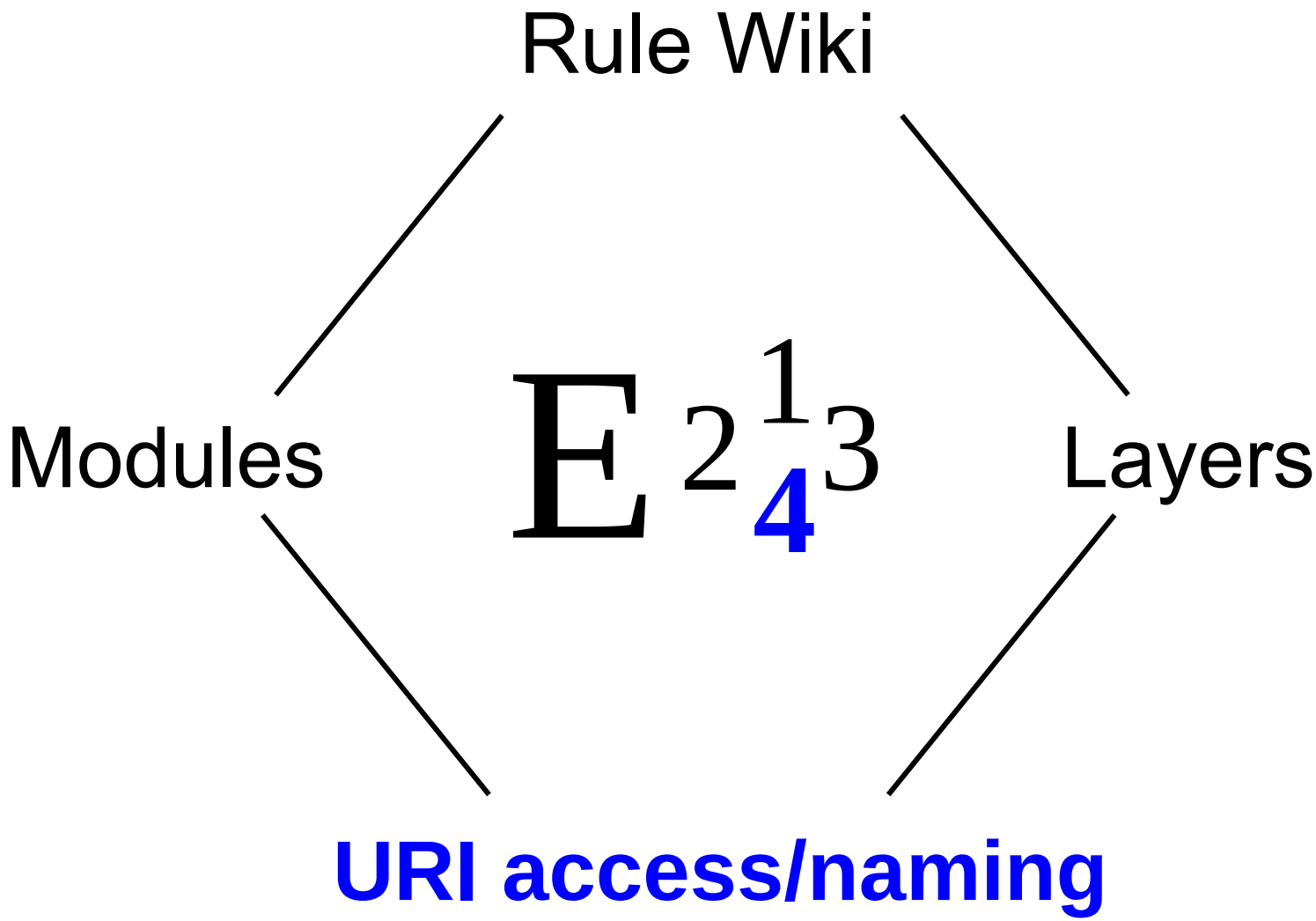
registration(?V:**Van**,CAD[?R:Decimal]) :-
  emission(?V,CO2[?E]),
  weight(?V,kg[?W]),
  emiweight(CAD[?R],CO2[?E],kg[?W]).

registration(?V:**Car**,CAD[?R:Decimal]) :-
  emission(?V,CO2[?E]),
  speed(?V,kmh[?S]),
  emispeed(CAD[?R],CO2[?E],kmh[?S]).

Terminology:

Vehicle > **Van**
Vehicle > PassengerVehicle
Van > MiniVan
PassengerVehicle > MiniVan
PassengerVehicle > **Car**




Rule Wiki
Modules
E 2 1 3 4
Layers
URI access/naming

E4: URIs for Access or Naming

- There have been attempts to differentiate the Web notion of URIs into two subnotions:
  - URLs (Uniform Resource Locators), for **access**
  - URNs (Uniform Resource Names), for **naming**

- In the context of Web knowledge representation, three central URI uses are emerging, given here in the order of further needed research

E4: URIs for Access, Naming, or Both (1)

A URI can be used **URL/access-style**, for module import, where it is an error if dereferencing the URI does not yield a knowledge base valid with respect to the expected representation language

Example: The `loyalty` module can be imported into the current rulebase using URL/access-style URI **http://modeg.org#loyalty**

E4: URIs for Access, Naming, or Both (2a)

A URI can be used **URN/naming-style**, as the identifier of an individual constant in the representation language, where URI dereferencing is not part of the formal knowledge representation. Dereferencing as part of the metadata about the informal knowledge representation retrieves ‘homepage’ of the individual

Example: The URI **http://en.wikipedia.org/wiki/Pluto** can be used URN/naming-style to refer to celestial body originally considered a planet (URI in angular brackets, <. . . >):

```
planet(<http://en.wikipedia.org/wiki/Pluto>,AD[?year]) :-
   lessThanOrEqual(1930,?year),
   lessThanOrEqual(?year,2006).
```

E4: URIs for Access, Naming, or Both (2b)

As part of formal rule knowledge, Pluto URI is used only for naming.
Rule can also be employed as metadata about informal knowledge via ('semantic search engine') queries like

planet(?which, AD[2005])

since one of its solutions is

?which = <http://en.wikipedia.org/wiki/Pluto>

whose dereferencing ('clicking') will then retrieve Pluto's Wikipedia entry

E4: URIs for Access, Naming, or Both (3)

A URI can be used **naming-style**, as identifier of a class, property, relation, or function, and at the same time **access-style**, where dereferencing yields knowledge base formally defining that identifier (perhaps partially only, as for an RDF Schema knowledge base just giving the superclasses of a class)

Example: for certain formal purposes a URI like **http://termeg.org#MiniVan** is needed just to provide a name; for other formal purposes, also to provide a total or partial definition found by using that same URI for access (say, the partial definition of being rdfs:subClassOf both http://termeg.org#Van and http://termeg.org#PassengerVehicle)

Conclusions (1)

- Essentials in $\mathrm{E}^{1}_{2}{}^{3}_{4}$ are variously interrelated
- For instance,
  - a Rule Wiki for assertional knowledge (E1)

  can be extended with
  - terminological knowledge (E3),

  both of which
  - can be kept in distributed modules (E2)
  - accessed by URIs (E4)

Conclusions (2)

- The four essentials can be transferred to (re)active rules for knowledge update, which have been increasingly studied in Web languages such as Reaction RuleML and Prova
- These rules have extra *event* and *action* parts:
  - Can also be combined with Controlled English (E1),
  - Modules are even more important here, for containing action side-effects (E2)
  - Terminologies can be directly added to formalize both event and action vocabularies (E3)
  - All kinds of URIs are also crucial for (re)active Web rules and Web Services (E4)