SWCLOS:
A Semantic Web Processor on CLOS
Advanced Discussion

Seiji Koide
National Institute of Informatics

Formal Semantics

- Platonic idealism (from Wikipedia)
  - Some contemporary linguistic philosophers construe "Platonism" to mean the proposition that universals exist independently of particulars (a universal is anything that can be predicated of a particular).
- Ontology
  - Ontology is the philosophical study of the nature of being, existence or reality in general, as well as of the basic categories of being and their relations.
- Formal …
  - How to describe things in objectivity
  - How to enable computation on ontology

Formal Semantics

- Equality
  - eq, eql, equal, equalp, =
  - Two names are lexically equal, then both equal?
  - Two names are different, then both different?
  - Two anonymous objects have the same structure and the same value, then both equal?
    - What do you mean by “same”?
  - Two classes share instances, then both equal?

Denotational and Extensional Semantics of RDF

based on RDF Semantics by
Patrick Hayes and Brian McBride

Russell&Norvig AIMA

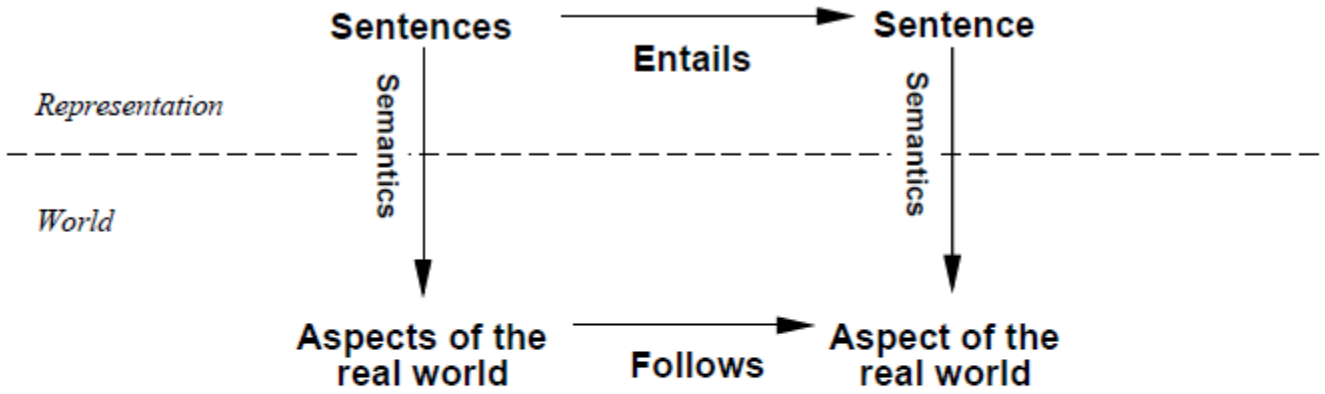
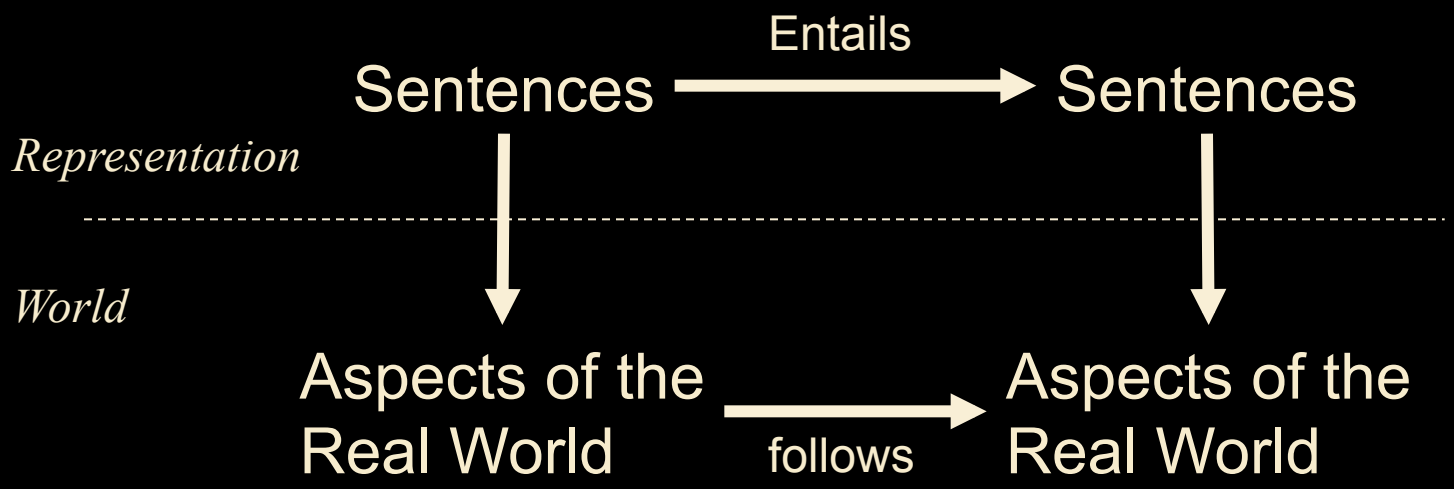
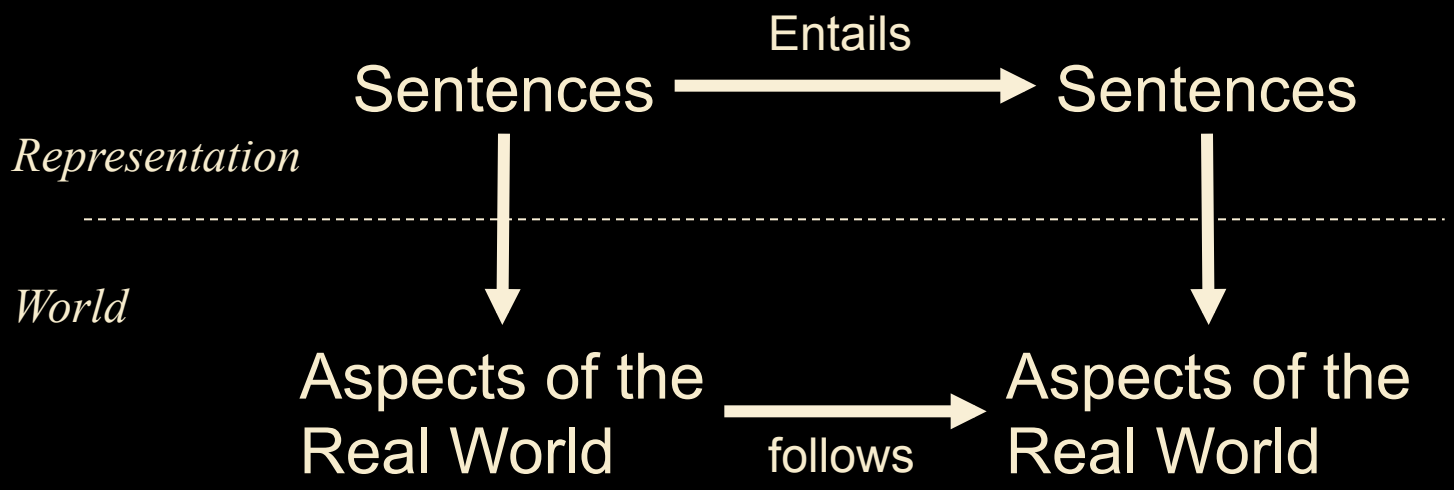


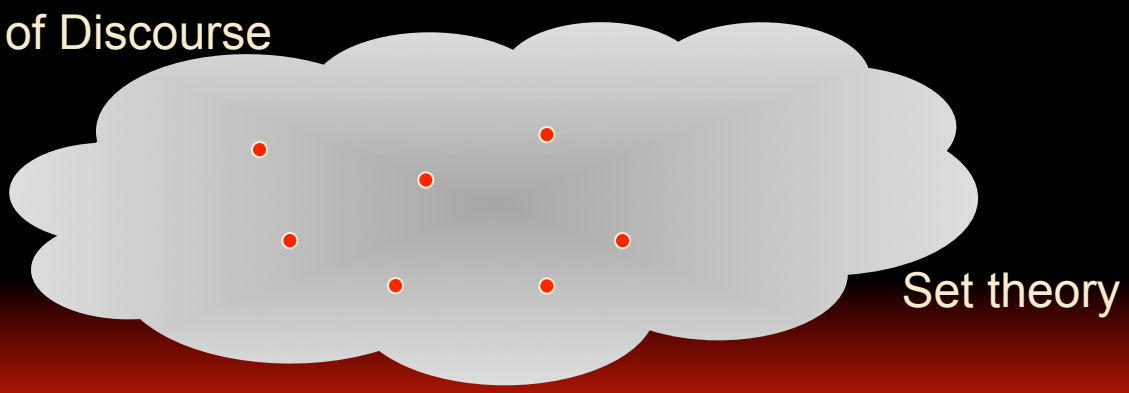
Figure 7.6 Sentences are physical configurations of the agent, and reasoning is a process of constructing new physical configurations from old ones. Logical reasoning should ensure that the new configurations represent aspects of the world that actually follow from the aspects that the old configurations represent.

Formal Semantics

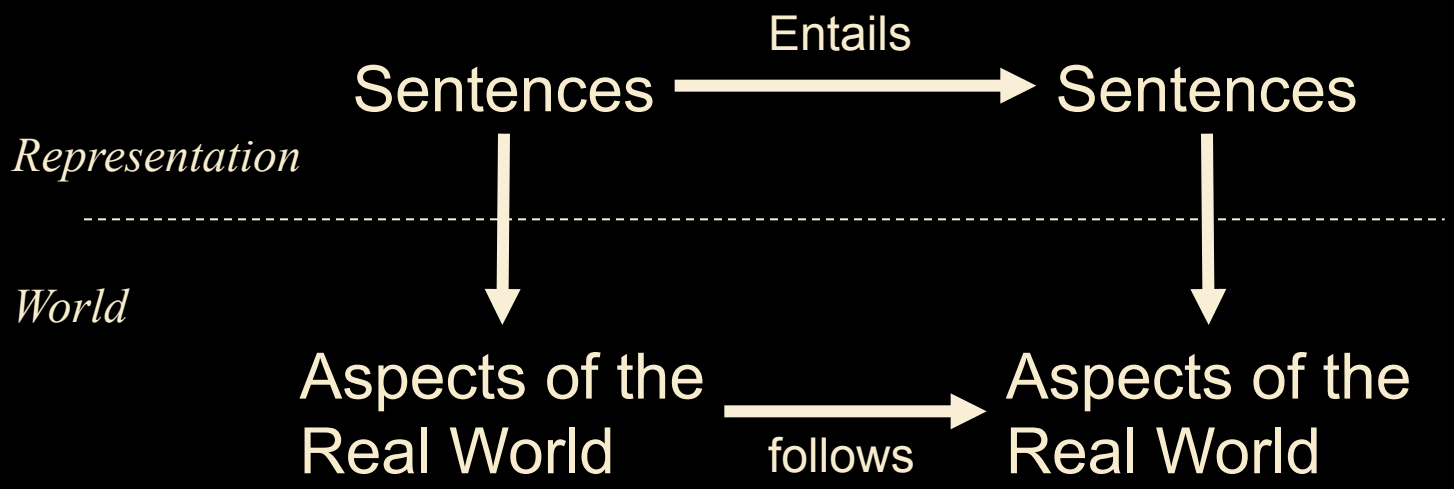


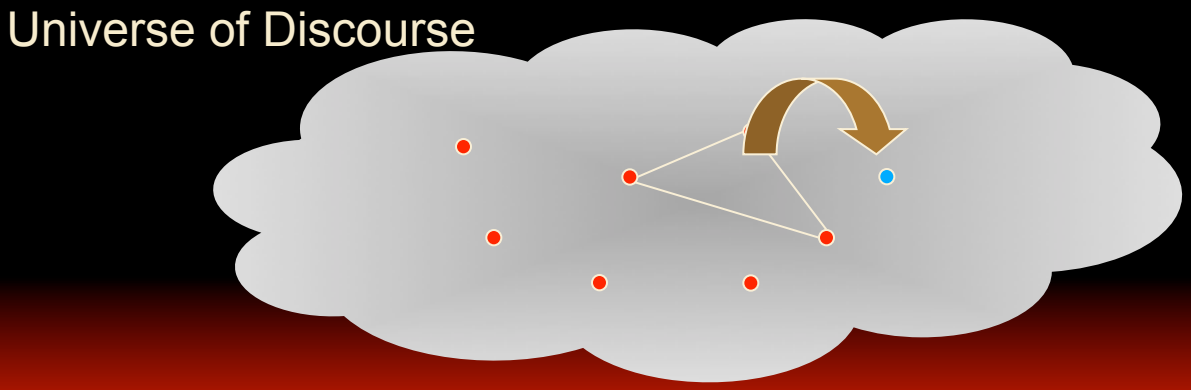
Formal Semantics



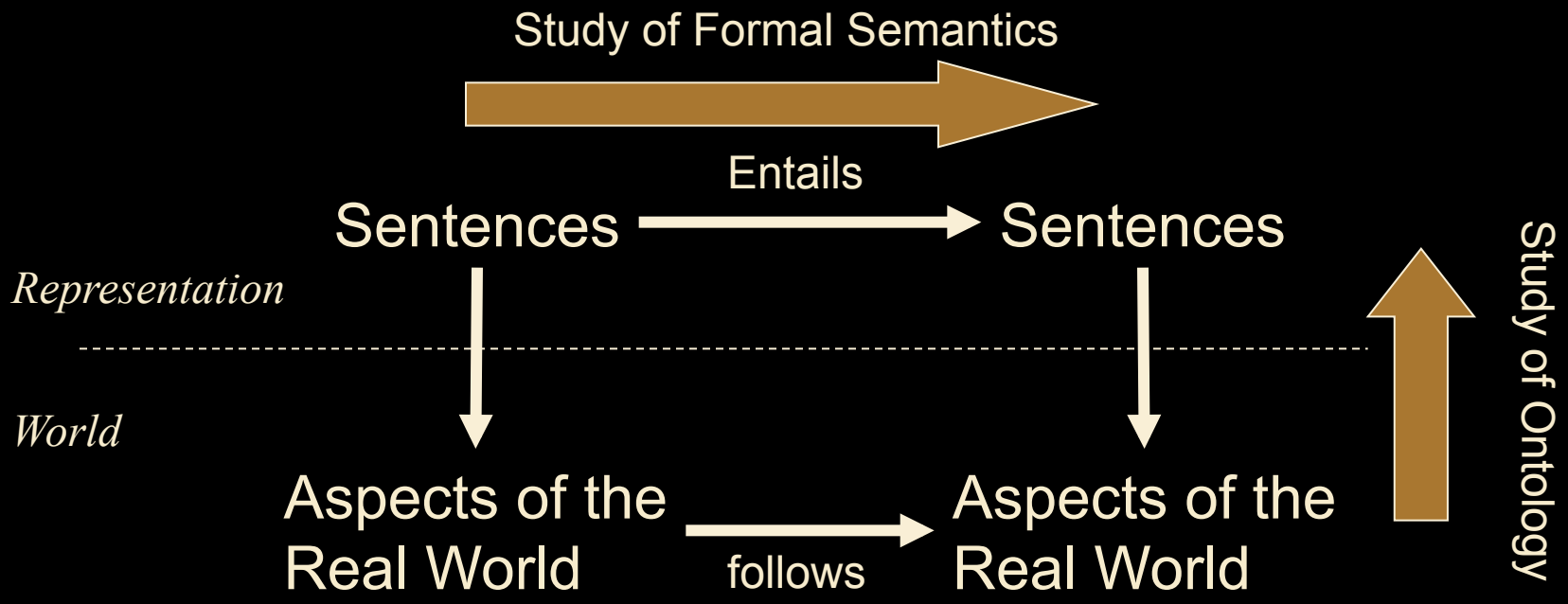


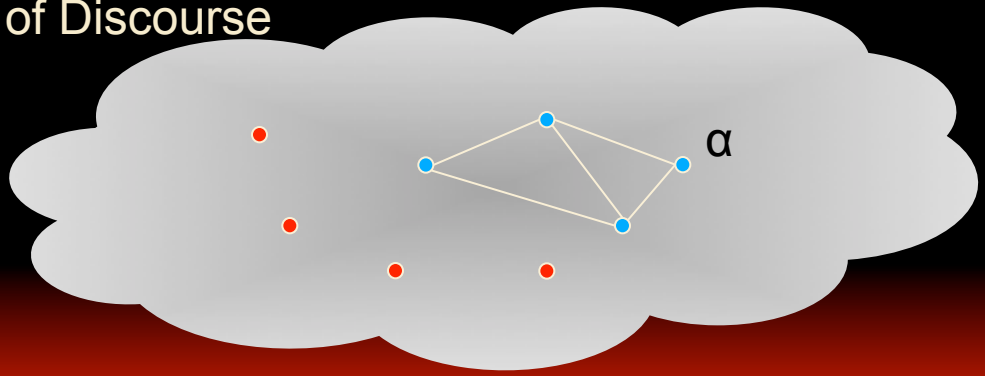
Formal Semantics





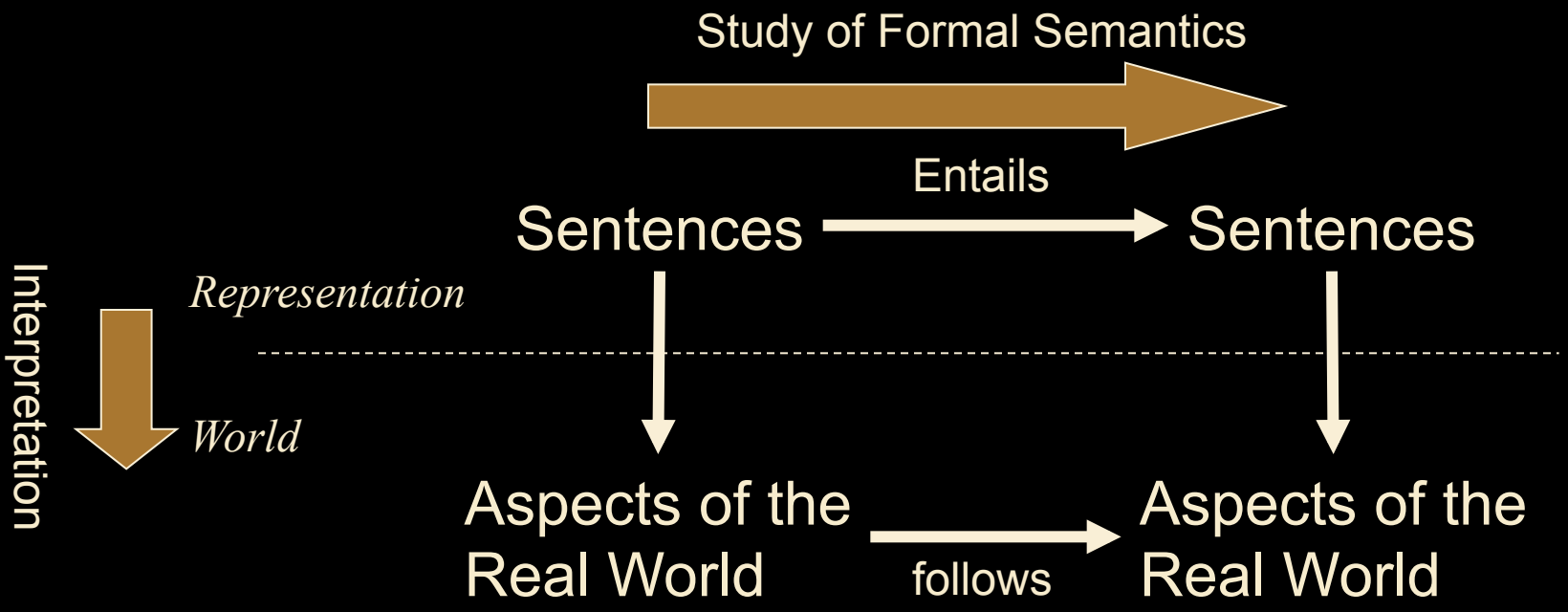
Formal Semantics

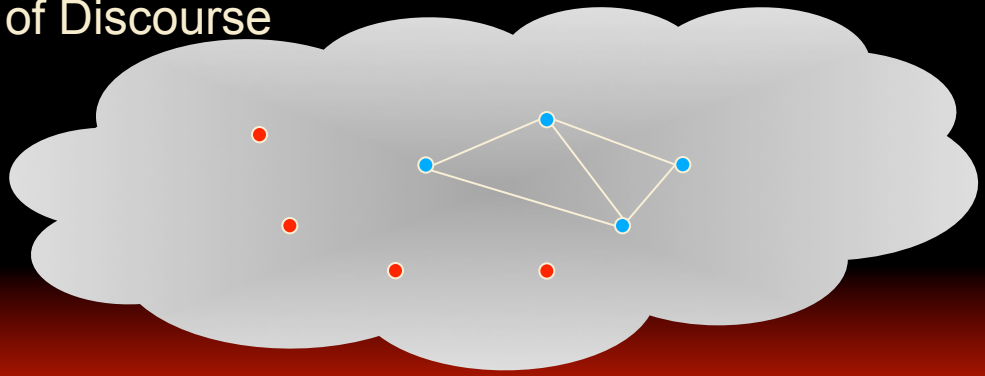




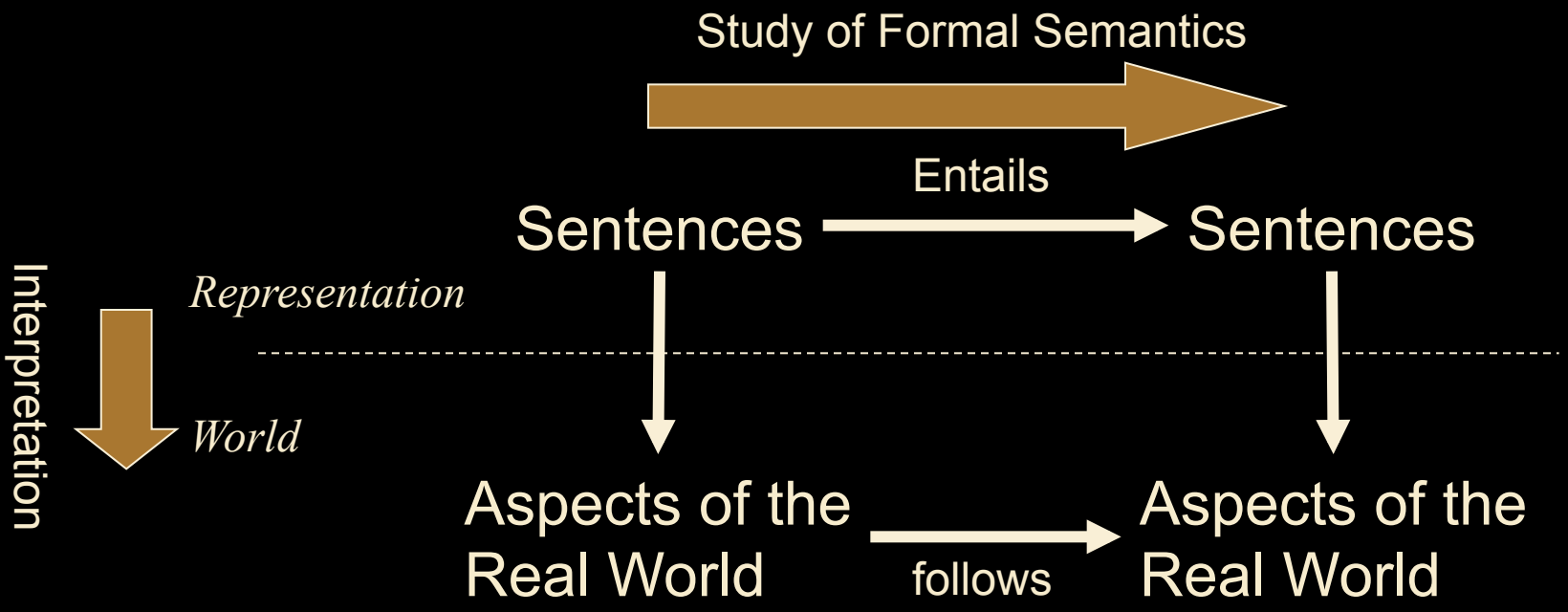
$KB \models \alpha$

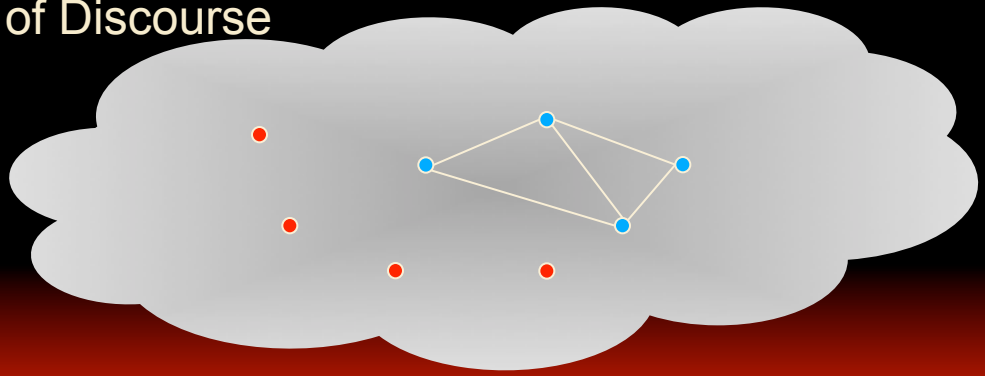
Formal Semantics





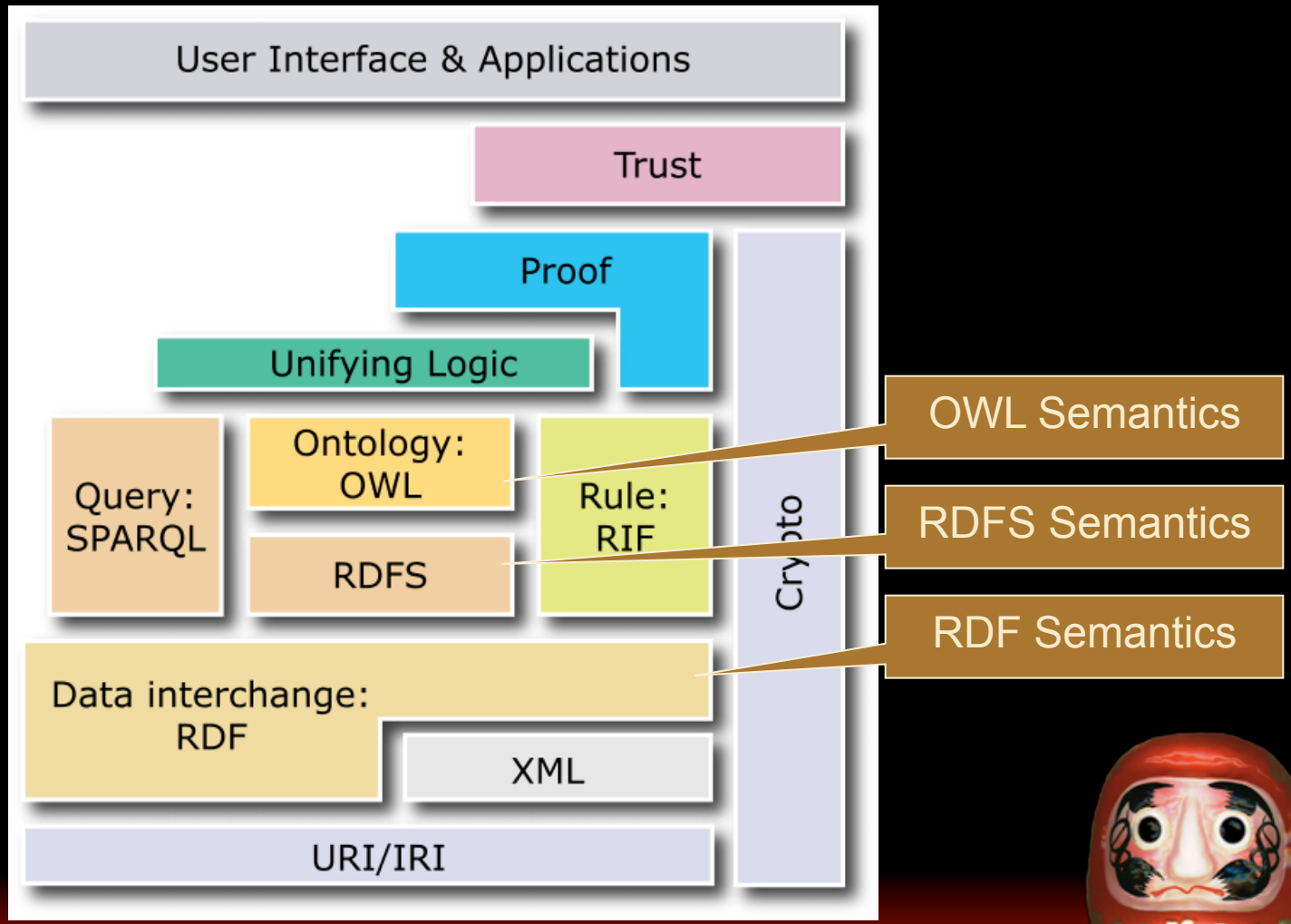
Formal Semantics



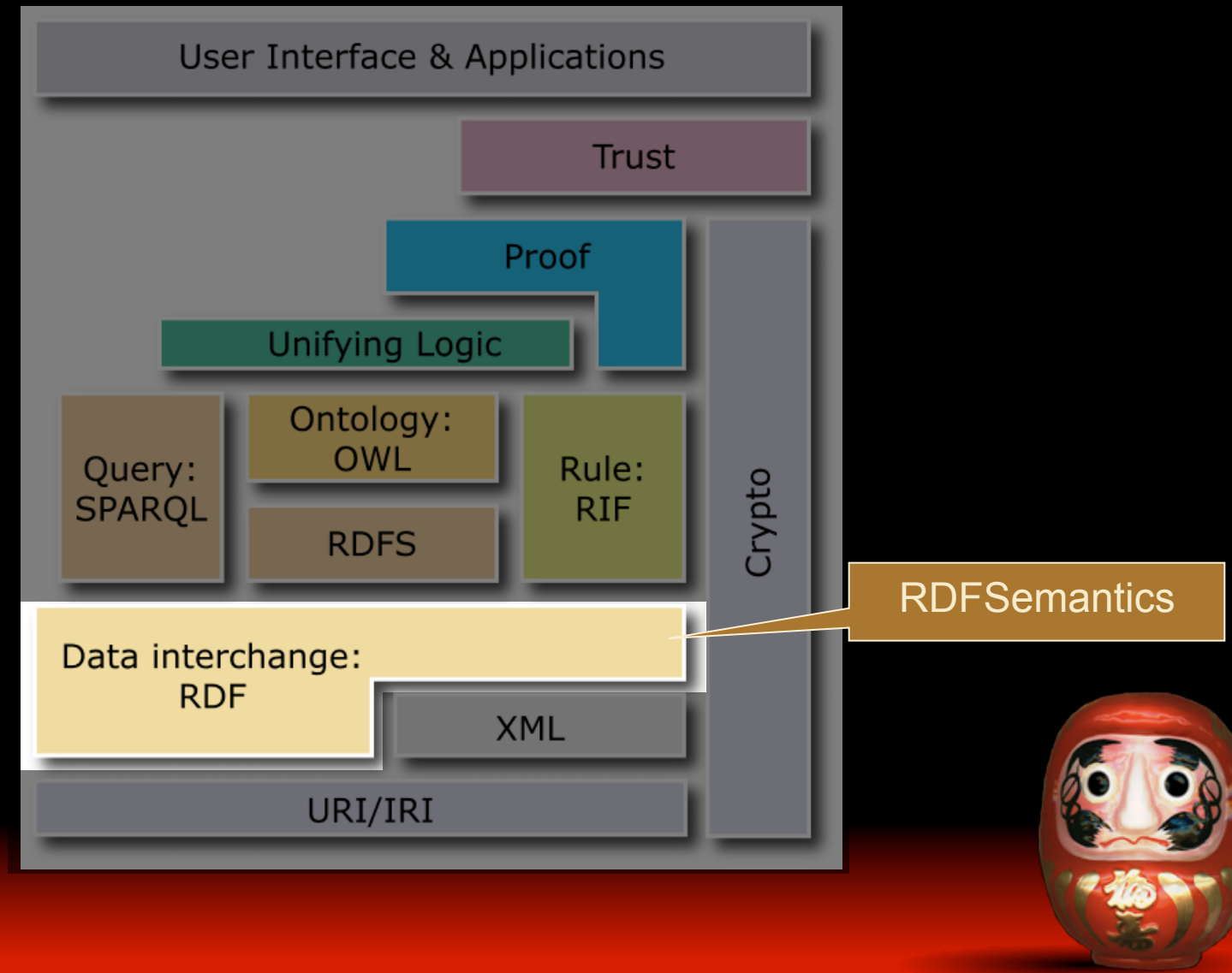


0 ×1 = 0
1 ×1 = 1
0 ＋1 = 1
1 ＋1 = 2

Semantic Web LayerCake

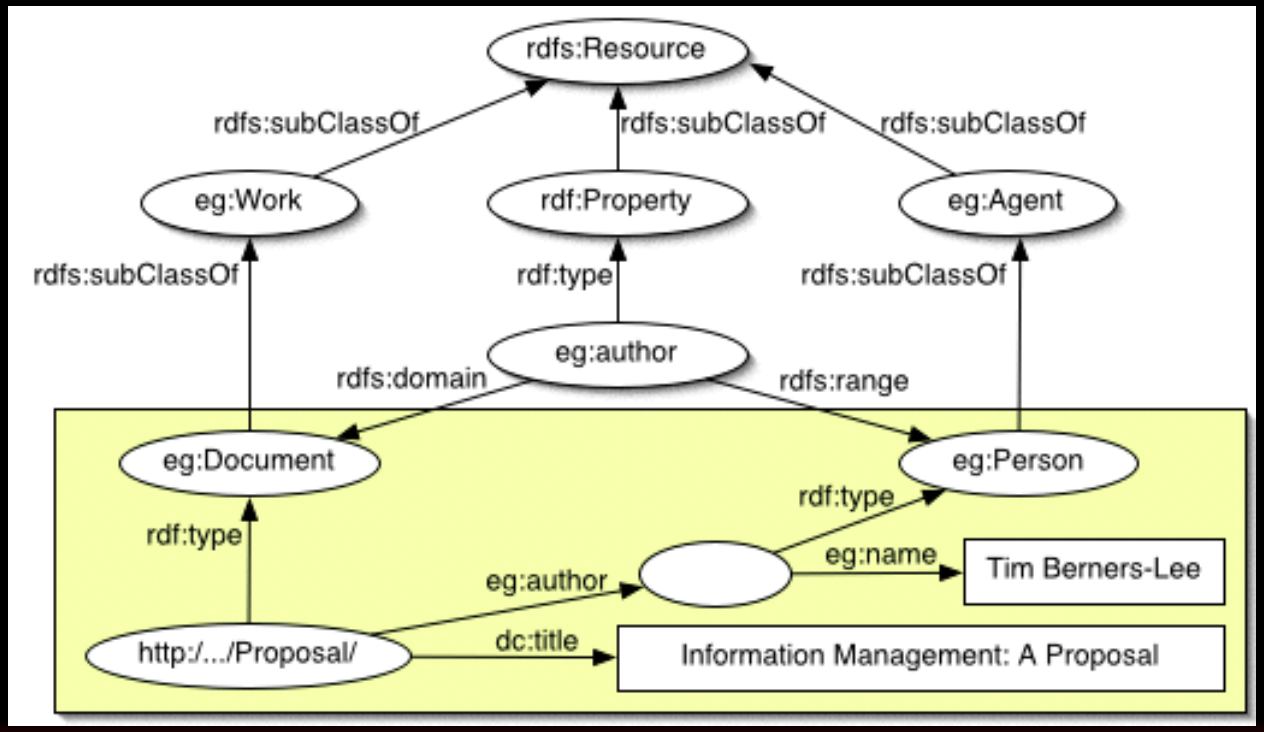


Semantic Web LayerCake



RDF Model Theory

- Directed labeled graph



RDF Model Theory

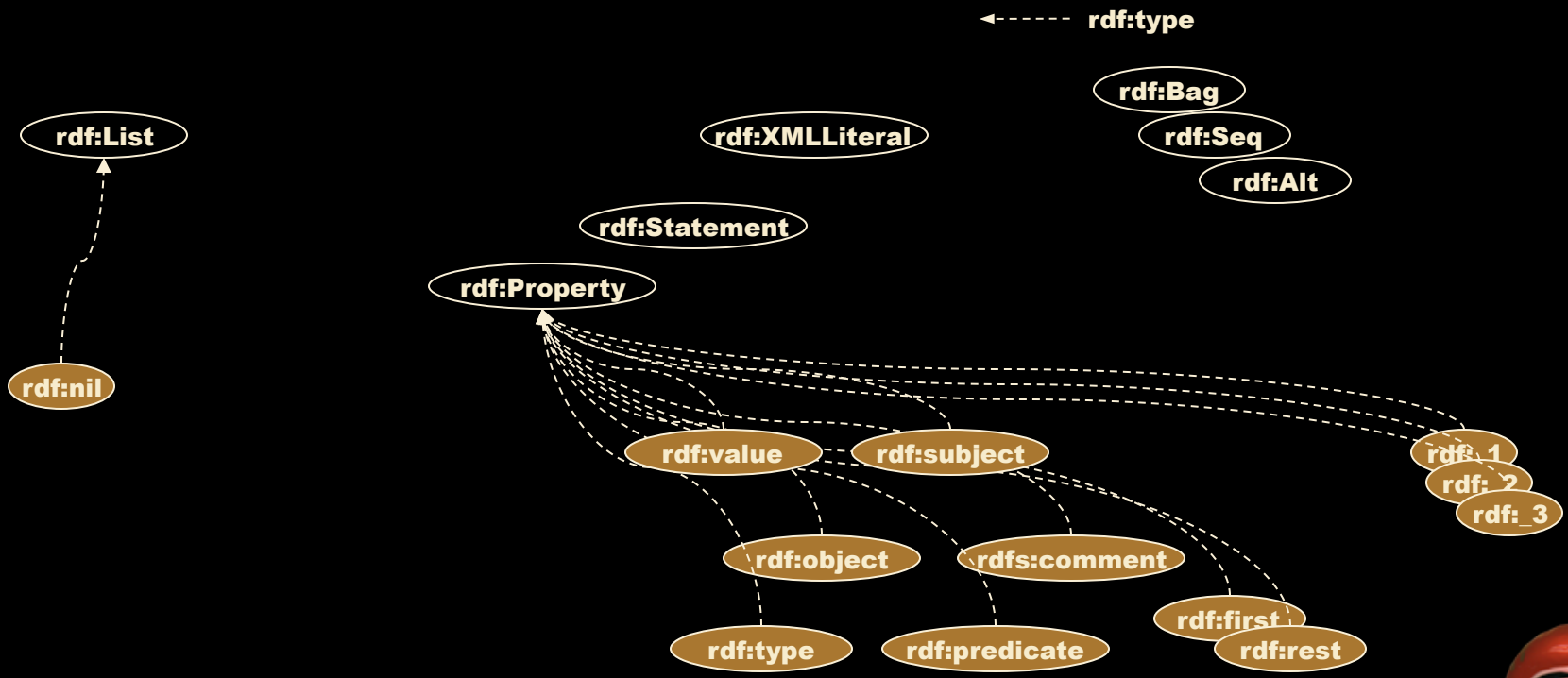
```
http:/…/Proposal/  eg:author  _:x1 .
_:x1  eg:name  “Tim Berners Lee” .
```

set of

```
http:/…/Proposal/  eg:author  _:x2 .
_:x2  eg:name  “Tim Berners Lee” .
```

- Two graphs described by two N-triples documents that shares a set of triples, in which there is no difference except only blank node identifiers, are equal.
- A partial graph of RDF is a partial set of triples of the graph.
- A graph that have no blank node is called ground graph.

RDF Vocabulary



RDF Simple Interpretation

- A non-empty set *IR* of resources, called the domain or universe of $\mathcal{I}$.
- A set *IP*, called the set of properties of $\mathcal{I}$.
- A mapping *IEXT* from *IP* into the powerset of *IR* × *IR* i.e., the set of sets of pairs <*x*,*y*> with *x* and *y* in *IR* .
- A mapping *IS* from URI references in $\mathcal{V}$ into (*IR* ∪ *IP*) .
- A mapping *IL* from typed literals in $\mathcal{V}$ into *IR*.
- A distinguished subset *LV* of *IR*, called the set of literal values, which contains all the plain literals in $\mathcal{V}$.

RDF Simple Interpretation

Interpretation

Representation

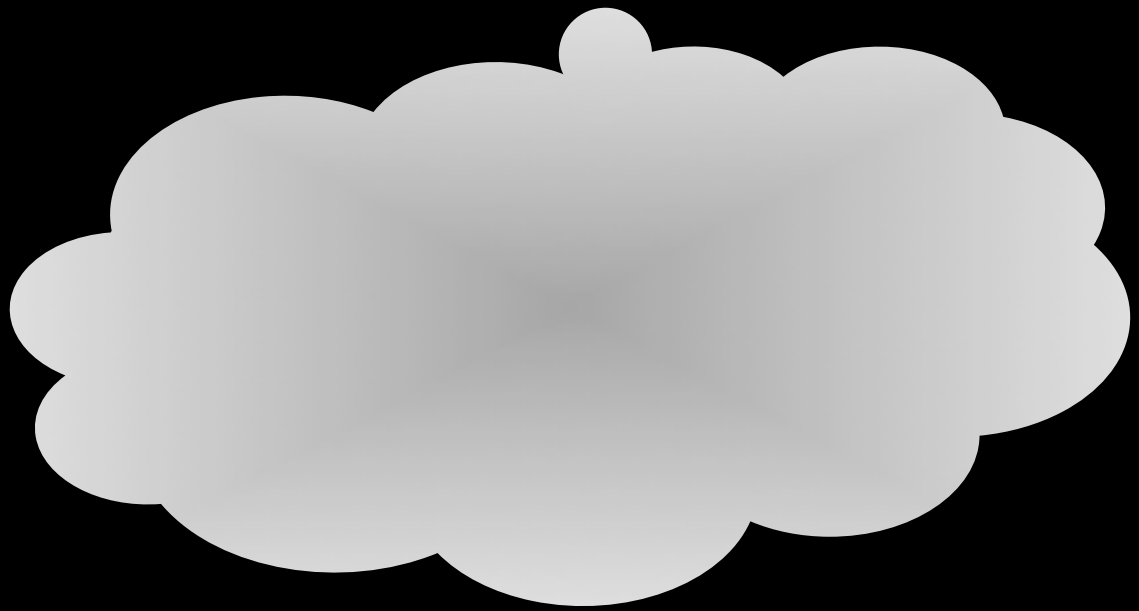
"xx"^^rdf:XMLLiteral

ex:a

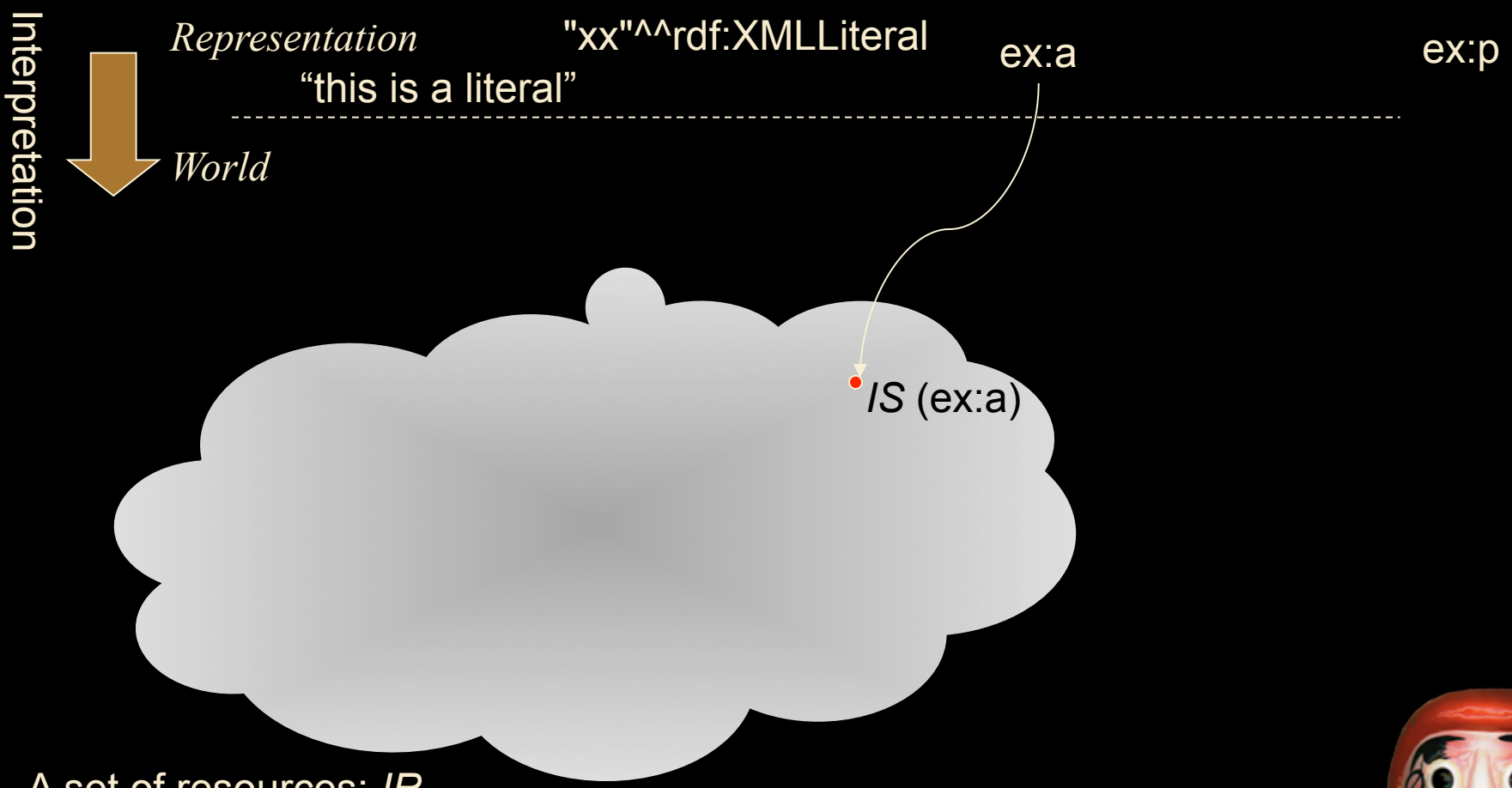
ex:p

"this is a literal"

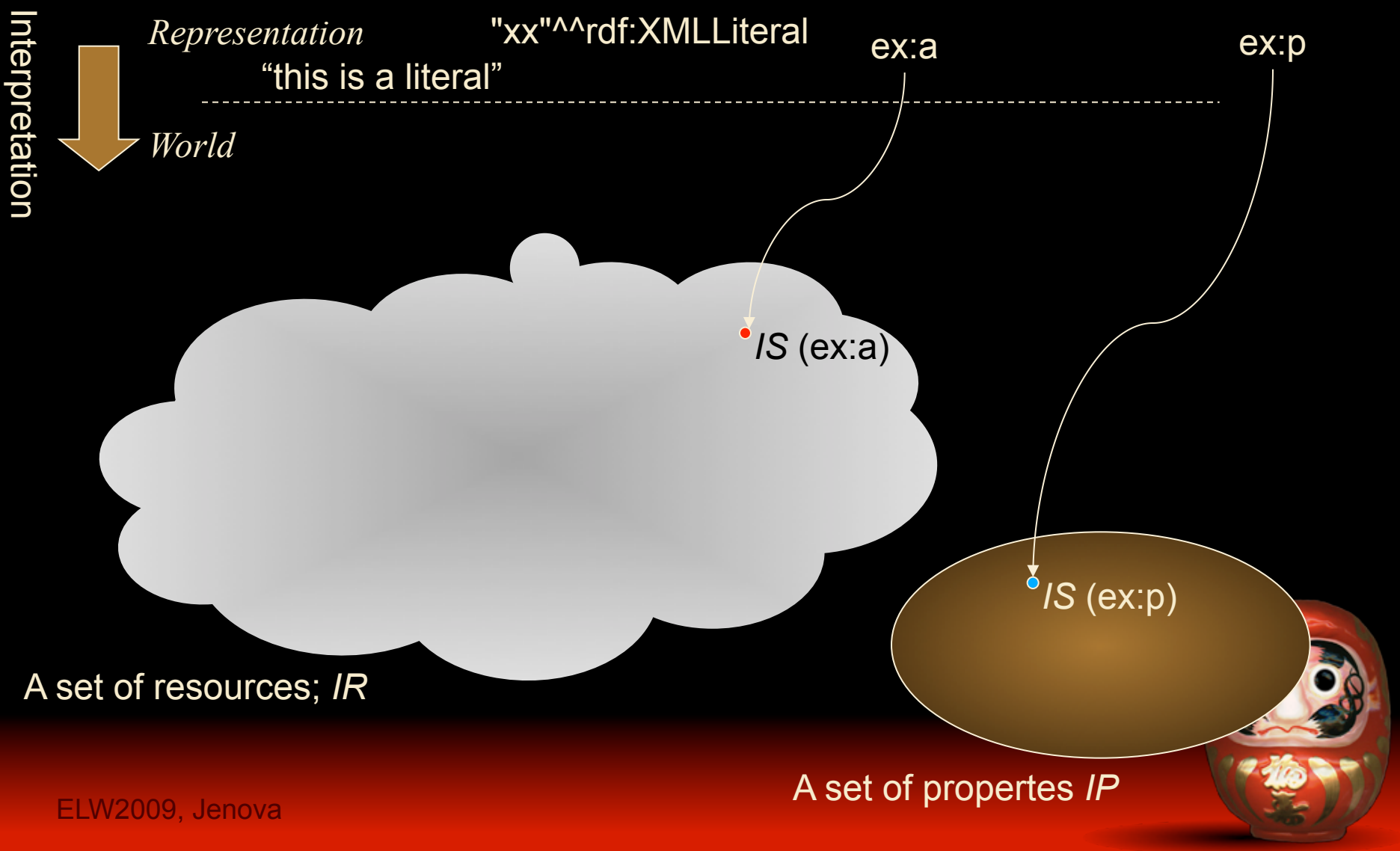
World



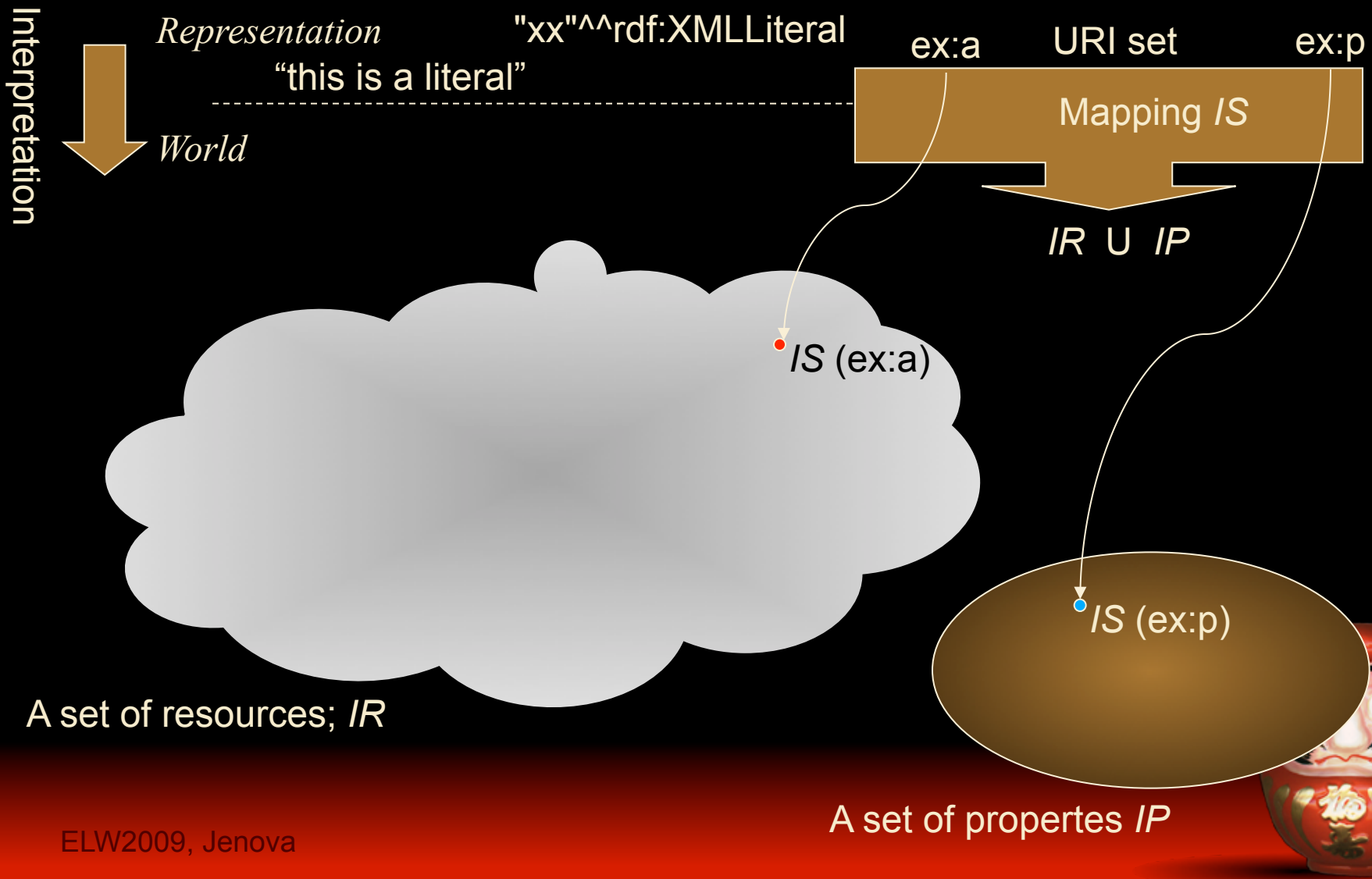
RDF Simple Interpretation



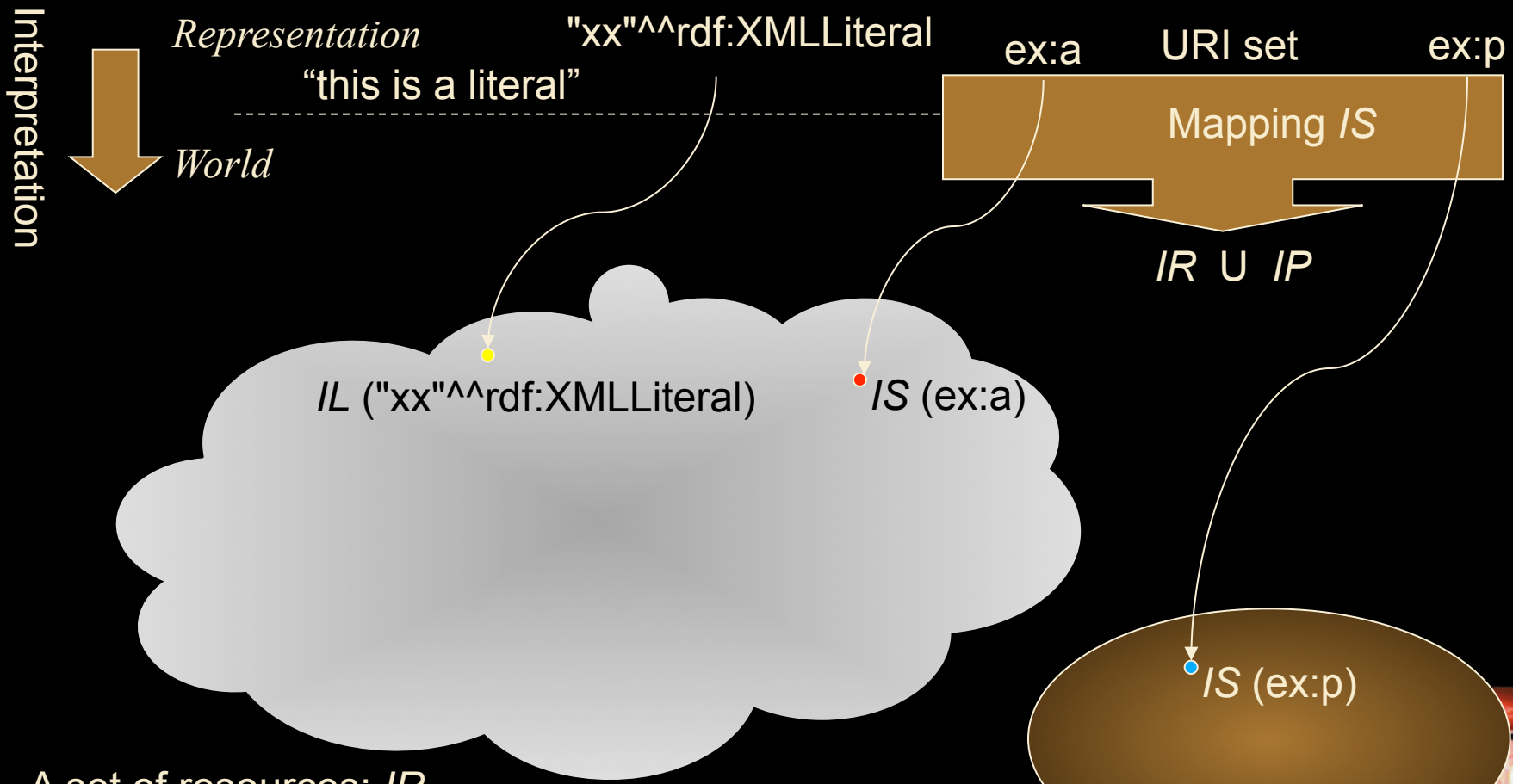
RDF Simple Interpretation



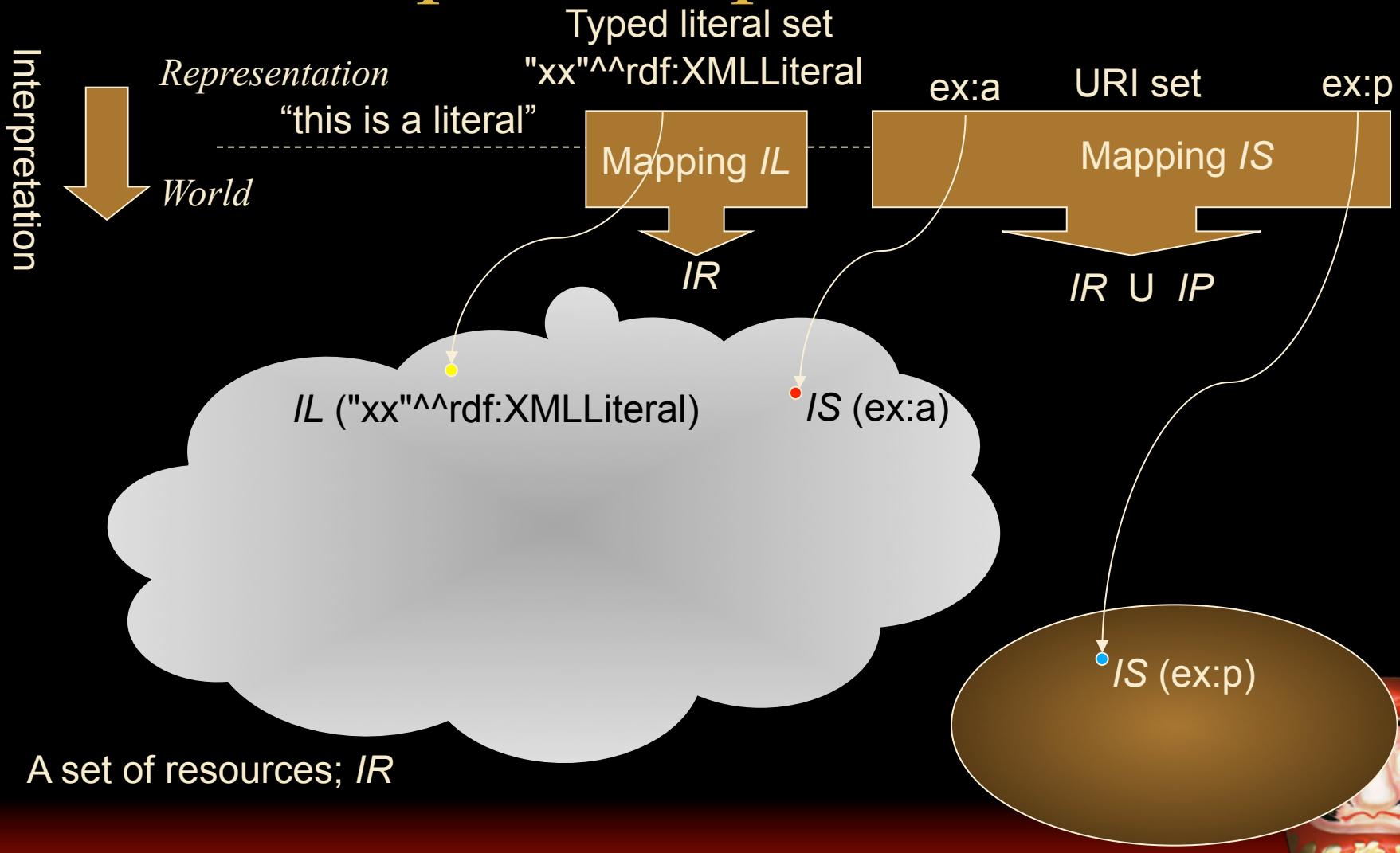
RDF Simple Interpretation



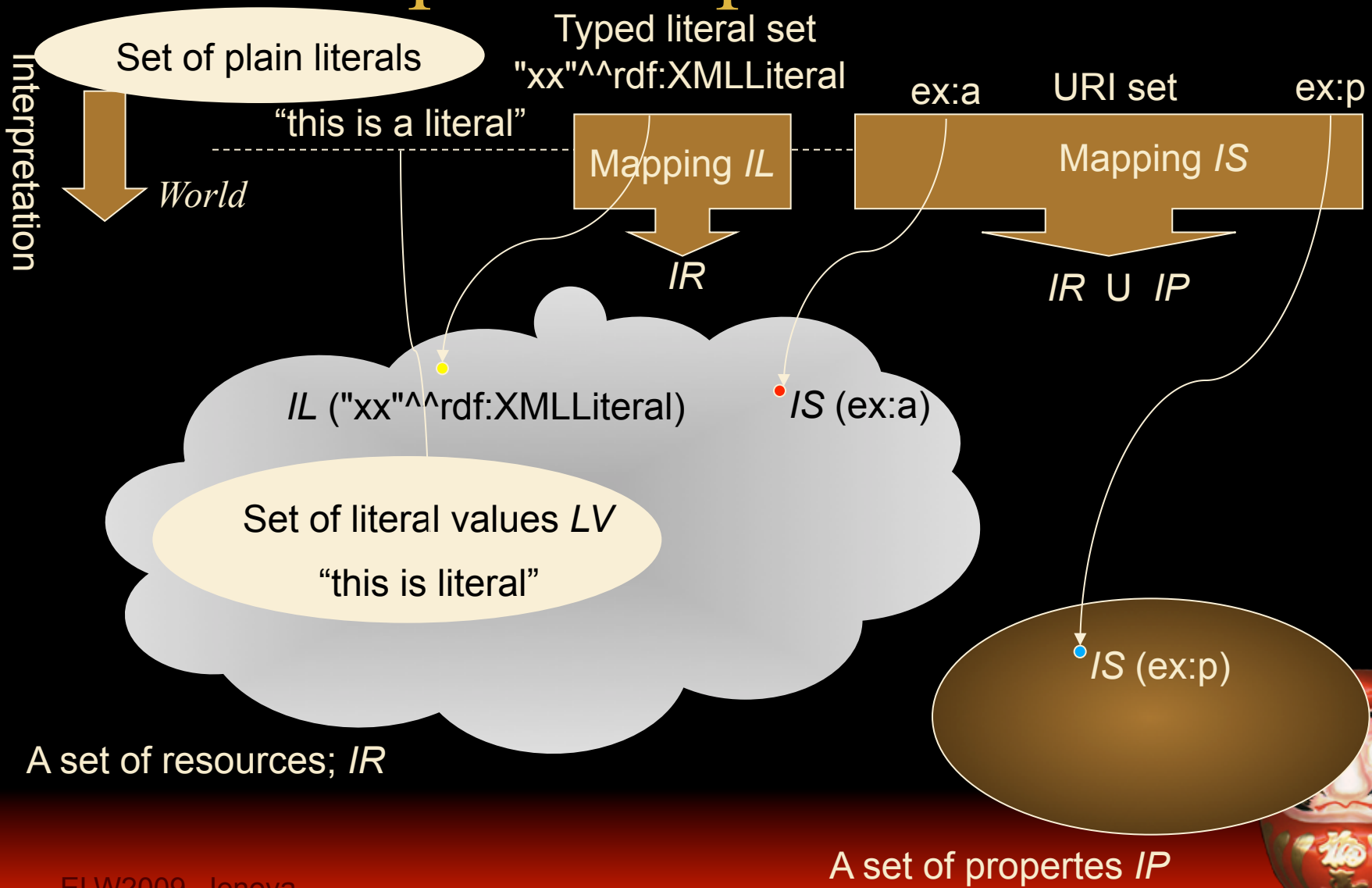
RDF Simple Interpretation



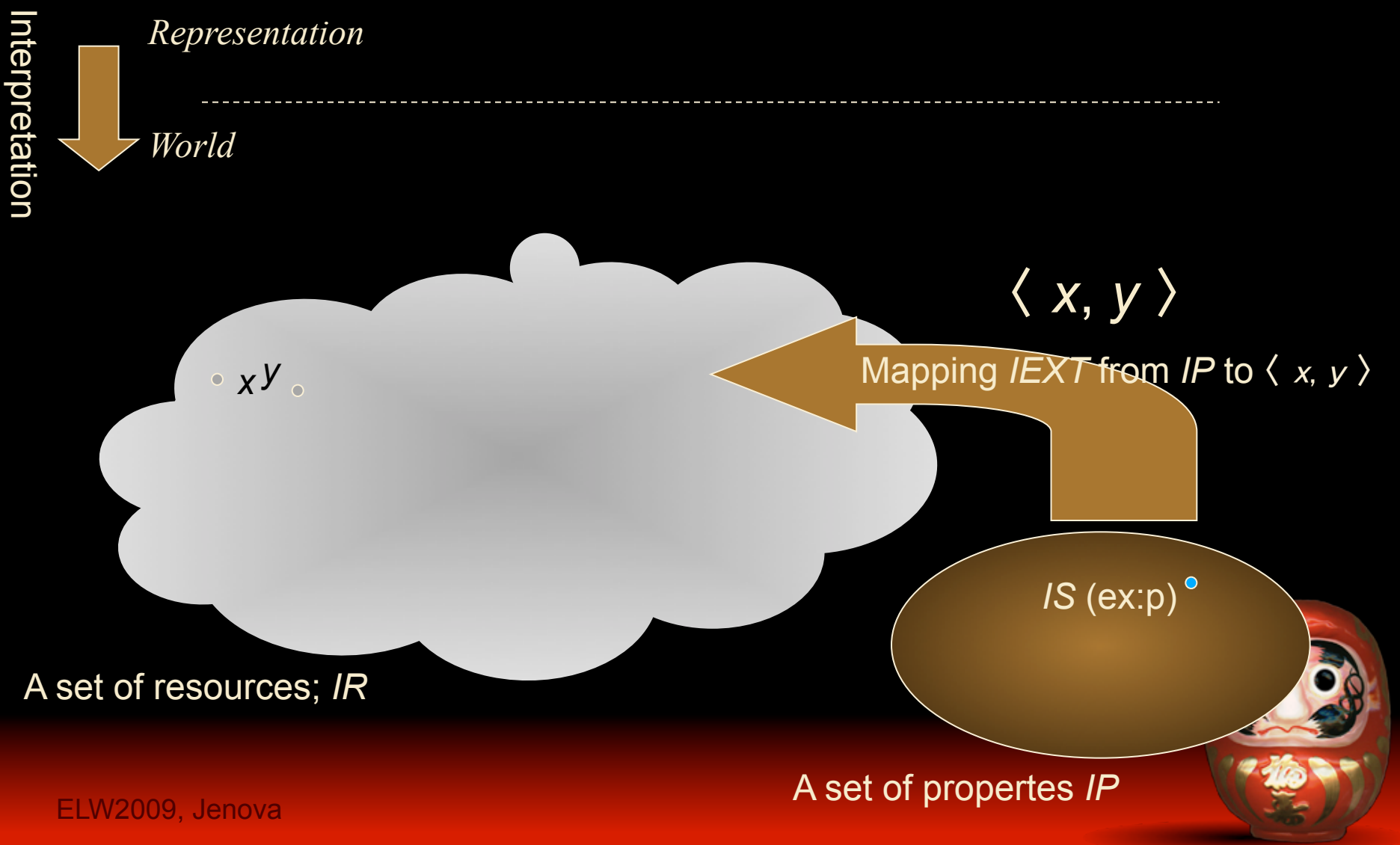
RDF Simple Interpretation



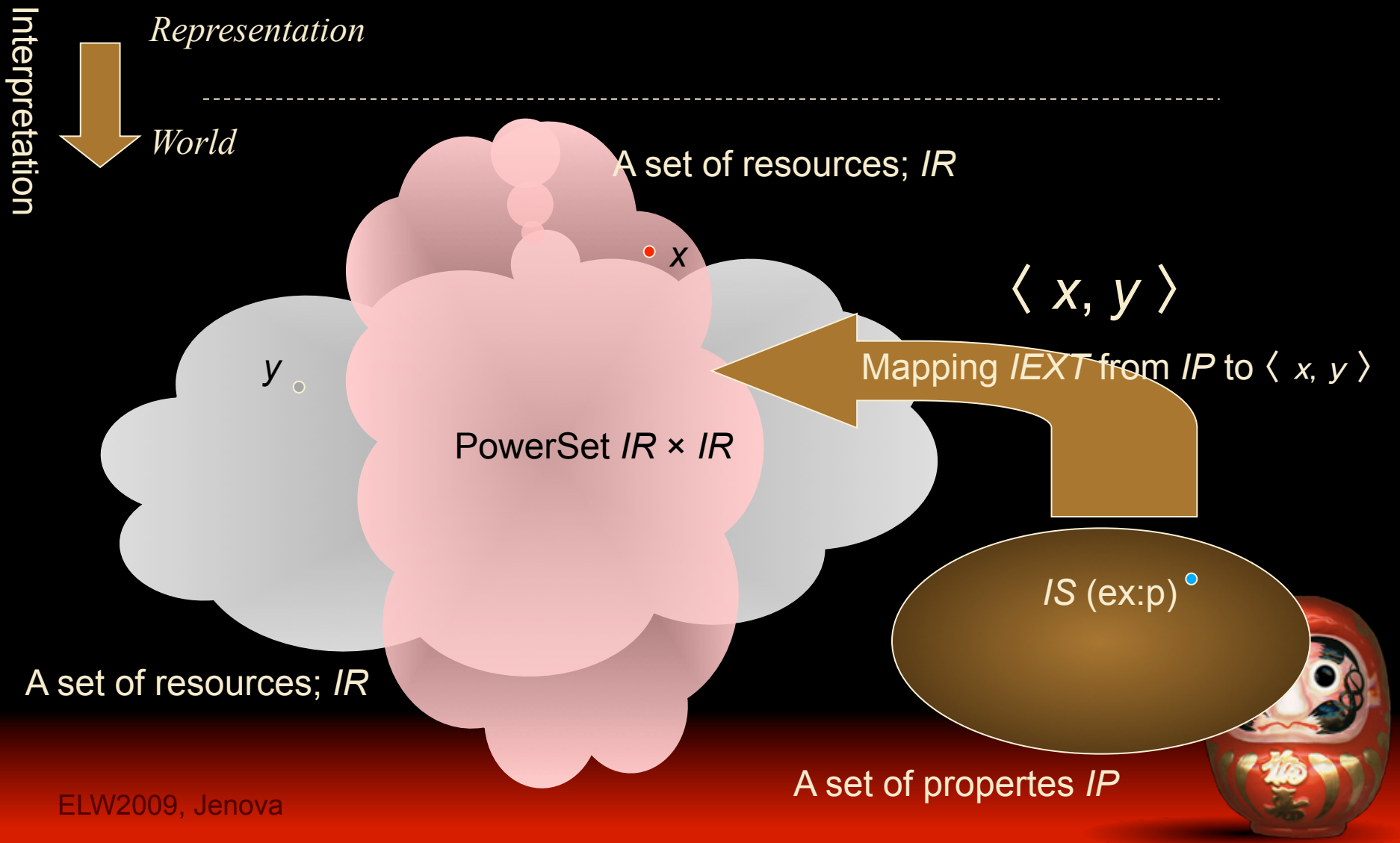
RDF Simple Interpretation



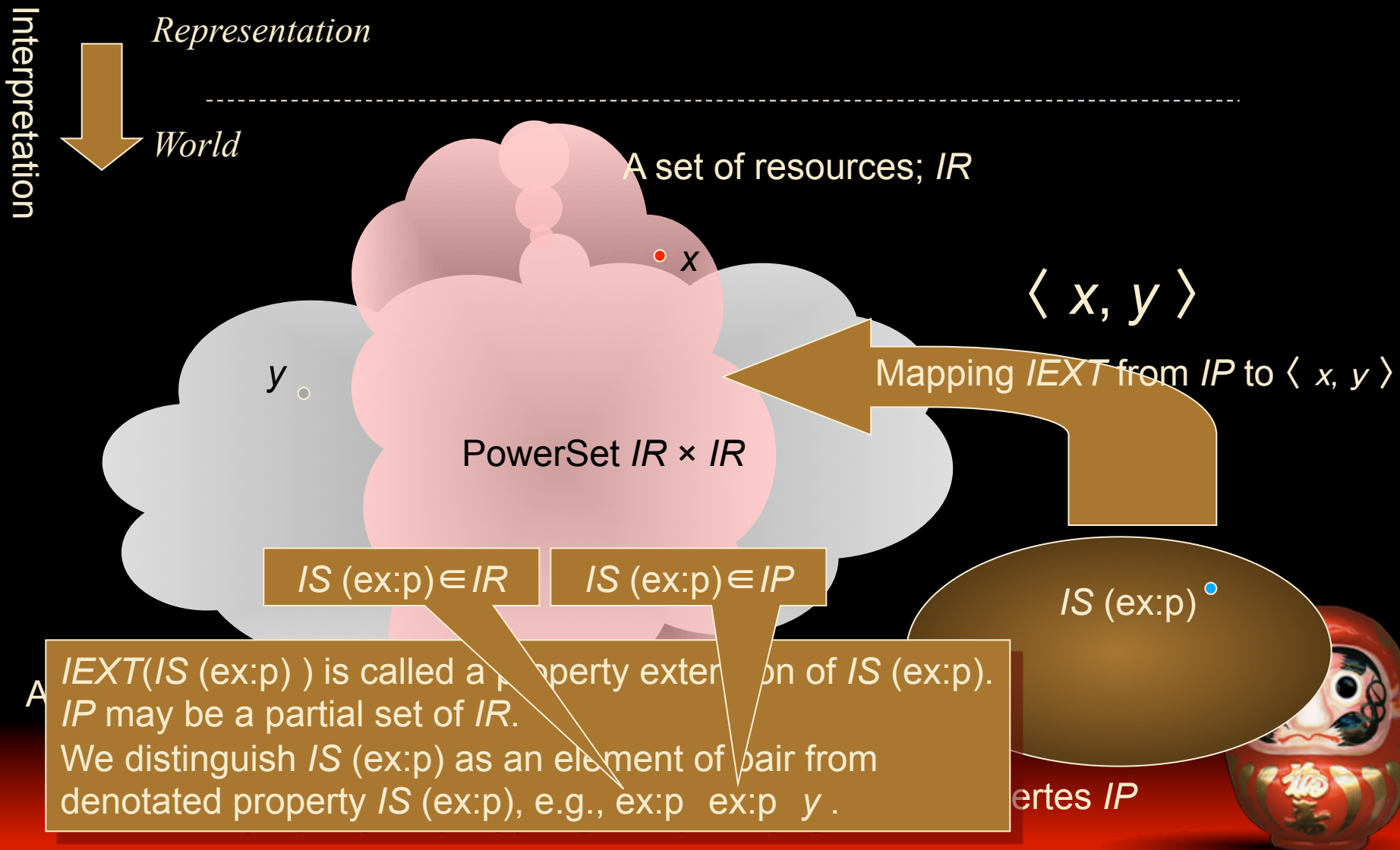
RDF Simple Interpretation



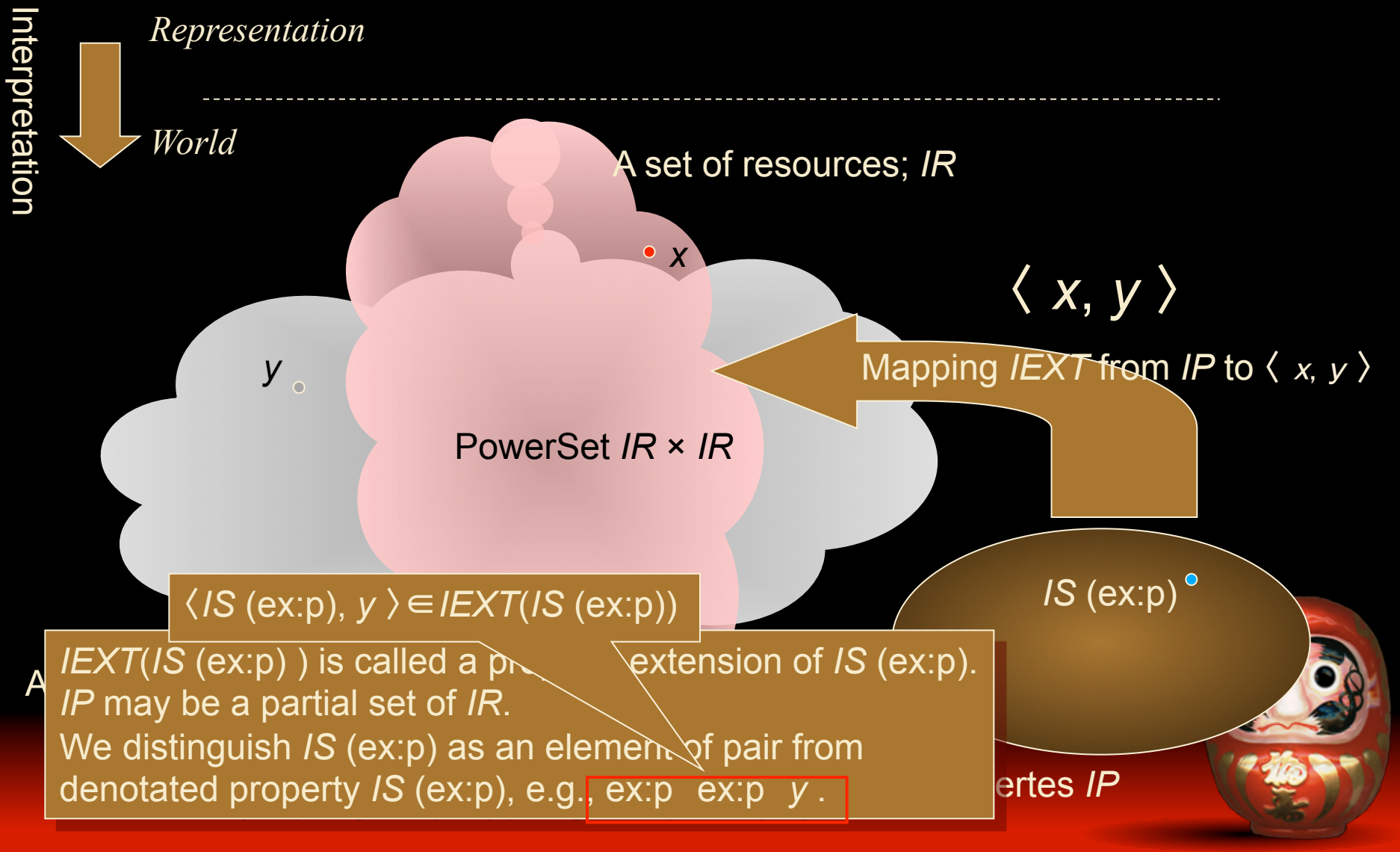
RDF Simple Interpretation



RDF Simple Interpretation



RDF Simple Interpretation



Semantic Conditions for Ground Graphs

- If E is a plain literal "aaa" in $\mathcal{V}$, then $\mathcal{I}(E) =$ aaa .
- If E is a plain literal "aaa"@ttt in $\mathcal{V}$, then $\mathcal{I}(E) =$ ⟨aaa, ttt⟩ .
- If E is a typed literal in $\mathcal{V}$, then $\mathcal{I}(E) = IL(E)$.
- If E is a URI reference in $\mathcal{V}$, then $\mathcal{I}(E) = IS(E)$.
- If E is a ground triple $s\ p\ o.$, then $\mathcal{I}(E) =$ true if s, p, and o are in $\mathcal{V}$, $\mathcal{I}(p)$ is in IP, and ⟨$\mathcal{I}(s)$, $\mathcal{I}(o)$⟩ is in $IEXT(\mathcal{I}(p))$, otherwise $\mathcal{I}(E) =$ false .
- If E is a ground RDF graph, then $\mathcal{I}(E) =$ false if $\mathcal{I}(E') =$ false for some triple E' in E, otherwise $\mathcal{I}(E) =$ true .

"soccur@en-US" = "football"@en

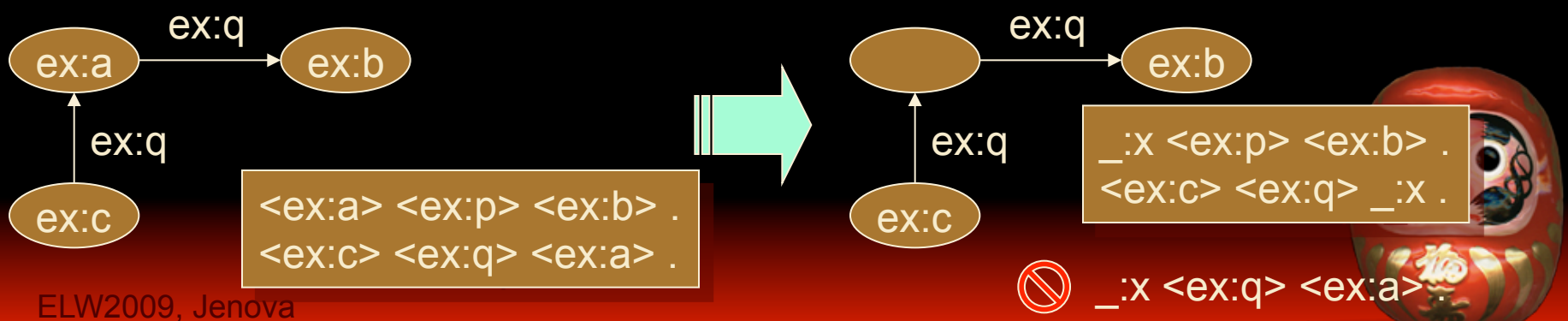
Simple Entailment between RDF Graphs

$S \models E$

- $\mathcal{I}$ satisfies E if $\mathcal{I}(E)$ = true, and a set S of RDF graphs (simply) entails a graph E if every interpretation which satisfies every member of S also satisfies E.
- If A entails B, then any interpretation that makes A true also makes B true, so that an assertion of A already contains the same "meaning" as an assertion of B.
- Through the notions of satisfaction, entailment and validity, formal semantics gives a rigorous definition to a notion of "meaning".
- Any process which constructs a graph E from some other graph(s), S is said to be (simply) valid if S entails E in every case, otherwise invalid.

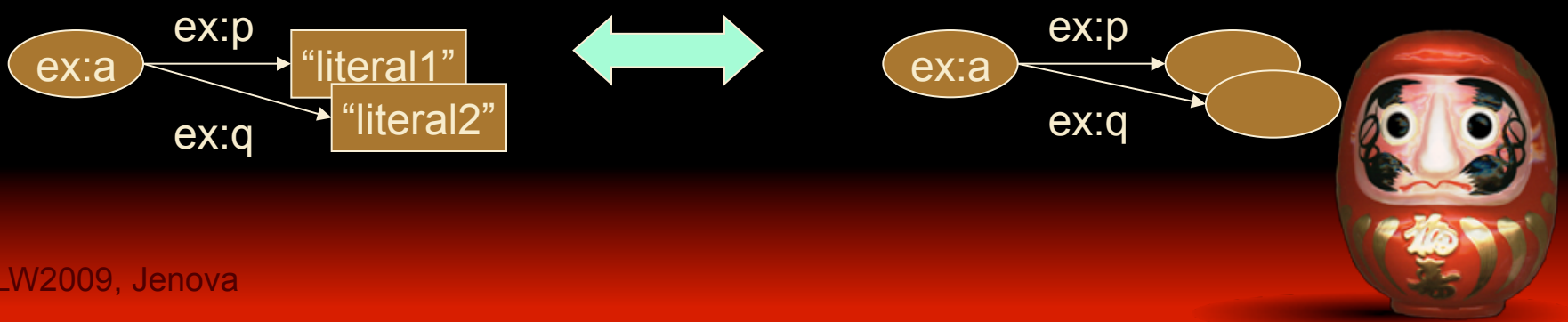
Simple Entailment Rules of RDF

Rule Name	If E contains:	then add:
se1	uuu aaa xxx .	uuu aaa _:nnn . where _:nnn identifies a blank node allocated to xxx by rule se1 or se2.
se2	uuu aaa xxx .	_:nnn aaa xxx . where _:nnn identifies a blank node allocated to uuu by rule se1 or se2.



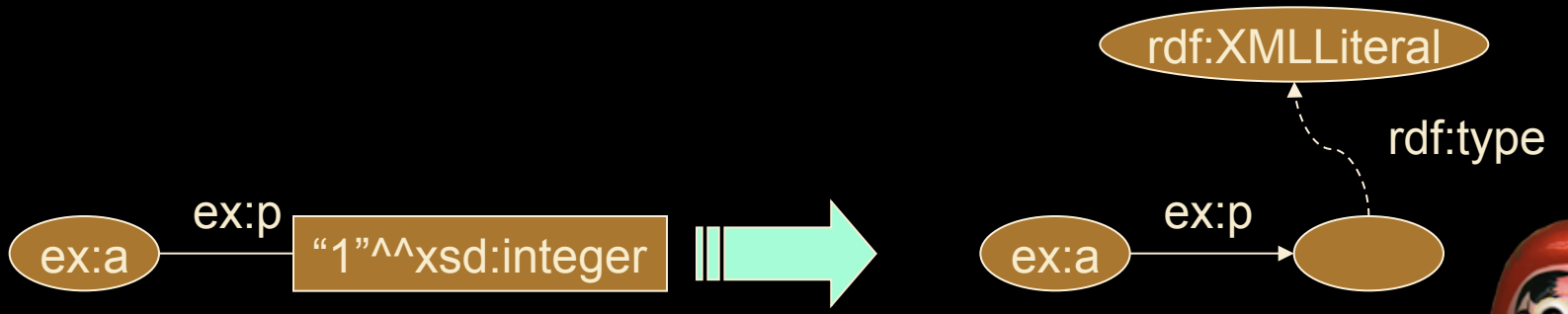
Literal Generalization/Instantiation Rule

Rule Name	If E contains:	then add:
lg	uuu aaa lll .	uuu aaa _:nnn . where _:nnn identifies a blank node allocated to the literal lll by this rule.
gl	uuu aaa _:nnn . where _:nnn identifies a blank node allocated to the literal lll by rule lg.	uuu aaa lll .



RDF entailment rules

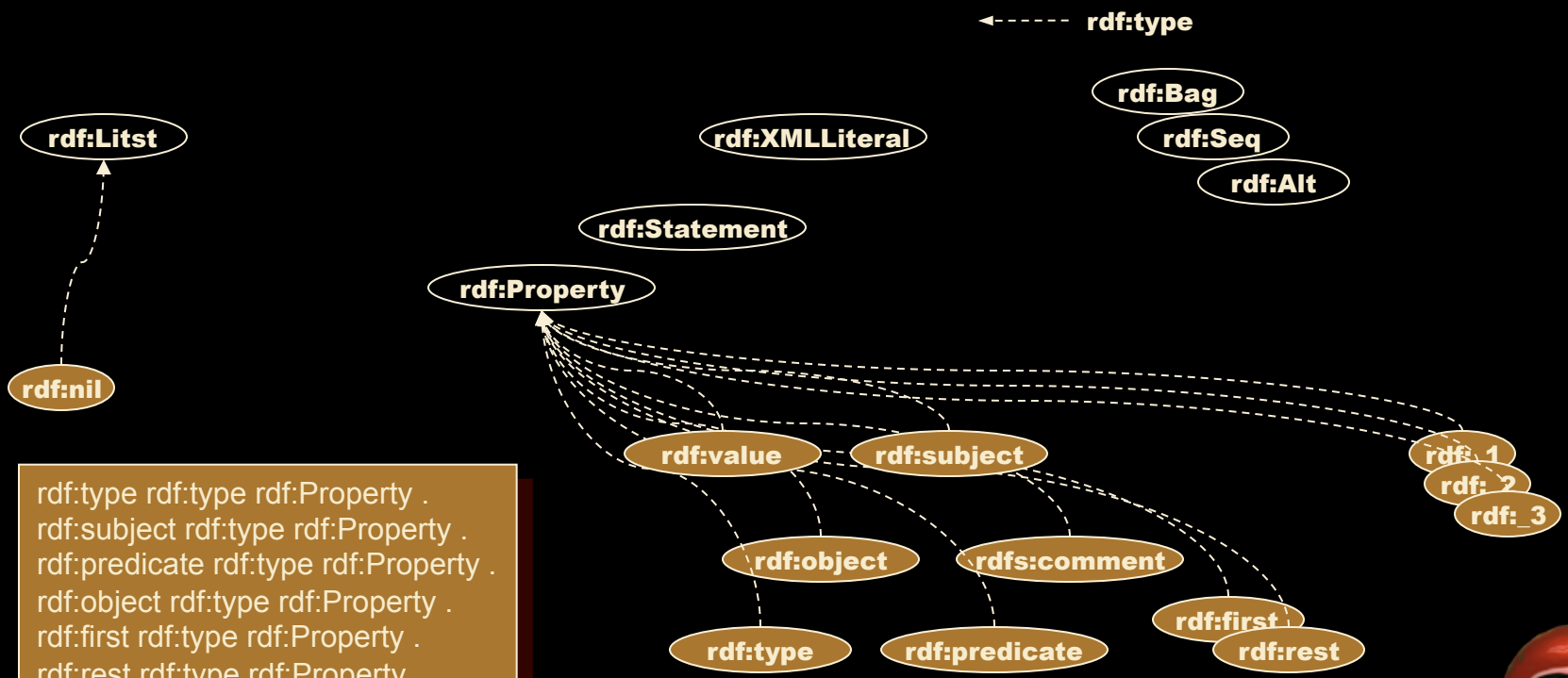
Rule Name	If E contains:	then add:
rdf1	uuu aaa yyy .	aaa rdf:type rdf:Property .
rdf2	uuu aaa lll . where lll is a well-typed XML literal .	_:nnn rdf:type rdf:XMLLiteral . where _:nnn identifies a blank node allocated to lll by rule lg.



RDF Semantic Condition

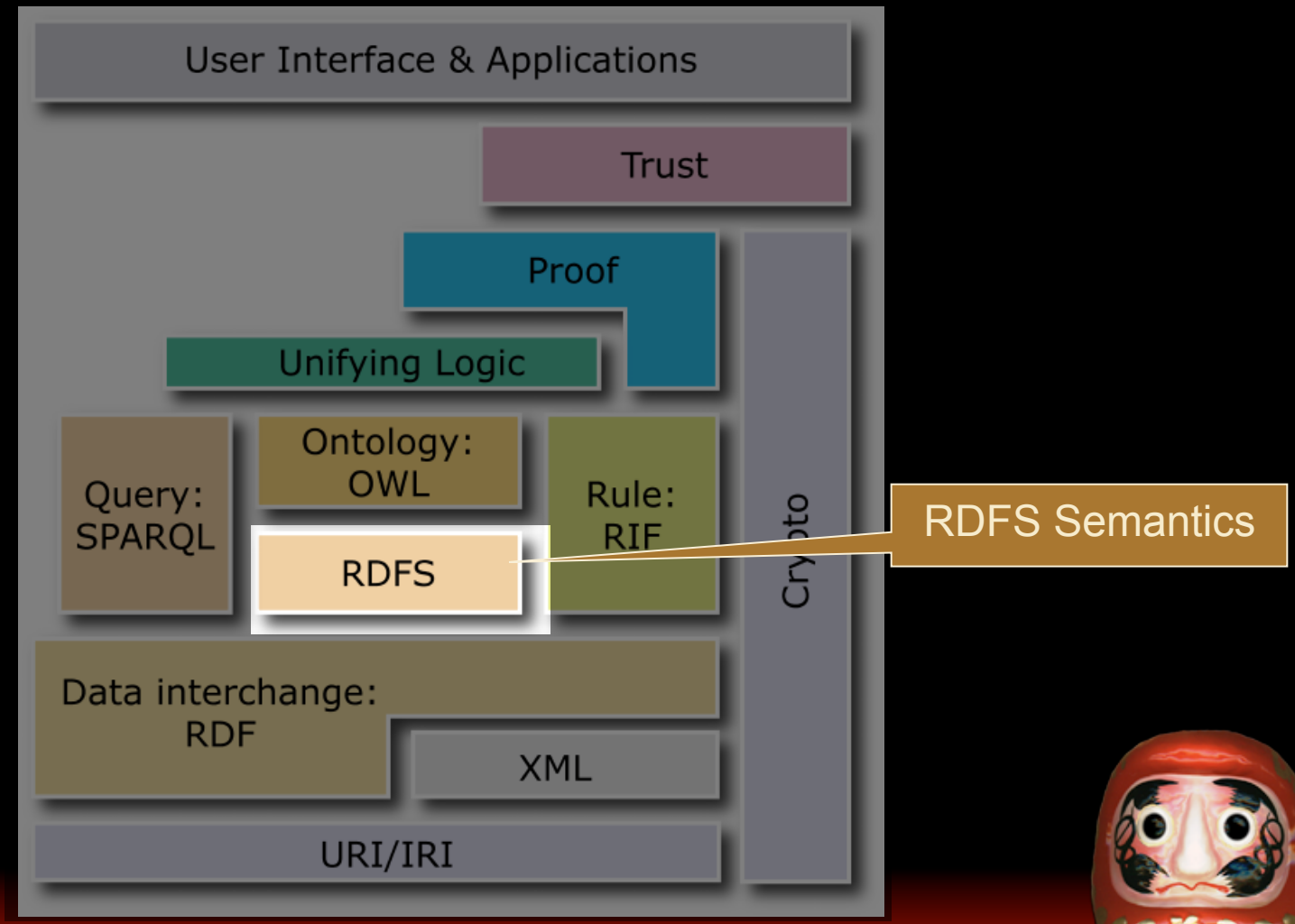
- $x \in IP$, *iff* $\langle x, I(\text{rdf:Property})\rangle \in IEXT(I(\text{rdf:type}))$
  - ✓ xxx rdf:type rdf:Property .
- $IEXT(I(\text{rdf:type})) \wedge$ xxx is a well-typed XML literal $\Rightarrow$ $IL(\text{“xxx”}\text{^^rdf:XMLLiteral})$ is an XML value of xxx, $IL(\text{“xxx”}\text{^^rdf:XMLLiteral}) \in LV$, $\langle IL(\text{“xxx”}\text{^^rdf:XMLLiteral}), I(\text{rdf:XMLLiteral})\rangle \in IEXT(I(\text{rdf:type}))$
  - ✓ *“xxx”*^^rdf:XMLLiteral rdf:type rdf:XMLLiteral .
- $IEXT(I(\text{rdf:type})) \wedge$ xxx is ill-typed XML literal $\Rightarrow$ $IL(\text{“xxx”}\text{^^rdf:XMLLiteral}) \notin LV$, $\langle IL(\text{“xxx”}\text{^^rdf:XMLLiteral}), I(\text{rdf:XMLLiteral})\rangle \notin IEXT(I(\text{rdf:type}))$

RDF Axiomatic Triples

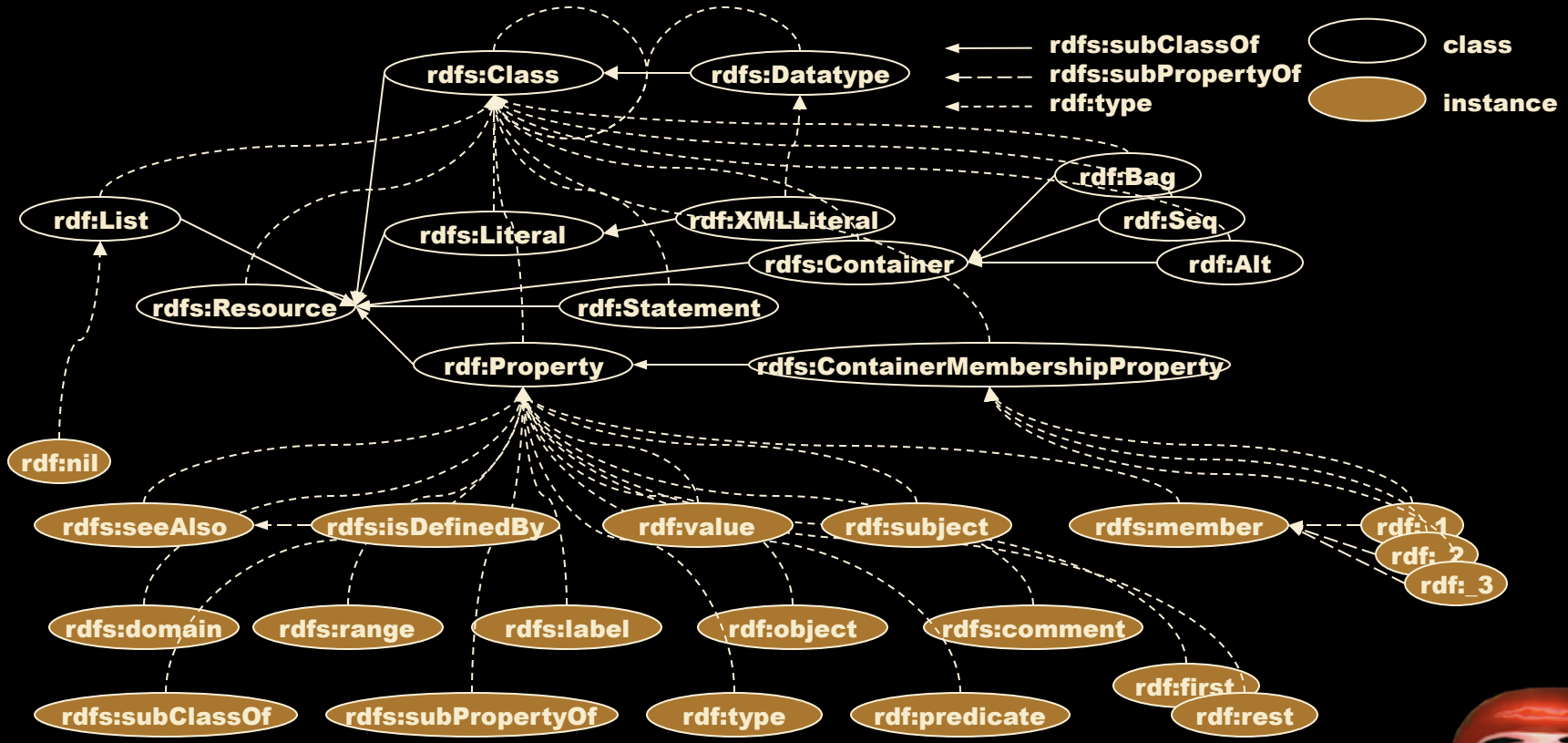


rdf:type rdf:type rdf:Property .
rdf:subject rdf:type rdf:Property .
rdf:predicate rdf:type rdf:Property .
rdf:object rdf:type rdf:Property .
rdf:first rdf:type rdf:Property .
rdf:rest rdf:type rdf:Property .
rdf:value rdf:type rdf:Property .
rdf:_1 rdf:type rdf:Property .
rdf:_2 rdf:type rdf:Property .
...
rdf:nil rdf:type rdf:List .

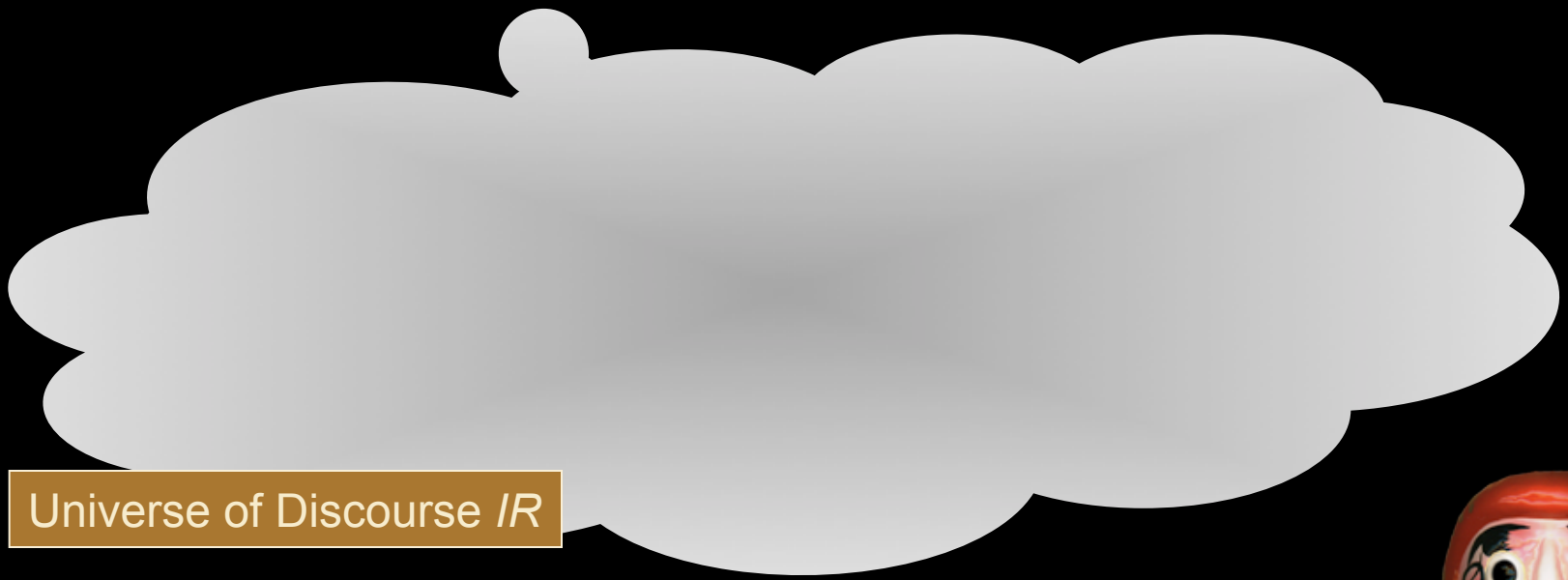
Semantic Web LayerCake



RDFS Vocabulary



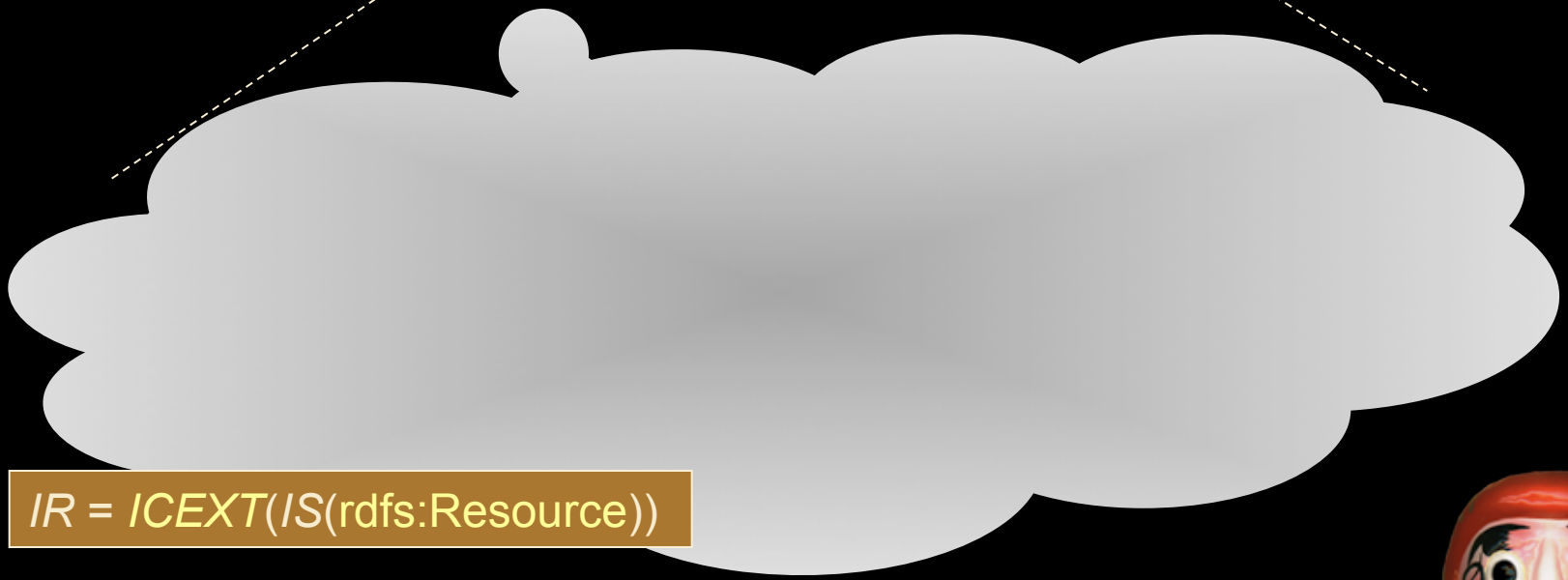
RDFS Semantic Condition



RDFS Semantic Condition

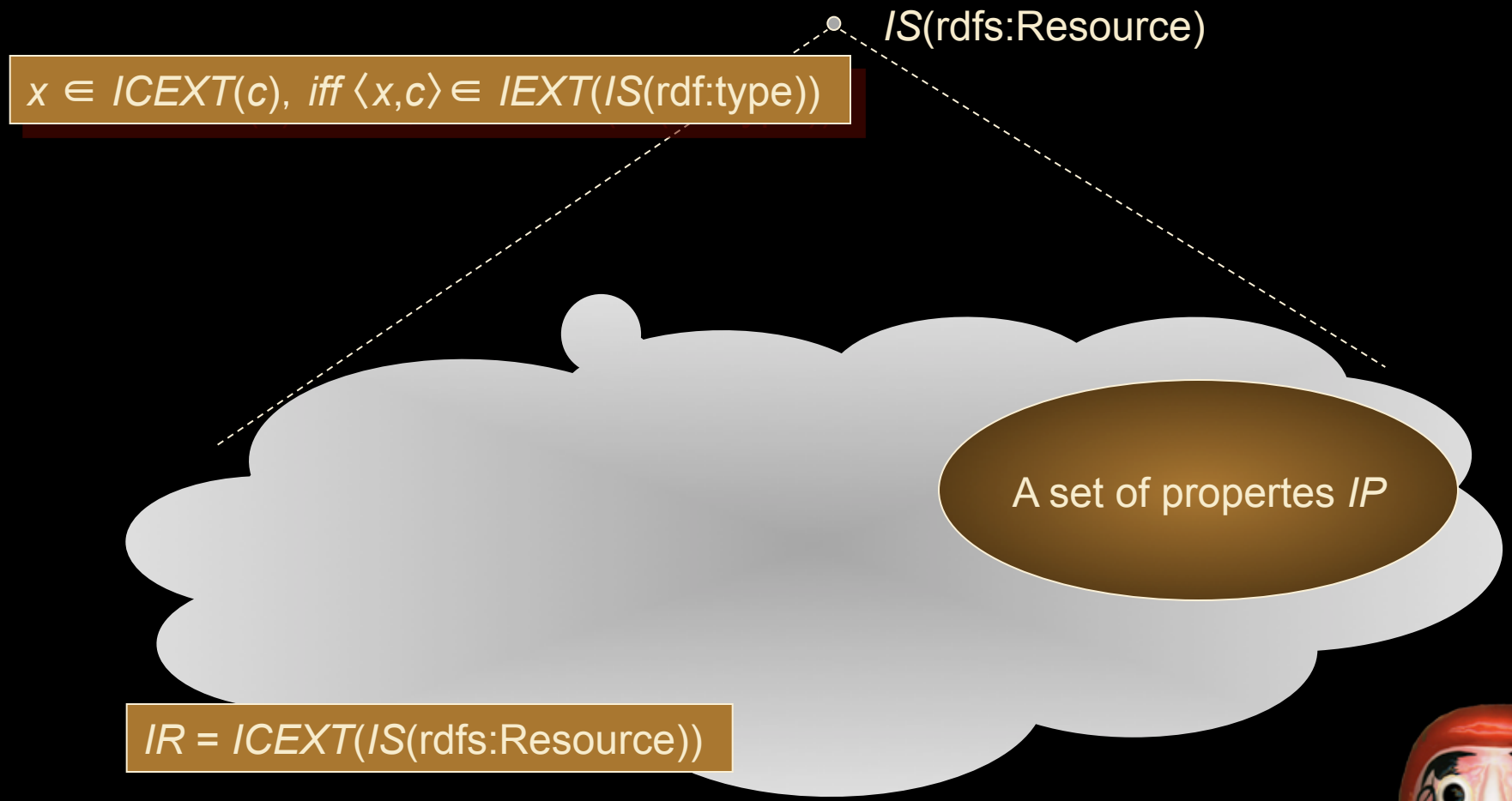
IS(rdfs:Resource)

$x \in ICEXT(c)$, *iff* $\langle x,c \rangle \in IEXT(IS(\text{rdf:type}))$

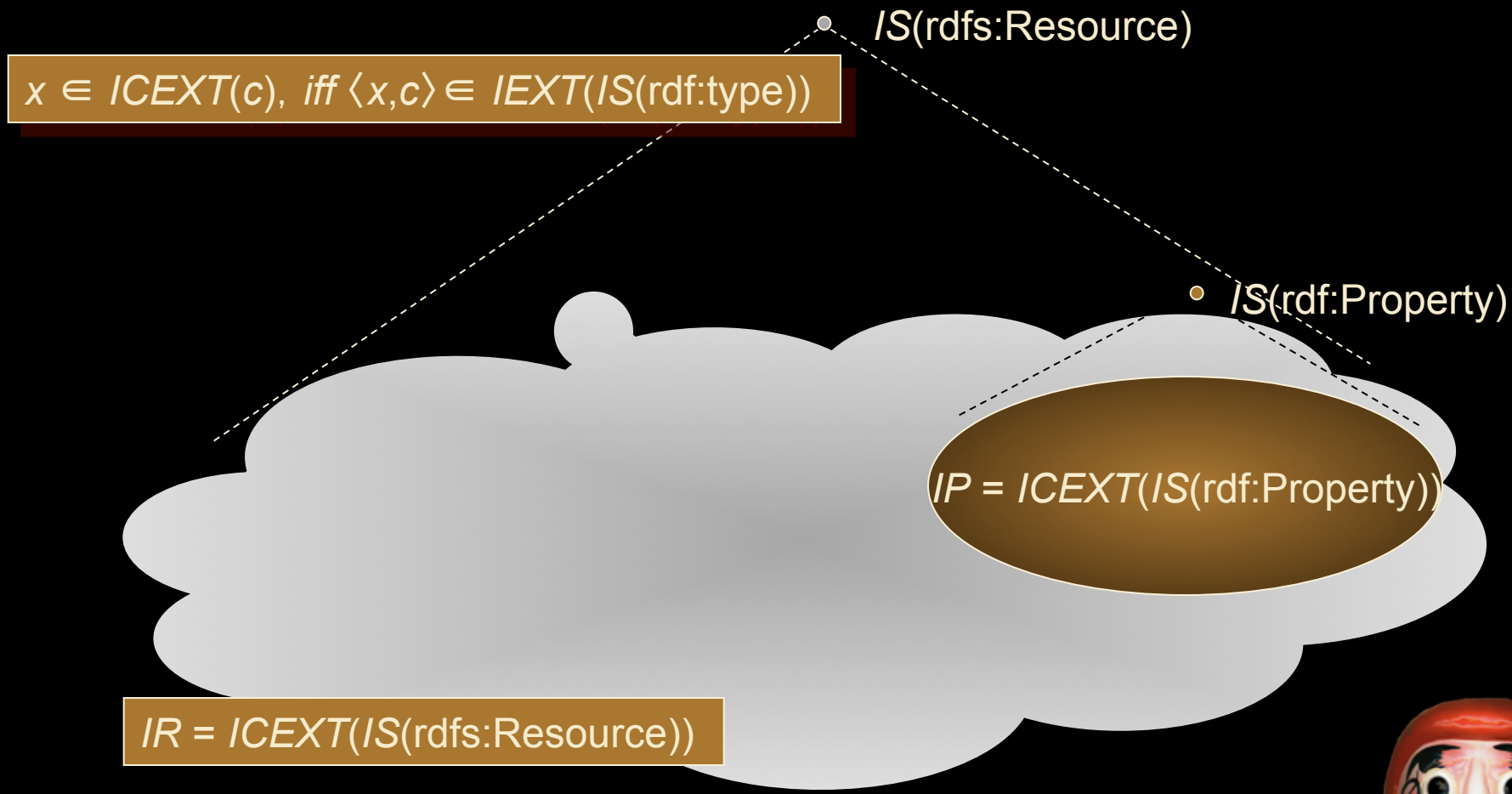


$IR = ICEXT(IS(\text{rdfs:Resource}))$

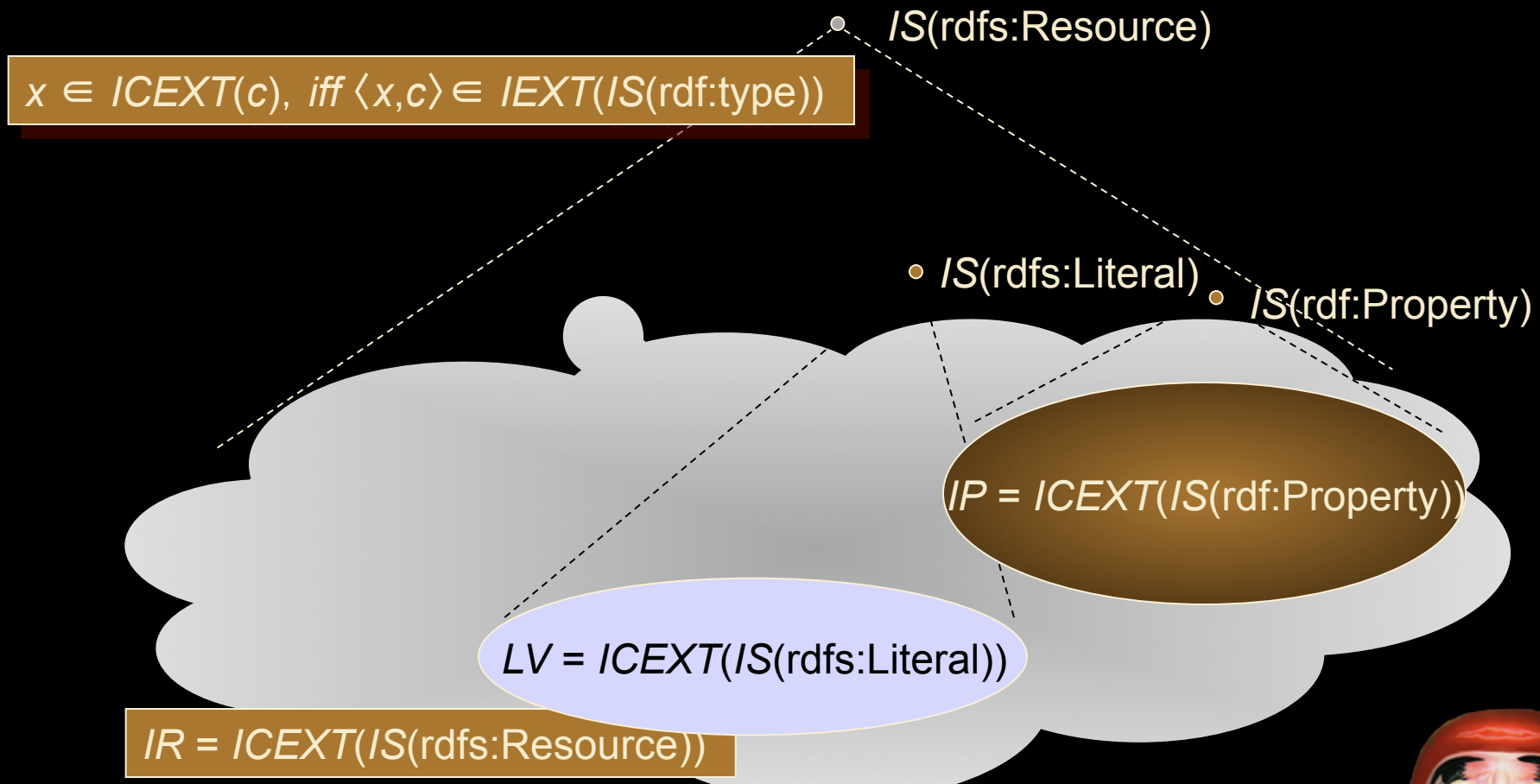
RDFS Semantic Condition



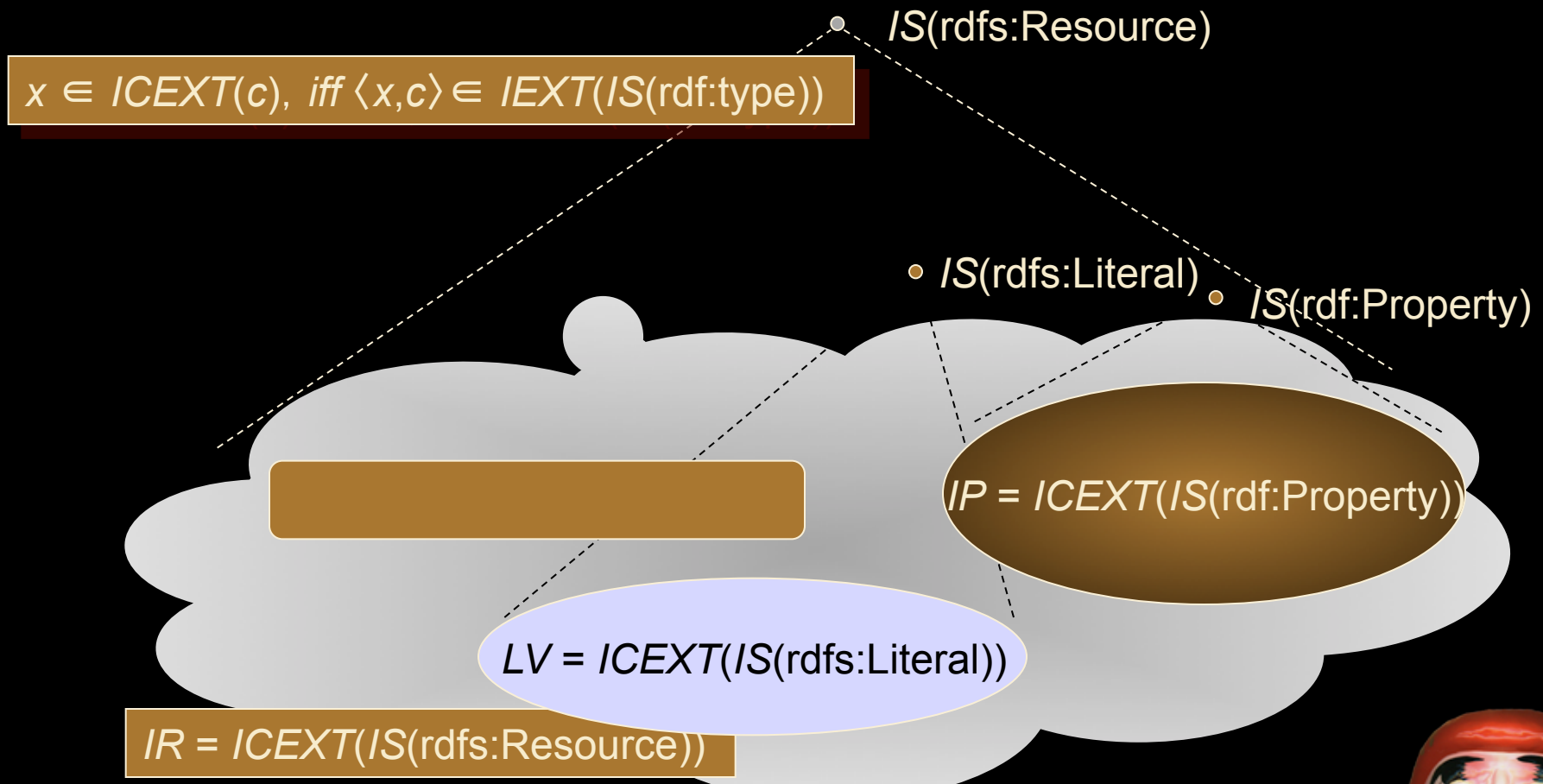
RDFS Semantic Condition



RDFS Semantic Condition

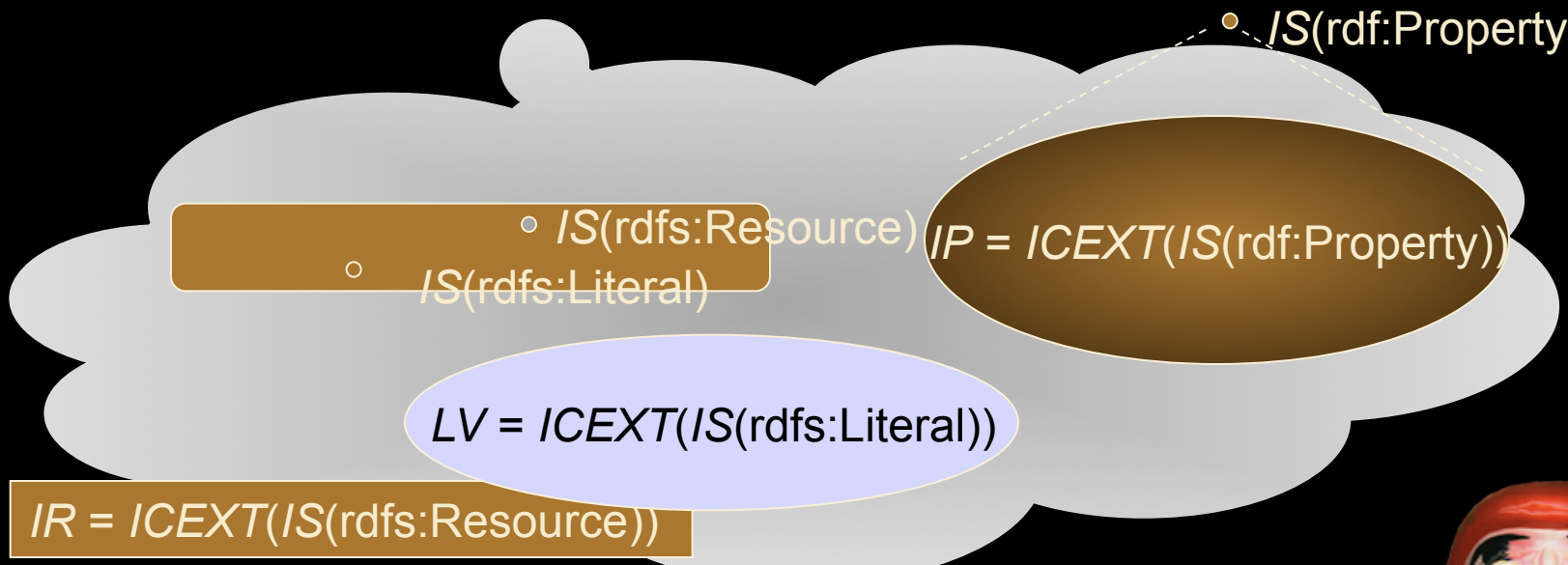


RDFS Semantic Condition



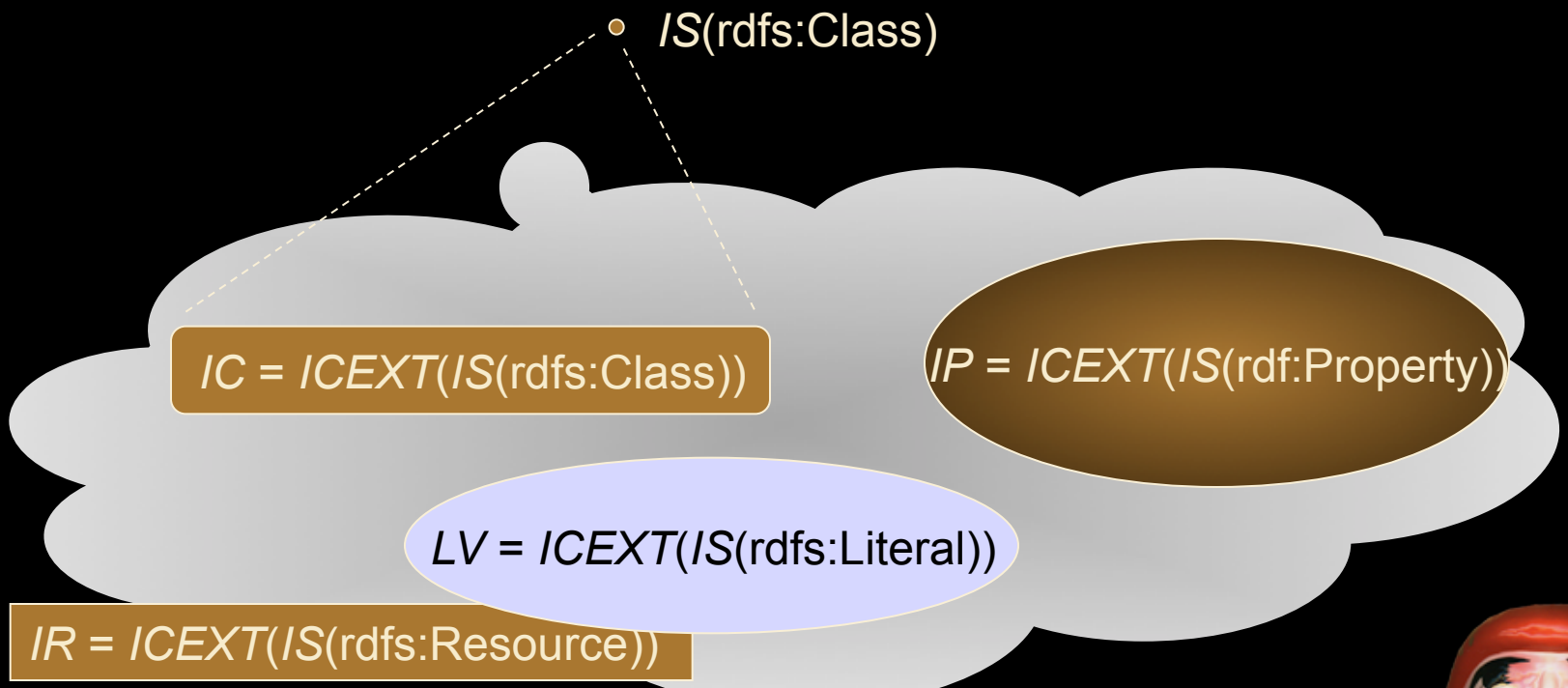
RDFS Semantic Condition

$x \in ICEXT(c)$, *iff* $\langle x,c \rangle \in IEXT(IS(\text{rdf:type}))$



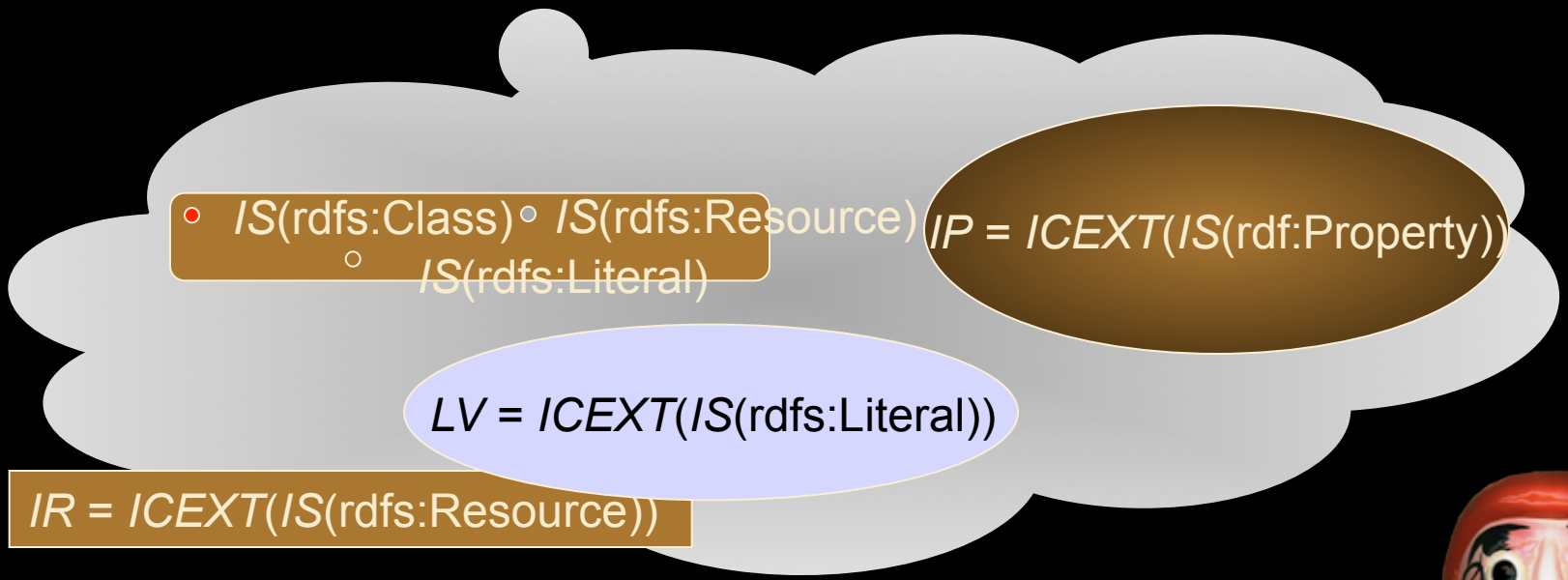
RDFS Semantic Condition

$x \in ICEXT(c)$, *iff* $\langle x,c \rangle \in IEXT(IS(\text{rdf:type}))$



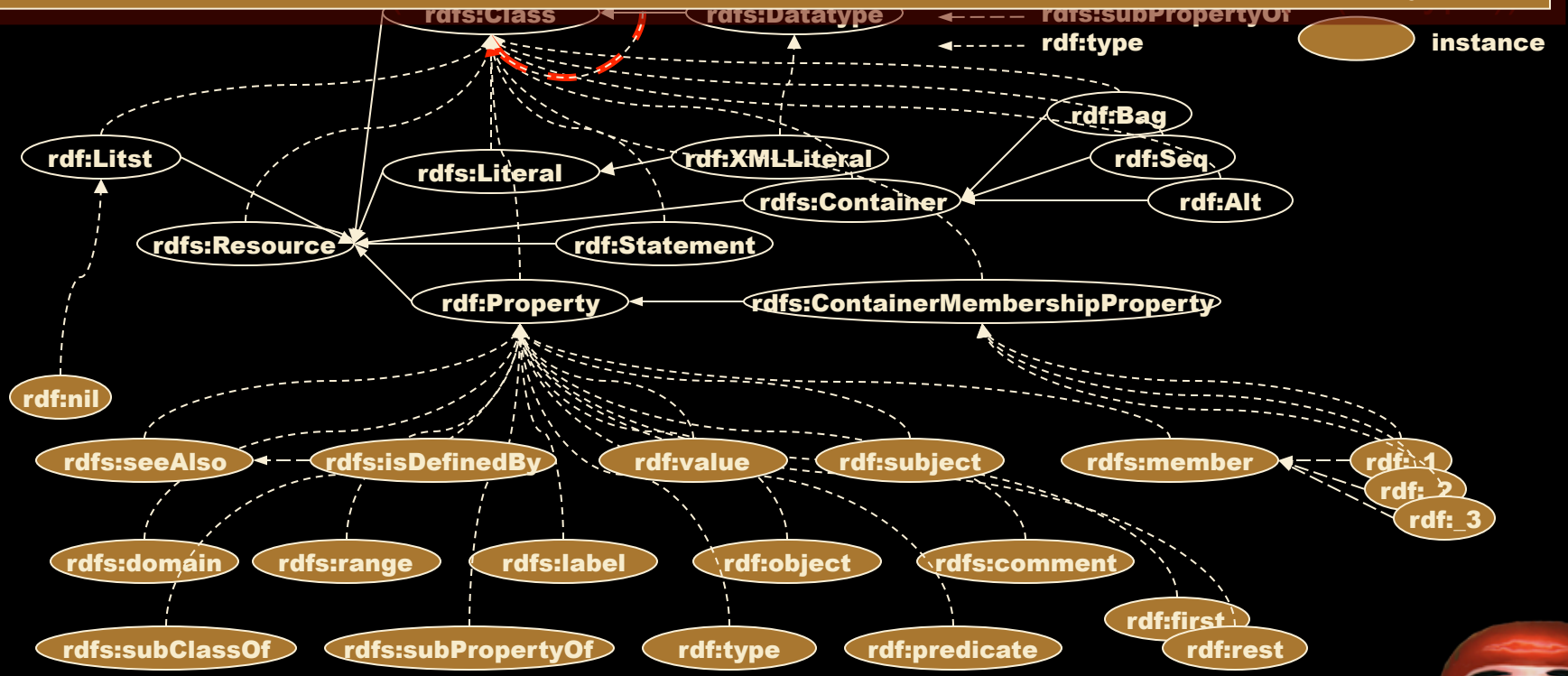
RDFS Semantic Condition

$x \in ICEXT(c)$, *iff* $\langle x,c \rangle \in IEXT(IS(\text{rdf:type}))$



RDFS Axiomatic Triples

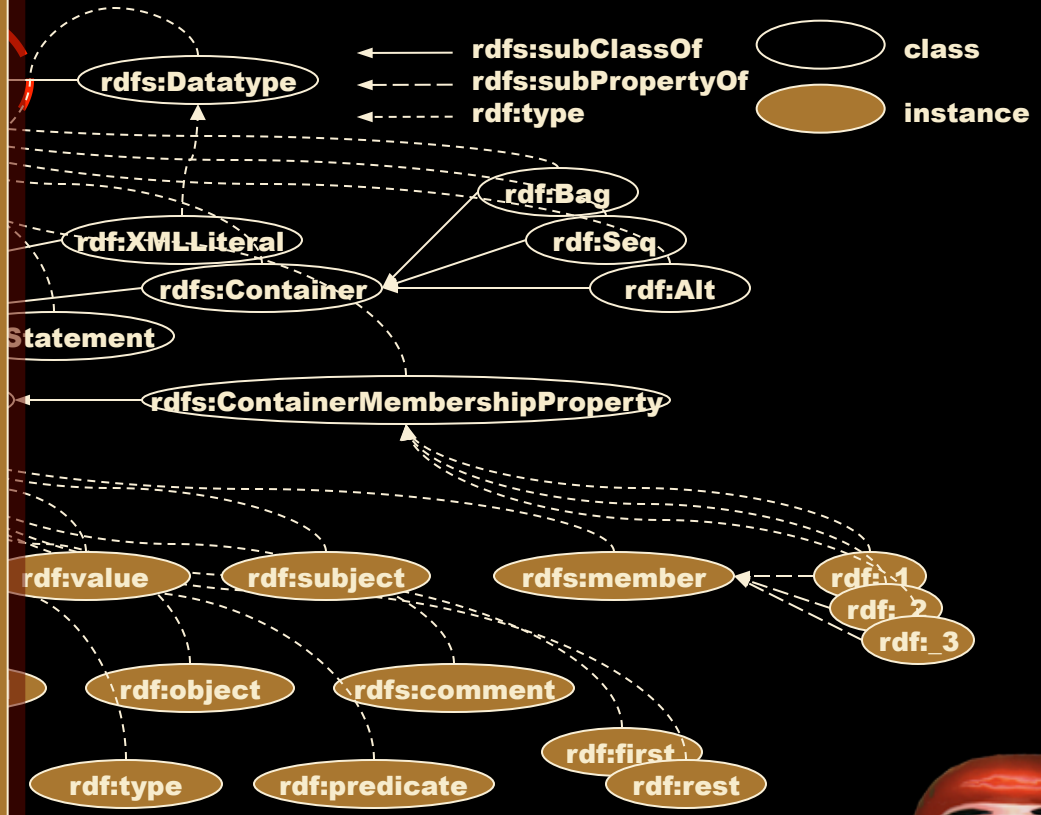
rdfs:Class ∈ *ICEXT*(rdfs:Class), *iff* ⟨rdfs:Class,rdfs:Class⟩ ∈ *IEXT*(*IS*(rdf:type))



atic Triples

rdf:type rdfs:domain rdfs:Resource .
rdfs:domain rdfs:domain rdf:Property .
rdfs:range rdfs:domain rdf:Property .
rdfs:subPropertyOf rdfs:domain rdf:Property .
rdfs:subClassOf rdfs:domain rdfs:Class .
rdf:subject rdfs:domain rdf:Statement .
rdf:predicate rdfs:domain rdf:Statement .
rdf:object rdfs:domain rdf:Statement .
rdfs:member rdfs:domain rdfs:Resource .
rdf:first rdfs:domain rdf:List .
rdf:rest rdfs:domain rdf:List .
rdfs:seeAlso rdfs:domain rdfs:Resource .
rdfs:isDefinedBy rdfs:domain rdfs:Resource .
rdfs:comment rdfs:domain rdfs:Resource .
rdfs:label rdfs:domain rdfs:Resource .
rdf:value rdfs:domain rdfs:Resource .

rdf:type rdfs:range rdfs:Class .
rdfs:domain rdfs:range rdfs:Class .
rdfs:range rdfs:range rdfs:Class .
rdfs:subPropertyOf rdfs:range rdf:Property .
rdfs:subClassOf rdfs:range rdfs:Class .
rdf:subject rdfs:range rdfs:Resource .
rdf:predicate rdfs:range rdfs:Resource .
rdf:object rdfs:range rdfs:Resource .
rdfs:member rdfs:range rdfs:Resource .
rdf:first rdfs:range rdfs:Resource .
rdf:rest rdfs:range rdf:List .
rdfs:seeAlso rdfs:range rdfs:Resource .
rdfs:isDefinedBy rdfs:range rdfs:Resource .
rdfs:comment rdfs:range rdfs:Literal .
rdfs:label rdfs:range rdfs:Literal .
rdf:value rdfs:range rdfs:Resource .



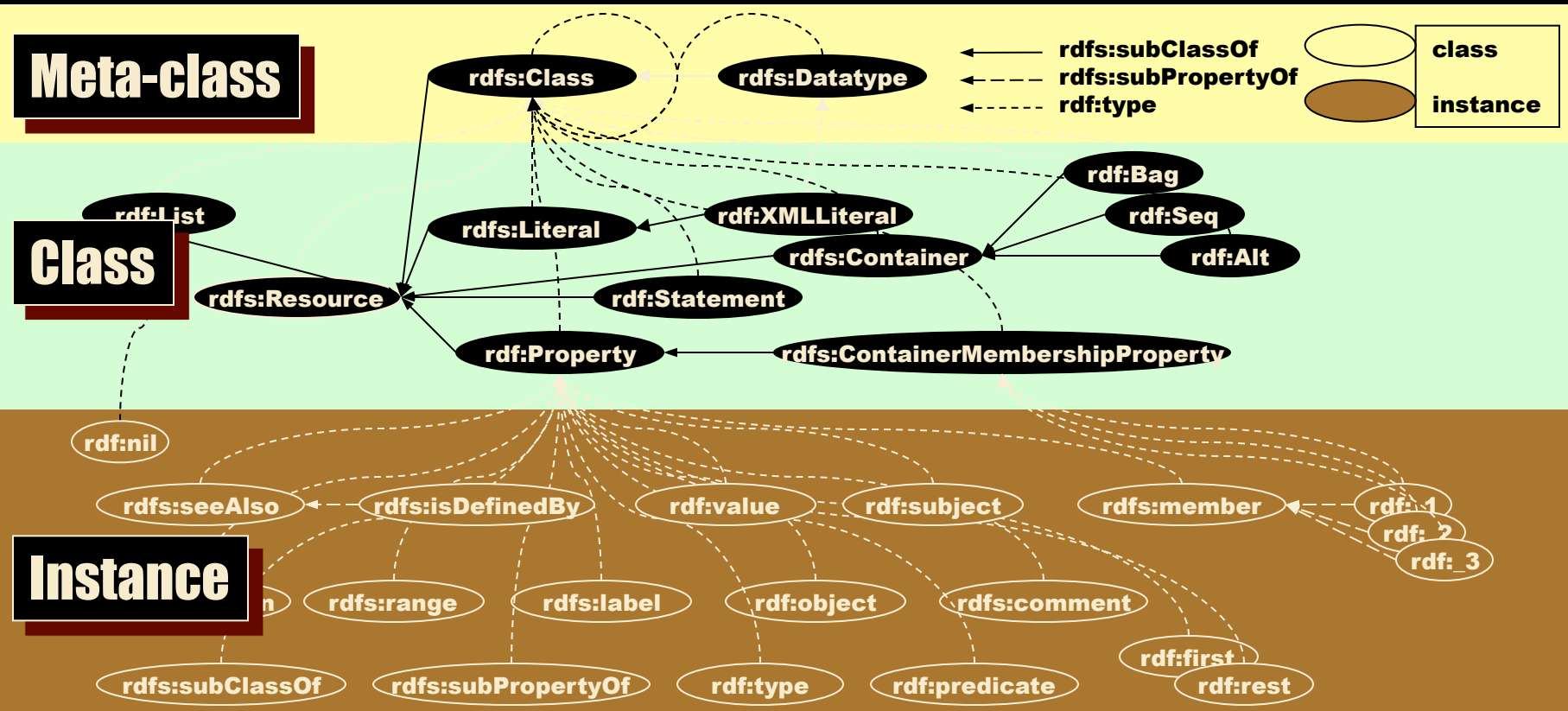
RDFS Entailment Rules

Rule Name	If E contains:	then add:
rdfs1	uuu aaa lll. where lll is a plain literal (with or without a language tag).	_:nnn rdf:type rdfs:Literal . where _:nnn identifies a blank node allocated to lll by rule rule lg.
rdfs2	aaa rdfs:domain xxx . uuu aaa yyy .	uuu rdf:type xxx .
rdfs3	aaa rdfs:range xxx . uuu aaa vvv .	vvv rdf:type xxx .
rdfs4a	uuu aaa xxx .	uuu rdf:type rdfs:Resource .
rdfs4b	uuu aaa vvv.	vvv rdf:type rdfs:Resource .
rdfs5	uuu rdfs:subPropertyOf vvv . vvv rdfs:subPropertyOf xxx .	uuu rdfs:subPropertyOf xxx .
rdfs6	uuu rdf:type rdf:Property .	uuu rdfs:subPropertyOf uuu .

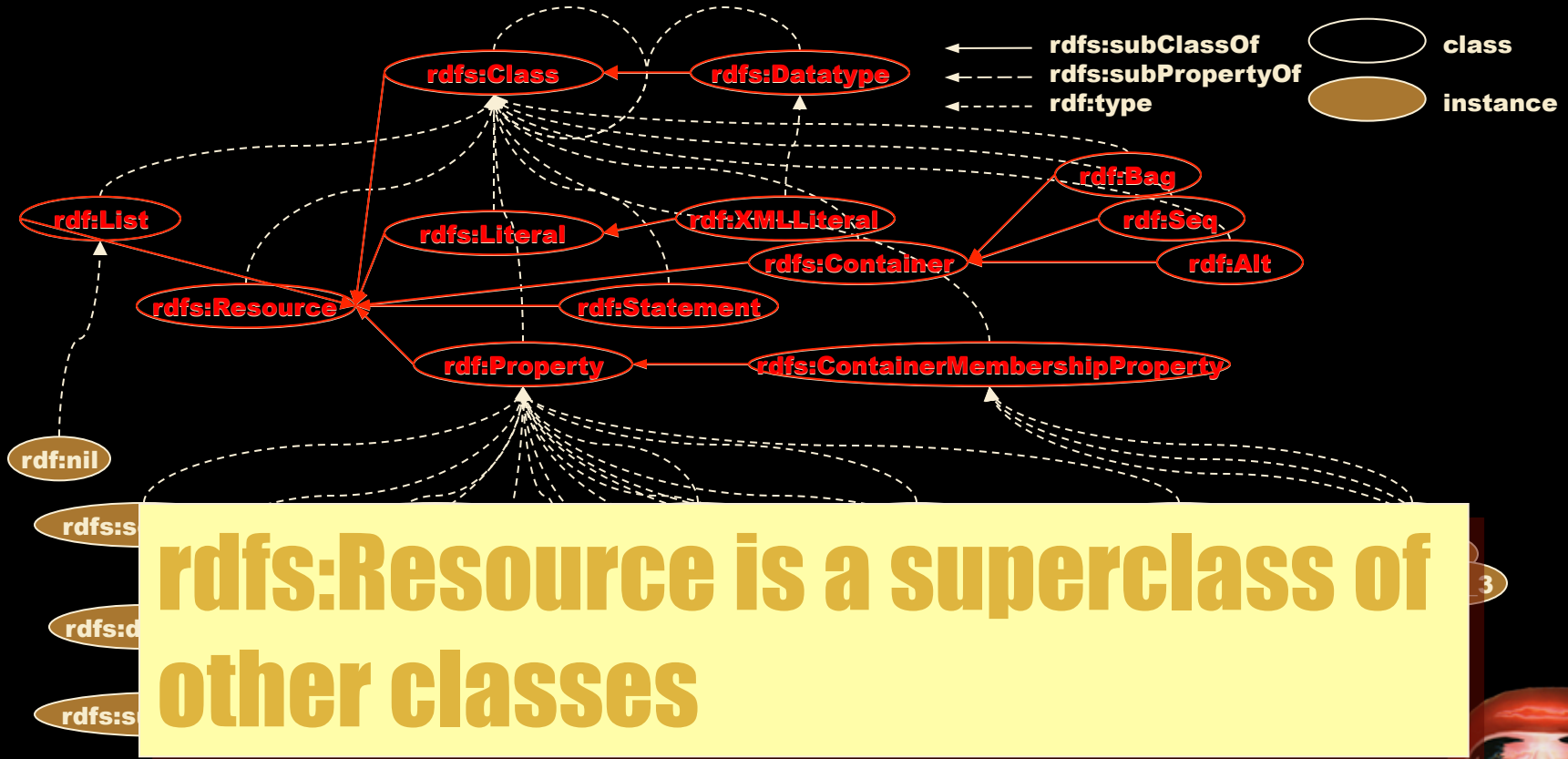
RDFS Entailment Rules

Rule Name	If E contains:	then add:
rdfs7	aaa rdfs:subPropertyOf bbb . uuu aaa yyy .	uuu bbb yyy .
rdfs8	uuu rdf:type rdfs:Class .	uuu rdfs:subClassOf rdfs:Resource .
rdfs9	uuu rdfs:subClassOf xxx . vvv rdf:type uuu .	vvv rdf:type xxx .
rdfs10	uuu rdf:type rdfs:Class .	uuu rdfs:subClassOf uuu .
rdfs11	uuu rdfs:subClassOf vvv . vvv rdfs:subClassOf xxx .	uuu rdfs:subClassOf xxx .
rdfs12	uuu rdf:type rdfs:ContainerMembershipProperty .	uuu rdfs:subPropertyOf rdfs:member .
rdfs13	uuu rdf:type rdfs:Datatype .	uuu rdfs:subClassOf rdfs:Literal .

Hierarchical Structure of RDFS

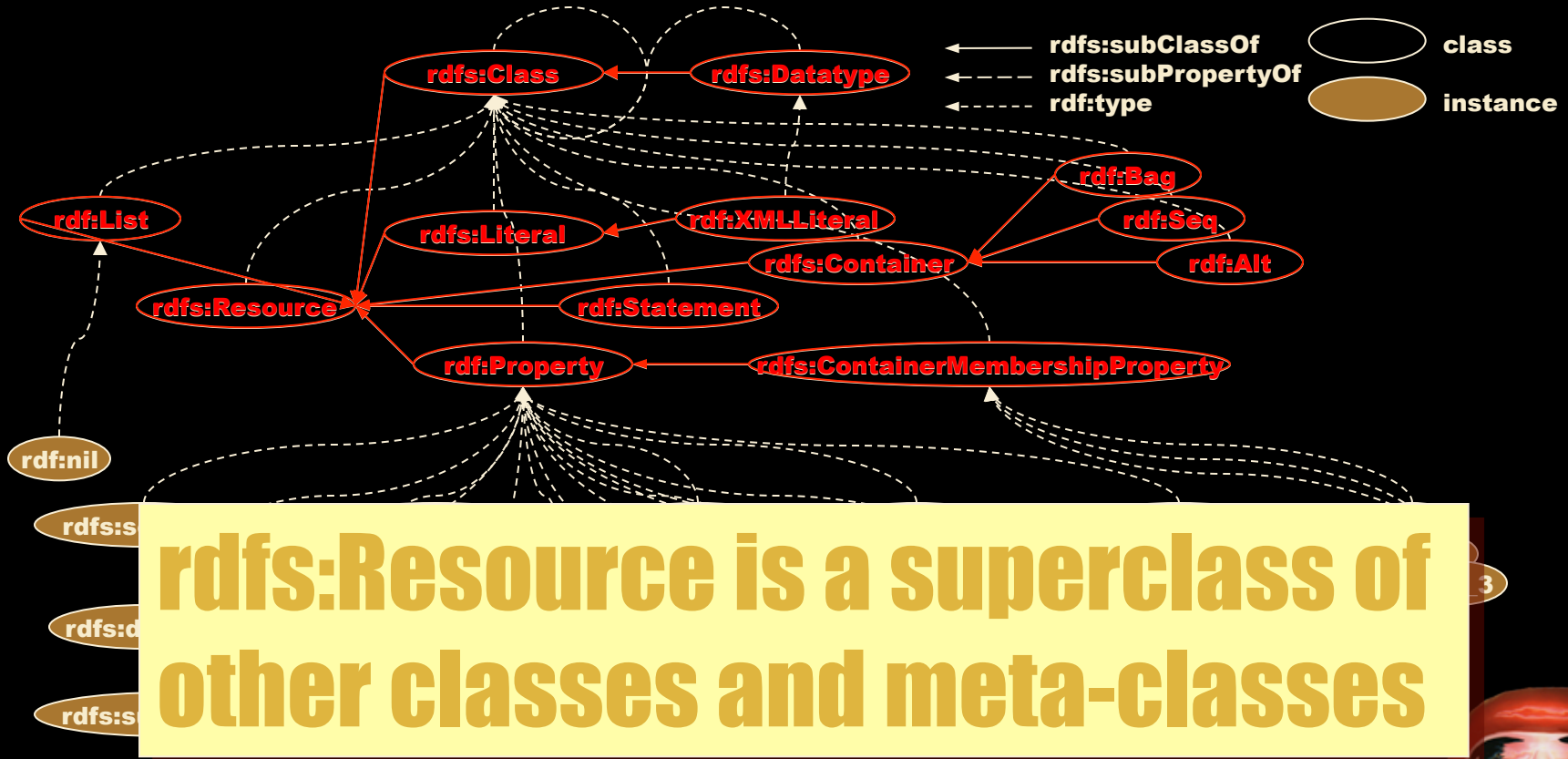


Transitivity Rule implies …



rdfs:Resource is a superclass of other classes

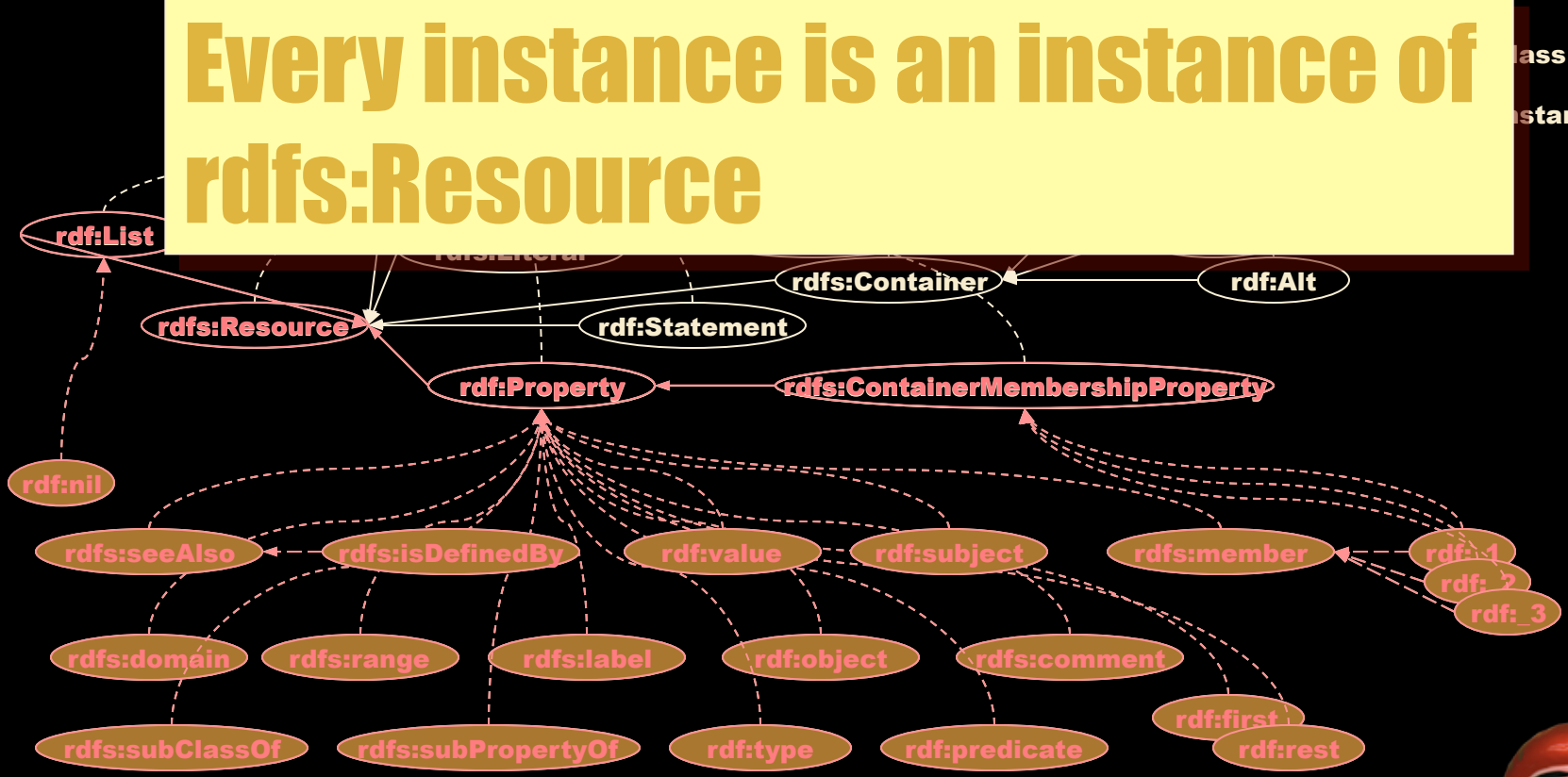
Transitivity Rule implies …



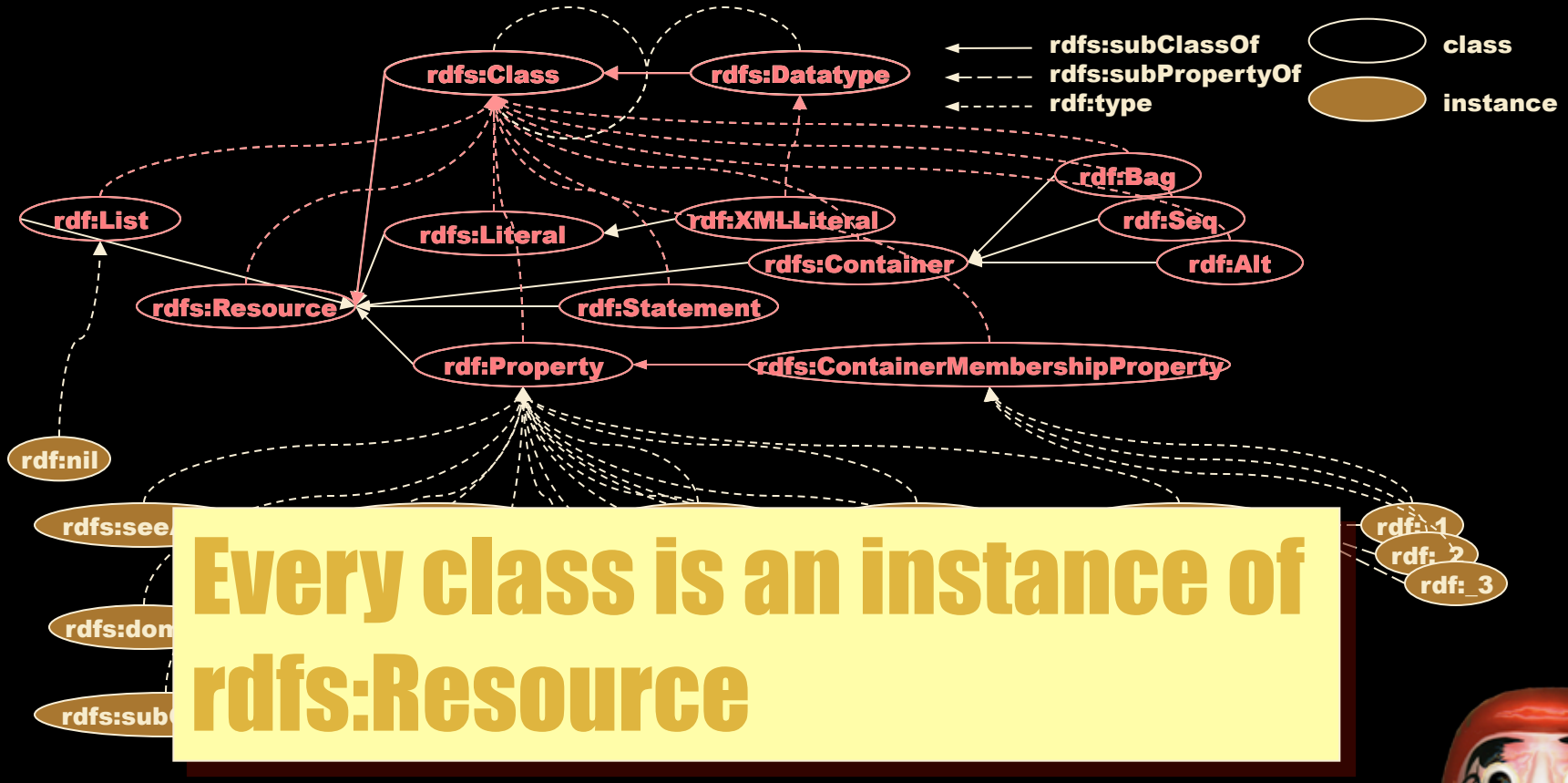
rdfs:Resource is a superclass of other classes and meta-classes

Subsumption Rule implies …

Every instance is an instance of rdfs:Resource

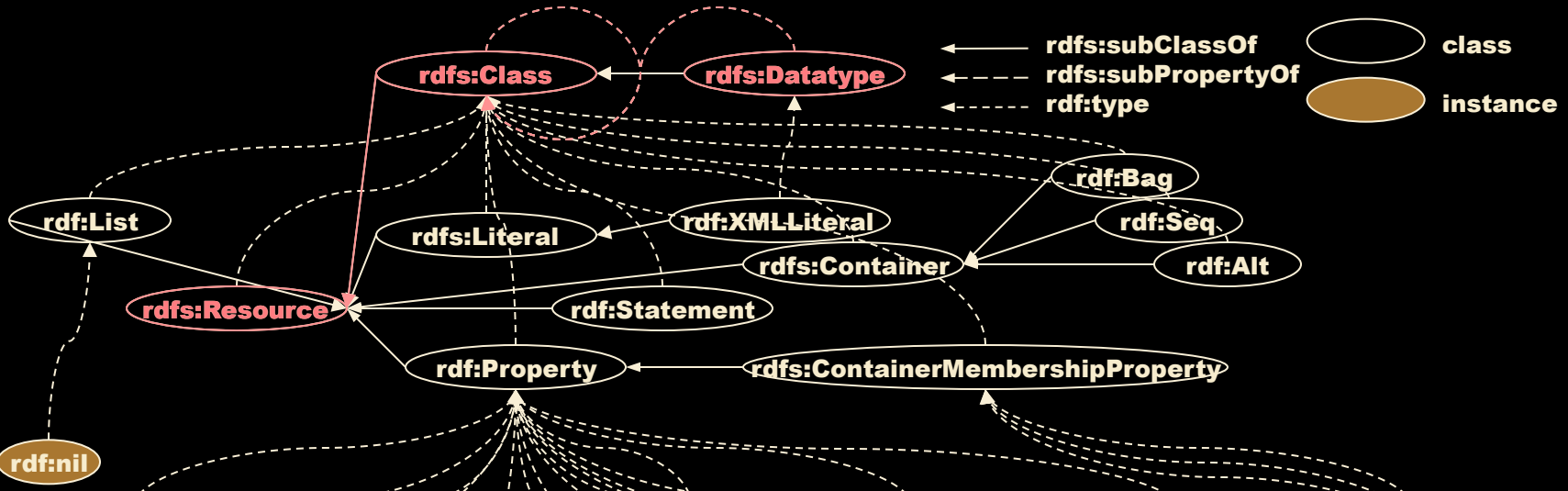


Subsumption Rule implies …



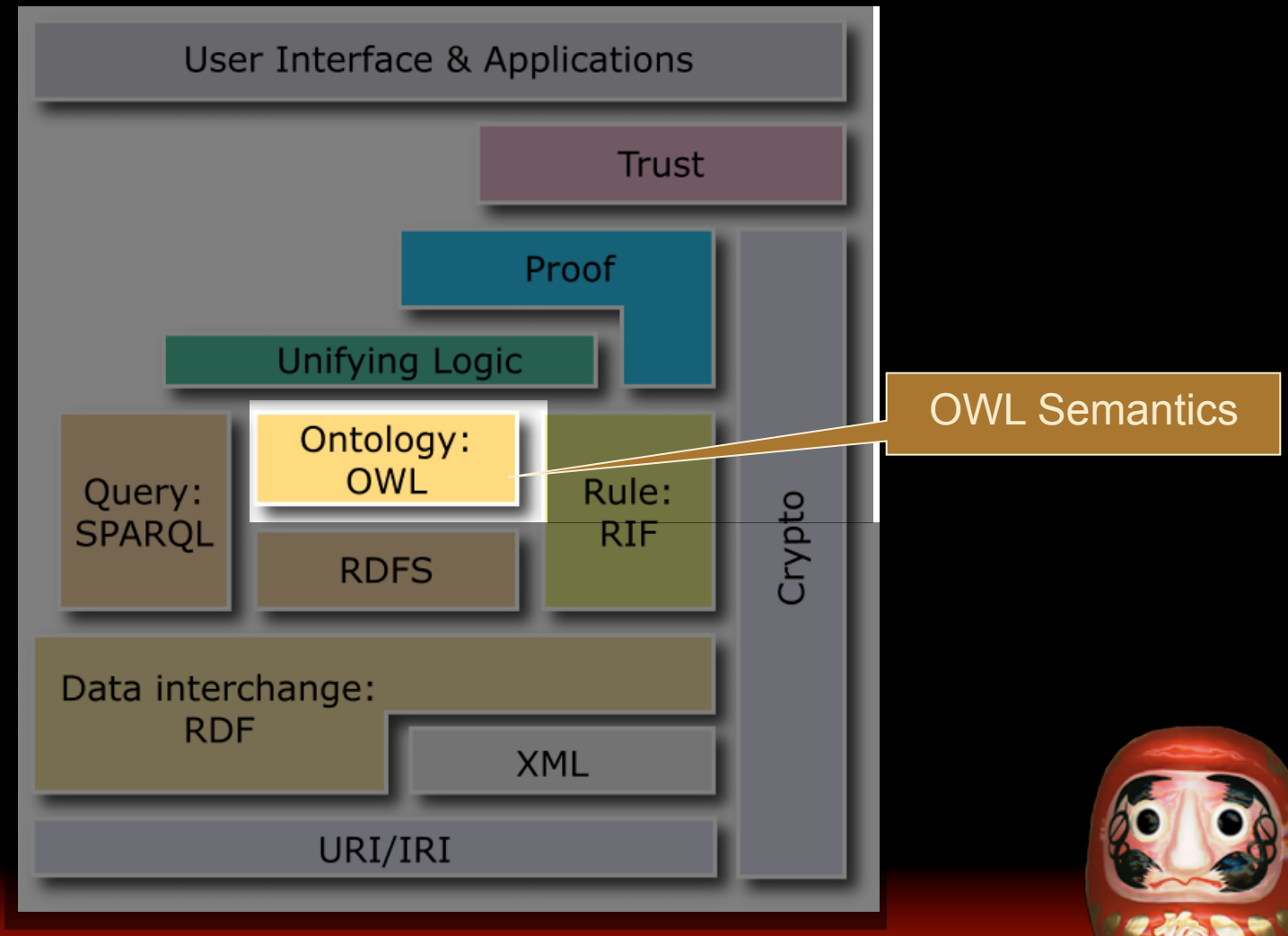
Every class is an instance of rdfs:Resource

Subsumption Rule implies …



rdfs:Datatype and rdfs:Class are also instances of rdfs:Resource

Semantic Web LayerCake



Three sub languages of OWL

- OWL Lite
  - **supports users primarily needing a classification hierarchy and simple constraints. It should be simpler to provide tool support for OWL Lite than the other sublanguages.**
- OWL DL
  - supports users who want the maximum expressiveness while retaining computational completeness (all conclusions are guaranteed to be computable) and decidability (all computations will finish in finite time).
- OWL Full
  - is meant for users who want maximum expressiveness and the syntactic freedom of RDF with no computational guarantees. For example, in OWL Full a class can be treated simultaneously as a collection of individuals and as an individual in its own right.

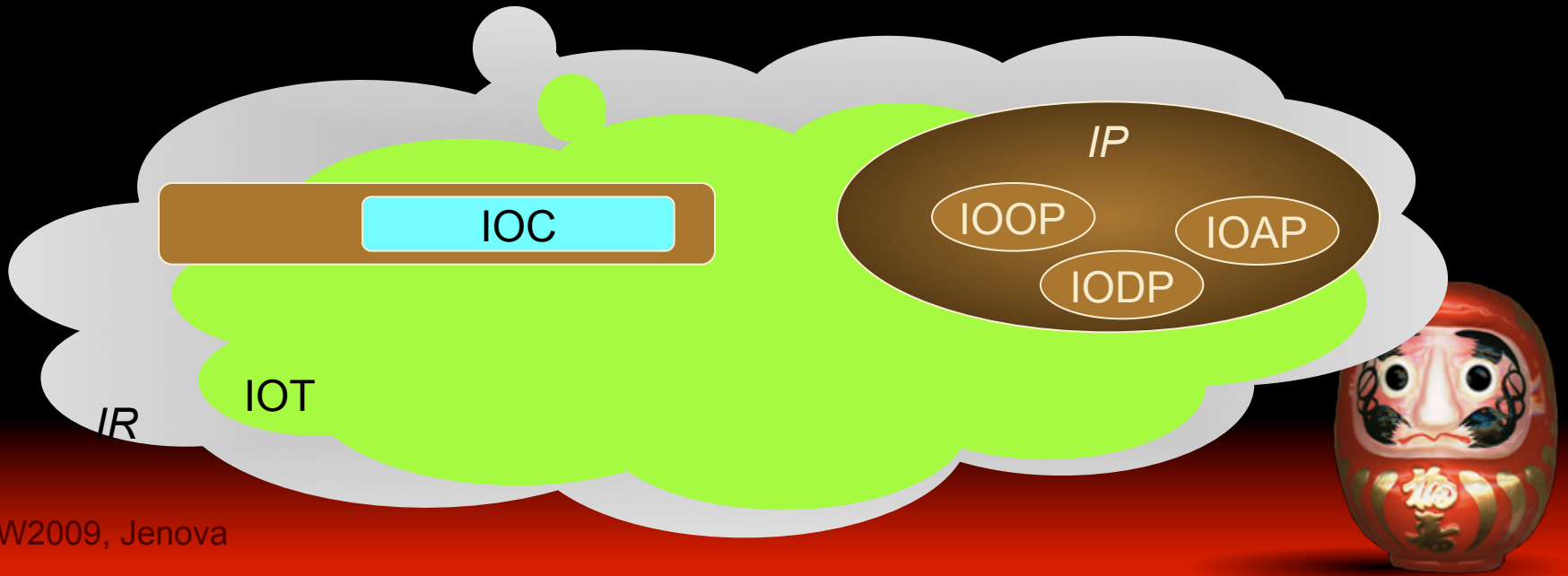
Conditions concerning the parts of the OWL universe and syntactic categories

If E is	then		
	$IS(E) \in$	$ICEXT(IS(E)) =$	and
owl:Class	*IC*	**IOC**	**IOC** $\subseteq$ *IC*
owl:Thing	**IOC**	**IOT**	**IOT** $\subseteq$ *IR* and **IOT** $\neq \varnothing$
owl:Restriction	*IC*	**IOR**	**IOR** $\subseteq$ **IOC**
owl:ObjectProperty	*IC*	**IOOP**	**IOOP** $\subseteq$ *IP*
owl:DatatypeProperty	*IC*	**IODP**	**IODP** $\subseteq$ *IP*
owl:AnnotationProperty	*IC*	**IOAP**	**IOAP** $\subseteq$ *IP*

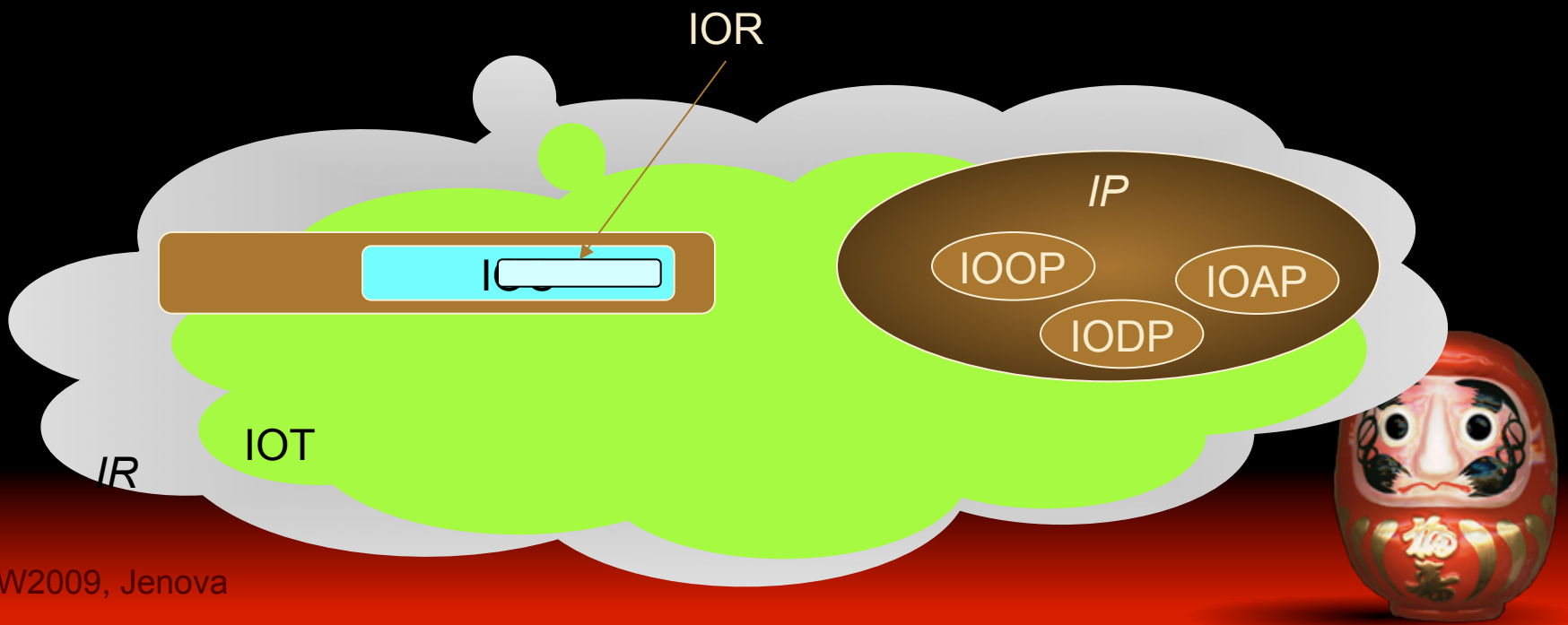
RDF-Compatible Model-Theoretic Semantics

RDF-Compatible Model-Theoretic Semantics

RDF-Compatible Model-Theoretic Semantics



RDF-Compatible Model-Theoretic Semantics



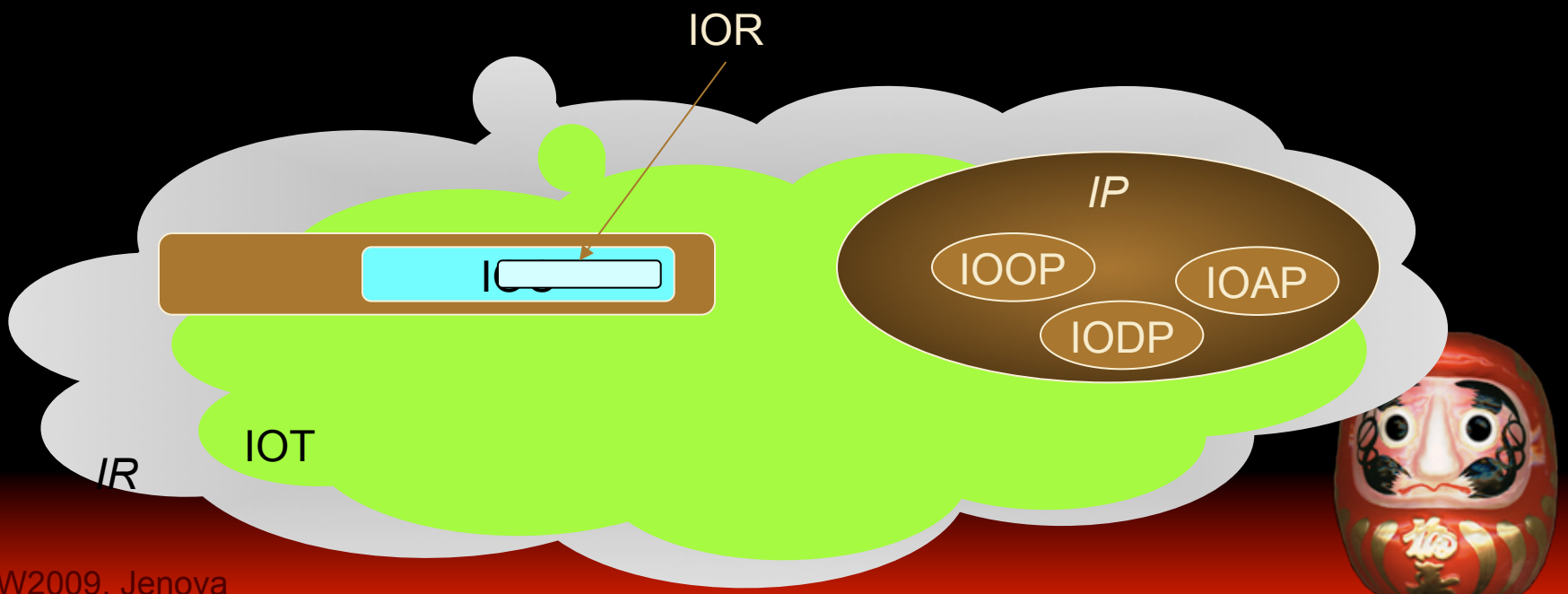
RDF-Compatible Model-Theoretic Semantics

OWL Full

IOT = *IR*
IOOP = *IP*
IOC = *IC*

OWL DL

LVI, IOT, IOC, IDC, IOOP, IODP, IOAP, IOXP, IL, and IX are all pairwise disjoint.
For v in the disallowed vocabulary, *IS*(v) ∈ *IR* − (LV ∪ IOT ∪ IOC ∪ IDC ∪ IOOP ∪ IODP ∪ IOAP ∪ IOXP ∪ IL ∪ IX).



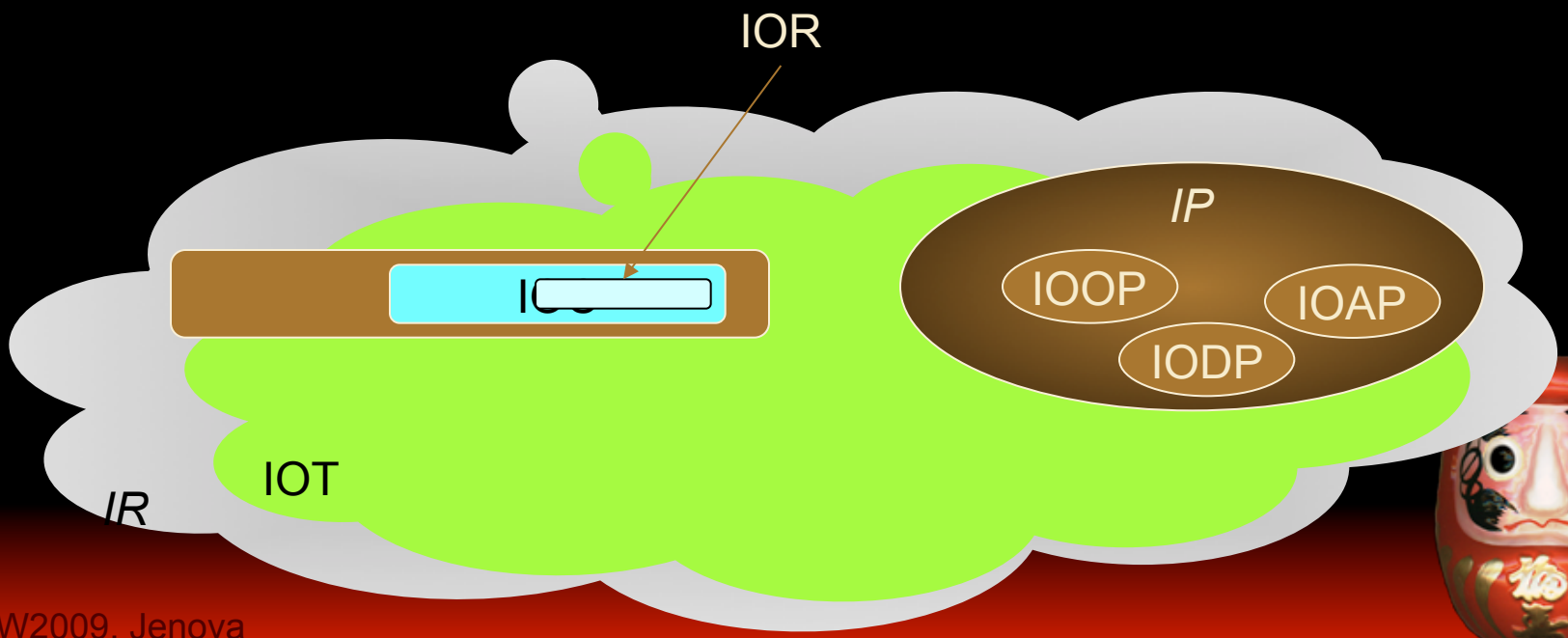
RDF-Compatible Model-Theoretic Semantics

OWL Full

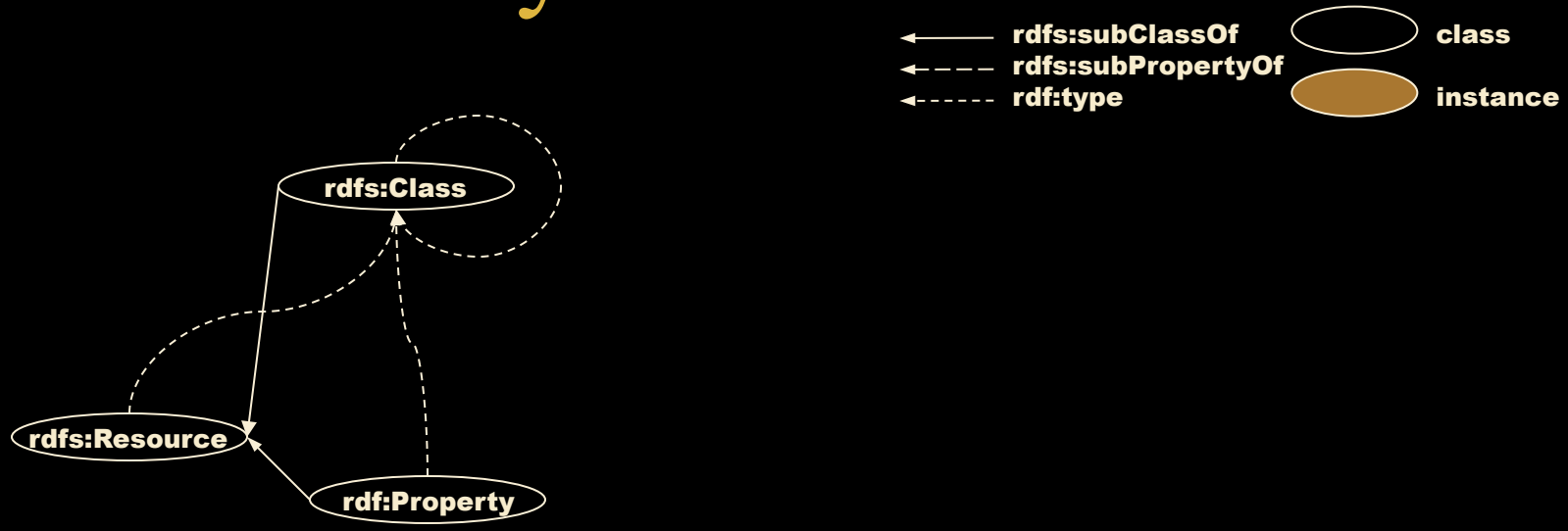
IOT ⊆ *IR*
IOOP ⊆ *IP*
IOC ⊆ *IC*

OWL DL

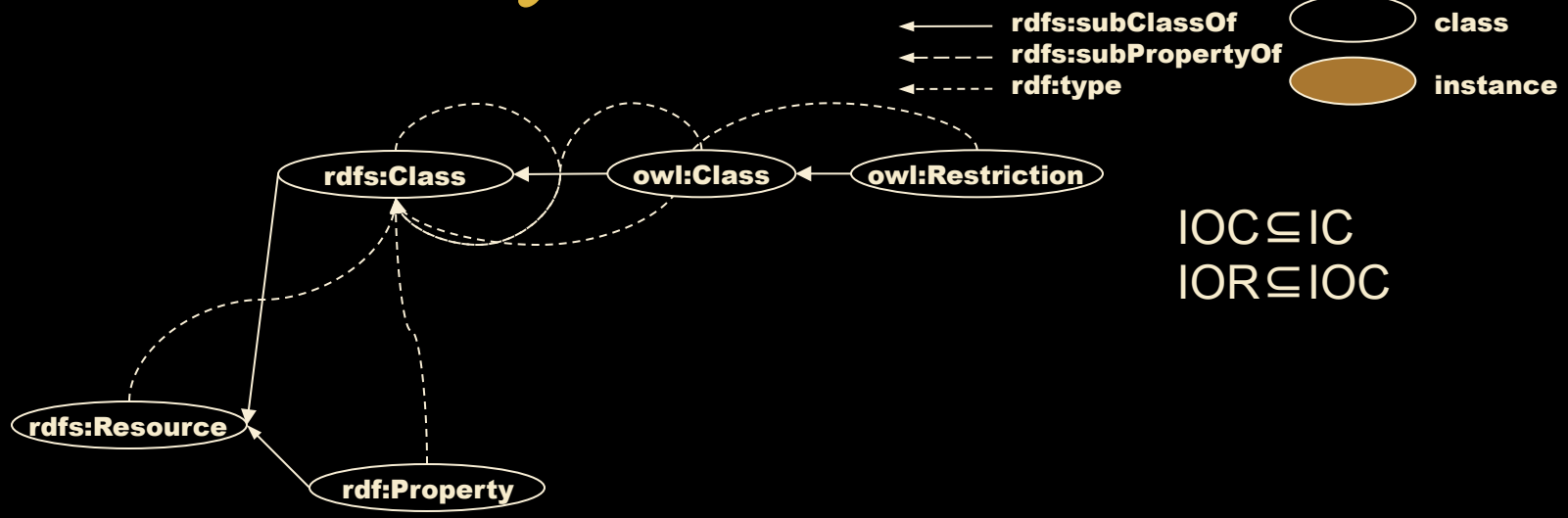
LVI, IOT, IOC, IDC, IOOP, IODP, IOAP, IOXP, IL, and IX are all pairwise disjoint.
For v in the disallowed vocabulary, *IS*(v) ∈ *IR* − (LV ∪ IOT ∪ IOC ∪ IDC ∪ IOOP ∪ IODP ∪ IOAP ∪ IOXP ∪ IL ∪ IX).



OWL-Full Style Connection



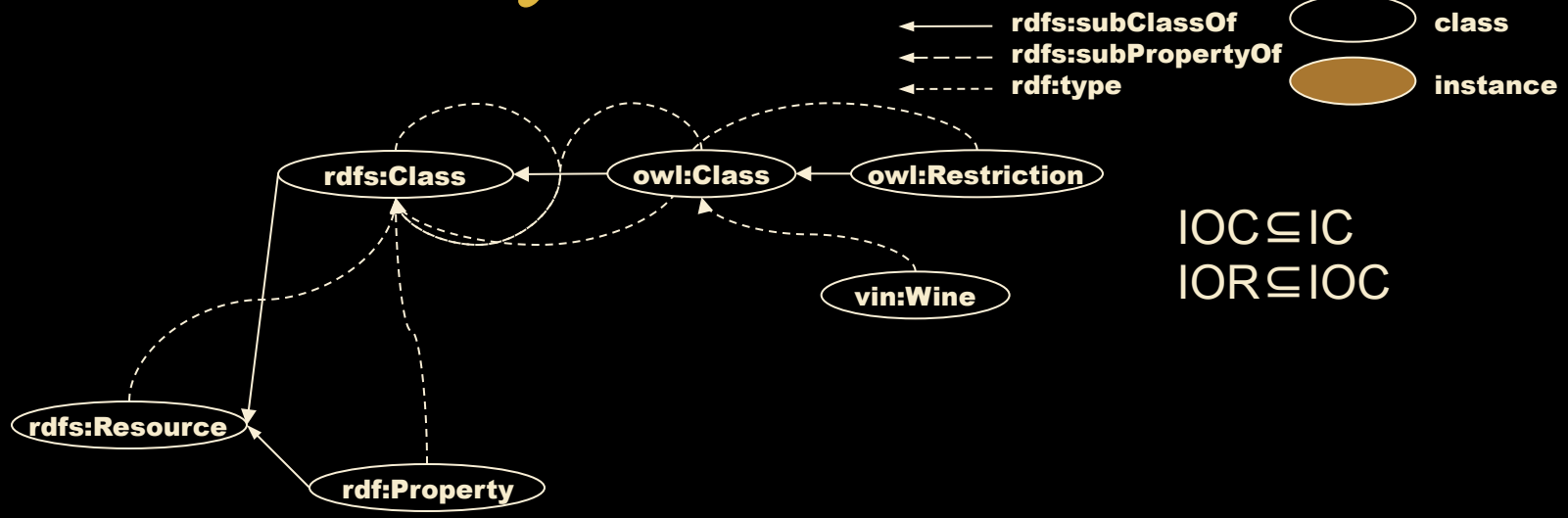
OWL-Full Style Connection



$IOC \subseteq IC$

$IOR \subseteq IOC$

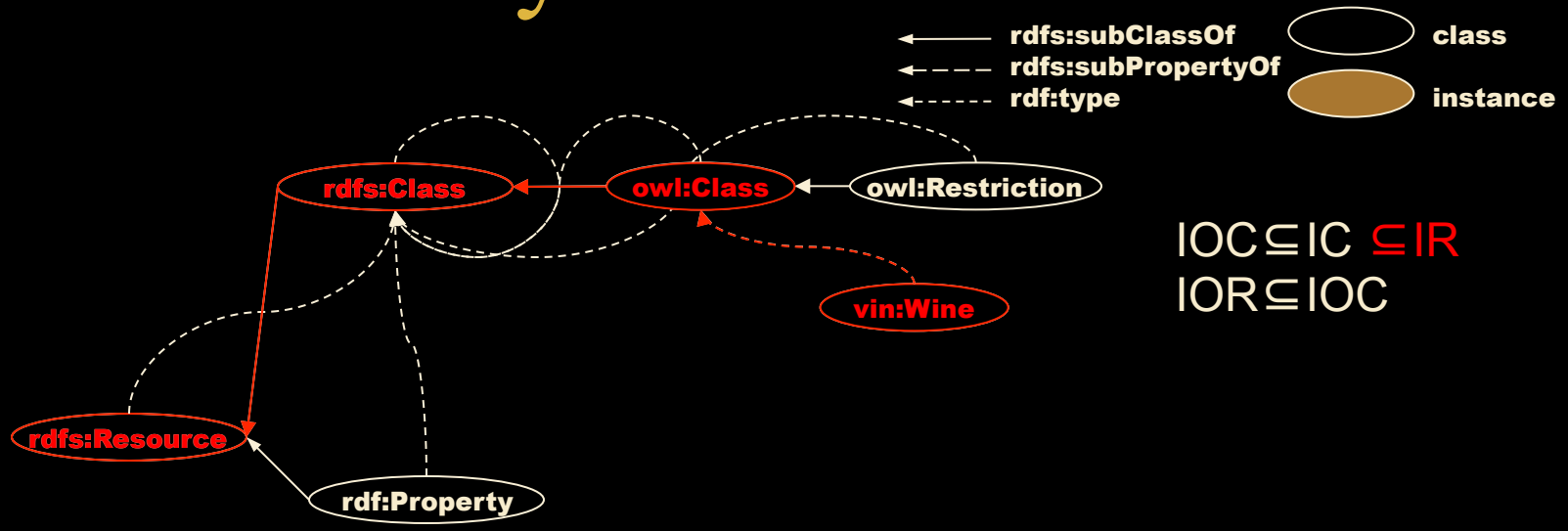
OWL-Full Style Connection



$IOC \subseteq IC$

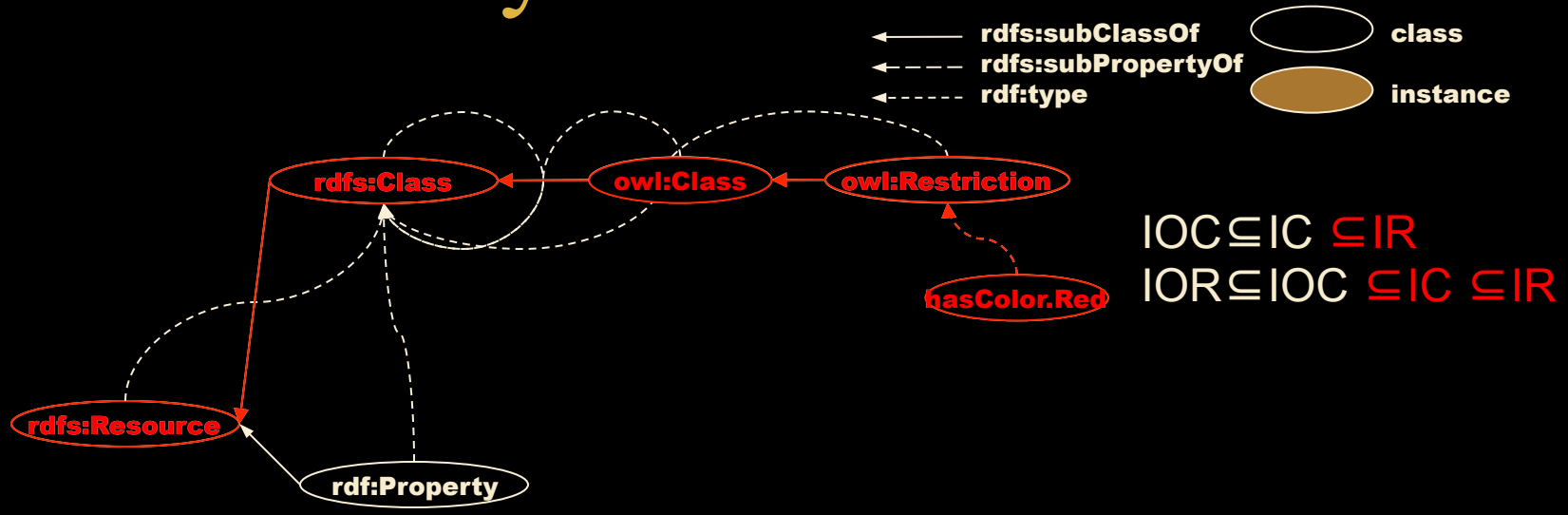
$IOR \subseteq IOC$

OWL-Full Style Connection



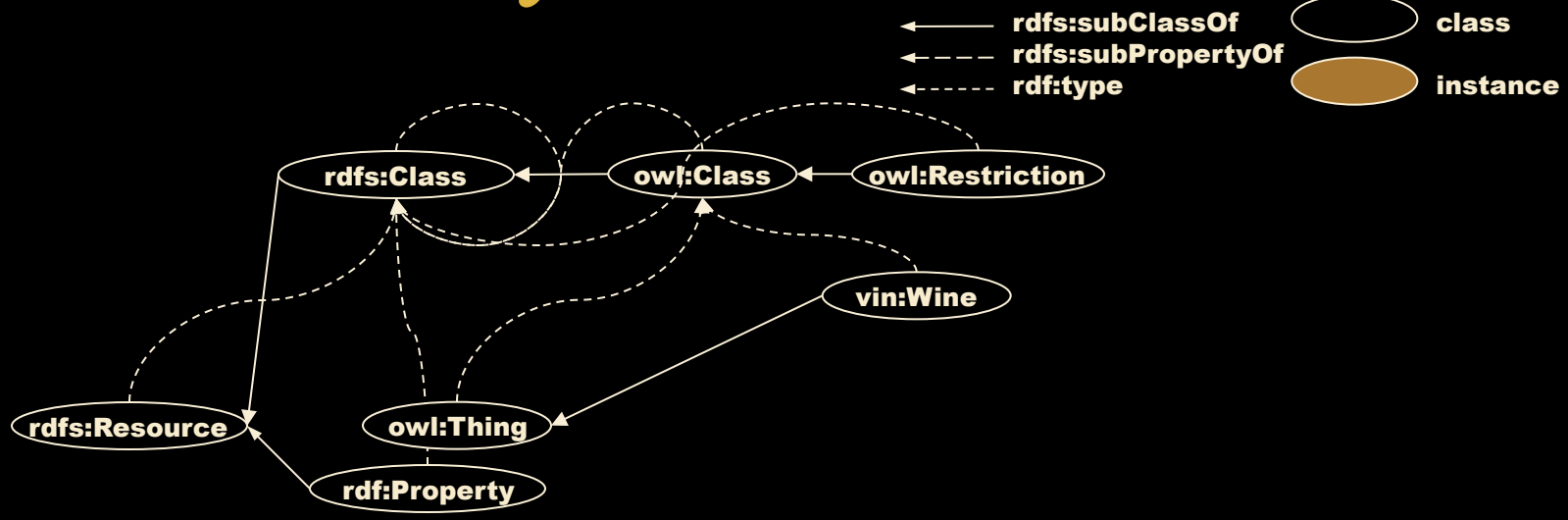
- All classes under owl:Class are in the domain of rdfs:Resource.

OWL-Full Style Connection

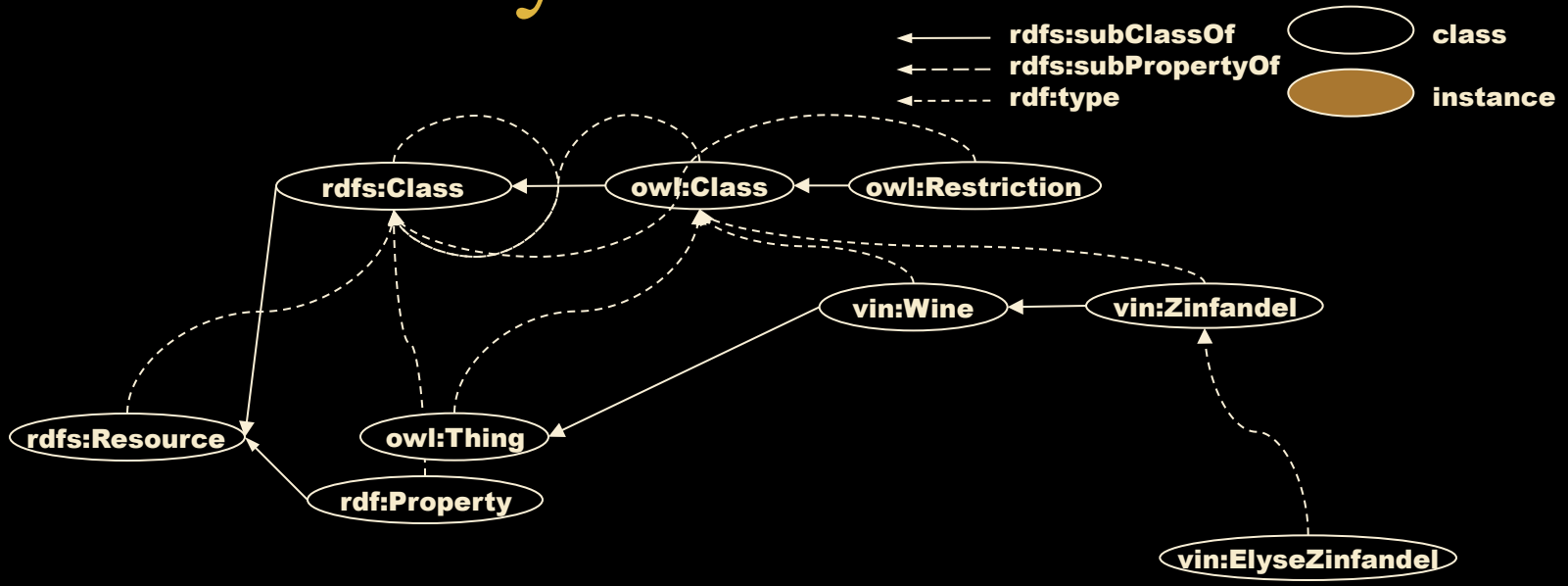


- All restrictions under owl:Restriction are in the domain of rdfs:Resource.

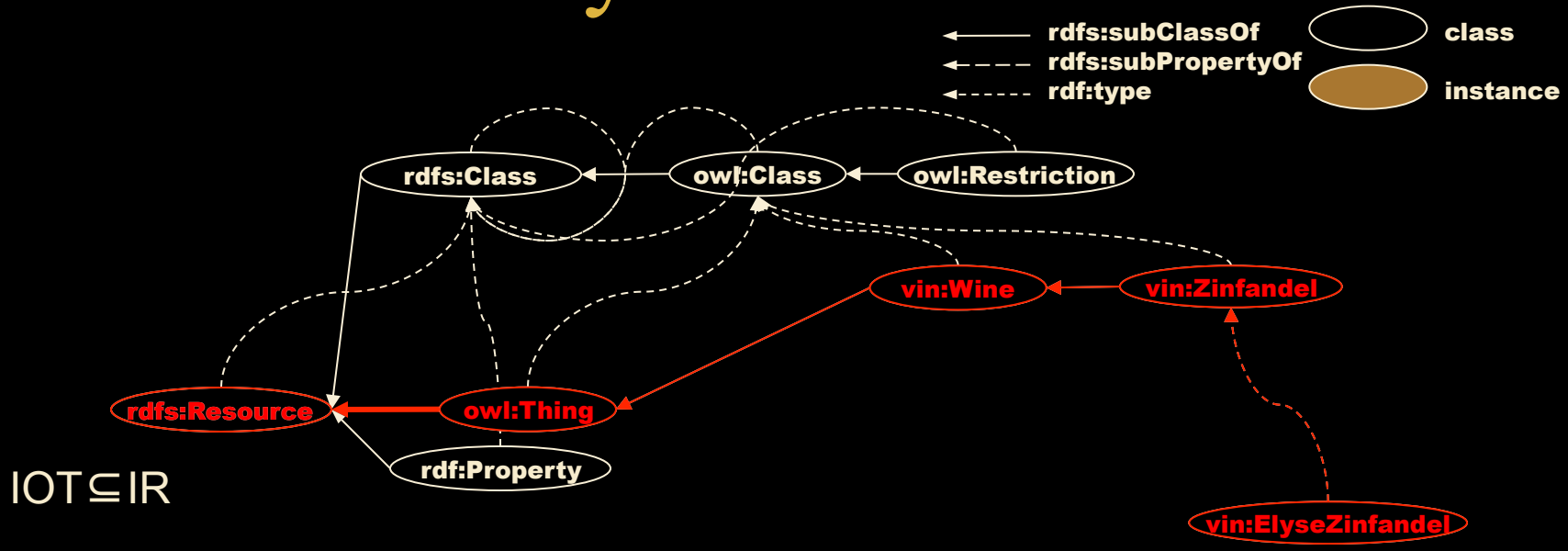
OWL-Full Style Connection



OWL-Full Style Connection



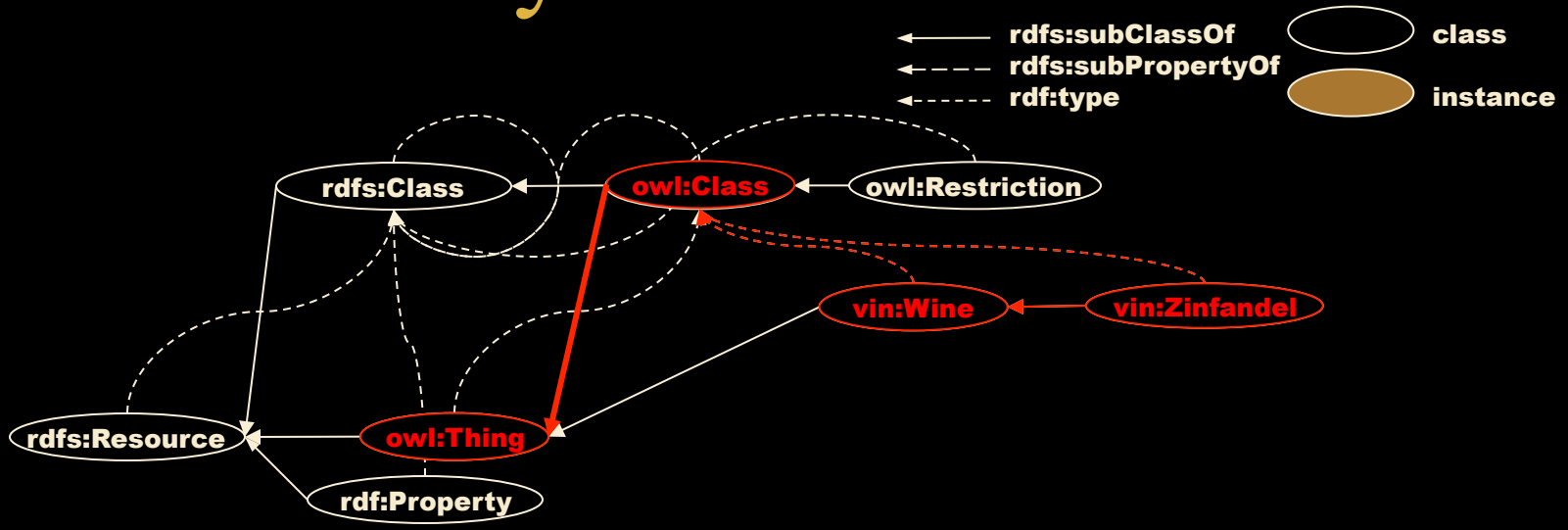
OWL-Full Style Connection



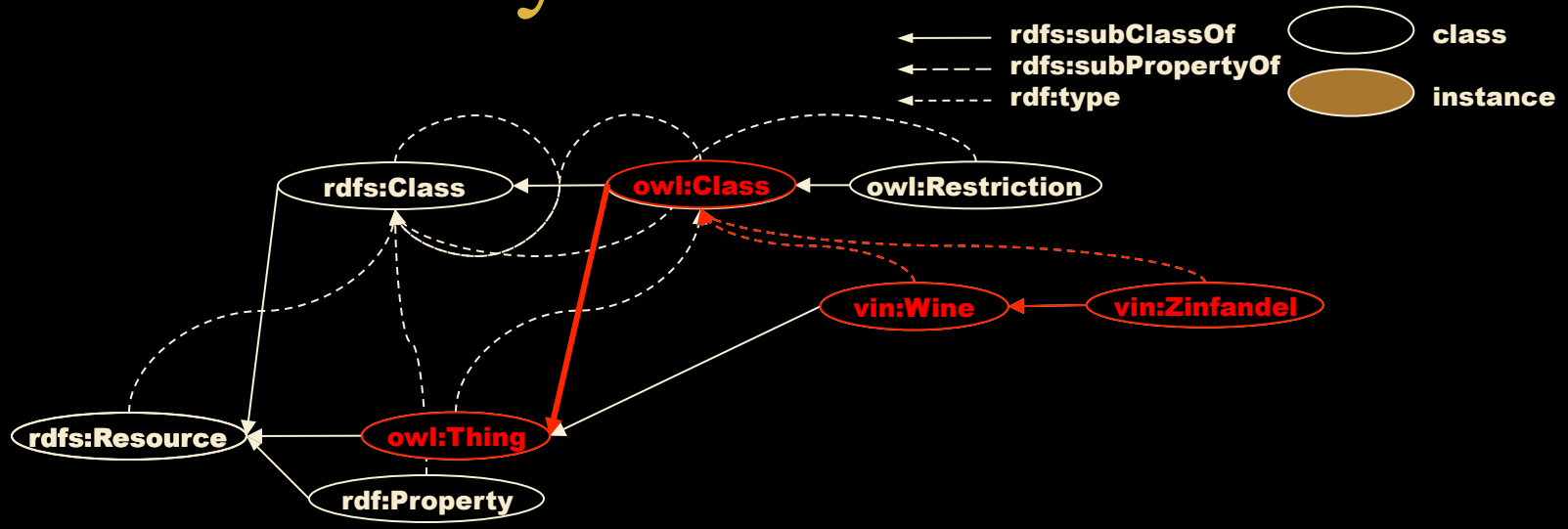
IOT⊆IR

- We need the new axiom to make OWL individuals be in RDFS universe.
  - `owl:Thing rdfs:subClassof rdfs:Resource .`

OWL-Full Style Connection



OWL-Full Style Connection



IOC ⊆IOT

- We need the new axiom to make OWL classes be in OWL universe.
  - `owl:Class rdfs:subClassOf owl:Thing .`

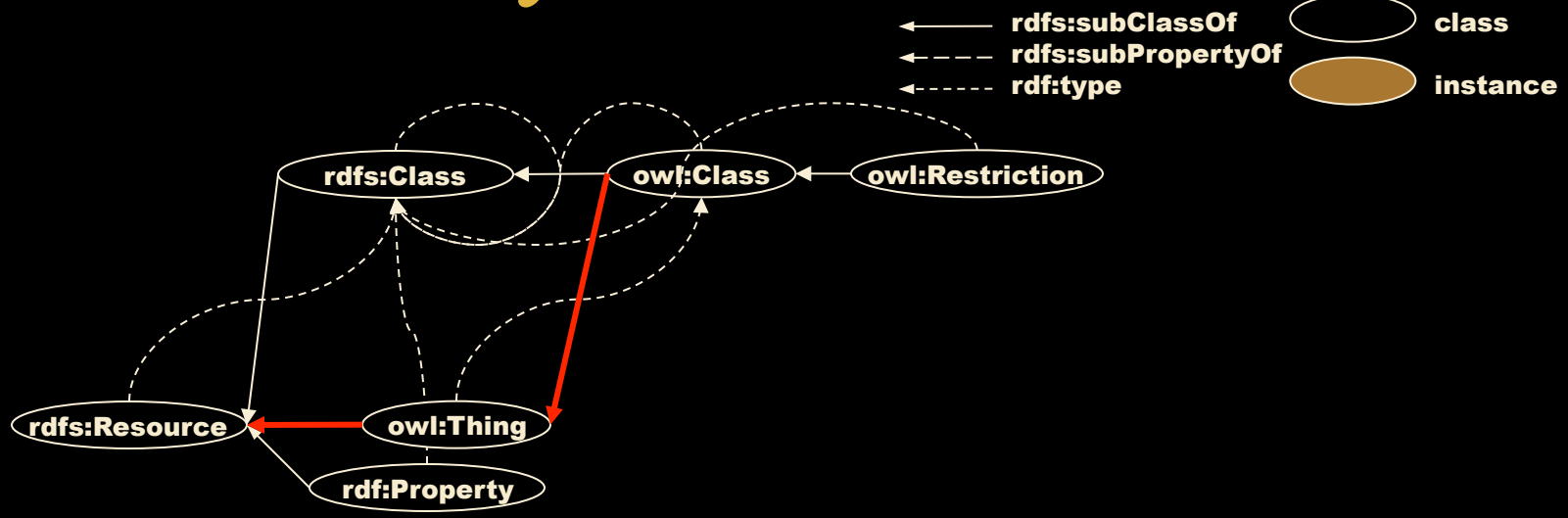
Conditions concerning the parts of the OWL universe and syntactic categories

If E is	then		
	$IS(E) \in$	$ICEXT(IS(E)) =$	and
owl:Class	*IC*	**IOC**	**IOC** $\subseteq$ *IC*
owl:Thing	**IOC**	**IOT**	**IOT** $\subseteq$ *IR* and **IOT** $\neq \varnothing$
owl:Restriction	*IC*	**IOR**	**IOR** $\subseteq$ **IOC**
owl:ObjectProperty	*IC*	**IOOP**	**IOOP** $\subseteq$ *IP*
owl:DatatypeProperty	*IC*	**IODP**	**IODP** $\subseteq$ *IP*
owl:AnnotationProperty	*IC*	**IOAP**	**IOAP** $\subseteq$ *IP*

Conditions concerning the parts of the OWL universe and syntactic categories

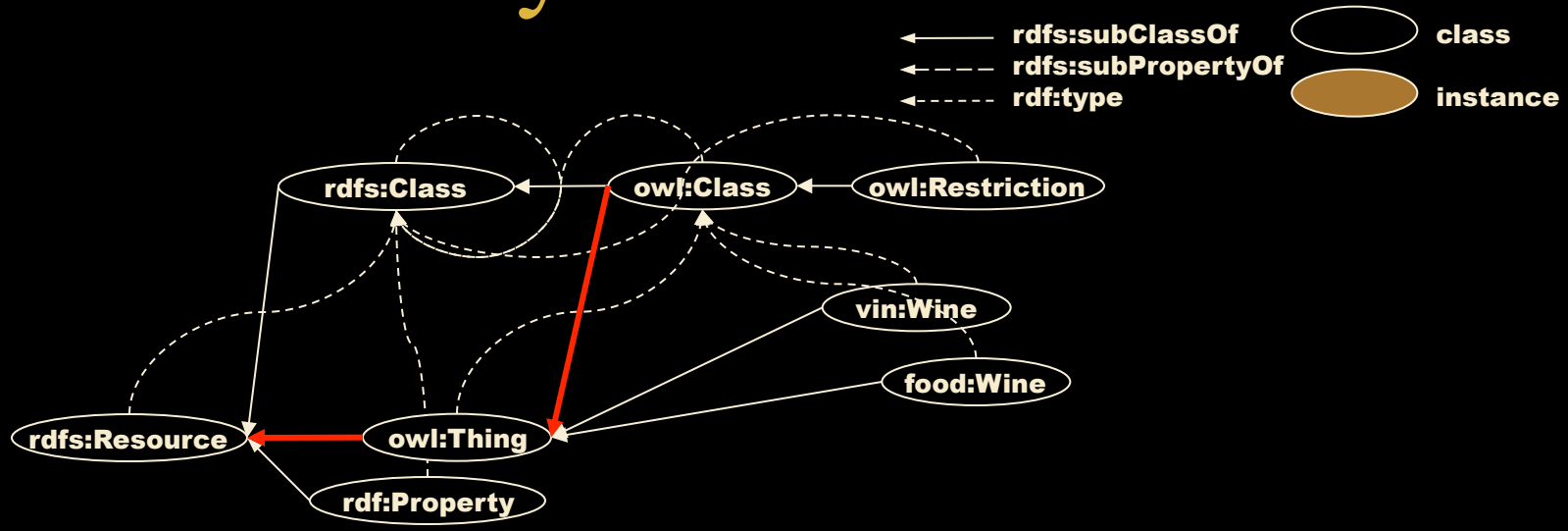
If *E* is	then		
	$IS(E) \in$	$ICEXT(IS(E)) =$	and
owl:Class	*IC*	**IOC**	**IOC** $\subseteq$ **IOT** $\subseteq$ *IC*
owl:Thing	**IOC**	**IOT**	**IOT** $\subseteq$ *IR* and **IOT** $\neq \varnothing$
owl:Restriction	*IC*	**IOR**	**IOR** $\subseteq$ **IOC**
owl:ObjectProperty	*IC*	**IOOP**	**IOOP** $\subseteq$ *IP*
owl:DatatypeProperty	*IC*	**IODP**	**IODP** $\subseteq$ *IP*
owl:AnnotationProperty	*IC*	**IOAP**	**IOAP** $\subseteq$ *IP*

OWL-Full Style Connection



- Classes may be treated as individual.

OWL-Full Style Connection



- Classes may be treated as individual.
  - `vin:Wine owl:sameAs food:Wine .`

Equality in OWL

- Equivalency as Cla
  - owl:equivalentClass, owl:intersectionOf, owl:unionOf, owl:complementOf, owl:oneOf
  - owl:disjointWith, owl:complementOf,
- Equivalency as Individual
  - owl:sameAs, owl:FunctionalProperty, owl:InverseFunctionalProperty
  - owl:differenceFrom, owl:distinctMembers
  - owl:oneOf
- Equivalency as Property
  - owl:equivalentProperty

http://www.w3.org/TR/owl-ref/#DescriptionAxiom

Equivalent Property

http://www.w3.org/TR/owl-ref/#equivalentProperty-def

- Property equivalence is not the same as property equality. Equivalent properties have the same "values" (i.e., the same property extension), but may have different intensional meaning (i.e., denote different concepts). Property equality should be expressed with the owl:sameAs construct. As this requires that properties are treated as individuals, such axioms are only allowed in OWL Full.

Equality in RDF Graph

http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/#section-graph-equality

Two RDF graphs *G* and *G′* are equivalent if there is a bijection *M* between the sets of nodes of the two graphs, such that:

- *M* maps blank nodes to blank nodes.
- *M(lit)=lit* for all RDF literals *lit* which are nodes of *G*.
- *M(uri)=uri* for all RDF URI references *uri* which are nodes of *G*.
- The triple *(s, p, o)* is in *G* if and only if the triple *(M(s), p, M(o))* is in *G′*.

With this definition, *M* shows how each blank node in *G* can be replaced with a new blank node to give *G′*.

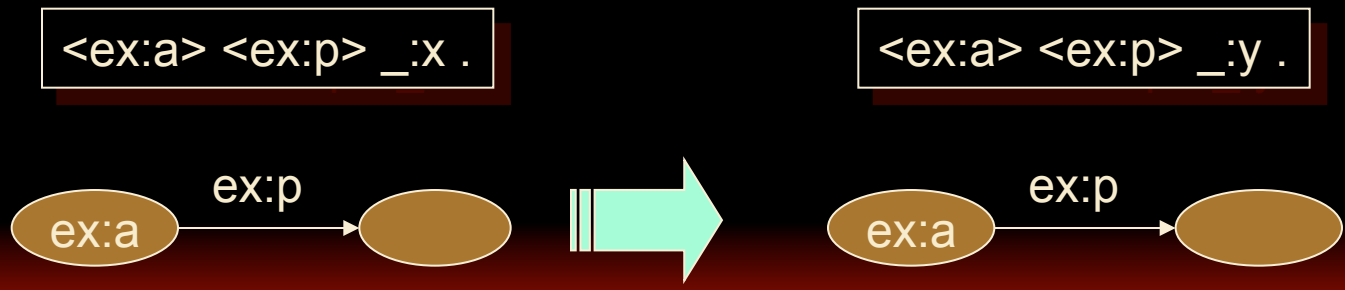
Non-Unique Name Assumption and Equality

- In Semantic Webs, two or more different names or URIs may denote same object.
- Negative identity of two or more graphs and nodes does not mean the difference.

Graph Equality in RDF

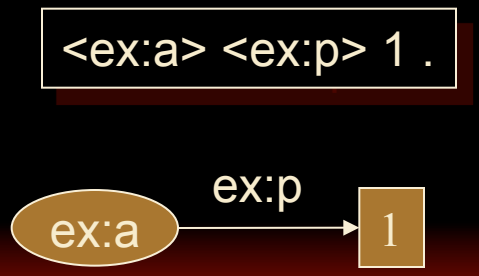
http://www.w3.org/TR/rdf-mt/#graphdefs

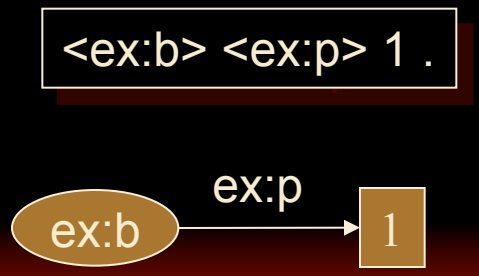
- Any instance of a graph in which a blank node is mapped to a new blank node not in the original graph is an instance of the original and also has it as an instance, and this process can be iterated so that any 1:1 mapping between blank nodes defines an instance of a graph which has the original graph as an instance. Two such graphs, each an instance of the other but neither a proper instance, which differ only in the identity of their blank nodes, are considered to be equivalent. We will treat such equivalent graphs as identical; this allows us to ignore some issues which arise from 're-naming' nodeIDs, and is in conformance with the convention that blank nodes have no label. Equivalent graphs are mutual instances with an invertible instance mapping.



Non-Unique Name Assumption and Equality

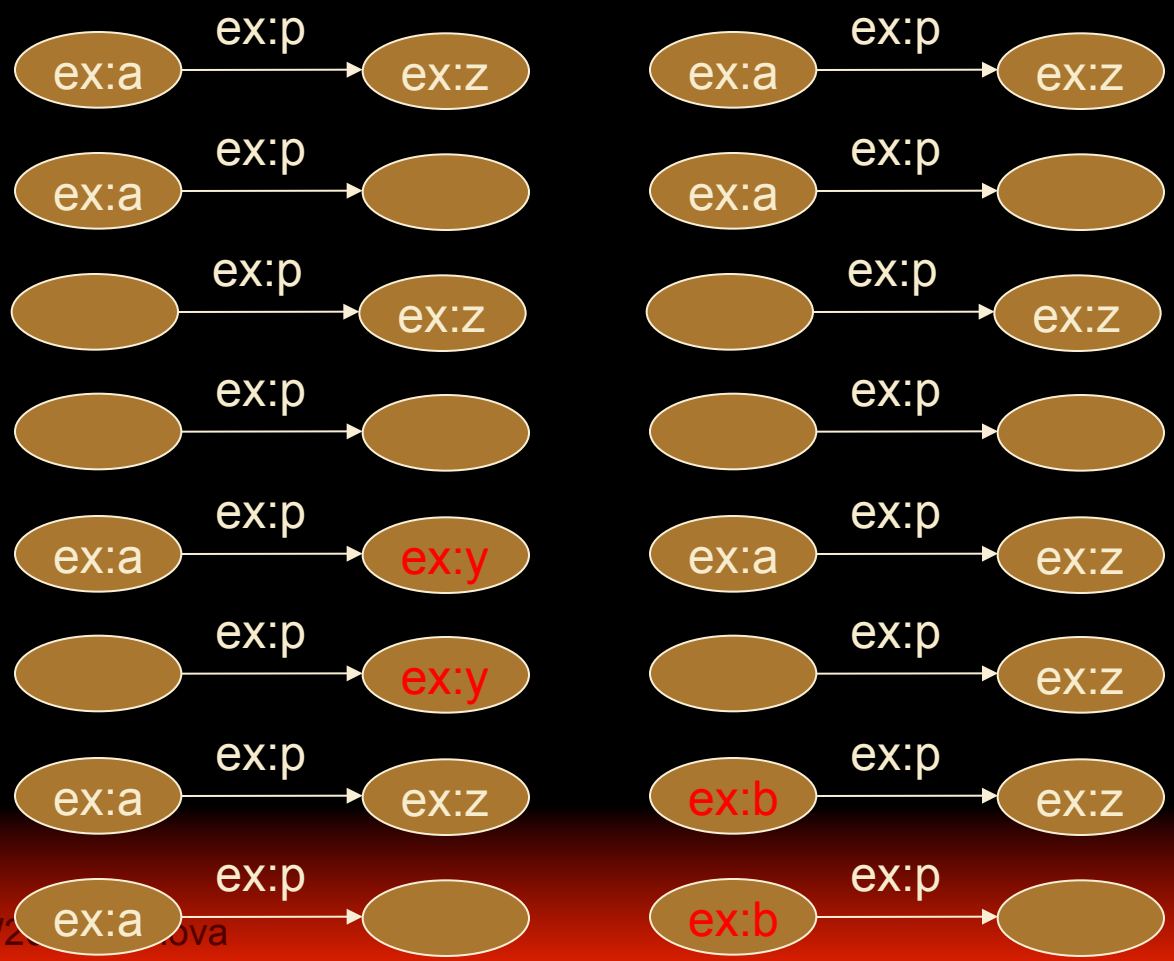
- ex:a and ex:b may be identical (denoted by different URIs) in non-UNA.
- Two graphs that show different appearance may be identical.





Non-Unique Name Assumption and Equality

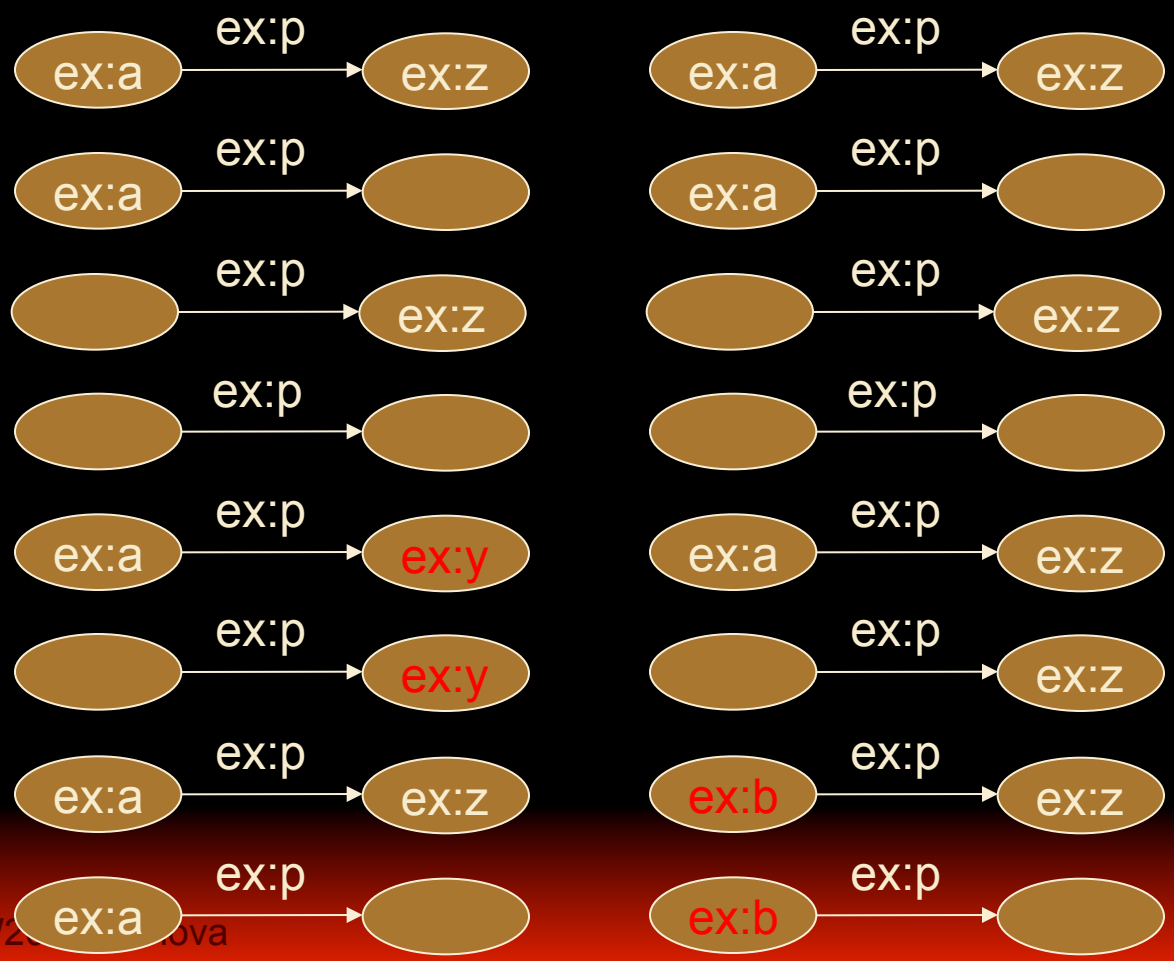
in UNA



	equal?	different?
ex:a —ex:p→ ex:z ; ex:a —ex:p→ ex:z	Yes.	No.
ex:a —ex:p→ ; ex:a —ex:p→	Yes.	No.
—ex:p→ ex:z ; —ex:p→ ex:z	Yes.	No.
—ex:p→ ; —ex:p→	Yes.	No.
ex:a —ex:p→ ex:y ; ex:a —ex:p→ ex:z	No.	Yes.
—ex:p→ ex:y ; —ex:p→ ex:z	No.	Yes.
ex:a —ex:p→ ex:z ; ex:b —ex:p→ ex:z	No.	Yes.
ex:a —ex:p→ ; ex:b —ex:p→	No.	Yes.

Non-Unique Name Assumption and Equality

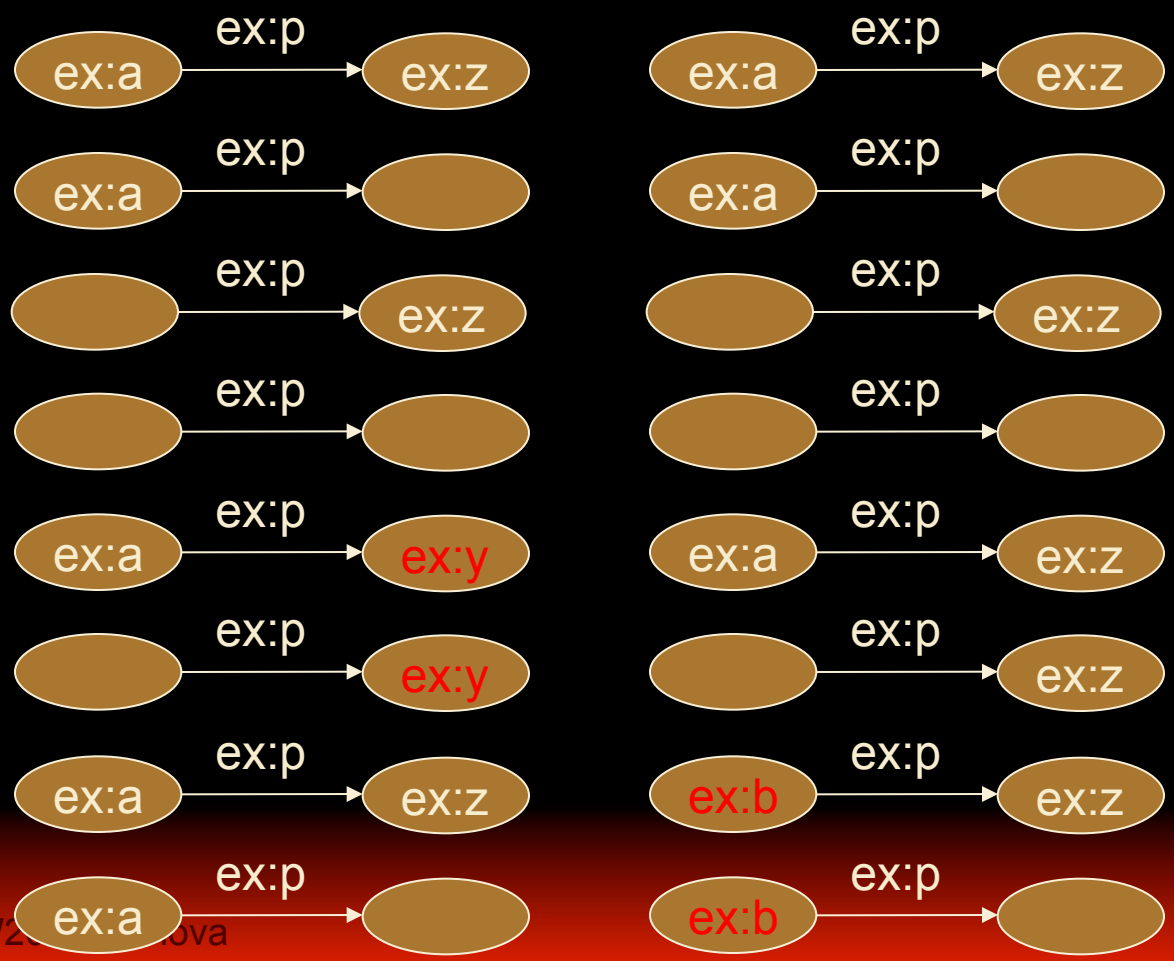
in non-UNA



equal?	different?
Yes.	No.
Yes.	No.
Yes.	No.
Yes.	No.
?	?
?	?
?	?
?	?

Non-Unique Name Assumption and Equality

Atomic UNA in non-UNA

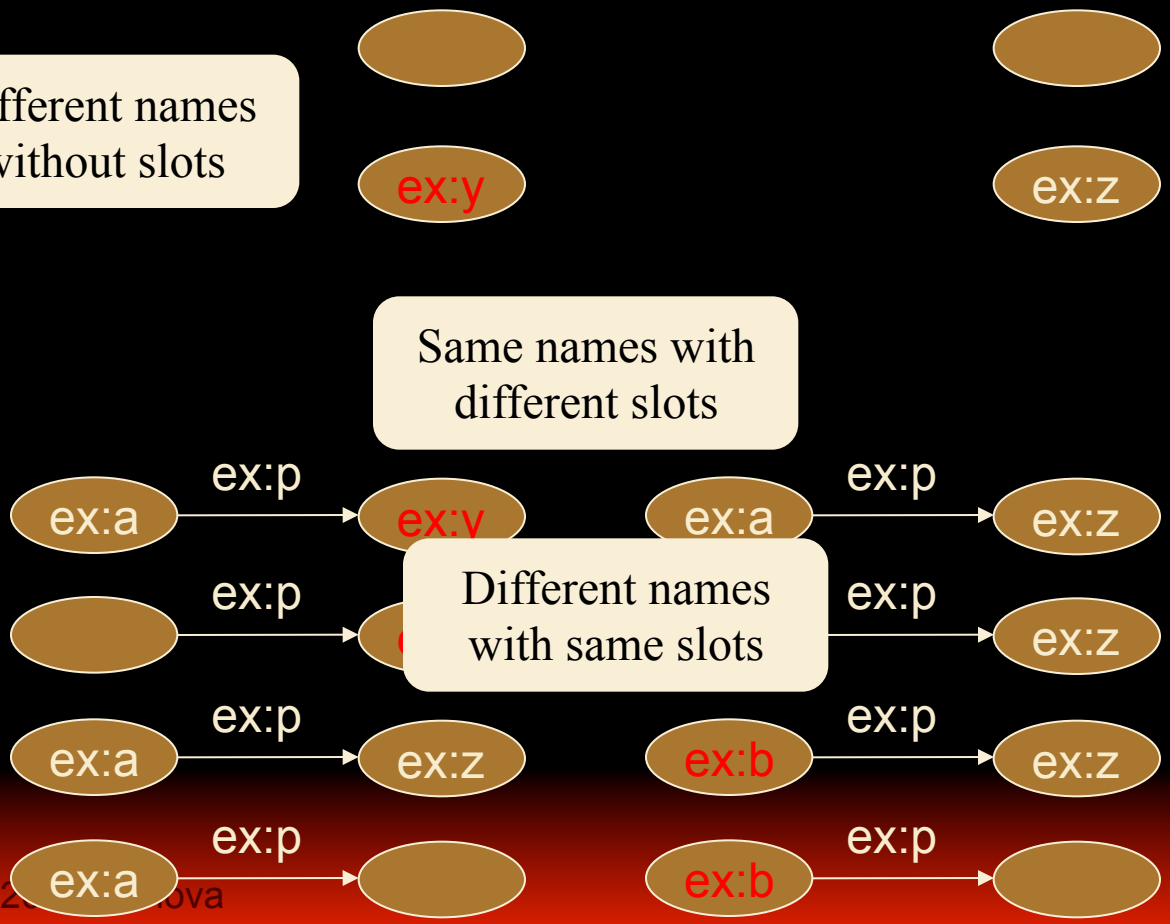


equal?	different?
Yes.	No.
Yes.	No.
Yes.	No.
Yes.	No.
No.	Yes.
No.	Yes.
?	?
?	?

Non-Unique Name Assumption and Equality

Atomic UNA in non-UNA

	equal?	different?
Different names without slots	Yes.	No.
	No.	Yes.
Same names with different slots	No.	Yes.
Different names with same slots	No.	Yes.
	?	?
	?	?



Non-Unique Name Assumption and Equality

- Two atomic objects must be different if the two have different names, because they are basics of composite objects.
- If slots of two objects are different, then different.
  - intensional logics
- If slots of two objects are same, then same in non-UNA?

Auto Epistemic Local Closed World

- Open World is troublesome.
  - owl:someValuesFrom restriction entails very few entailments.
    - It may be satisfiable somewhere you do not know.
- After ontology building, agents want to infer the world which they know.
- User-Settable Closed World
  - owl:intersectionOf, owl:unionOf, owl:one of
  - owl:allValesFrom + owl:maxCardinality

Russell&Norvig AIMA

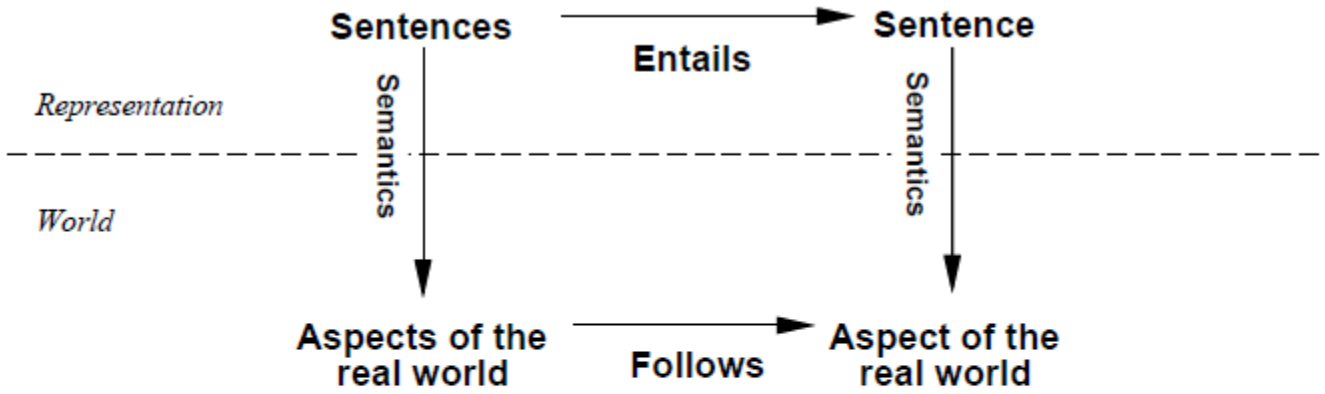
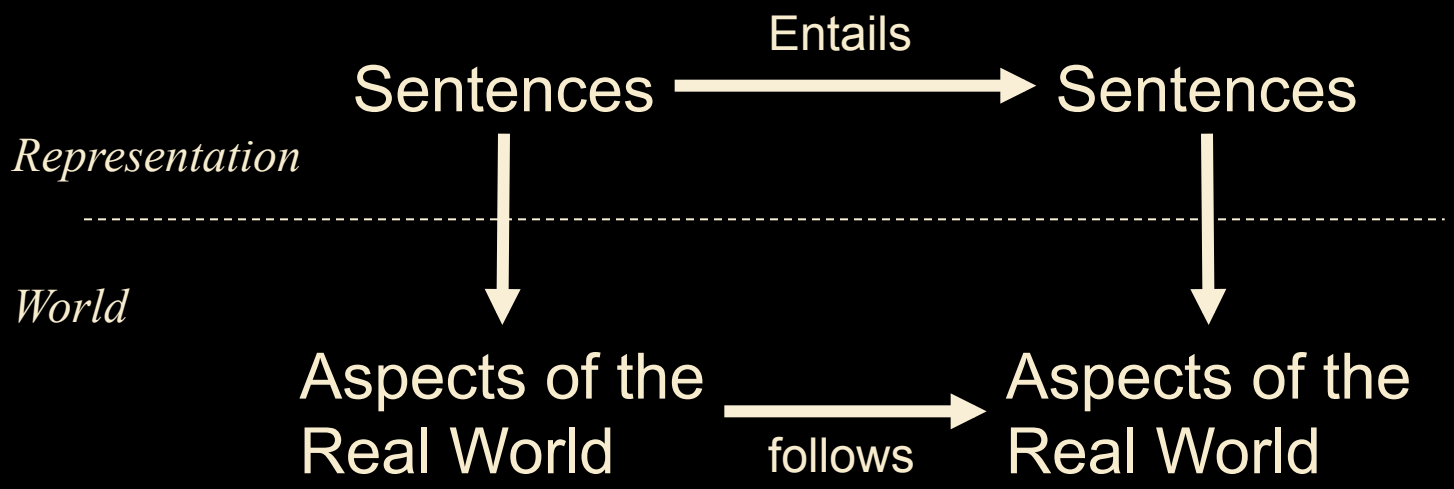
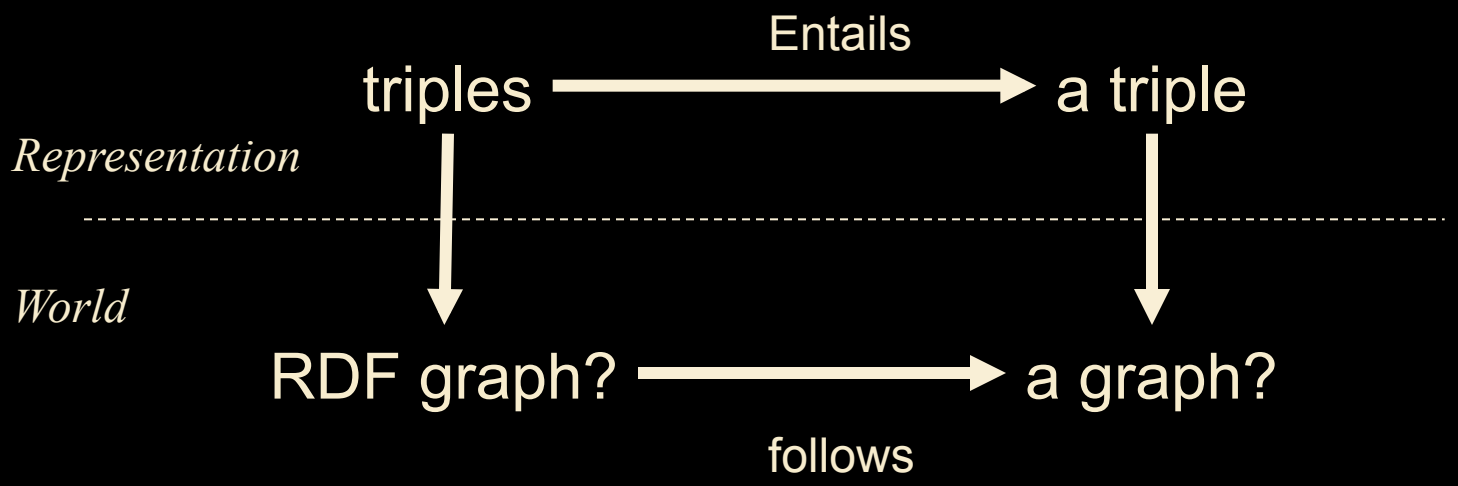


Figure 7.6 Sentences are physical configurations of the agent, and reasoning is a process of constructing new physical configurations from old ones. Logical reasoning should ensure that the new configurations represent aspects of the world that actually follow from the aspects that the old configurations represent.

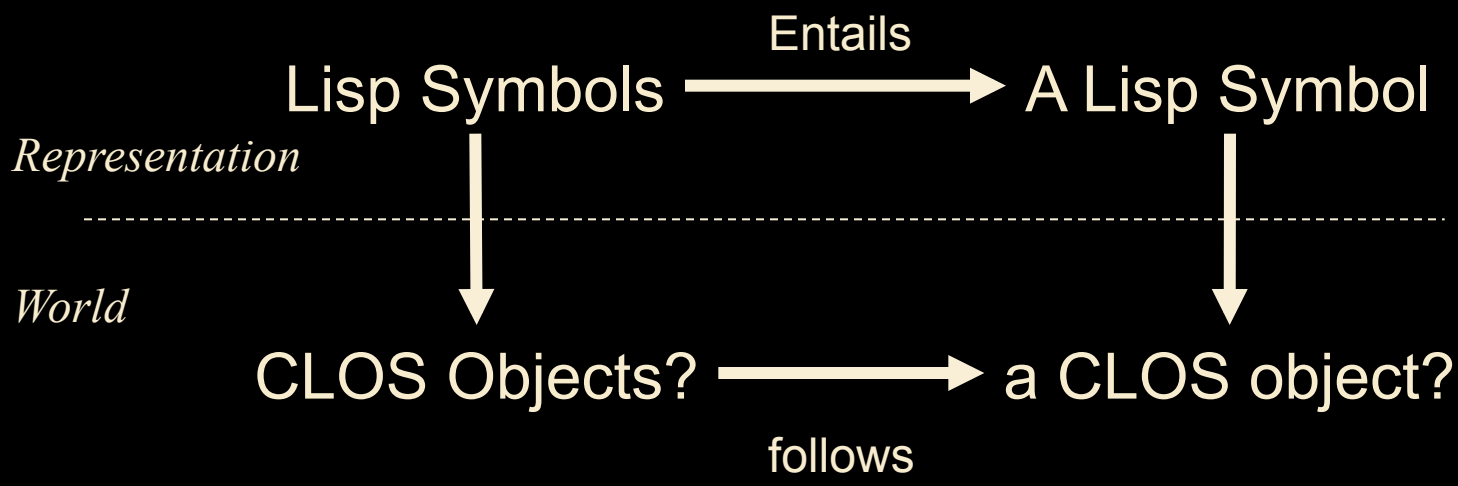
Formal Semantics



Formal Semantics



Formal Semantics



Semantics in Lisp and RDF

based on Brian Cantwell Smith
1983

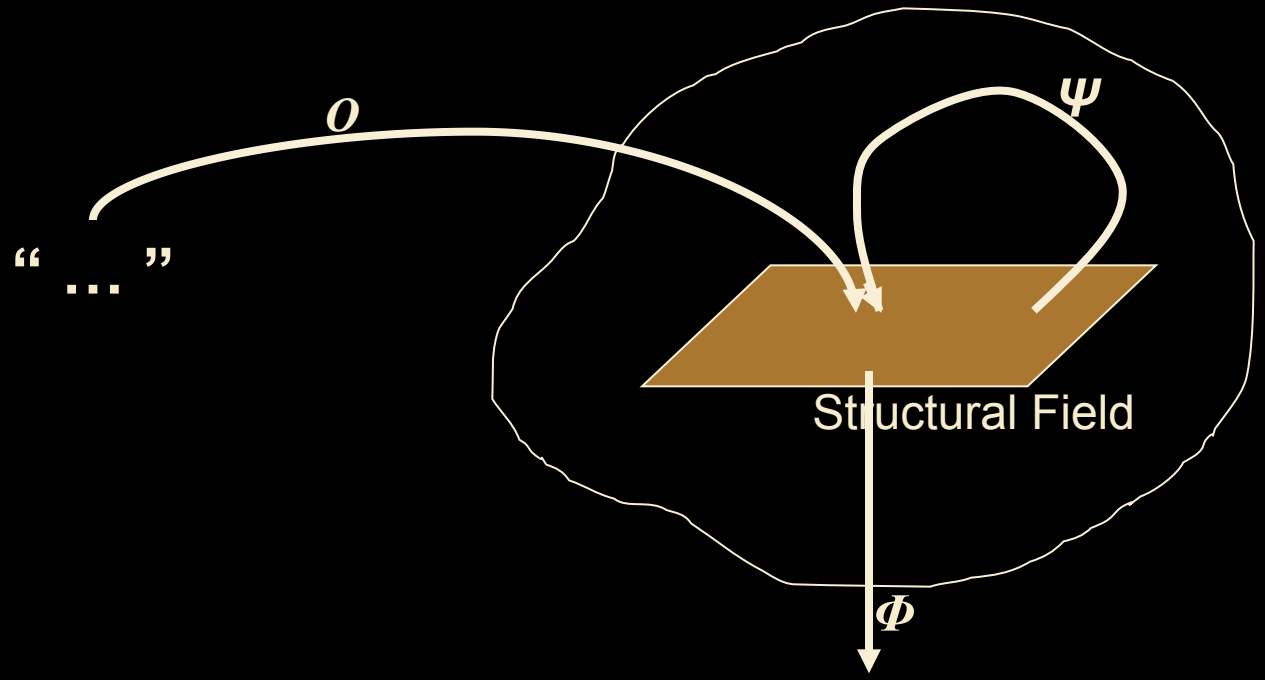
A Simple Semantic Interpretation Function

symbol ***s***
sign ***s***

1. The **objects and events in the world** in which a computational process is embedded, including both real-world objects like *cars* and *caviar*, and set-theoretic abstractions like numbers and functions.
2. The **internal elements, structures, or processes inside the computer**, including *data structures*, *program representations*, *execution sequences* and so forth.
3. The **notational or communicational expressions**, in some externally observable and consensually established medium of interaction, such as *strings of characters*, *streams of words*, or *sequences of display images* on a computer terminal.

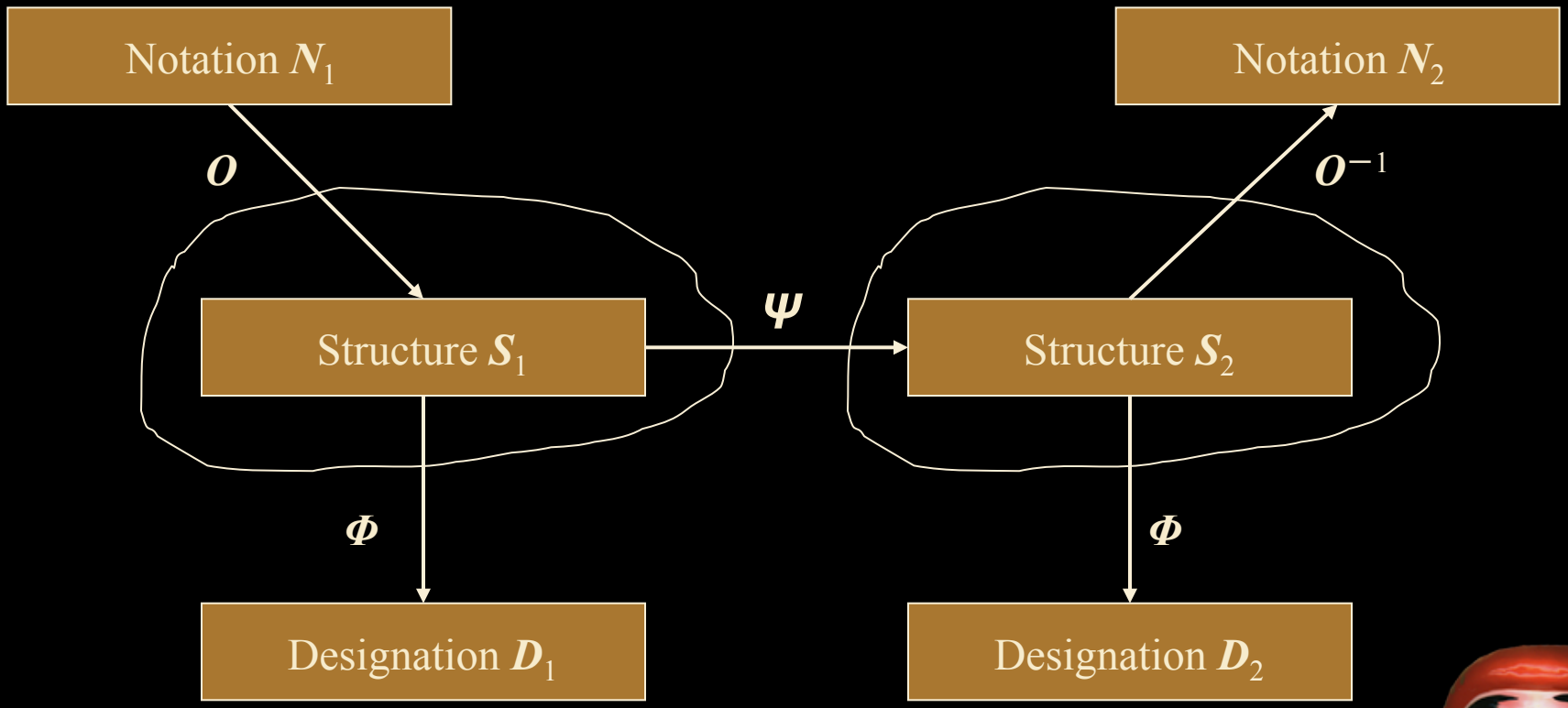
significance ***d***
interpretation of ***s***

Semantic Relationships in a Computational Process

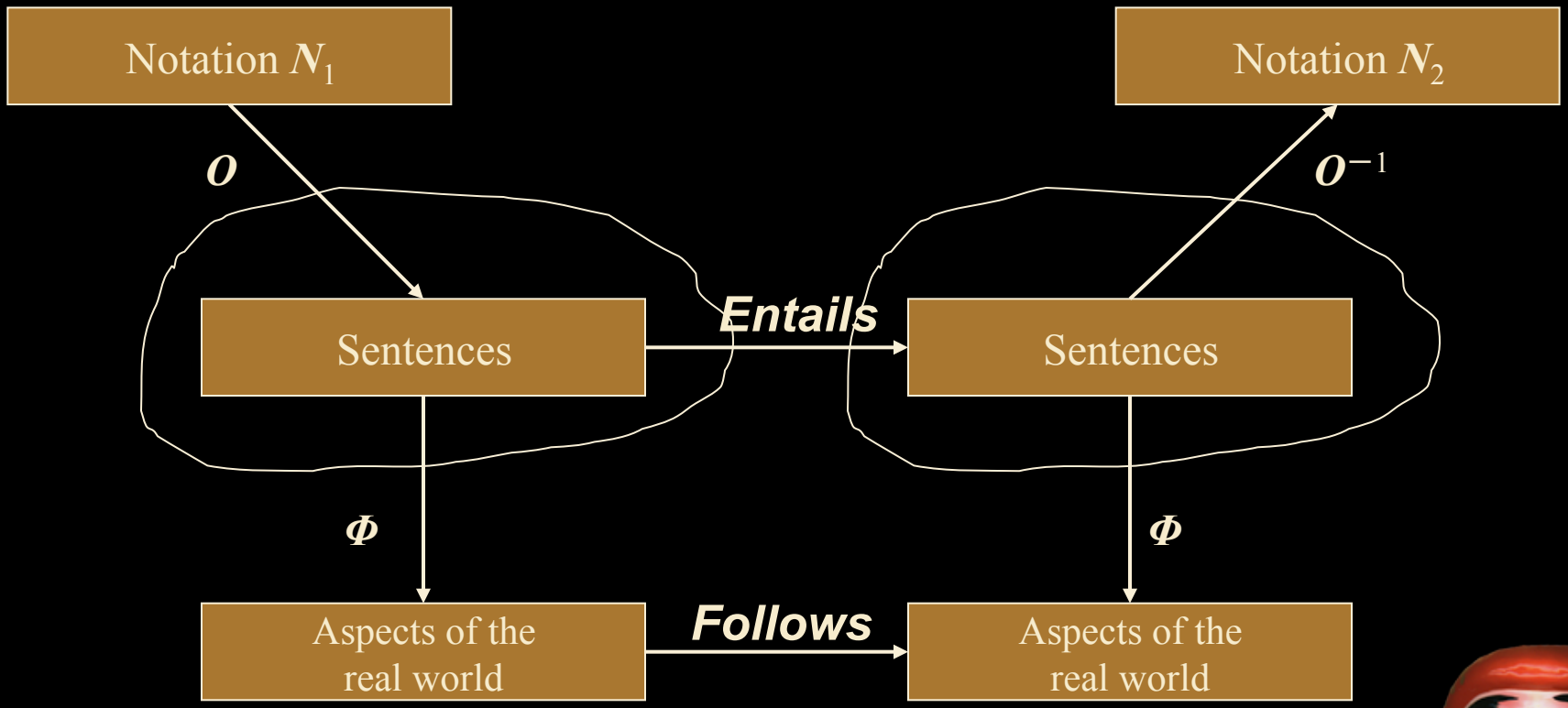


$\Phi(O(\text{“T.S.Eliot”}) = T.S.Eliot$

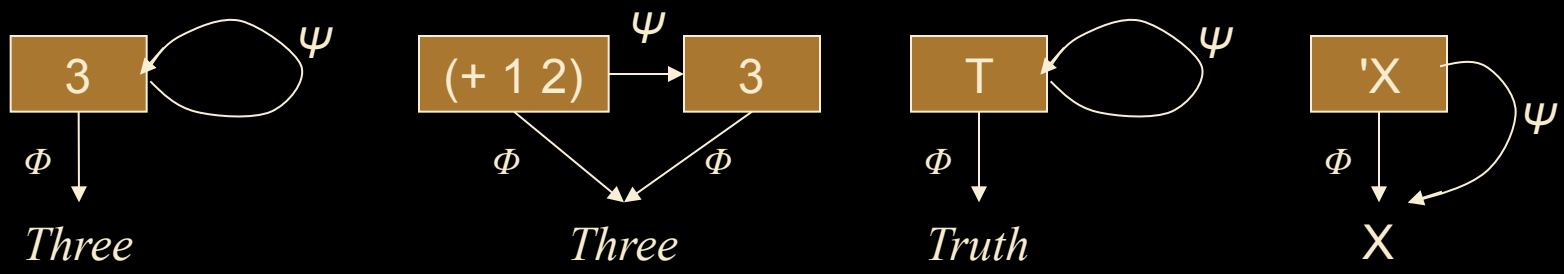
A Framework for Computational Semantics

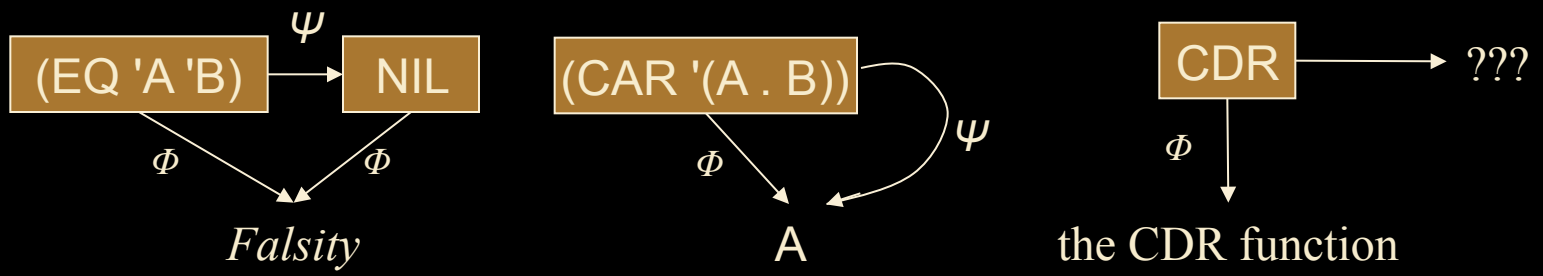


Russell&Norvig AIMA

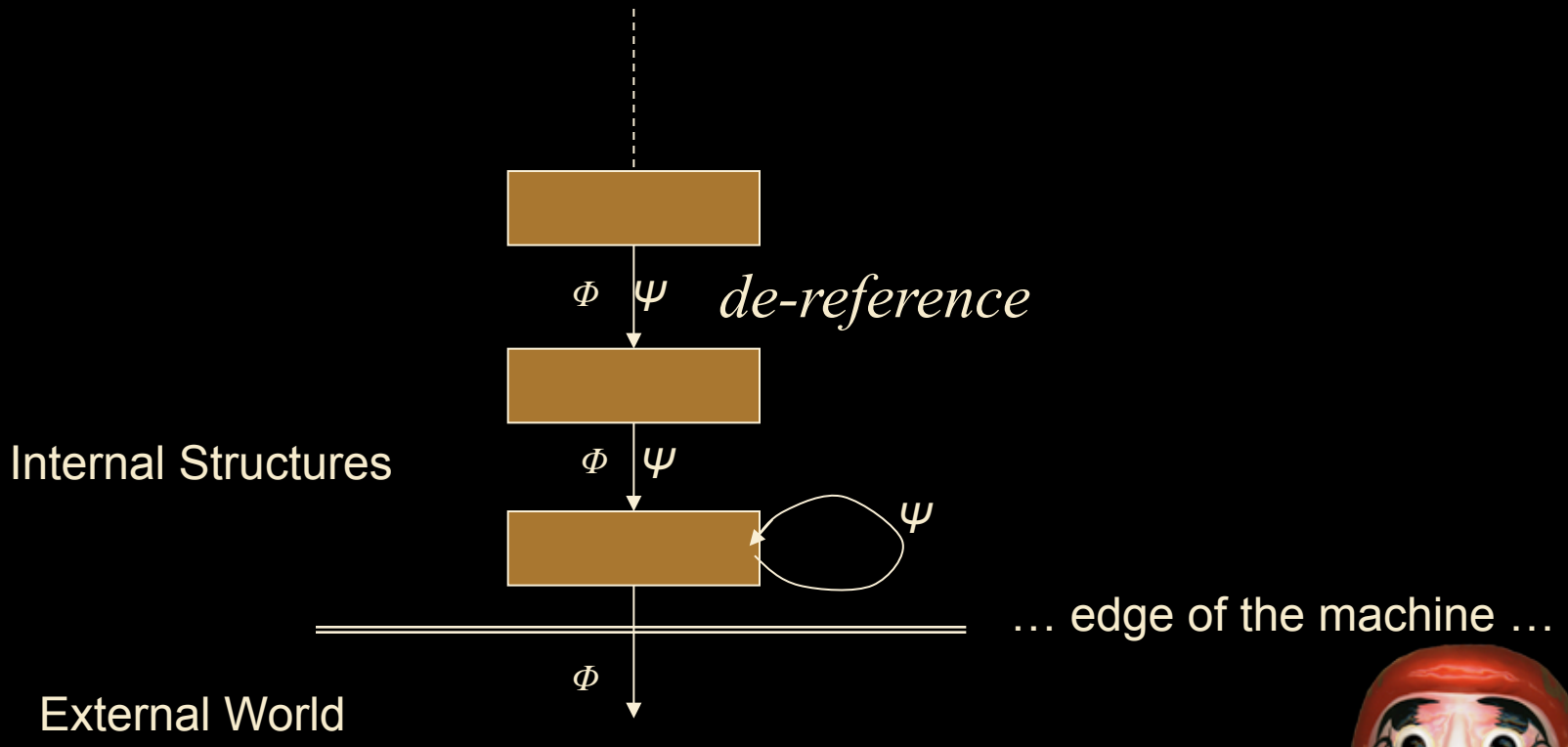


LISP Evaluation vs. Designation

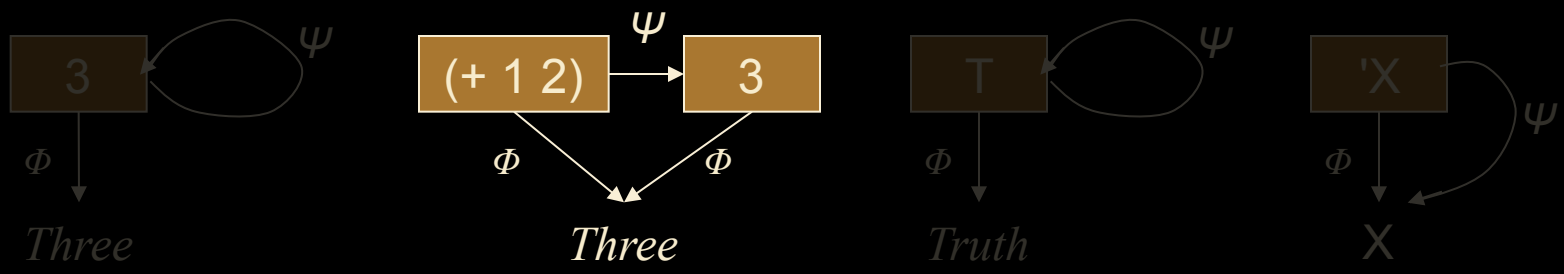


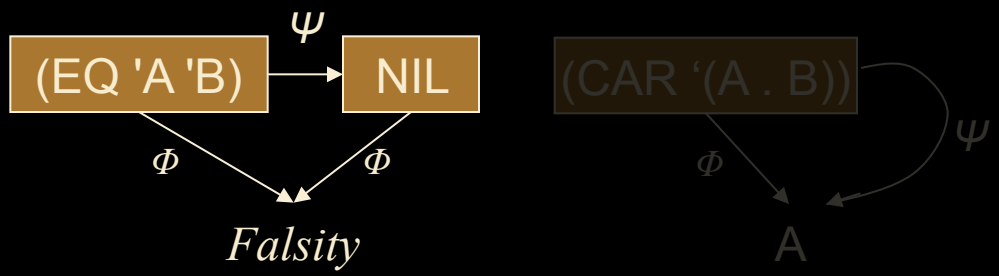


LISP's *"De-reference If You Can"* Evaluation Protocol

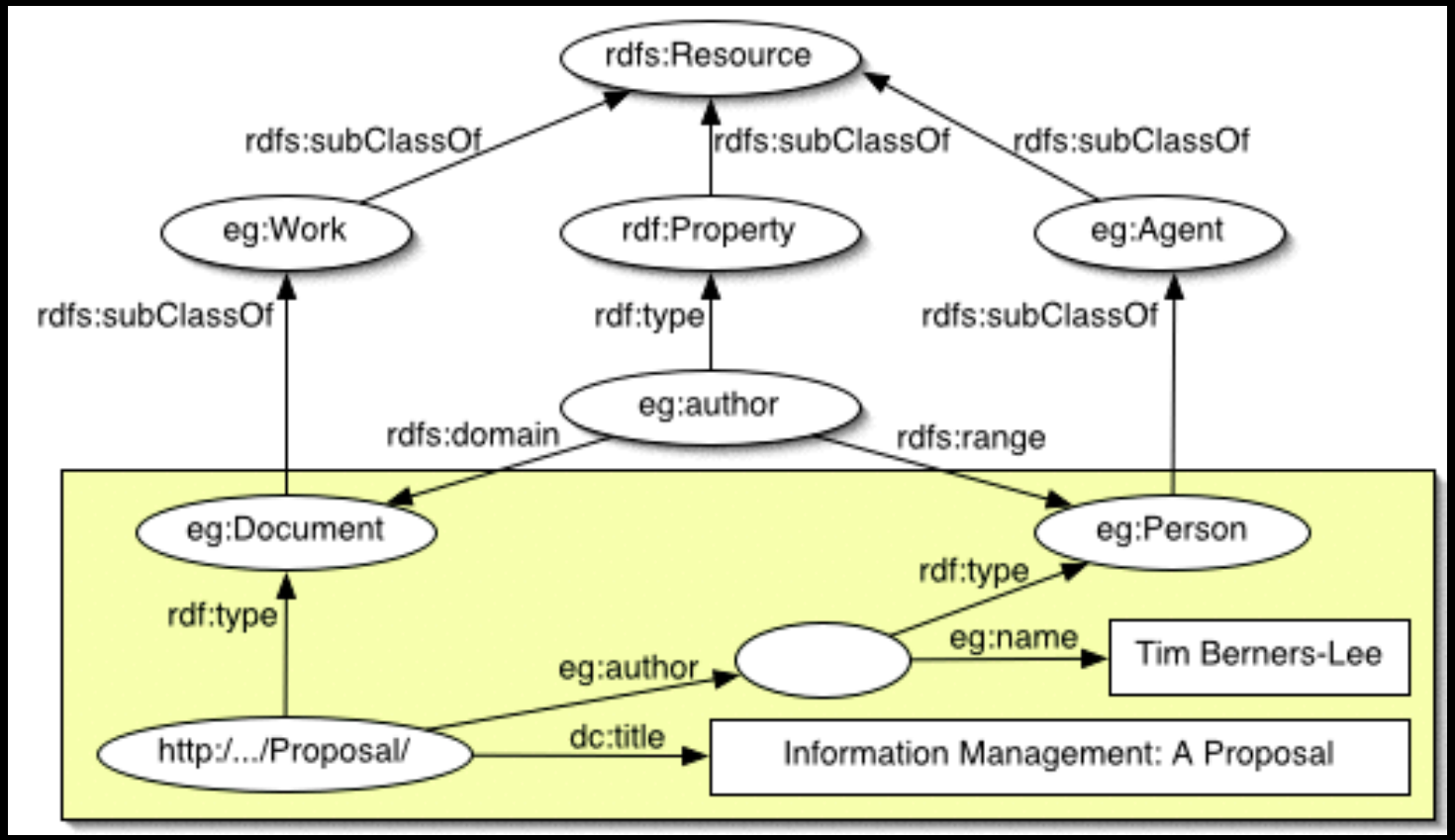


LISP Evaluation vs. Designation



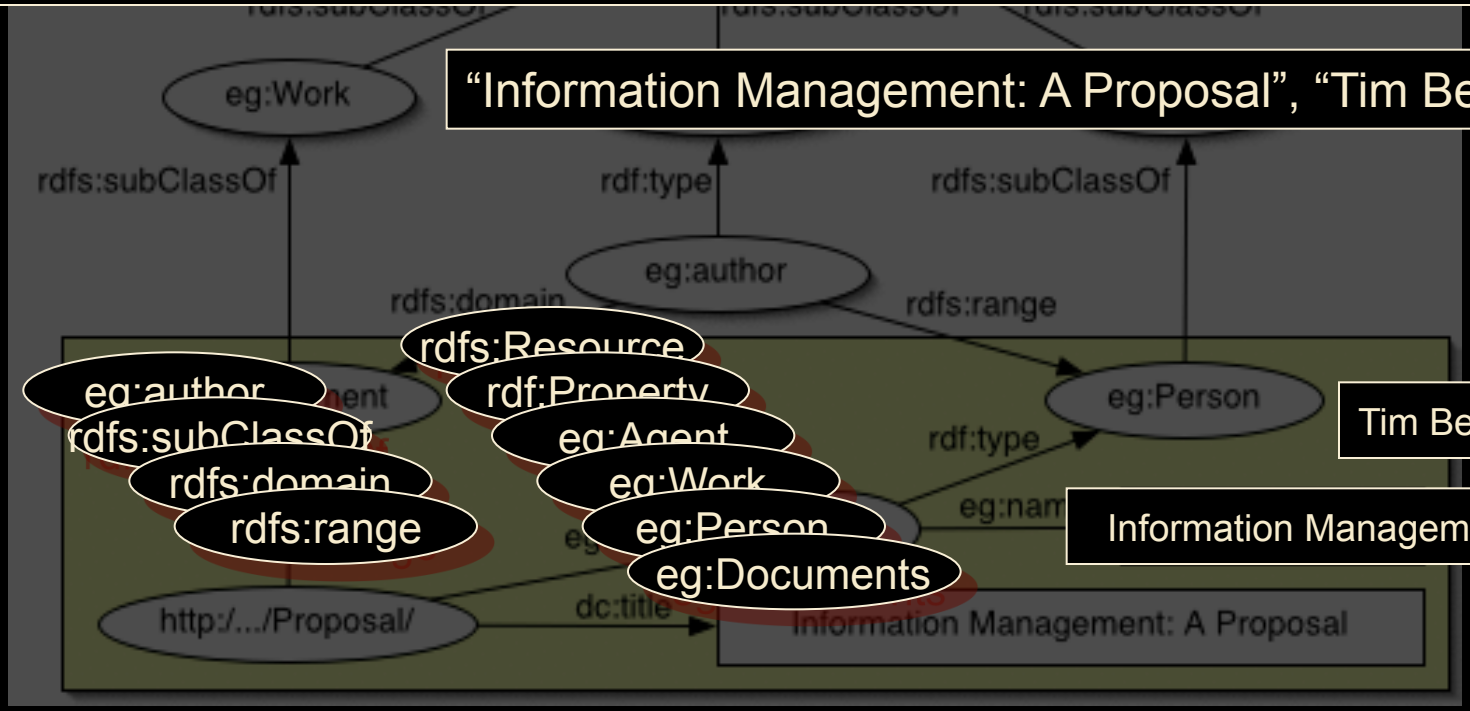


Model Theoretic Semantics of RDF



Model Theoretic Semantics of RDF

rdfs:Resource, rdf:Property, rdfs:subClassOf, rdfs:domain, rdfs:range
eg:Agent, eg:Work, eg:author, eg:Person, eg:Documents



"Information Management: A Proposal", "Tim Berners-Lee"

Tim Berners-Lee

Information Management: A Proposal

Model Theoretic Semantics of RDF

URI References

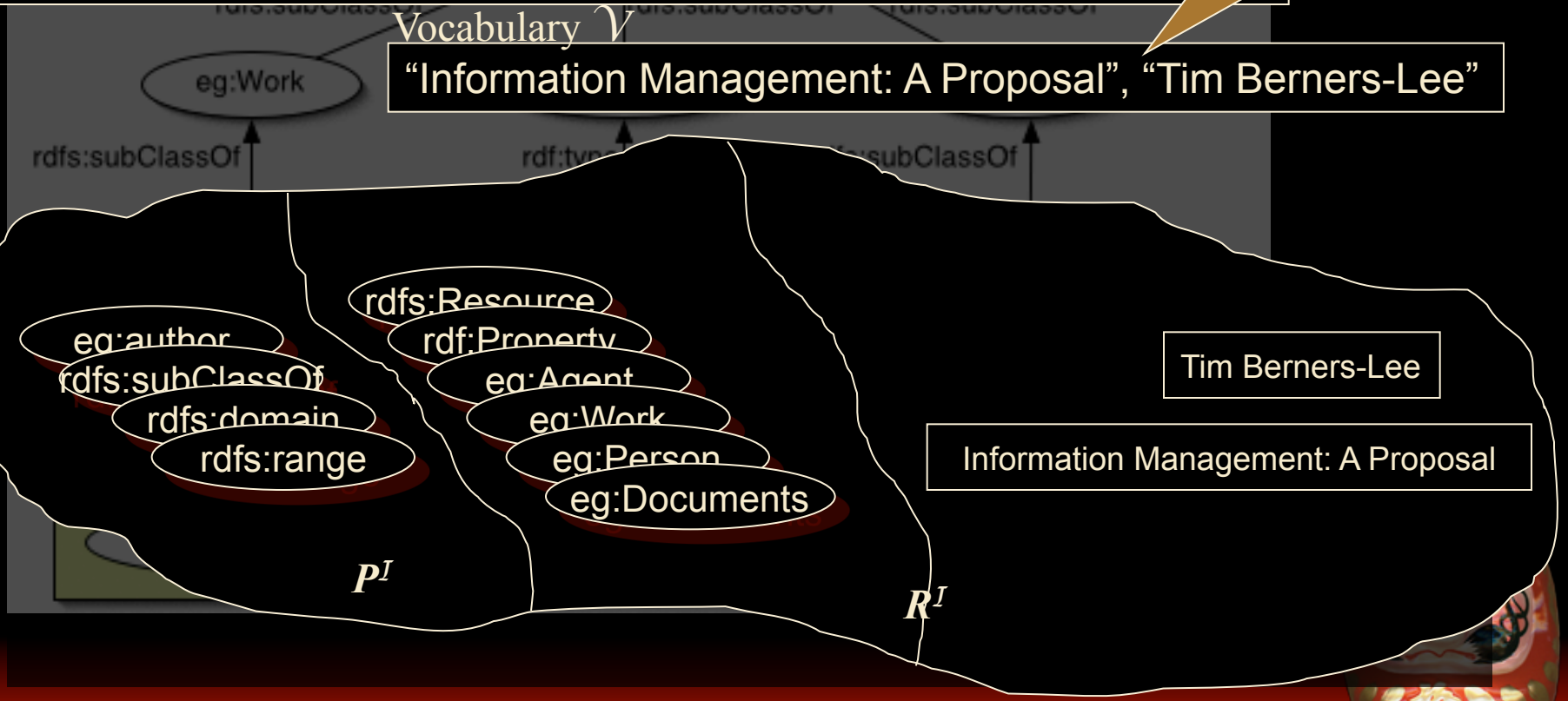
Vocabulary $\mathcal{V}$

rdfs:Resource, rdf:Property, rdfs:subClassOf, rdfs:domain, rdfs
eg:Agent, eg:Work, eg:author, eg:Person, eg:Documents

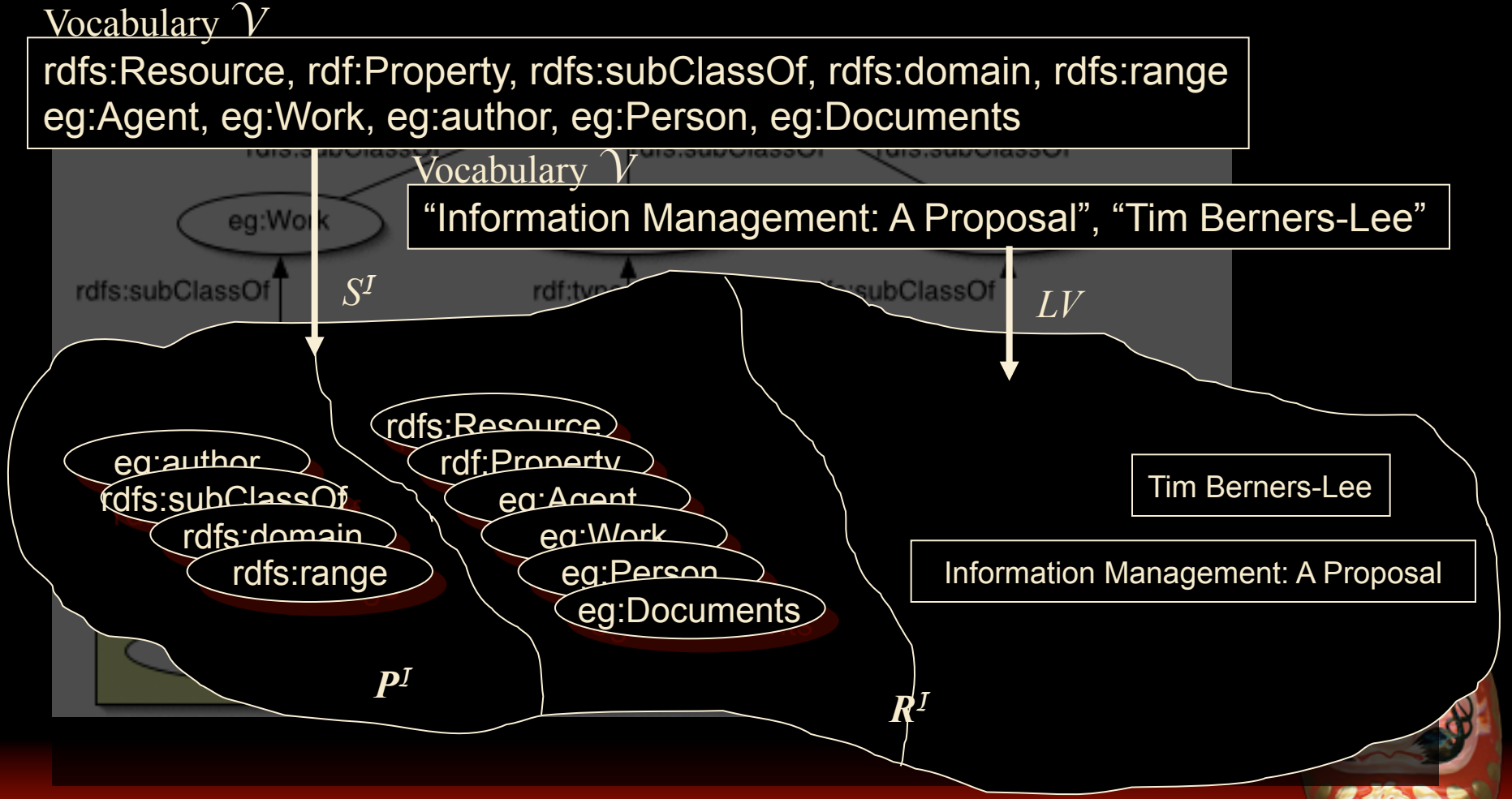
Plane Literals

Vocabulary $\mathcal{V}$

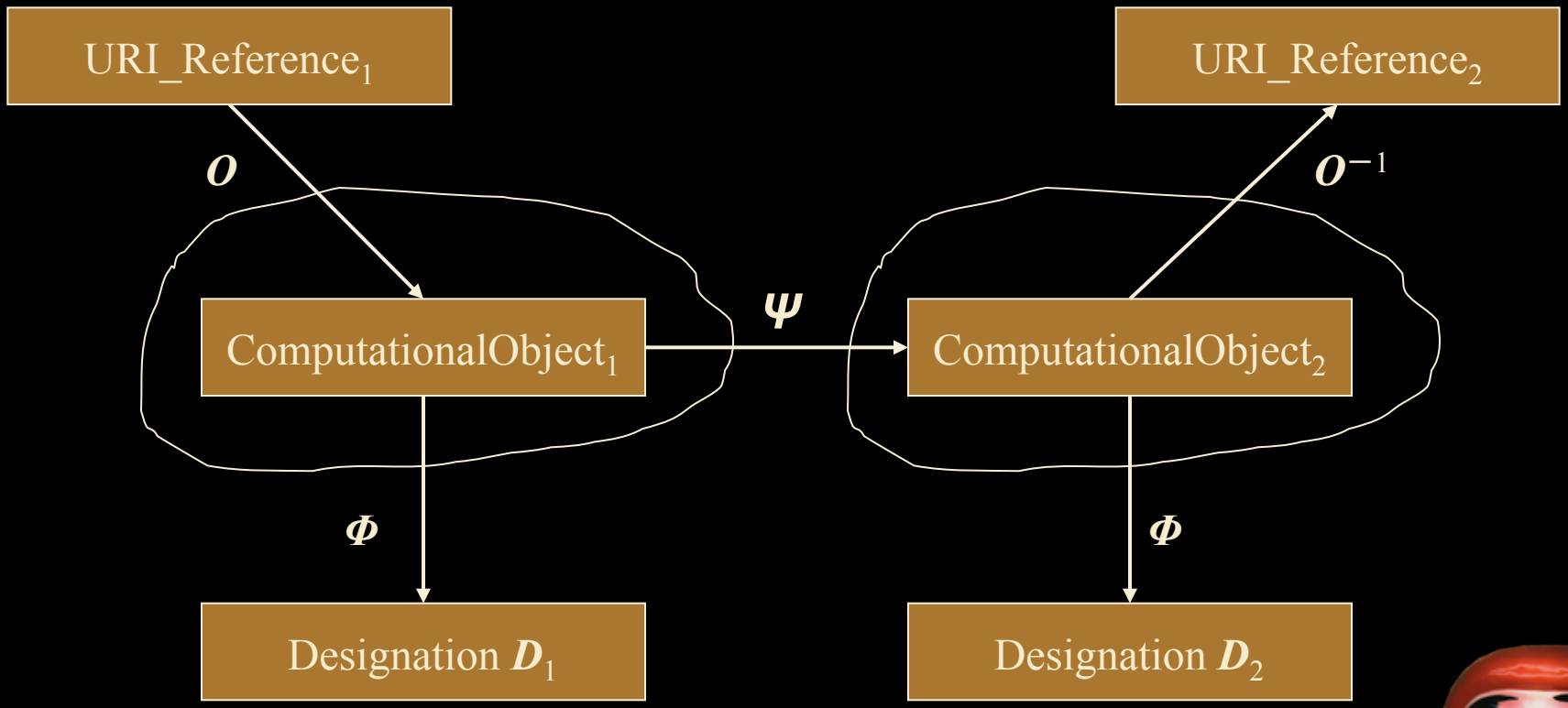
"Information Management: A Proposal", "Tim Berners-Lee"



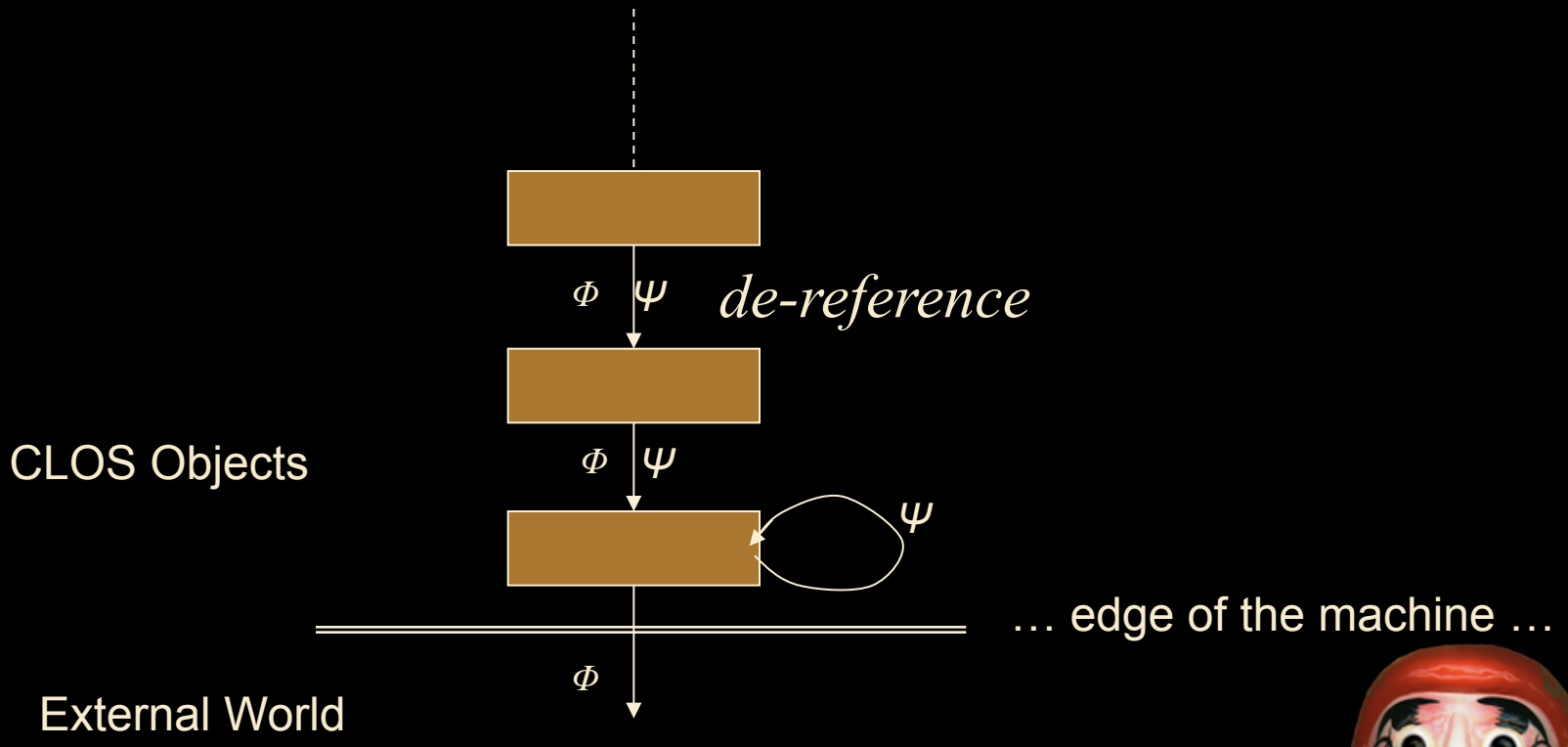
Model Theoretic Semantics of RDF



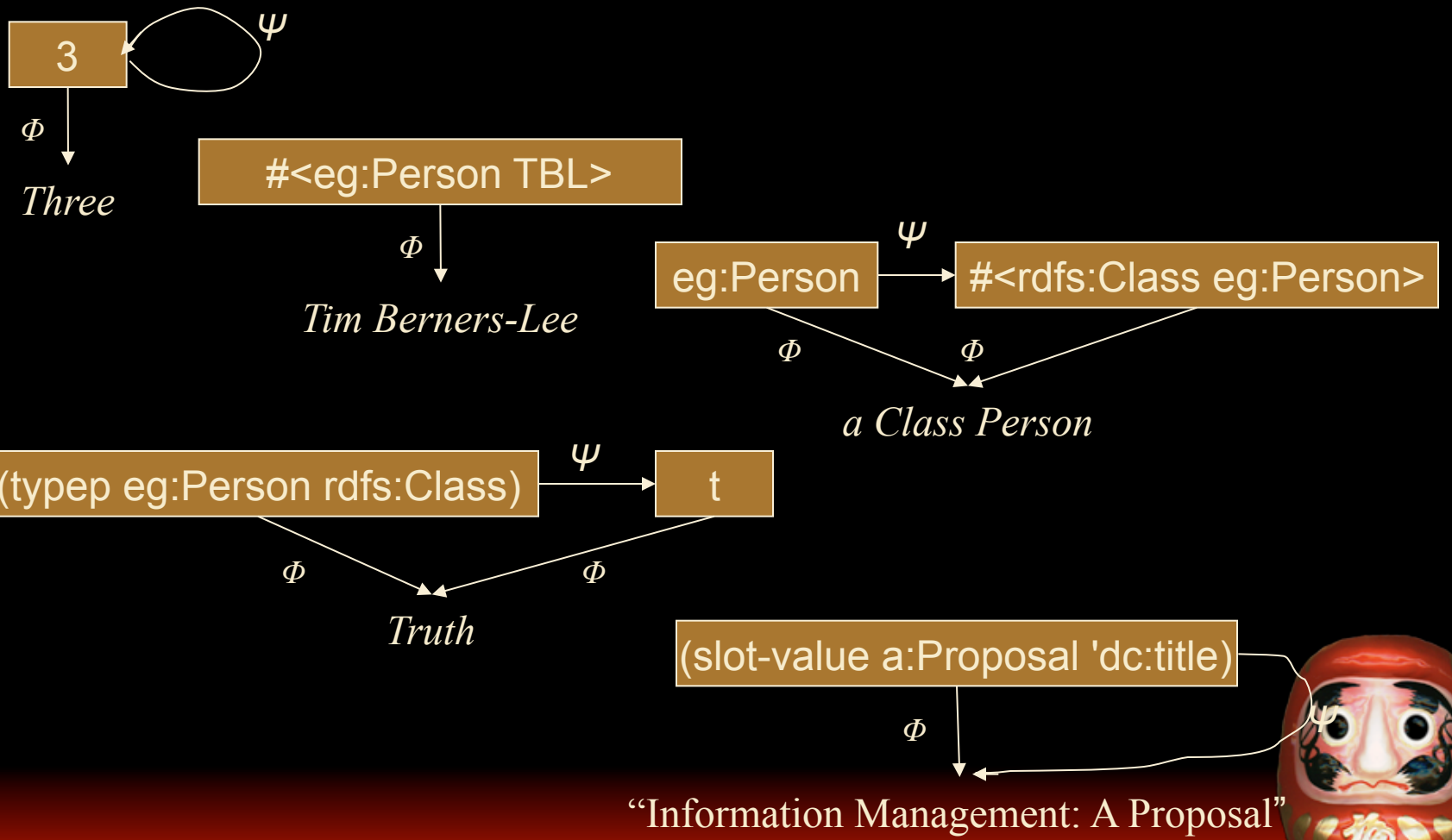
A Framework for Computational Semantics for RDF



SWCLOS's *"De-reference If You Can"* Evaluation Protocol



Evaluation vs. Designation in SWCLOS



Continue to the paper