Description Logics

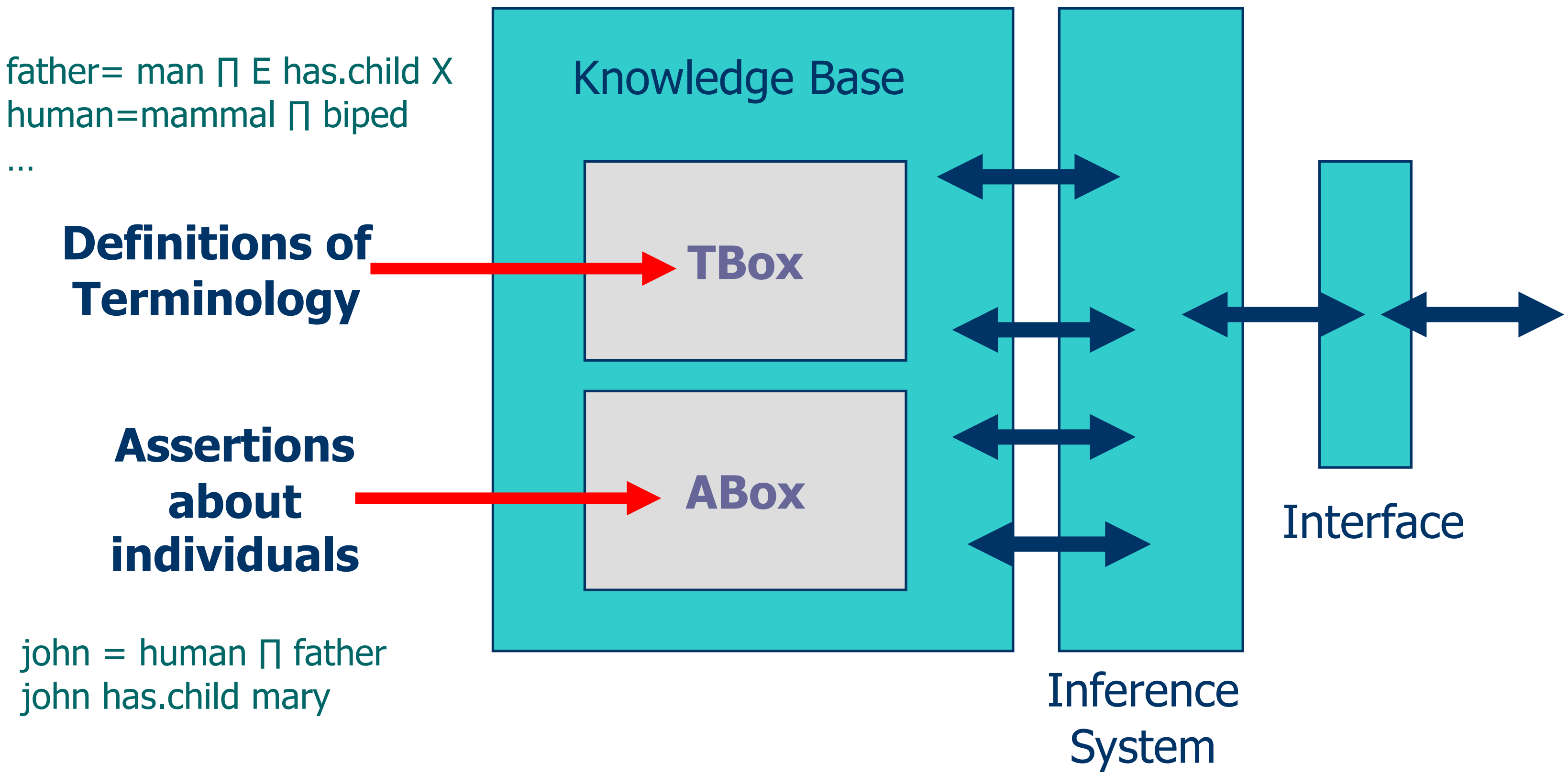
What Are Description Logics?

- 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) based on a decidable fragments of FOL
  - Provision of inference services
    - Sound and complete decision procedures for key problems
    - Implemented systems (highly optimized)
- Formal basis for OWL (DL profile)

DL Paradigm

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

Typical Architecture



Division into TBox and ABox has no logical significance, but is made for conceptual and implementation convenience

DL defines a family of languages

- The expressiveness of a description logic is determined by the **operators** that it uses
  - Adding or removing operators (e.g., $\neg$, $\cup$) increases or decreases the kinds of statements expressible
  - Higher expressiveness usually means higher reasoning 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 restrict.	**∃ R.C**	**Human ∃ hasChild.Male**
Top (univ. conc.)	T	T
bottom (null conc)	⊥	⊥

Examples of ALC concepts

- **Person ∧ ∀hasChild.Male** *(everybody whose children are all male)*
- **Person ∧ ∀hasChild.Male ∧∃hasChild.T** *(everybody who has a child and whose children are all male)*
- **Living_being ∧ ¬Human_*being*** *(all living beings that are not human beings)*
- **Student ∧ ¬∃interestedIn.Mathematics** *(all students not interested in mathematics)*
- **Student ∧ ∀drinks.tea** *(all students who only drink tea)*
- **∃hasChild.Male V ∀hasChild.⊥** *(everybody who has a son or no child)*

Other Constructors

The general DL model has additional 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}

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$^{(D)}$**

Note: R->H and Q->N

Complexity of reasoning in Description Logics

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

Base description logic: $\mathcal{A}$ttributive $\mathcal{L}$anguage with $\mathcal{C}$omplements

$\mathcal{ALC} ::= \bot \mid \top \mid A \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \exists R.C \mid \forall R.C$

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^-
- $\sqcap$ – role intersection[3]: $R \sqcap S$
- $\sqcup$ – role union: $R \sqcup S$
- $\neg$ – role complement: $\neg R$ full
- $\circ$ – 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: $\mathrm{Tr}(R)$
- $\mathcal{H}$ – 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 $\mathcal{Q}$, $\mathcal{I}$ and $\mathcal{O}$ 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, a male, and has a child Mary

Subsumption: $D \subseteq C$?

- Concept C subsumes D iff on every interpretation *I*

  $I(D) \subseteq I(C)$
- This means the same as $\forall(x)(D(x) \rightarrow C(x))$ for complex statements D & C
- 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

- **Concept satisfiability** is C (necessarily) 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 defined in their logic
- One important inference is **classification** (computation of subsumption)

Necessary vs. Sufficient

- *Necessary* properties of an object are common to all objects of that type
  - Being a *man* is a **necessary** condition for being a *father*
- *Sufficient* properties 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** typically specify **both** *necessary and sufficient* properties

Subsumption

- Meaning of Subsumption

  A more general concept or description ***subsumes*** *a more specific one. Members of a subsumed concept are necessarily members of a subsuming concept*

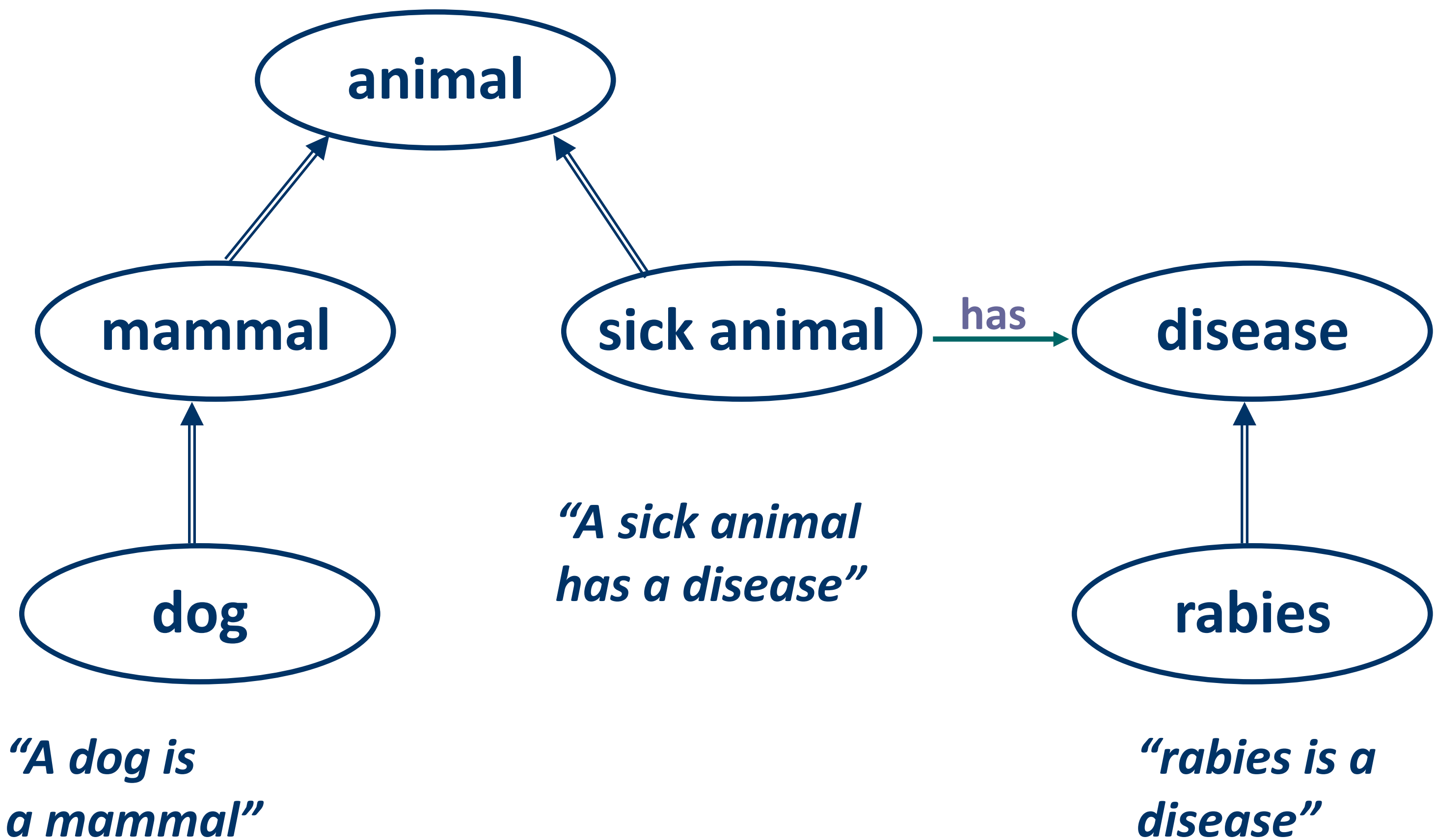
- Example: *Animal* subsumes *Person;* (aka IS-A, rdfs:subClassOf)

- Two ways to formalize 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: instances of subsumed concept are necessarily a subset of subsuming concept's instances

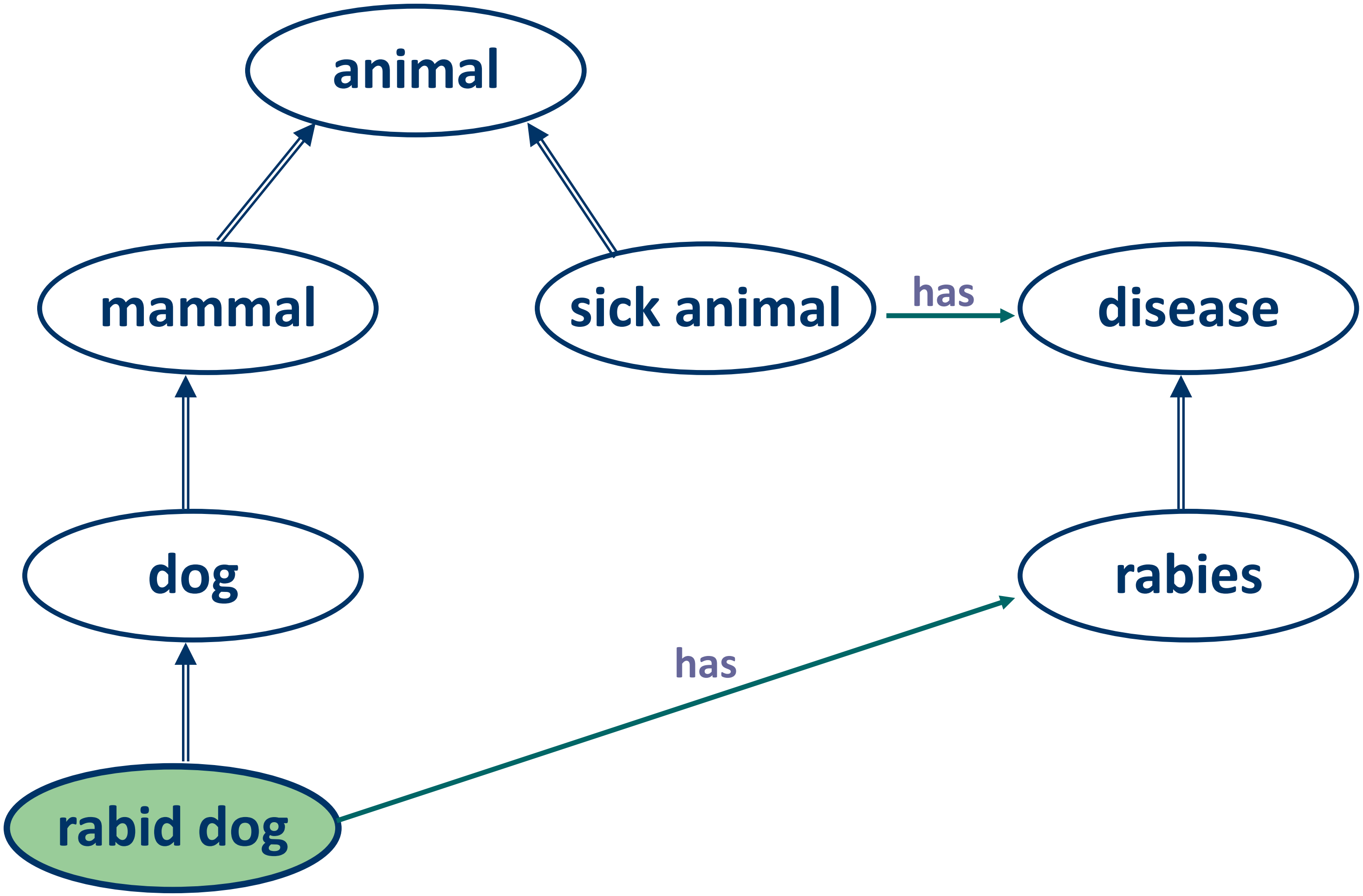
    E.g., the set of all persons is a subset of all animals

How Does Classification Work?



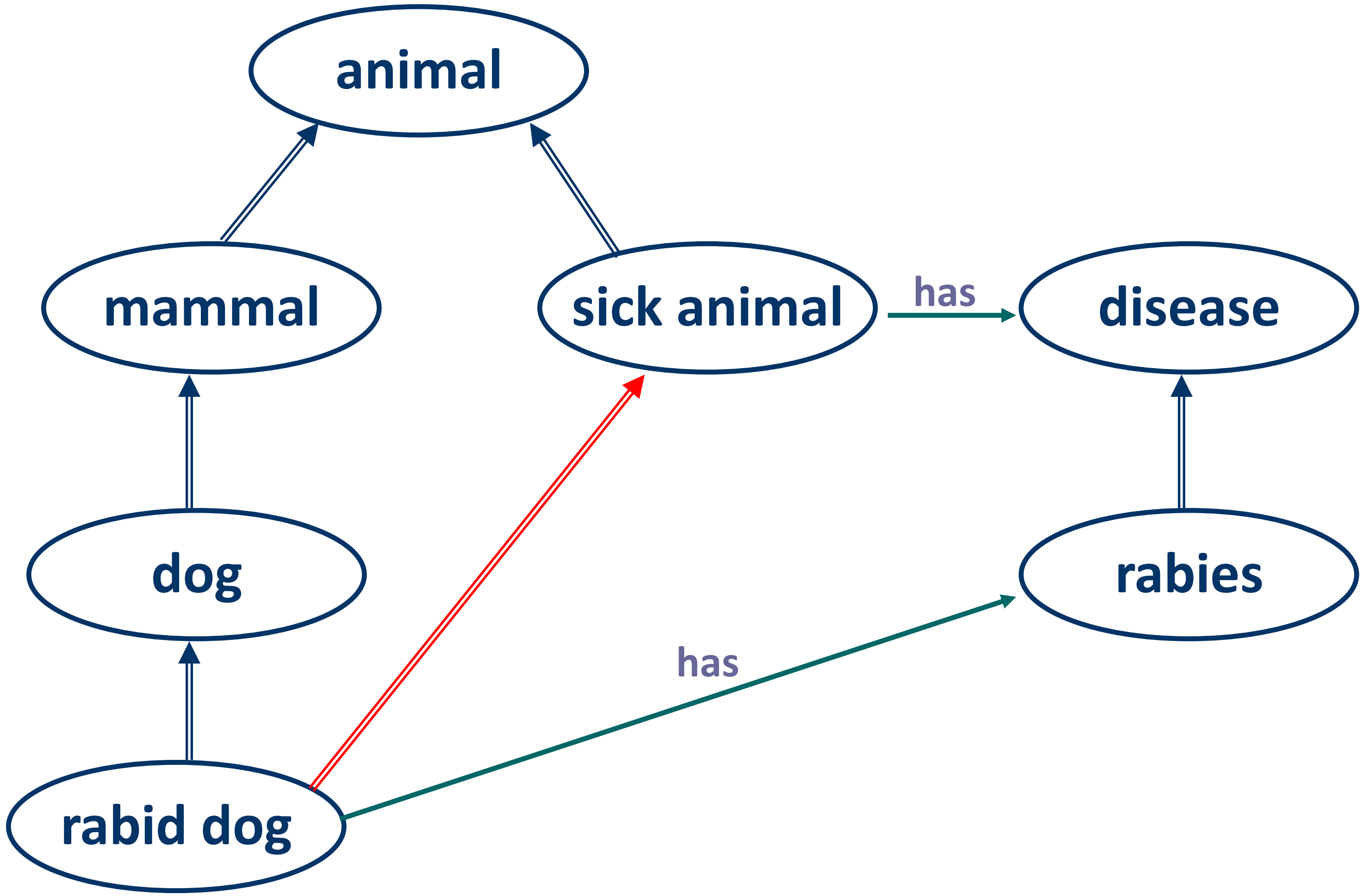
A sick animal is **defined** as something that is both an animal and has at least one thing that is a kind of a disease

Defining a "rabid dog"



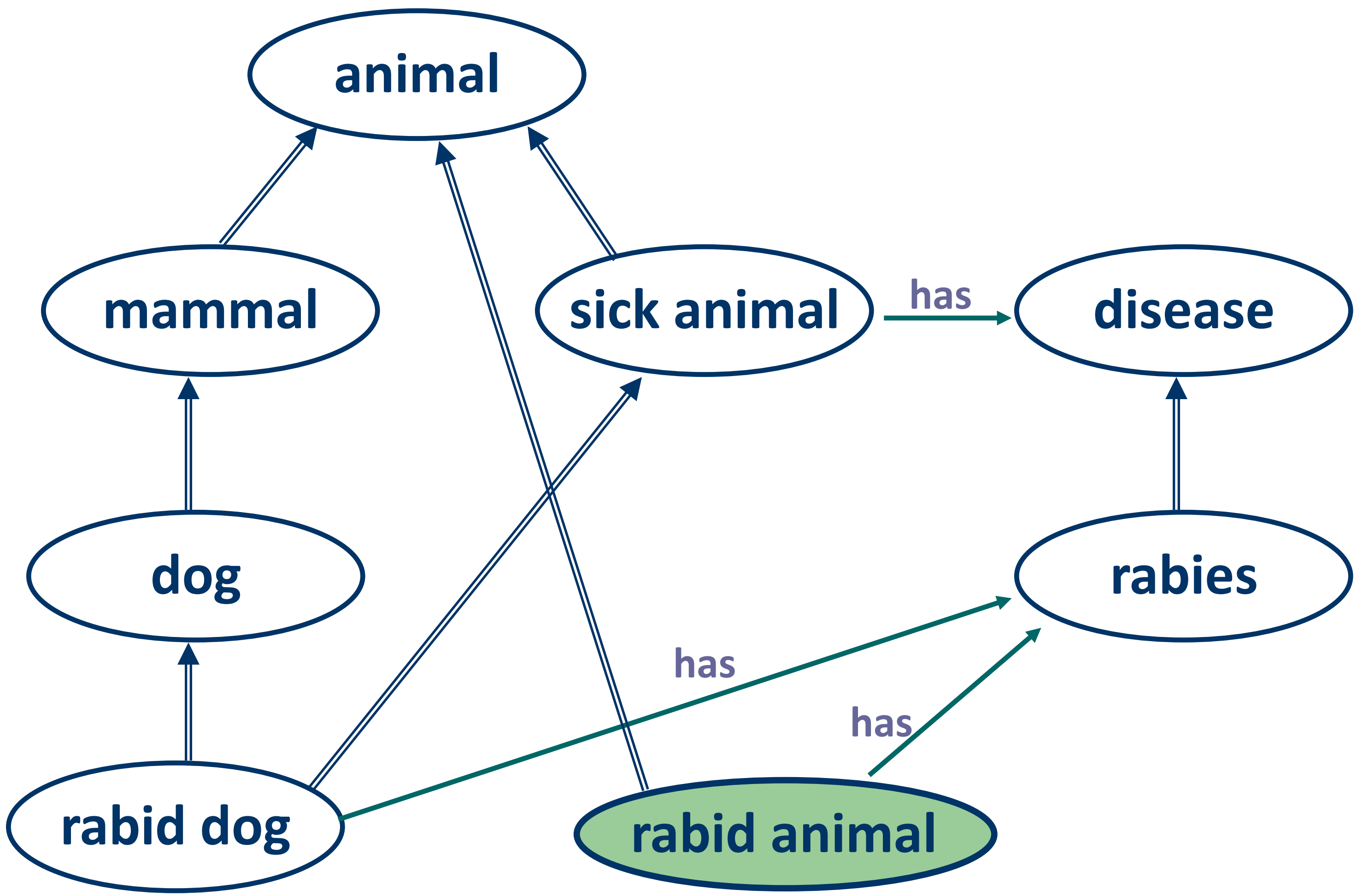
A rabid dog is **defined** as something that is both a dog and has at least one thing that is a kind of a rabies

Classification as a “sick animal”



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

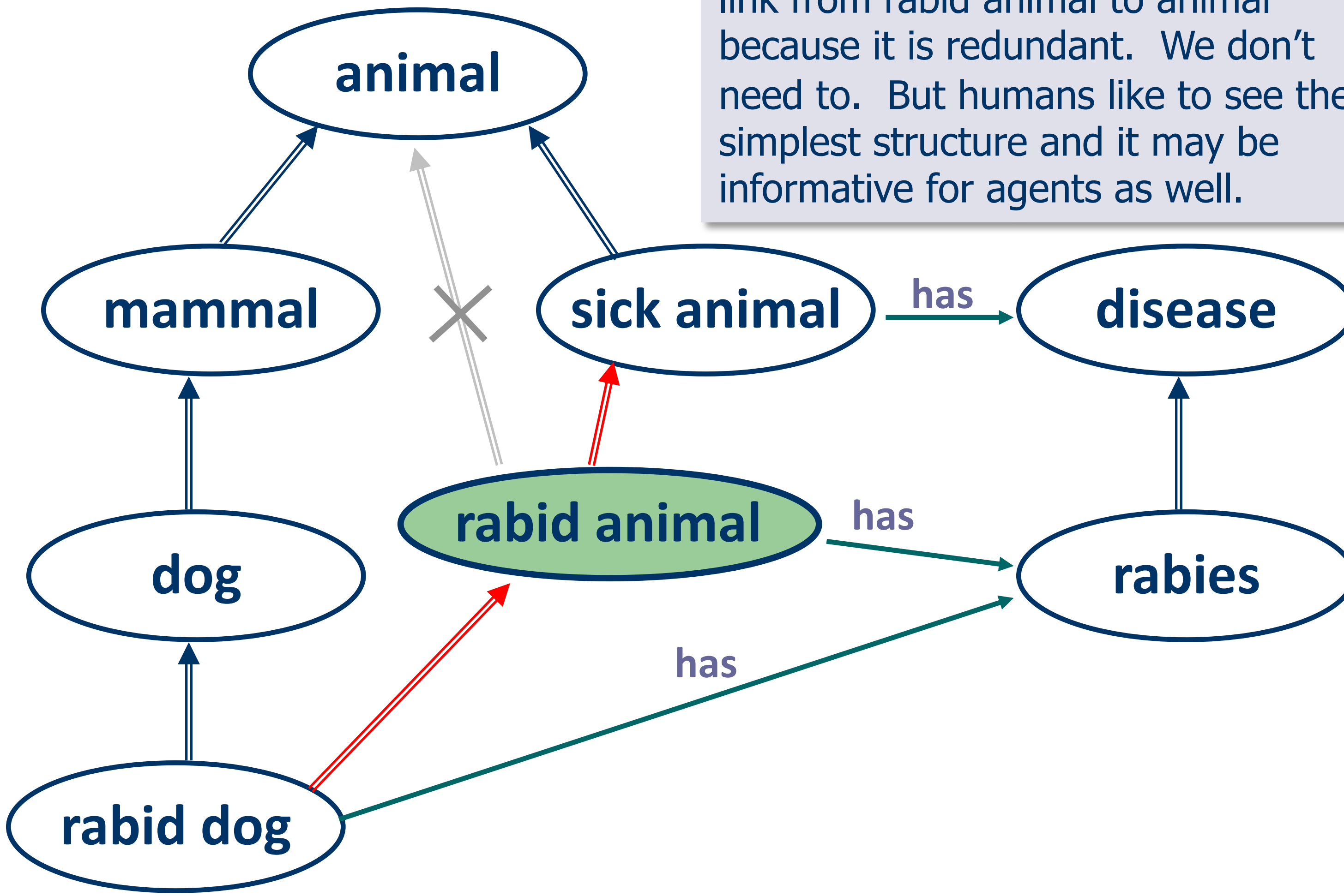
Defining “rabid animal”



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

DL reasoners places concepts 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

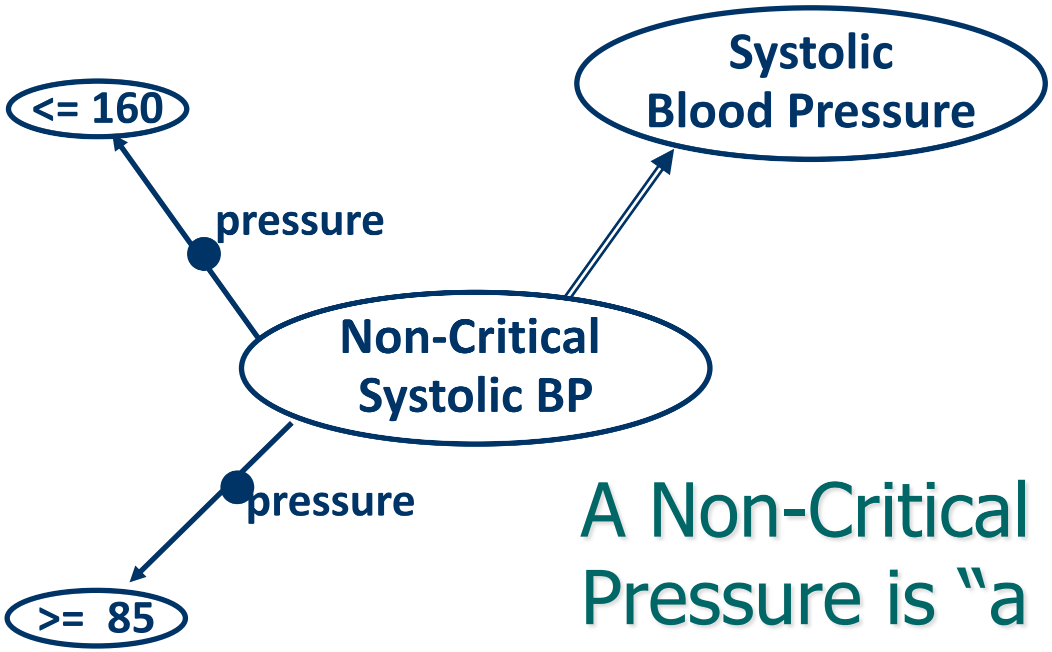
Primitive versus Structured (Defined)

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

Classification is very useful

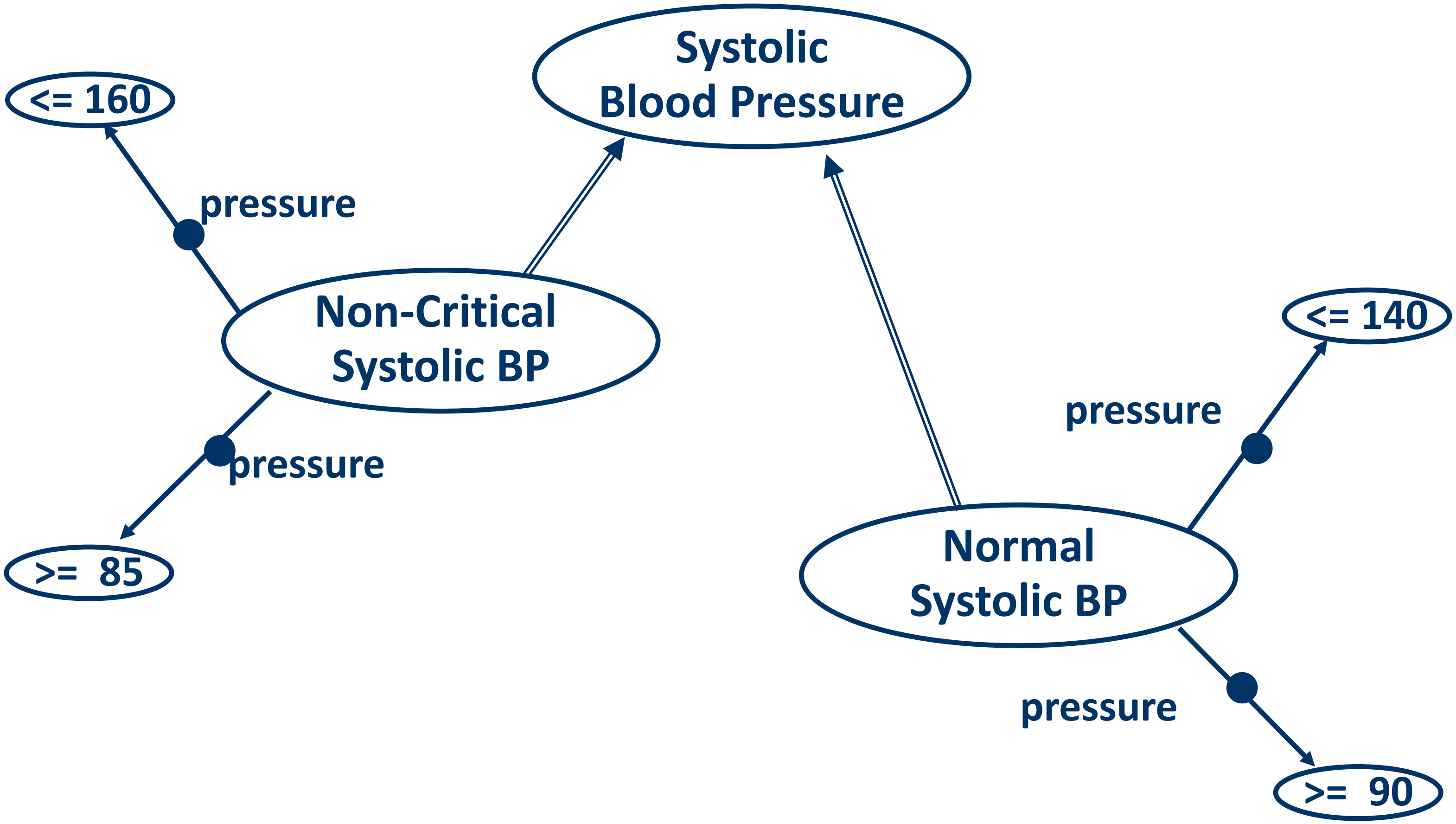
- Classification is a powerful kind of reasoning that is very useful
- Many expert systems can be usefully thought of as doing “heuristic classification”
- Logical classification over structured descriptions and individuals is also quite useful
- But… can classification ever deduce something about an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

Example: Blood Pressure



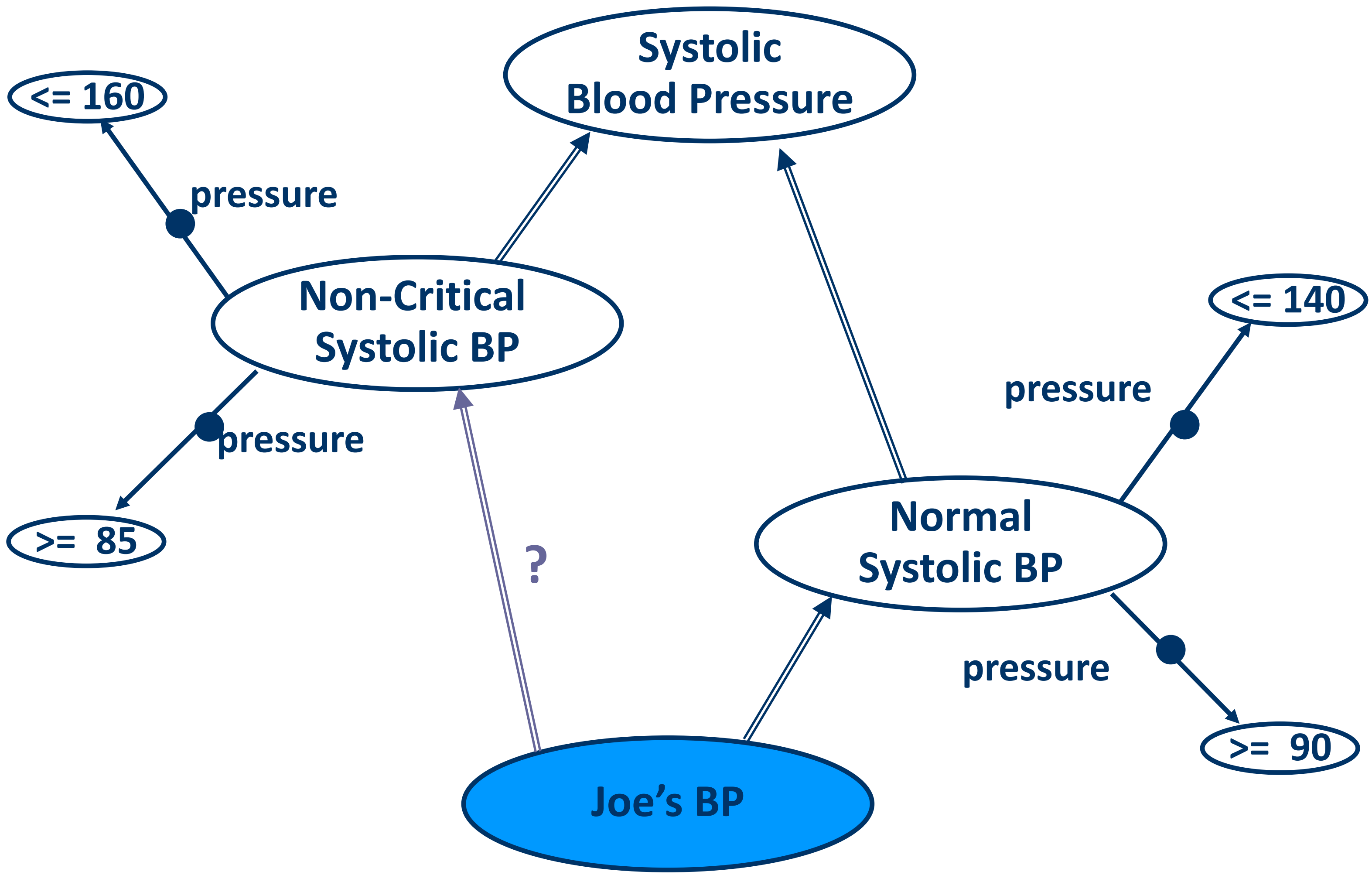
A Non-Critical Blood Pressure is "a Systolic B.P. between 85 and 160."

Example: Blood Pressure

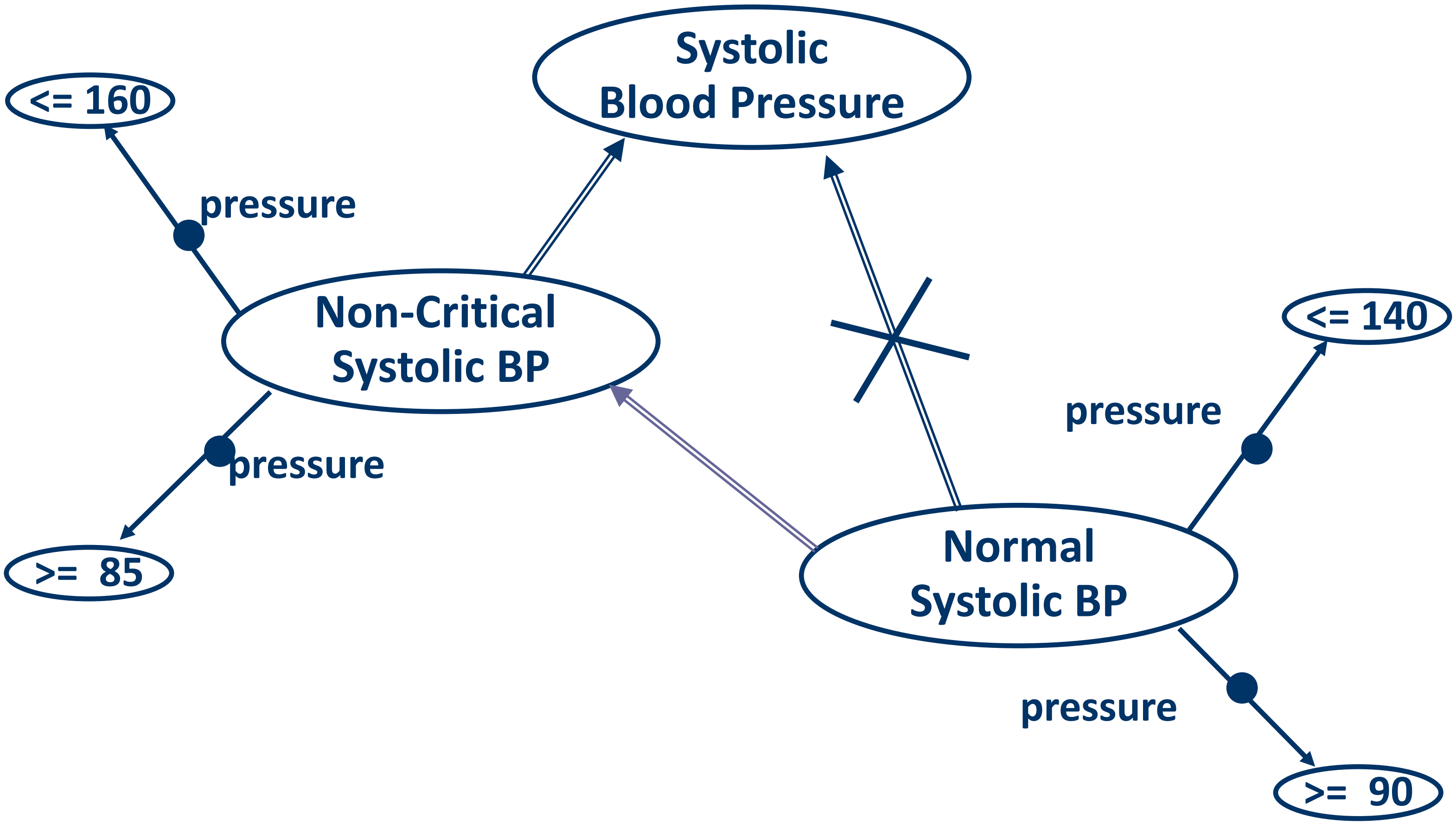


Normal Systolic B.P. is “a Systolic B.P. between 90 and 140.

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