Frames – object-oriented representation

FOL and production systems are representation methods that are "flat", in the sense that each representation unit is independent of the others. Knowledge about an object could be scattered all over the knowledge base, among unrelated sentences. In big KBs, this can become critical, therefore it is important to organize knowledge representation somehow.

In 1975, Marvin Minsky introduced "frames" as knowledge representation for object-oriented groups of procedures to recognize and deal with new situations.

In representations using frames, facts and rules are grouped in terms of the kind of objects they belong to. Thus, knowledge is not seen as just a collection of sentences, but rather it is structured in terms of what knowledge is about (i.e. the objects of knowledge).

Generic and individual frames

Individual frames represent objects; generic frames represent categories or classes of objects.

Individual frames are similar to WMEs in production systems. An individual frame is a named list of slots into which values, called fillers, can be dropped.

```
( Frame_name
      < slot_name1   filler1 >
      < slot_name2   filler2 >
      ... )
```

The names of individual frames begin with a lower-case; the names of generic frames begin with upper-case.

Fillers are atomic values (numbers or strings) or names of other individual frames or generic frames.

Slot names begin with upper-case and are prefixed with :.

```
(tripLeg10
    <: INSTANCE-OF  TripLeg>
    <: Destination  toronto> ...)

(toronto
    <: INSTANCE-OF  CanadianCity>
    <: Province  ontario>
    <: Population  4.5M> ...)

(CanadianCity
    <: IS-A  City>
    <: Province  CanadianProvince>
    <: Country  canada>)
```

Individual frames have a special slot called :INSTANCE-OF, whose filler is the name of a generic frame which indicates the category of the object represented by that individual frame (the individual frame is an instance of the generic frame).

Generic frames have a special slot called :IS-A, whose filler is the name of a more general generic frame. We say that the generic frame is a specialization of the more general one.

Slots of generic frames can have attached procedures

IF-ADDED and IF-NEEDED

The rule $Parent(x,y) \Leftarrow Mother(x,y)$ can be procedurally interpreted as following:

1. IF-NEEDED – whenever we have to solve a goal that matches $Parent(x,y)$, we reduce it to solving $Mother(x,y)$ (backward chaining). Procedurally, the connection between mothers and parents is made when we must prove something about parents.
2. IF-ADDED – whenever a fact that matches $Mother(x,y)$ is added to the KB, we also add $Parent(x,y)$ (forward chaining). The connection between mothers and parents is made when we know something new about $Mother(x,y)$.

(Trip
  <: TotalCost [IF-NEEDED computeCost] > ...)

(Lecture
  <: DayOfWeek WeekDay>
  <: Date [IF-ADDED ComputeDayofWeek] > ...)

A slot can have both a filler and an attached procedure in the same frame.

Inheritance

In a frame system, the reasoning involves creating individual instances of generic frames and filling/inferring values into slots. Generic frames can be used to fill in values that are not explicitly given at the creation of the instances. Generic frames can also trigger additional actions when fillers are provided. We can determine a value in an instance by inheritance (the child frames inherit properties from their parents frames).

For example, if we are interested in the filler for the slot :Country of the toronto frame, we can use :INSTANCE-OF that indicates to the generic frame CanadianCity, where the value is given (toronto inherits the :Country property from CanadianCity). If toronto had not a filler for :Province, we would still know by inheritance that we should look for an instance of CanadianProvince.

Inheritance of attached procedures

(ExpensiveTrip
  <: IS-A Trip> ...)

(ExoticExpensiveTrip
  <: IS-A ExpensiveTrip> ...)

if we create an instance of the frame Trip and we want to find a filler for :TotalCost in this instance, we use the attached procedure IF-NEEDED.

The same procedure can be used through inheritance if we create an instance of ExoticExpensiveTrip.

If we create an instance of the frame Lecture with the date specified explicitly:

```
(lecture1
    <:INSTANCE-OF Lecture>
    <:Date 20NOV> ...)
```

then the attached IF-ADDED procedure would be executed, filling the slot :DayOfWeek.

If we later change the :Date slot, the procedure will execute again, changing the filler for :DayOfWeek.

In a frame system, we use an inherited value only if we cannot find a filler otherwise (it is defeasible).

A slot filler in a generic frame can be overridden explicitly in its instances and its specializations:

```
(Elephant
    <:is-A Mammal>
    <:EarSize large>
    <:Color grey> ...)

(raja
    <:INSTANCE-OF Elephant>
    <:EarSize small> ...)

(RoyalElephant
    <:is-A Elephant>
    <:Color white> ...)

(clyde
    <:INSTANCE-OF RoyalElephant> ...)
```

A frame system allows for multiple inheritance. An individual/generic frame can be an instance/a specialization of more than one generic frame.

```
(AfricanElephant
    <:is-A Elephant>
    <:is-A AfricanAnimal> ...)
```

Reasoning with frames

In a frame system, reasoning starts when the system recognizes an object being an instance of a generic frame and applies the procedures triggered by that instance. These procedures can produce new data or changes in the KB. The system operation stops when no more procedures are applicable.

The system operates in a three-step loop:

1. Someone (a user or an external system) declares that an object exists, instantiating a generic frame.
2. Any slot fillers that are not provided explicitly, but can be inherited by the new instance, are inherited.
3. For each slot with a filler, if an IF-ADDED procedure can be inherited, then it is executed. By its execution, new slots can be filled or new frames can be instantiated; then goto 1.

If a filler is requested by a user, an external system or an attached procedure, then:

1. if the filler exists, then the value is returned;
2. Otherwise, any IF-NEEDED procedure that can be inherited is executed, computing the filler. The execution can also fill other slots or instantiate new frames.

if no result is produced by the steps above, then the value of the slot is unknown.

Obs. The inheritance of the values is done when the individual frame is created, but IF-NEEDED procedures are invoked only by request.

The constraints between slots are expressed by the attached IF-ADDED and IF-NEEDED procedures. The programmer decides whether reasoning is data-directed or goal-directed.