Inheritance: An overview

L. Morgenstern
Knowledge Representation
CS 4725
Columbia, Spring 1999

Outline:

- Semantic Networks

- Description Logics
  - Definitions vs. Assertions
  - Subsumption
  - Classification

- Inheritance with Exceptions
  - Multiple Paths
  - Conflicted and Preempted Paths
  - Credulous vs. Skeptical Reasoning

What are semantic networks?

Semantic networks can be viewed broadly or narrowly

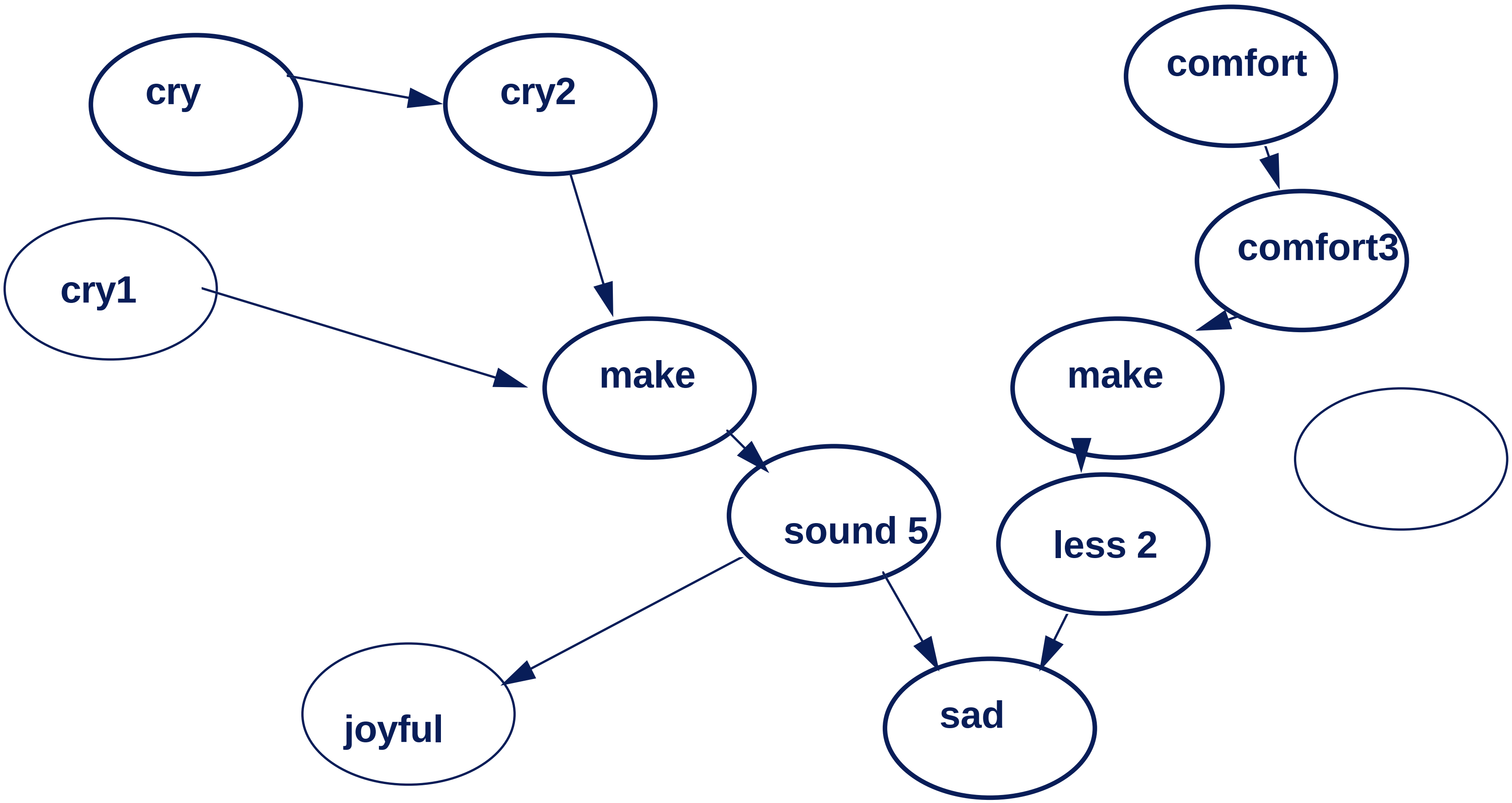
- Broadly, a semantic network is any graph where
  --- the nodes represent concepts and
  --- the links represent relations between the concepts

- Narrowly, a semantic network is a graph where
  --- the nodes represent some restricted set of concepts
  --- the links represent some very restricted set of relations
  --- a clear semantics is given to nodes and, especially, links

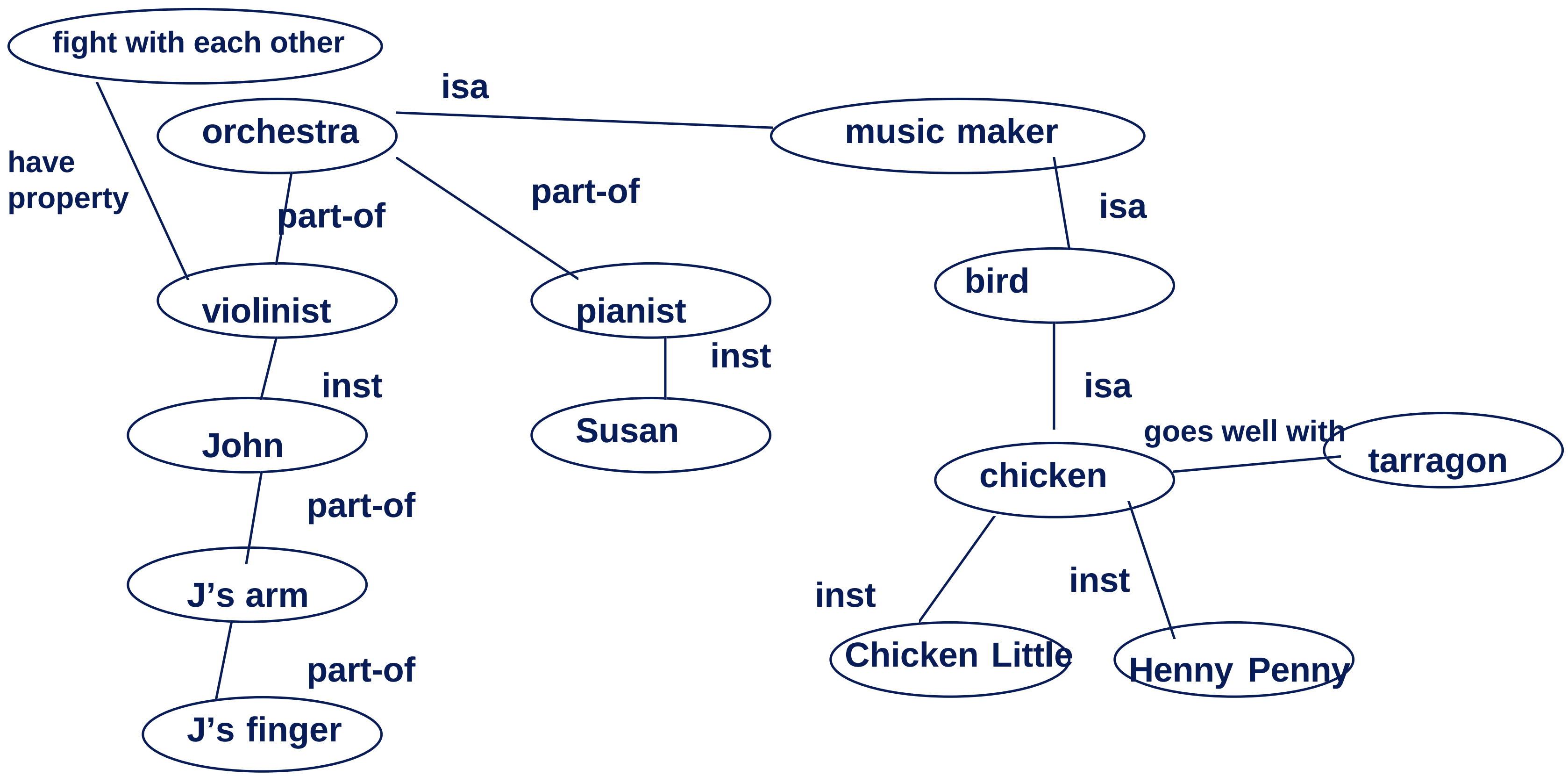
Examples of narrowly defined semantic networks are
--- terminological or description logics
--- inheritance networks (with exceptions)

In the early days of semantic networks (60s, 70s), semantic networks were broadly defined. Anything was okay.

example: Quillian: natural language understanding



Semantic networks have been used for natural language understanding, commonsense knowledge representation



What are we saying here? Is John's finger a part of the orchestra? Does Henny Penny go well with tarragon? Does John fight with himself? Are roast chicken music makers? What's going on? Problem: no semantics for concepts or links

Since the late 1970s, research in semantic networks has focussed on giving a clear semantics for the network (nodes and links) and providing sound and complete inference mechanisms. Networks have thus become much simpler.

Description Logics

- --- equivalent to a subset of first-order logic
- --- has some of the properties of frames: roles, slots, fillers
- --- subsumption and classification algorithms (polynomial)

Inheritance Networks with Exceptions

- --- allows subset, membership, cancels links
- --- polynomial algorithms to determine if there is a path between 2 nodes

Old, unsound networks continue to be used (in medicine, MeSH & UMLS), but there are efforts to be these on a sounder footing

Description Logics

(KL-ONE, Classic, Loom: Brachman, Levesque, Patel-Schneider, etc.)

**Aim: provide a language for querying, organization,
that is considerably more expressive than a database language
but is much more efficient than first-order logic
(Note: “pictorial” notion has been discarded)**

<concept>::=   /* atoms are primitive concepts */
<atom> | (AND <concept1> ... <conceptn>) | (ALL <role> <concept>) | (SOME <role>)
<role> ::= <atom> | (RESTR <role> <concept>)

Examples:
Can define California-white-wine as
(AND WHITE-WINE (ALL region CALIFORNIA-REGION))
Can define Relaxed-meal as
(SOME (RESTR course WINE-COURSE))

Concepts are node in network;
all links are subset (is-a) or membership (inst)

The interesting questions here are:

- Given two concepts, does one subsume another? (subsumption)
  (note that A subsumes B if B is a subset of A)
- Given a concept and a graph, where does the concept fit?
  (classification)

The subsumption algorithm: (Brachman & Levesque, AAAI 1984)

To check if a subsumes b:

1. Flatten both a and b by removing all nested AND operators
   So, (AND x (AND y z) w) becomes (AND x y z w)
2. Collect all arguments to an ALL for a given role.
   So, (AND (ALL R (AND a b c)) (ALL r (AND X))) becomes
   (AND (ALL r (AND a b c X)))
3. Assuming a is now of the form (AND a1 ... an)
   b is now of the form (AND b1 ... bm),
   return true iff
   (i) if ai is an atom or a SOME, then one of the bj is ai
   (ii) if ai is (ALL r x), then one of the bj is (ALL r y) where x subsumes y.

The subsumption algorithm: (Brachman & Levesque, AAAI 1984)

To check if a subsumes b:

1. Flatten both a and b by removing all nested AND operators
   So, (AND x (AND y z) w) becomes (AND x y z w)
2. Collect all arguments to an ALL for a given role.
   So, (AND (ALL R (AND a b c)) (ALL r (AND X))) becomes
   (AND (ALL r (AND a b c X)))
3. Assuming a is now of the form (AND a1 ... an)
   b is now of the form (AND b1 ... bm),
   return true iff
   (i) if ai is an atom or a SOME, then one of the bj is ai
   (ii) if ai is (ALL r x), then one of the bj is (ALL r y) where x subsumes y.

Example:

(AND person (ALL child doctor))
  subsumes
(AND
  (AND person (ALL child rich))
  (AND male (ALL (RESTR child rich)
        (AND doctor
        (SOME (RESTR specialty surgery))))))

The class of people whose children are all doctors
subsumes
the class of people all of whose children are rich
and who are all males each of whose rich children
is a doctor who is a surgeon.

Complexity of subsumption is $O(n^2)$, where n is length of expression.

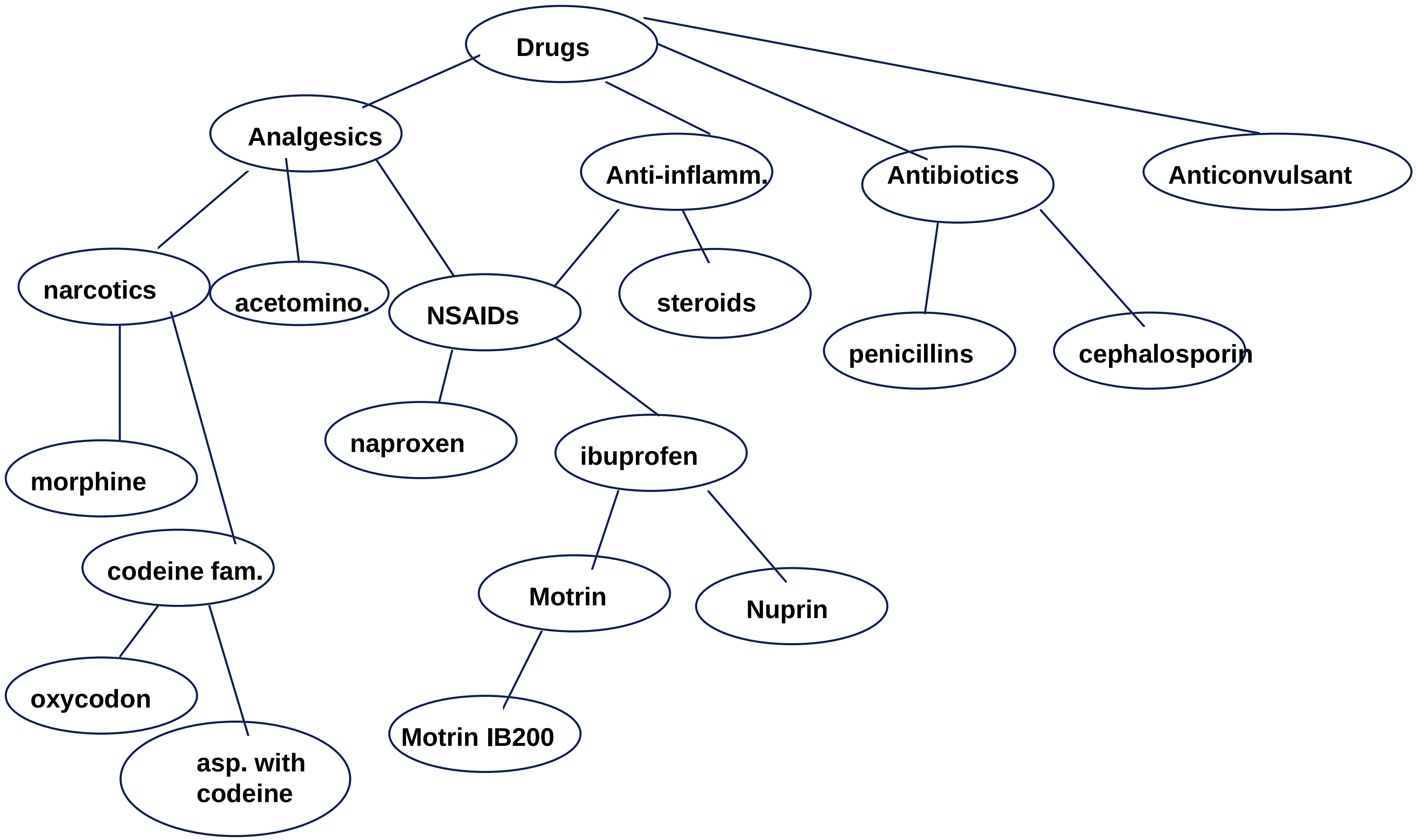
Classification is done using subsumption.
Take a concept, find a place in the tree such where it is subsumed by a node, but can't be subsumed by the node's children.
Note that one must take into account that several nodes may subsume a particular node.

Worst case would have to examine each node in tree.

Complexity of classification: $m(O(n^2))$, where m is no. of nodes in tree

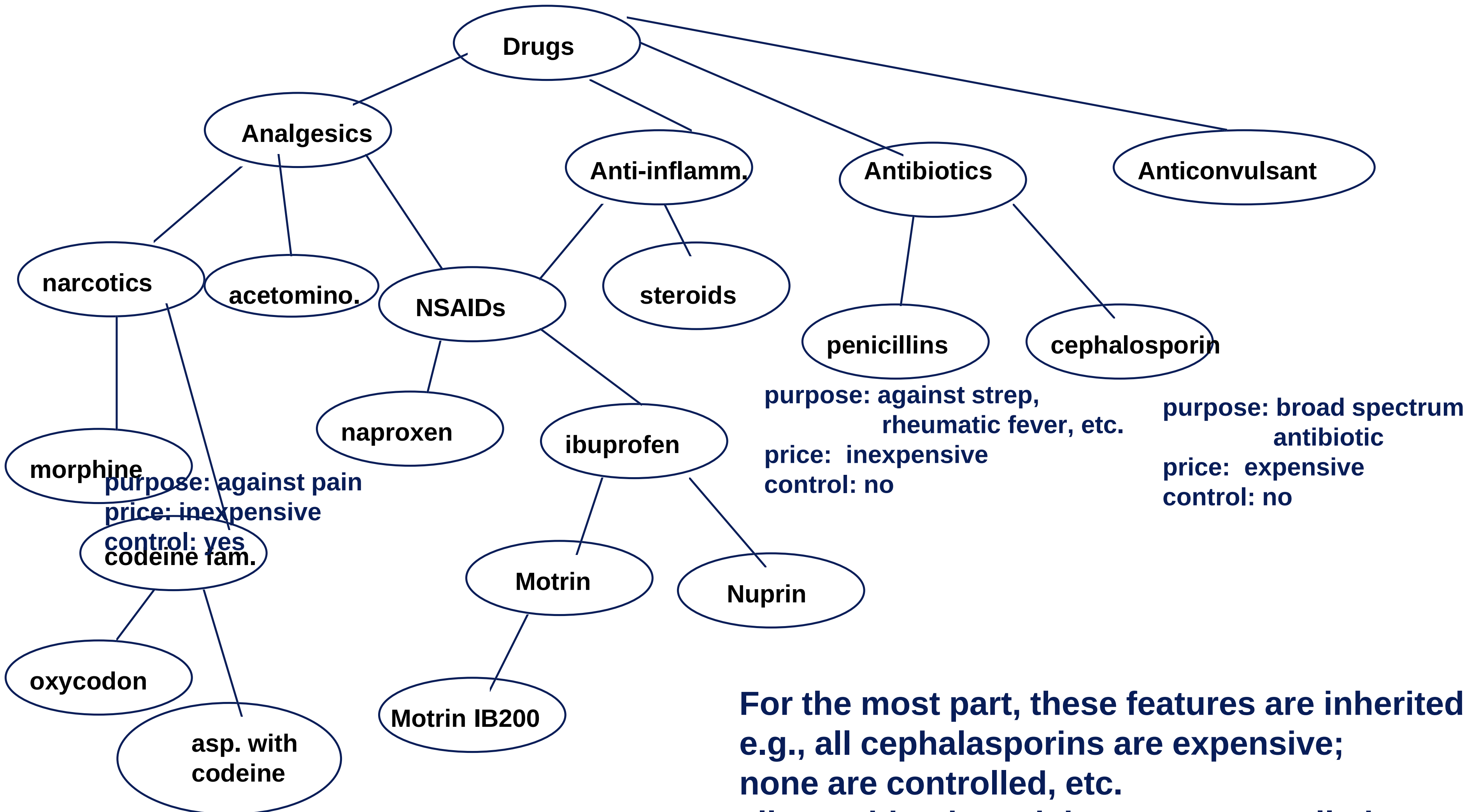
Note: in practice, power of description logics used for standard inheritance logic + slots (roles) and fillers

This is an example of a standard inheritance network:



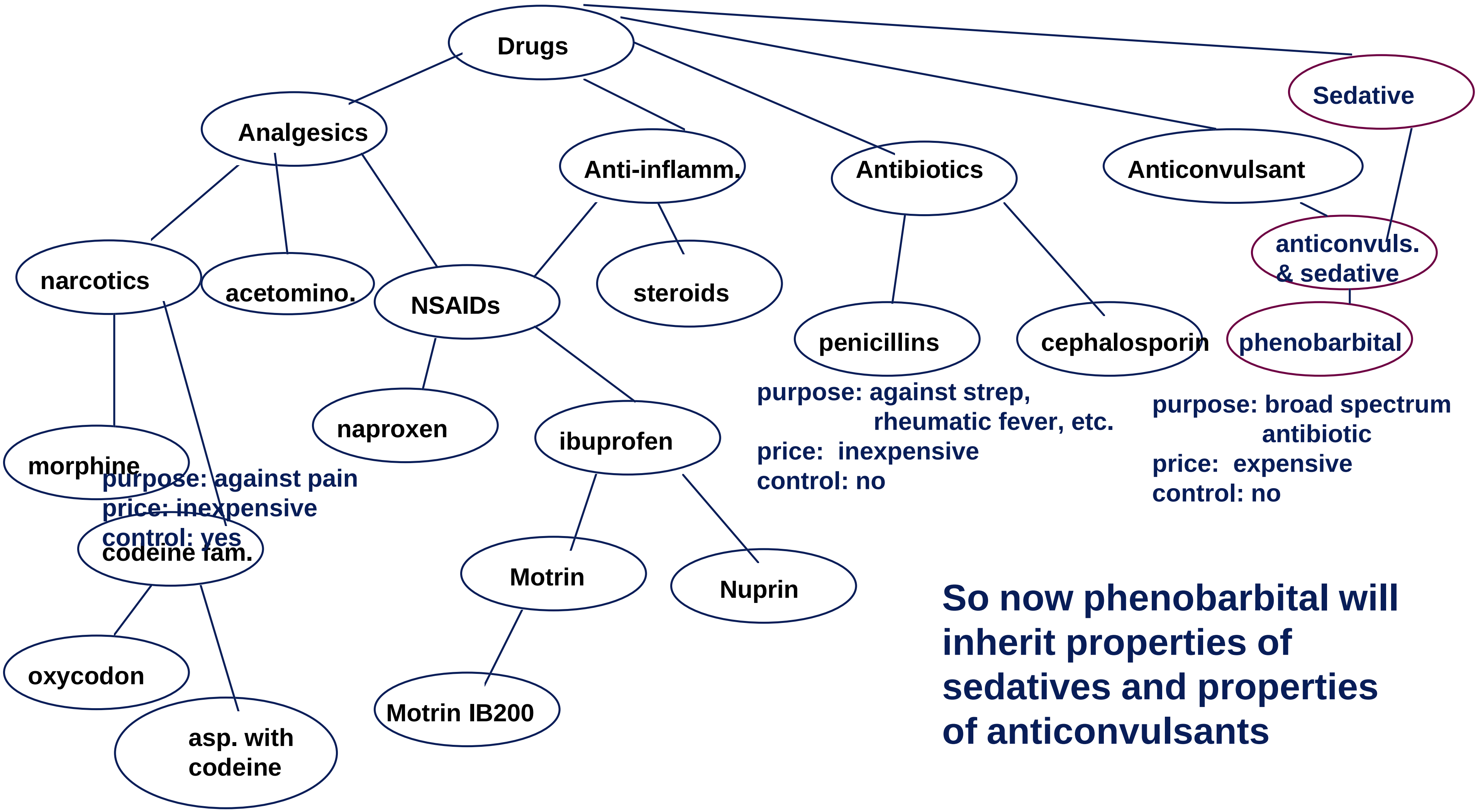
When we add slots and fillers, we can say more interesting things. For example, in the drug formulary network, we may be interested in representing facts about price, purpose, control information, etc.

Saying these facts is enabled by the syntax of description logics, which allow: and (all r1 concept1) .... (all rn conceptn)



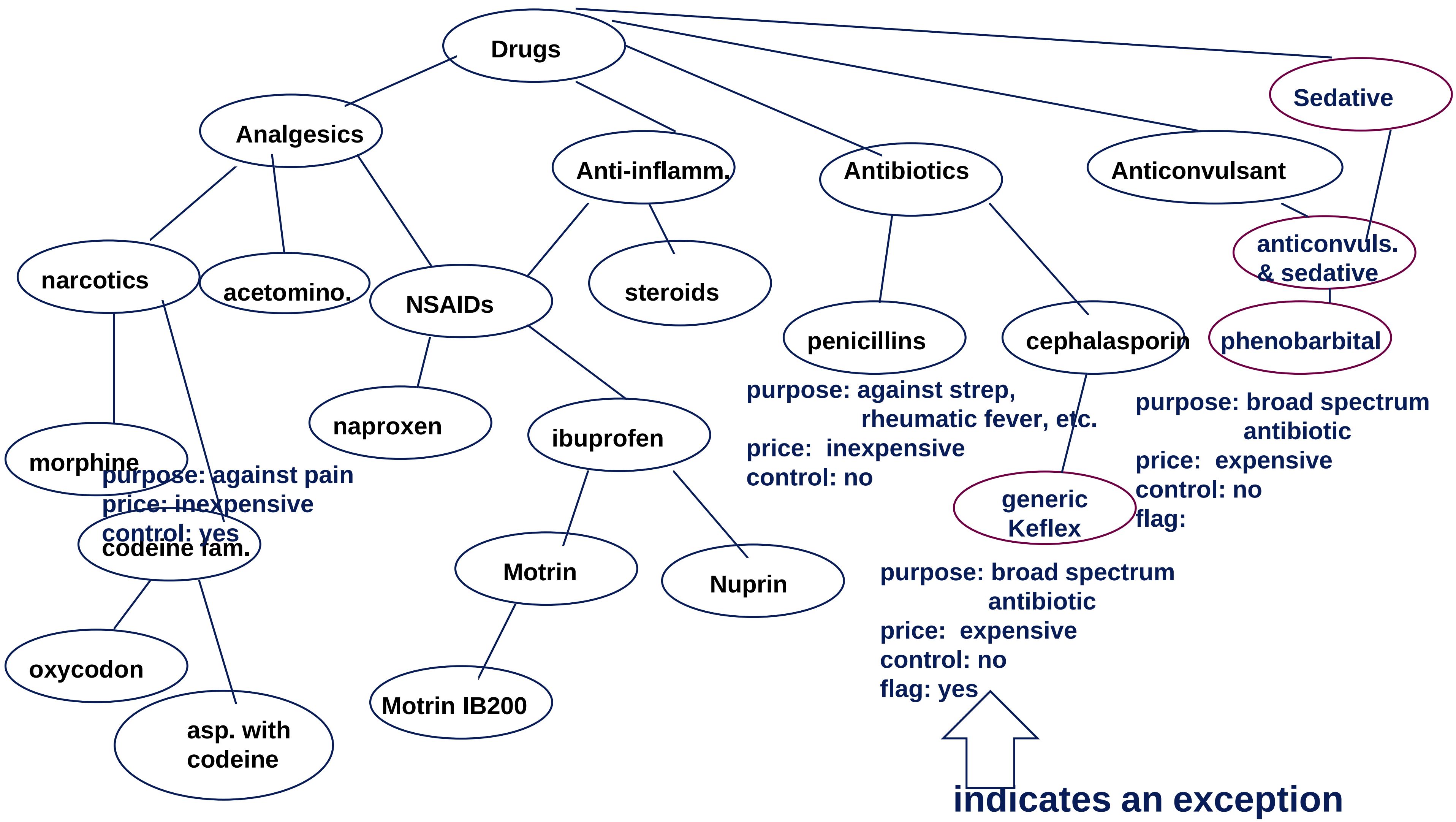
For the most part, these features are inherited; e.g., all cephalasporins are expensive; none are controlled, etc. all morphine-based drugs are controlled

Description logics also allow the creation of one concept using conjunction. Thus, we may consider the class of all drugs that are both anticonvulsants and sedatives

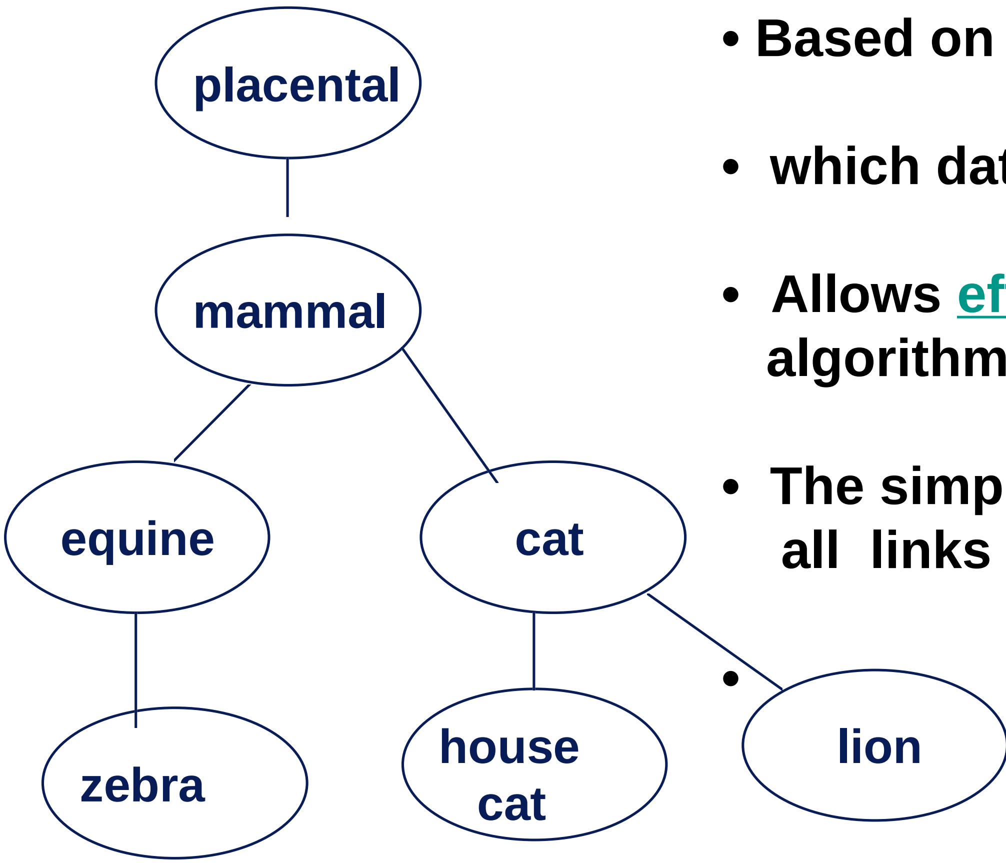


So now phenobarbital will inherit properties of sedatives and properties of anticonvulsants

What description logics won't let you do: exceptions
So there's no way to say that cephalosporins are typically expensive but one particular type is moderately priced.
Well, almost no way. You can "cheat" by adding another role -- a flag -- and turning it on if there's an exception ....

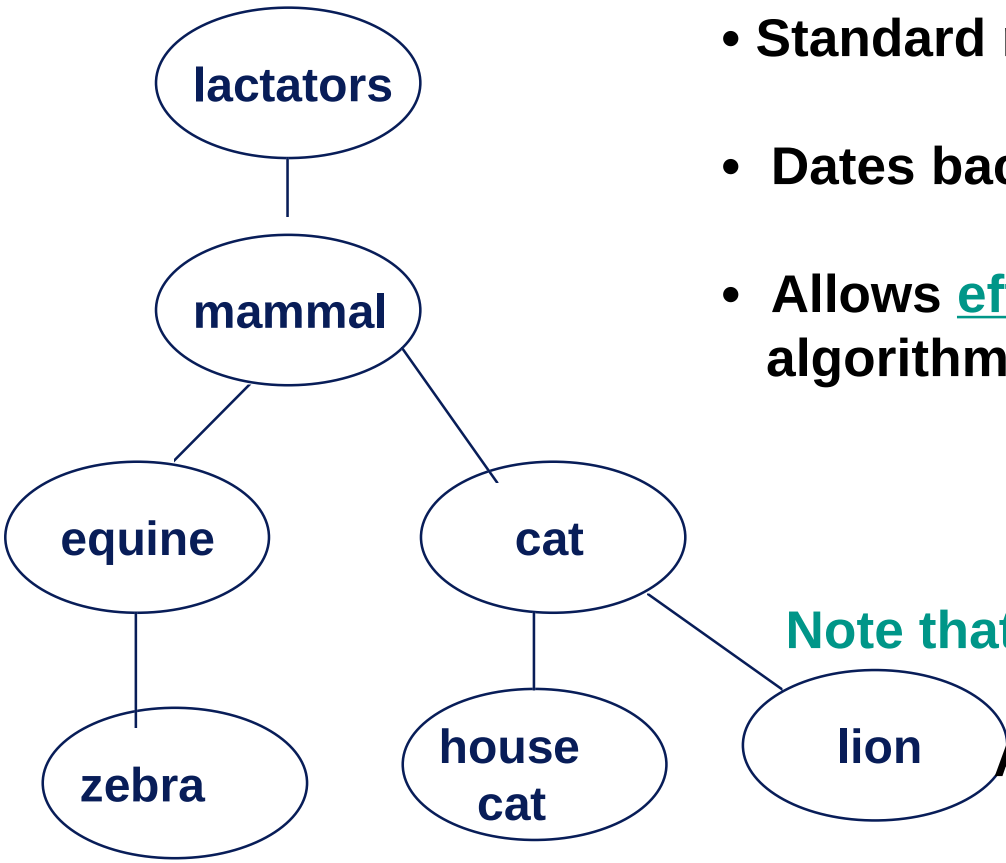


Inheritance with exceptions



- Based on standard monotonic inheritance
- which dates back to Aristotle
- Allows efficient, graph-based algorithms for reasoning
- The simplest form of semantic network; all links are is-a links = subset
- i.e.. lions are a subset of cats

Brief Review of Inheritance



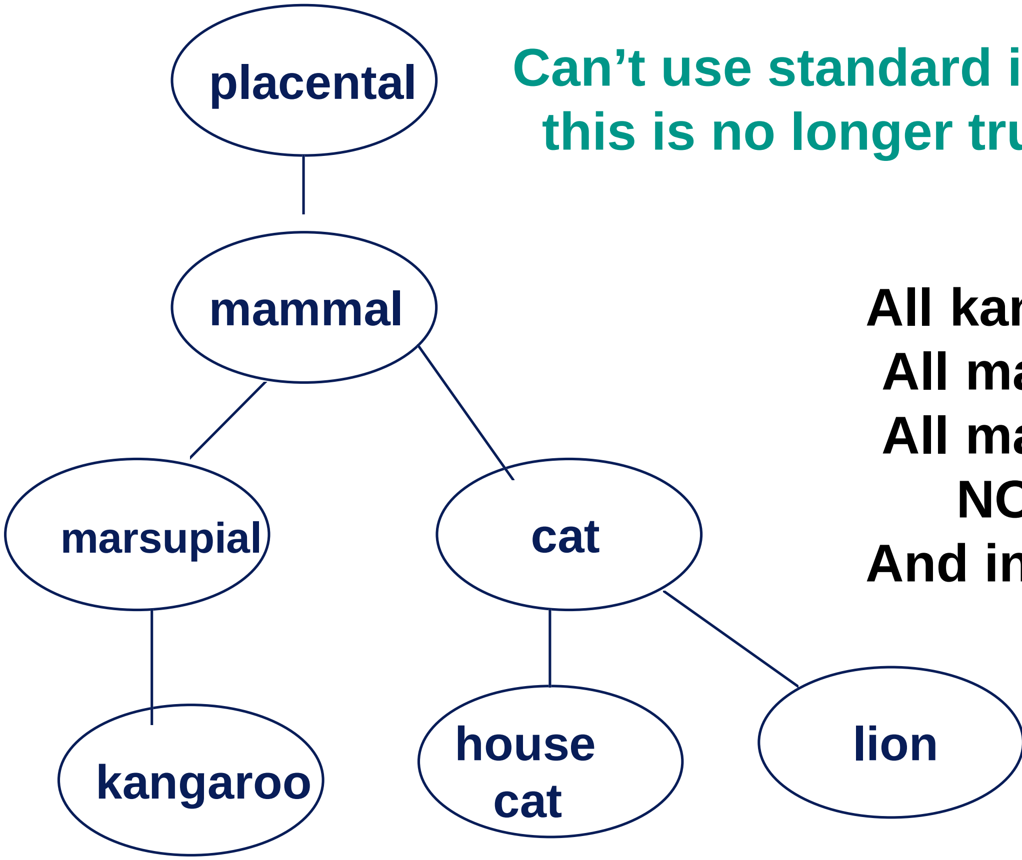
- Standard monotonic inheritance
- Dates back to Aristotle, Porphyry
- Allows efficient, graph-based algorithms for reasoning

Note that this translates into logical rules:

All zebras are equines
All lions are cats
All equines are mammals
All mammals are lactators ....

But what happens when we need to talk about exceptions?

what happens when we need to talk about exceptions?

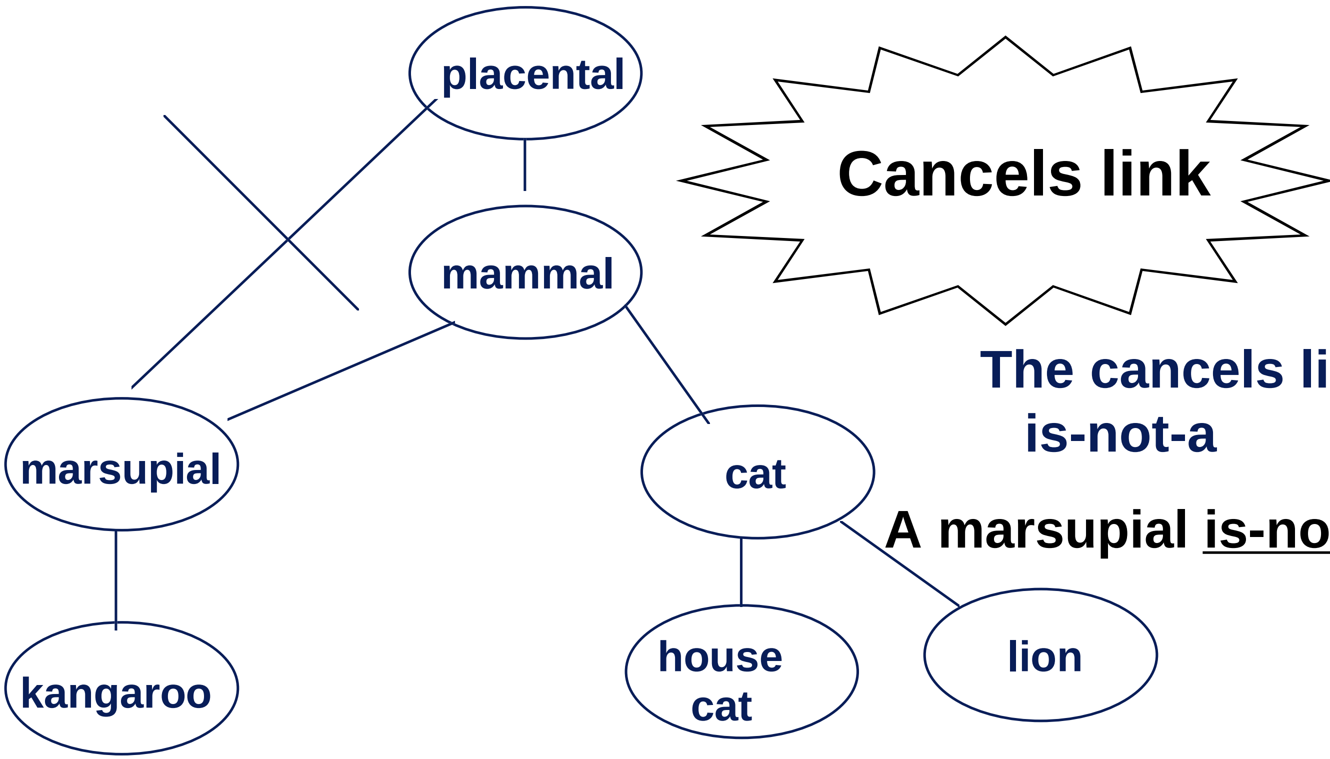


Can’t use standard inheritance network:
this is no longer true!

All kangaroos are marsupials
All marsupials are mammals
All mammals are placental?
NO!!!
And in particular, all marsupials are not

<u>We need to represent exceptions within the inheritance network</u>

So we Introduce
Inheritance Networks with Exceptions

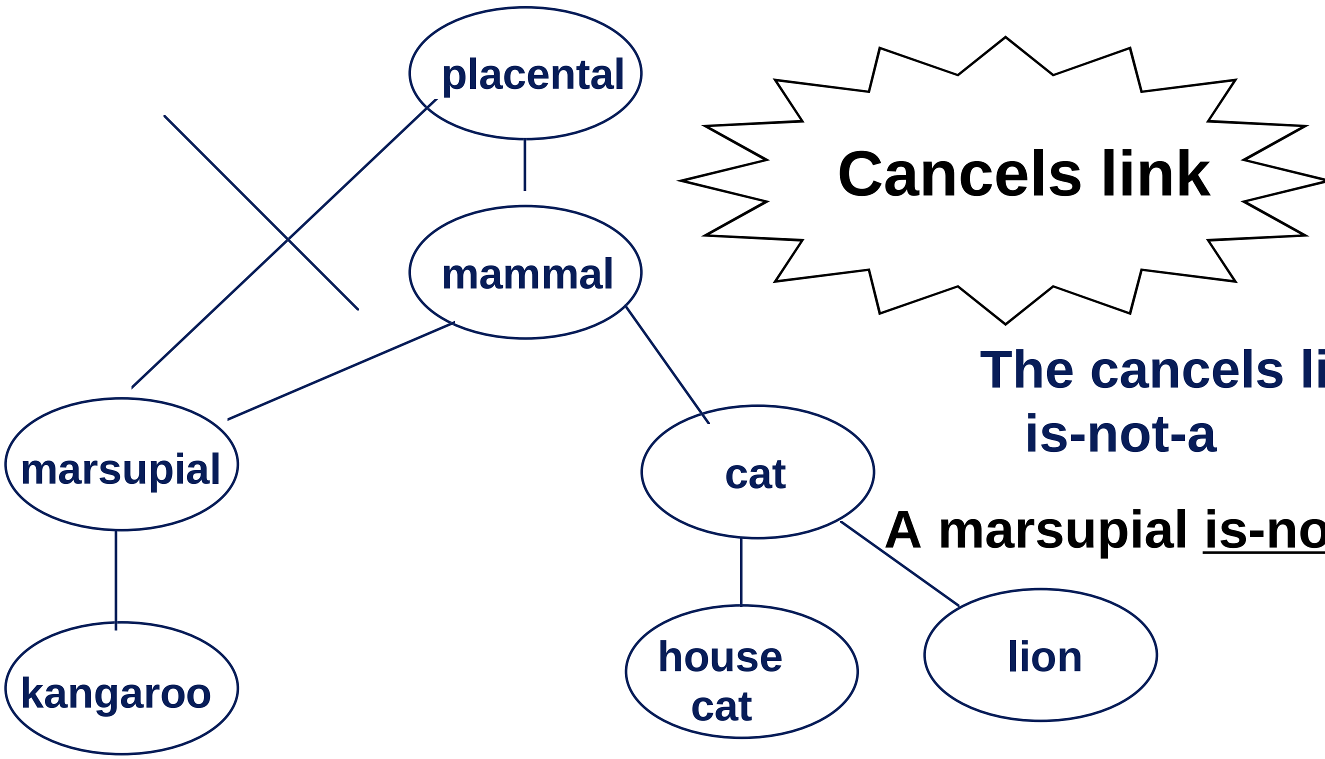


The cancels link reads:
is-not-a

A marsupial <u>is-not-a</u> placental animal

Note that this potentially changes the semantics of <u>all</u> the links.
Obviously it is no longer the case that all mammals are placental!
Rather, the Is-a link is a "close-to-subset" link; most mammals are placental.
These new links are called defeasible.
Some systems mix strict is-a and defeasible is-a;
here we stick to all defeasible links

So we Introduce
<u>Inheritance Networks with Exceptions</u>

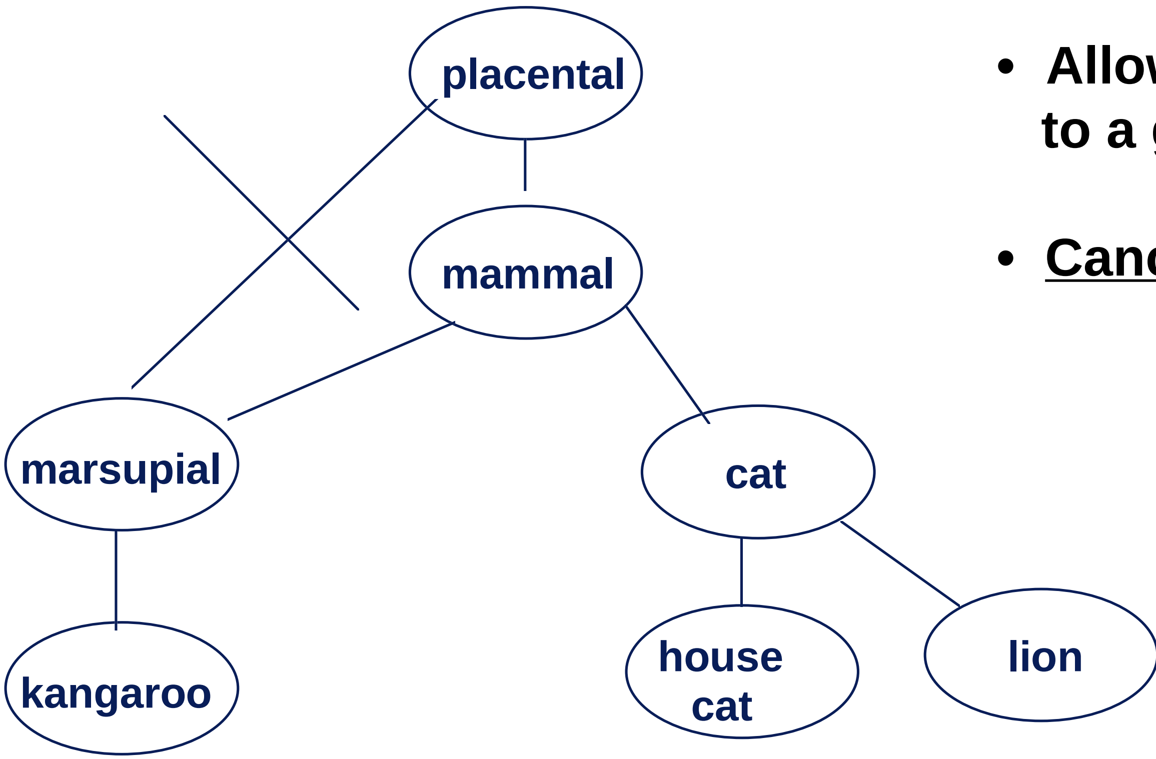


The cancels link reads:
is-not-a

A marsupial <u>is-not-a</u> placental animal

So this is what we are saying here:
All cats are mammals; all kangaroos are marsupials;
~~all~~ marsupials are mammals;
all <u>most</u> mammals have placentas
marsupials do <u>not</u> have placentas

Review of Inheritance



Inheritance with Exceptions

- **Allows us to specify <u>exceptions</u> to a general class**
- **<u>Cancels link:</u> nonmonotonicity**

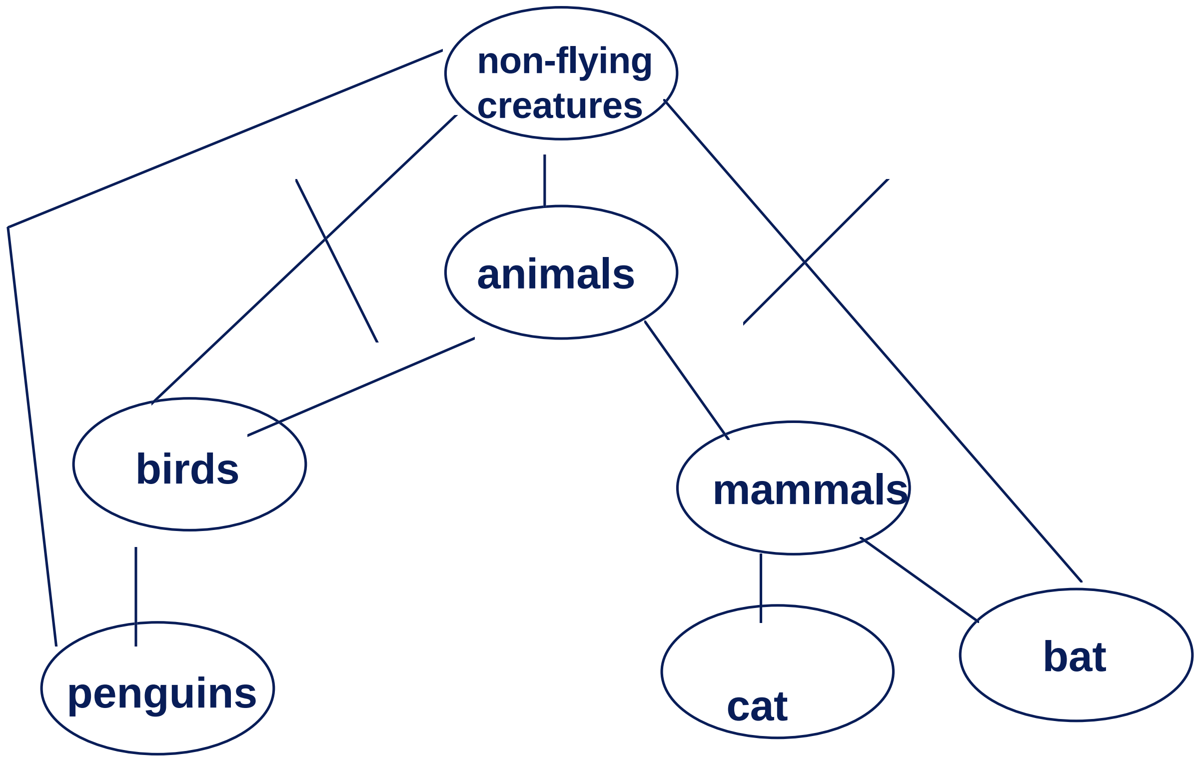
We could write this in a nonmonotonic logic: ***(translational semantics)***

mammal(x) : placental(x)
----
placental(x)

marsupial(x) ==> not placental(x)
kangaroo(x) ==> marsupial(x)

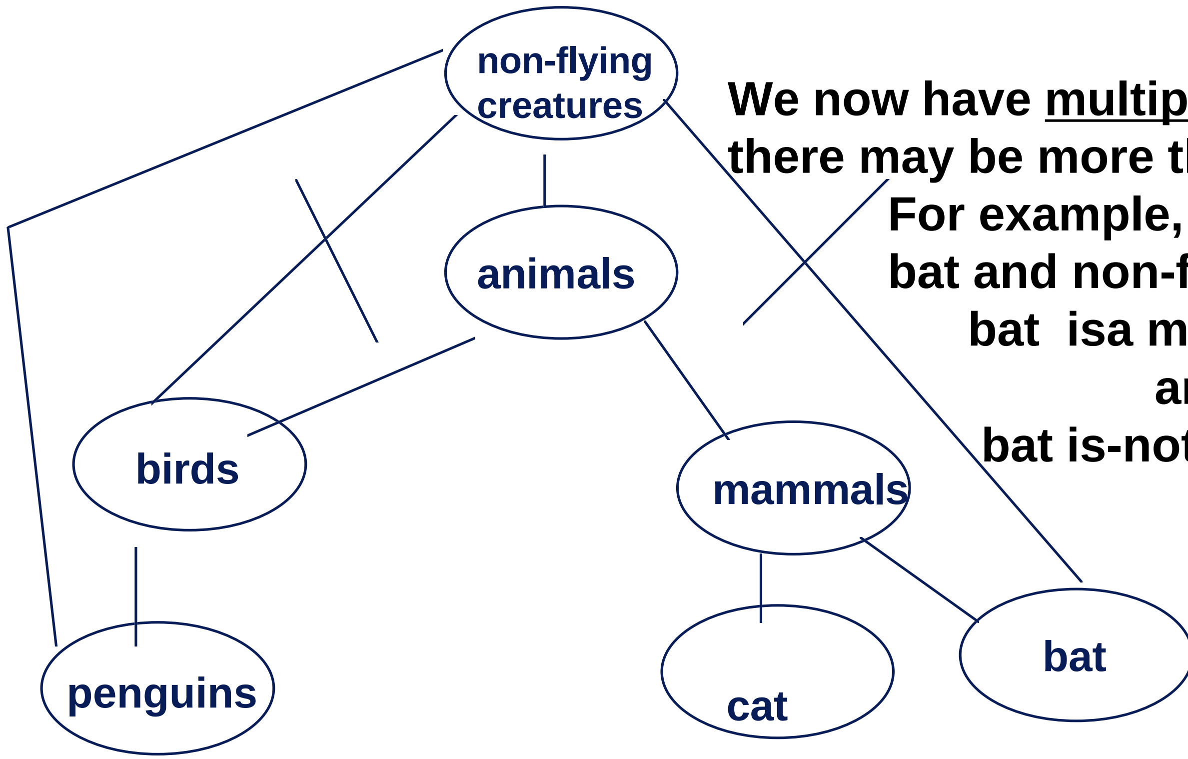
So if Kiri is a kangaroo, we conclude that Kiri is not placental

Note that cancels links also aren't strict;
we can have exceptions to exceptions



What we are saying here:
Most animals don't fly; mammals usually don't fly but bats do fly;
Birds usually do fly, but penguins don't fly.

Dealing with multiple inheritance:



We now have <u>multiple inheritance</u>:
there may be more than one path between 2 nodes

For example, there are 2 paths between
bat and non-flying creature

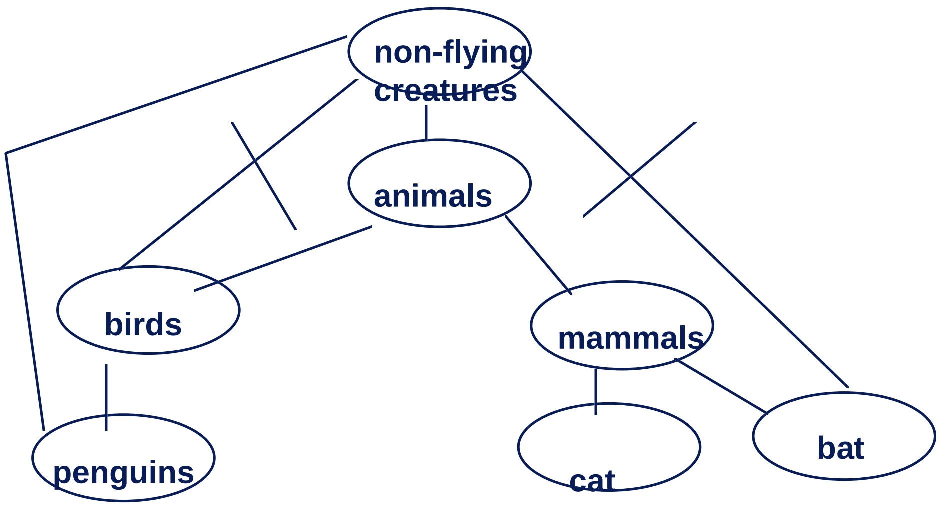
bat isa mammal isa animal isa non-flying
and
bat is-not-a non-flying creature

And there are 3 paths between penguins and non-flying creatures:

- penguin isa bird isa animal isa non-flying creature
- penguin isa bird is-not-a non-flying creature
- penguin isa non-flying creature

Question:
which path should we use?

Dealing with multiple inheritance: choosing the right path

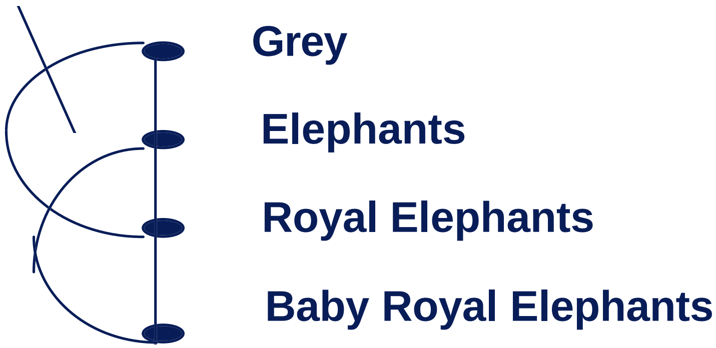


It might seem that all we need to do is choose the shortest path, but it turns out that won't always work, especially if we allow redundant links ....

E.G.., consider:

There is a path of length 2 that says baby royal elephants are grey and a path of length 2 that says baby royal elephants are not grey!

Dealing with multiple inheritance: choosing the right path



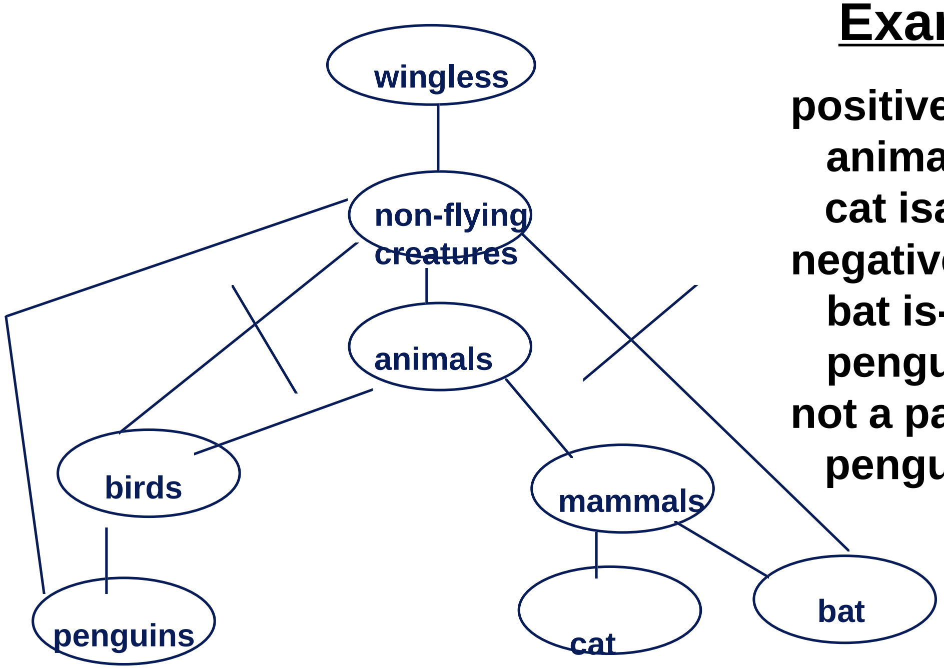
The shortest-path criterion will not work;
instead we use the criterion of <u>specificity</u>

To define specificity, we must give a formal definition of path-based inheritance

(Touretzky, 1986; Horty, Thomason, and Touretzky, 1987, 1990; Stein, 1989, ..., 1992)

Formal definition of path-based inheritance (Horty, 1994; Stein, 1992)

- Links may be positive (<u>is-a</u>) or negative (<u>is-not-a; cancels</u>)
- A path is a restricted sequence of positive and/or negative link
    - positive paths are made up of positive links
    - negative paths are positive paths, with one negative link at end



<u>Examples of paths:</u>

positive paths:
  animal isa non-flying-creature isa wingless c.
  cat isa mammal isa animal isa ...
negative paths:
  bat is-not-a non-flying creature
  penguin is a bird is-a non-flying creature
not a path at all:
  penguin is a bird is-a non-flying cr. isa wingless c.

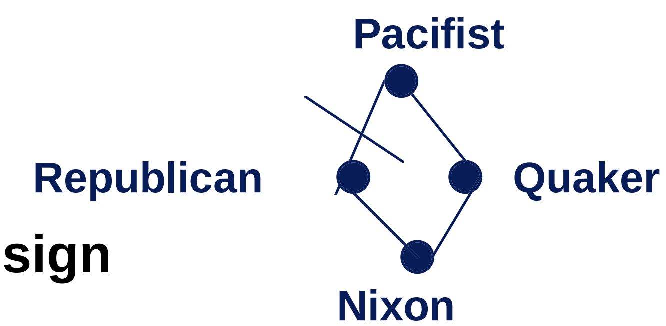
- A context: a network and a set of paths (arising out of the network)

In case of multiple paths, how do we choose the "right" path?

A path is <u>inheritable</u> or <u>undefeated</u> in a context if it is:

- **constructible**

  i.e. can recursively be built out of paths in the network



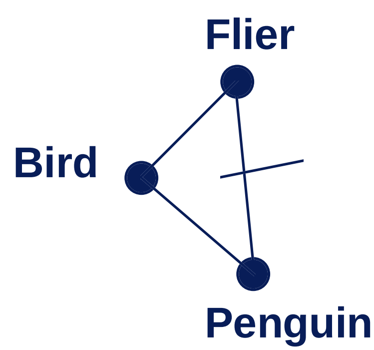
a conflict: is Nixon a pacifist?

- **not conflicted**

  conflicted if there is a path of opposite sign with same starting and ending point ***already in the context***

- **not preempted**

  preempted if there is a conflicting path with more direct information about the path's endpoint

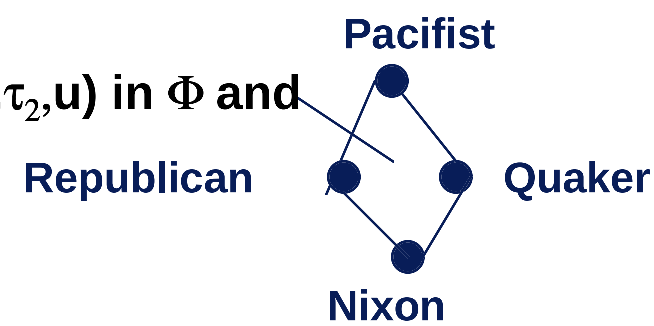


An inheritable path forms a credulous extension

Conflicts and Preemptions in more detail:

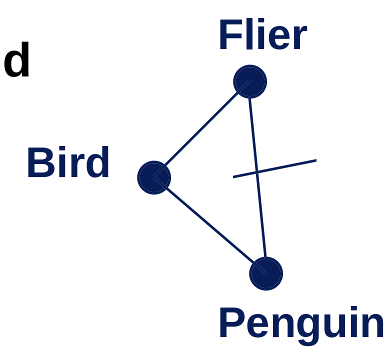
Conflicts:
A path of the form $\pi(x,\sigma,y)$ is conflicted with a path of the form $\overline{\pi}(x,\tau,y)$

Preemption:
<u>A positive path</u> $(x,\sigma,u) \dashrightarrow y$ is preempted in
π the context (Γ,Φ) if there is a node v such that
(i) either v=x or there is a path of the form $\pi(x,\tau_1,v,\tau_2,u)$ in Φ and
(ii) $v \nrightarrow y$ in Γ



<u>A negative path</u>

$(x,\sigma,u) \nrightarrow y$ is preempted in
π the context (Γ,Φ) if there is a node v such that
(i) either v=x or there is a path of the form $\pi(x,\tau_1,v,\tau_2,u)$ in Φ and
(ii) $v \dashrightarrow y$ in Γ

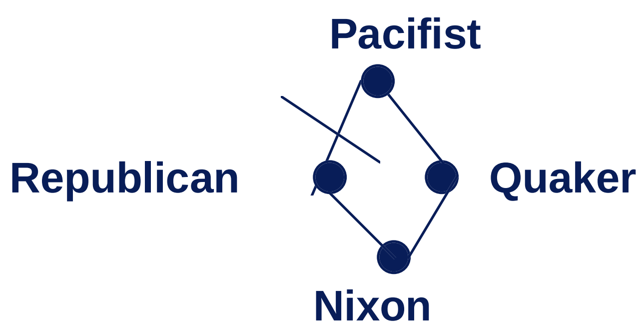


Note that the definition of preemption formalizes the notion of <u>specificity</u>; Penguins are more specific than birds

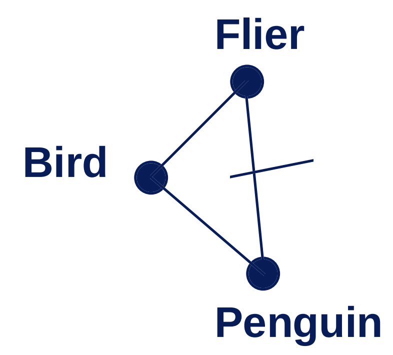
What do we do when there are conflicts, preemptions in the network?

•For the Nixon diamond, 2 approaches:
---- accept *either* that Nixon is a pacifist or that he's not a pacifisit
(that is, accept one coherent path: credulous reasoning)

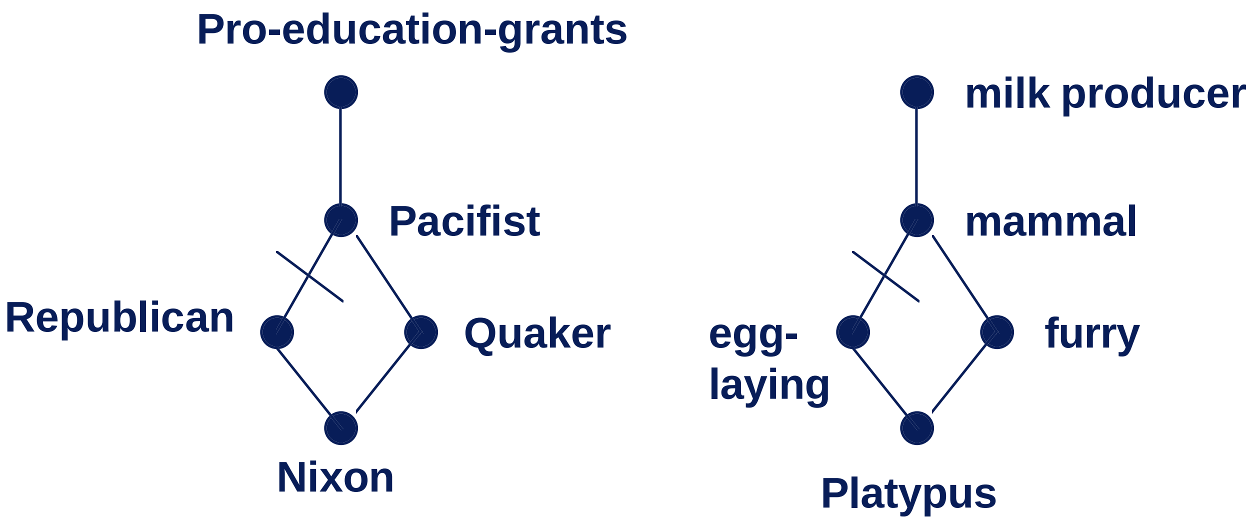
--- decide to conclude nothing about Nixon's pacifism
because of the potential conflict
(skeptical reasoning)



•For the penguin triangle, only one reasonable approach:
---- conclude that penguins are not fliers
(the argument that penguins are not fliers
is stronger than the argument that penguins are fliers,
due to specificity)

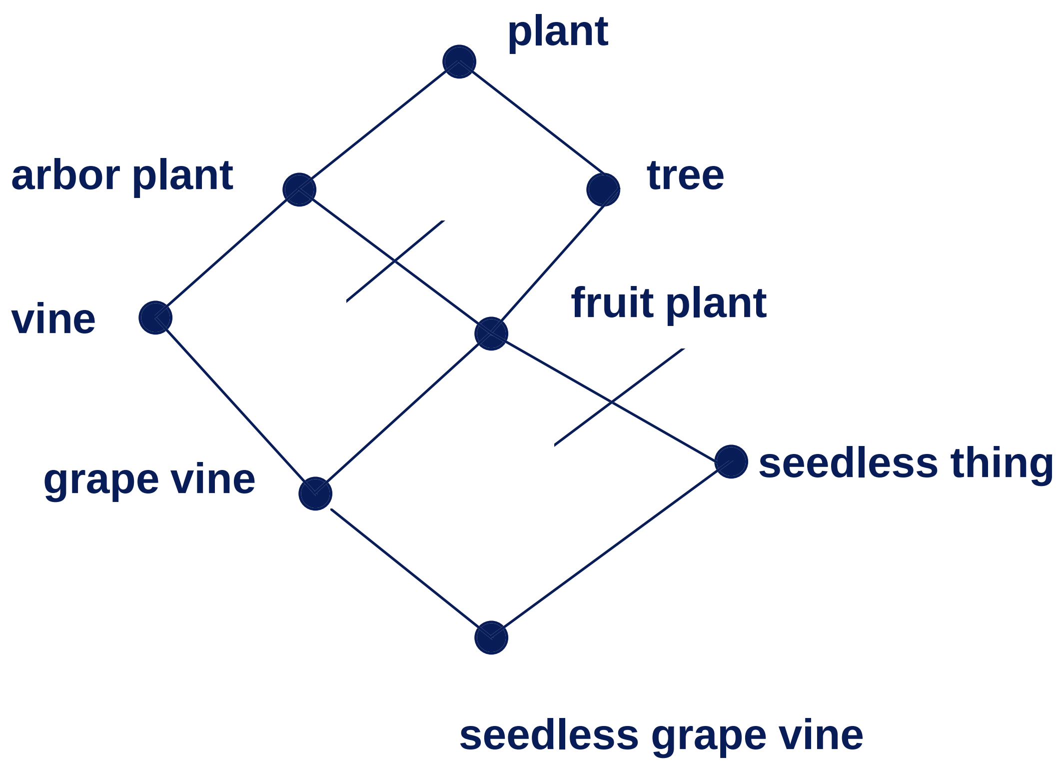


Should we propagate ambiguity?



Ambiguity Propagation: where it makes sense

(Stein, 1989, 1990, 1992)



Whether or not a seedless grape vine is a fruit plant, it’s certainly a plant!
Classical skeptical inheritance is ambiguity blocking; can’t reason beyond conflict
If we take “ideally skeptical inheritance” as intersection of credulous extensions, get the desired conclusion: seedless grape vines (and grape vines) are plants

Computing Inheritance

•Upwards inheritance is efficient (polynomial)
downwards inheritance is intractable (NP-hard) (Selman & Levesque, 89)

•Stein gives a polynomial algorithm based on an
upward traversal, removal of problematic, preempted edges
(AIJ, 1992)