# Semantic Networks

- Using logic as representation formalism has some practical problems
- Some more intuitive and easy to use formalisms are needed
- Cognitive psychology affirms that:
  - The representation and query of knowledge are done based on its relations
- Semantic networks is a KR formalism that allows to represent knowledge as a graph where:
  - The nodes represent concepts
  - The edges (directed) represent relations among concepts

# Semantic Networks

- They are a constrained version of predicate calculus language
- The goal of this formalism is to represent declaratively the elements of a domain
- Some specific mechanisms of reasoning can be defined to make some queries about the knowledge that are represented
  - Are two concepts related?
  - What relations link two concepts?
  - What is the nearest concept that are related to a pair of concepts?
- A richer semantics can be defined to allow more complex queries
  - Concept taxonomies (class/subclass/instance)
  - Generalization/Specialization

# Limitations of semantic networks

- There are many formalisms under the name semantic networks with different expressive capabilities but always with a formal reasoning model
- Different levels of abstraction are mixed in the representation
  - Concepts/instances/values
  - Relations/properties
- A more structured formalism is necessary
- A formal semantic model for reasoning is necessary

# Frames

- Formalism that gives a structure to semantic networks
- A Frame is a collection of *attributes* and the description of their characteristics
- The *relations* connect frames
- There is an explicit separation between relations and attributes
- Relations and attributes have also properties to describe their semantics
- Some Frame languages you may know: Entity-Relationship Diagrams, UML

# Frames

- The reasoning formalism that supports the declarative part is *description logic* (this is also the logic object oriented languages are based on)
  - Concept inclusion (specialization/generalization)
  - Value and attribute inheritance
  - Set relations (union, intersection, membership, transitivity)
- Frames usually add also a procedural formalism to complement the declarative formalism
  - Functions and methods that reduce the computational cost of the inference

# Elements of the formalism

- A frame represents a concept
- It has a declarative part (attributes) and a procedural part (methods)
- The declarative part allows to describe the semantics of the concept (characteristics)
- The procedural part allows to define how to obtain information or to perform computations related to its attributes or to its relations with other frames
- A frame is described by a name, a list of attributes and a list of methods

# Elements of the formalism - Frames

Description of a frame:

**Frame** <name>
  slot <slot-name>[1]
  slot <slot-name>[1]
  ...
  slot <slot-name>[1]
  methods [2]
    **procedure** <method-name> (parameters) [I/noI]
    ...
    **function** <method-name> (parameters) return <datatype> [I/noI]

[1] Each slot can have modifiers that can override the global definition of the slot

[2] The procedures/actions that describe the methods use the variable F as a self reference to the frame or instance of frame that calls the method

# Elements of the formalism - Relations

- A relation connects frames
- A relation has characteristics that describe its semantics, properties and behaviour
- Relations are the basis of the inference on frames: inheritance of slots
- We will distinguish two kinds:
  - Taxonomical: is-a (class/subclass), instance-of (instance/class)
  - User defined: The rest of relations

# Elements of the formalism - Slots

- The slots describe the characteristics of a frame
- They have a set of characteristics (facets) that allow to describe their semantics
  - Domain, range, cardinality, default value, ...
- The allow to define procedures that can perform computations triggered by events (demons)
- There are four kinds of demons:
  - `If-needed` (when the value of the slot is accessed)
  - `if-added` (when a slot is given a value),
  - `if-removed` (when the value of a slot is deleted)
  - `if-modified` (when the value of a slot is modified)
- Demons have no parameters
- We can declare how slots are affected by inheritance

# Elements of the formalism - Slots

Description of a slot:

**Slot** `<name>`
++ domain (list of frames)
++ range `<primitive-datatype>`
++ cardinality (1 or N)
value (value or list of values)
demons `<demon-type>`
**procedure** `<procedure-name>` / **function**`<function-name> returns <datatype>*`
inheritance (by taxonomic relations: YES/NO; by user defined relations: YES/NO)

- The facets marked ++ are compulsory for the description of a slot
- The procedures/functions associated to demons do not have parameters. The variable `F` is used as self reference
- The demons `if-needed` only can be defined as functions
- By default the inheritance by taxonomical relations is YES and by user defined relations is NO
- To consult the value of a slot the syntax `<frame-name>.<slot-name>` is used. The result could be a single value or a list of values depending on the cardinality of the slot

# Elements of the formalism - Methods

The methods are procedures or functions that allow to obtain information from a frame

- A method could be called from an abstract frame (classes) or from the instance of a frame
- They can be
  - Inheritable (the subclasses/instances can call the method)
  - Non inheritable (only the frame that define the method can call it)
- Methods can have parameters

# Elements of the formalism - Relations

Relations allow to connect frames to represent the relationships among concepts

- Their semantic is defined from a set of properties: Domain, range, cardinality, inverse, transitivity, composition, ...
- Procedural mechanisms can be defined (demons) that are triggered by certain events:
  - `If-added`: if the relation between two instances is created
  - `If-removed`: if the relation between two instances is deleted
- The behaviour of a relation to inheritance of properties can also be defined (what slots can be inherited). Since relations are defined in both directions, the slots are inherited in the adequate direction (from the frame where the slot is defined to the frame that has to inherit the slot)

# Elements of the formalism - Relations

**Relation** <name>
++ domain (list of frames)
++ range (list of frames)
++ cardinality (1 or N)
++ inverse <name> (cardinality: 1 or N)
transitive YES/NO [NO by default]
composed NO/<description of the composition> [NO by default]
demons (<demon-type> procedure <procedure-name>
inheritance (list of slots) [empty list by default]

- The properties marked ++ are compulsory to describe a relation
- The procedures associated to demons have no parameters. They use the variables D and R as a reference to the source and destination frames respectively of the relation that has been added or removed between two frames.

# Elements of the formalism - Programming

- The syntax `<frame-name>.<relation-name>` will return the frame (if cardinality is 1) or the list of frames (if cardinality is N) that are connected to the current frame through the relation `<relation-name>`
- To access to the actual cardinality of a relation the function `card(<frame-name>.<relation-name>)` can be used
- Predefined relations:
  - relation `is-a` (inverse: `has-as-subclass`) transitive YES
  - relation `instance-of` (inverse: `has-as-instances`) composition: `instance-of` $\otimes$ `is-a`

# Elements of the formalism - Programming

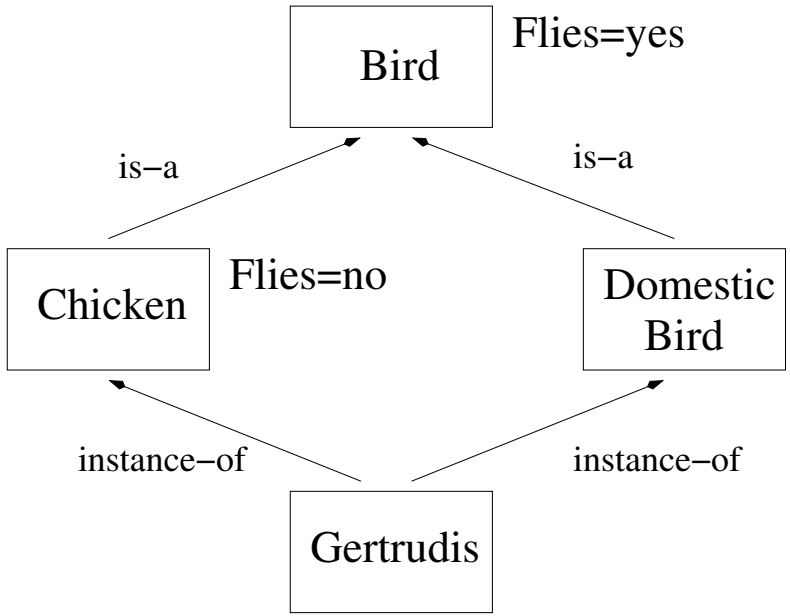
- Predefined boolean functions
  - `slot?(<frame>)` returns if `<frame>` has this slot or not (activating inheritance if needed)
  - `relation?(<frame>)`
    returns if `<frame>` is related with any other through the specified relation
  - `relation?(<frame-o>,<frame-d>)`
    returns if the specified relation links `<frame-o>` and `<frame-d>`

# Inheritance

- Inheritance is the main deduction mechanism for frames
- It allows to obtain the value or values of a slot or its definition from another frame through the relations that connect them
- Taxonomic relations has inheritance activated by default (the definition of the slots is inherited)
- On the rest of relation inheritance has to be activated explicitly (the value of the slot is inherited)
- Given a frame it is possible that the representation allows to inherit the value of a slot from multiple relations or frames (Multiple inheritance)

# Multiple inheritance

- Multiple inheritance is valid depending on the semantics of the inherited slot




- The **inferential distance algorithm** allows to determine which frame is the correct to inherit from

# Inferential distance algorithm

1. Search for the set of frames that allow to inherit the slot value $\longrightarrow$ **Candidates**
2. Delete from Candidates all frames that are ascendant of another frame in the list
3. If the number of candidates is:
   1. 0 $\longrightarrow$ the slot can not be inherited
   2. 1 $\longrightarrow$ this is the correct value
   3. $> 1$ $\longrightarrow$ Multiple inheritance problem if slot cardinality is not N
4. Sometimes a problem of multiple inheritance can be solved procedurally (demons)