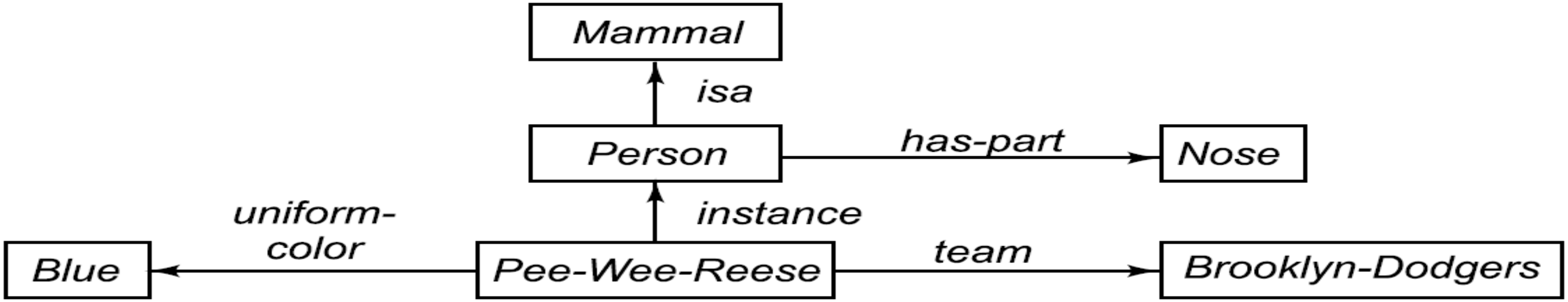
Weak Slot-and-Filler Structures

- Index assertions by the entities they describe.
- Make it easy to describe properties of relations.
- Are a form of object-oriented programming.
- Support both monotonic and Nonmonotonic inference.

A Semantic Network



Representing Nonbinary Predicates

- **Unary Predicates** can be rewritten as binary ones.

$$man(Marcus)$$

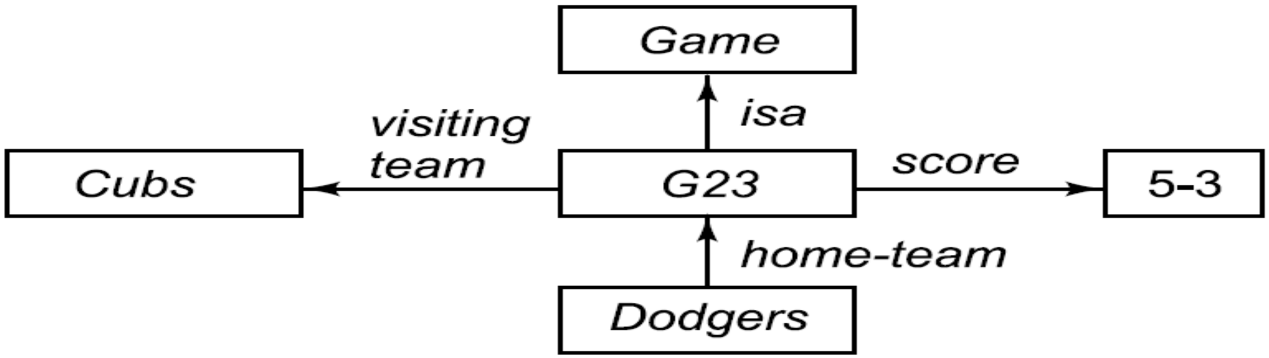
could be rewritten as

$$instance(Marcus,\ Man)$$

- **N-Place Predicates**

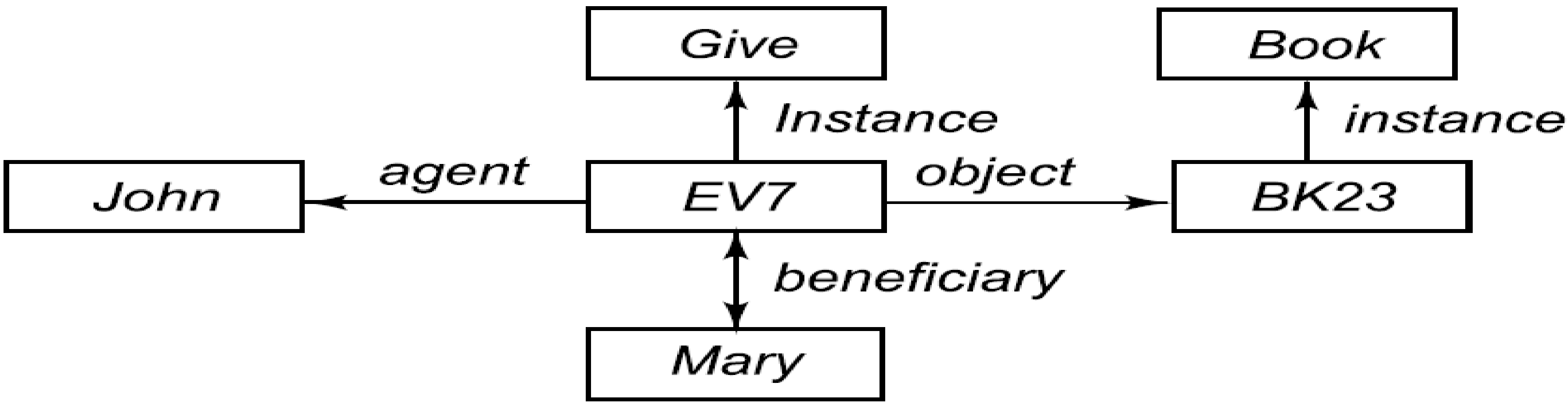
$$score(Cubs,\ Dodgers,\ 5\text{-}3)$$

becomes



A Semantic Net Representing a Sentence

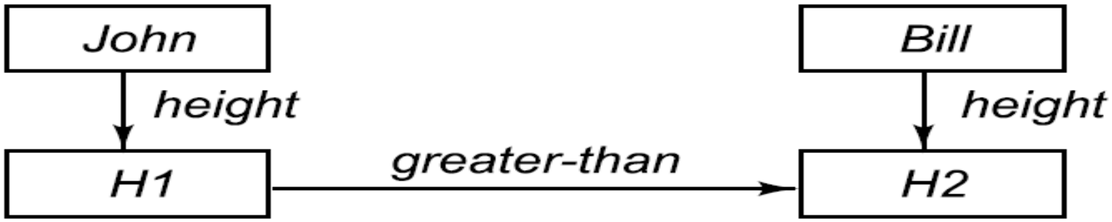
- “John gave the book to Mary.”



Some Important Distinctions

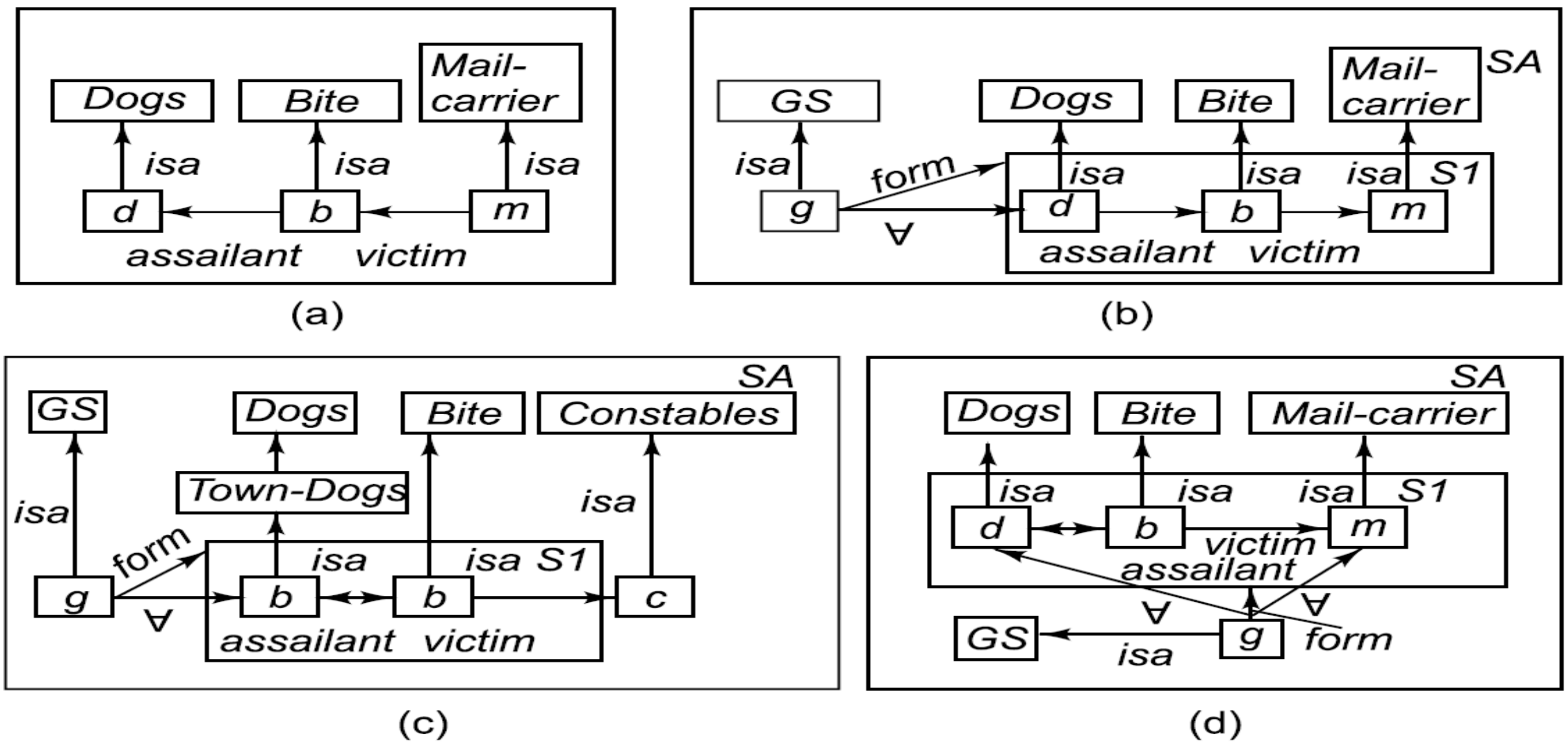
- First try :

- Second try :



- Third try :

Partitioned Semantic Nets



a) The dog bit the mail carrier.
b) Every dog has bitten a mail carrier.
c) Every dog in town has bitten the constable.
d) Every dog has bitten every mail carrier.

A Simplified Frame System

Person	
isa :	*Mammal*
cardinality :	6,000,000,000
* *handed* :	*Right*
Adult-Male	
isa :	*Person*
cardinality :	2,000,000,000
* *height* :	5-10
ML-Baseball-Player	
isa :	*Adult-Male*
cardinality :	624
**height* :	6-1
* *bats* :	equal to handed
* *batting-average* :	.252
* *team* :	
* *uniform-color* :	

A Simplified Frame System (Cont'd)

Fielder	
isa :	*ML-Baseball-Player*
cardinality :	376
**batting-average :*	.262
Pee-Wee-Reese	
instance :	*Fielder*
height :	5-10
bats :	*Right*
batting-average :	.309
team :	*Brooklyn-Dodgers*
uniform-color :	*Blue*
ML-Baseball-Team	
isa:	*Team*
cardinality :	26
* *team-size :*	24
* *manager :*	

A Simplified Frame System (Cont’d)

Brooklyn-Dodgers	
instance :	*ML-Baseball-Team*
team-size :	24
manager :	*Leo-Durocher*
players :	{*Pee-Wee-Reese*,...}

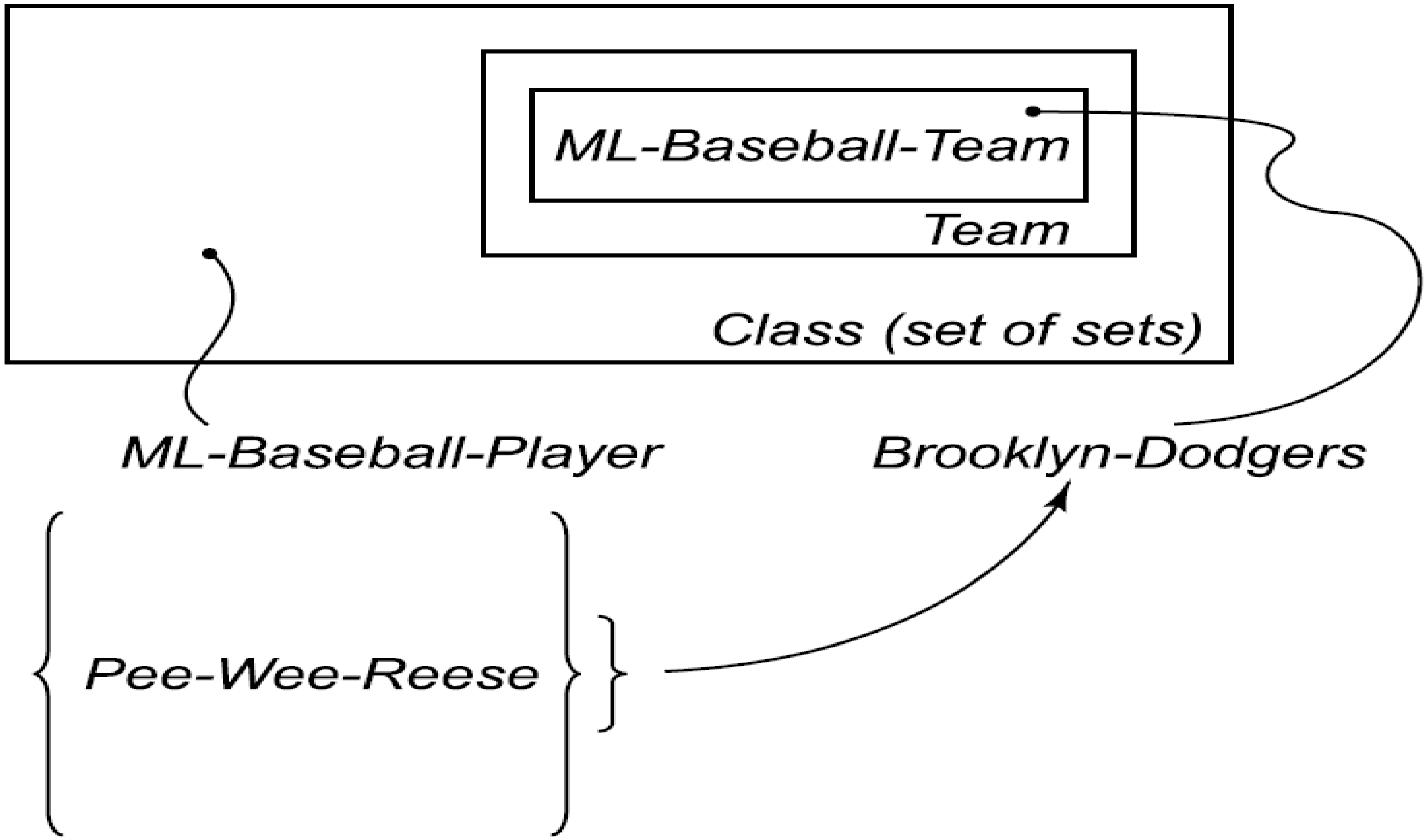
Representing the Class of All Teams as a Meta Class

Class	
instance :	*Class*
isa :	*Class*
* *cardinality* :	
Team	
instance :	*Class*
isa :	*Class*
cardinality :	{the number of teams that exist}
**team-size* :	{each team has a size}
ML-Baseball-Team	
isa :	*Mammal*
instance :	*Class*
isa :	*Team*
cardi nality :	26 {the number of baseball teams that exist}
* *team-size* :	24 {default 24 players on a team}
* *manager* :	

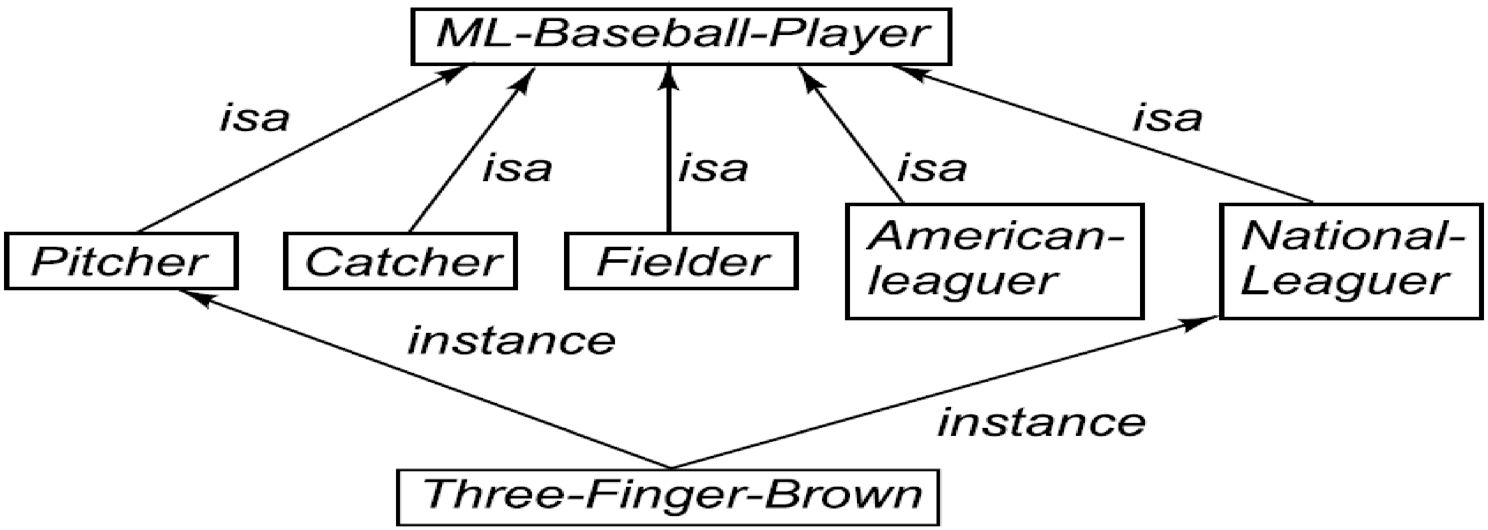
Representing the Class of All Teams as a Meta Class (Cont'd)

Brooklyn-Dodgers	
instance :	*ML-Baseball-Team*
isa :	*ML-Baseball-Player*
team-size :	*24*
manager :	*Leo-Durocher*
** uniform-color :*	*Blue*
Pee-Wee-Reese	
instance :	*Brooklyn-Dodgers*
instance :	*Fielder*
uniform-color :	*Blue*
batting-average :	*.309*

Classes and Metaclasses



Representing Relationships among Classes



ML-Baseball-Player
  is-covered-by : {*Pitcher, Catcher, Fielder*}
  {*American-Leaguer, National-Leaguer*}

Pitcher
  isa : *ML-Baseball-Player*
  mutually-disjoint-with : {*Catcher, Fielder*}

Representing Relationships among Classes (Cont'd)

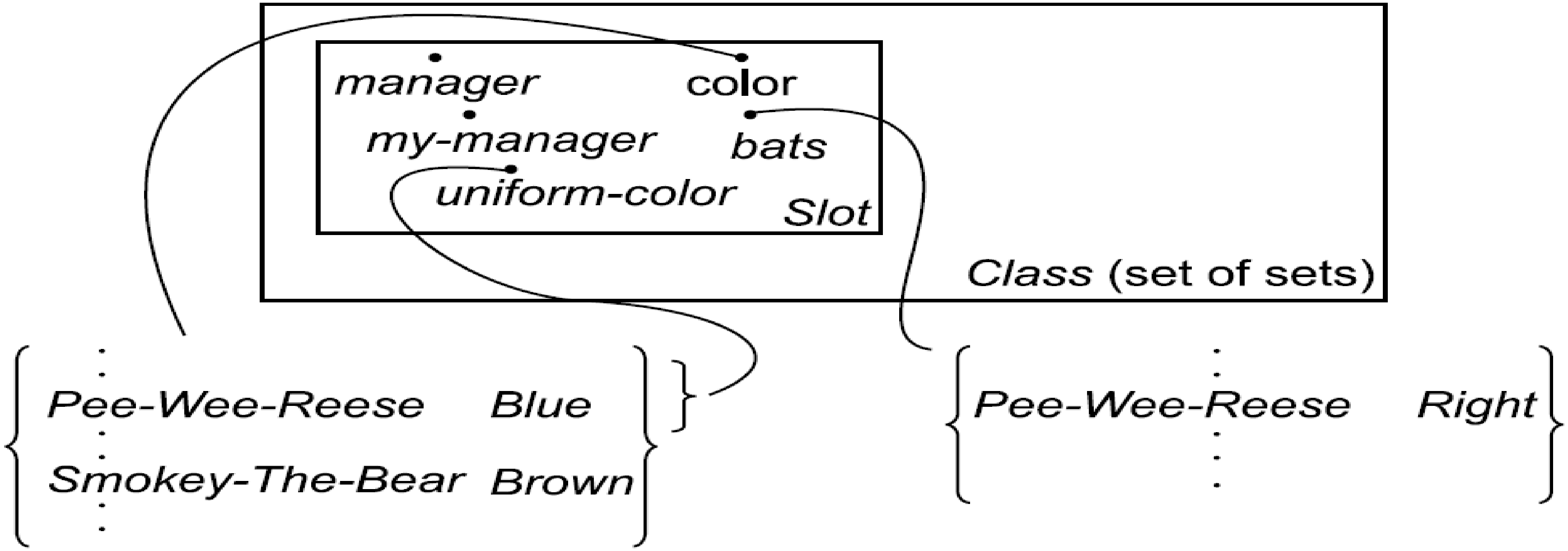
Catcher	
isa :	*ML-Baseball-Player*
mutually-disjoint-with:	*{Pitcher, Fielder}*
Fielder	
isa :	*ML-Baseball-Player*
mutually-disjoint-with :	*{Pitcher, Catcher}*
American-Leaguer	
isa :	*ML-Baseball-Player*
mutually-disjoint-with :	*{National-Leaguer}*
National-Leaguer	
isa :	*ML-Baseball-Player*
mutually-disjoint-with :	*{American-Leaguer}*
Three-Finger-Brown	
instance :	*Pitcher*
instance :	*National-Leaguer*

Slots as Full – Fledged Objects

We want to be able to represent and use the following properties of slots (attributes or relations) :

- The classes to which the attribute can be attached.
- Constraints to either the type or the value of the attribute.
- A value that all instances of a class must have by the definition of the class.
- A default value for the attribute.
- Rules for inheriting values for the attribute.
- Rules for computing a value separately from inheritance.
- An inverse attribute.
- Whether the slot is single-valued or multi-valued.

Representing Slots as Frames, I



Representing Slots as Frames, II

Slot	
isa :	*Class*
instance :	*Class*
* *domain* :	
* *range* :	
* *range-constraint* :	
* *definition* :	
* *default* :	
* *transfers-through* :	
* *to-compute* :	
* *inverse* :	
* *single-valued* :	
manager	
instance :	*Slot*
domain :	*ML-Baseball-Team*
range :	*Person*
range-constraint :	λx {*baseball-experience x.manager*)
default :	
inverse :	*manager-of*
single-valued :	TRUE

Representing Slots as Frames, III

my-manager

instance :	*Slot*
domain :	*ML-Baseball-Player*
range :	*Person*
range-constraint :	λx (*baseball-experience x.my-manager*)
to-compute :	λx (x.*team*).*manager*
single-valued :	TRUE

color

instance :	*Slot*
domain :	*Physical-Object*
range :	*Color-Set*
transfers-through :	*top-level-part-of*
visual-salience :	*High*
single-valued :	FALSE

Representing Slots as Frames, IV

uniform-color

instance :	*Slot*
isa :	*color*
domain :	*team-player*
range :	*Color-Set*
range-constraint :	not *Pink*
visual-salience :	*High*
single-valued :	FALSE

bats

instance :	*Slot*
domain :	*ML-Baseball-Player*
range :	{*Left, Right, Switch*)
to-compute :	λx *x.handed*
single-valued :	TRUE

Associating Defaults with Slots

batting-average

instance :	*Slot*
domain :	*ML-Baseball-Player*
range :	*Number*
range-constraint :	$\lambda x\ (0 \leq x.\mathit{range\text{-}constraint} \leq 1)$
default :	.252
single-valued :	TRUE

fielder-batting-average

instance :	*Slot*
isa :	*batting-average*
domain :	*Fielder*
range :	*Number*
range-constraint :	$\lambda x\ (0 \leq x.\mathit{range\text{-}constraint} \leq 1)$
default :	.262
single-valued :	TRUE

A Shorthand Notation for slot-Range Specification

ML-Baseball-Player
bats : MUST BE {*Left*, *Right*, *Switch*}

Representing Slot - Values

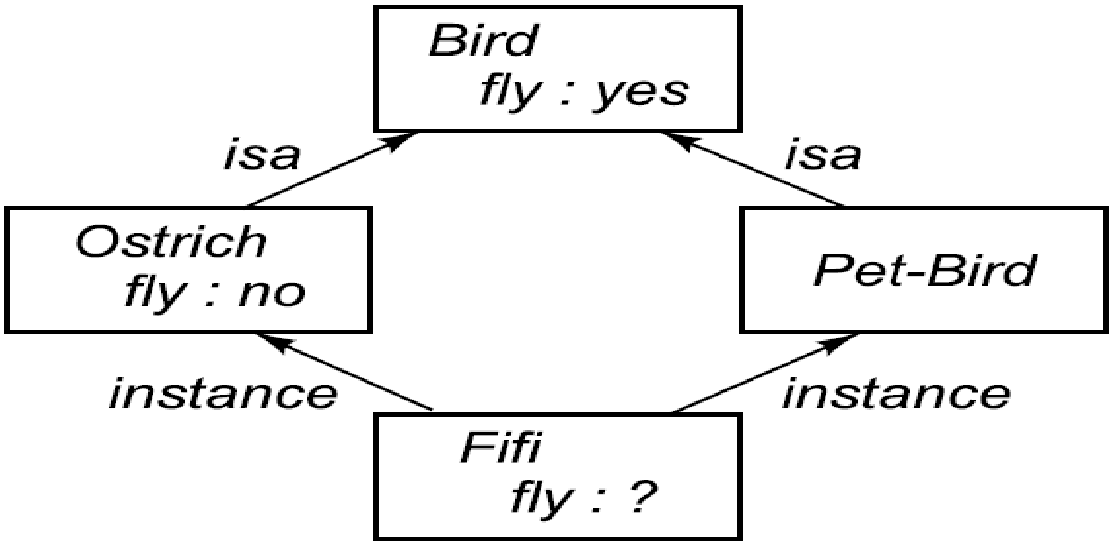
- As simple frames :

John
height : 72
Bill
height :

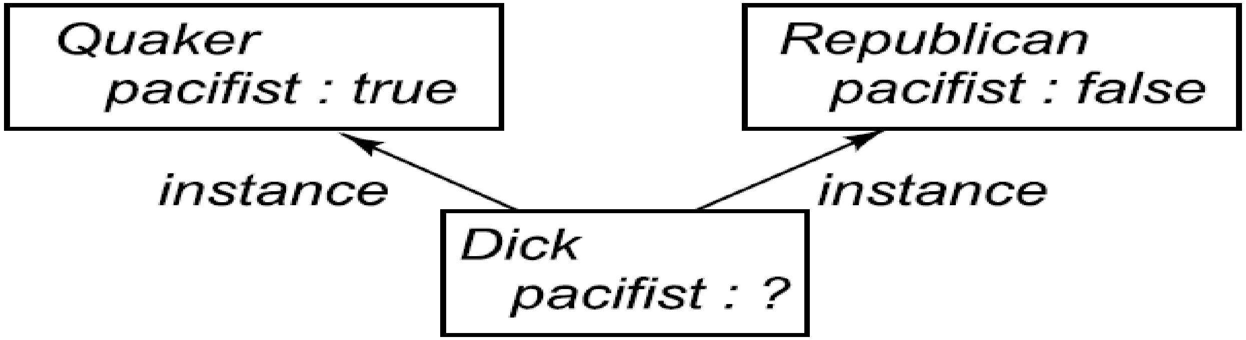
- Using Lambda Notation :

John
height : 72; $\lambda x\ (x.height > Bill.height)$
Bill
height : $\lambda x\ (x.height < John.height)$

Tangled Hierarchies

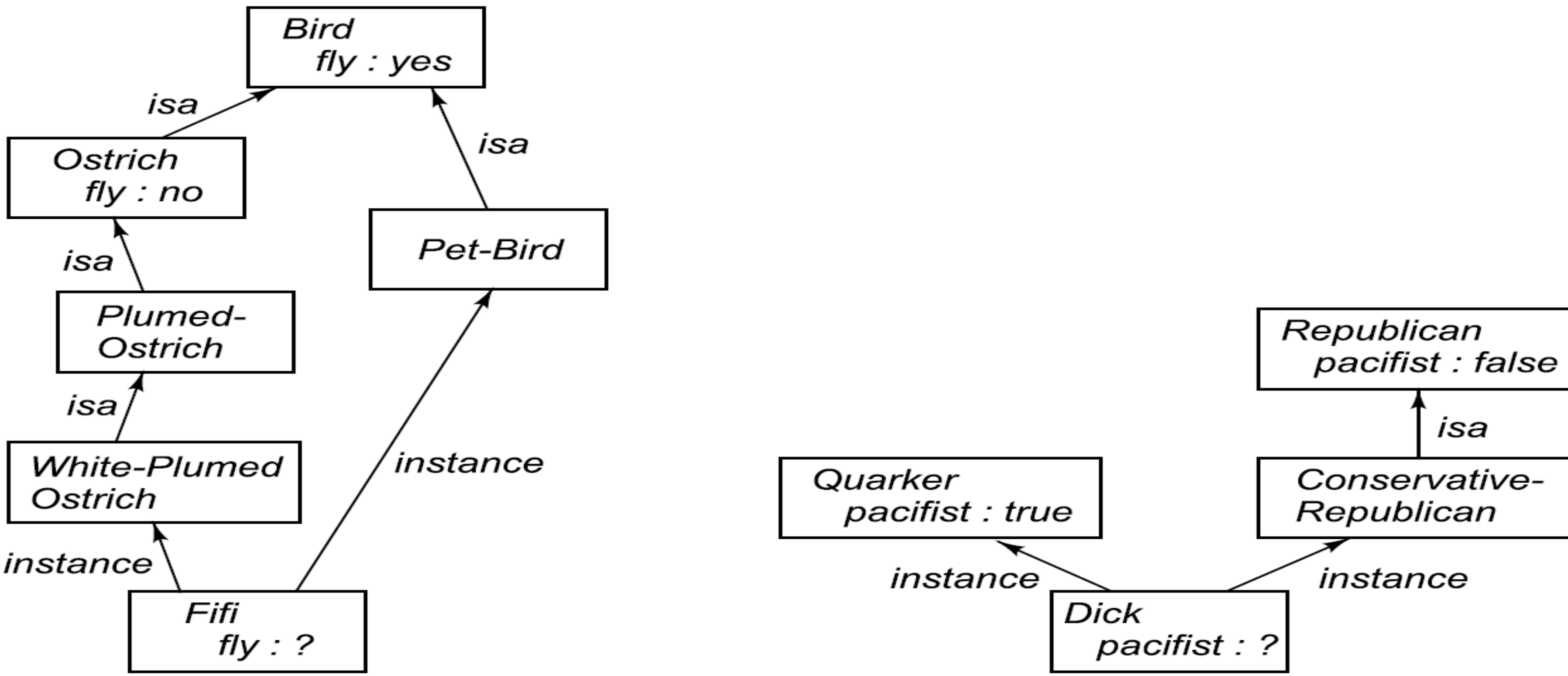


(a)



(b)

More Tangled Hierarchies



(a) (b)

Defining Property Inheritance

- **Inferential Distance :**

Class1 is closer to Class2 than to Class3, if and only if Class1 has an inference path through Class2 to Class3 (in other words, Class2 is between Class1 and Class3).

We can now define the result of inheritance as follows: The set of competing values for a slot S in a frame F contains all those values that

- **Can be derived from some frame X that is above F in the isa hierarchy**
- **Are not contradicted by some frame Y that has a shorter inferential distance to F than X does**

Algorithm : Property Inheritance

To retrieve a value V for slot S of an instance F do:

1. Set CANDIDATES to empty.

2. Do breadth-first or depth-first search up the isa hierarchy from F, following all instance and isa links. At each step, see if a value for S or one f its generalizations is stored.

(a) If a value is found, add it to CANDIDATES and terminate that branch of the search.
(b) If no value is found but there are instance or isa links upward, follow them.
(c) Otherwise, terminate the branch.

3. For each element C of CANDIDATES do:

(a) See if there is any other element of CANDIDATES that was derived from a class closer to F than the class from which C came.
(b) If there is, then, remove C from CANDIDATES.

4. Check the cardinality of CANDIDATES:

(a) If it is 0, then report that no value was found.
(b) If it is 1, then return the single element of CANDIDATES as V.
(c) If it is greater than 1, report a contradiction.