Description Logics

What Are Description Logics?

- **A family of logic based KR formalisms**
  - **Descendants of** semantic networks **and** KL-ONE
  - **Describe domain in terms of** concepts **(classes),** roles **(relationships) and** individuals
- **Distinguished by:**
  - Formal semantics **(typically model theoretic)**
    - **Decidable fragments of FOL**
    - **Closely related to Propositional Modal & Dynamic Logics**
  - **Provision of** inference services
    - **Sound and complete decision procedures for key problems**
    - **Implemented systems (highly optimized)**

Description Logics

- Major focus of KR research in the 80's
  - Led by Ron Brachman – (AT&T Labs)
  - Grew out of early network-based KR systems like semantic networks and frames.
- Major systems and languages –
  - 80s: KL-ONE, NIKL, KANDOR, BACK, CLASSIC, LOOM
  - 90s: FACT, RACER, …
  - 00s: DAML+OIL, OWL, Pellet, Jena, FACT++
- Used as the basis for the Semantic web languages DAML+OIL and OWL
- Some commercial systems

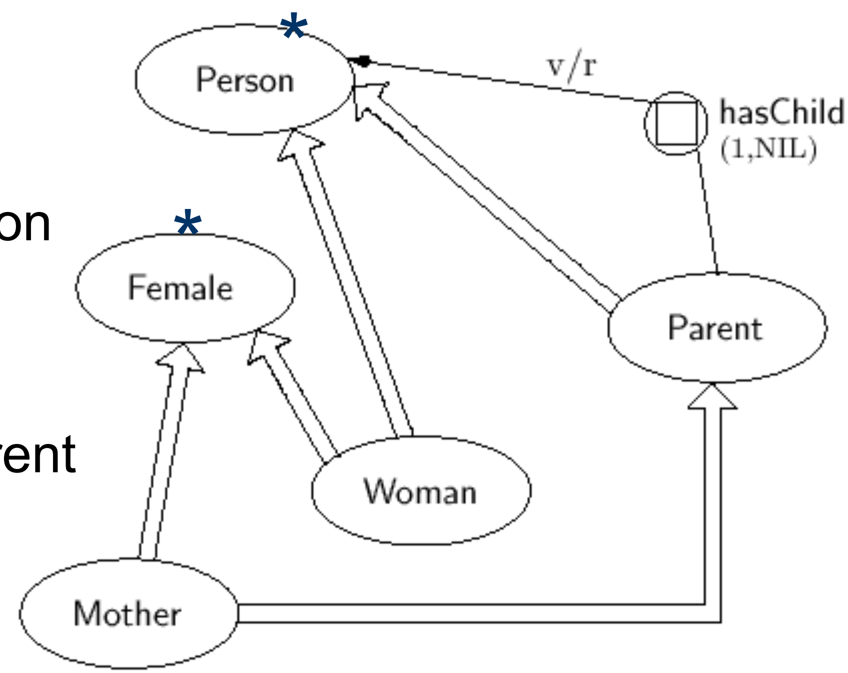
Description Logics

- Thought to be well-suited for the representation of and reasoning about
  - ontologies
  - terminological knowledge
  - Configurations and configuration problems
  - database schemata
    - schema design, evolution, and query optimization
    - source integration in heterogeneous databases/ data warehouses
    - conceptual modeling of multidimensional aggregation

Example of Network KR

- Person, Female, etc. are concepts
- hasChild is a property of Person
  - hasChild relates Parent to Person
  - Nil means infinity. A Parent is a Person with between 1 and infinity children
- Large arrows are "IS-A" links
  - A Mother is a (specialization of a) Parent
- Concepts are either primitive or definitions.
  - Primitive concepts have only *necessary* properties
  - Defined concepts have *necessary* and *sufficient* conditions

Graphical notation introduced by KL-ONE



DL Paradigm

- A **Description Logic** is mainly characterized by a set of constructors that allow one to build complex descriptions or terms out of **concepts** and **roles** from atomic ones
  - **Concepts** correspond to classes
    - and are interpreted as sets of objects,
  - **Roles** correspond to relations
    - and are interpreted as binary relations on objects
- Set of axioms for asserting **facts** about concepts, roles and **individuals**

Basic Concepts of a DL

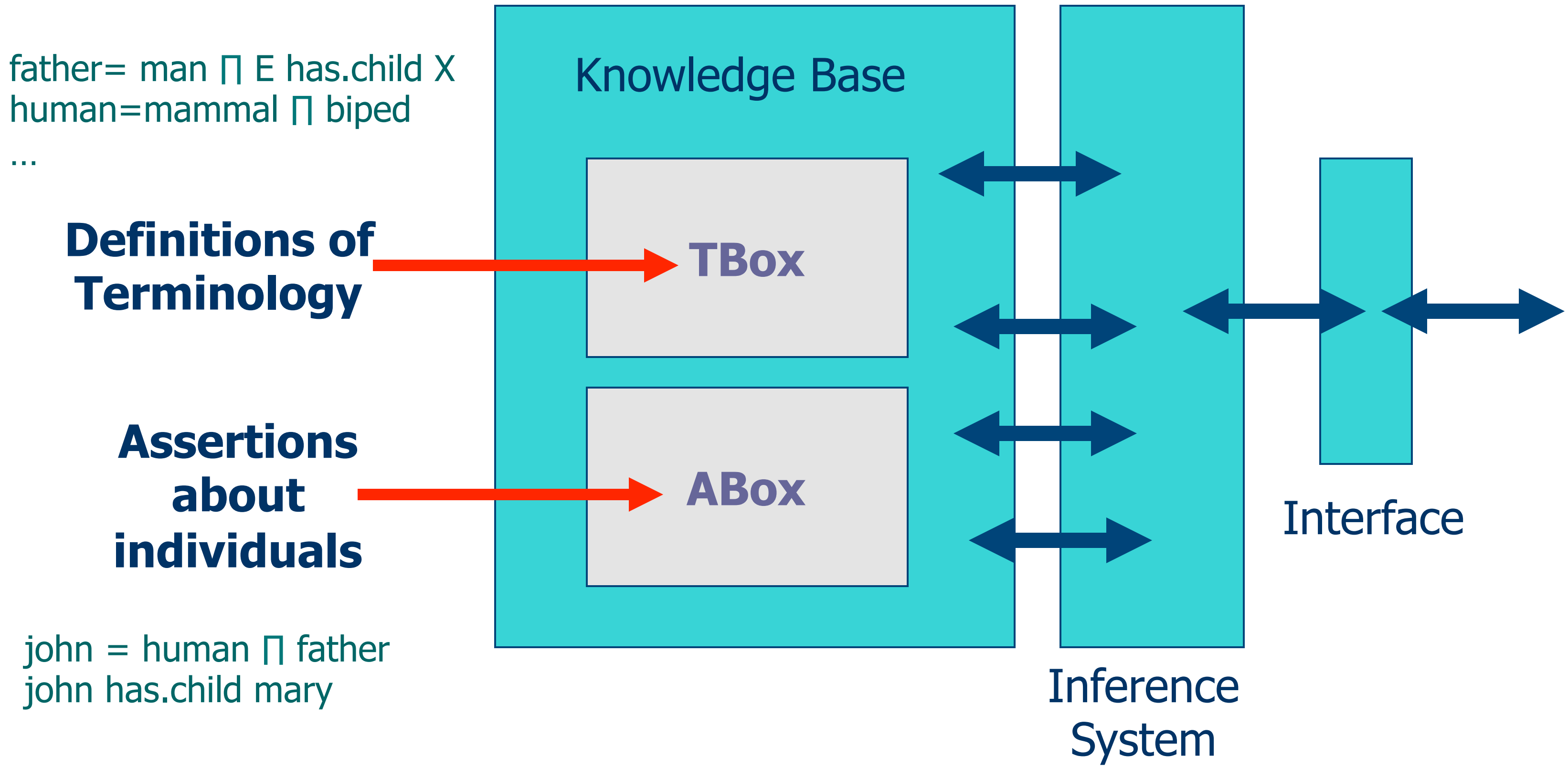
- Individuals are treated exactly the same as constants in FOL
  - *john*
- Concepts are exactly the same as Unary Predicates in FOL
  - *Person(john)*
- Roles are exactly the same as Binary Predicates in FOL
  - *has_mother(john, mary)*

Descriptions

- As in FOL, we are dealing with (ultimately) sets of individuals and relations between them
- The basic unit of semantic significance is the *Description*
- *"We are **describing** sets of individuals"*
- Description logics differ in the operators allowed
- If a "happy father" is a man with both a son and daughter and all of whose children are either rich or happy, then we describe it in DL as

  HappyFather = Man ∩ ∃hasChild.Female ∩ ∃hasChild.Male ∩ ∀hasChild.(Rich ∪ Happy)

Typical Architecture



The division into TBox and ABox doesn't have a logical significance, but is made for conceptual and implementation convenience.

A family of languages

- The expressiveness of a description logic is determined by the operators that it uses
- Add or eliminate certain operators (e.g., $\neg$, $\cup$), and the statements that can be expressed are increased/reduced in number
- Higher expressiveness implies higher complexity
- *AL* or *Attributive Language* is the base and includes just a few operators
- Other DLs are described by the additional operators they include

AL: Attributive Language

Constructor	**Syntax**	**Example**
atomic concept	C	Human
atomic negation	~ C	~ Human
atomic role	R	hasChild
conjunction	C ∧ D	Human ∧ Male
value restriction	R.C	Human ∃ hasChild.Blond
existential rest. (lim)	∃R	Human ∃ hasChild
Top (univ. conc.)	T	T
bottom (null conc)	⊥	⊥

for concepts C and D and role R

ALC

ALC is the smallest DL that is propositionally closed (i.e., includes full negation and disjunction) and include booleans (and, or, not) and restrictions on role values

constructor	**Syntax**	**Example**
atomic concept	C	Human
negation	**~ C**	**~ (Human V Ape)**
atomic role	R	hasChild
conjunction	C ^ D	Human ^ Male
disjunction	**C V D**	**Nice V Rich**
value restrict.	∃R.C	Human ∃ hasChild.Blond
existential rest.	**∃R.C**	**Human ∃ hasChild.Male**
Top (univ. conc.)	T	T
bottom (null conc)	⊥	⊥

Other Constructors

Constructor	Syntax	Example
number restriction	>= n R	>= 7 hasChild
	<= n R	<= 1 hasmother
inverse role	R-	haschild-
Transitive role	R*	hasChild*
Role composition	R ∘ R	hasParent ∘ hasBrother
Qualified # restric.	>= n R.C	>= 2 hasChild.Female
Singleton concepts	{<name>}	{Italy}

∀ and ∃ deserve special attention.

- Note that they only can come before a Role:

  $\forall HasChild.Girl$ $\exists isEmployedBy.Farmer$

- Remember, they describe sets of individuals.

- $\forall HasChild.Girl$ *would be interpreted as:*

  The set { x | ∀(y)(HasChild(x,y) → Girl(y)) }

  Note the conditional: Are you in that set?.

- $\exists isEmployedBy.Farmer$ *would be:*

  The set { x | ∃(y)(isEmployedBy(x,y) ∧ Farmer(y)) }

Special names and combinations

See http://en.wikipedia.org/wiki/Description_logic

- S = ALC + transitive properties
- H = role hierarchy, e.g., rdfs:subPropertyOf
- O = nominals, e.g., values constrained by enumerated classes, as in owl:oneOf and owl:hasValue
- I = inverse properties
- N = cardinality restrictions (owl:cardinality, maxCardonality)
- $^{(D)}$ = use of datatypes properties
- R = complex role axioms (e.g. (ir)reflexivity, disjointedness)
- Q = Qualified cardinality (e.g., at least two female children)

➔ **OWL-DL is SHOIN$^{(D)}$**

➔ **OWL 2 is SROIQ$^{(D)}$**

DL Description Logic Complexit ×

www.cs.man.ac.uk/~ezolin/dl/

Complexity of reasoning in Description Logics

Note: the information here is (always) incomplete and updated often

Base description logic: *A*ttributive *L*anguage with *C*omplements

$\mathcal{ALC}$ ::= ⊥ | ⊤ | A | $\neg C$ | $C \sqcap D$ | $C \sqcup D$ | $\exists R.C$ | $\forall R.C$

Concept constructors:

- ☐ $\mathcal{F}$ – functionality[2]: $(\leq 1\ R)$
- ☐ $\mathcal{N}$ – (unqualified) number restrictions: $(\geq n\ R)$, $(\leq n\ R)$
- ☐ $\mathcal{Q}$ – qualified number restrictions: $(\geq n\ R.C)$, $(\leq n\ R.C)$
- ☐ $\mathcal{O}$ – nominals: $\{a\}$ or $\{a_1, \ldots, a_n\}$ ("one-of")
- ☐ μ – least fixpoint operator: $\mu X.C$

Forbid complex roles[5] in number restrictions[6]

Role constructors:

trans reg

- ☐ $\mathcal{I}$ – role inverse: R^-
- ☐ ∩ – role intersection[3]: $R \cap S$
- ☐ ∪ – role union: $R \cup S$
- ☐ ¬ – role complement: $\neg R$ full
- ☐ o – role chain (composition): $R \circ S$
- ☐ * – reflexive-transitive closure[4]: R^*
- ☐ id – concept identity: $id(C)$

TBox (concept axioms):

- ◉ empty TBox
- ○ acyclic TBox ($A \equiv C$, A is a concept name; no cycles)
- ○ general TBox ($C \sqsubseteq D$, for arbitrary concepts C and D)

RBox (role axioms):

OWL-Lite OWL-DL OWL 1.1

- ☐ $\mathcal{S}$ – role transitivity: $Tr(R)$
- ☐ $\mathcal{H}$ – role hierarchy: $R \sqsubseteq S$
- ☐ $\mathcal{R}$ – complex role inclusions: $R \circ S \sqsubseteq R$, $R \circ S \sqsubseteq S$
- ☐ s – some additional features (check it to see)

Reset You have selected a Description Logic: $\mathcal{ALC}$

Complexity of reasoning problems[7]

Reasoning problem	Complexity[8]	Comments and references
Concept satisfiability	**PSpace-complete**	• Hardness for $\mathcal{ALC}$: see [80]. • Upper bound for $\mathcal{ALCQ}$: see [12, Theorem 4.6].
ABox consistency	**PSpace-complete**	• Hardness follows from that for concept satisfiability. • Upper bound for $\mathcal{ALCQO}$: see [17, Appendix A].

Important properties of the description logic

Finite model property	**Yes**	$\mathcal{ALC}$ is a notational variant of the multi-modal logic $\mathbf{K}_m$ (cf. [77]), for which the finite model property can be found in [4, Sect. 2.3].
Tree model property	**Yes**	$\mathcal{ALC}$ is a notational variant of the multi-modal logic $\mathbf{K}_m$ (cf. [77]), for which the tree model property can be found in [4, Proposition 2.15].

Maintained by: Evgeny Zolin
Please see the list of updates

Any comments are welcome:
EZolin@cs.man.ac.uk

http://www.cs.man.ac.uk/~ezolin/dl/

Notes:

1. The letters $\mathcal{O}$, $\mathcal{I}$ and $\mathcal{Q}$ are customary written in various orders, e.g., $\mathcal{ALCQIO}$ but $\mathcal{SHOIQ}$. Here we do not reflect this tradition, but rather use a uniform naming scheme.

OWL as a DL

- OWL-DL is $SHOIN^{(D)}$
- We can think of OWL as having three kinds of statements
- Ways to specify classes
  - the intersection of humans and males
- Ways to state axioms about those classes
  - Humans are a subclass of apes
- Ways to talk about individuals
  - John is a human, john is a male, john has a child mary

OWL Class Constructors

Constructor	DL Syntax	Example	(Modal Syntax)
intersectionOf	$C_1 \sqcap \ldots \sqcap C_n$	Human $\sqcap$ Male	$C_1 \wedge \ldots \wedge C_n$
unionOf	$C_1 \sqcup \ldots \sqcup C_n$	Doctor $\sqcup$ Lawyer	$C_1 \vee \ldots \vee C_n$
complementOf	$\neg C$	$\neg$Male	$\neg C$
oneOf	$\{x_1 \ldots x_n\}$	{john, mary}	$x_1 \vee \ldots \vee x_n$
allValuesFrom	$\forall P.C$	$\forall$hasChild.Doctor	$[P]C$
someValuesFrom	$\exists P.C$	$\exists$hasChild.Lawyer	$\langle P \rangle C$
maxCardinality	$\leqslant n P$	$\leqslant$1hasChild	$[P]_{n+1}$
minCardinality	$\geqslant n P$	$\geqslant$2hasChild	$\langle P \rangle_n$

- XMLS **datatypes** as well as classes in $\forall P.C$ and $\exists P.C$
  - E.g., $\exists$hasAge.nonNegativeInteger
- Arbitrarily complex **nesting** of constructors
  - E.g., Person $\sqcap$ $\forall$hasChild.(Doctor $\sqcup$ $\exists$hasChild.Doctor)

OWL Axioms

Axiom	DL Syntax	Example
subClassOf	$C_1 \sqsubseteq C_2$	Human $\sqsubseteq$ Animal $\sqcap$ Biped
equivalentClass	$C_1 \equiv C_2$	Man $\equiv$ Human $\sqcap$ Male
disjointWith	$C_1 \sqsubseteq \neg C_2$	Male $\sqsubseteq$ $\neg$Female
sameIndividualAs	$\{x_1\} \equiv \{x_2\}$	{President_Bush} $\equiv$ {G_W_Bu
differentFrom	$\{x_1\} \sqsubseteq \neg\{x_2\}$	{john} $\sqsubseteq$ $\neg${peter}
subPropertyOf	$P_1 \sqsubseteq P_2$	hasDaughter $\sqsubseteq$ hasChild
equivalentProperty	$P_1 \equiv P_2$	cost $\equiv$ price
inverseOf	$P_1 \equiv P_2^-$	hasChild $\equiv$ hasParent$^-$
transitiveProperty	$P^+ \sqsubseteq P$	ancestor$^+$ $\sqsubseteq$ ancestor
functionalProperty	$\top \sqsubseteq \leqslant 1 P$	$\top \sqsubseteq \leqslant$1hasMother
inverseFunctionalProperty	$\top \sqsubseteq \leqslant 1 P^-$	$\top \sqsubseteq \leqslant$1hasSSN$^-$

- $\mathcal{I}$ **satisfies** $C_1 \sqsubseteq C_2$ iff $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$; satisfies $P_1 \sqsubseteq P_2$ iff $P_1^{\mathcal{I}} \subseteq P_2^{\mathcal{I}}$
- $\mathcal{I}$ satisfies ontology $\mathcal{O}$ (is a **model** of $\mathcal{O}$) iff satisfies every axiom in $\mathcal{O}$

Subsumption: $D \subseteq C$?

- Concept C subsumes D iff on every interpretation *I*
  - $I(D) \subseteq I(C)$
- This means the same as (for complex statements D and C) the assertion:
  - $\forall(x)(D(x) \rightarrow C(x))$
- Determining whether one concept *logically* contains another is called the *subsumption problem.*
- Subsumption is undecidable for reasonably expressive languages
  - e.g.; for FOL: does one FOL sentence imply another
- and non-polynomial for fairly restricted ones

Other reasoning problems

These problems can be reduced to subsumption (for languages with negation) and to the satisfiability problem, as well

- **Concept satisfiability** is C empty?
- **Instance Checking** Father(john)?
- **Equivalence** CreatureWithHeart ≡ CreatureWithKidney
- **Disjointness** C ⊓ D
- **Retrieval** Father(X)? X = {john, robert}
- **Realization** X(john)? X = {Father}

Definitions

- A **definition** is a description of a concept or a relationship
- It is used to assign a meaning to a term
- In description logics, definitions use a specialized logical language
- Description logics are able to do limited reasoning about concepts expressed in their logic
- One important inference is classification (computation of subsumption)

Necessary vs. Sufficient

- *Necessary* properties of an object are properties common to all objects of that type
  - Being a man is a necessary condition for being a father
- *Sufficient* properties are properties that allow one to identify an object as belonging to a type and need not be common to all members of the type
  - Speeding is a sufficient reason for being stopped by the police
- Definitions often specify both *necessary and sufficient* properties

Subsumption

- Meaning of Subsumption

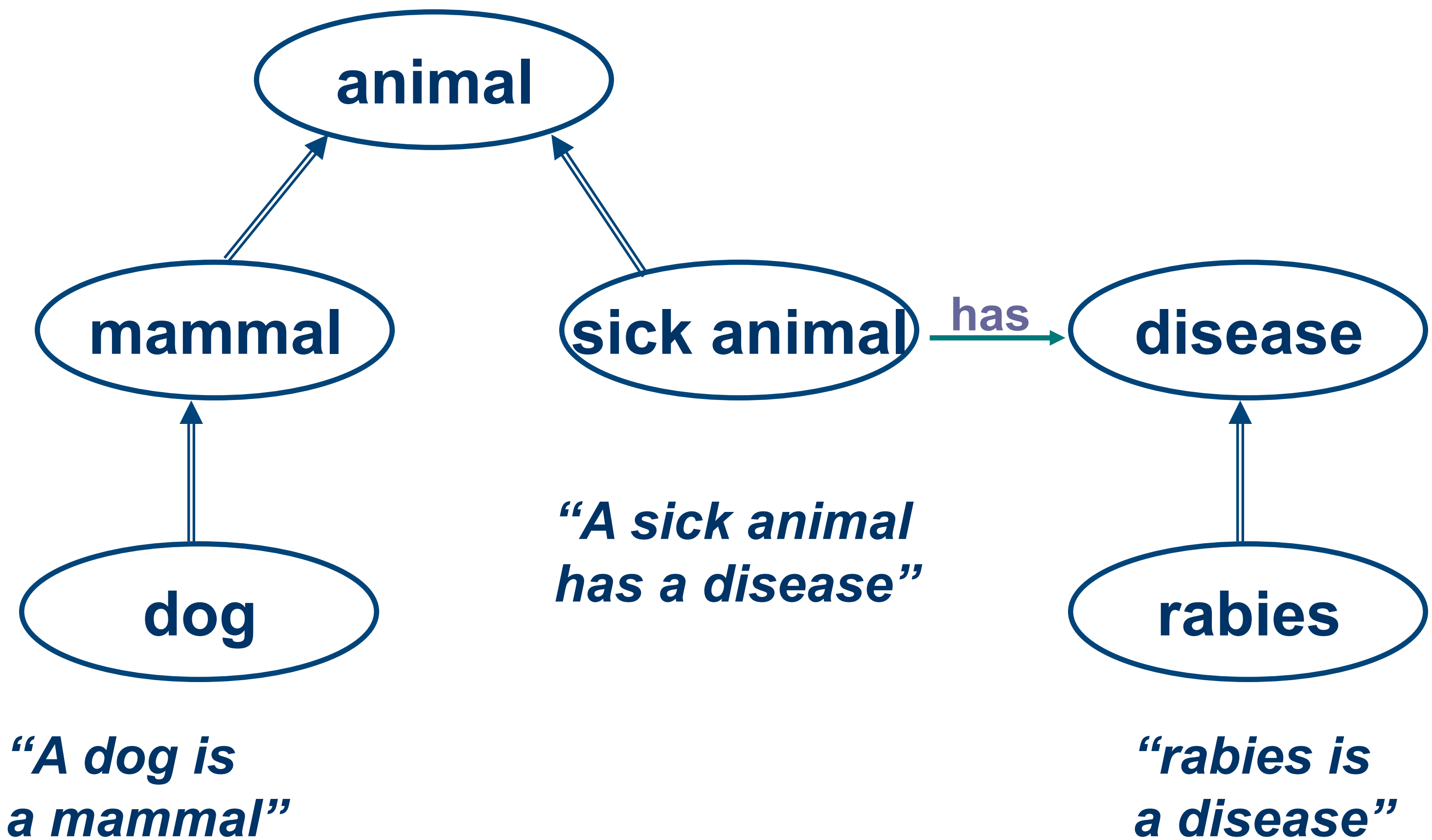
  A more general concept or description is said to subsume a more specific one. Members of a subsumed concept are necessarily members of a subsuming concept

- Two ways to formalize the meaning of subsumption
  - Using logic
    - Satisfying a subsumed concept implies that the subsuming concept is satisfied also

      E.g., if john is a person, he is also an animal
  - Using set theory
    - The instances of subsumed concept are necessarily a subset of the subsuming concept's instances

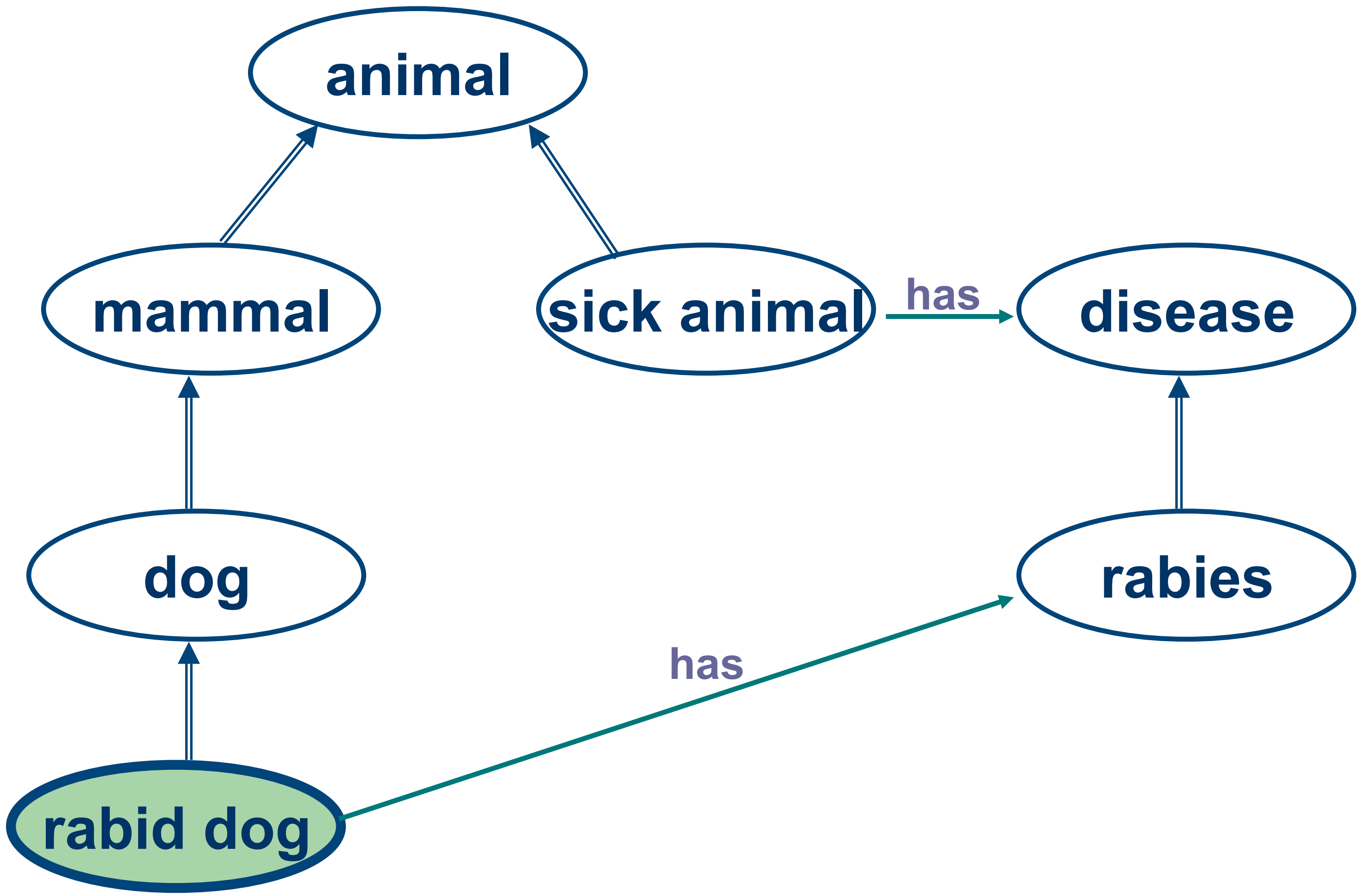
      E.g., the set of all persons is a subset of all animals

How Does Classification Work?



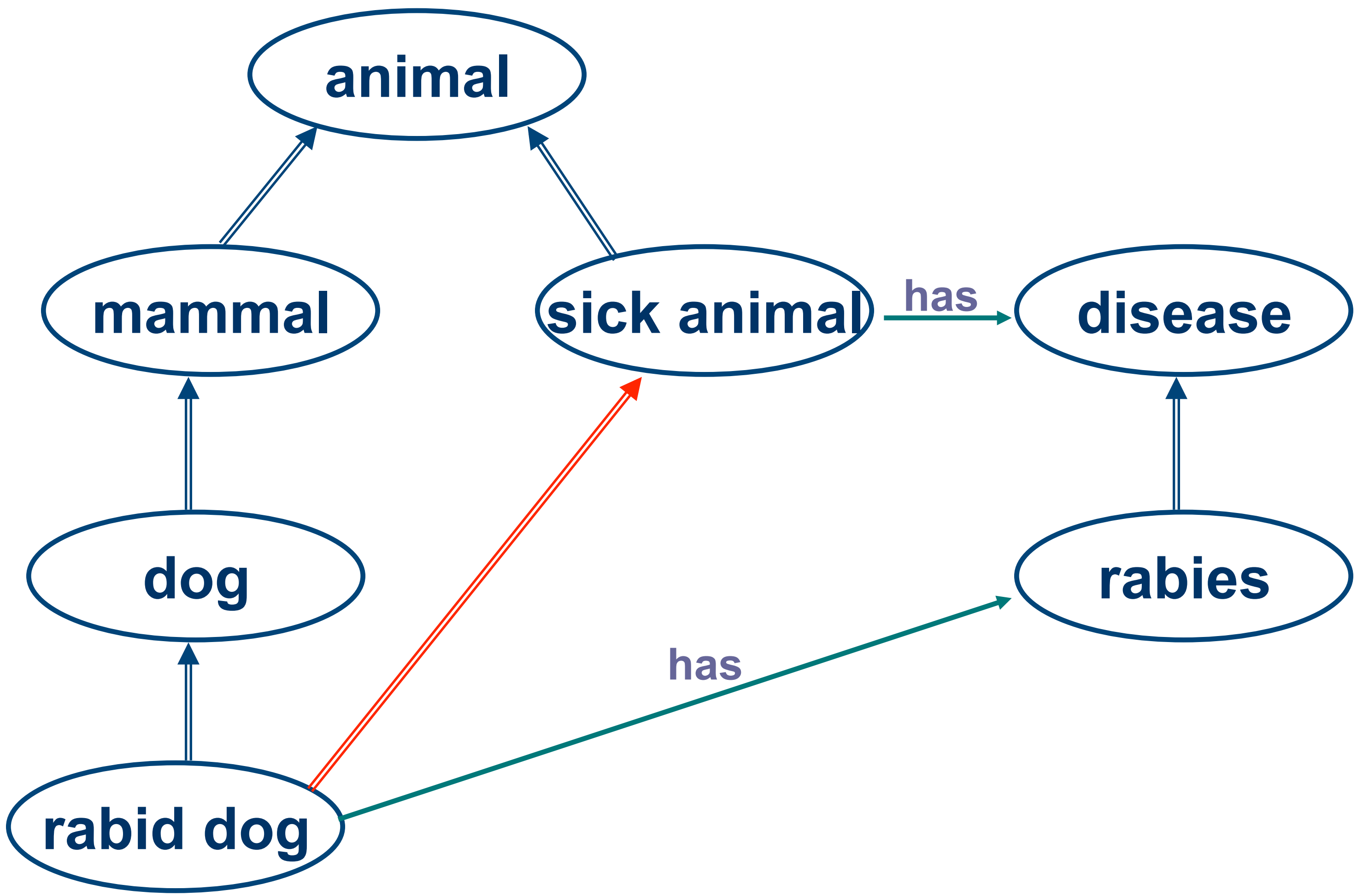
A sick animal is defined as something that is both an animal and has at least one thing that is a kind of a disease

Defining a “rabid dog”



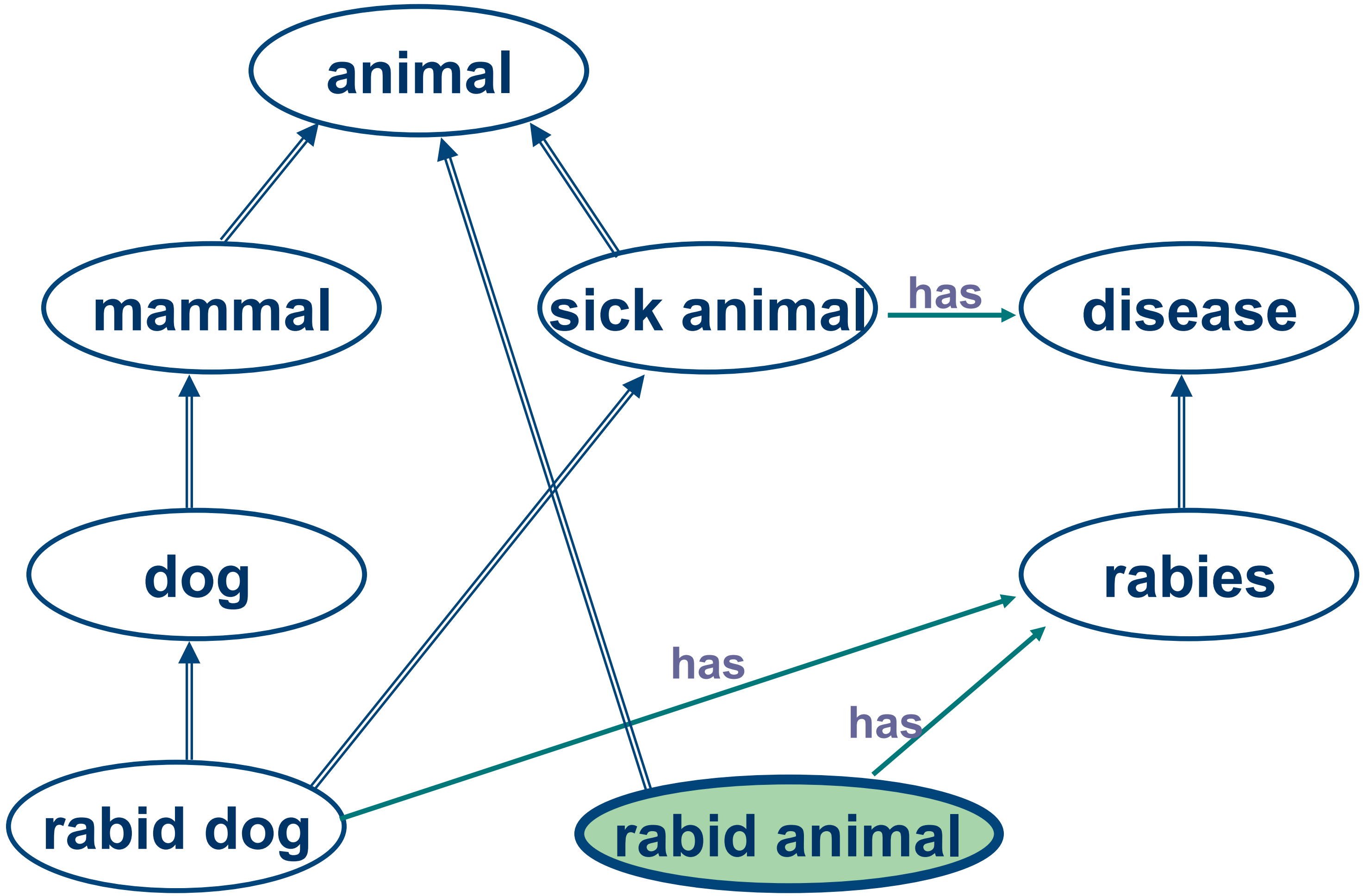
A rabid dog is defined as something that is both a dog and has at least one thing that is a kind of a rabies

Classification as a “sick animal”



We can easily prove that s rabid dog is a kind of sick animal

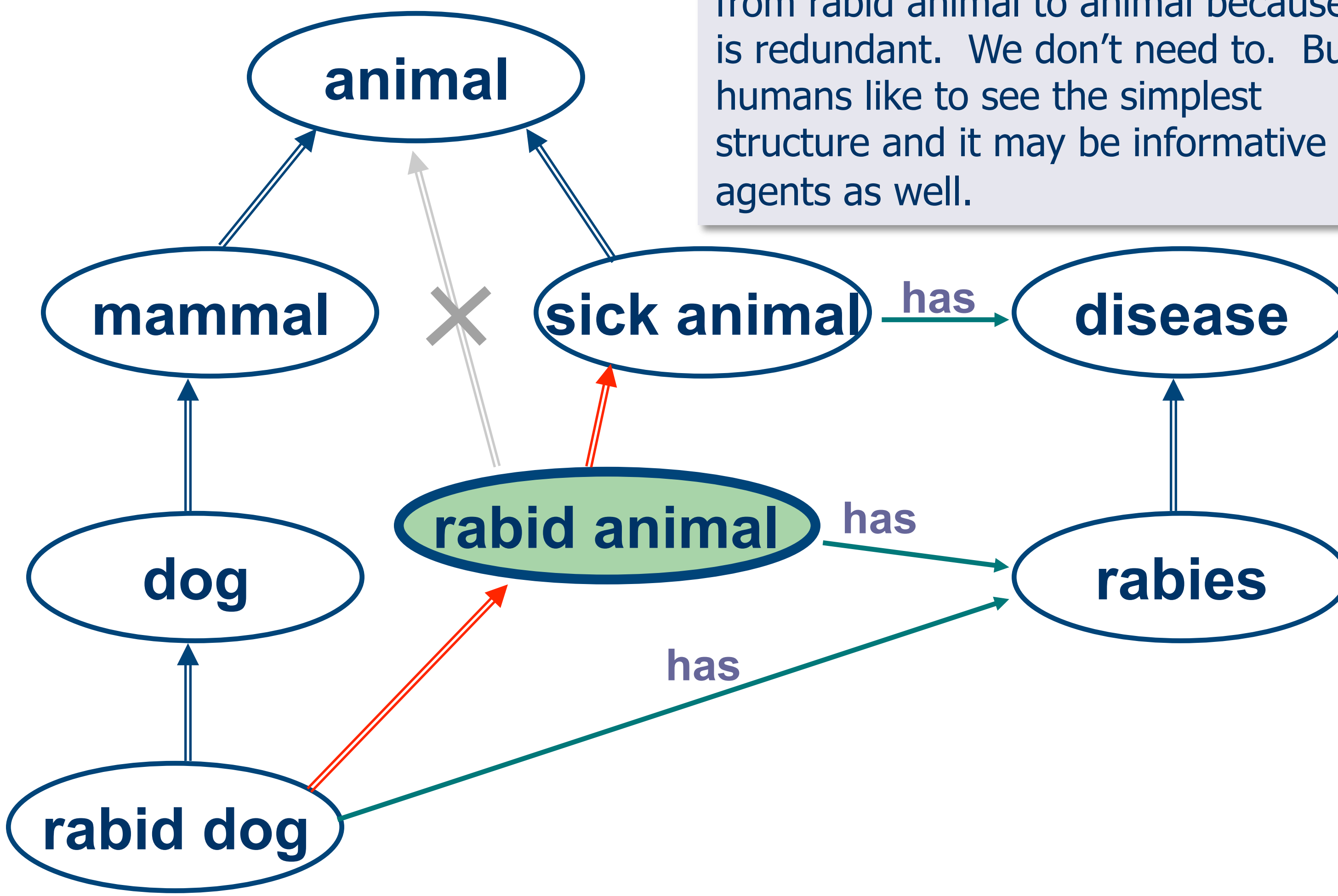
Defining “rabid animal”



A rabid animal is defined as something that is both an animal and has at least one thing that is a kind of a rabies

Loom Places Concept in Hierarchy

Note: we can remove the subclass link from rabid animal to animal because it is redundant. We don't need to. But humans like to see the simplest structure and it may be informative for agents as well.



We can easily prove that s rabid dog is a kind of rabid animal

Primitive versus Structured (Defined)

- Description logics reason with definitions
  - They prefer to have *complete* descriptions
  - A complete definition includes both necessary conditions and sufficient conditions
- This is often impractical or impossible, especially with natural kinds.
- A “primitive” definition is an incomplete definition
  - This limits the amount of classification that can be done automatically
- Example:
  - Primitive: a Person
  - Defined: Parent = Person with at least one child

Intentional versus Extensional Semantics

- **Extensional Semantics** are a model-theoretic idea. They define the meaning of a description by enumerating the set of objects that satisfy the description.
- **Intensional Semantics** defines the meaning of a description based on the intent or use of the description.
- Example:
  - Morning-Star Evening-Star
    - Extensional: Same object, namely Venus
    - Intensional: Different objects, one meaning Venus seen in the morning and one in the evening.

Definition vs. Assertion

- A **definition** is used to describe *intrinsic* proper-ties of an object. The parts of a description have meaning as a part of a composite description of an object
- An **assertion** is used to describe an *incidental* property of an object. Asserted facts have meaning on their own.
- Example: “a black telephone”

  Could be either a description or an assertion, depending on the meaning and import of “blackness” on the concept telephone.

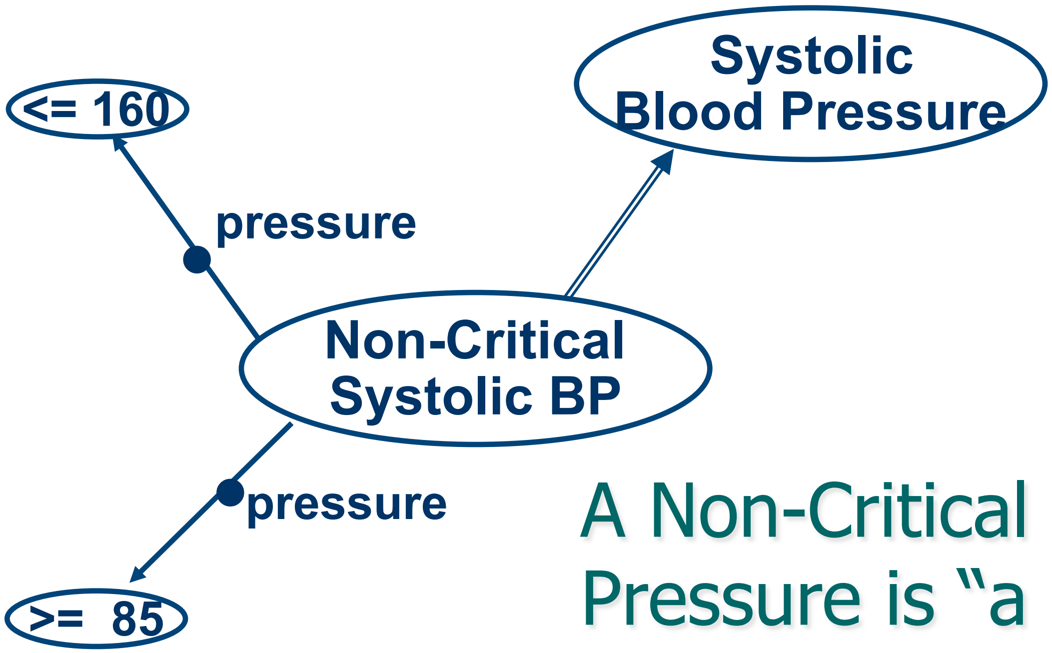
Definition versus Assertion

- In English, “a black telephone” is ambiguous
  - (1) A black telephone is a common sight in an office
  - (2) A black telephone is on the corner of my desk
- KR languages should not be ambiguous so typically distinguish between descriptions of classes and descriptions of individuals
- KR languages often also allow additional assertions to be made that are not part of the definition (In OWL called annotation properties)

Classification is very useful

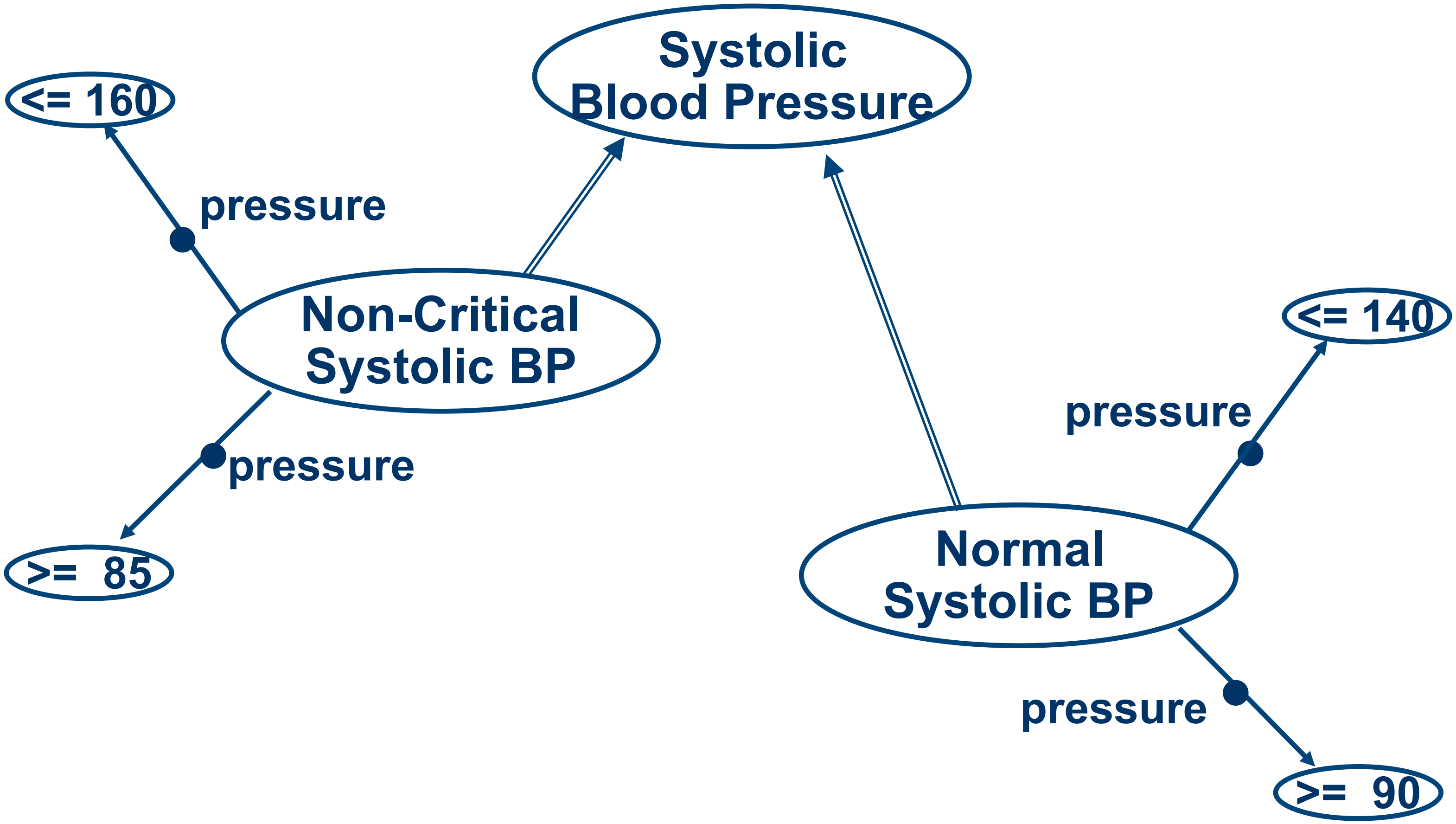
- Classification is a powerful kind of reasoning that is very useful
- Many expert systems can be usefully thought of as doing “heuristic classification”
- Logical classification over structured descriptions and individuals is also quite useful.
- But… can classification ever deduce something about an individual other than what classes it belongs to?
- And what does *that* tell us?

Example: Blood Pressure



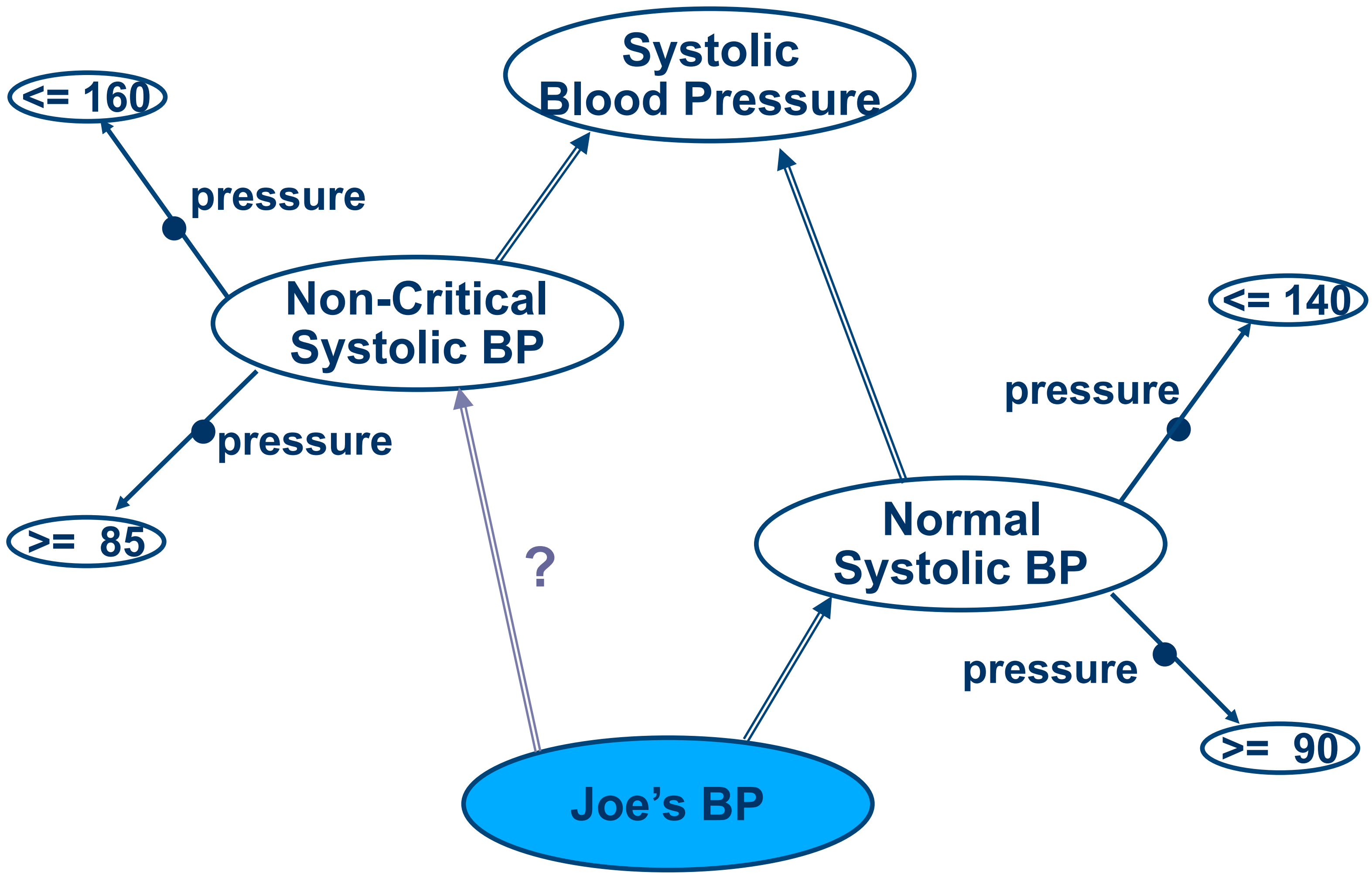
A Non-Critical Blood Pressure is "a Systolic B.P. between 85 and 160."

Example: Blood Pressure

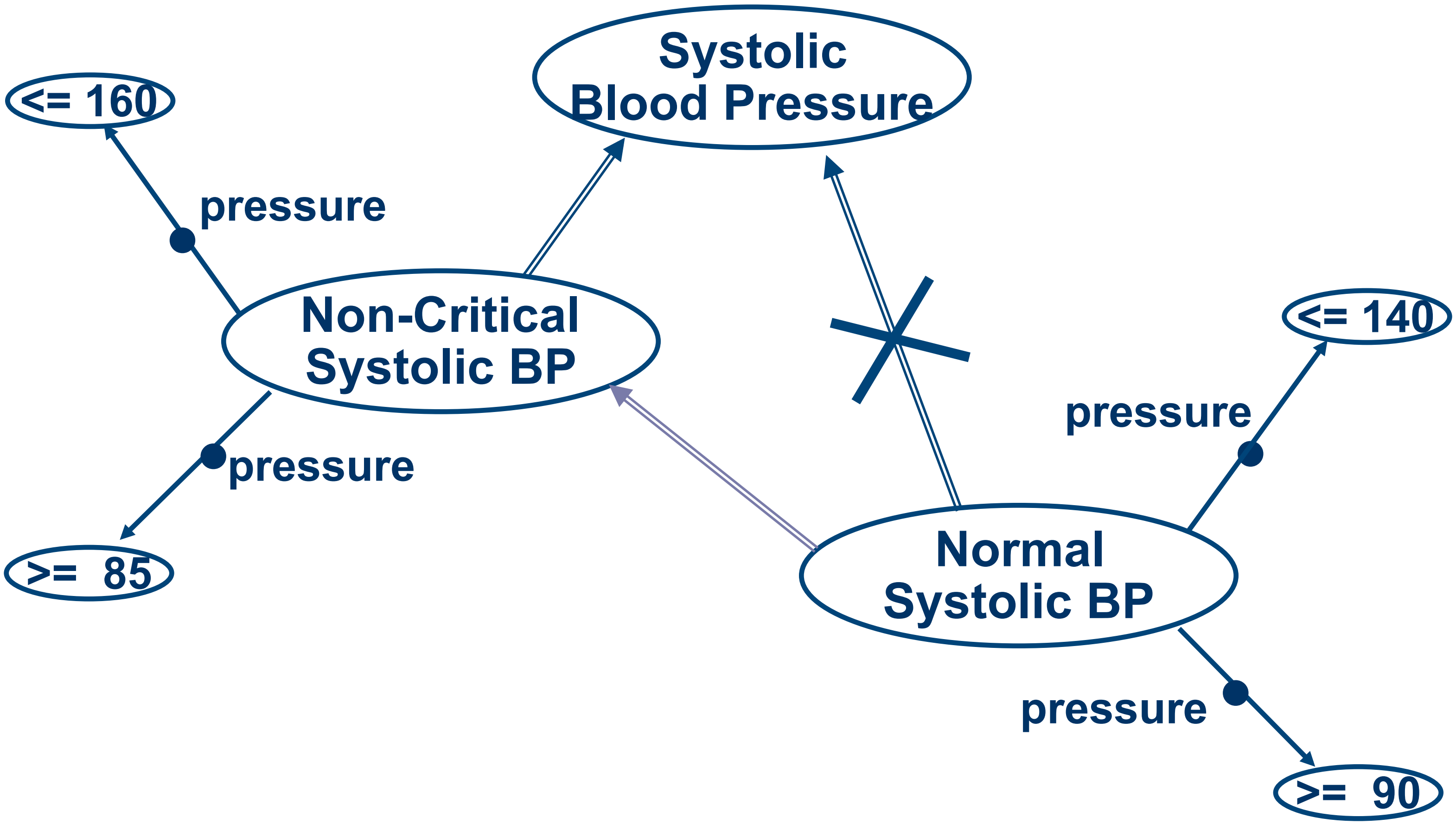


Normal Systolic B.P. is “a Systolic B.P. between 90 and 140.

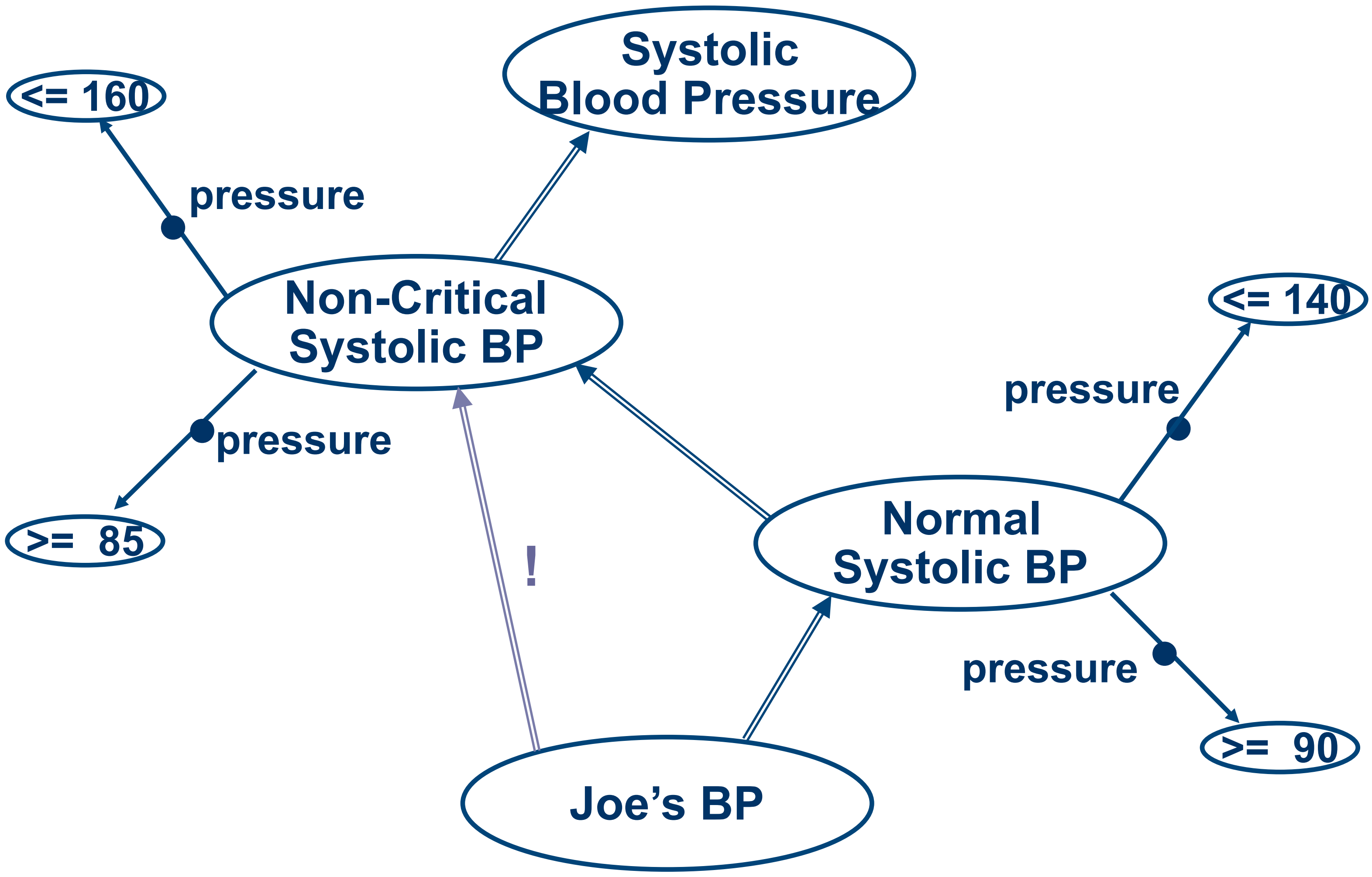
If Joe’s BP is Normal is it also Non-Critical?



Concept Classification Infers Normal BP is Subsumed by Non-Critical BP



With Classified Concepts the Answer is Easy to Compute



Incidental properties

- If we allow incidental properties (e.g., ones that don’t participate in the description mechanism) then these can be deduced via classification

Some DL reasoners

- See http://en.wikipedia.org/wiki/Description_logic
  - CEL, free (for non-commercial use), LISP
  - Cerebra Engine, commercial, C++
  - FaCT++, free, open-source, C++
  - KAON2 free (for non-commercial usage), Java
  - MSPASS free, open-source, C
  - Pellet free, open-source, Java
  - RacerPro commercial, LISP
- DIG is a standard interface to a DL reasoner that predates RDF and today uses XML
- Protégé uses DIG and can thus use any of several DL reasoners that have a DIG interface

Dig API: http://dig.sourceforge.net/