Introduction and Terms

Lynne Plettenberg
Ontologist, Highfleet, Inc.

SEMANTIC **T**ECHNOLOGY FOR
INTELLIGENCE, **D**EFENSE, AND **S**ECURITY

October 26, 2010

Lynne Plettenberg
plettenberg@highfleet.com

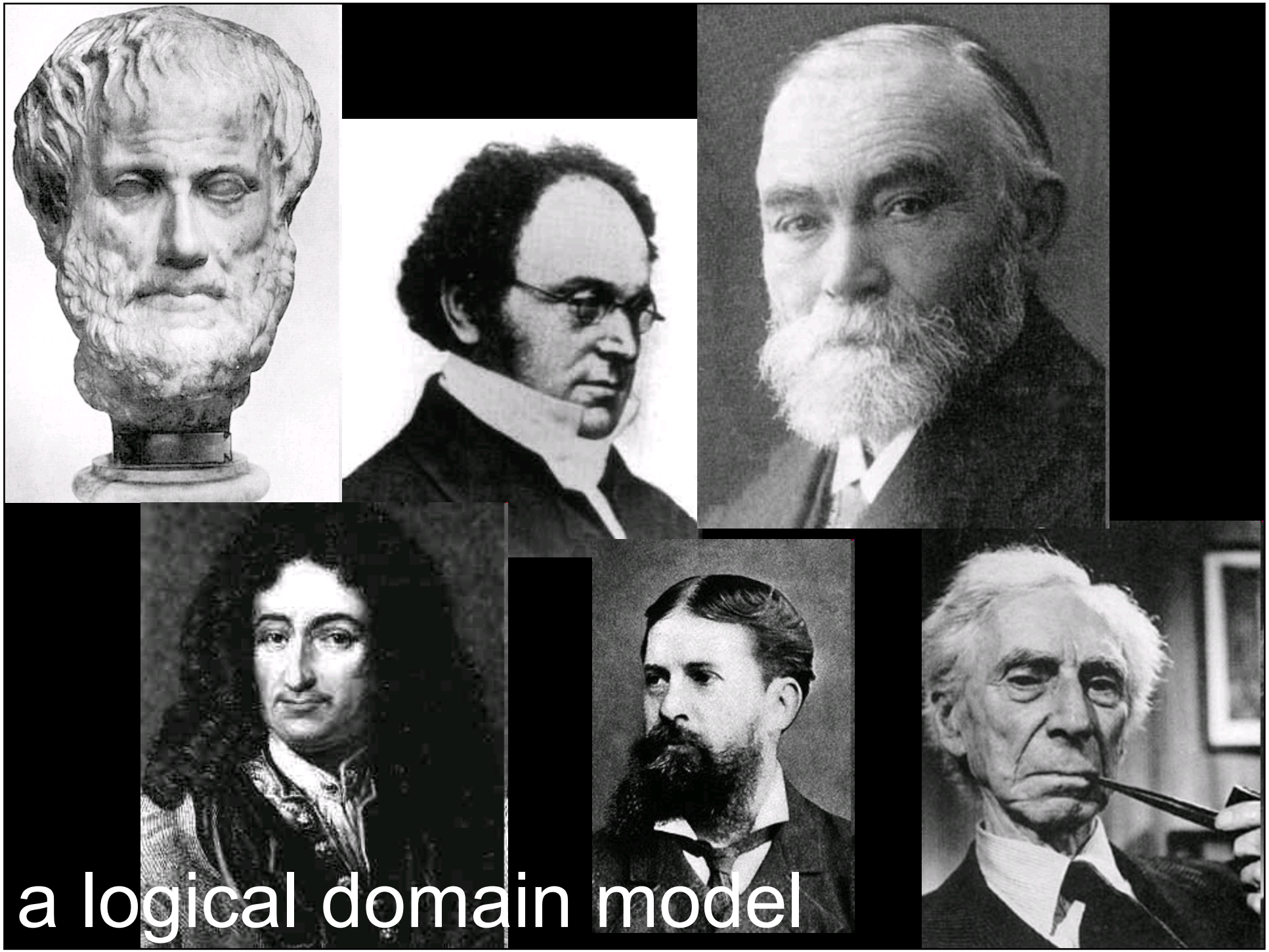
Chris Schanck
cschanck@highfleet.com

www.highfleet.com

Topics

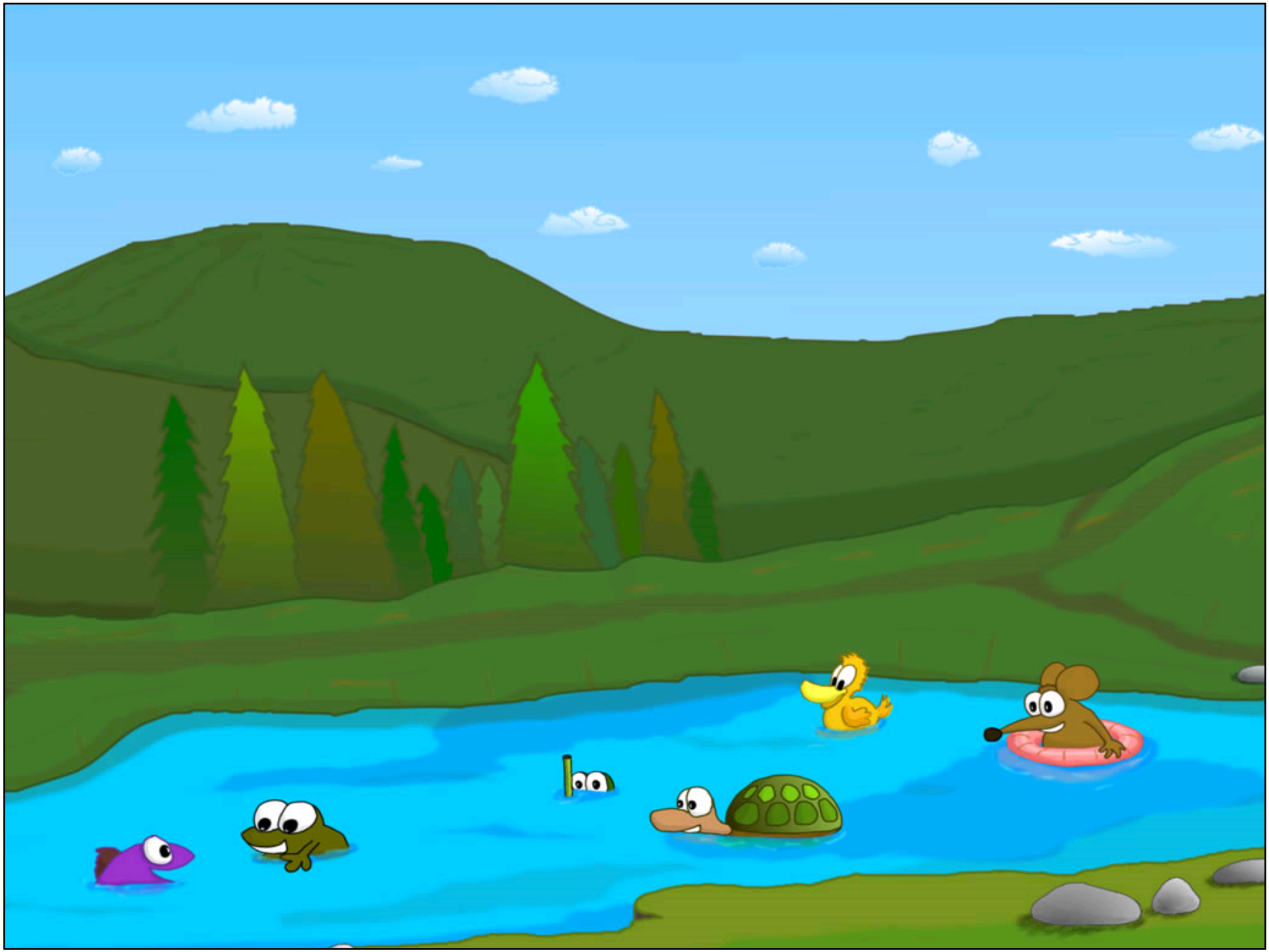
- Quick Intro: What is an ontology?
- Design of a toolset for realizing ontology
- Ontology development best practices
- Semantic federation best practices
- Reification for provenance and security

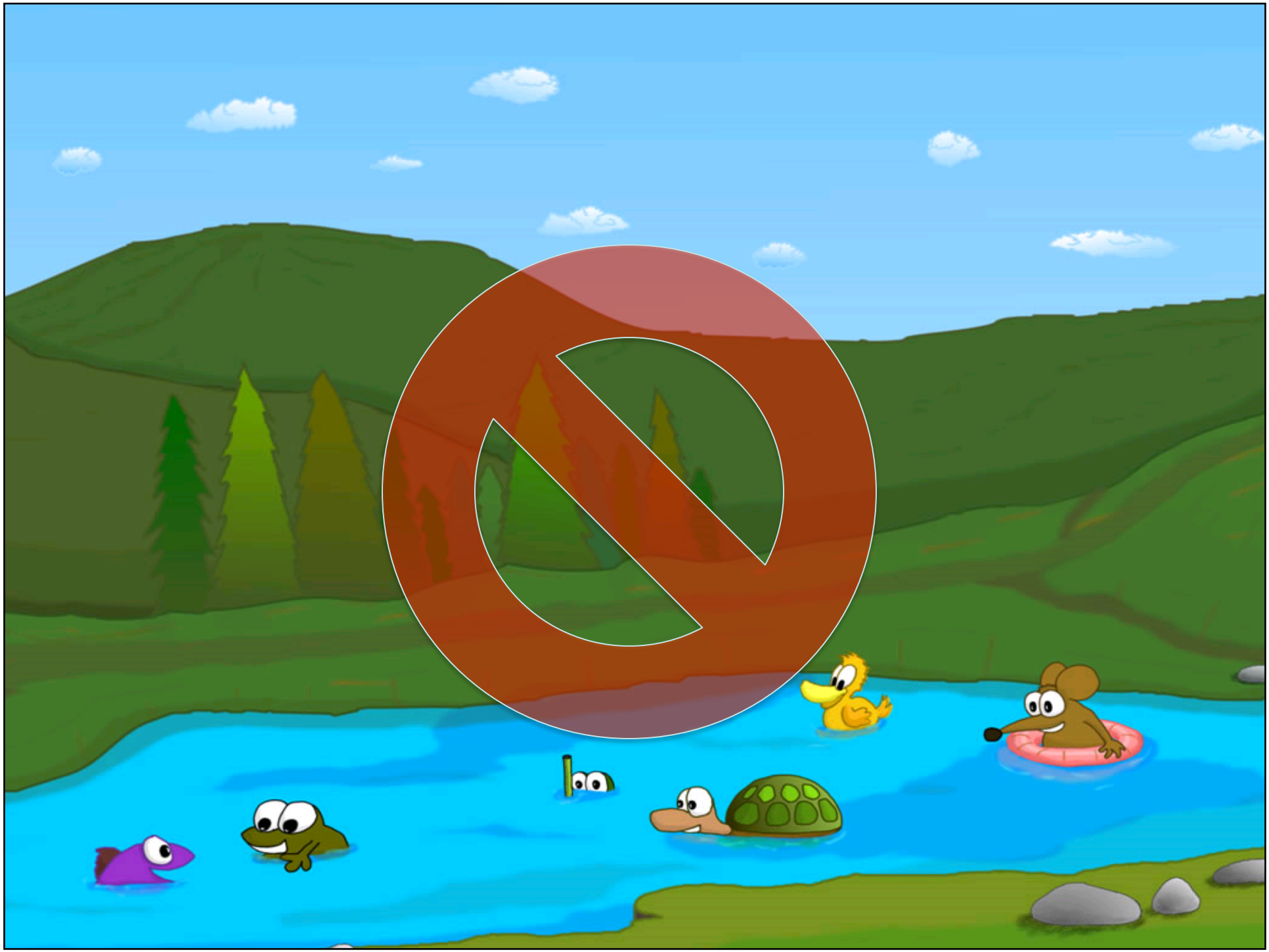
What is an ontology?

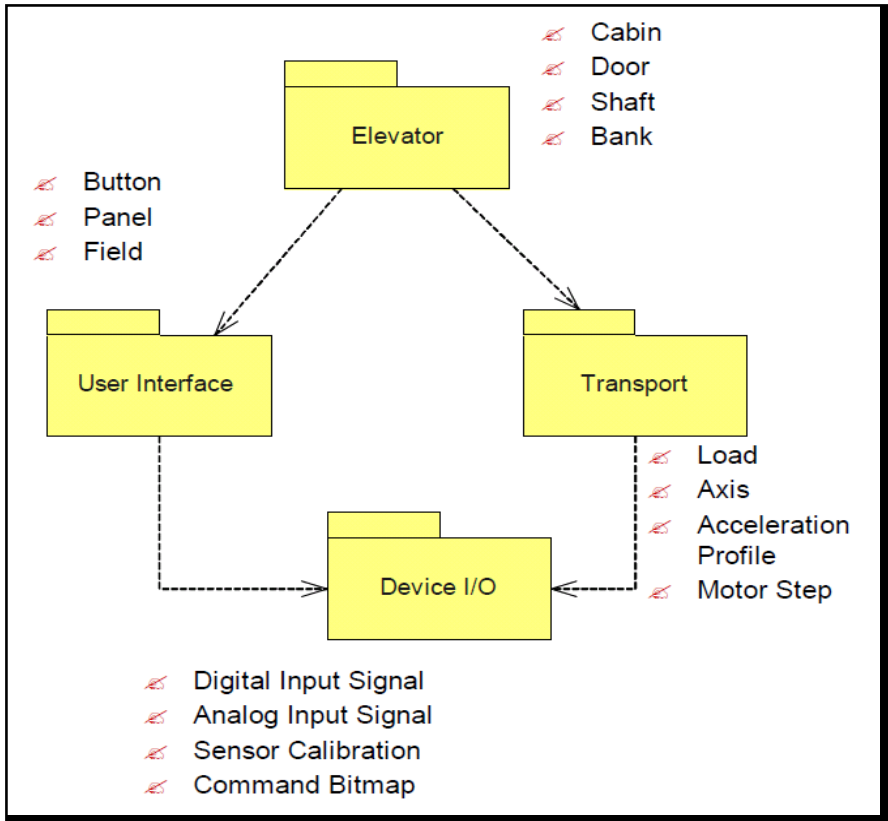


a logical domain model

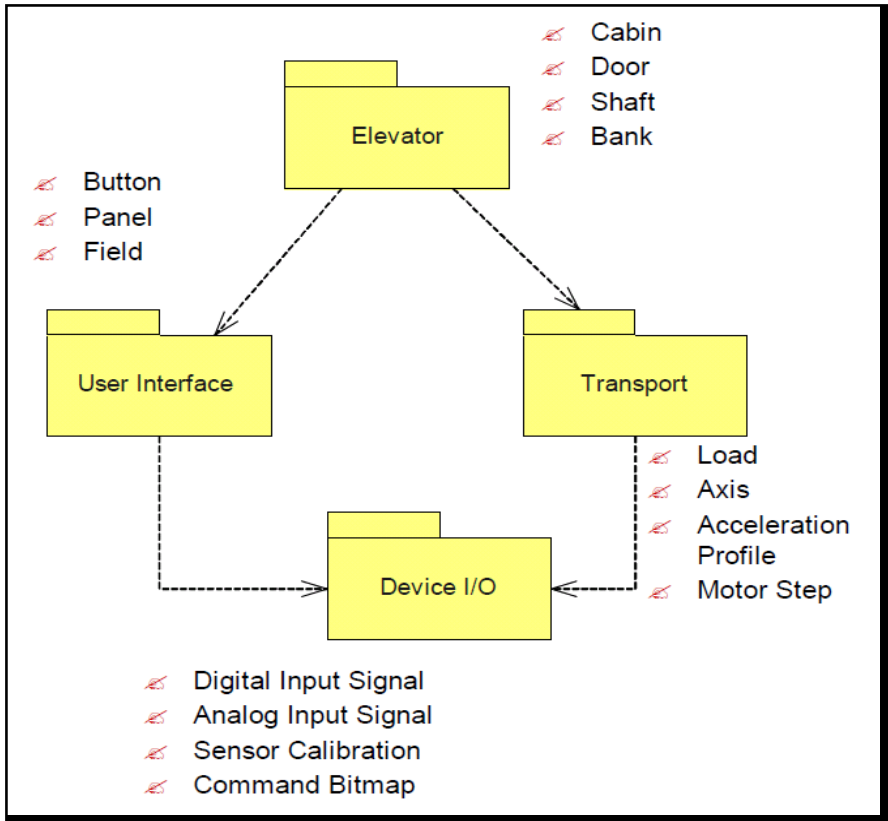

reflects the real world

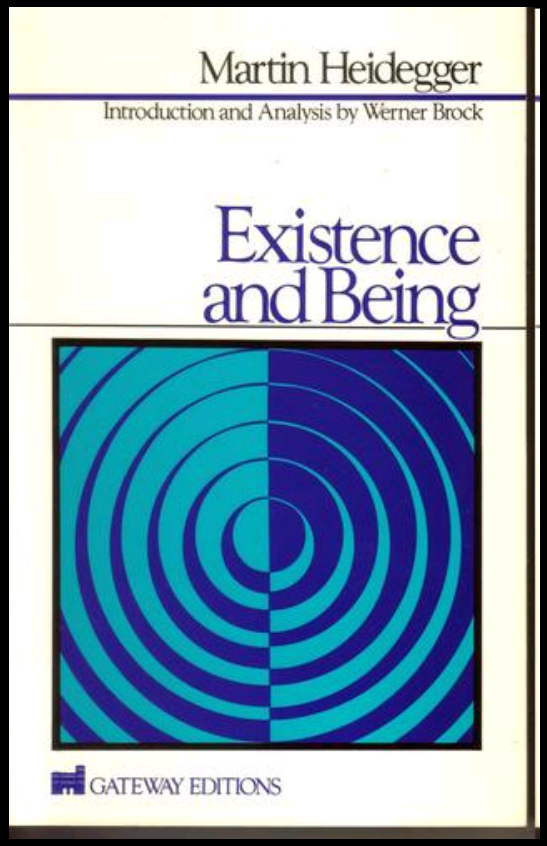




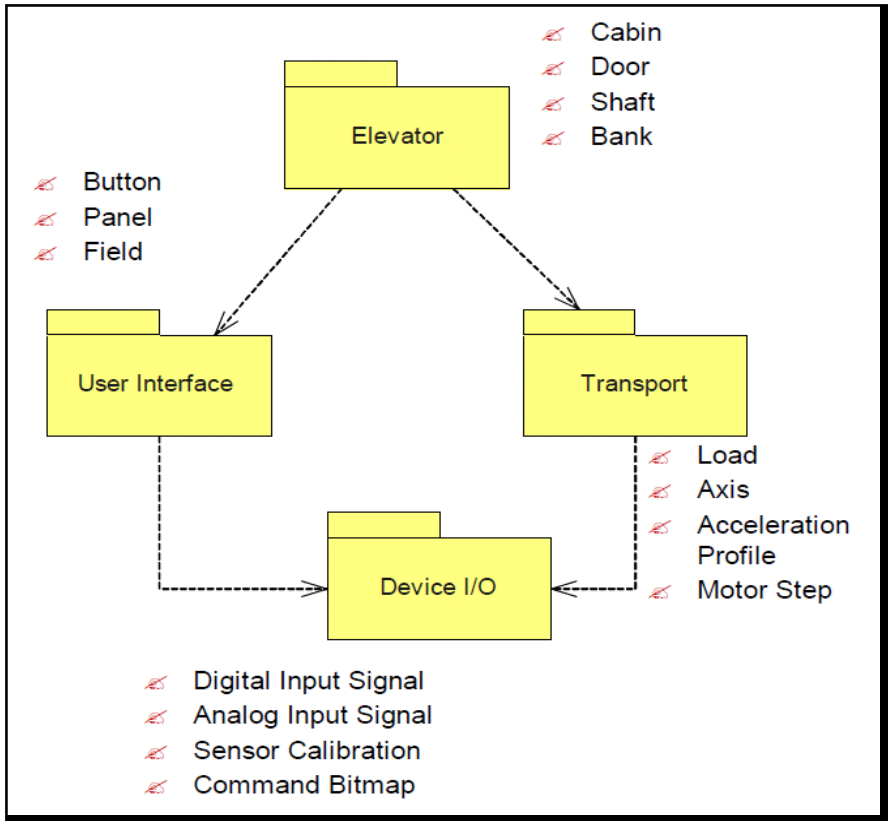


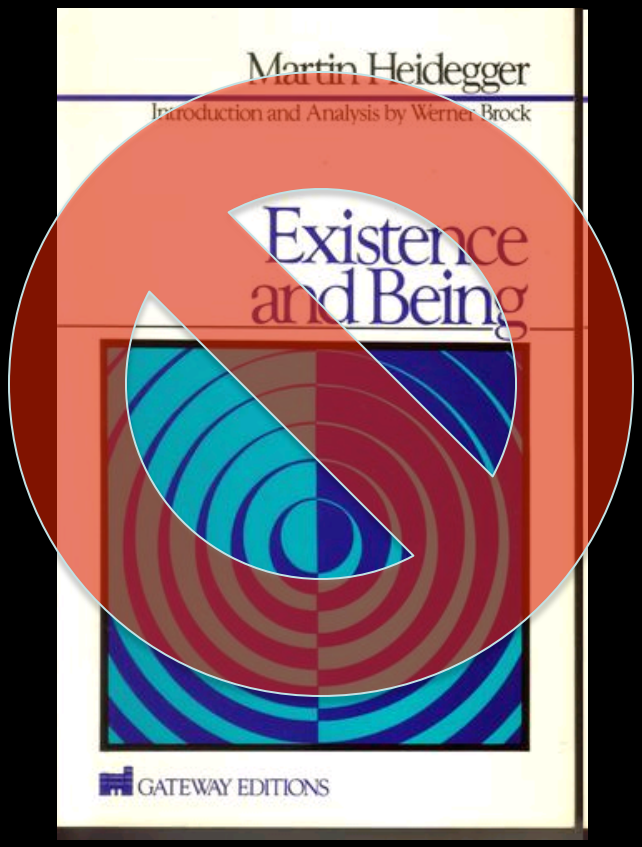
machine readable



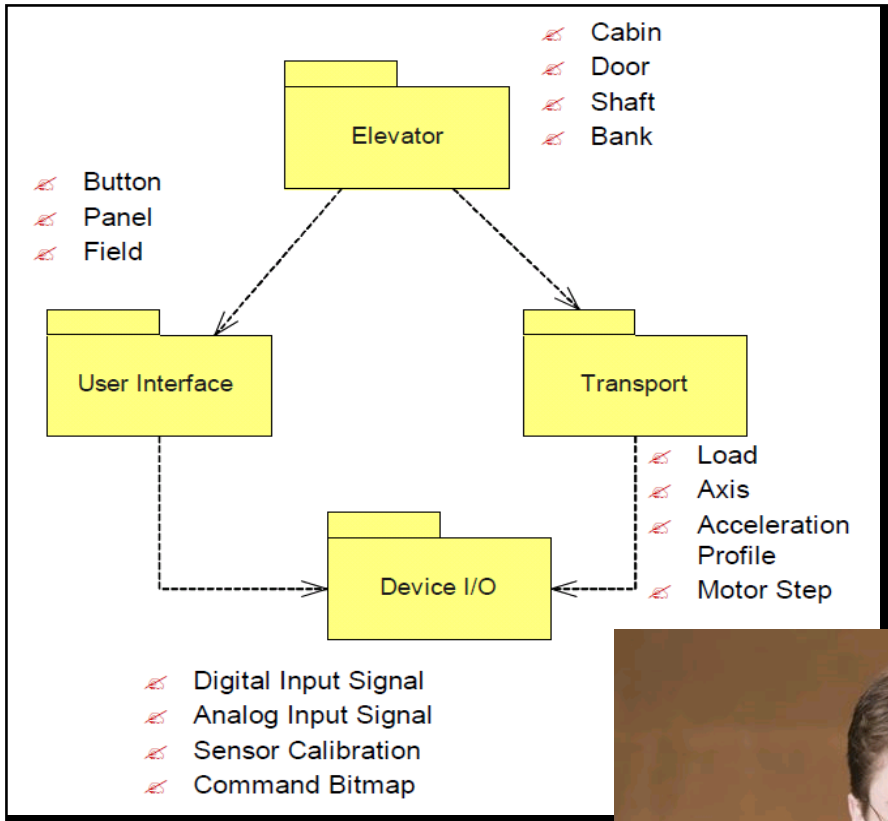


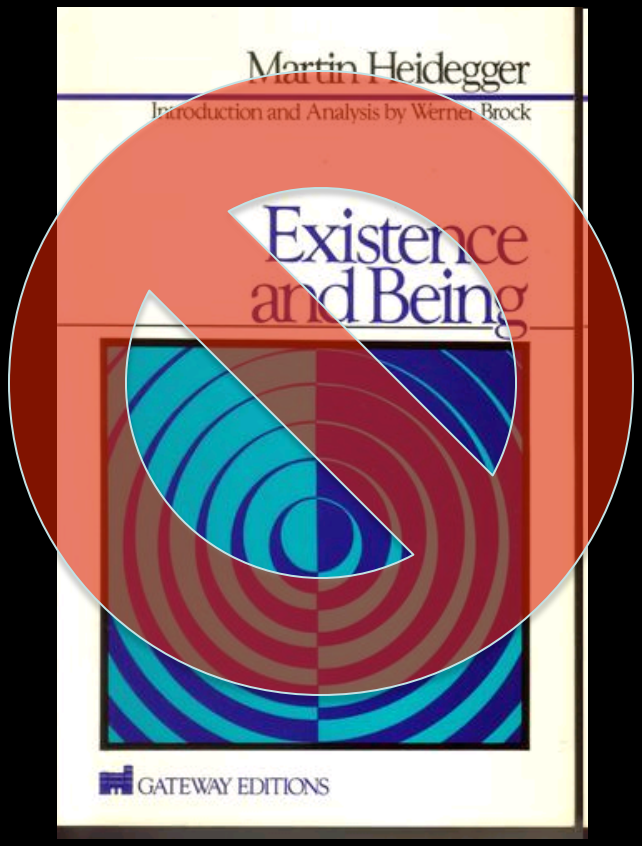
machine readable



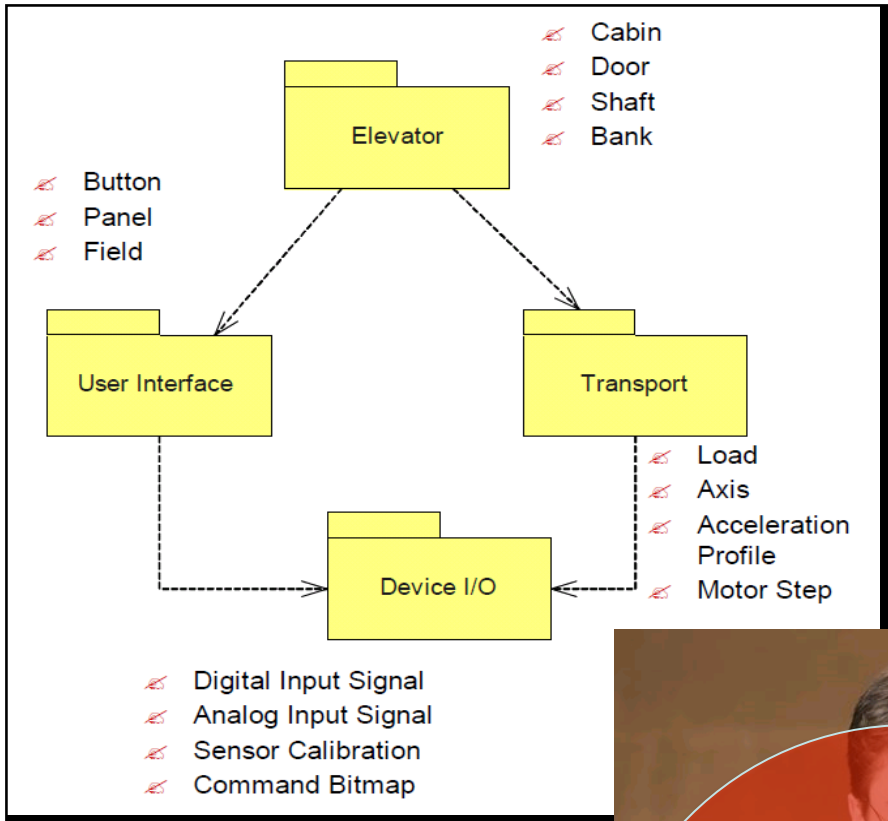


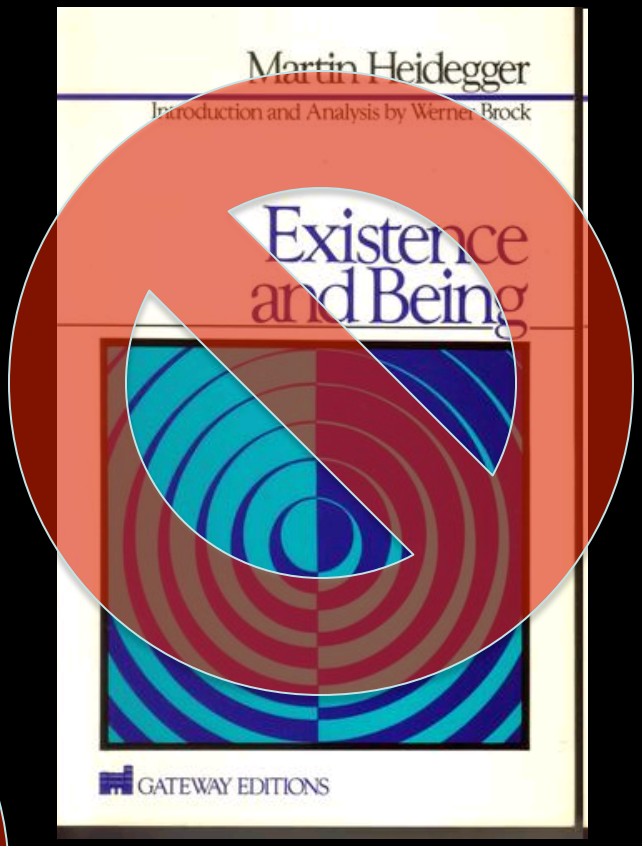
machine readable



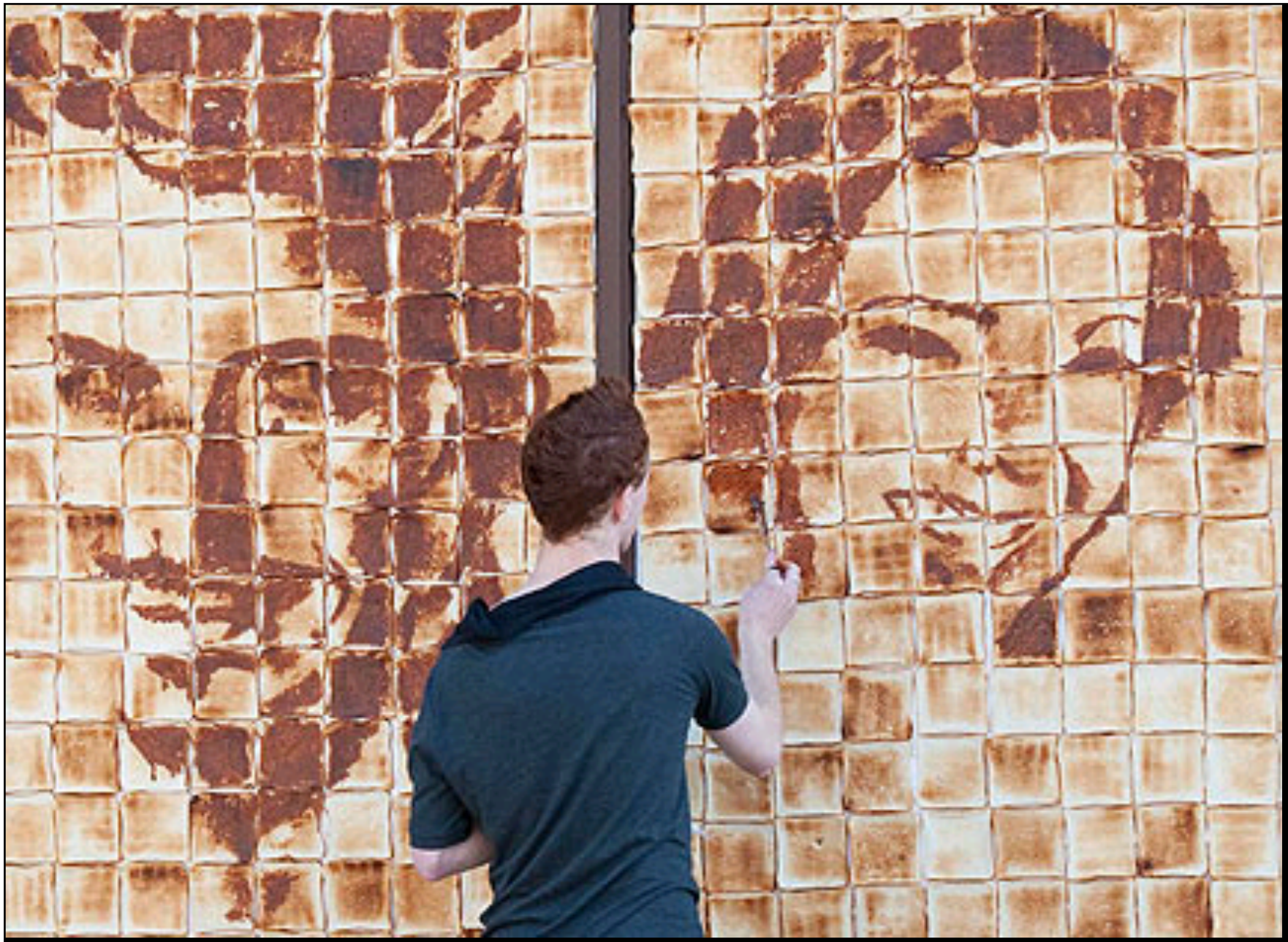


machine readable





machine readable



greater than the sum of its parts

greater than the sum of its parts

Abstract

Medical science often suffers from having so few data and so much imperfect knowledge that a rigorous probabilistic analysis … is seldom possible. Physicians nevertheless seem to have developed an ill-defined mechanism for reaching decisions despite a lack of formal knowledge regarding the interrelationships of all the variables that they are considering. This report proposes a quantification scheme which attempts to model the inexact reasoning processes of medical experts…utilized in a rule-based computer diagnostic system. One such system, a clinical consultation program named , is described in the context of the proposed model of inexact reasoning.

A model of inexact reasoning in medicine

Edward H. Shortliffe

Bruce G. Buchanan

not new

Mathematical Biosciences
Volume 23, Issues 3-4, April 1975, Pages 351-379

Abstract
Medical science often suffers from having so few data and so much imperfect knowledge that a rigorous probabilistic analysis … is seldom possible. Physicians nevertheless seem to have developed an ill-defined mechanism for reaching decisions despite a lack of formal knowledge regarding the interrelationships of all the variables that they are considering. This report proposes a quantification scheme which attempts to model the inexact reasoning processes of medical experts…utilized in a rule-based computer diagnostic system. One such system, a clinical consultation program named MYCIN, is described in the context of the proposed model of inexact reasoning.

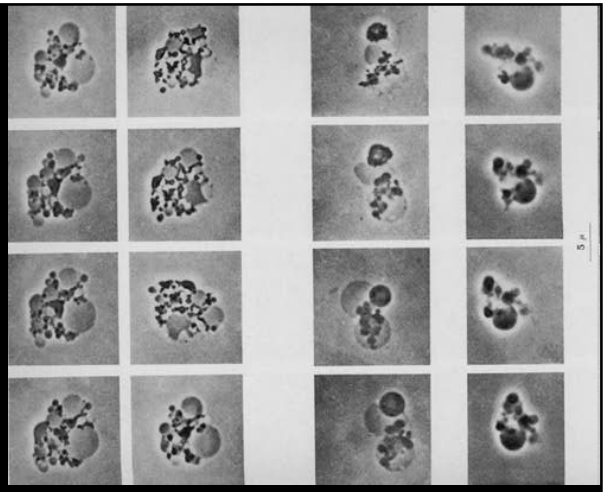
Why do we bother?

- We believe that data-intensive computer systems employing the discipline of ontology are more
  - Accurate
  - Flexible
  - Reusable
  - Responsive
  - Knowledge capture
- and our experience bears this out.

Sampling of recent projects

- Five Ongoing Intelligence Agency Projects
  - Analysis
  - Enterprise Ontology
  - Network Analysis
  - Network Health and Status
  - Determining ROI on Intelligence Collection
- University of New Haven/Institute for the Study of Violent Groups (ISVG)
- FreedTV
  - Social Networks-based recommender for entertainment content.
- Purus Technologies
  - semantic federation to support complex decision making.

How we build ontology



- Modified first-order logic
- Statements & entailments
- Knowledge Frame Language (KFL) & ECLIF
  - *properties*, n-ary relations, constants
  - function terms
  - rules and integrity constraints

Tools Overview

Chris Schanck
Director of Software Engineering, Highfleet, Inc.

SEMANTIC TECHNOLOGY FOR
INTELLIGENCE, DEFENSE, AND SECURITY

October 26, 2010

Overview Goals

- High level product overview
- Examine capabilities of HighFleet Suite across several different user roles
- Point out usage patterns that will be most productive

What's a “HighFleet Suite”?

- Ontology Development Support
  - KFL parsing, deploying; small data load, querying.
- Server Management
  - Start/Stop etc.
- Scalable extensional server
  - Billions of assertions

Architecture Overview

- Model Creation
- Data Store
- Inference Engine
- Scalability
- Interoperability

Model Creation

- KFL Ontology Modeling language, derived from the KIF first-order calculus effort.
- Used as syntactic sugar over Common Logic variant (ECLIF)
- Encompasses type and rule definition with extensions for measurement and time
- GUI deploy wizard and command line compilation support

Data Store

- Purpose-built hyper-graph store, leveraging an array of RDBMS technologies
- More a graph storage layer than triple store
- Optimized for concurrent, ACID access
- Allows 'what if' exploration through transactional visibility
- Range of scale-out options

Inference Engine

- Common Logic based FOL rules engine
- Static Semantic, In-Process Semantic, and Physical query optimizations
- Heavily optimized for parallelized execution
- Deep caching, multiple attack strategies
- Baked-in transactions, time handling, measurement
- Soup-to-nuts integration with the data store

Scalability – Rules

- Optimized for dense logic, not non-productive taxonomies and type hierarchies
- However, reasons efficiently over large taxonomies by treating terms as terms rather than properties.
- Rule-bases with thousands of predicates are fine
- There are some limits, but they tend to be sensible
- Distinction between schema and extensional data

Scalability – Extensional Data

- Distinguishing between schema and data, emphasizing scaling out extensional data
- Use RDBMS's for what they are good at: ACID and Indexing
- Ignore RDBMS modelling conventions
- Allow for fine-grained index control and statistics gathering

Scalability – Query Processing

- Vertical scalability with complete 64-bit support for large hardware situations
- Horizontal scalability by sharing an extensional store across multiple load-balancing query engines
- Intra-engine scaling with intelligent query program overlap reuse.

Interoperability / Flexibility

Runs (and tested!) across a variety of platforms, 64/32 bit, Windows, Linux, Mac OSX

Java API, JDBC-like.

REST-ful interface with complete access via HTTP/ JSON protocol

The HighFleet Suite IODE

- Basic Administration
- Ontology Development
- Manipulating Data
- Advanced Administration
- Miscellaneous

Ontology Model Development

- KFL Model → Deployed XKS
- Use Non-Persistent XKS for Ontology Development
  - Fast for testing, low overhead
- Use Postgres/Oracle for serious deployments

Ontology Development Browsing & Ontodoc

- Database → Browser
  - Traverse a loaded ontology
  - Show type hierarchies, instances, graphs of relations, etc.
  - Eminently searchable
- Ontology Doc Generation
  - Generates HTML description of a loaded ontology

Ontology Development Types of XKS's

- Ontology Development XKS – Non Persistent
  - Designed to give exactly the same behavior as the more expensive standalone XKS servers
  - IODE license is all you need
- Persistent XKS
  - Licensed individually by assertion size
  - H2/Postgres/Oracle

Quote – Types of XKS's

- *"The idea is that the non-persistent XKS and the standalone scalable XKS perform (in terms of correctness) exactly the same, since they share the same core engine with its highly parallel architecture."*

Extensional Data Overview

- Data loaded in terms of the model
- Scalable, performant, persistent (license dependent)
- Tools: Query tool, Questionator, Asserter

Quote – Extensional Data

- *"Remember, creating a model in a vacuum is interesting, but any model is far more useful if you can express data in terms of that model in a scalable manner. The HighFleet Suite is designed to give you a model development environment that works as well for models with tiny data as large data."*

Querying

- Query Tool
  - Text query based
- Questionator
  - Graphical UI for building/saving queries
  - Graphing of results
    - In both cases results can be exported

Fact Asserter

- Used to add data to an XKS
  - Add single facts interactively or files in bulk
- Generally, machine generated data might not need IC checking
- It is often useful to drop/restore indexes around a really large bulk load

Delete Assertions

- GUI Fact Deleter supports deleting
- Non-Persistent XKS is designed to mostly let you ignore deletions
- Usually, you don't delete, but instead denote with a time span

More *Interesting* Administration

- Backup/Export (data + schema or data only)
- Import Data, Restore Database
- Rename, Delete, Clone
- Configure

Exposing processing to the user : PseudoRels and PseudoAssertions

- Mostly for HighFleet internal use, they can be useful for debugging
- PseudoRel (showCompilationInfo (foo ?x)) will dump the query plan to the log as the query is run
- PsuedoAssertion (owi.RelationIndex ?rel ?pos ?indexed) drops/creates indexes (useful for large ingest cases)

Future Features - Incremental

- Query optimization improvements
- More forgiving error reporting
- More work on web interface
- MORE

Future Features – Large Scale

- 64-bit off-heap caching, clustered caching
- User-controlled materialization
- Mutable (forward) ontology modifications
- eclipse/IDE based structured KFL editing
- Multi-lingual logic processing

Developing a User-facing Ontology Application, Soup to Nuts

Lynne Plettenberg
Ontologist, Highfleet, Inc.

SEMANTIC TECHNOLOGY FOR
INTELLIGENCE, DEFENSE, AND SECURITY

October 26, 2010

Starting Points

- Data
- Loose query requirements
- Interface plans
- Business goals

Demo: Keshvad application

- Data
  - Movements and Communications of Persons of Interest (who, where & when)
  - Simple geographic data
  - Parents, spouses, group membership
- Requirements
  - Social network analysis
  - Support sophisticated target tracking
  - Fully exploit the data

Best practice for ontology development: ontological layers

Interface
Describes the end use of the system (sample queries, display)

Application-specific domain
Describes the world from a particular (enterprise's) perspective

Application-independent domain
Concepts useful to many perspectives

System support
Basic concepts like relation

Best practice: reuseable upper level ontology

Interface
Describes the end use of the system (sample queries, display)

Application-specific domain
Describes the world from a particular (enterprise's) perspective

Application-independent domain
Concepts useful to many perspectives

System support
Basic concepts like relation

Highfleet Upper and Mid- Level Ontologies (ULO and MLO)

Best practice: reuseable upper level ontology

- Property, Material Role
- Relations, Subsumption
- Dates, numbers, strings
- Binary Relation, Transitive Closure BR
- Event
- gt, lt

Best practice: Leverage industry-standard domain models

Interface
Describes the end use of the system (sample queries, display)

Application-specific domain
Describes the world from a particular (enterprise's) perspective

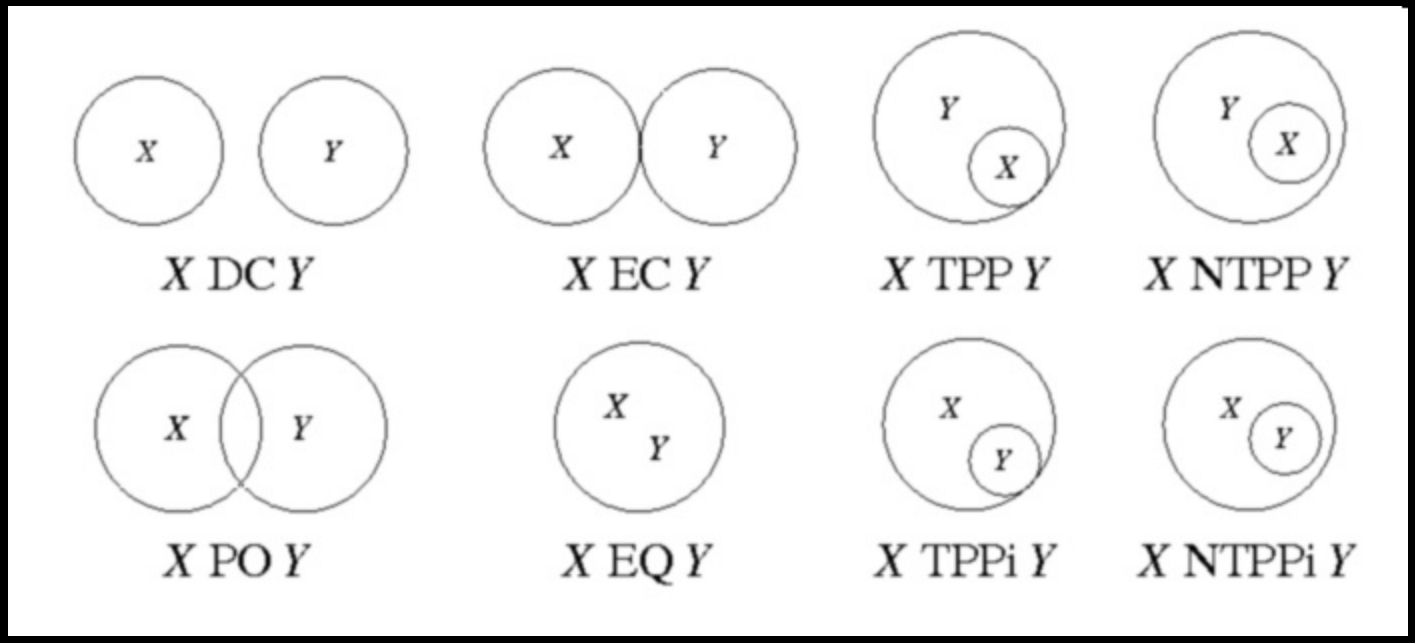
Application-independent domain
Concepts useful to many perspectives

System support
Basic concepts like relation

Use existing standards and best practices
(even if it hurts a little)

Best practice: Leverage industry-standard domain models

- Region Connection Calculus (RCC)



Randell, D. A., Cui, Z. and Cohn, A. G.: A spatial logic based on regions and connection, Proc. 3rd Int. Conf. on Knowledge Representation and Reasoning, Morgan Kaufmann, San Mateo, pp. 165–176, 1992.

Model Development Task

Interface
Describes the end use of the system (sample queries, display)

Application-specific domain
Describes the world from a particular (enterprise's) perspective

Application-independent domain
Concepts useful to many perspectives

Keshvad ontology

Interface
target tracking & network visualization

Application-specific domain
movements & communications

Application-independent domain
social networking & geography (RCC)

System support

Deep dive: exploiting a movement event

1. There was a movement.
   (Movement trv-02)
2. This movement occurred from Jan 24 to Jan 26.
   (occursAt trv-02 (Span (Date "2008 Jan 24") (Date "2008 Jan 25")))
3. This movement was from the city Shiraz.
   (from trv-02 (keyCity "Shiraz" IR "Fars"))
4. This movement was to the city Zahedan.
   (to trv-02 (keyCity "Zahedan" IR "Sistan e Baluchestan"))
5. Keshvad made this movement.
   (agent trv-02 id-00)
6. The city Zahedan is located in the country Iran.
   (locatedIn (keyCity “Shiraz” “IR” “Fars”) (keyCountry “IR”))

What can you conclude from these facts?

Deep dive: exploiting a movement event

- Where was Keshvad located at the start of Jan 24th? At the end of Jan 26th?

```
(=>  (and     (Movement ?e)
              (from ?e ?place)
              (agent ?e ?a)
              (hasStartTime ?e ?t))
     (holdsIn ?t (locatedIn ?a ?place)))
```

If you move *from* some place, then at the start of that movement you are located at that place.

- Similar rule for *to*

Deep dive: exploiting a movement event

- When was Keshvad in Iran?
- (TransitiveBR locatedIn)

  If a is located in b, and b is located in c, then a is located in c.

Deep dive: exploiting a movement event

- When was Keshvad in Iran?
- (TransitiveBR locatedIn)

  If a is located in b, and b is located in c, then a is located in c.

```
(=>  (and    (locatedIn ?a ?b)
             (locatedIn ?b ?c))
     (locatedIn ?a ?c))
```

Deep dive: exploiting a movement event

```
(schema (TransitiveBR ?Rel)
        (=>     (and    (?Rel ?a ?b)
                        (?Rel ?b ?c))
                (?Rel ?a ?c)))

(=>     (and    (holdsIn ?t (locatedIn ?a ?b))
                (holdsIn ?t (locatedIn ?b ?c)))
        (holdsIn ?t (locatedIn ?a ?c)))
```

Deep dive: exploiting a movement event

- It's Jan 28th. Where was Keshvad last seen and how long ago?

```
(=>
    (and        (Agent ?Agent)
                (Time ?QueryTime)
                (maxTermf ?T (?T ?Y)  (and      (atOrBefore ?T ?QueryTime)
                                                (observedIn ?T ?Agent ?Y))
                ?MaxTime)
                (observedIn ?MaxTime ?Agent ?Region)
                (ifThenElse
                        (temporalIntersection ?MaxTime ?QueryTime ?Int)
                        (= ?Elapsed (hours 0))
                        (and        (endingInstant ?MaxTime ?Left)
                                    (startingInstant ?QueryTime ?Right)
                                    (duration (Span ?Left ?Right)?Elapsed))))
    (lastSeenIn ?QueryTime ?Agent ?Region ?Elapsed))
```

Best practice: test-driven development

- Sample data set & test cases
- Ideally working w/ interface developer or from client req's
- Store test cases as ontological data for easy handoff

Best practice: test-driven development

- Show test queries and run a few

Best Practices for Client Application Development

Semantically Integrating Legacy Databases

Lynne Plettenberg
Ontologist, Highfleet, Inc.

SEMANTIC TECHNOLOGY FOR
INTELLIGENCE, DEFENSE, AND SECURITY

October 26, 2010

What are we trying to do?

- We want to convert multiple structured data sources
  - different formats
  - internal and external
  - constantly changing
- into new, actionable information
  - could not be discovered from a single source
  - does not exist as the intersection of row and column
  - needs to be current
- within time and budget constraints.
  - a few weeks is not a reasonable query response time
  - minimize use of expensive resources

Best practice: create a logical model of each data source

Problem: Existing data federation processes are time-consuming and brittle.

Solution: Generate logical models of sources & map them to a single ontology.

- Central ontology means less mapping, **lower cost, faster implementation.**
  - "Every lab has a different LIMS, but they all use the same chemistry."
  - Lower cost to add new data sources.

- Separates the logic of integration from the implementation of integration for **greater flexibility, easier maintenance.**
  - Traditional integration tools require you begin with the end in mind.
  - Semantic approach is more responsive to schema and requirement changes.
  - Allows you to build the model incrementally.

Best practice for ontology development: ontological layers

Interface
Describes the end use of the system (sample queries, display)

Application-specific domain
Describes the world from a particular (enterprise's) perspective

Application-independent domain
Concepts useful to many perspectives

System support
Basic concepts like relation

SemFed adds layers

Interface
Describes the end use of the system (sample queries, display)

Application-specific domain
Describes the world from a particular (enterprise's) perspective

Application-independent domain
Concepts useful to many perspectives

System support
Basic concepts like relation

SemFed adds layers

Interface
Describes the end use of the system (sample queries, display)

Application-specific domain
Describes the world from a particular (enterprise's) perspective

Application-independent domain
Concepts useful to many perspectives

Connection
Describes how data interrelates

Base
Logical model of each data source

System support
Basic concepts like relation

Automatic Data Source Analysis

Interface
Describes the end use of the system (sample queries, display)

Application-specific domain
Describes the world from a particular (enterprise's) perspective

Application-independent domain
Concepts useful to many perspectives

Connection
Describes how data interrelates

Base
Logical model of data sources

System support
Basic concepts like relation

Generated by SemFed Toolkit

Metadata analysis and model generation

- If data is messy, clean it up
  - Primary key detection
  - Type inference
- If data is orderly, dig out every nugget of info
  - Aspect relations
  - Modeling recommendations
- Any structured data source
- Configure and iterate

SemFed Toolkit: KFL Generator

- Run KFL Generator against the CIA World Factbook

Metadata analysis and model generation

- View some generated KFL
- Mapping varies with source type
- Contains sufficient information to reverse engineer the source!

Demo: Keshvad ontology with remote data

- Collected open-source data about activities of violent groups (who, where & when)
  - 937 tables
- CIA World Factbook
  - 33 tables or as RDF

Best practice: Apply "thin coats" of connection ontology

- Mapping properties, mapping relations
- Extra rigor now = extra flexibility later
- Per-source mapping

- View some connecting ontology
- Domain ontology extensions were minimal

SemFed adds layers

Interface
Describes the end use of the system (sample queries, display)

Application-specific domain
Describes the world from a particular (enterprise's) perspective

Application-independent domain
Concepts useful to many perspectives

Connection
Describes how data interrelates

Base
Logical model of each data source

System support
Basic concepts like relation

Problem

- Source 1
  - (keyCityTBL 2821440)
  - (locatedIn (keyCityTBL 2821440) Incident-01)

- Source 2
  - (keyCity Tehran IR)
  - (locatedIn Keshvad (keyCity Tehran IR))

Best practice: Use ontology to find equivalent entities

- Establish equivalence sets
- Materialize equivalence sets
- Enforce equivalence sets **at the base level**

Optimization for SemFed

- Ingest/extract static/slow-changing data
- Adjust caching time-to-live

Maintenance

- Accommodating a schema change
- Accommodating a requirements change

Reification and proofs in ECLIF

Support of advanced metadata applications in the XKS

Bill Andersen, Highfleet, Inc.

STIDS 2010 Tutorial Session
October 26, 2010

Customers ask about...

- Multi-level security filtering
- Discretionary access control
- Provenance
- Modularity
- Contextual reasoning
- ...
- We're trying to do something about that!

Goals of this presentation

- Present our plans for support of advanced metadata applications in ECLIF / XKS
  - Compare with RDF/OWL reification
  - Explain our approach
- Solicit your input
  - What do(n't) you like?
  - What haven't we thought of?
  - Would you use this? For what applications?

Intended audience

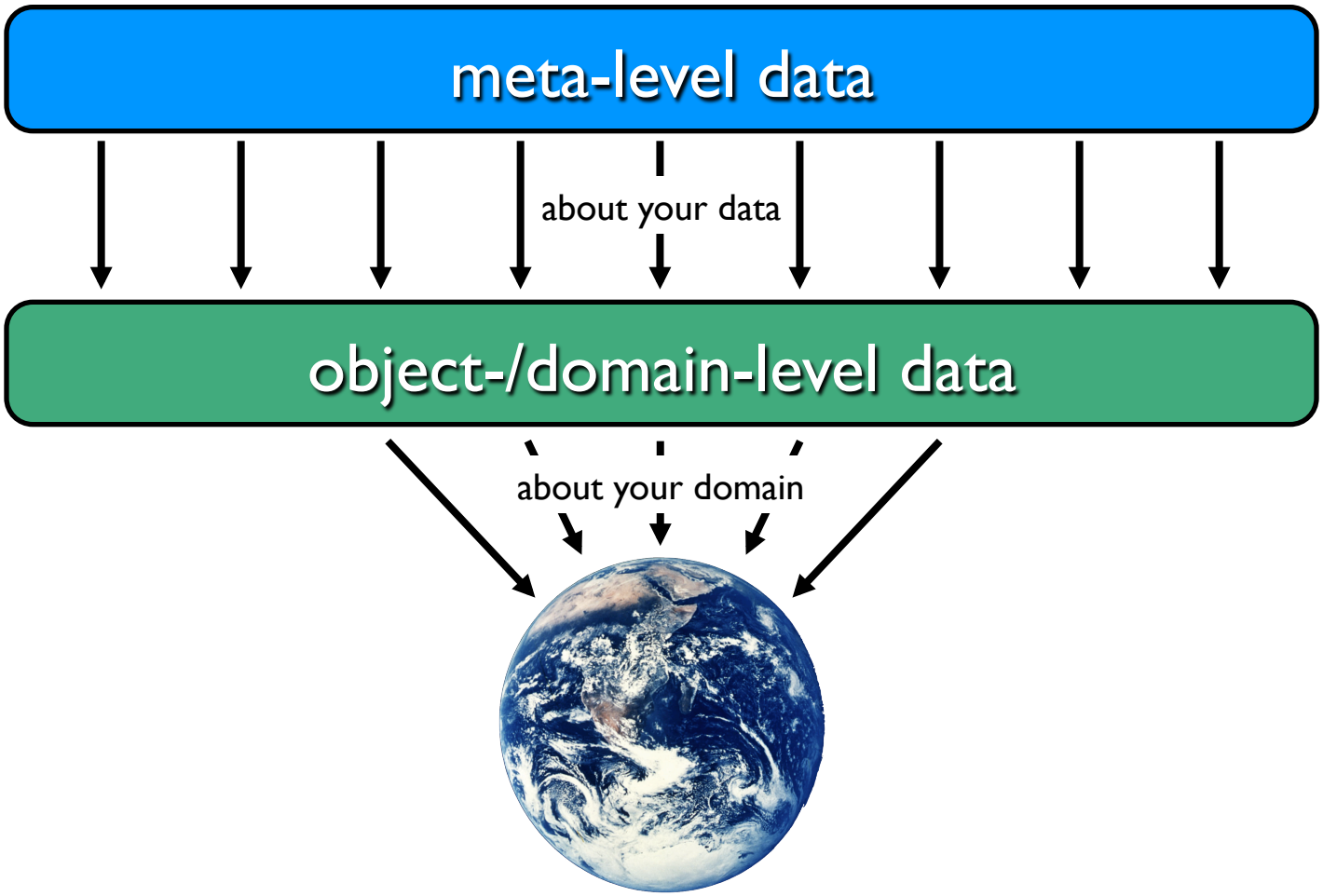
(in order)

- Software engineers who want implement metadata in ontology-based applications
- Ontologists and technologists who are interested in applications of meta-level reasoning
- Project leads who need to understand the advantages of using meta-data in knowledge intensive applications

Outline

- Metadata and its applications
  - Reification and metadata
- Reification in RDF
- Reification in ECLIF
- Extension of reification to inference
- Examples and applications

Metadata


meta-level data
about your data
object-/domain-level data
about your domain

Why we need metadata

- Tracking history of data
  - Transaction time, provenance
- Controlling data access
  - Multilevel security, privacy
- And more ...

- To do this we have to talk about data

Reification

- ... is a mechanism for making sentences into objects of reference – talking about data

  source([likes(bob, pizza)], joe)

- ... is the way we refer to data so we can use metadata to talk about it
- ... has lots of ways to approach it with various ramifications

Reification in RDF

Reification in RDF

an RDF triple T:

```
<bob likes pizza>
```

and its reified form T_R:

```
{ <s1 rdf:type rdf:Statement>,
  <s1 rdf:subject bob>,
  <s1 rdf:predicate likes>,
  <s1 rdf:object pizza> }
```

The reified form, T_R, of a triple T is a set of four triples made by introducing a new name for T and relating it to T's constituent parts by special RDF vocabulary

Usually, blank names are used for the new name

Note on RDF semantics

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

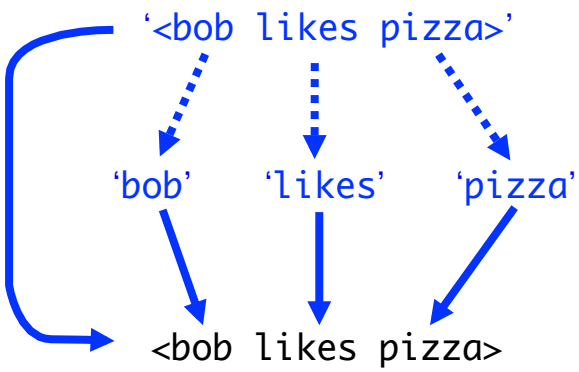
- RDF semantics are based on graphs
- An RDF graph is a set of triples
  - This is important as we'll see later
- RDF entailments are relativized to graphs
  - No inter-graph entailments
- Same goes for OWL / RDFS

RDF reification semantics - 1

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

- RDF reification is not quotation
  - The relata of the reified form are not names
- Reified form says what an RDF triple talks about, not what it is

meta-language w/quotation



RDF reification

```
<s1 rdf:type rdf:Statement>
<s1 rdf:subject bob>
<s1 rdf:predicate likes>
<s1 rdf:object pizza>
```

```
<bob likes pizza>
```

RDF reification semantics - 2

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

- Intentionally weak entailment semantics ...
  - $T_R \subseteq G_1 \nvDash_{RDF} \exists(G_2) T \in G_2$
  - $T \in G_1 \nvDash_{RDF} \exists(G_2) T_R \subseteq G_2$
- ... lead to identity problems
  - To which T in which G does T_R refer?
  - Nothing in the language spec
  - Requires semantic extension(s) to address

Operational issues

- SPARQL can query T and T_R in a single query
  - Assuming you've placed T and T_R in known graphs so it they be jointly accessed
- RDFS and OWL give you no way to do inference that involves reification
  - You can refer to specific graph contents but not about their RDFS/OWL entailments

RDF reification – summary

- RDF reification lets you use sets of triples to talk about the subject matter of other triples
- You use extra-logical means (SPARQL) or semantic extensions implemented in your own code to connect the two
- RDF reification is blind to RDFS / OWL entailments

Reification in ECLIF

Reification in ECLIF

- Immediate goals same as RDF reification
  - Say things about asserted data
  - Query about those things
- Simple extension to ECLIF language
  - Syntax
  - Semantics

ECLIF reification syntax

Normal GAF

```
(likes bob pizza)
```

Reified GAF

```
(stmt s1 (likes bob pizza))
```

Fancier cases

```
(stmt s1 (between milwaukee chicago green-bay))

(stmt s1 (holdsIn (Date “2010”) (likes bob pizza)))
```

ECLIF reification semantics

- ECLIF has no "graphs" in its model theory
  - More exactly, there is only one "graph"
  - ECLIF reification thus differs from RDF
- For any ground atomic formula F
  - $F_R \models_{ECLIF} F$, or...
    - The reified form entails the non-reified GAF
    - Converse does not hold → regress

Digression – types and tokens

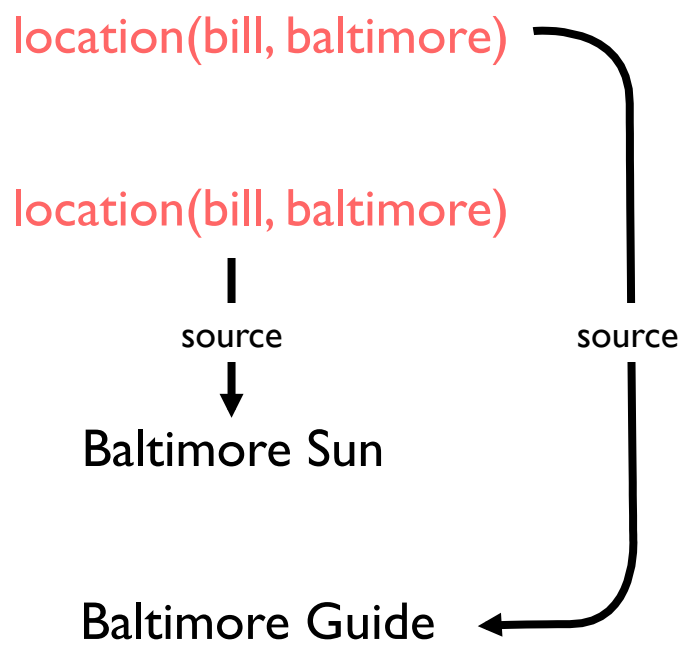
When we assert a sentence, we mean to say that it is true

We all know asserting it twice doesn't make it "more true"

This makes sense only if you're only talking about uses versus mentions of a sentence

Two mentions of a sentence could very well differ in their properties

We call those separate mentions sentence tokens



More types and tokens

- RDF includes the type/token distinction
- RDF reifications are intended to be references to triple tokens
  - A token in RDF is the appearance of a triple *of a given form* in a graph
- This form is the triple type
  - It is distinguished by the values of rdf:subject, rdf:predicate, and rdf:object

So, ECLIF reification is...

- ECLIF reification is a mechanism for naming tokens of ECLIF sentence types
  - Sentence types are not propositions
- It is possible to have multiple tokens of the same type in a single KB
  - Bags of sentence tokens (wrt type) vs sets
  - RDF semantics precludes this possibility
  - Multiple applications

Life is simpler this way

- You're probably reifying because you want to say something about your data
- You can simulate distinct RDF-like "graphs" by adding a graph-membership predicate
- Those simulated graphs will not segregate reified and object-level statements
- But they will allow you to keep statements in separate "containers" for

Simple example - stmt

Data

```
(stmt s1 (likes bob pizza))
(stmt s2 (likes bob beer))
(source s2 sue)
```

Queries

```
(stmt ?s (likes bob pizza))   => {{?s:s1}}


(stmt ?s (likes bob ?x))      => {{?s:s1,?x:pizza},{?s:s2,?x:beer}}


(and (stmt ?s (likes bob ?x)) => {{?s:s2,?x:beer,?source:sue}}
     (source ?s ?source))
```

What's new?

- Sentential operator – stmt
  - Used in ground data and in queries wherever atomic formulas may appear
  - Uses may not be nested
- Vocabulary for quantifying over statements
  - Statement - used to type and quantify over sentence tokens
  - statementType - used to map sentence tokens to sentence types

Summary

- ✔ Minimal ontologist effort to use
- ✔ Low storage, optimization & processing overhead
- ✔ Combined meta and object-level query
- ✔ No performance impact if reified data not used
- ✘ No HO quantification in current prototype
- ✘ Like RDF this is not real quotation
- ✘ No entailments

Proofs in ECLIF

Reification and inference working together

Problem statement

- Reification
  - Gives us names for sentence tokens that we can use to express metadata
- Inference
  - Generate “new” statements from asserted statements
- Reification + inference?
  - What would this look like?

Extra-logical approach

- Instrument query engine with machinery to construct proof objects alongside answer generation
- No necessary connection between
  - proof objects and domain entities
  - proof structure and ontology structure
- Return proof objects alongside answers
- This approach used by current XKS (≤ 4.x)

Logical approach

- Extend logic with meta-logical operators that construct proofs along with answers
- Proofs (along with statements) become part of extended UoD
  - Extension of reification approach
  - Can quantify over them → very powerful
- Prototype under development in XKS

Digression

- First we have to look at ECLIF's properties as a logical system ...
- ECLIF semantics are WFS
  - Under non-mon Lloyd-Topor transform
- WFS is a logic programming semantics for interpretation of axioms in clause form
  - $H \leftarrow G_1 \wedge G_2 \wedge \ldots \wedge G_n \quad (n \geq 0)$
- Proofs will reflect this underlying logic

Syntax

Extend stmt operator to all sentences - not just GAFs

```
(stmt a1 (=> (hasBrother ?x ?y) (hasSibling ?x ?y)))
```

Sentential operator – prove

```
(prove ?p (hasAncestor p235 ?x))
```

May enclose any formula

```
(prove ?p (and (hasParent p235 ?x)
               (hasBrother ?x ?y)))
```

May be nested

```
(prove ?p1 (and (hasParent p235 ?x)
                (prove ?p2 (hasBrother ?x ?y))))
```

Proofs

- Proofs are trees
  - Non-leaf nodes are (ground) theorems
    - Unification (var instantiation) omitted
  - Leaf nodes are axioms (trivial theorems)
  - Non-leaf nodes result from application of proof operators to other nodes
- Each different proof represents a unique way of deriving a given answer - Key!

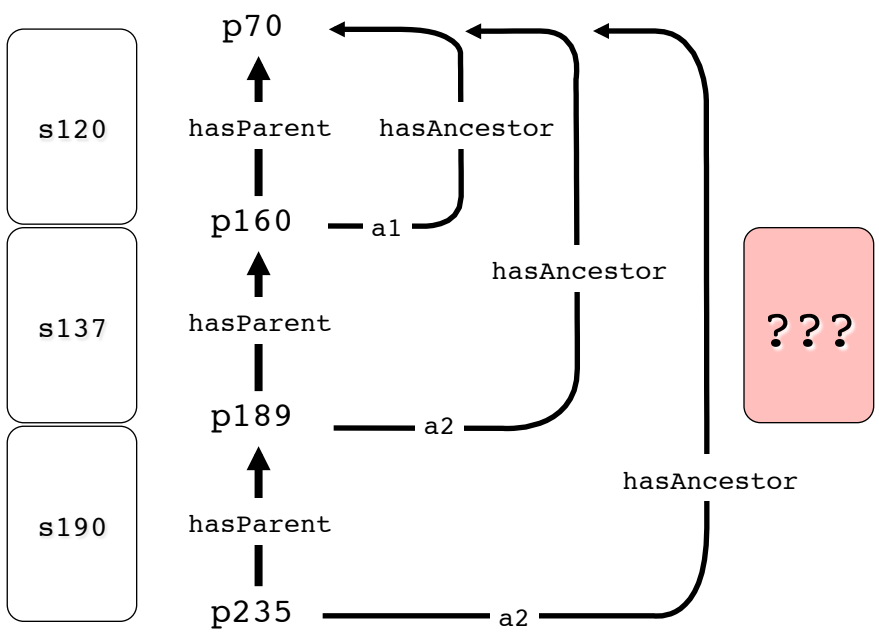
Example

simple ontology

```
a1: (<=
      (hasAncestor ?x ?y)
      (hasParent ?x ?y))

a2: (<=
      (hasAncestor ?x ?y)
      (and
        (hasParent ?x ?z)
        (hasAncestor ?z ?y)))
```

some reified data

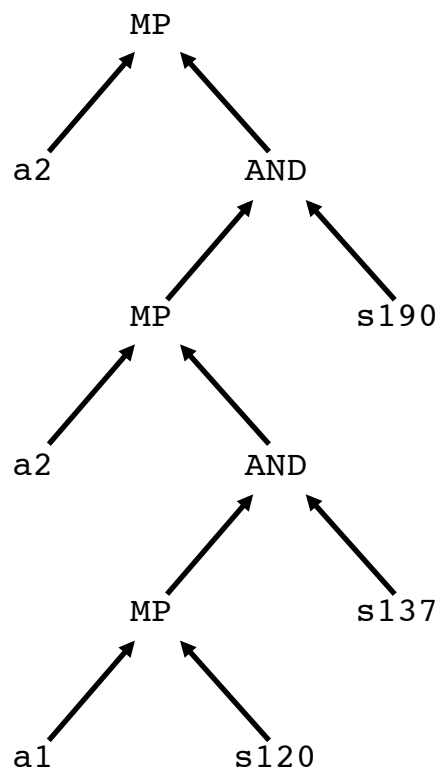
```
(stmt s120 (hasParent p160 p70))

(stmt s137 (hasParent p189 p160))

(stmt s190 (hasParent p235 p189))
```



A sample proof

```
(prove ?p (hasAncestor p235 p70))
```

?p : (MP a2 (AND s190 (MP a2 (AND s137 (MP a1 s120)))))

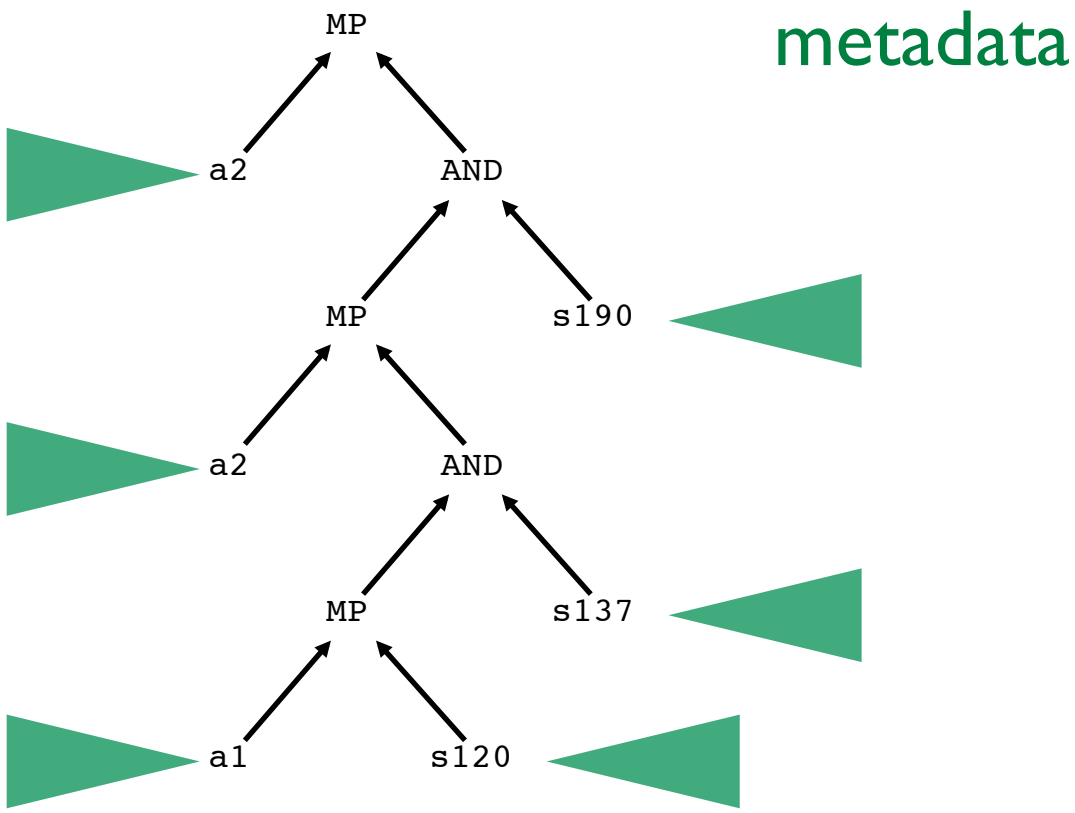


How this works

- Class of proofs – Proof
- Reified statements prove themselves
  - Statement is a subclass of Proof
- Proof operators construct proofs
- Proofs can be quantified over
  - Their parts can be inspected
  - Parts that are statements carry metadata

Proofs and metadata - recipe

```
(and (prove ?p (hasAncestor p235 p70))
     formula for proof analysis)
```



Application examples

Multi-level security filtering

```
(and (prove ?p (hasAncestor p235 p70))
     (proofSecLevelLTE ?p SecLevel_S))
```

Provenance

```
(and (prove ?p (hasAncestor p235 p70))
     (exists (?a)
       (and (axiom ?p ?a)
            (hasSource ?a BaltimoreSun))))
```

Probability / confidence

```
(and (prove ?p (hasAncestor p235 p70))
     (proofConfidence ?p ?conf))
```

Proof complexity

```
(and (prove ?p (hasAncestor p235 p70))
     (count (and (subProof ?p ?s) (= ?s (MP ? ?)))
            ?count))
```

Summary

- ✔ Metadata with inference
- ✔ No ontology impact
- ✔ Processing overhead ≈ proof complexity
- ✔ Combined meta and object-level queries
- ✔ No performance impact if proofs not needed
- ✘ Proof identity an issue – order dependence
  - ⊗ Requires commutative algebraic operators

Questions?