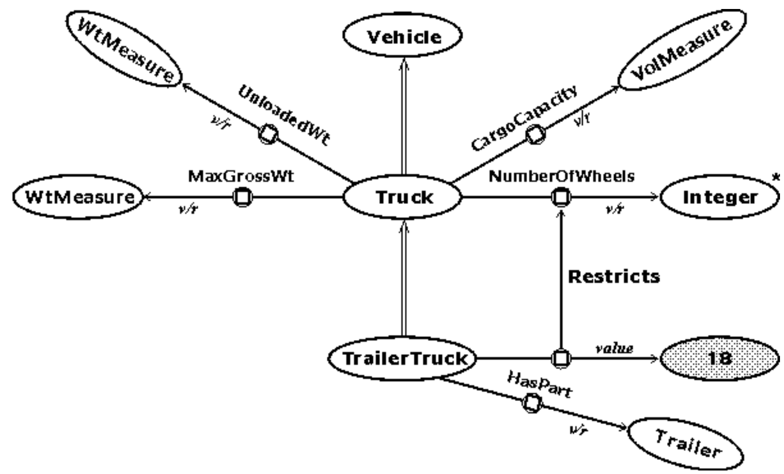
2. Object-Oriented Representation

CS227

Spring 2011

What is Object-Oriented?

$$\forall x\, P(x) \vee \forall x\, Q(x) \Rightarrow \forall x\, (P(x) \vee Q(x))$$
$$\exists x\, (P(x) \wedge Q(x)) \Rightarrow \exists x\, P(x) \wedge \exists x\, Q(x)$$
$$\exists x\, P(x) \wedge \forall x\, Q(x) \Rightarrow \exists x\, (P(x) \wedge Q(x))$$
$$\forall x\, P(x) \Rightarrow P(c)$$
$$P(c) \Rightarrow \exists x\, P(x)$$



- Flat representation
- Each sentence is self-contained
- Can be independently understood
- Information about an entity is scattered in multiple sentences

vs

- Sentences are grouped
- Structured and organized
- Usually a correspondence with the user interface
- Translatable to logic

Outline

- Background
  - Frames
  - Semantic Networks
  - Object-orientation
- An example object-oriented KR language
  - Inheritance reasoning
  - Other inference operations
- Applications

Frames

- When one encounters a new situation, one selects from memory a structure called a *Frame.* This is a remembered framework to be adapted to fit reality by changing details as necessary – Marvin Minsky 1974
- Example: Birthday Party

DRESS ——— SUNDAY BEST.
PRESENT —– MUST PLEASE HOST. MUST BE BOUGHT AND GIFT-WRAPPED.
GAMES ——— HIDE AND SEEK. PIN TAIL ON DONKEY.
DECOR ——— BALLOONS. FAVORS. CREPE-PAPER.
PARTY-MEAL–CAKE. ICE-CREAM. SODA. HOT DOGS.
CAKE ——— CANDLES. BLOW-OUT. WISH.
SING BIRTHDAY SONG.
ICE-CREAM —– STANDARD THREE-FLAVOR.

Reasoning Operations on Frames

- Expectation: How to select an initial frame to meet some conditions
  - Child’s birthday party
- Elaboration: How to select and assign sub-frames to represent additional details
  - North American birthday party vs an Asian Party
- Alteration: How to find a frame to replace one that does not fit well
  - No gifts allowed
- Novelty: What to do if no acceptable frame can be found?
- Learning: What frames should be stored or modified as a result of experience?

Semantic Network

- Directed labeled graphs used to represent concepts and the relationships between them
  - A concept refers to things in the application domain
    - Animal, Vertebra, Cat, etc

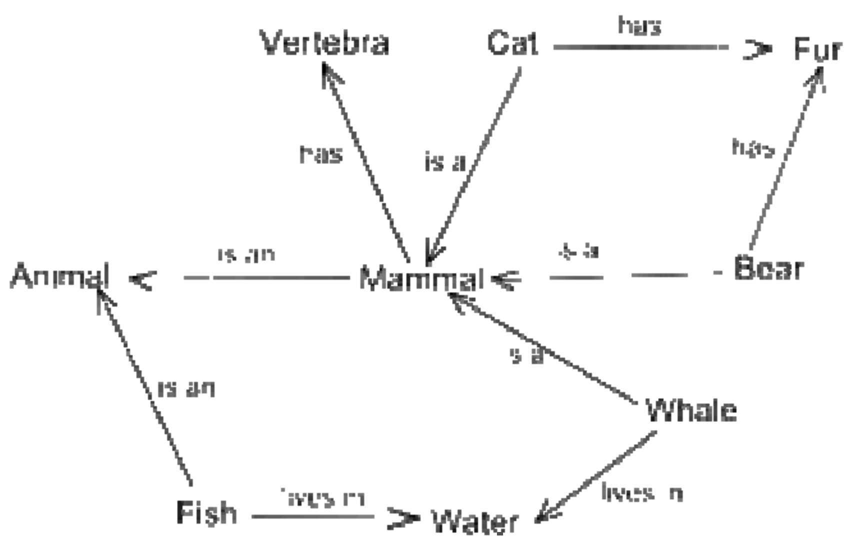
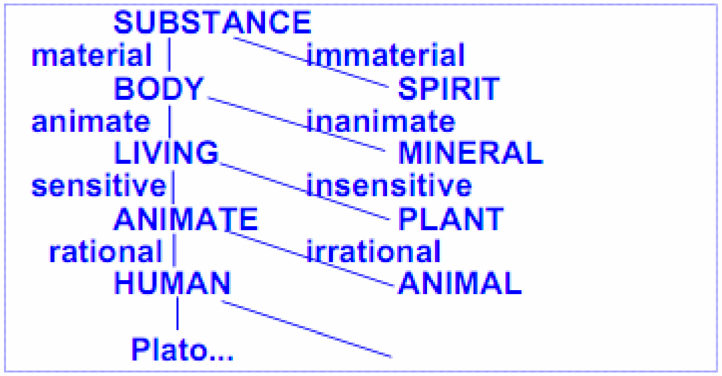


Image adapted from Wikipedia: http://en.wikipedia.org/wiki/Semantic_network

Semantic Network

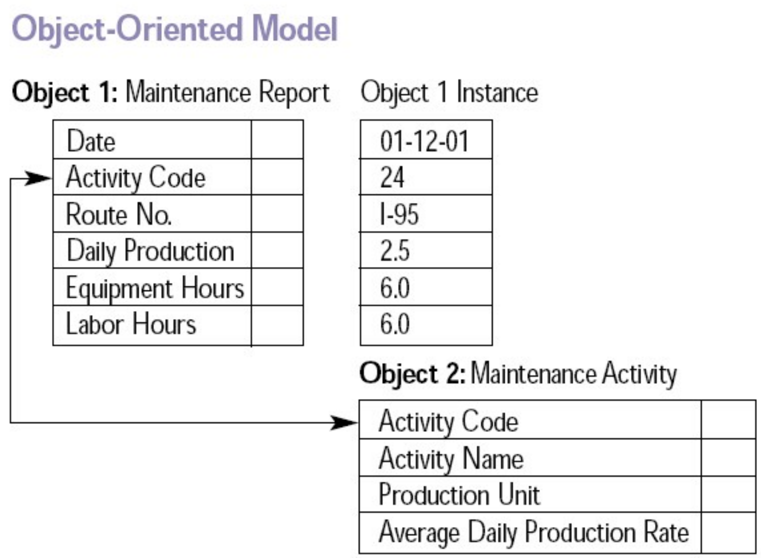
- Such representations have had a long distinguished history in philosophy and science
  - Prophyry's tree (3rd AD)
  - Charles Peirce's existential graphs (1890's) used in philosophy / logic
  - O. Stelz's concept hierarchies (1920's) – psychology
  - Ross Quillian's associative memory model (1966) – Psychology / computer science



Porphyry, 3rd century AD.

Object-Oriented Languages

- Many modern programming languages support features such as data abstraction and inheritance: Java
- Object databases represent information as objects and support object-oriented programming



Observation

- Graphical representations have several features in common
  - A hierarchy of classes
  - Classes have properties that can inherit
  - Facets provide further descriptors of values
- Such representations are pervasive in modern systems and have been formalized in a variety of ways
- In this lecture we will consider one such representation language

An Example Object-Oriented KR Language

- Loosely based on Open Knowledge Base Connectivity
  - Provides a knowledge model
    - The knowledge model was developed based on extensive survey of object-centered representation and reasoning systems under development in mid-nineties [Karp '92]
    - It incorporates common features found across a broad range of those systems
  - Defines a set of knowledge base operations
    - Define the basic inference operations that a object-oriented KR system must support
- Even though this work is over 10 years old, its simplicity and elegance is still compelling and relevant to modern systems

Knowledge Model

- A universe of discourse consists of all entities about which knowledge is to be expressed
- The universe of discourse always includes all constants of the following basic types:
  - Integers
  - Floating point numbers
  - Strings
  - Symbols
  - Lists
  - Classes
- The universe also includes logical constants: *true* and *false*

Class

- Classes are sets of entities
- All sets of entities are considered to be classes
  - A class is equivalent to a unary relation

Example:
Person is a class and represents set of all people

Individual

- Each of the entities in a class is said to be an instance of that class
- An individual is an entity that is not a set

Examples:

- John is an individual, and is not a set, and is an instance of Person

instance-of

- The class membership relation *instance-of* holds between an instance and a class is a binary relation that maps entities to classes

(<=> (*holds* ?C ?I) (*instance-of* ?I ?C))

- The relation type-of is defined as the inverse of relation *instance-of*

(<=> (*type-of* ?C ?I) (*instance-of* ?I ?C))

subclass-of

- A class CSub is a subclass of class CSuper if and only if all instances of Csub are also instances of CSuper

(<=> (subclass-of ?Csub ?Csuper)
  (forall ?I (=> (instance-of ?I ?Csub)
    (instance-of ?I ?Csuper))))

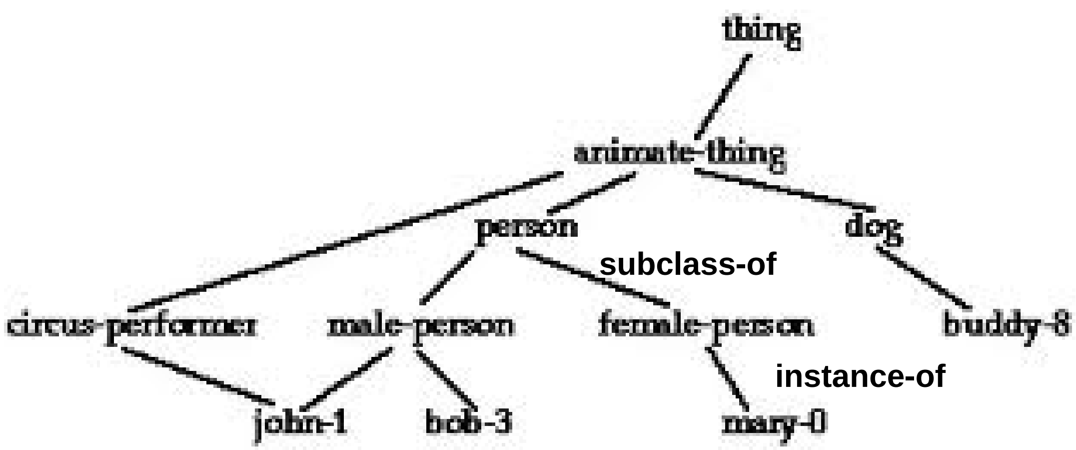


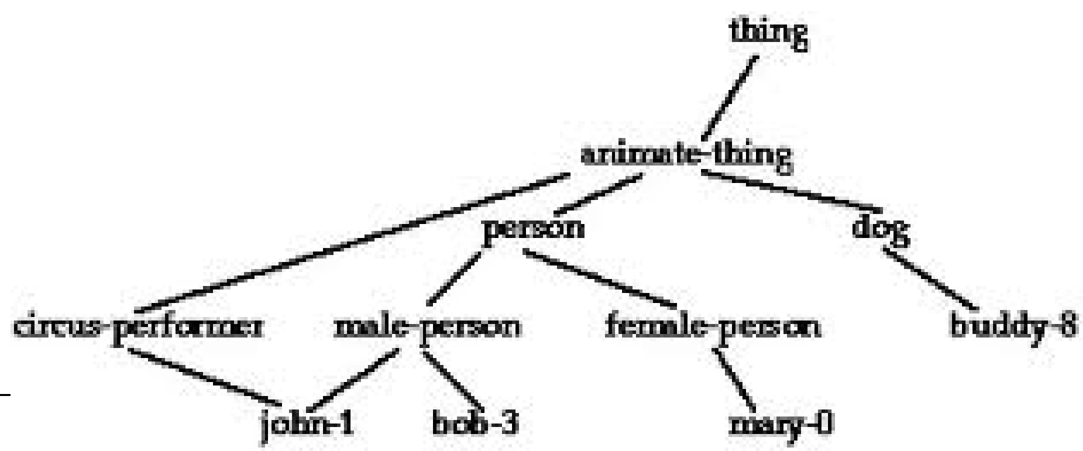
Image adapted from Chris Riesbeck (notes for CS325 at NWU)

subclass-of

- The subclass-of relationship is transitive
  - If A is a subclass of B, and B is a subclass of C, A is a subclass of C

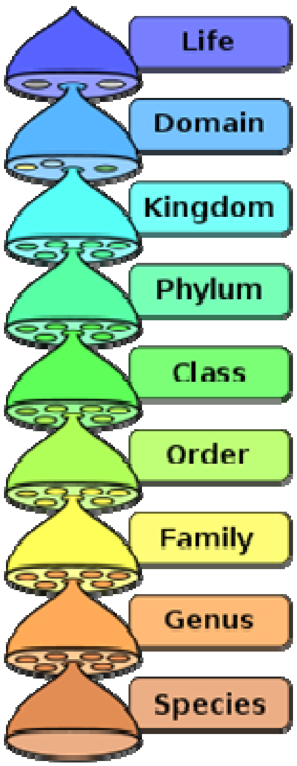
```
(=>
    (and (subclass-of ?a ?b)
         (subclass-of ?b ?c))
            (subclass-of ?a ?c)))
```

Male-person is a
subclass of animate-thing



Meta Class

- A class can be an instance of another class
  – A class that has another class as an instance is referred to as a *meta class*



Example:

Linnaen Taxonomy

Animal, Ecysozoan, Eumetzoa, …

Biological Classification

Kingdom, Phylum, Order, …

The class Animal is an instance of class Kingdom

Animal is an instance of Kingdom

Cow is not an instance of Kingdom

Picture credit: http://en.wikipedia.org/wiki/Kingdom_(biology)

Own Slots

- An entity has associated with it a collection of *own* slots
  - An *own* slot describes the direct properties of an entity
    - If the age of *John* is 42, then the value of the *own* slot *age* of *John* is 42
  - An *own* slot is equivalent to a binary relation
    - (age John 42)
  - Both classes and individuals can have *own* slots
    - A class *Person* has an own slot *average-age*: average of the age of all the instances of the class *Person*

Template Slots

- A class has associated with a collection of *template* slots
  - Template slots describe properties of the instance of a class
  - Own slots of a class describe the properties of the class itself

Example:

Class: Blue-Eyed-Person
template slot: eye-color
template-slot-value: Blue

Individual: John
instance-of: Blue-Eyed-Person
own slot: eye-color
own slot value: Blue

  - The values of template slots are ***inherited*** to instances as values of the corresponding own slot

Template Slots

- Formally, each value V of a template slot S of a class C represents the assertion that the relation template-slot-value holds for the relations S, the class represented by C, and the entity represented by V

  (template-slot-value S C V)

- The relation in turn holds between each instance I of class C and value V

  (S I V)

```
(=> (template-slot-value ?S ?C ?V)
    (=> (instance-of ?I ?C) (holds ?S ?I ?V))
```

Necessary vs Sufficient Properties

- Necessary properties of a class are necessarily true

Example:

A Person has an age, a parent, a height, …

- Sufficient properties are properties sufficient to conclude the class membership

Example:

A Blue-Eyed-Person is a Person with eye-color Blue

If an instance of unknown type had an age, it does not mean that it is a Blue-Eyed-Person

Primitive vs Defined Classes

- Primitive class is a class that only has necessary properties

Example: Person

- Defined class is a class that can be defined in terms of another class by giving it sufficient properties

Example: Blue-Eyed-Person

a Person with eye-color Red

Domain

- If a slot S has *domain* C, then every entity *I* that has a value for slot *S*, must be an *instance-of* C

Example:

The domain of slot *has-salary* is class *Employee*

```
(=> (DOMAIN ?S ?C)
        (=> (holds ?S ?F ?V) (instance-of ?F ?C))
        (=> (template-slot-value ?S ?F ?V)
                (or (= ?F ?C) (subclass-of ?F ?C))))
```

Range

- If a slot S has *range* C, then every entity *I* that has a value *V* for slot *S*, *V* must be an *instance-of* C

Example:

The range of slot *has-salary* is class *Amount-In-Dollars*

(=> (range ?S ?C)
(=> (holds ?S ?F ?V) (instance-of ?V ?C))
(=> (template-slot-value ?S ?F ?V) (instance-of ?V ?C)))

Facets

- An own slot can have associated with it an own facet that provides more information about that slots

Example:

John has exactly two Friends

John

own slots: has Friends

cardinality: 2

Facets

- An template slot can have associated with it template slots that inherit like the template slot values

Example:

Person has exactly 1 Father

Person

template slot: has-Parent

cardinality: 1

Default Values

- Each slot can have two kinds of values
  - Known true: values that are definitely known to be true
  - Default: values that are generally true but could be overridden

Example:

Class: Stanford Student
template slot: residence-city
default value: Stanford

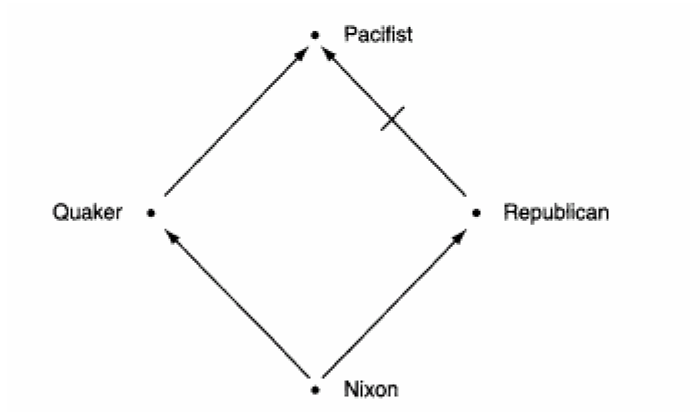
template slot: studies-at
known true: Stanford

Problems with Default Values

- When a class has more than one super class, its instances may inherit values that conflict with each other

Example:

- Nixon/Diamond example:
  - Nixon is a both Republican and Quaker, Republicans have a non-pacifist policy, and Quakers have a pacifist policy. One of the two values must be blocked from inheriting



Problems with Default Values

- When a class has more than one super class, its instances may inherit values that conflict with each other

Example:

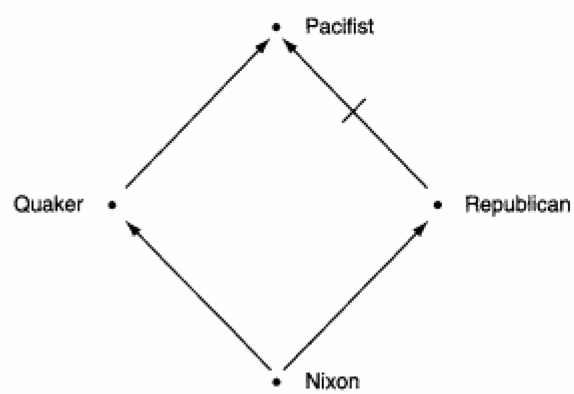
- Burgundy house example:
  - A Red-House has template-slot-value for color as Red
  - Burgundy House is a subclass of Red House
  - We want the value color of the roof to be overridden to burgundy

Methods for Dealing with Conflicting Values

- Skeptical Inheritance
- Override Inheritance

Skeptical Inheritance

- Inherit values only when the inherited values do not lead to contradiction
  - In the Nixon Diamond example, neither pacifist nor non-pacifist value will be inherited
  - The problem of contradiction is avoided
  - Nothing is done to resolve the contradiction



Override Inheritance

- A value specified at a more specific class overrides the value at a more general class
  - In the burgundy house example, the value of color specified for the more specific class overrides the value inherited from the super class
  - The values specified at a super-class are lost

Other Methods for Defaults

- Allow a designer to associate priorities with inherited values
- Provide heuristics to guess the correct value
  - Textbook chapter on inheritance reasoning has examples

Accessing the Knowledge Base

- Two kinds of interfaces
  - A set of function calls
  - An interface based on a logical language

Systematic Design of Functions

- For each kind of knowledge, provide: add, delete, put, replace

add-class-subclass
delete-class-subclass
put-class-subclass
replace-class-subclass

add-instance-type
delete-instance-type
put-instance-type
replace-instance-type

add-slot-value
delete-slot-value
put-slot-value
replace-slot-value

add-facet-value
delete-facet-value
put-slot-value
Replace-slot-value

Mandatory vs Optional

- A reasoning system must support a set of core operations called mandatory operations
- Rest of the operations can be defined in terms of the mandatory operations

Example:

Mandatory operation: get-class-subclasses

Optional operation: subclass-of-p

Question:

What is the core set of operations for a reasoning system?

Logical Interface

- Basic operations: Tell, Untell and Ask
- Assertion language: instance-of, subclass-of, template-slot-value,

Sentence	Effect of Executing (tell ⟨sentence⟩)
(`slot` frame value)	(**add-slot-value** frame slot value :slot-type :own)
(`facet` slot frame value)	(**add-facet-value** frame slot facet value)
(`instance-of` frame class)	(**add-instance-type** frame class)
(`type-of` class frame)	(**add-instance-type** frame class)
(`primitive` class)	Not a **tellable** sentence
(`template-slot-value` slot class value)	(**add-slot-value** class slot value :slot-type :template)
(`template-facet-value` facet slot class value)	(**add-facet-value** class slot facet value :slot-type :template)
(`class` class)	(**create-class** class)
(`individual` individual)	(**create-individual** individual)
(`slot-of` slot frame)	(**attach-slot** frame slot :slot-type :own)
(`facet-of` facet slot frame)	(**attach-facet** frame slot facet :slot-type :own)
(`template-slot-of` slot class)	(**attach-slot** slot class :slot-type :template)
(`template-facet-of` facet slot class)	(**attach-facet** class slot facet :slot-type :template)
(`subclass-of` class superclass)	(**add-class-superclass** class superclass)
(`superclass-of` superclass class)	(**add-class-superclass** class superclass)

Controlling the Inference

- Inference Level
  - Direct: return only the directly asserted values
  - Taxonomic: return only the inherited values
  - All-inferable: return all possible values
- Completeness
  - Return true if all values have been computed with certainty

Reasoning Operations on Frames

- Expectation: How to select an initial frame to meet some conditions
  - Child’s birthday party
    - Sufficient properties
- Elaboration: How to select and assign sub-frames to represent additional details
  - North American birthday party vs an Asian Party
    - Inheritance
- Alteration: How to find a frame to replace one that does not fit well
  - No gifts allowed
    - Defaults
- Novelty: What to do if no acceptable frame can be found?
  - Not covered
- Learning: What frames should be stored or modified as a result of experience?
  - Not covered

Limitation of Object-Oriented Representation

- Since such representations rely heavily on binary relations, it is difficult to represent ternary relations
  - A is between B and C
- A standard workaround is to reify the binary relation

A has

    own slot is-between: B, C

A has

    own slot is-between: Between-Objects-1

Between-Objects-1 has

    own slot objects: B, C

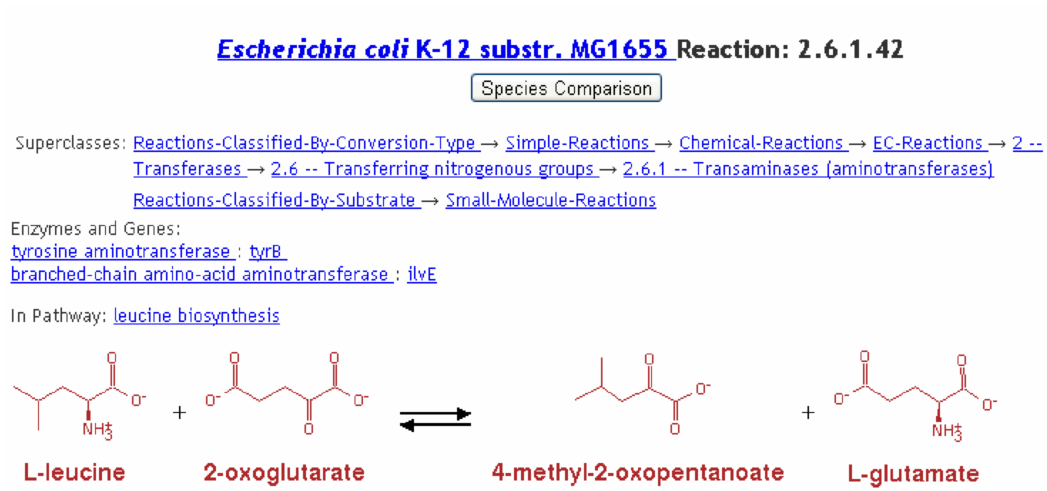
Limitations of Object Oriented Representation

- Negation cannot be represented uniformly
  - Jim does not have pneumonia
- Disjunction cannot be represented naturally
  - Jim has either Mumps or Rubella
- Quantification is not a part of the language
  - All of Jim’s diseases are infectious
- The semantics of the relations (as in any logic) are not intrinsic to the representation

Adapted from Mark Musen

Practical Uses

- The knowledge model introduced here is sufficient for several practical systems in use today

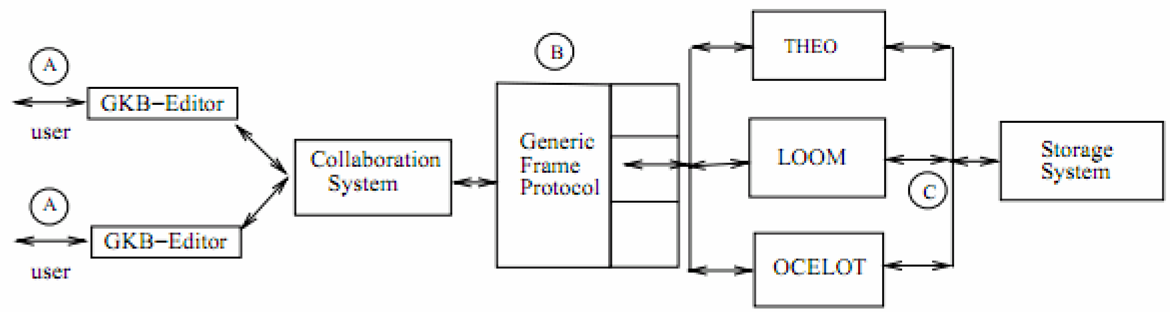


Literature based curation

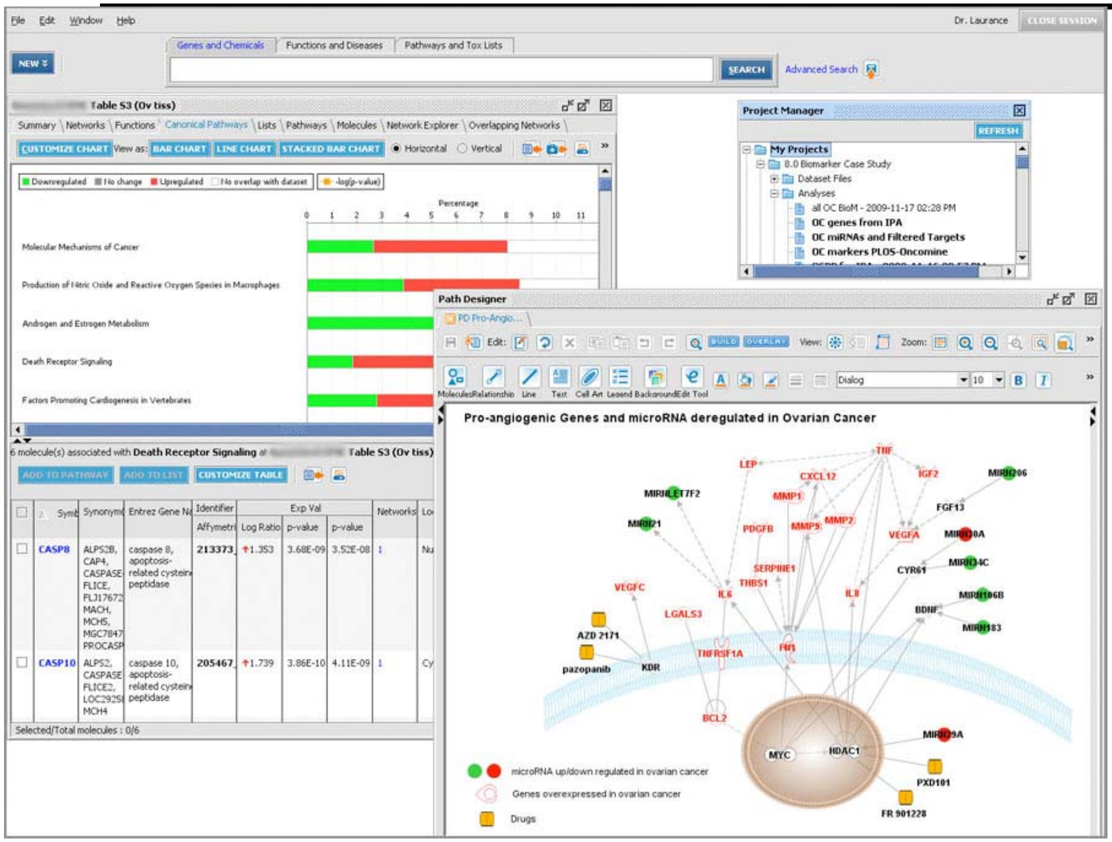
Literature-based curation of the entire genome, and of transcriptional regulation, transporters, and metabolic pathways.

www.ecocyc.org

Practical Uses



Practical Uses



Ingenuity Pathway Analysis

Search scientific literature

Analyze experimental data

Build dynamic pathway models

www.ingenuity.com

Contemporary Systems

- Representation languages based on description logics are popular
  - Next three lectures will be devoted to representation and reasoning with structured descriptions
    - Next two lectures will be given by Ron Brachman
  - A commonly used language that uses description logic is OWL
    - A language called OWL (http://www.w3.org/TR/owl2-overview/)
    - An API for accessing OWL knowledge (http://owlapi.sourceforge.net/)

Reading Material

- Required Reading
  - Chapter 8 of B&L Textbook: Frames
  - Chapter 10 of B&L Textbook: Inheritance
- Optional Reading
  - OKBC: A Programmatic Foundation for Knowledge Base Interoperability