CS156 In-Class Exercise #3:

Semantic Networks and Frame-Based Representations

GROUP MEMBER NAMES:

1.     Mohit Kunder

2.    Rashmi Vishwanath Bhat

3.    Alisha Rath

4.    Farhan Ansari

ENTER “COMPLETE” OR “INCOMPLETE” here COMPLETE

# INTRODUCTION

Although we have not formally discussed semantic networks in class or in homework assignments, they are an important AI representation as well as being related to frame-based representations. So, this in-class exercise will involve doing some reading about semantic networks. Then you will be asked to construct a semantic network and then translate it to a frame-based representation.

Semantics networks are discussed in section 10.5 (pages 329-333) in the required textbook. Frame representations will be covered in this exercise and if time allows during lecture.

It is suggested that each person in the group read the text up each set of questions in an exercise, e.g., in Exercise 1, Exercise 2, …etc. Then the group collectively discusses what was read, answers any questions that other group members might have, and then answer the exercise questions together.

The topic of rule-based representation will continue to be covered during lecture and either homework or in-lecture exercises.

# BACKGROUND

In some cases, more domain-specific knowledge may be needed than that required to solve a problem using search. In such instances, some form of representing and manipulating the knowledge is needed. Knowledge is stored in a knowledge base using a particular representation and inference techniques or algorithms are used to manipulate the knowledge.

Various representation schemes have been developed for knowledge representation and these have been categorized as follows:

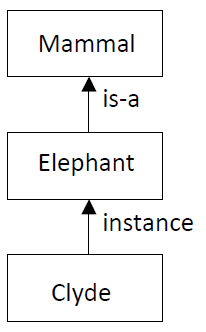
* Logical representation schemes – Inference rules and proof procedures are used to find solutions to problem instances, e.g. first-order predicate logic.
* Rule-Based schemes – Knowledge is represented as antecedent and consequences in “IF-THEN” type syntax. A rule-based inference engine helps find a solution to a problem.
* Procedural representation schemes– Knowledge is represented as a list of instructions to solve problems, e.g. production rule systems and essentially consist of a number of if then-else rules.
* Network representation schemes – Information is represented as a graph, e.g. semantic networks, conceptual graphs. The nodes in the graph represent objects or concepts and the arcs relationships between them.
* Structured representation schemes – These schemes extend networks such that every node in the network is a complex structure that contains information regarding attributes of the object. These complex structures usually consist of slots with attached values, e.g., scripts and frames.
* Structures for representing incomplete and inconsistent knowledge, e.g., Bayesian reasoning, and Dempster-Shafer theory.

One of the problems encountered in representing knowledge is the **frame** problem. This problem refers to the difficulty of representing facts that change often as well as those that do not. How successful a program is at solving a problem is largely dependent on using the correct scheme to represent the knowledge needed to solve the problem.

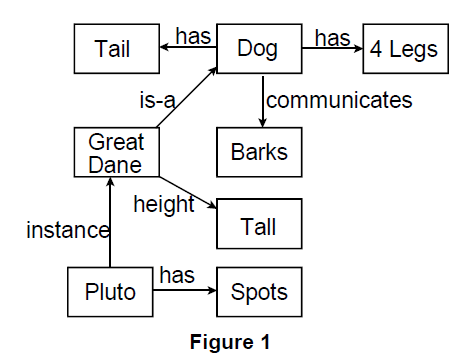
# Semantic Networks

A semantic network is a graph consisting of nodes, where each node represents a fact or a concept, and arcs, where each arc represents a relationship between two nodes. Inference rules and algorithms are needed to answer questions regarding the knowledge represented in a semantic network. However, in this section we will only examine how knowledge can be represented in a semantic network.

Suppose that we want to represent the statement that Clyde is an elephant:



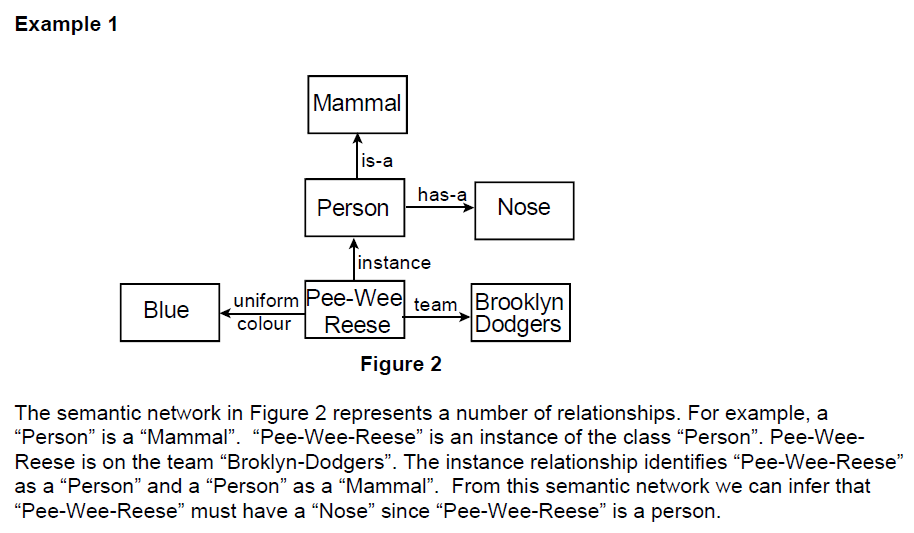
Clyde is referred to as an instance of the class Elephant. In turn any member of the class Elephant is a Mammal. Thus, the class Elephant is a subclass of the class Mammal and therefore Clyde is indirectly a Mammal. Semantic networks are used to represent inheritance relationships such as the example above. In order to differentiate between instances and subclasses the term **instance** is used to represent a relationship between a class and an instance and the term **is-a** is used to represent a relationship between a subclass and a superclass. Semantic networks also represent **has-a**/**has** relationships between objects. Both these relationships are illustrated in the figure below.

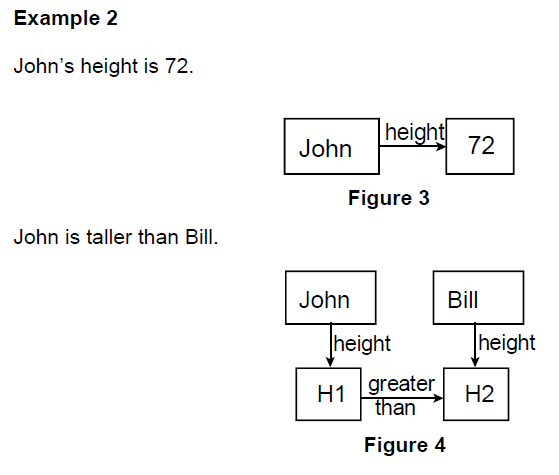


Note that in addition to is-a, instance and has/has-a relationships other relationships can also be represented in a semantic network. For example, the relationship between a dog and barks is “communicates”. An instance of a class and each subclass of a class inherits all the properties of that class. Hence, since “Great Dane” is a subclass of the class “Dog” every member of the class “Great Dane” has a tail and 4 legs. Similarly, the instance “Pluto” of the class “Great Dane” will also have a tail and 4 legs. In addition to the inherited properties an instance or subclass can have properties specific to the particular subclass or instance. For example, all Great Danes are Tall.

The instance “Pluto” has spots in addition to possessing all the other properties of great Danes.

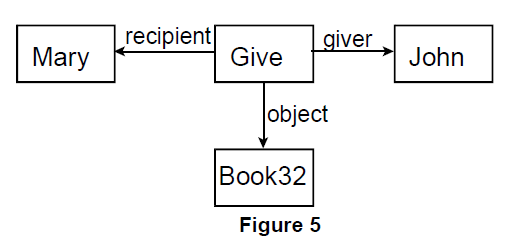
More examples of semantic networks are illustrated below:





**Example 3**

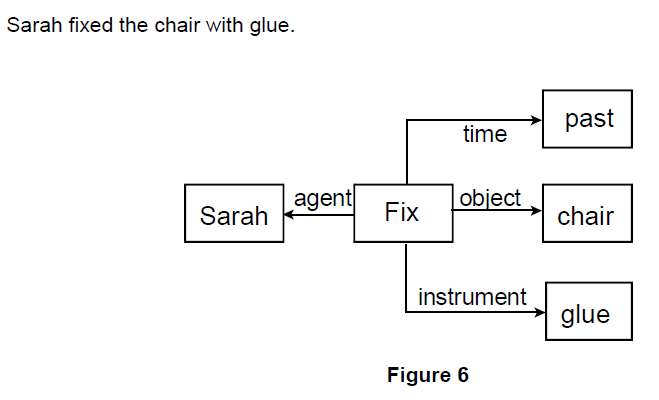
Thus, far we have examined binary relationships, i.e. relationships between at most two objects. However, it may be necessary to represent n-ary (n >2) relationships using a semantic network. In this case the n-ary relationship must be converted into a number of binary relationships. Suppose that we want to represent the fact that “John gives Mary the book”, i.e. give (john, mary, book32). This relationship involves three objects. The relationship is broken down into a number of binary relationships as indicated in Figure 5.



A number of problems were experienced with using semantic networks to represent knowledge. Firstly, the relationships that can be represented using a semantic network are very general and unstructured. Thus, semantic networks cannot be used to represent complex domains. Attempts were made to standardize the relationships that could be represented in a semantic

network and so possibly alleviate some of the problems. The development of a verb-oriented approach was one of the results of these endeavors.

This approach defines the following **case relationships**: agent, object, instrument, location and time. Every sentence is represented using a **case frame**. A case frame represents a particular action and basically consists of arcs linking nodes representing different participants involved in an action. Figure 6 represents the corresponding semantic net using this approach to represent the statement:



**Exercise 1**

Construct and insert below a semantic network to represent each of the following statements:

1. Pompeian(Marcus), Blacksmith(Marcus).

2. Mary gave the green colored vase to her favorite cousin.

3. John went downtown to deposit his money in a bank.

Answer:

1. Pompeian(Marcus), Blacksmith(Marcus).

Marcus is a pompeian by nationality and blacksmith by professor and marcus is a person.

A diagram of a person

Description automatically generated

1. Mary gave the green colored vase to her favorite cousin. Mary is the giver of the vase which is green color, the recipient of the vase is Mary’s cousin who is her favorite.

A black background with white squares

Description automatically generated

1. John went downtown to deposit his money in a bank. John is the agent who deposited object money to destination bank in the past. The bank is in location downtown.

A black and white cross with white squares

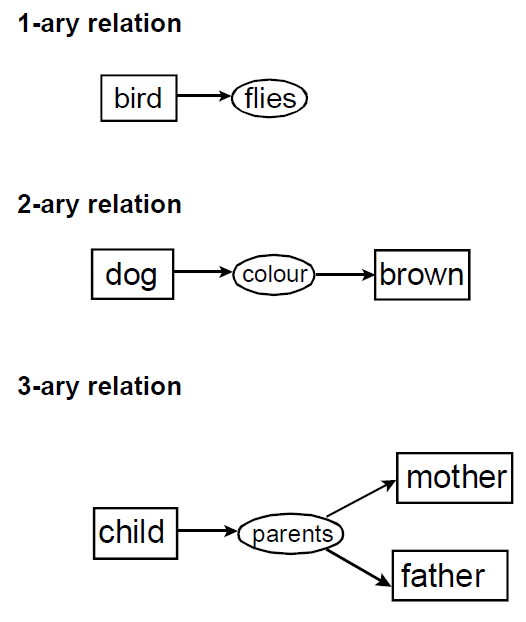
Description automatically generated

Conceptual Graphs

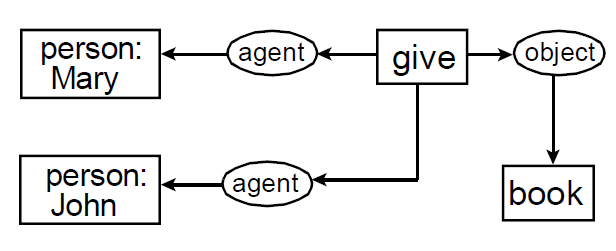
A conceptual graph is a connected bipartite graph. Conceptual relation nodes represent relations between concepts. Thus, the arcs connecting nodes are not labeled. This is one of the main differences between conceptual graphs and semantic networks. A conceptual graph consists of concepts, represented by boxes and conceptual relations are represented by ellipses. Concept nodes only have arcs to conceptual relation nodes (thus the graph is

bipartite). Concept nodes represent concrete objects, e.g. cat, telephone, restaurant, etc, and abstract objects, e.g. love, beauty and loyalty.

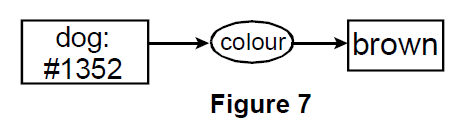
In order to represent a relation of arity of n a conceptual relation node must have n arcs. Let us look at some examples to make this clearer.



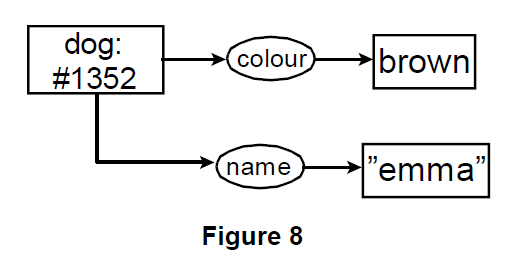
Let us look at a more detailed example. Suppose that we want to represent the relation that Mary gave John the book. The corresponding conceptual graph is illustrated below:



Notice that each concept node represents an individual and specifies the type of the individual. If an individual object is unknown a **unique marker** can be used in place of the name of the object. A unique marker is comprised of a hash symbol (#) followed by a number. Each object has its own unique marker. In the first conceptual graph in Figure 7 the name of the dog is known. However, in the second conceptual graph it is not known and a unique marker is used for this purpose.



The first conceptual graph in Figure 7 is equivalent to the conceptual graph in Figure 8.

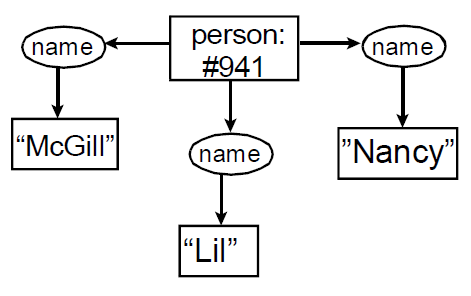


Note that when individuals of a particular type appear on their own (i.e. without the type specified before the individual name) the name of the individual must appear in inverted commas, e.g. “emma”.

Example 1

Suppose that we want to represent the following information using a conceptual graph:

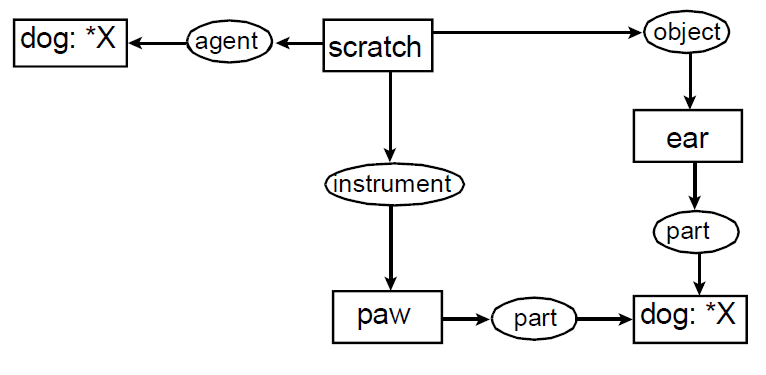
Her name was McGill and she called herself Lil, but everyone knew her as Nancy.



In addition to individual markers conceptual nodes can also contain **generic markers**. A generic marker is represented by an asterisk \*. A generic marker is used to represent an unspecified individual of a type. The type dog specified in a node is equivalent dog:\*. Name variables can also be used, e.g., \*X to indicate an unspecified individual. Let us look at an example of this. Suppose that we want to represent the following information:

The dog is scratching its ear with its paw.

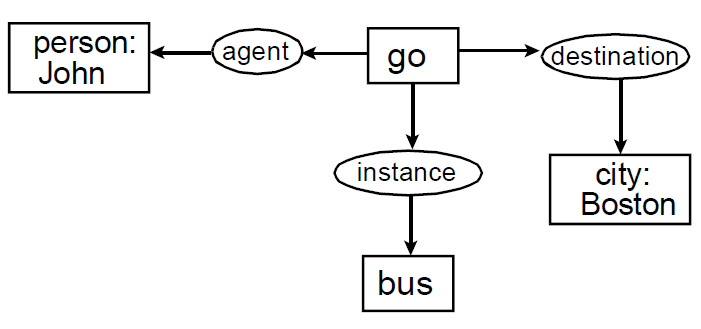
We need to indicate that the paw and ear belong to the same dog. We can use name variables for this purpose. This is illustrated below:



The format that we have used thus far to represent conceptual graphs is referred to as the **Display Form (DF)**. An alternative notation is the **Linear Form (LF)** which is a more compact notation. Suppose that we wanted to represent the following statement:

John is going to Boston by bus.

The conceptual graph in DF notation:



**Exercise 2**

Translate and insert below each of the following sentences into a conceptual graph:

1. Jane gave Tom an ice cream.

A black background with white ovals and words

Description automatically generated

2. Basketball players are tall.

A black background with white circles

Description automatically generated

3.Paul cut down the tree with an axe.

A black background with white squares

Description automatically generated

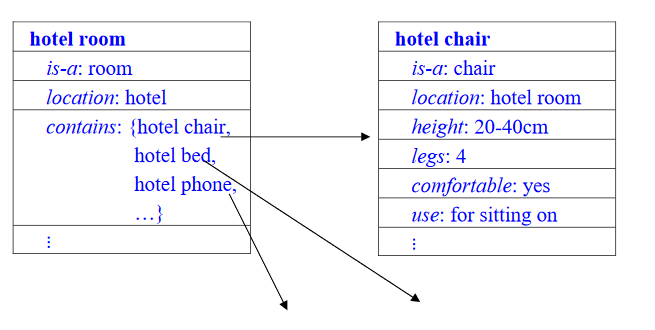
1. Place all the ingredients in a bowl and mix thoroughly.

A black background with white ovals and white text

Description automatically generated

Frame Based Systems – The Basic Idea

A frame consists of a selection of slots which can be filled by values, or procedures for calculating values, or pointers to other frames. For example:



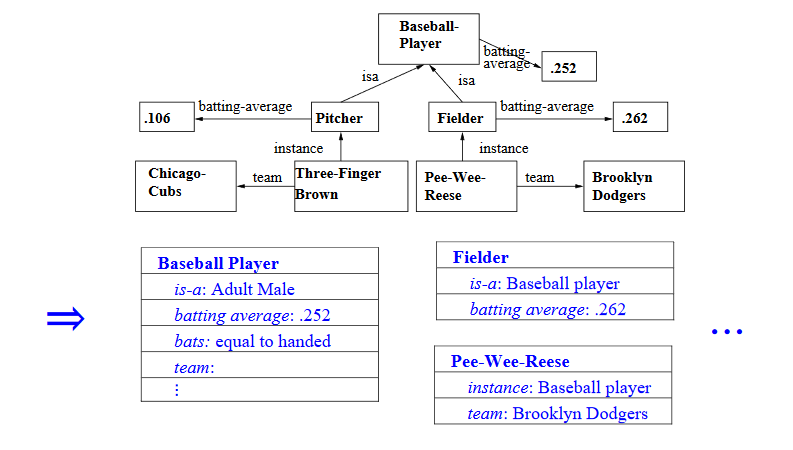
A complete frame based representation will consist of a whole hierarchy of frames connected together by a network of links/pointers. Hotel room is-a: room location: hotel contains: {hotel chair, hotel bed, hotel phone, ...}...hotel chair is-a: chair location: hotel room height: 20-40cm legs: 4 comfortable: yes use: for sitting on...

Frames as a Knowledge Representation

The simplest type of frame is just a data structure with similar properties and possibilities for knowledge representation as a semantic network, with the same ideas of inheritance and default values. Frames become much more powerful when their slots can also contain instructions(procedures) for computing things from information in other slots or in other frames. The original idea of frames was due to Minsky (1975) who defined them as “data-structures for representing stereotyped situations”, such as going into a hotel room. This type of frames are now generally referred to as scripts. Attached to each frame will then be several kinds of information. Some information can be about how to use the frame. Some can be about what one can expect to happen next, or what one should do next. Some can be about what to do if our expectations are not confirmed. Then, when one encounters a new situation, one can select from memory an appropriate frame and this can be adapted to fit reality by changing particular details as necessary.

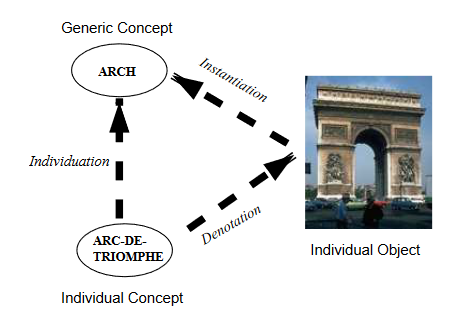
Converting Semantic Networks to Frames

It is easy to construct frames for each node of a semantic net by reading off the links, e.g.



Set Theory as a Basis For Frame Systems

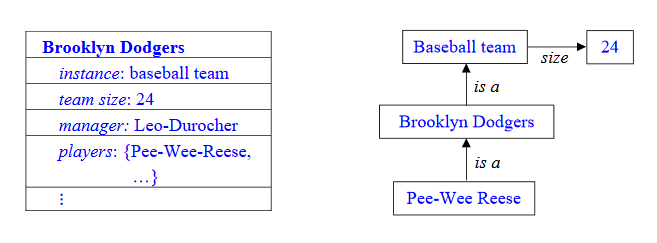
The relationship between real world instances, the representation of instances, and the representation of sets/classes of instances, is already quite familiar, e.g.



Clearly *is-a* corresponds to subset ⊆, and *instance* corresponds to element ∈. Then set theory concepts such as *transitivity, intersection*, etc. apply automatically to our frames.

Problems with Sets That Are Also Instances

Consider the frame that we might create for the Brooklyn Dodgers baseball team:

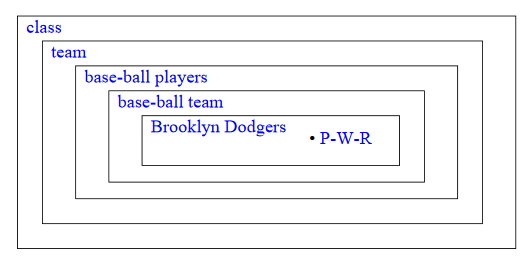


This is obviously an instance of a baseball team, but it is also a set/class of players of which Pee-Wee-Reese is an instance.

We need to be very careful about how we set up the hierarchy because we clearly don’t want Pee-Wee-Reese to inherit the general properties of baseball teams (e.g. size 24), but we do want Brooklyn Dodgers to inherit the properties that baseball teams have.

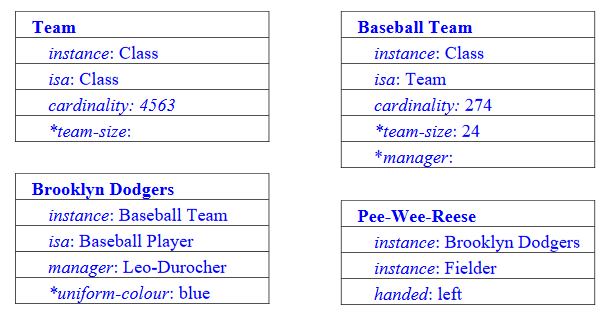
Meta-Classes

Our Brooklyn Dodgers problem is quite general. From a set of instances, we want to use inheritance to infer the instance properties from the general knowledge (default properties) of the set. But the class is also an entity in its own right, and may possess properties that belong to the class as a whole rather than to the individual instances. It is useful to make a distinction between regular classes whose instances are individual entities, and meta-classes which are classes whose instances are other classes, e.g.



Representing Meta-Classes in Frames

We can see that each frame that corresponds to a set/class needs to contain attributes about the set itself, as well as attributes to be inherited by each element of the set. We distinguish them by prefixing the latter with an asterisk (\*). For our previous example:



So Brooklyn Dodgers inherits team size 24, but Pee-Wee Reese doesn’t.

Slots as Full-Fledged Objects

We have seen that frame-based representations can be made much more powerful by allowing the slot fillers to become more than simple values. This includes being frames in their own right, with a full range of hierarchical arrangements, inheritance, etc. The main filler properties we generally want to represent are:

1.Details about whether the slot is single or multi-valued.

2.Constraints on the ranges of values or type of values.

3.Simple default values for the attribute.

4.Rules for inheriting values for the attribute.

5.Rules for computing values separately from inheritance.

6.The classes/frames to which it can be attached.

7.Inverse attributes.

Naturally, the frame system interpreter must know how to process such frames

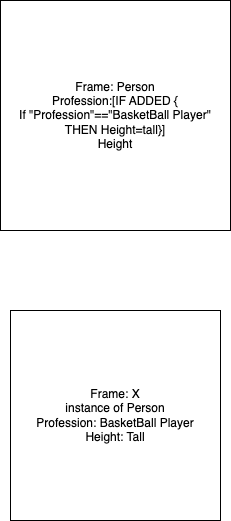
**Exercise 3**

Translate the semantic networks/conceptual graphs from Exercise 2 (1 to 4) to a Frame-Based Representation. Insert your frame-based representation below.

1. Jane gave Tom an ice cream



1. Basketball players are tall

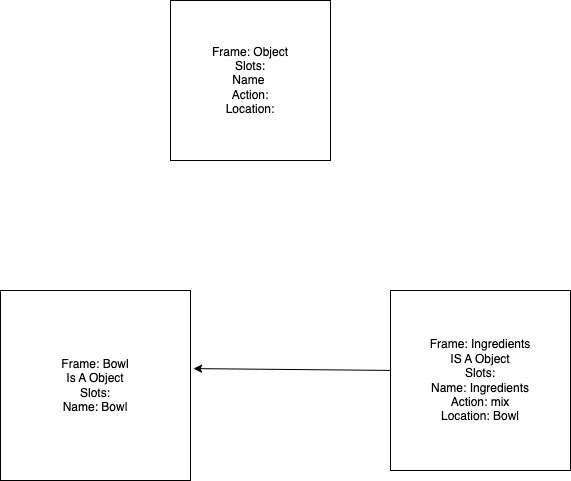


1. Paul cut down the tree with an axe.

A group of white squares with black text

Description automatically generated

1. Place all the ingredients in a bowl and mix thoroughly



**DONE**