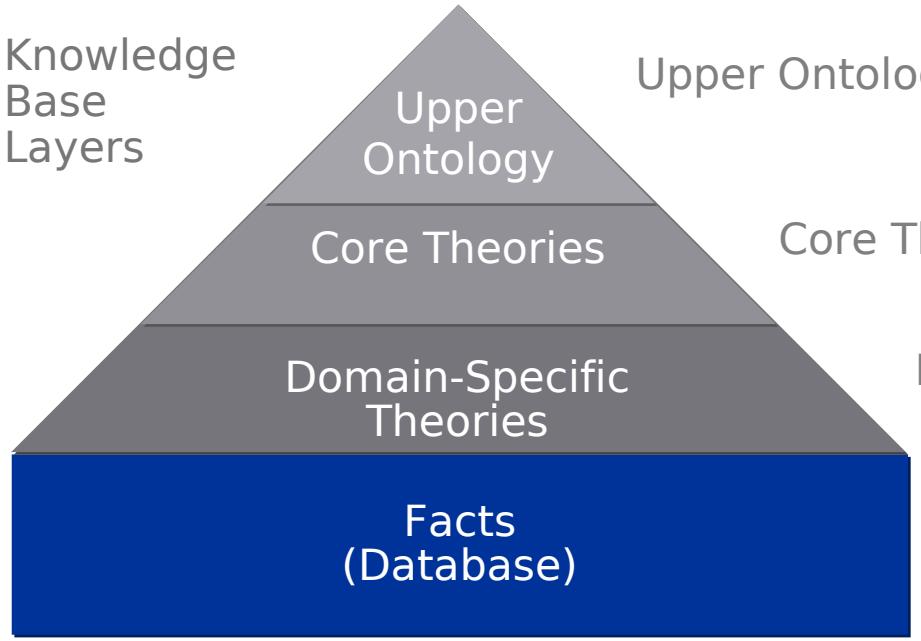
Survey of Knowledge Base Content

- ***Introduction***
- Fundamental Expression Types
- Top Level Collections
- Time and Dates
- Spatial Properties and Relations
- Event Types
- Information
- More Content Areas

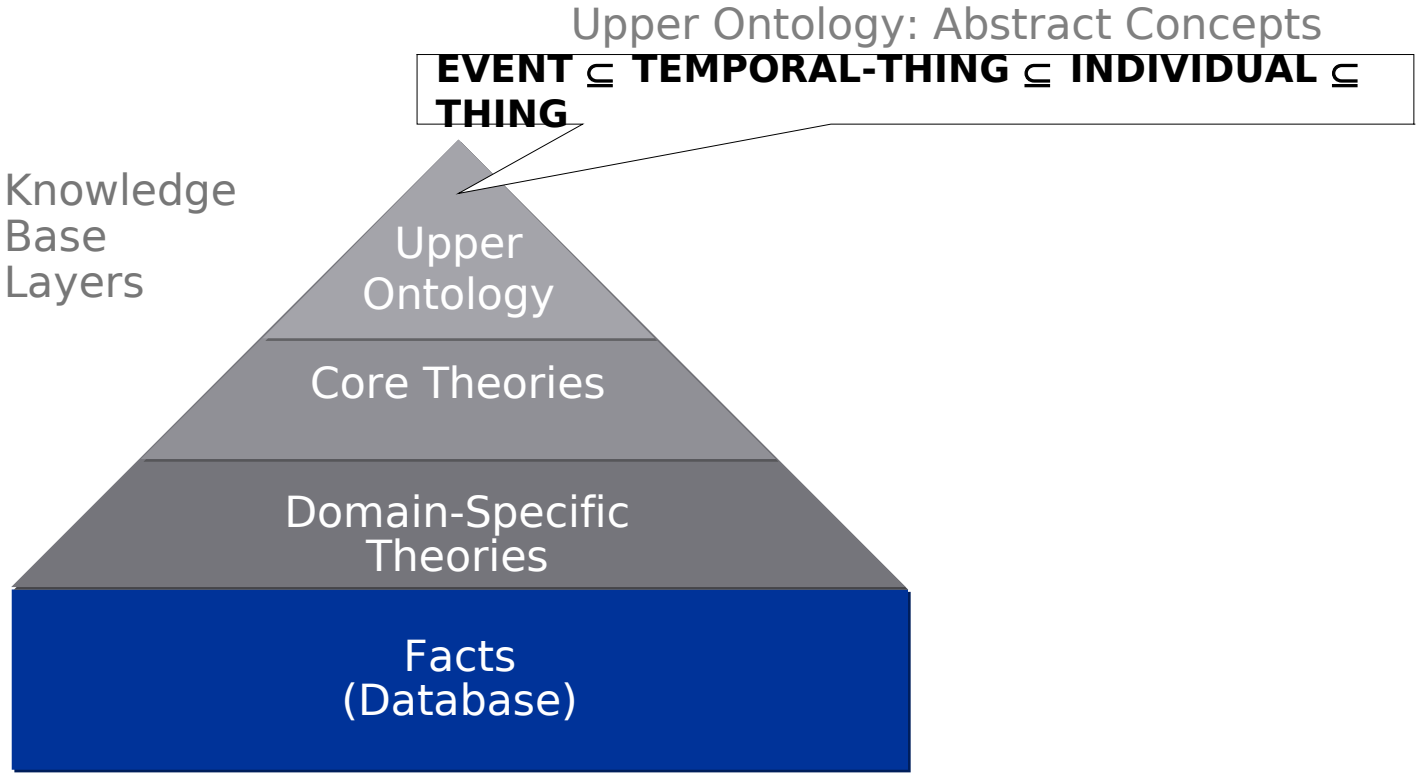
The Form and Content Of The Knowledge Base

- The main advantage of Cyc over other systems for representing knowledge is its use of a formal language in which inferential connections between concepts and statements are encoded in a machine accessible way.
- The content of the Knowledge Base comprises:
  - A vast taxonomy of concepts and relations
  - A rich formal representation of their interconnections

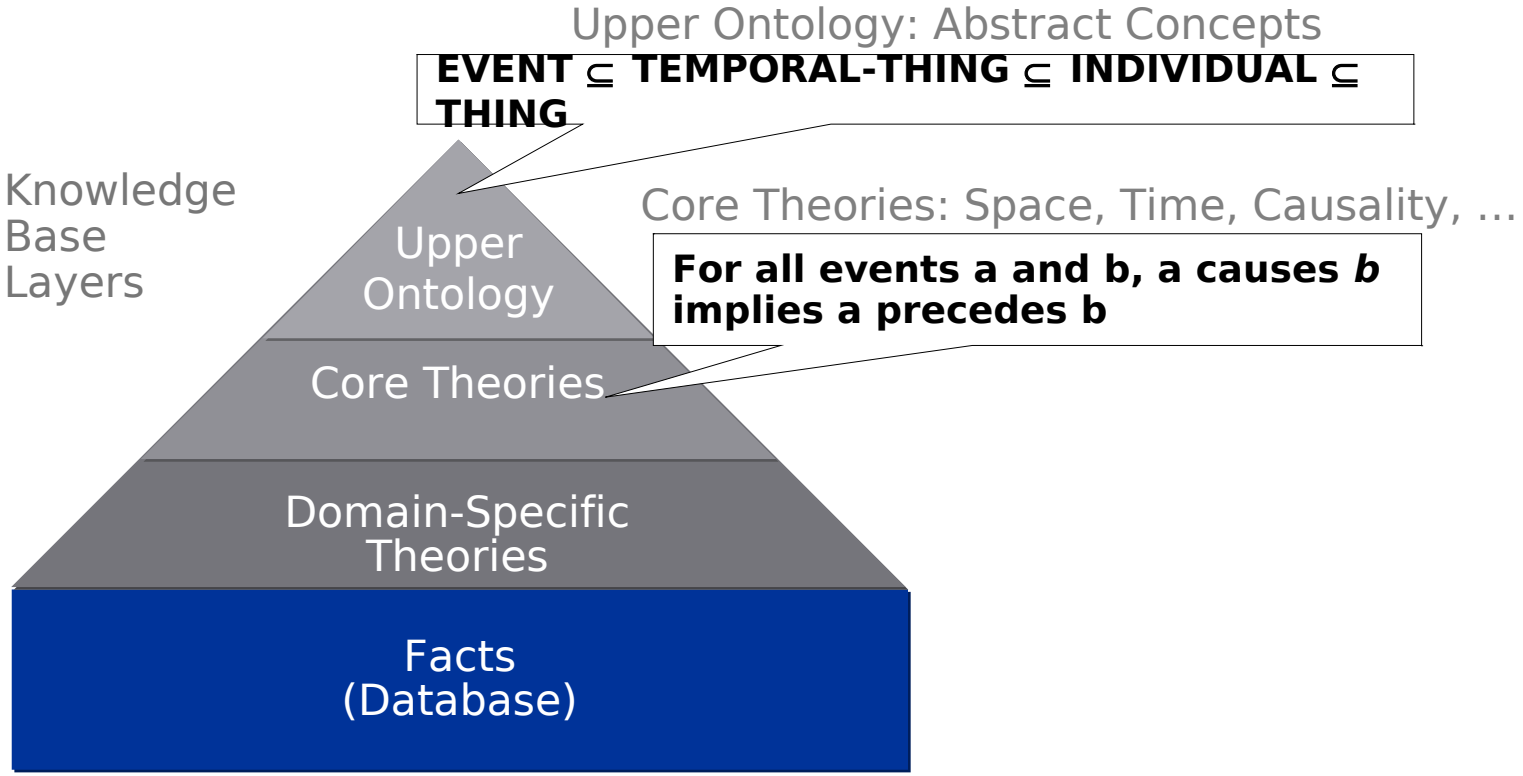
Arrangement, by Generality



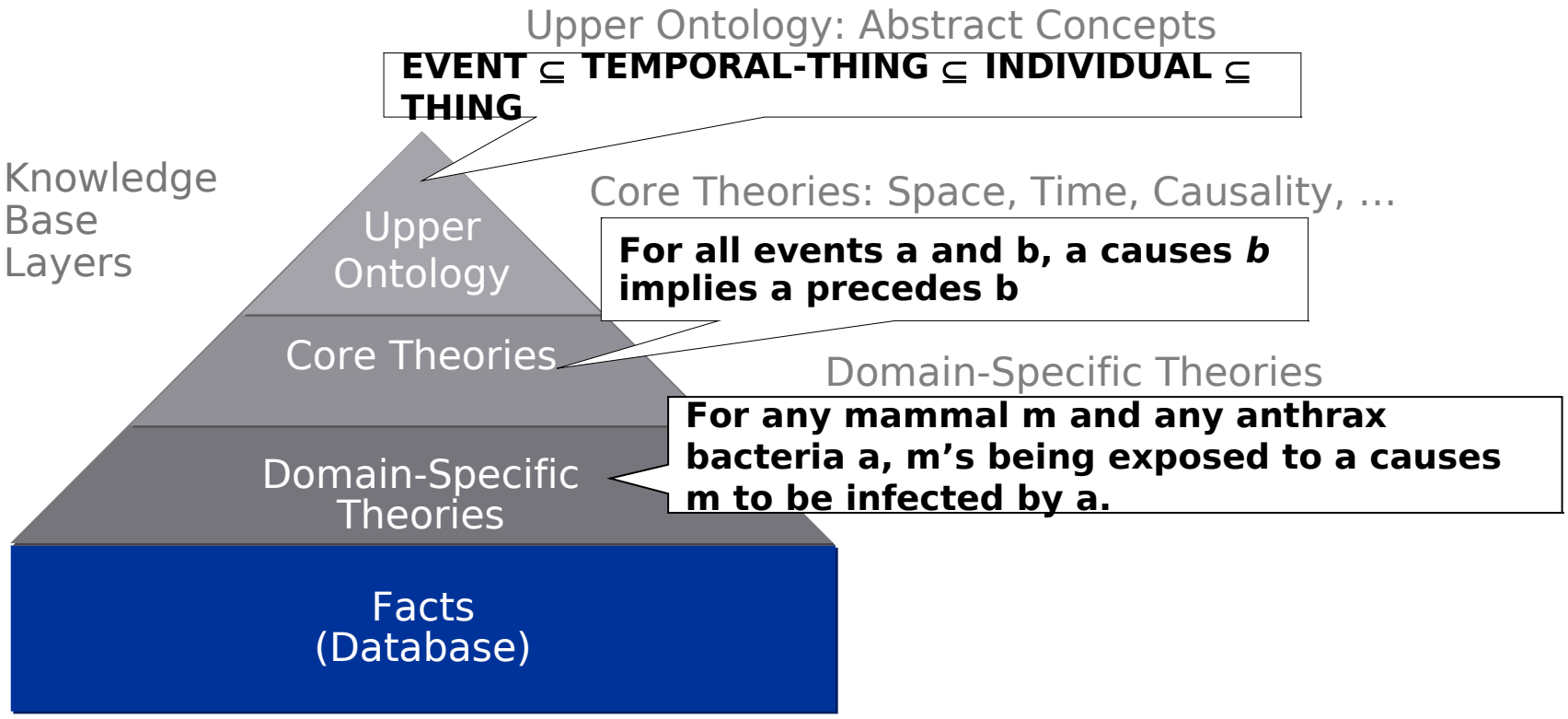
Arrangement, by Generality



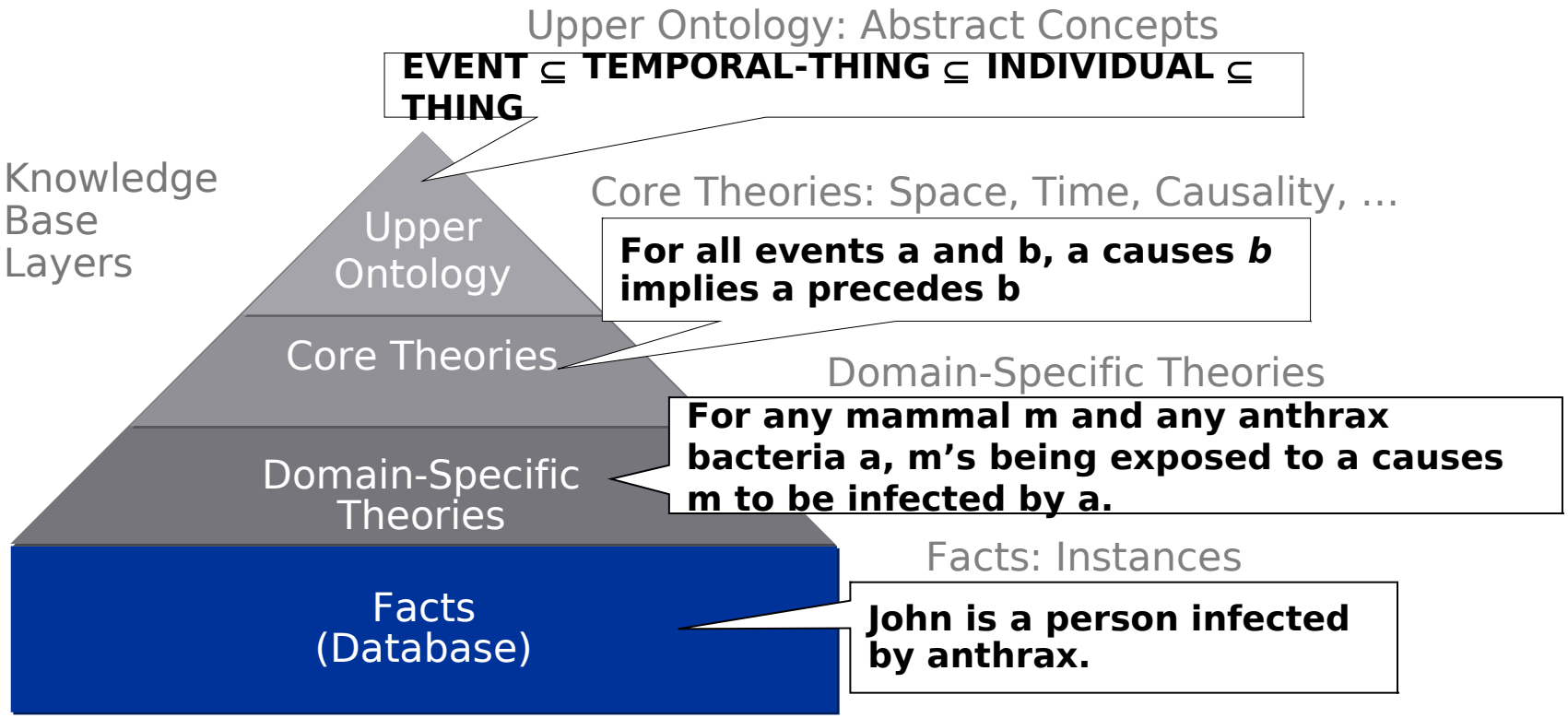
Arrangement, by Generality



Arrangement, by Generality



Arrangement, by Generality



Summary

- The KB is a vast taxonomy of concepts and relations
- CycL is a rich formal representation of their interconnections
- The KB can be thought of as made up of layers ordered by degree of generality

Survey of Knowledge Base Content

- Introduction
- ***Fundamental Expression Types***
- Top Level Collections
- Time and Dates
- Spatial Properties and Relations
- Event Types
- Information
- More Content Areas

Constants

Constants denoting individuals, collections, and collections of collections:

- #$GeorgeWBush, #$Sudan, #$0-TheDigit
- #$WorldLeader, #$Country
- #$AutomobileTypeByBrand, #$WineTypeByColor

Functions

Functions and Functional Expressions:

- #$PresidentFn
  (#$PresidentFn #$Mexico)
- #$MotherFn
  (#$MotherFn (#$PresidentFn #$UnitedStates))
- #$GroupFn
  (#$GroupFn #$Person)

Terms Used to Relate: #$isa and #$genls

Fundamental terms relating individuals to collections, and interrelating collections:

- #$isa

  (#$isa #$GeorgeWBush #$WorldLeader)

  (#$isa #$Cat #$OrganismClassificationType)

- #$genls

  (#$genls #$Cat #$Carnivore)

Terms Used to Relate: #$typeGenls

Fundamental terms relating individuals to collections, and interrelating collections:

- #$typeGenls

  (#$typeGenls #$OrganismClassificationType #$BiologicalLivingObject)

- (#$genls #$Cat #$Carnivore)
- (#$isa #$Cat #$OrganismClassificationType)

Terms Used to Relate: #$disjointWith

Fundamental terms relating individuals to collections, and interrelating collections:

- #$disjointWith

    (#$disjointWith #$Fish #$Mammal)

Other Terms Used to Relate

Other Relational Terms:

- #$biologicalRelatives
  (#$biologicalRelatives #$JerryLeeLewis #$JimmySwaggart)
- #$geographicalSubregions
  (#$geographicalSubregions #$UnitedStates #$Utah-State)
- #$greaterThan
  (#$greaterThan 25 3)
- #$orbits
  (#$orbits #$MoonOfEarth #$PlanetEarth)
- #$authorOfLiteraryWork-CW
  (#$authorOfLiteraryWork-CW #$HermanMelville #$MobyDickNovel)

Connecting Relational Terms: #$genlPreds and #$genlInverse

Terms Connecting Relational Terms:

- #$genlPreds
  (#$genlPreds #$geographicalSubregions #$physicalParts)
  (#$genlPreds #$physicalParts #$temporallyIntersects)
  (#$genlPreds #$father #$biologicalRelatives)

- #$genlInverse
  (#$genlInverse #$causes-EventEvent #$startsAfterStartingOf)
  (#$genlInverse #$father #$biologicalRelatives)

Predicates for Well-formedness: #$arity and #$argxIsa

Predicates For Describing Syntactic and Semantic Conditions for Well-Formedness:

- #$arity

  (#$arity #$biologicalMother 2)

- #$arg1Isa, #$arg2Isa, etc.

  (#$arg1Isa #$biologicalMother #$Animal)

  (#$arg2Isa #$biologicalMother #$FemaleAnimal)

Logical Connectives: #$or

#$or

(#$or
(#$isa #$ChrisX #$CollegeGraduate)
(#$isa #$ChrisX #$ComputerProgrammer)
(#$isa #$ChrisX #$Genius))

Logical Connectives: #$and and #$not

#$and, #$not

```
(#$not
   (#$and
      (#$isa #$ChrisX #$MalePerson)
      (#$isa #$ChrisX #$FemalePerson)))
```

Logical Connectives: #$implies

#$implies

(#$implies
    (#$isa #$ChrisX #$MalePerson)
    (#$not (#$isa #$ChrisX #$FemalePerson)))

Quantifiers: #$forAll

#$forAll

```
(#$forAll ?COUNTRY
   (#$forAll ?PERSON
      (#$implies
         (#$and
           (#$isa ?COUNTRY #$Superpower)
           (#$headsGovernment ?COUNTRY ?PERSON))
      (#$hasStatus ?PERSON #$WorldLeader))))
```

Quantifiers: #$thereExists

#$thereExists

```
(#$forAll ?ANIMAL
    (#$implies
        (#$isa ?ANIMAL #$Vertebrate)
        (#thereExists ?PART
            (#$and
                (#$isa ?PART #$Tongue)
                (#$anatomicalParts ?ANIMAL ?PART)))))
```

Summary

- Constants denote individuals, collections, and collections of collections
- Functions and Functional Expressions denote terms
- Terms used to relate: #$isa, #$genls, #$typeGenls, #$disjointWith
- Connecting Relational Terms: #$genlPreds and #$genlInverse
- Predicates For Describing Well-Formedness: #$arity and #$argxIsa
- Logical Connectives and Quantifiers

Survey of Knowledge Base Content

- Introduction
- Fundamental Expression Types
- ***Top Level Collections***
- Time and Dates
- Spatial Properties and Relations
- Event Types
- Information
- More Content Areas

Some Top Level Collections

#$Dog (the collection of all dogs)

#$isa: #$OrganismClassificationType
#$BiologicalTaxon
#$BiologicalSpecies
#$DomesticatedAnimalType

#$genls: #$CanineAnimal

But these are only the immediate categories

45 Collections of which #$Dog" is a Specialization

Agent Agent-Generic AirBreathingVertebrate Animal AnimalBLO BilateralObject BiologicalLivingObject CanineAnimal Carnivore CarnivoreOrder ChordataPhylum Coelomates Container-Underspecified Dog EukaryoticOrganism Eutheria FrontAndBackSidedObject Heterotroph HexalateralObject Homeotherm HumanScaleObject Individual IndividualAgent LeftAndRightSidedObject Location-Underspecified Mammal NaturalTangibleStuff NonPersonAnimal OrganicStuff Organism-Whole PartiallyTangible PerceptualAgent Region-Underspecified SentientAnimal SolidTangibleThing SomethingExisting SpatialThing SpatialThing-Localized System-Generic TemporalThing TerrestrialOrganism Thing TopAndBottomSidedObject Trajector-Underspecified Vertebrate

11 Collections of which #$Dog is an Instance

#$OrganismClassificationType
#$ConventionalClassificationType_
#$ExistingObjectType
#$TemporalStuffType
#$ObjectType
#$Collection
#$SetOrCollection
#$MathematicalThing
#$MathematicalOrComputationalThing
#$Intangible
#$PartiallyIntangible
#$Thing

Summary

- The collections hierarchy
- Specialization (#$genls)
- Instance (#$isa)

Survey of Knowledge Base Content

- Introduction
- Fundamental Expression Types
- Top Level Collections
- ***Time and Dates***
- Spatial Properties and Relations
- Event Types
- Information
- More Content Areas

Functions Which Return Time Intervals

#$YearFn, #$DayFn, #$MinuteFn, #$SecondFn

Example:

(#$YearFn 2000)

Functions Which Return Time Intervals: Composite Expressions

#$YearFn, #$DayFn, #$MinuteFn, #$SecondFn

Example:

```
(#$SecondFn 59
  (#$HourFn 23
    (#$DayFn 31
      (#$MonthFn #$December
        (#$YearFn 2000)))))
```

Time As A Quantity

Some Functions Denoting Quantities of Time:

#$YearsDuration, #$HoursDuration, #$SecondsDuration

Example:

(#$duration (#$YearFn 2000) (#$YearsDuration 1))

Relations Between Temporal Things

- #$temporalBoundsIntersect
- #$temporallyIntersects
- #$startsAfterStartingOf
- #$endsAfterStartingOf
- #$endsAfterEndingOf
- #$startingDate
- #$temporallyContains
- #$temporallyCooriginating
- #$temporalBoundsContain
- #$temporalBoundsIdentical
- #$startsDuring
- #$overlapsStart
- #$startingPoint
- #$simultaneousWith
- #$after

Relations of Types of Intervals

- #$TemporalStuffType
- #$TemporalObjectType
- #$TemporallyDisjointIntervalType
- #$CyclicalIntervalGroupType

Relations of Types of Intervals

- #$includedInIntervalType
- #$subsumesIntervalType
- #$intersectsIntervalType
- #$subsumedByIntervalType
- #$followingIntervalType

Summary

- Functions which return time intervals
- Time as a quantity
- Relations between temporal things
- Relations of time intervals

Survey of Knowledge Base Content

- Introduction
- Fundamental Expression Types
- Top Level Collections
- Time and Dates
- ***Spatial Properties and Relations***
- Event Types
- Information
- More Content Areas

Spatial Properties and Relations

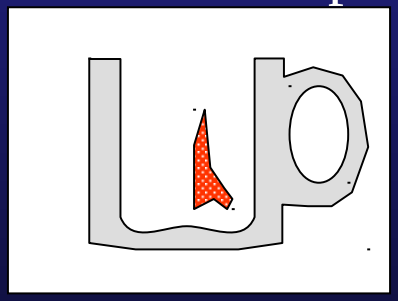
- Surfaces, Portals and Cavities
- Shape Attributes (63)
- Types of Spatial Symmetry
- Direction and Orientation Vocabulary
- Relative Positions of Objects
- Nearness and Location
- Being Between
- ‘In-’ Predicates (~ 60)
- Connections Predicates (~ 65)
- Mereological Relations

Senses of ‘In’

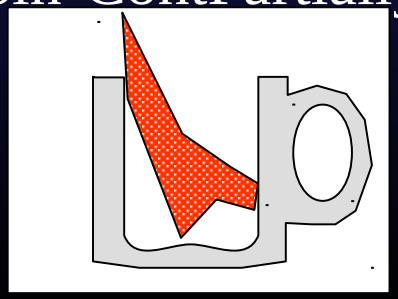
- Can the inner object leave by passing between members of the outer group?
  - Yes -- Try #$in-Among

Senses of ‘In’

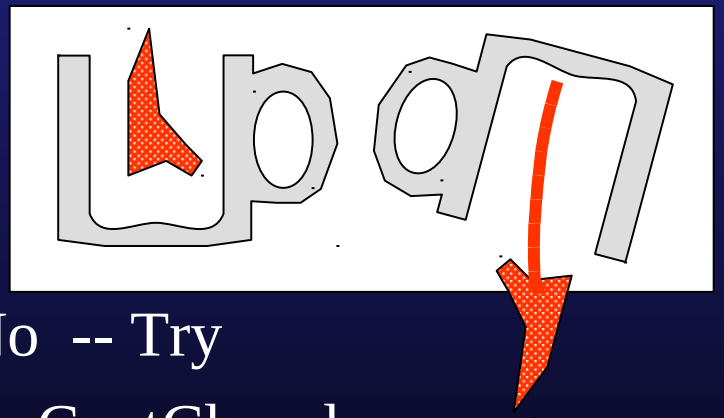
- Does part of the inner object stick out of the container?
  - None of it. -- Try #$in-ContCompletely



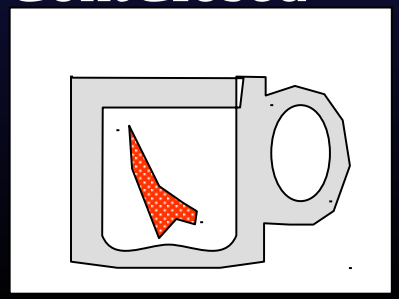
  - Yes -- Try #$in-ContPartially



- If the container were turned around could the contained object fall out?
  - Yes -- Try #$in-ContOpen



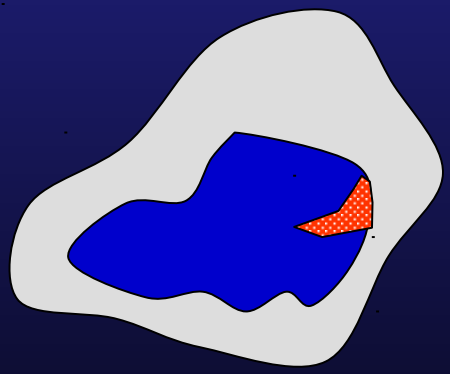
  - No -- Try #$in-ContClosed



Senses of ‘In’

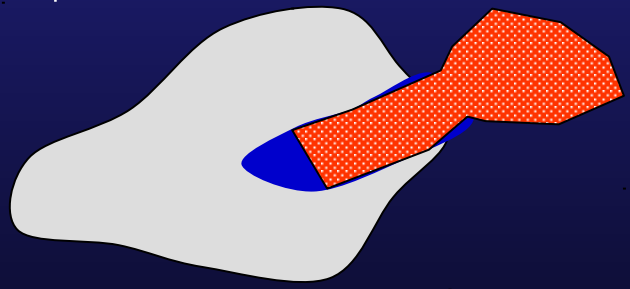
Is it attached to the inside of the outer object?

- Yes -- Try #$connectedToInside



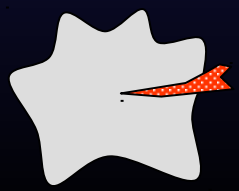
Can it be removed, if enough force is used, without damaging either object?

- Yes -- Try #$in-Snugly or #$screwedIn



Does the inner object stick into the outer object?

- Yes -- Try #$sticksInto



Senses of ‘Part’

- #$parts
- #$intangibleParts
- #$subInformation
- #$subEvents
- #$physicalDecompositions
- #$physicalPortions
- #$physicalParts
- #$externalParts
- #$internalParts
- #$anatomicalParts
- #$constituents
- #$ingredients

Organizations

- #$governingBody
- #$WholeOrganizationFn
- #$parentCompany
- #$subOrgs-Command
- #$subOrgs-Permanent
- #$subOrgs-Temporary
- #$subOrgs-OnlyDuringOperation
- #$physicalQuarters
- #$hasHeadquartersIn Country
- #$officeInCountry
- #$memberTypes
- #$organizationHead
- #$PolicyFn

Summary

- Senses of ‘In’
  - #$in-Among
  - #$in-ContCompletely, #$in-ContPartially, #$in-ContClosed, #$in-ContOpen
  - #$connectedToInside, #$in-Snugly, #$screwedIn, #$sticksInto
- Senses of ‘Part’
- Organizations

Survey of Knowledge Base Content

- Introduction
- Fundamental Expression Types
- Top Level Collections
- Time and Dates
- Spatial Properties and Relations
- ***Event Types***
- Information
- More Content Areas

Some Events Types

- #$PhysicalStateChangeEvent
- #$TemperatureChangingProcess
- #$BiologicalDevelopmentEvent
- #$ChangingDeviceState
- #$CuttingNails
- #$Cracking
- #$Carving
- #$ShapeChangeEvent
- #$MovementEvent
- #$GivingSomething
- #$DiscoveryEvent
- #$Buying
- #$Thinking
- #$Baking
- #$Singing
- #$PumpingFluid

Roles and ActorSlots (the world's largest collection)

- Agency or initiating an event
- Objects acted on or changed
- Objects created or destroyed
- Facilitating objects or stuff in an event
- Slots of motion and location
- Instruments
- Beneficiary/maleficiary
- Specialized actor roles, like #$plaintiffs

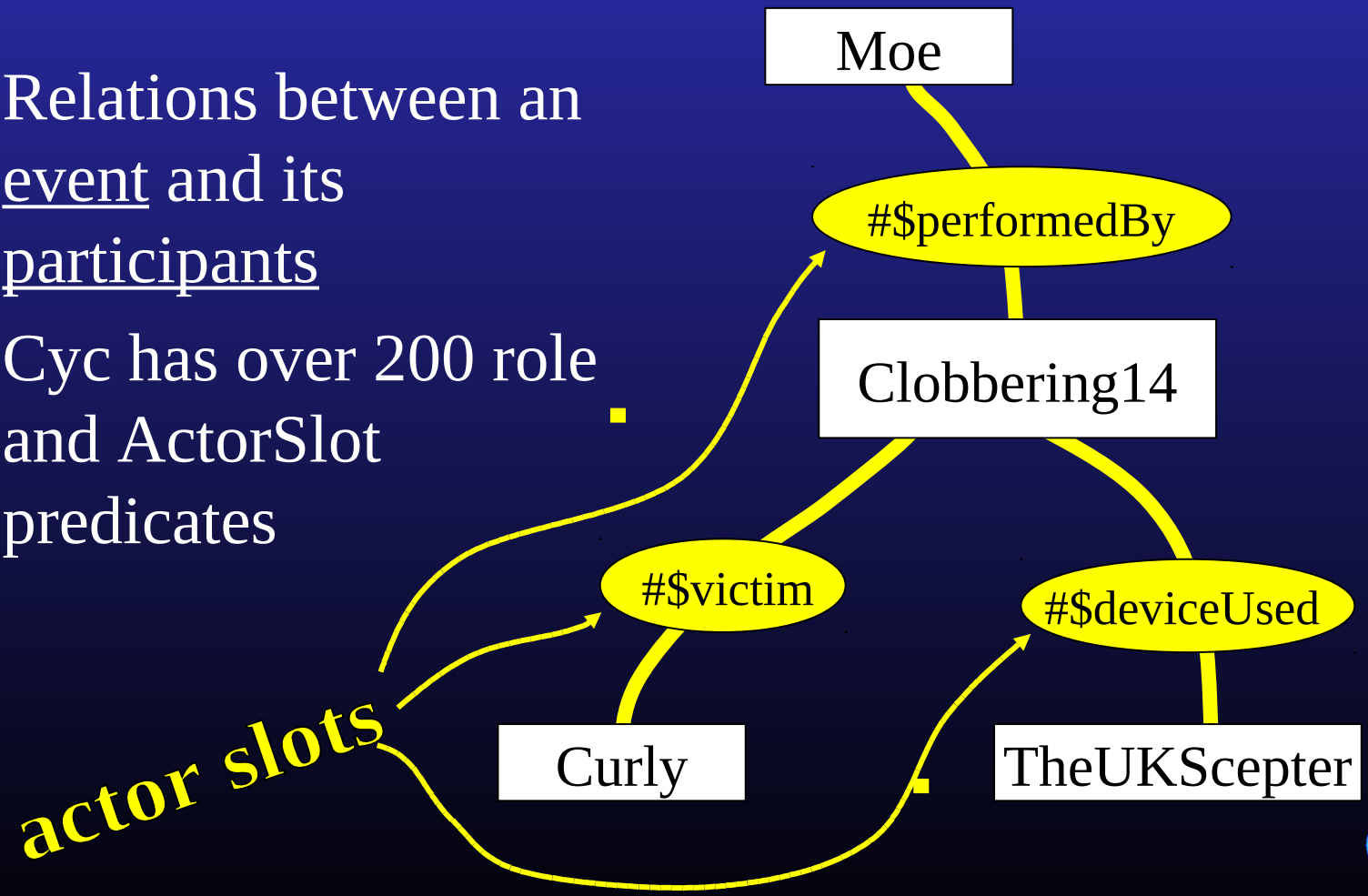
Roles and ActorSlots

“Moe clobbered Curly with the British scepter.”

Roles and ActorSlots

"Moe clobbered Curly with the British scepter."

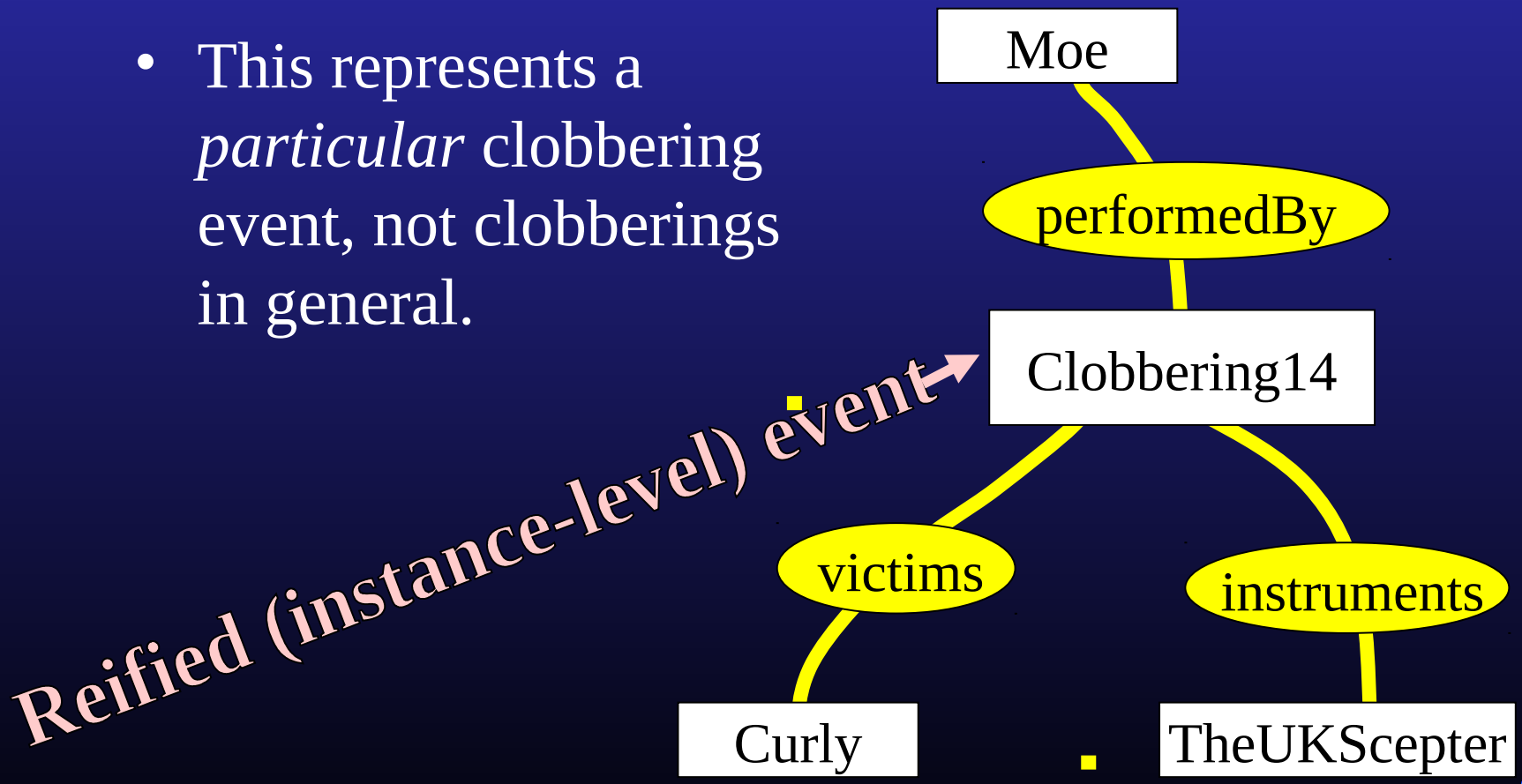
- Relations between an event and its participants
- Cyc has over 200 role and ActorSlot predicates



Roles and ActorSlots

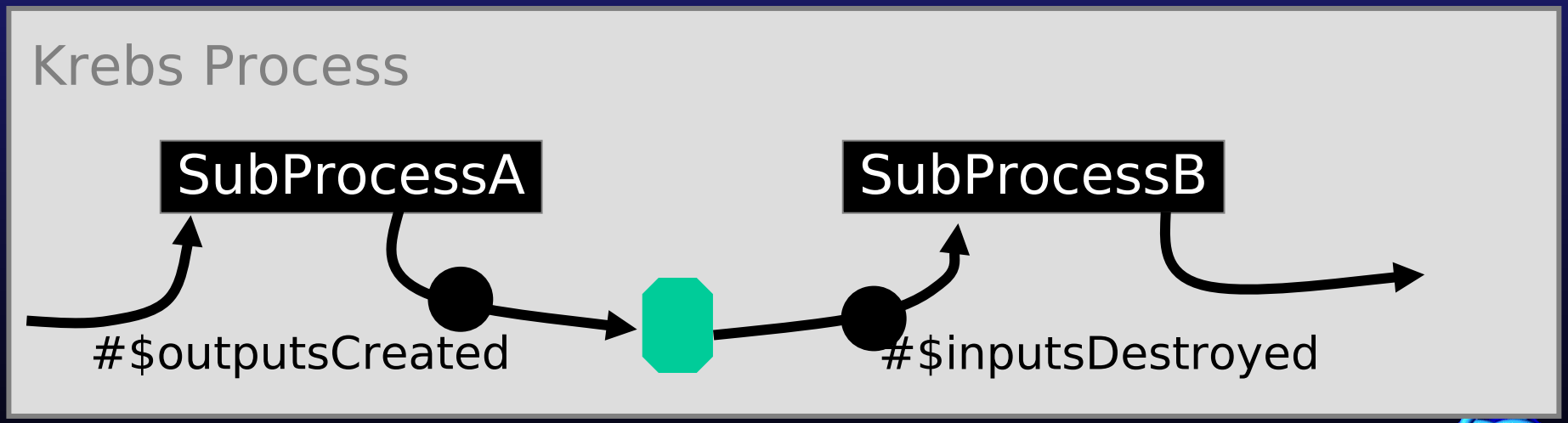
"Moe clobbered Curly with the British scepter."

- This represents a *particular* clobbering event, not clobberings in general.



Roles in events and subevents

- A product of one subevent of the Krebs Process is the input to another.
- Hence, different ActorSlot predicates.



Summary

- Cyc has a large variety of predicates for representing roles preformed in events and the actors who perform them.
- Events are represented by relating actors to the event.
- The product of one sub-event in the input to another.

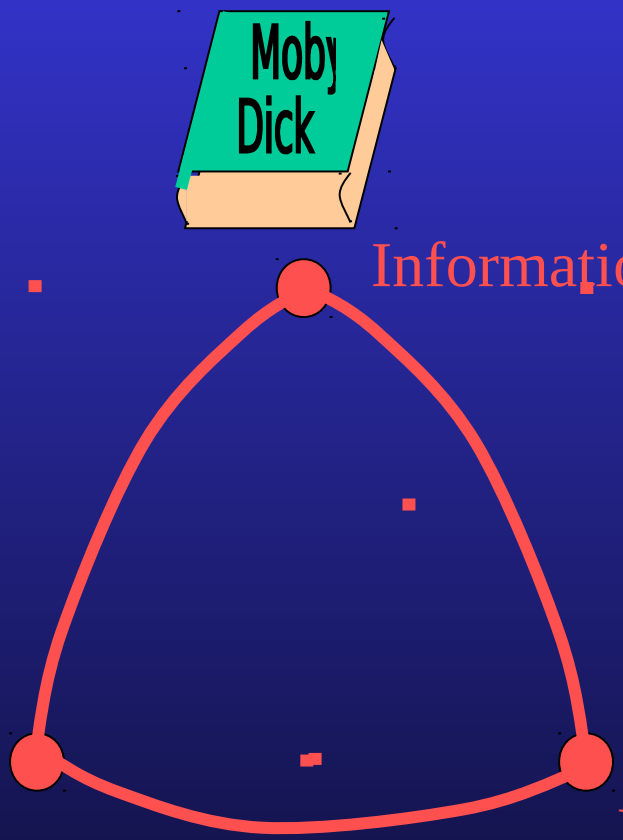
Survey of Knowledge Base Content

- Introduction
- Fundamental Expression Types
- Top Level Collections
- Time and Dates
- Spatial Properties and Relations
- Event Types
- ***Information***
- More Content Areas

Information

- Information-Bearing Things
  - Books, web-page copies, radio broadcasts, utterances
- Abstract strings, characters
- Propositional Content
- Conceptual Works

What is “Moby Dick” ?



InformationBearingThing
(IBT)

AbstractInformationStructure
(AIS)

PropositionalInformationThing
(PIT)

“‘ T i s M o b y D i c k !”

```
(#$thereExists ?SEE
  (#$and
    (#$isa ?SEE Seeing)
    (#$objectPerceived ?SEE #$MobyDick)
    (#$perceiver ?SEE #$CaptainAhab)))
```

InformationBearingThing
(IBT)

Problem: These are not what people usually ask about.

(and
(isa ?SEE Seeing)
(objectPercieved ?SEE MobyDick)
(perceiver ?SEE CaptainAhab))

“‘ T i s M o b y
D i c k !”

AbstractInformationStructure
(AIS)

PropositionalInformationThing
(PIT)

#$InformationBearingThing
(IBT)

#$ConceptualWork
(CW)

#$MobyDickTheBook-CW

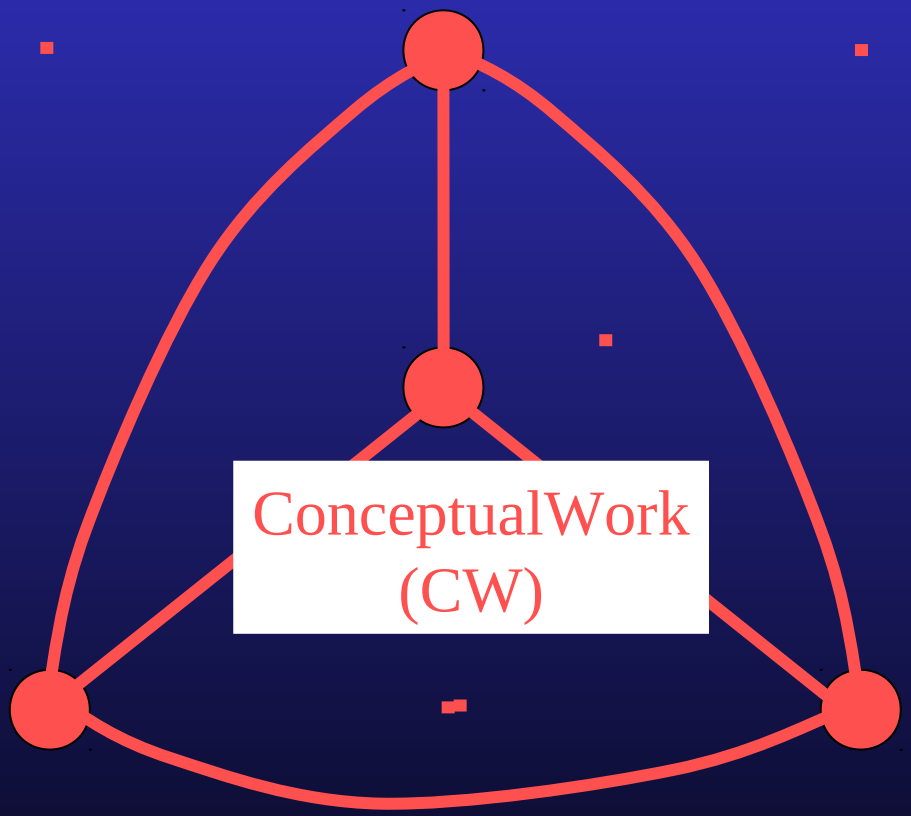
(and
(isa ?SEE Seeing)
(objectPercieved ?SEE MobyDick)
(perceiver ?SEE CaptainAhab))

“‘ T i s M o b y
D i c k !”

#$AbstractInformationStructure
(AIS)

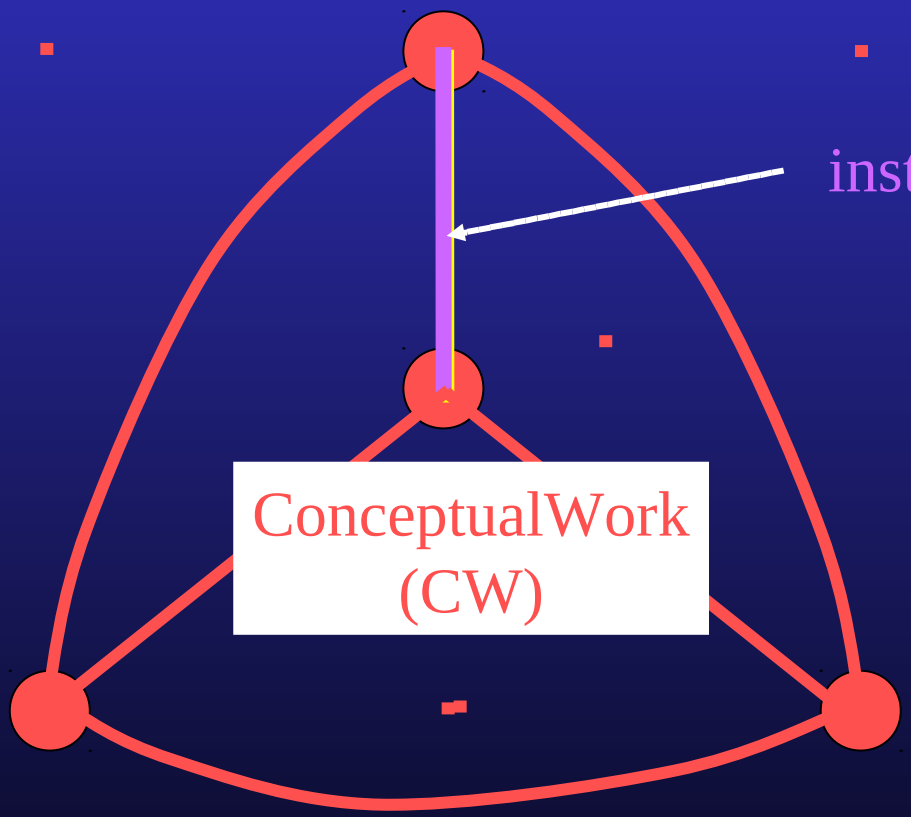
#$Proposition and
#$PropositionalInformationThing
(PIT)

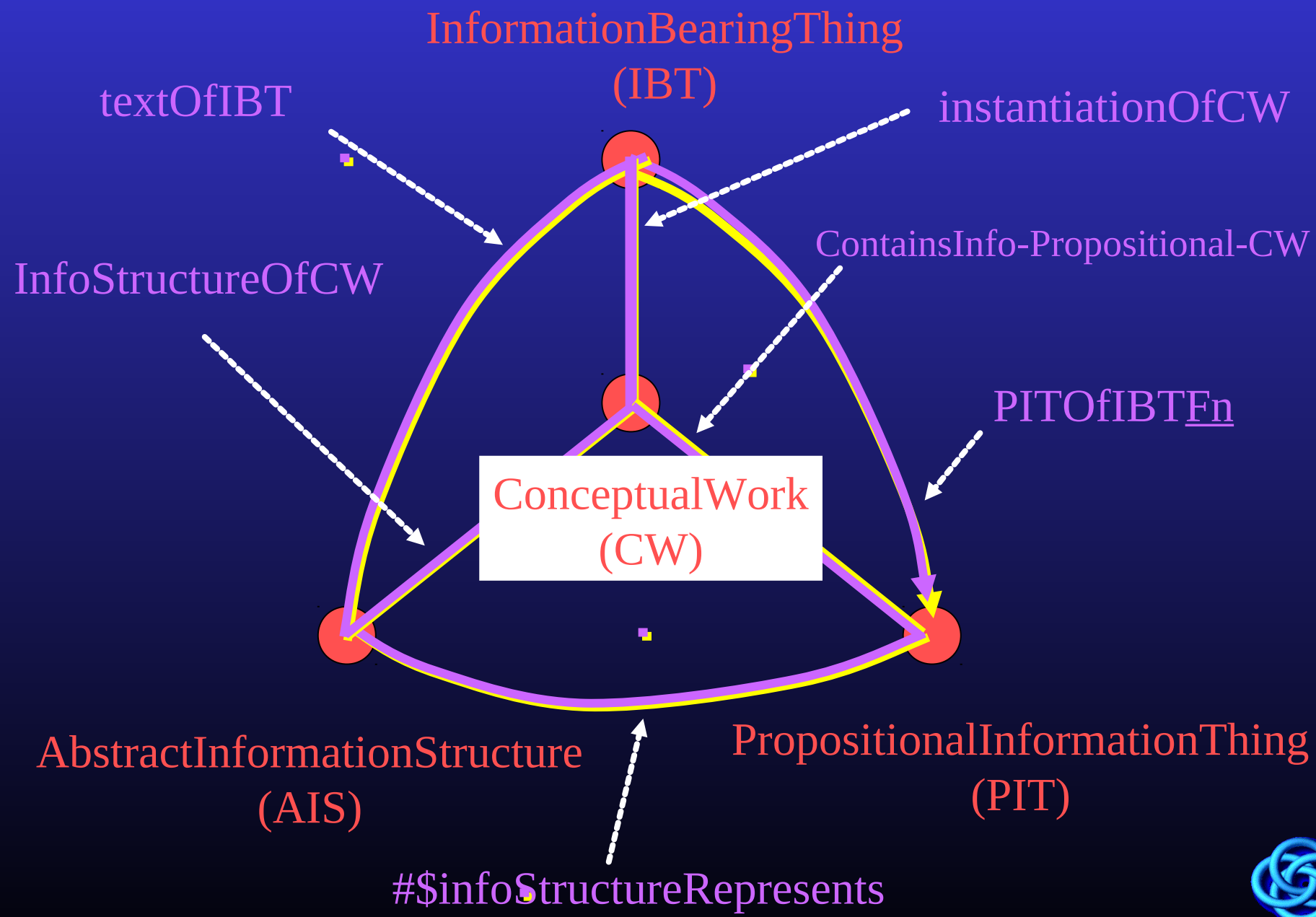
InformationBearingThing
(IBT)



AbstractInformationStructure
(AIS)

PropositionalInformationThing
(PIT)





Summary

- InformationBearingThing
- AbstractInformationStructure
- PropositionalInformationThing
- ConceptualWork
- Relating these categories

Survey of Knowledge Base Content

- Introduction
- Fundamental Expression Types
- Top Level Collections
- Time and Dates
- Spatial Properties and Relations
- Event Types
- Information
- ***More Content Areas***

Emotion

- Feeling Attributes Types
  - #$Abhorrence
  - #$Adulation
  - #$Relaxed-Feeling
  - #$Gratitude
  - #$Anticipation-Feeling
  - Over 120 of these
- Relations Pertaining to emotions
  - #$contraryFeelings
  - #$feelsTowardsObject
  - #$appropriateEmotion
  - #$feelsTowardsPersonType
  - #$actionExpressesFeeling

Propositional Attitudes

Relations Between Agents and Propositions

- #$goals
- #$intends
- #$desires
- #$hopes
- #$expects
- #$beliefs
- #$opinions
- #$knows
- #$rememberedProp
- #$perceivesThat
- #$seesThat
- #$tastesThat

Biology

- Organisms classified by:
  - Taxon
  - Habitat
  - Source of Nutrients
- Some scientific, #$ChordataPhylum, some not, #$Worm
- Organism Anatomy
  - Gross Anatomy
  - Cell biology
  - Physiological Processes
- Life stages

Materials

- Common Substances
- Attributes of Materials
- States Of Matter
- Solutions
- Electrical Conductivity
- Thermal Conductivity
- Structural Attributes
- Tangible Attributes

Devices

- Specializations Of #$PhysicalDevice
- Device States
- Device Actions
- Device Predicates
- Device Purposes

Food

- Food Types
- Edibility
- Preparing food
- Consuming food
- Hunger

Weather

- Weather Attributes
  - #$ClearWeather
  - (#$LowAmountFn #$Raininess)
- Weather Events
  - #$TornadoAsEvent
  - #$SnowProcess
- Weather Objects
  - #$CloudInSky
  - #$TornadoAsObject

Geography

- Geopolitical Entities
- Addresses
- Specific Ethnic and Language information
- Borders
- Districts, States, etc.
- Seas, islands, straits,etc.

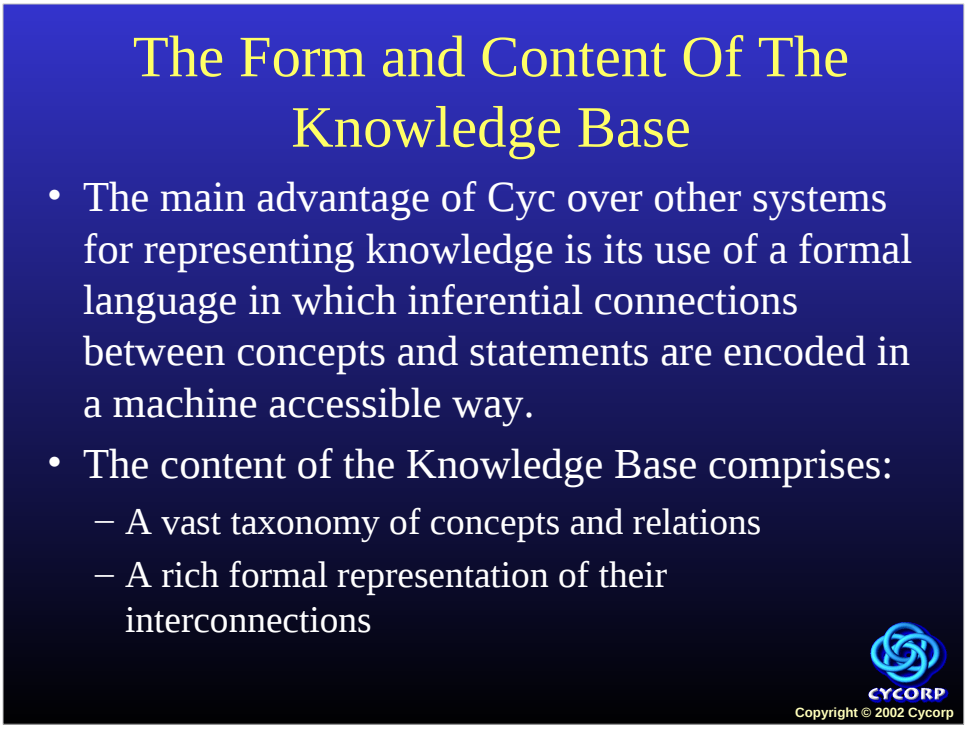
Summary

- Emotion
- Propositional Attitudes
- Biology
- Materials
- Devices
- Food
- Weather
- Geography

Survey of Knowledge Base Content

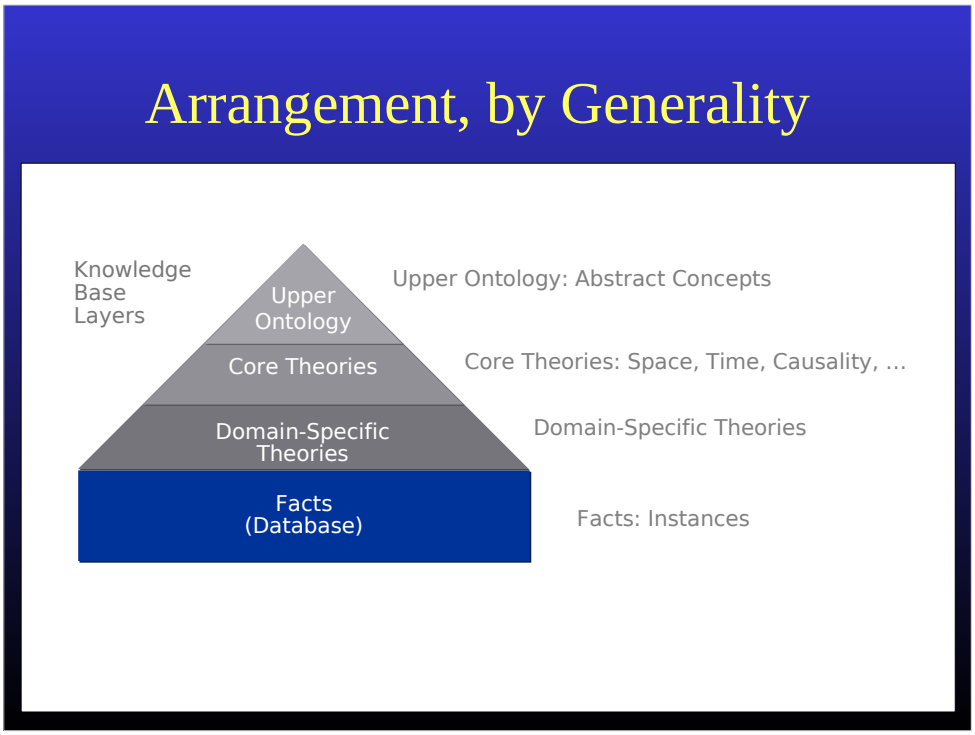
- ***Introduction***
- Fundamental Expression Types
- Top Level Collections
- Time and Dates
- Spatial Properties and Relations
- Event Types
- Information
- More Content Areas

This set of lessons (tutorial) is intended to give a broad overview of the content of the Knowledge Base. In an effort to expose you to as much of the knowledge base as possible, we included many examples to illustrate the topics we address. Some of the slides in this tutorial are so simple and self-explanatory that we will give no explanation, others will have many examples of a common type, only some of which will be explained, with the expectation that the reader can fill in the explanation for the rest.



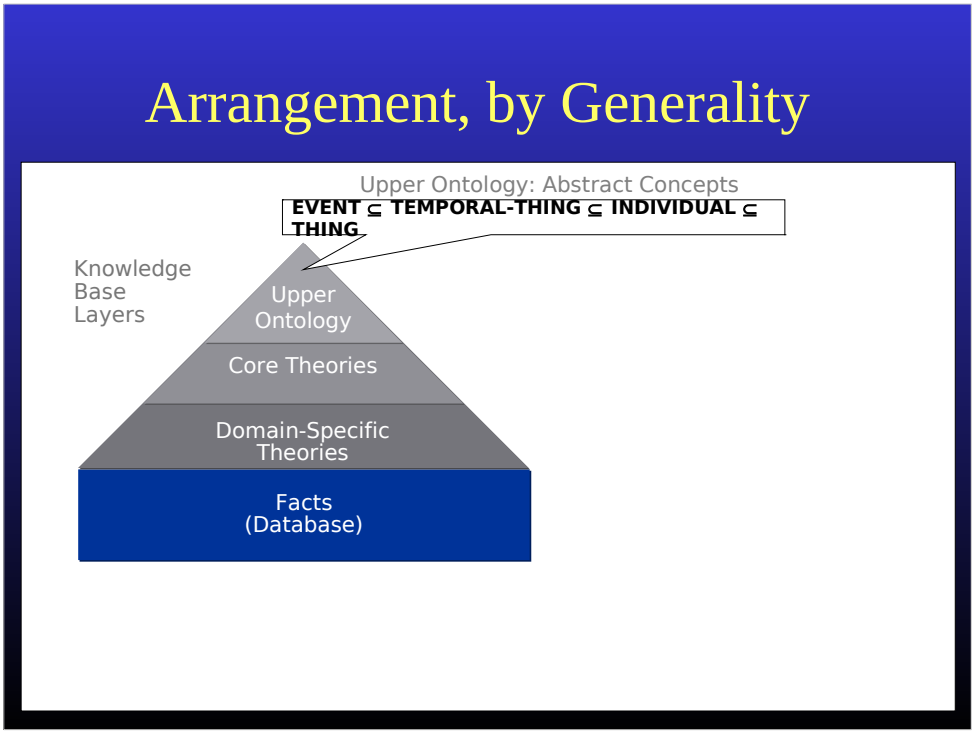
There are many methods for representing knowledge, including written documents, text files, databases, etc. The advantage that Cyc has over these methods is the language in which its knowledge is written, CycL. In CycL, the meanings of statements and inferential connections between statements are encoded in a way that is accessible to a machine. At the present time Natural Languages are virtually meaningless to machines. I can say "all animals have spinal cords. All dogs are animals. My pet is a dog." From these sentences, a *person* can infer that my pet has a spinal cord, but a *machine* cannot, at least not until a machine can understand English sentences.

In the formal language Cyc uses, inference is reduced to a matter of symbol manipulation, and thus something that a machine can do. When an argument is written in CycL, its meaning is encoded in the shape, or symbolic structure, of the assertion it contains. Determining whether or not an argument is valid can be achieved by checking for certain simple physical patterns in the CycL sentence representing its premises and conclusions.

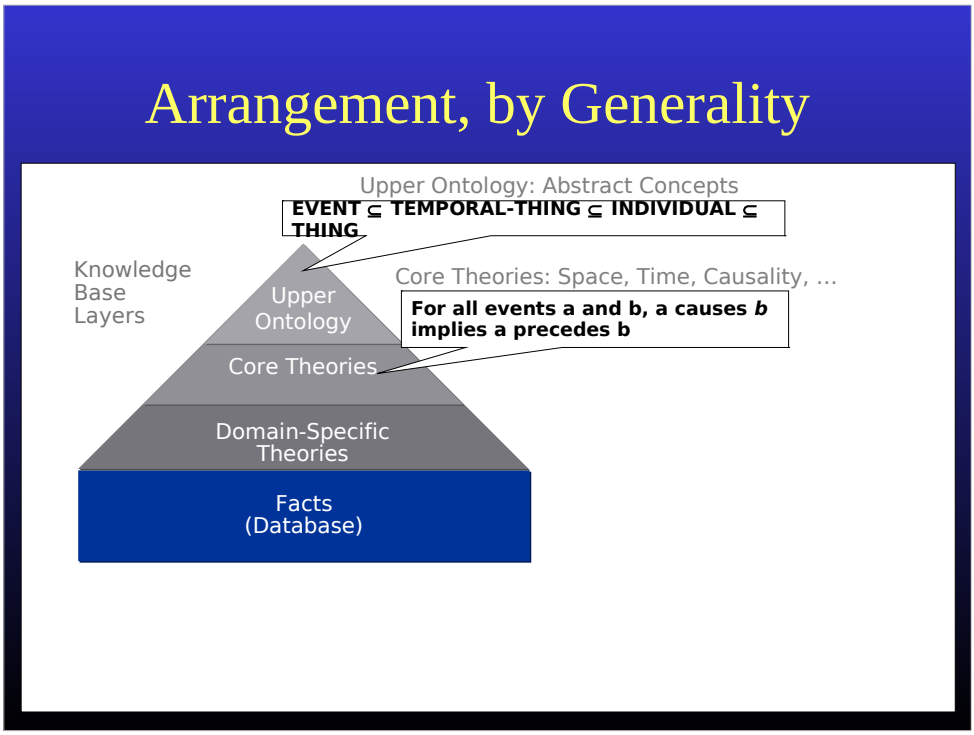


The Knowledge Base (KB) itself comprises a massive taxonomy of concepts and specifically-defined relationships that describe how those concepts are related.

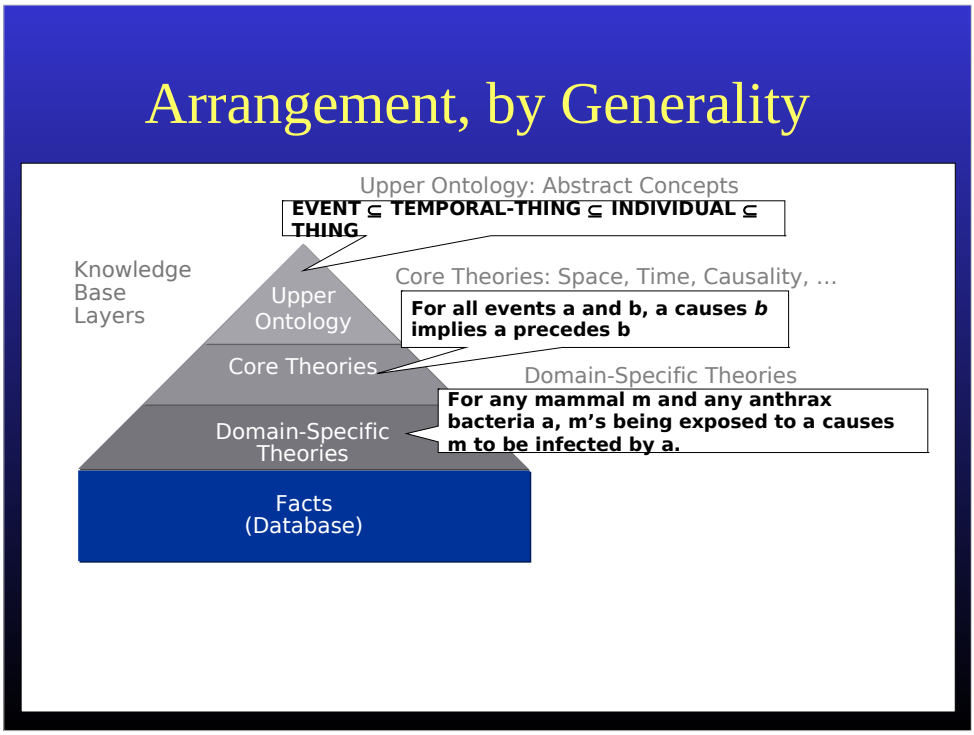
This figure represents the context of the knowledge arranged by degrees of generality, with a small layer of abstract generalizations at the top and a large layer of real-world facts at the bottom.



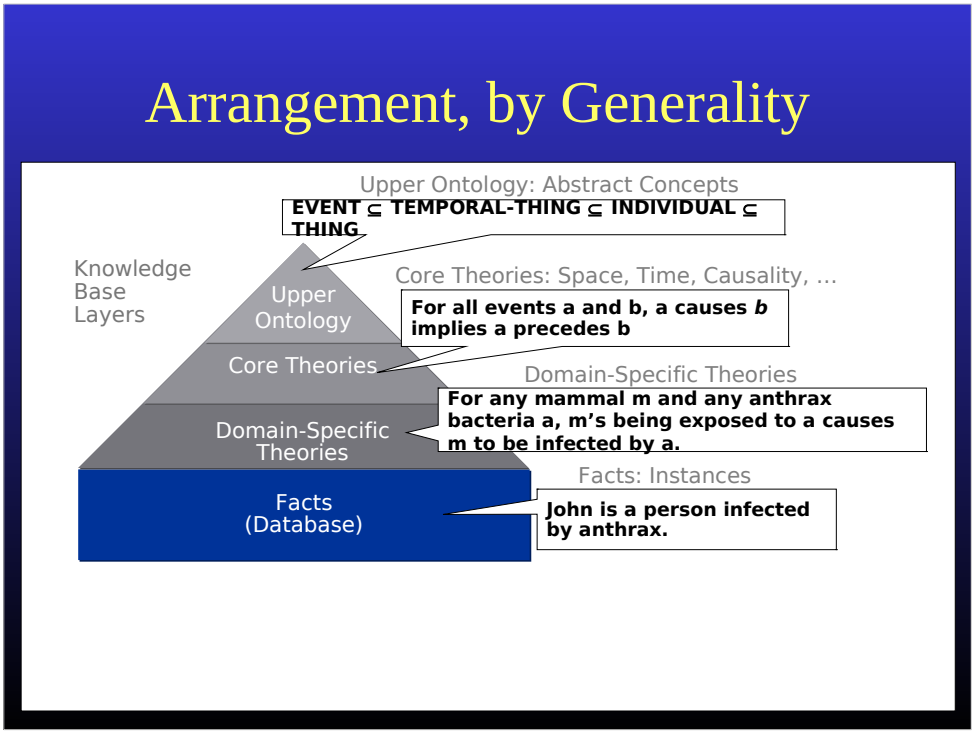
The Upper Ontology doesn't say much about the world at all. It represents very general relations between very general concepts. For example, it contains the assertions to the effect that every event is a temporal thing, every temporal thing is an individual, and every individual is a thing. "Thing" is Cyc's most general concept. Everything whatsoever is an instance of "thing."



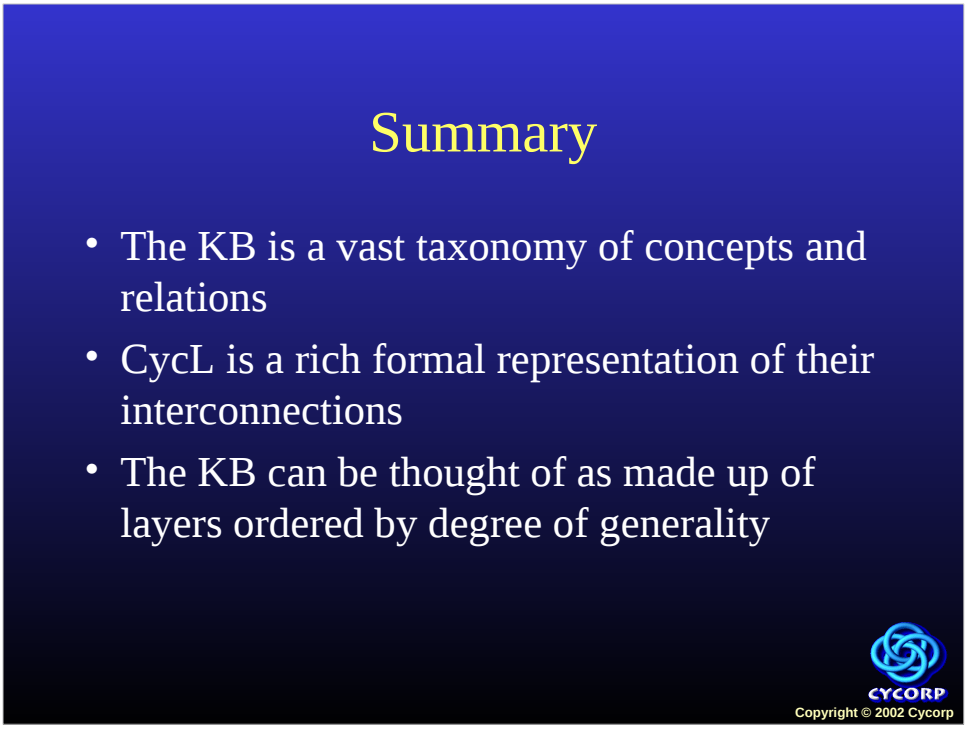
The KB contains several core theories that represent general facts about space, time, and causality. These are the theories that are essential to almost all common-sense reasoning.



Domain-Specific Theories are more specific than core theories. These theories apply to special areas of interest like military movement, the propagation of diseases, finance, chemistry, etc. These are the theories that make Cyc particularly useful, but are not necessary for common sense reasoning.



The final layer contains what is sometimes called "ground-level facts." These are statements about particular individuals in the world. For example, "John has anthrax" is a specific statement about one person. Generalizations would not go here, they would go in a layer above. Anything you can imagine as a headline in a newspaper would probably go here.



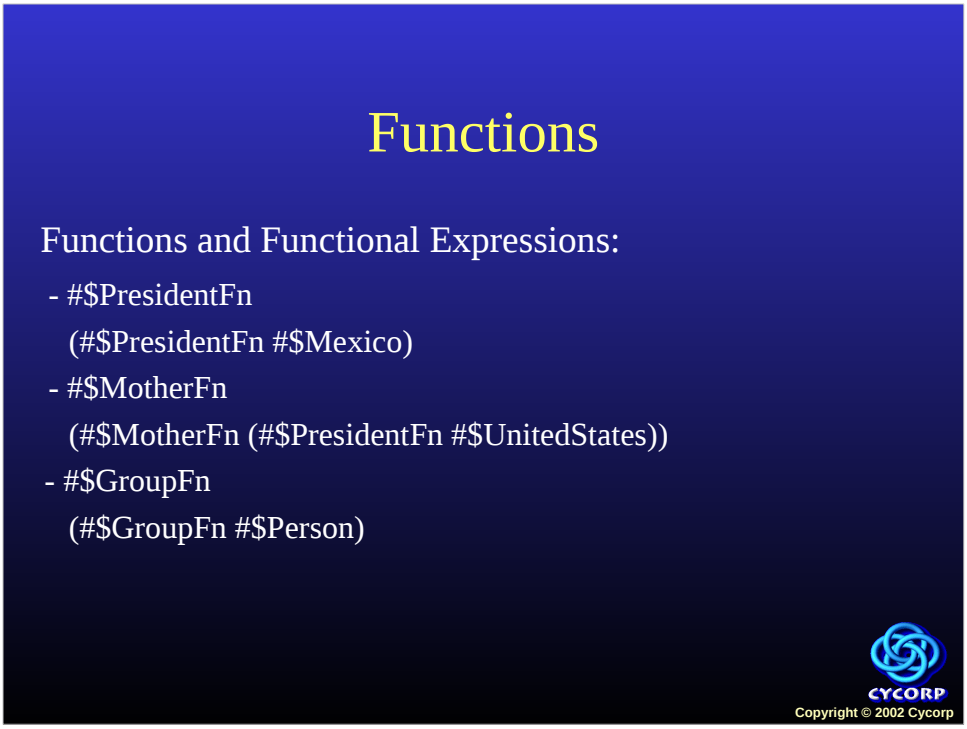
This concludes the introduction to the tutorial that surveys the contents of the Knowledge Base.

In this lesson we will focus on the fundamental expression types of CycL. Generally speaking, fundamental expression types are the building blocks of language. They are the things that you probably think of when you think of grammar. In English we have nouns, adjectives, verbs, etc. Here, we're going to focus on the fundamental expression types in CycL, such as constants, functions, terms, predicates, quantifiers, etc.

Constants can denote individuals, collections, or collections of collections. #$GeorgeWBush, #$Sudan, and #$0-TheDigit are all constants that denote a specific individual. #$Sudan denotes Sudan, the country in Africa . #$0-TheDigit denotes zero, which is an individual that is a specific abstract object.

#$WorldLeader and #$Country are constants that denote collections. #$WorldLeader, for example, denotes the collection of all world leaders. #$Country does not denote a specific country, but rather the collection of all countries.

Collections of collections can be more confusing. They denote all of the collections of all x. For example, the members of the collection #$AutomobileTypeByBrand are all of the kinds of vehicles characterized by being of a certain brand. The members are not the vehicles themselves, but they are kinds of vehicles. So collections of collections don't denote physical clusters or groups; instead they denote an abstract group whose members are distinguished by the value of a shared attribute.



Functions take arguments and return results. Consider the following examples.

#$PresidentFn takes a country as its argument and returns the name of a president as its result. So, (#$PresidentFn #$Mexico) takes #$Mexico as its argument and returns Vicente Fox as its result. Another way of saying this is that (#$PresidentFn #$Mexico) denotes Vicente Fox.

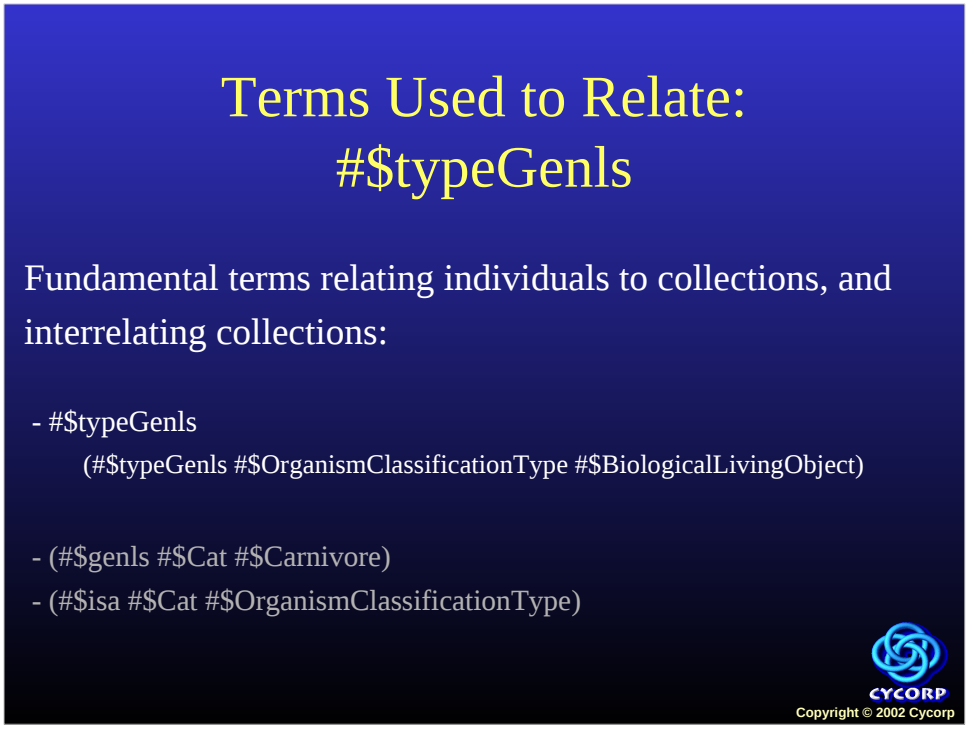
The function #$MotherFn does the obvious; it denotes the name of an animal's biological mother. We can build compound functional expressions by putting two functions together, as in (#$MotherFn (#$PresidentFn #$UnitedStates)). When this lesson was written, this expression denoted Barbara Bush, who was the mother of the then current president of the United States.

The function #$GroupFn denotes not an individual, but a collection. Thus, in this case, (#$GroupFn #$Person) denotes the very large collection of all groups (or collections) of people. Note that members of these groups can overlap, thus the groups include Americans, smokers, plumbers, athletes, members of the 1992 Boston Red Sox, etc.

#$isa is the most basic term in CycL. This term is used to say that something is part of a collection. Everything belongs to at least one collection.

(#$isa #$GeorgeWBush #$WorldLeader) says that George Bush is a world leader. It also says that George Bush is an individual in the collection of world leaders. (#$isa #$Cat #$OrganismClassificationType) denotes a case of a collection of collections. #$Cat does not refer to a specific cat, but to the collection of all cats. This expression says that #$Cat is a member of the collection #$OrganismClassificationType.

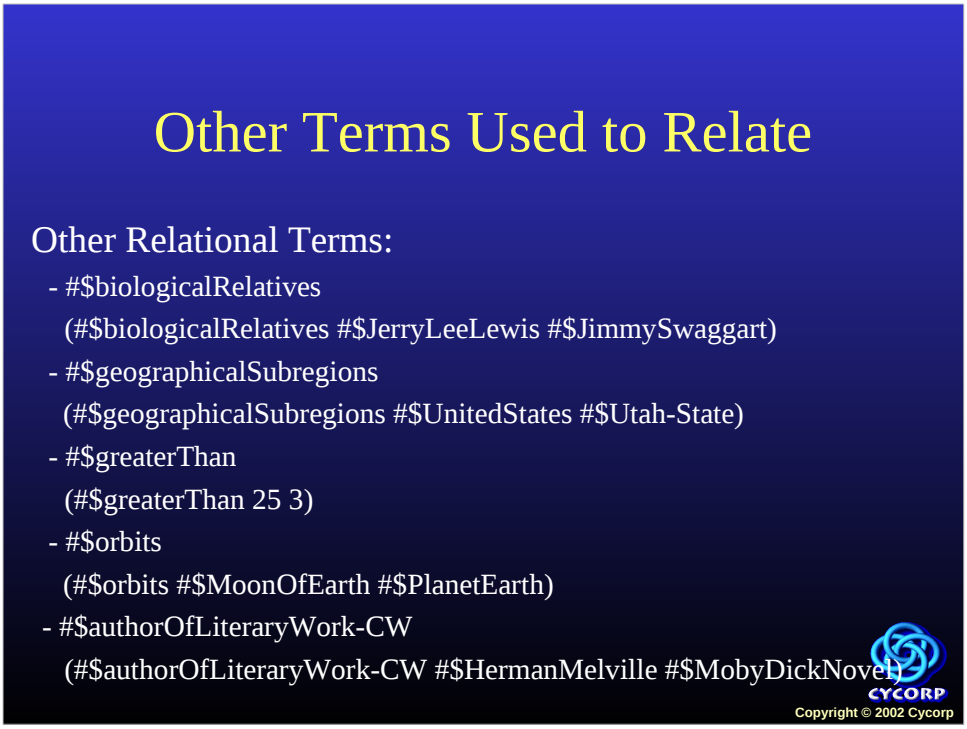
The #$genls term is used to say that one collection is a sub-collection of another. Thus, if instead of using #$isa above, we had said (#$genls #$Cat #$OrganismClassificationType) we would have been saying that every individual in the collection #$Cat is also in the collection #$OrganismClassificationType. This is false, of course, since Tigger The Cat is not a type of organism classification. The expression (#$genls #$Cat #$Carnivore) says that every cat is a carnivore. Thus, if Tigger is an individual in the collection of all Cats, Tigger is also in the collection of all Carnivores.



The relation #$typeGenls is difficult to understand. In order to gain a better understanding of this term, let me translate the first example on the slide. It says "every collection that is an Organism Classification Type is such that it has #$BiologicalLivingObject as a genls." In other words, #$OrganismClassificationType is a collection of collections whose members can be generalized from the members of #$BiologicalLivingObject. Because this concept is difficult to understand, let me state this one more time in different words. If we take all of the members of #$BiologicalLivingObject and generalize them into smaller collections such as #$Cat, #$Dog, etc., those collections could also be found as members in the collection of collections called #$OrganismClassificationType.

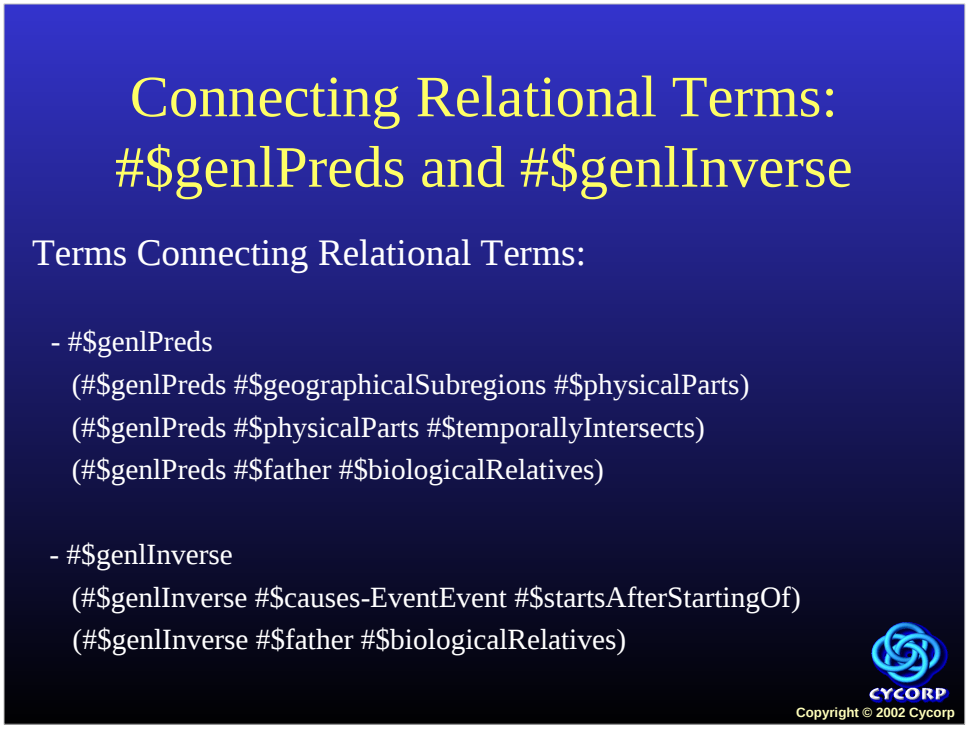
This relation is distinct from the relation in the previous example of #$Cat being a sub-collection of Carnivore. In that case, Carnivore is a collection of individuals (with names like Tigger and Rover), and we grouped some of those individuals into a sub-collection that we called #$Cat. In the current example, #$Tigger and #$Rover would be explicit members of #$BiologicalLivingObject, but not explicit members of #$OrganismClassificationType (whose members would have names like #$Cat and #$Dog). However, because we know that Cat is an Organism Classification Type and members of collections of Organism Classification Types can be generalized from the members of Biological Living Objects, we can infer that any member of the collection Cat is also in the collection of all Biological Living Objects.

The term #$disjointWith is rather simple. It means that nothing exists that is a member of both collections to which it's referring. Thus, the example on the slide says that nothing is both a fish and a mammal, or rather, no individual exists that is a member of the collection #$Fish and also a member of the collection #$Mammal.



There are many more terms used to denote relationships in CycL. The relationships that the terms on this slide denote are obvious in how they are named. Consider the first example. #$biologicalRelatives relates two terms, in this case Jerry Lee Lewis and Jimmy Swaggart. This example asserts that these two are biological relatives (as opposed to being related legally or in some other manner).

Notice that the relationships on the slide are between individuals, not collections of individuals. The vast majority of relational terms in CycL are used to relate one individual to another individual.



CycL has terms that relate one relational term to another. The two main terms for accomplishing this are #$genlPreds and #$genlInverse.

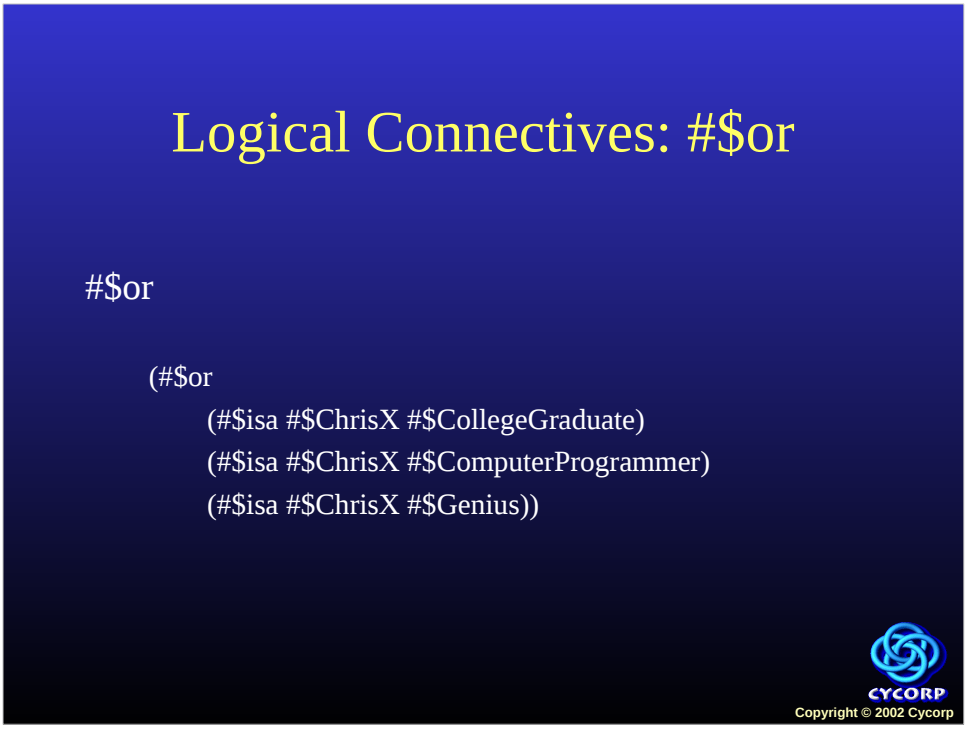
We have already discussed these two terms in previous lessons but let's review them again here. #$genlPreds is used to say that if something is true of the first term, then it will be true of the second term as well. Refer to the examles on the slide. The first example says that if a thing x is a geographical subregion of thing y, then thing x is also a physical part of thing y. The second example says that if something is a physical part of thing y, it also temporally intersects (or exists at the same time as) thing y. The last example says that if x is a father to y, then x is also a biological relative to y.

#$genlInverse is just like #$genlPreds, but you flip the terms. Consider the first example under #$genlInverse on the slide. This example says that if x is an event that causes y, then y starts after the start, or beginning, of x. The second example says that if x is a father to y, then y is a biological relative to x. Compare the meaning of this sentence to the meaning of the second example sentence for #$genlPreds. Notice that the x and y changed places in the second half of the sentence.

CycL has predicates that are used to describe syntactic and semantic conditions for writing well-formed sentences. These are #$arity and #$argxisa.

#$arity denotes the number of arguments that a predicate must have -- the syntactic constraint. Consider the example on the slide, (#$arity #$biologicalMother 2). It says that any assertion using the predicate #$biologicalMother must include exactly two arguments (presumably the name of the mother and the name of that which is mothered).

The #$argxIsa predicate imposes semantic constraints. It constrains the meanings of the terms that are legal arguments for that predicate. The first #$argxIsa example on the slide, (#$arg1Isa #$biologicalMother #$Animal), says that the first argument of a sentence that uses the predicate #$biologicalMother must be an animal. Thus, the first argument cannot be a door or a car, etc. The second example, (#$arg2Isa #$biologicalMother #$FemaleAnimal), says that the second argument in a sentence using #$biologicalMother must be a female animal. Thus the sentence (#$biologicalMother #$Jim #$John) would be well-formed as far as the first argument is concerned, since Jim is an animal, but not as far as the second argument is concerned, since John is presumably not a female animal.



Let's now review some CycL logical connectives that we've seen in previous lessons.

The logical connective #$or relates two or more assertions in such a way that an expression which uses this connective is true if any (one or more) of the assertions are true. Refer to the example on the slide. Lets' say that we want to hire Chris for an entry-level programming position. If Chris fills any of the following qualifications, we'll extend him an offer:

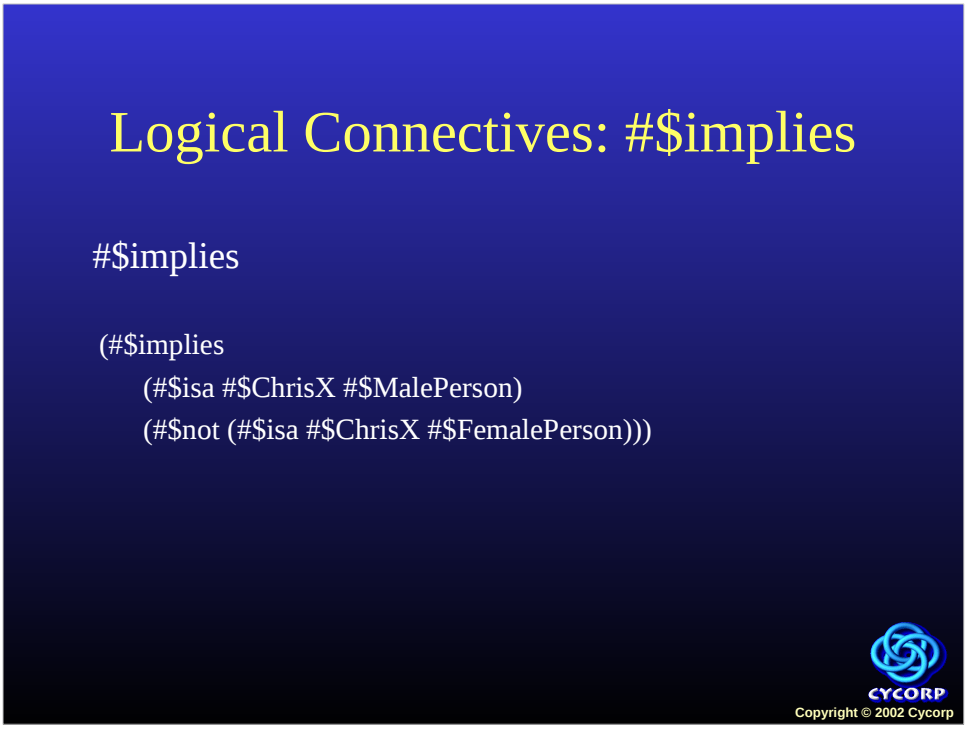
- is a college graduate
- is a computer programmer
- is a genius

Logical Connectives: #$and and #$not

#$and, #$not

```
(#$not
  (#$and
    (#$isa #$ChrisX #$MalePerson)
    (#$isa #$ChrisX #$FemalePerson)))
```

The #$not and the #$and connectives in the example on this slide are used to say that it is not the case that the following assertions are both true. Thus it is not true that ChrisX is both a male person and a female person.



The connective #$implies is used to say that if the first assertion is true, then the following assertion is true as well. Notice that the second assertion in the example on the slide is preceded by the #$not connective, meaning that the opposite of what the assertion claims is true. Therefore, this example says that if Chris is a male person, that implies that Chris is not a female person.

Now let's review some CycL quantifiers.

#$forAll is the universal quantifier. When used with variables, this term allows you to say things like the example on the slide: For any ?COUNTRY and any ?PERSON, if that ?COUNTRY is a #$Superpower, then the ?PERSON who is the head of its government has the status of #$WorldLeader.

Quantifiers: #$thereExists

#$thereExists

```
(#$forAll ?ANIMAL
   (#$implies
      (#$isa ?ANIMAL #$Vertebrate)
      (#thereExists ?PART
         (#$and
            (#$isa ?PART #$Tongue)
            (#$anatomicalParts ?ANIMAL ?PART)))))
```

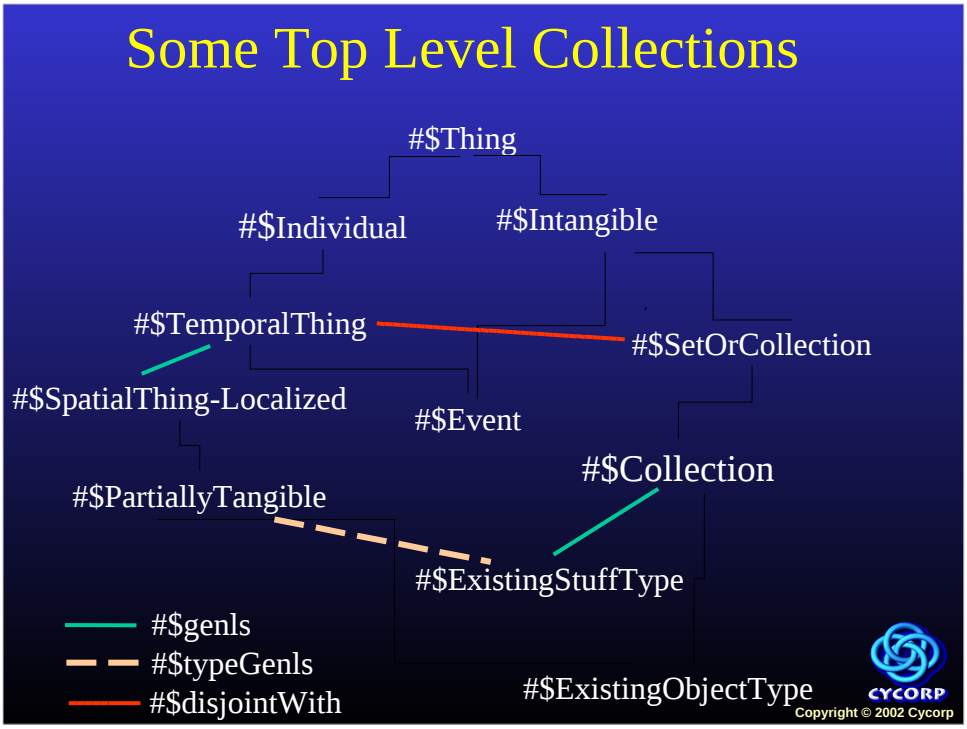
#$thereExists is the existential quantifier. When used with variables, this term allows you to say that something exists. The example on this slide says that every vertebrate has a tongue as an anatomical part. To be more literal in the translation, it says that for every animal, it follows from that animal being a vertebrate that it has a part that is a tongue and that part is one of its anatomical parts.

Summary

- Constants denote individuals, collections, and collections of collections
- Functions and Functional Expressions denote terms
- Terms used to relate: #$isa, #$genls, #$typeGenls, #$disjointWith
- Connecting Relational Terms: #$genlPreds and #$genlInverse
- Predicates For Describing Well-Formedness: #$arity and #$argxIsa
- Logical Connectives and Quantifiers

This concludes the lesson on Fundamental Expression Types in CycL.

The purpose of this lesson is to expose you to some of the top level collections in Cyc and give you an idea of how they relate to each other.



This slide shows the organization of some of the most fundamental collections in Cyc. For example, the collection #$Intangible is related to #$Thing with a solid green line, meaning that #$Intangible generalizes to (or is a sub-collection of) #$Thing. #$Thing is the supreme collection. Everything in the universe (even intangible things) is an instance of #$Thing. Consequently, every collection is a specialization of of #$Thing.

#$Intangible is the collection of all things that cannot be touched. #$TemporalThing is the collection of things that exist in time. An integer would not be in this collection because integers do not exist at a specific time -- they are abstract, or timeless. Instances of #$SpatialThing-Localized are simply those things that have a location in space.

#$ExistingStuffType and #$ExistingObjectType correspond to the common-sense notions of “stuff” and “object” respectively. Water is stuff, but a lake is an object. The collection #$Water is stuff-like in that each portion of an instance of #$Water is also an instance of #$Water. #$Lake is object-like in that typically any proper part of a lake is not itself a lake.

Let's look at a specific lower-level collection in detail. The collection #$Dog is asserted in Cyc to be an instance of the following collections: #$OrganismClassificationType, #$BiologicalTaxon, #$BiologicalSpecies, and #$DomesticatedAnimalType.

#$Dog is asserted to be a specialization of the collection #$CanineAnimal for the obvious reason that every instance of #$Dog is also an instance of #$CanineAnimal.

"Genls" is a transitive relation, so any genls of a genls of #$Dog is itself a genls of #$Dog. For example, #$CanineAnimal is a genls of #$Dog and #$Carnivore is a genls of #$CanineAnimal. Consequently, #$Carnivore is a genls of #$Dog. By the same reasoning, every genls of #$Carnivore is also a genls of #$Dog. This slide shows all collections that are genls of #$Dog.

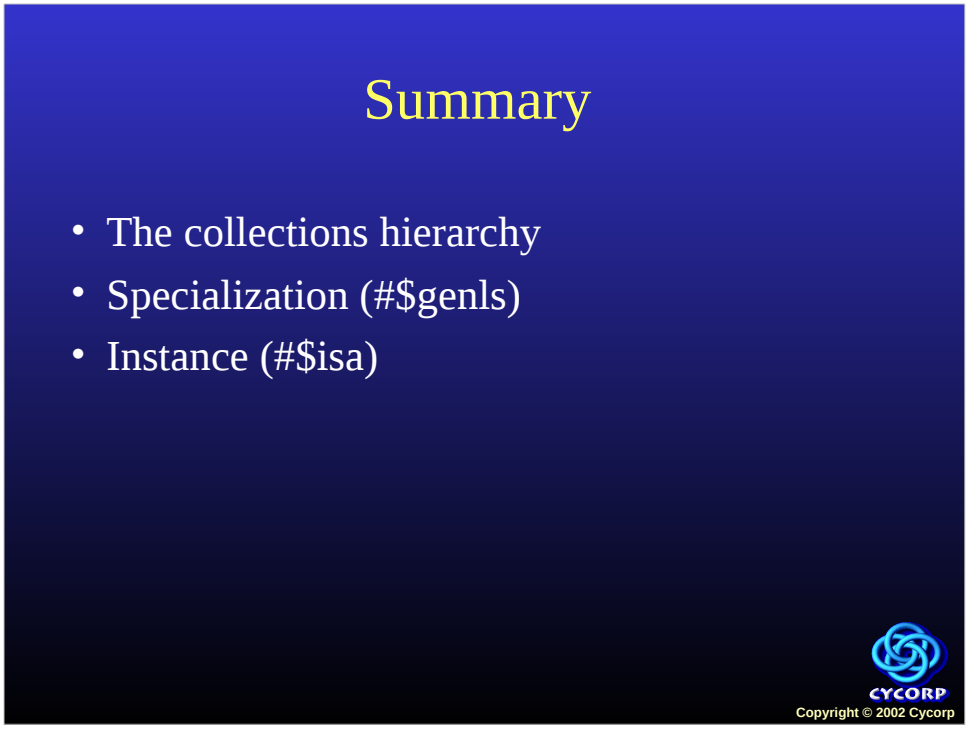
To take a few examples, Cyc knows that the members of #$Dog are also: agents, air-breathing vertebrates, heterotrophs (they require organic nutrients in order to survive), hexalateral objects (meaning that it is appropriate to refer to a front side, a back side, a top, a bottom, a left side, and a right side -- unlike a tennis ball which has no sides), perceptual agents (they can perceive things), and spatially localized things (they have a location in physical space), etc.

In contrast to the previous list of 45 collections of which #$Dog is a spec, this is a list of the collections of which #$Dog is an instance. Earlier we listed the four collections of which #$Dog is asserted to be a member. From this, Cyc knows that #$Dog is actually an instance of eleven collections. This is because each of the first four collections collectively are specializations of seven other collections.

#$Dog is an instance of #$ExistingObjectType, meaning that any part of a dog is not a dog on its own (a dog's leg is not a dog).

#$Dog is an instance of #$TemporalStuffType, meaning that anything that is a dog at one time will always be a dog (this is not true of a teacher, for example, who can be a teacher one year and an attorney the next year).

#$Dog is an instance of #$OrganismClassificationType which means, roughly, that this collection is used in the scientific classification of organisms.



This concludes the lesson on top level collections in Cyc.

Fundamental to all discussions of causality and reasoning is knowing what happened before what. Therefore, time is crucial to all reasoning. This lesson will cover the basics of representing time and dates in Cyc.

Functions Which Return Time Intervals

#$YearFn, #$DayFn, #$MinuteFn, #$SecondFn

Example:

(#$YearFn 2000)

What the functions on this slide denote seems obvious. For example, (#$YearFn 2000) denotes the year 2000.

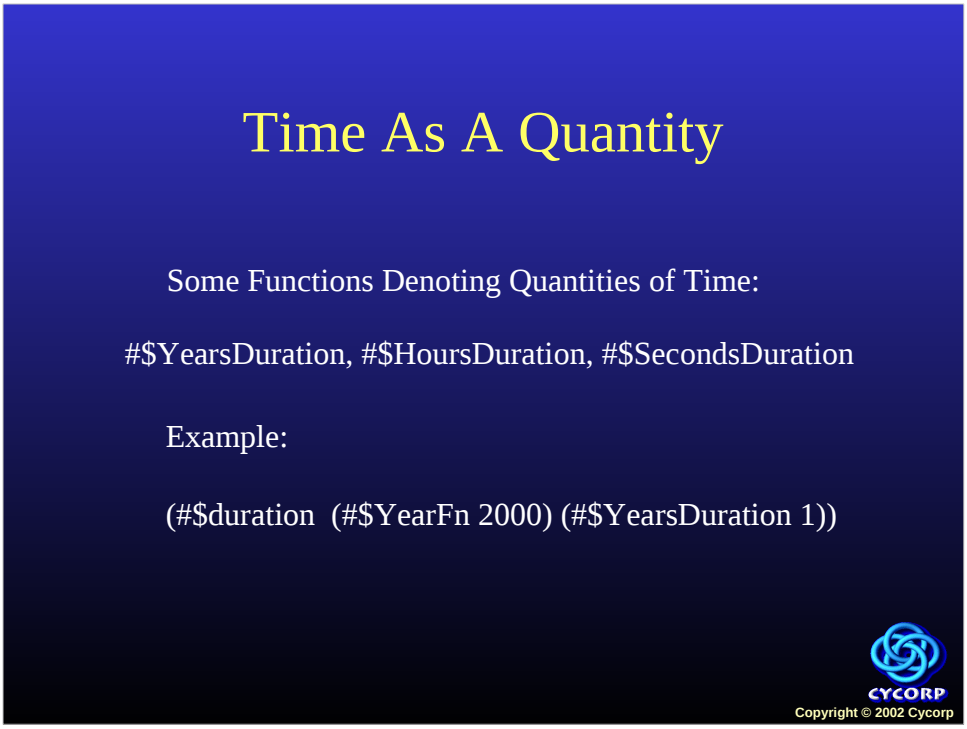
Functions Which Return Time Intervals: Composite Expressions

#$YearFn, #$DayFn, #$MinuteFn, #$SecondFn

Example:

```
(#$SecondFn 59
  (#$HourFn 23
    (#$DayFn 31
      (#$MonthFn #$December
        (#$YearFn 2000)))))
```

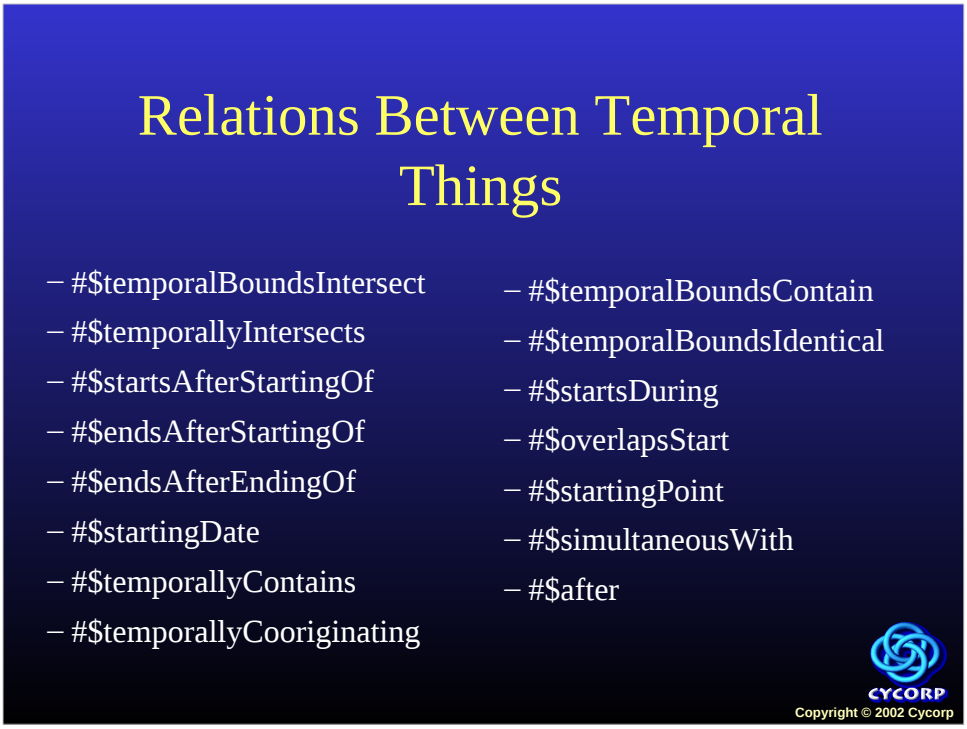
You can combine the functions together to form a composite functional expression like the example on the slide. This expression denotes the last second of the year 2000. A more literal translation would be “the fifty-ninth second of the twenty-third hour of the thirty-first day of the month of December of the year 2000.”



There are also functions that denote intervals of time, like years, which have a place on the timeline.

The example on the slide says that the duration of the year 2000 is a one year duration. If instead the example said (#$duration (#$YearFn 2000) (#$YearsDuration 3)), it would correctly assert the false idea that the duration of the year 2000 is three years.

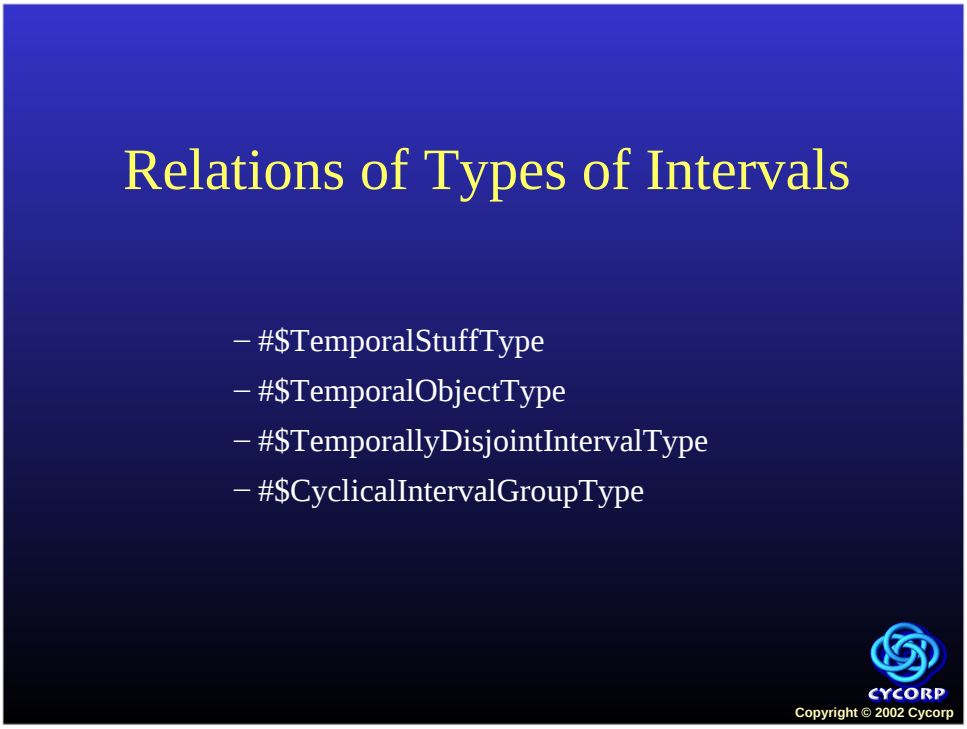
These quantities are simply abstract objects that measure the length of an interval of time.



Most reasoning is more concerned with duration, or chunks of time, than specific points in time. Therefore, Cyc has many predicates for describing relationships between chunks of time.

The predicates listed on the slide are only a sample of what is available in Cyc. Most mean just what they say, so #$startsAfterStartingOf could be applied to this hour and last hour to say that this hour starts after the starting of last hour.

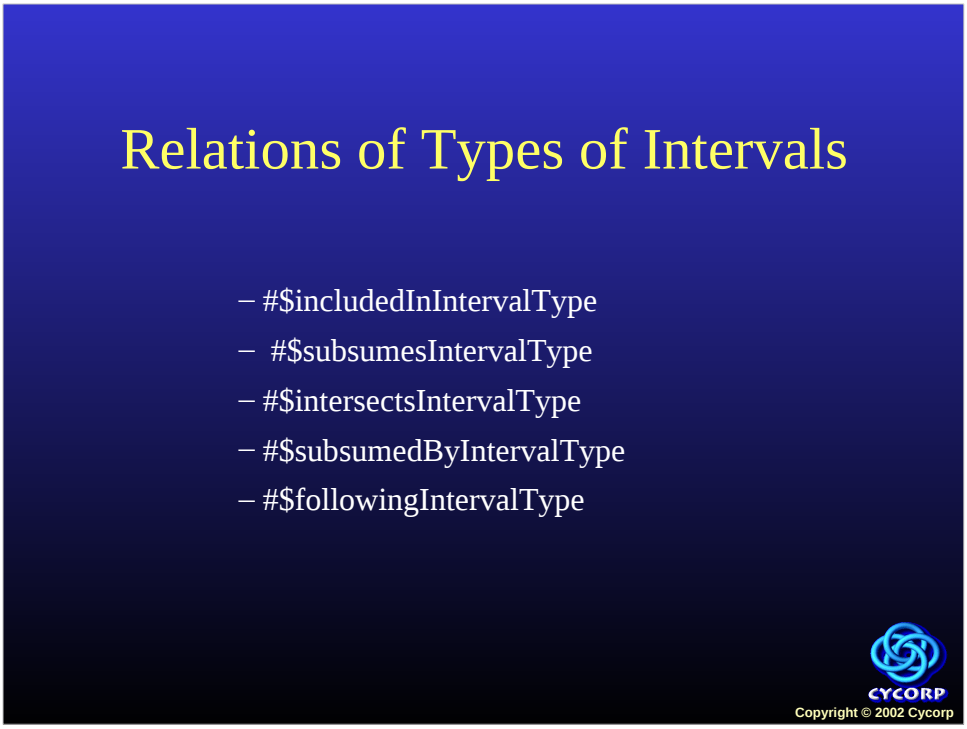
The last four predicates in the column on the right are the least commonly used because they are restricted to relating points in time.



Now we will consider a couple of the relationships that are so important for reasoning.

The first collection listed on the slide, #$TemporalStuffType, is a collection of events such that any proper time slice of any one of its members (events) is itself a member (event) in that collection. In this sense, the #$TemporalStuffType collection is similar to the #$ExistingStuffType collection; however, #$TemporalStuffType applies to proper time slices of events as opposed to portions of objects. For an example of #$TemporalStuffType, think of walking. Disregarding issues of granularity, a time slice of walking would represent what happens at any point in the walking event.

#$TemporalObjectType is a collection of events such that any proper time slice of any one of those events is not itself in that collection. This corresponds to #$ExistingObjectType. For example, consider a marathon run. Any proper time slice of the run would be a shorter run, and therefore would not be a marathon run. Another example would be the event of making a cake. Any proper time slice of the process would be beating eggs or stirring or preheating the oven, and would not represent everything that happens in making a cake.



There are also predicates which allow us to make assertions about the inter-relatedness of time intervals. The predicates listed on the slide allow us to make assertions like the following:

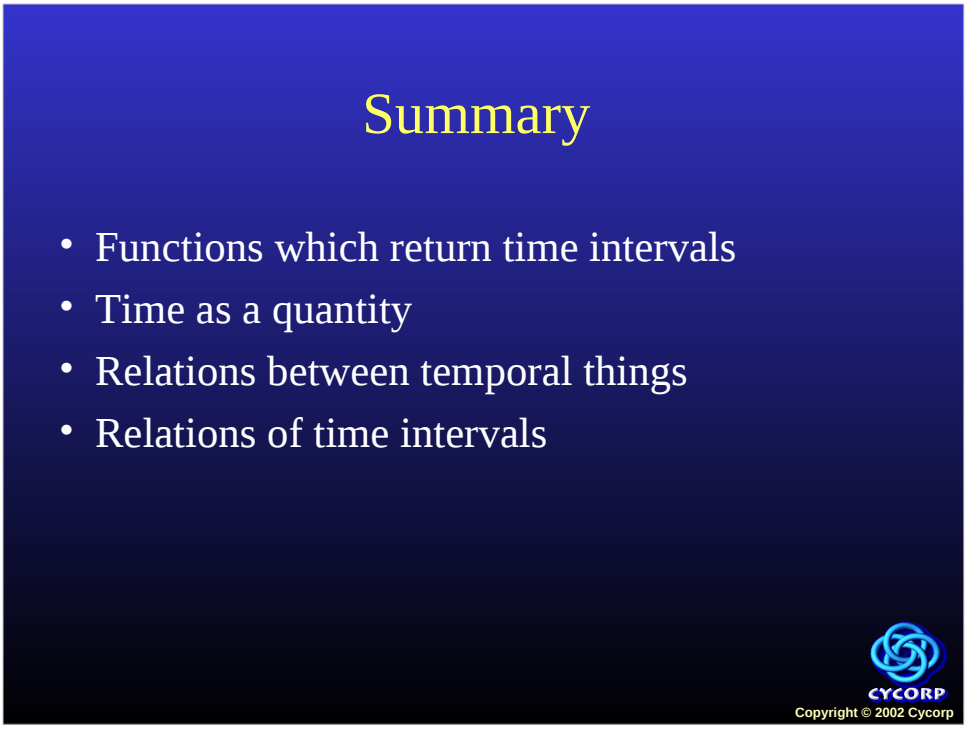
Every February 29th is subsumed by a February: (#$includedInIntervalType (#$DayFn 29 #$February) #$February).

Every February subsumes a Wednesday.

Every February intersects some winter season (in a theory applying to the Northern Hemisphere).

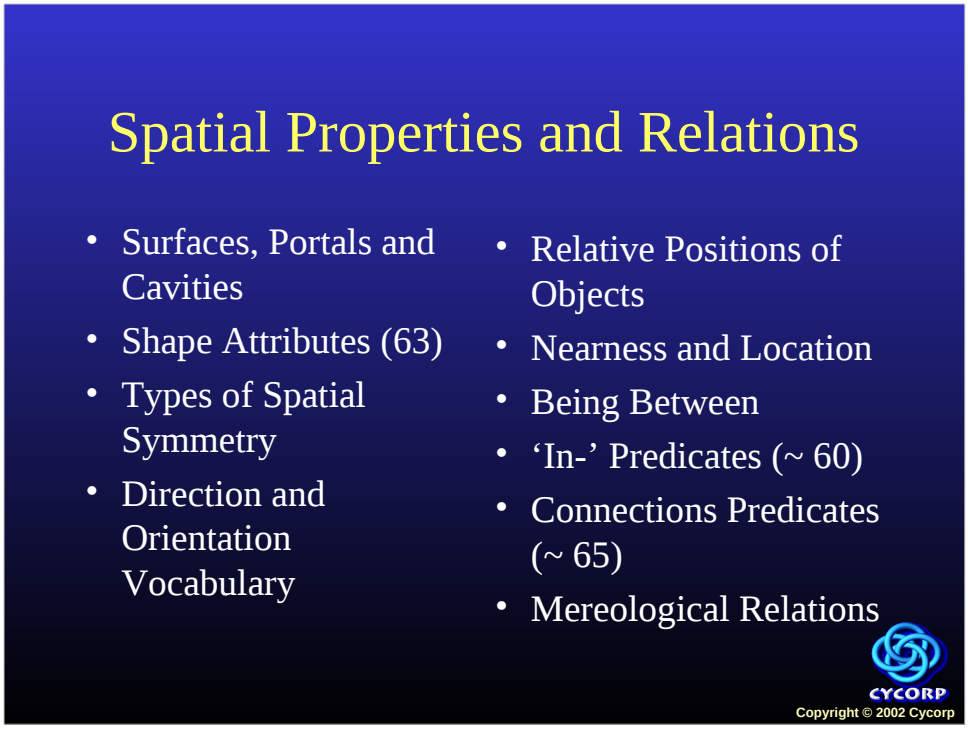
The day Jim was born occurred in a February.

Every Tuesday is followed by a Wednesday.



This concludes the lesson on representing time and dates in Cyc.

Just as Cyc has many ways of representing aspects of time, Cyc has many ways of describing spatial properties and relationships. Fortunately, the ideas that we present in this lesson are familiar and much of the vocabulary is self-documenting, so we won't add much explanation.



Cyc has various ways of expressing relative location, such as those listed on this slide.

Notice how many different shape attributes Cyc has -- 63! These include attributes like #$Arc, #$Linear-Planar, #$RoundShape, and #$Amorphous.

Types of Spatial Symmetry include things like bilateral and radial.

You can specify direction and orientation with ideas like "in front of" and "above," but Cyc requires specifics like #inFrontOf-Directly and #$inFrontOf-Generally.

Cyc differentiates between various senses of "between." For example, you can specify the distance between two objects on a line and you can specify the distance between two objects on a path (a path which might bend or even be circular like a track).

Senses of ‘In’

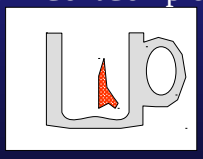
- Can the inner object leave by passing between members of the outer group?
  - Yes -- Try #$in-Among

CYCORP

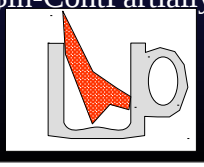
What do people mean when they use the word ‘in’? Use the questions on this and the two following slides as a guide when making decisions as to what sense of ‘in’ to use in a given situation.

Senses of ‘In’

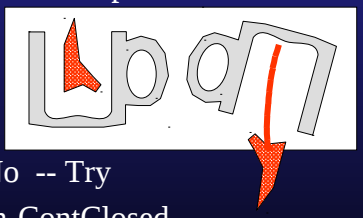
- Does part of the inner object stick out of the container?
  - None of it. -- Try #$in-ContCompletely



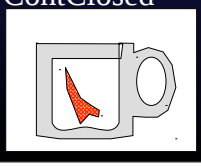
  - Yes -- Try #$in-ContPartially



- If the container were turned around could the contained object fall out?
  - Yes -- Try #$in-ContOpen



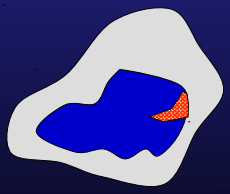
  - No -- Try #$in-ContClosed



Senses of ‘In’

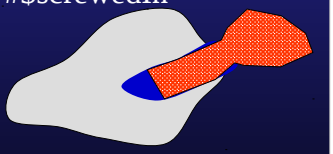
Is it attached to the inside of the outer object?

- Yes -- Try #$connectedToInside



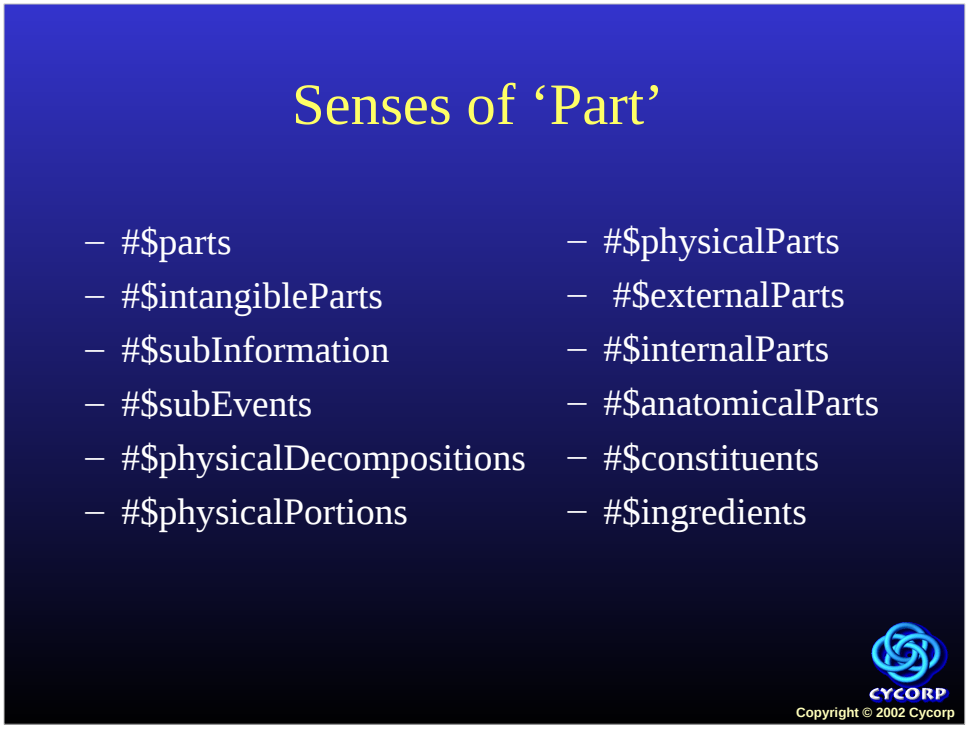
Can it be removed, if enough force is used, without damaging either object?

- Yes -- Try #$in-Snugly or #$screwedIn



Does the inner object stick into the outer object?

- Yes -- Try #$sticksInto



The list of predicates on this slide represents represents different aspects of "being a part of somethig." The predicate #$parts is the most general predicate in this list.

#$subInformation is totally abstract. It means that one piece of information is part of another. For example, "Jim and Mary went to the store" has "Mary went to the store" as a part of its information.

#$physicalDecompositions allows you to refer to an arbitrary physical chunk of an object. This is distinct from #$physicalPortions which refers to a portion that contains a representative sample of everything in the whole. So a physical portion of a salad with five ingredients might have representatives of all five ingredients, whereas a physical decomposition might have only one, two, or three of the ingredients.

#$physicalParts is what most people think of when they think of an object's parts. This is used to refer to the physically-separable parts of an object, even if the parts are glued or welded together, such that each part has its own identity (i.e. the wheels of a car, the tail of a dog, the fingers on a hand, etc.).

#$anatomicalParts refers to parts like physical parts, but anatomical parts each have their own anatomical function (i.e. the throat of a dog, the nervous system of a person, the eye of a bird).

Similar to physical objects, organizations also have parts. Cyc has a large number of terms to describe organizations. This slide shows a sampling of those terms, most of which are self-explanatory.

Notice that both predicates and functions are listed here. #$PolicyFn denotes the collection of all policies of an organization.

Summary

- Senses of ‘In’
  - #$in-Among
  - #$in-ContCompletely, #$in-ContPartially, #$in-ContClosed, #$in-ContOpen
  - #$connectedToInside, #$in-Snugly, #$screwedIn, #$sticksInto
- Senses of ‘Part’
- Organizations

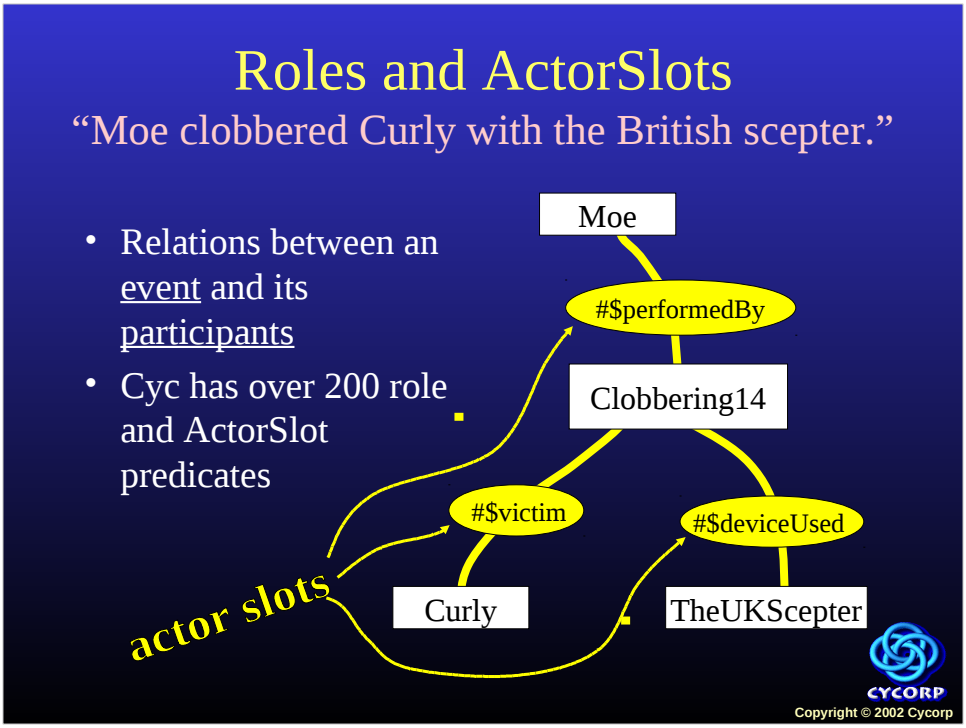
This concludes the lesson on spatial properties and relations.

Cyc uses around 37,000 different event types to describe what happens in the world. This large number includes events that range from the extremely general to the extremely specific. The extremely specific events might only be used once because they were created for a very specific situation and therefore involve a large number of qualifiers. For example, the sub-collection of all events in which garbage containing an explosive part has been disposed of by encasing in….

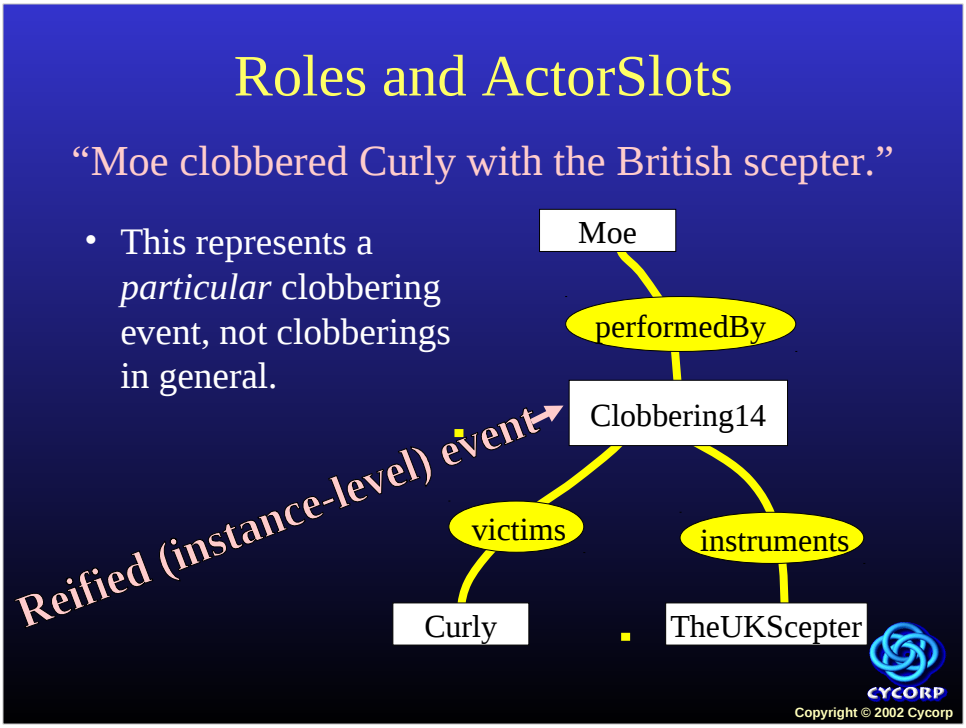
The slide lists common, everyday event types that are self-explanatory. To use them, you would represent a particular instance of an event and then represent the relationships of the participants in that event via predicates called Actor Slots. We'll review some examples of this after a few more slides.

In Cyc, "roles" and "actor slots" are used to describe the kinds of things listed on this slide. "Objects destroyed" includes objects that you can't get back out once you put them in. For example, eggs are destroyed in the #$BakingACake event. Think of facilitating objects as helpers, or assistants. For example, an electric mixer is a facilitating object in the #$BakingACake event. Slots (predicates) of motion and location are used when describing moving events, as in moving an egg from a carton to the bowl. The beneficiary is the recipient in an event.

Refer to the slide for an example of a particular instance of an action. The following two slides will diagram how we use roles and actor slots in representing this action.



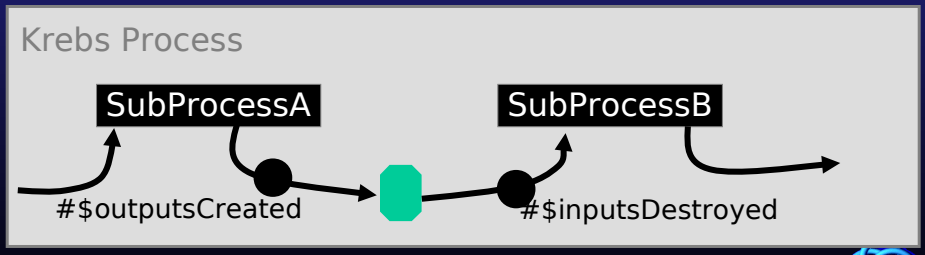
First of all, Cyc represents events by relating the event to its participants (not by relating participants to each other). Therefore, this diagram shows Clobbering14 in the center with all other aspects, or actor slots, related directly to the event, not to other aspects of the event. #$performedBy is an actor slot that is used to relate Moe to Clobbering 14. This is like saying "Clobbering, Moe did it" and "Clobbering, Curly received it"as opposed to "Moe clobbered Curly." Because Cyc describes events in this manner, we must have very specific ways of representing roles and actor slots. Cyc has over 200 role and actor slot predicates.



Clobbering14 would be an instance of the event that would be called #$Clobbering. It is crucial that events be described in terms of a particular instance and not the event in general because Cyc links participants (performers, victims, etc.) to each particular event. Therefore, events must be reified (represented in the knowledge base with a particular name).

Roles in events and subevents

- A product of one subevent of the Krebs Process is the input to another.
- Hence, different ActorSlot predicates.



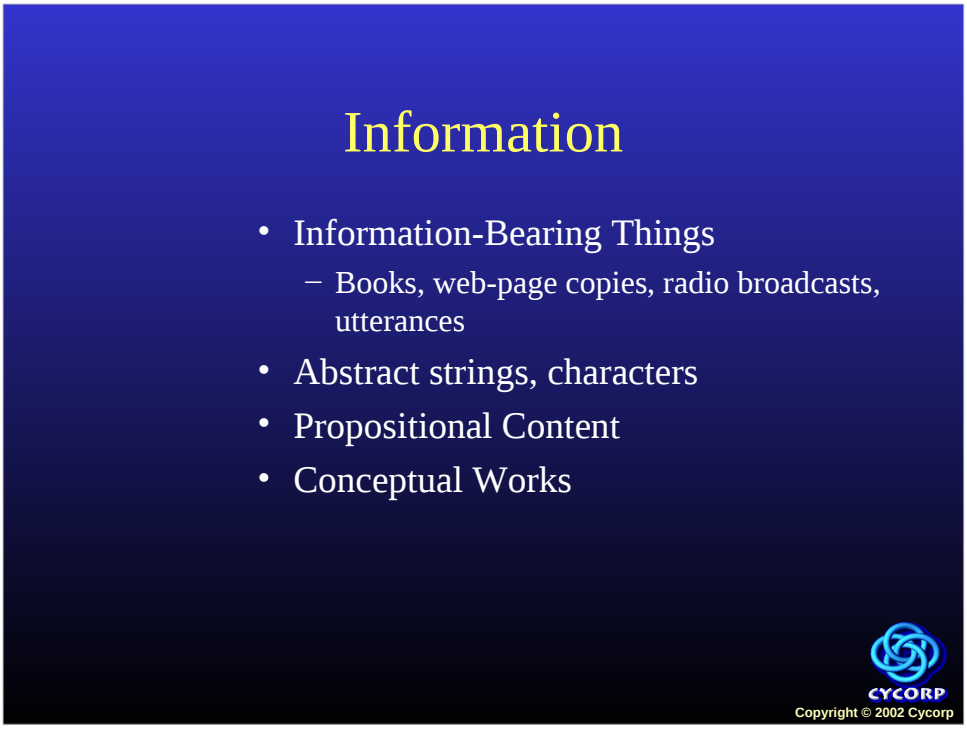
Because we describe events in Cyc in terms of a particular instance and not the event in general, roles and subevents allow us to represent more about an event. Consider the Krebs Process, a kind of biochemical process in which any instance of the process will be represented by multiple subevents related to a parent event. The first subevent creates an output that becomes an input to the second subevent. The second subevent destroys that input. So the same object (the black ball in the diagram on the slide) uses a different ActorSlot predicate (output/input) for each of the different subevents it relates to.

Summary

- Cyc has a large variety of predicates for representing roles preformed in events and the actors who perform them.
- Events are represented by relating actors to the event.
- The product of one sub-event in the input to another.

This concludes the lesson on event types in Cyc.

Information is one of the trickiest things to ontologize (represent in the Knowledge Base) because everything contains some kind of information. Movies, books, plays, and TV shows are all obvious examples of things that contain information, but the rings in a tree stump and the color of the sky contain information as well.



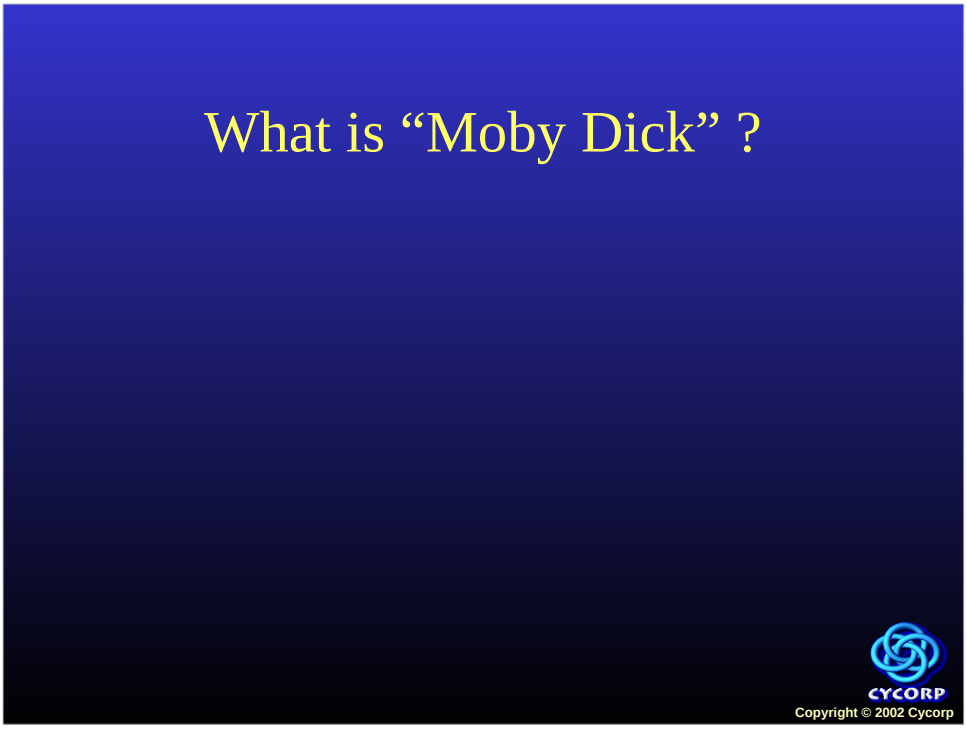
There are three obvious categories of things that contain information: information-bearing things, abstract strings and characters, and propositional content. However, in designing Cyc we discovered a fourth area which we call conceptual works.

Information-bearing things are the physical embodiments of information (i.e. an individual newspaper, which you can use to read or to wrap fish).

Information is encoded in abstract strings and characters. They are the abstract symbols and structures like words, sounds, and handwriting that are used to convey information. These are the most common elements used to intentionally convey information.

Propositional Content is what is encoded. It is the information that the abstract strings and characters combine to represent.

The following slides will further explicate the above ideas and introduce the idea of a conceptual work.



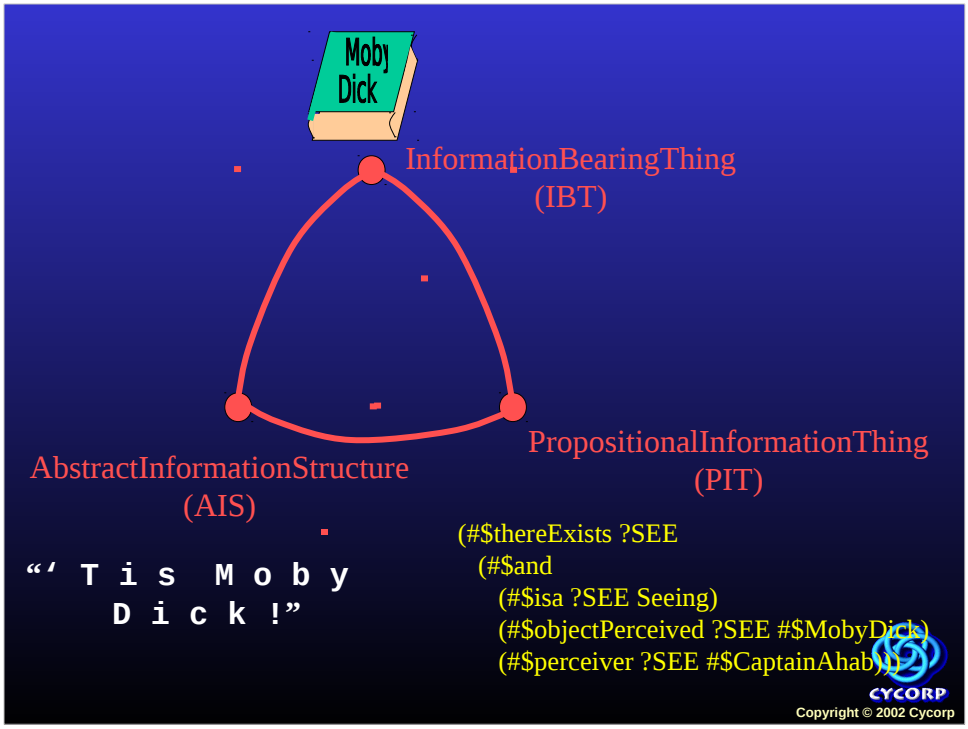
This slide presents an example that we will tease apart through the remainder of this lesson.

Consider “Moby Dick” and the three main categories of information detailed on the previous slide. As we continue with this lesson, you will see that “Moby Dick” doesn’t fit entirely into any one of those categories.

Question: What is “Moby Dick”?

Answer: A conceptual work.

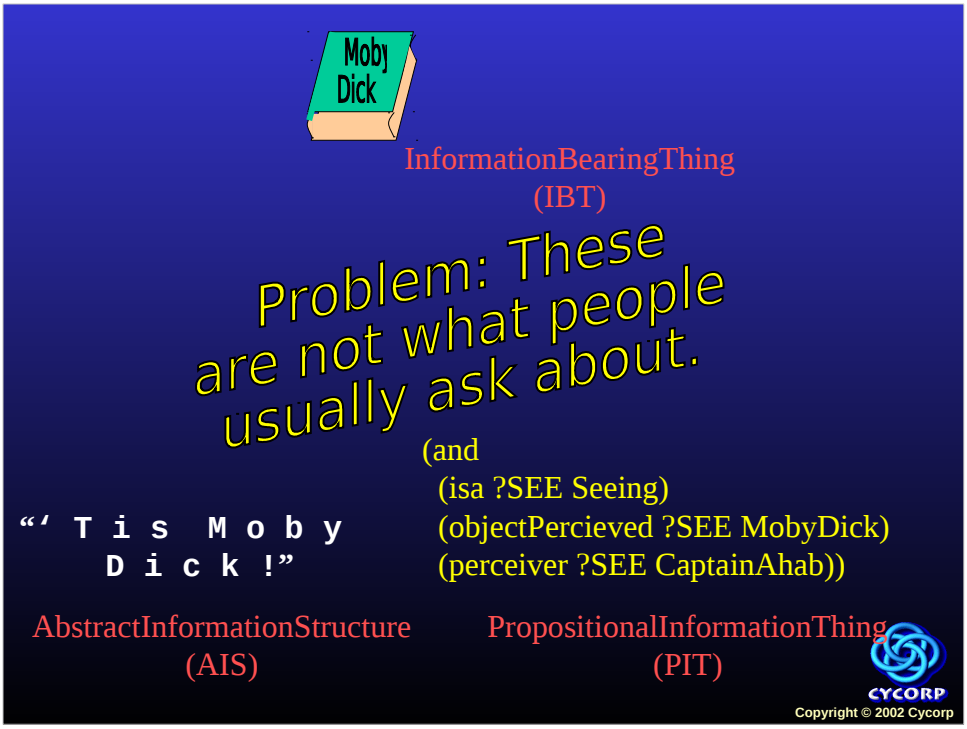
Let’s see why....



If I were to say that I like “Moby Dick,” I might mean that I like the way my special edition leather-bound copy of “Moby Dick” looks on the shelf. In this case I would be referring to the aspect of “Moby Dick” that is an information bearing thing (IBT).

If I were to say that I like “Moby Dick,” (albeit unlikely) I might mean that I like the specific sequence of the letters in the work called “Moby Dick.” In this case, I would be referring to the abstract information structure of “Moby Dick” (AIS).

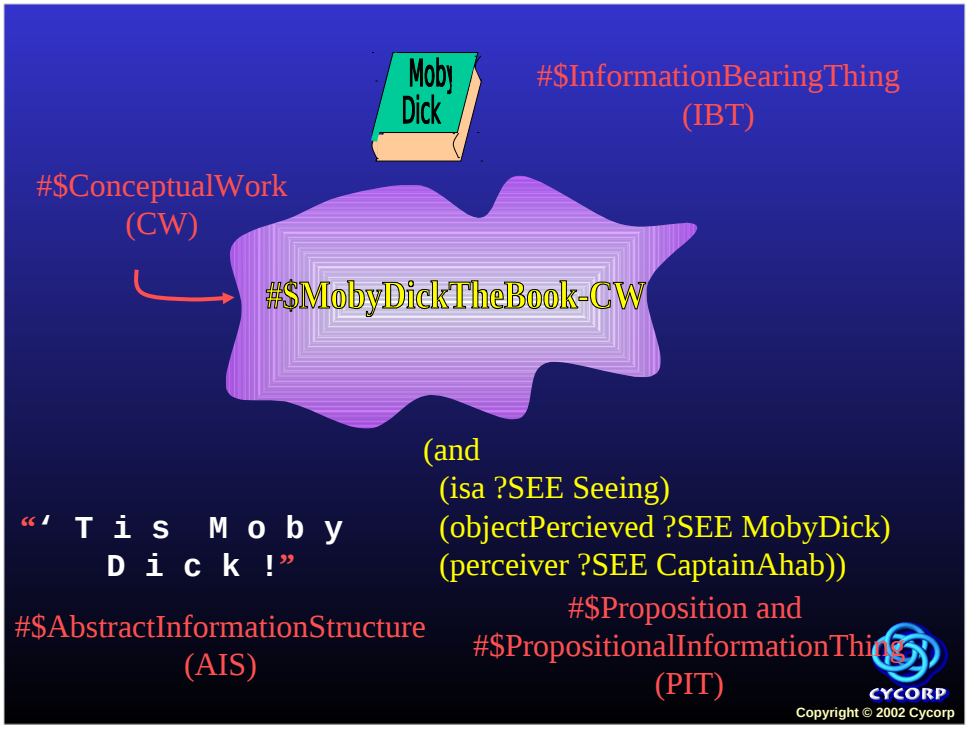
If I were to say that I like “Moby Dick,” I might mean that I had read the story in several languages and liked it each time I read it. In this case, I would be referring to the propositional information thing (PIT) that “Moby Dick” can denote. This concept is important, as it is what allows Cyc to be independent of language. A PIT that is expressed in CycL (a portion of which is given on the slide) can then be expressed in any desired language because PIT’s are independent of AIS’s (the sequence of symbols). One PIT could have multiple AIS’s.



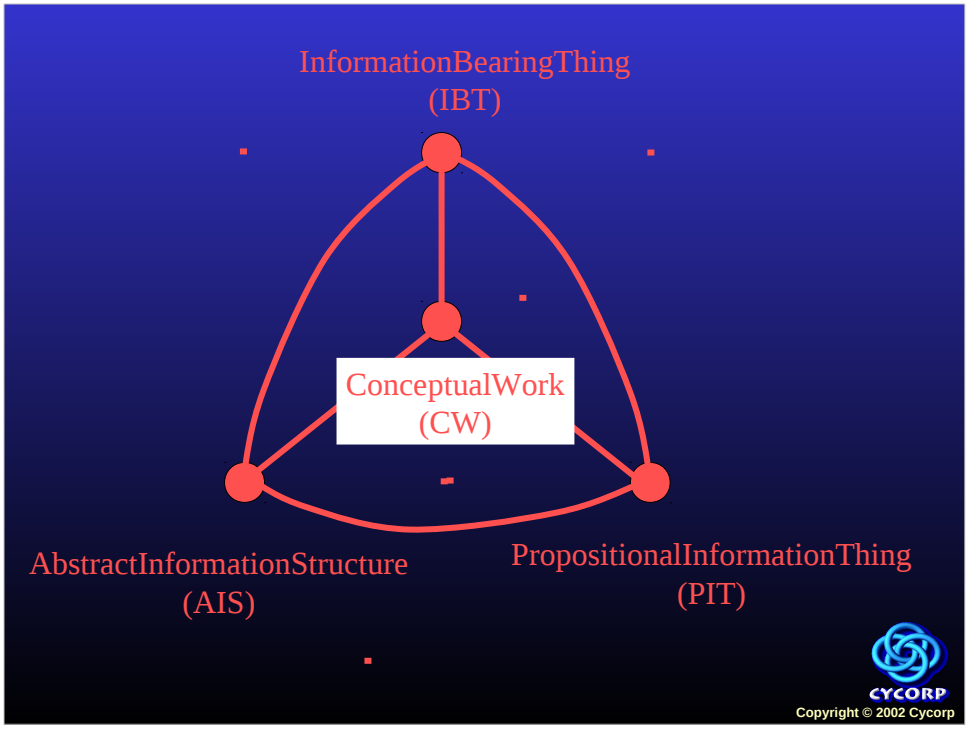
Since most people don't normally use the name of a book to refer to the paper on which it is printed, when they say "Moby Dick," they are probably *not* using "Moby Dick" as an information-bearing thing (IBT).

Similarly, people don't normally use the name of a book to refer to the enormously long sequence of symbols (letters) that make the text of the book (the sequence of all of the letters in the entire work, including the small sequence on the slide, "'-T-i-s--M-o-b-y--D-i-c-k-!"). Thus, "Moby Dick" probably does not refer to an abstract information structure (AIS).

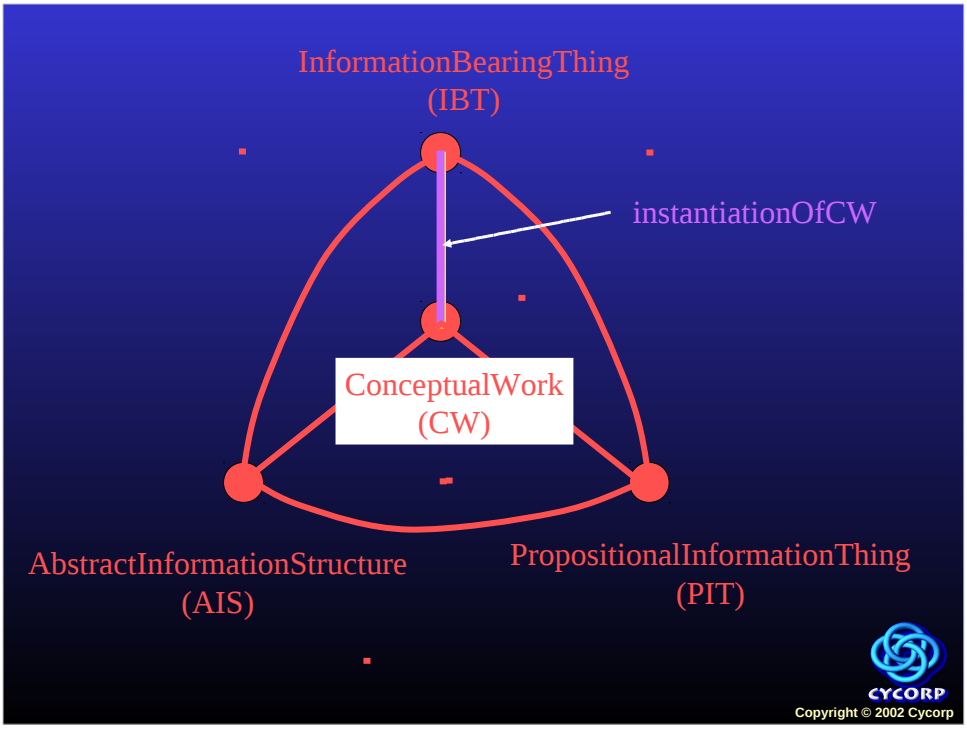
In the same way that "Don Quixote" probably refers to a novel that is written in Spanish, when people say "Moby Dick" they are probably referring to the original English version. In fact, some experts might argue as to the validity of reading the novel in a language other than the original English. Thus, when people say "Moby Dick," they are probably not referring just to the propositional information thing (PIT), but to something more, which includes the language in which it was written.



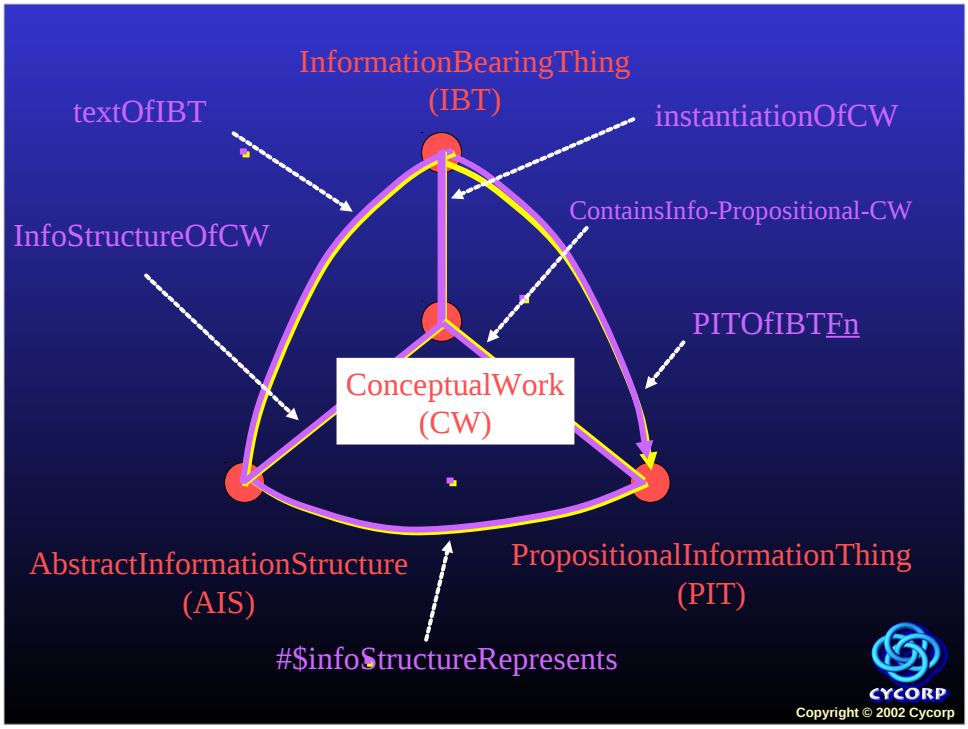
Hence, we know that “Moby Dick” denotes something that does not fit into any one of the three obvious categories for representing information. So we now understand the need for the fourth category -- Conceptual Works. When someone says “Moby Dick” they are probably referring to the thing that Cyc knows as #$MobyDickTheBook-CW.



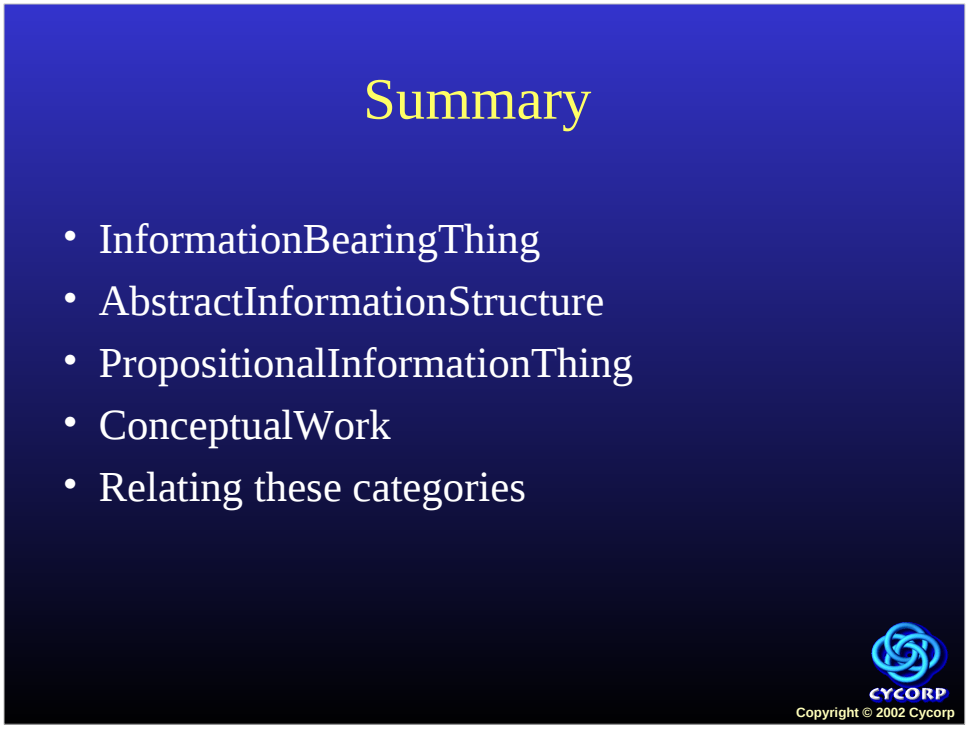
#$MobyDickTheBook-CW is embodied in thousands of different IBT's around the world (all the copies of all the editions, in all the libraries, homes, schools, etc.) and is represented by a specific AIS, and is associated with a specific PIT.



In Cyc, we relate conceptual works to other things, like IBT's, via a large number of relations. Although these relations have no common-sense usage, they allow us to specify to Cyc the part of the conceptual work to which we are referring. So, in order to refer to a specific copy of "Moby Dick" we would refer to the IBT that is an #$instantiationOfCW of "Moby Dick."



This slide presents a glimpse of the various relations between conceptual works and the other categories pertaining to information.



This concludes the lesson on how Cyc represents Information.

The purpose of this lesson is to introduce you to some of the content areas in Cyc. We have already studied the Information area. This lesson will provide a more general overview of some of the other content areas in Cyc.

The column on the left of the slide lists some collections of emotional attributes. The collection called #$Abhorrence includes things that are related to feelings of abhorrence. All the degrees of all of the ranges of abhorrence should be in this collection. Entering knowledge about emotions into Cyc is facilitated by using relations like those that are listed on the right of the slide.

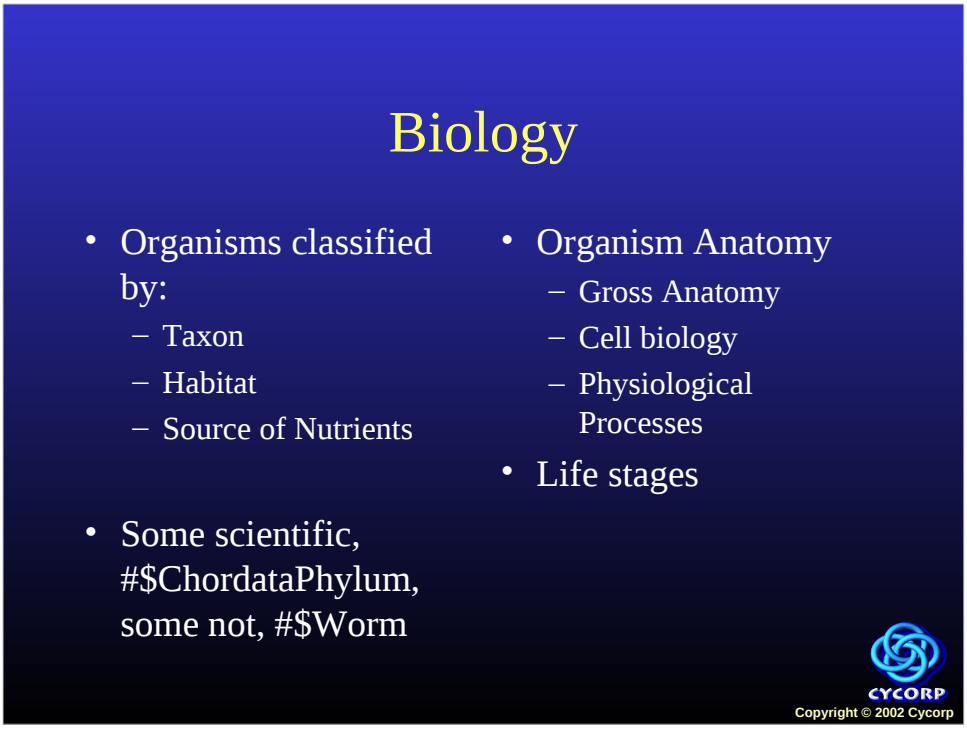
For instance, Cyc knows five different emotions that are related to Abhorrence with the #$contraryFeelings relation. Three of these emotions are #$Enjoyment, #$Adulation, and #$Love. Thus, Cyc knows that something is wrong when a person both abhors and loves a thing.

Similarly, we can tell Cyc which emotions are appropriate for a given situation according to the role that a person plays in the situation. For instance, we can say that a given emotion is an #$AppropriateEmotion for the groom at a wedding, but not for the wedding coordinator.

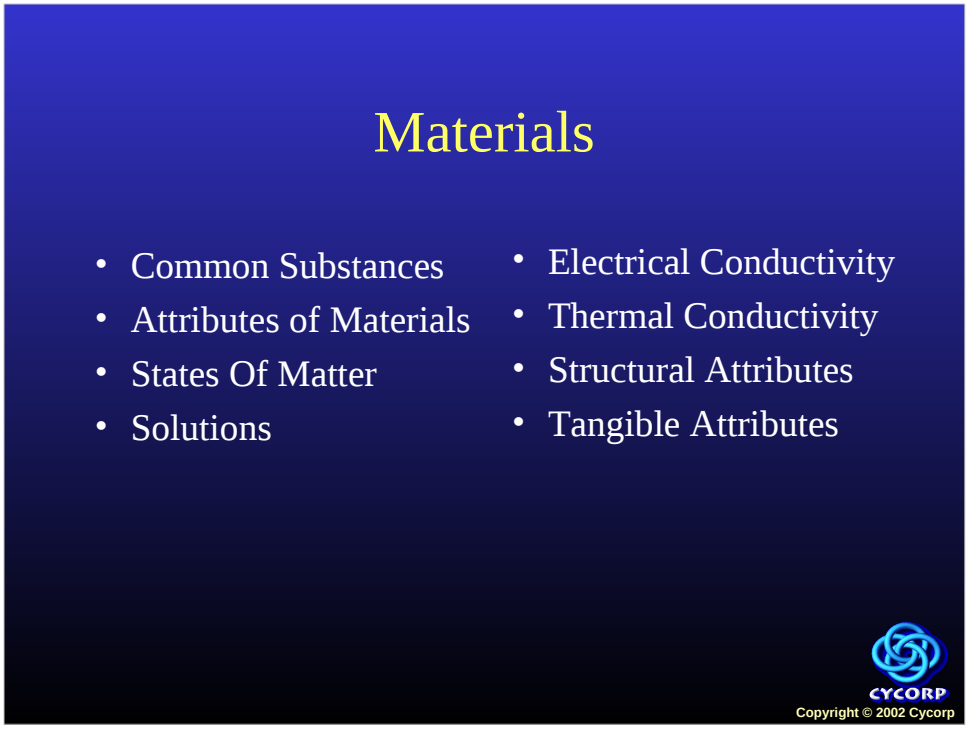
Another emotion-representing relation is #$feelsTowardPersonType. This relation allows us to tell Cyc how a person feels about a group of people. #$actionExpressesFeeling is used to tell Cyc things like laughter expresses amusement.

Another content area in Cyc is Propositional Attitudes. This slide lists some of the predicates that relate intelligent agents to propositions. They are fairly self-explanatory, but we'll discuss the #$desires relation as an example. In the sentence "Jim desires that the sky will be cloudy," Jim is the agent and #$desires would be used to represent the verb phrase "desires that," which relates Jim to the proposition "the sky will be cloudy."

Similar to relating emotion predicates to each other, we can relate propositional attitude predicates to other propositional attitude predicates. For example, Cyc can infer that if Jim #$knows something to be true, then Jim does not hope it will be true; rather, Jim believes it is true, and it is not his goal for it to be true.



Yet another another content area in Cyc is biology. In Cyc, organisms are classified according to their biological taxon (such as #$Rat, #$Vertebrate, and #$Bee), their habitat (where they tend to live), and what they eat. Some classifications are scientific, while others are not, like #$Worm. Cyc can also take anatomical descriptions of organisms that say what organs they have and how they are related (for example, how they work in processes like metabolism and digestion). We can even describe an organism's life stages to Cyc.



Cyc has a diverse Materials vocabulary for describing types of substances, physical properties of substances, and tangible attributes of substances.

You can discuss all states of matter (solid, liquid, etc.) in Cyc. This allows us to assert, for example, that glass is classified as a solid tangible thing and yet exists in a liquid state.

Structural Attributes include things like colloid and crystalline.

Tangible attributes are used to describe the perception of how something feels. It is important, however, to note that something that is a solid tangible thing is not necessarily in a solid state of matter. For example, consider a piece of wood. That piece is a solid tangible thing in that it can be touched and its shape is independent from its container, but the water that is in it is in a liquid state of matter and that water can account for 20 to 60 percent of the piece of wood.

Cyc knows about lots of types of devices, from #$BlowDryer, #$Gun, #$Engine, #$PhotocopyMachine, #$Fishhook, and #$Wheel, to #$AtomicClock.

Various attributes of devices can be represented within Cyc. You can describe the state of a device, like #$PausedState, #$DeviceOn, and #$OffHook (for a phone). You can describe actions of devices, like the spinning of a washing machine. You can describe the power rating, operating cost, etc. of a device. Finally, you can tell Cyc the purpose of a device with the #$PrimaryFunction predicate. For example, the primary function of a hammer is to be used in a hitting event.

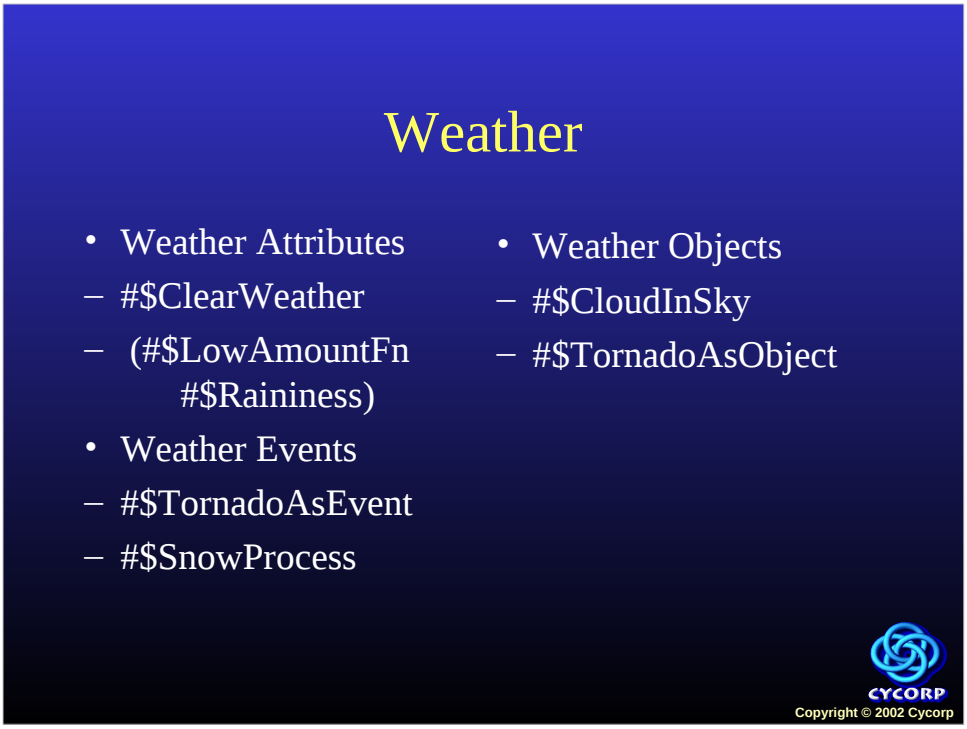
Another content area in Cyc is food. Cyc knows a lot about food.

There are various food types in Cyc, such as the #$Fruit collection, the #$meat collection, etc.

You can tell Cyc who can typically eat a food with the #$EdibleByFn function.

#$PreparingFoodOrDrink is the collection of events that are associated with preparing a #$FoodAndDrink item (whether you start “from scratch” or use a mix). Similarly, #$ConsumingFoodOrDrink is the collection of events in which a person or other animal ingests some portion of food or drink through its mouth

Cyc also has more specific food-related vocabulary. For example, to say that a person is a vegetarian, you can say that he or she #$eatsWillingly #$VegetarianCuisine.



All normal human expressions about weather, another content area in Cyc, can be expressed in CycL.

You would say that a tornado occurred on Thursday by referring to an instance of the #$TornadoAsEvent collection.

You would distinguish the snowing event from snow itself by referring to an instance of the collection called #$SnowProcess.

In order to say that a tornado hit my house on Thursday, you'd refer to an instance of the collection called #$TornadoAsObject.

Cyc has an extensive vocabulary for describing the geography content area.

Cyc has various ways of encoding physical addresses, breaking them down into street, zip code, etc.

Cyc also has various predicates for describing characteristics of the populace like religion, language, etc.

Cyc has lots of predicates for physical relations between things (like borders), geo-political subdivisions (like voting districts), and natural land divisions (like islands and seas).

Summary

- Emotion
- Propositional Attitudes
- Biology
- Materials
- Devices
- Food
- Weather
- Geography

This concludes the lesson on additional content areas in Cyc.