Frame Based Representation Languages

Frame languages

Typical characteristics include

- OO representation languages
- Class - subclass taxonomies
- Prototype descriptions of class instances
- Frame KR language performs standard inferences:
  - inheritance of attributes, constraints and values
  - type checking of attribute values
  - checking number of attribute values

Historical Perspective

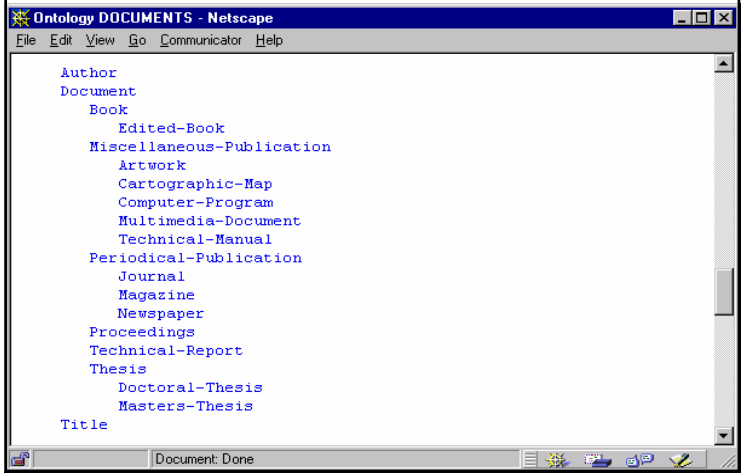
- Frame based KR systems were first developed in the mid 70's
- A seminal paper was
  - *A Framework for Representing Knowledge, Marvin Minsky, MIT-AI Laboratory Memo 306, June, 1974.*
  - http://web.media.mit.edu/~minsky/papers/Frames/frames.html
- It dealt more a new approach to organizing and using knowledge using stereotypical instances than a KR system or language.

  Here is the essence of the theory: When one encounters a new situation (or makes a substantial change in one's view of the present problem) one selects from memory a structure called a Frame. This is a remembered framework to be adapted to fit reality by changing details as necessary. A frame is a data-structure for representing a stereotyped situation, like being in a certain kind of living room, or going to a child's birthday party. Attached to each frame are several kinds of information. Some of this information is about how to use the frame. Some is about what one can expect to happen next. Some is about what to do if these expectations are not confirmed.

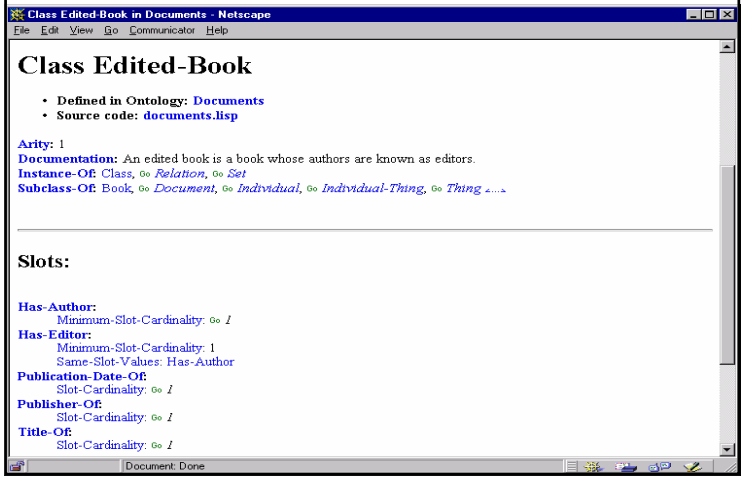
Historical Perspective

- Inspired by Minsky's vision as well as OO ideas in the wind (from Simula and Smalltalk) many researchers developed new OO AI representation systems
- **FRL**: Frame Representation Language, The FRL Primer, R. Bruce Roberts and Ira P. Goldstein, AI memo 408, July 1977, ftp://publications.ai.mit.edu/ai-publications/0-499/AIM-408.ps
- **KRL**: D.G. Bobrow and T. Winograd. An Overview of KRL A Knowledge Representation Language. Cognitive Science, 1:3 -- 46, 1977.
- **KL-ONE**: Knowledge Language One (~1980)
- **LOOPS** (Xerox) Lisp Object-Oriented Programming System, "The LOOPS Manual", Bobrow & Stefik, Xerox Corp 1983
- **KEE**: Knowledge Engineering Environment, Teknowledge (~83)

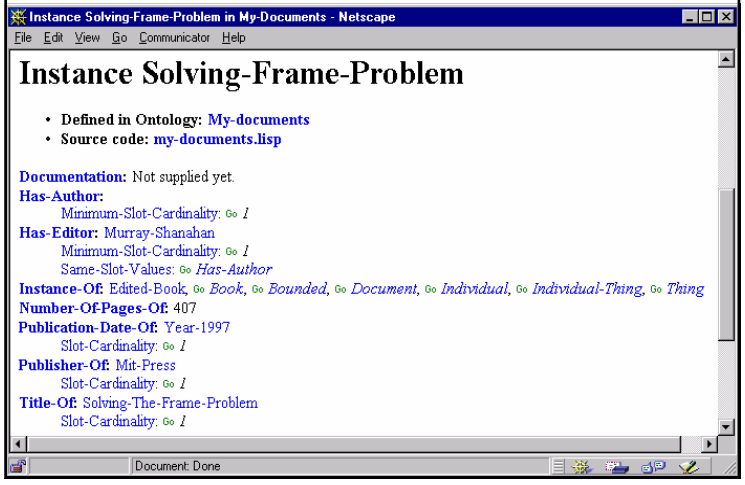
Example Class-Subclass Taxonomy



Example Class Frame



Example Instance Frame



Frames and Own Slots

- Frame - Logical Constant
  - Object, Relation, Function
- Own Slot - Binary relation
  - A frame has a set of own slots
  - An own slot has a set of values at a frame
  - Semantics:
    - OKBC: Own slot S has value V at frame F
    - KIF: (S F V)
- Example

  Fred

  Favorite-Food: Broccoli Salsa
  - KIF Translation
    - (Favorite-Food Fred Broccoli)
    - (Favorite-Food Fred Salsa)

Own Facets

- Own Facet- Ternary relation
  - An own slot at a frame has a set of own facets
  - An own facet at an own slot at a frame has a set of values
  - Semantics:
    - OKBC: Own facet Fa has value V at own slot S at frame Fr
    - KIF: (Fa S Fr V)
  - Restricted to relations whose 1st arg is a binary relation
- Example

  Fred
  Favorite-Food: Broccoli Salsa
  Value-Type: Human-Food
  - KIF Translation:

    (Value-Type Favorite-Food Fred Human-Food)

Classes and Instances

- Class - Unary relation

  E.g., Person, Horse, Product, Gate
- Instance-Of - A binary relation
  - (<=> (Instance Of ?I ?C) (holds ?C ?I))

    Note: (<=> (holds ?R @args)
    (and (relation ?R)
    (member (listof @args) ?R)))

  E.g.-

  (Instance-Of Richard Person)
  (Person Richard)
  "Richard is an instance of Person"

Subclass and Superclass

Subclass Of

(<=> (Subclass-Of ?Csub ?Csuper)
(forall ?I (=> (Instance-Of ?I ?Csub)
(Instance-Of ?I ?Csuper))))

Note: Subclass-Of is transitive

Superclass Of

(<=> (Superclass-Of ?Csuper ?Csub)
(Subclass-Of ?Csub ?Csuper))

Example

Document
Book
Edited-Book
(Subclass-Of Edited-Book Book)
(Subclass-Of Book Document)

Class Frames and Template Slots

Class Frame

- Constant denoting a class (unary relation)

Template Slot

- A class frame has a set of template slots
- A template slot has a set of values at a class frame
- Describes own slot values at each instance of the class
- Semantics:
  - OKBC: Template slot S has value V at frame F
  - KIF: (Template-Slot-Value S F V)
- Template slot values inherit to subclasses and instances

  (=> (Template-Slot-Value ?S ?C ?V)
  (and (=> (Instance-Of ?I ?C) (holds ?S ?I ?V))
  (=> (Subclass-Of ?Csub ?C)
  (Template-Slot-Value ?S ?Csub ?V))))

Example Class Frame and Instance

Male-Person
*Gender: Male

Fred
Instance-Of: Male-Person
Gender: *Male*

(Note: "*" denotes template slot.
Italics denotes inferred value.)

KIF Translation

(Template-Slot-Value Gender Male-Person Male)

(Instance-Of Fred Male-Person)

Inference: (Gender Fred Male)

Template Facets

- A template slot at a class frame has a set of template facets
- A template facet at a template slot at a class frame has values
- Describes own facet values at each instance of the class
- Semantics:
  - OKBC: Template facet Fa has value V at template slot S at class frame Fr
  - KIF: (Template-Facet-Value Fa S Fr V)
- Template facet values inherit to subclasses and instances

```
(=> (Template-Facet-Value ?F ?S ?C ?V)
    (and (=> (Instance-Of ?I ?C) (holds ?F ?S ?I ?V))
         (=> (Subclass-Of ?Csub ?C)
             (Template-Facet-Value ?F ?S ?Csub ?V))))
```


Example Class Frame and Instance

Male-Person
*Gender: Male
Person
*Age:
Unit: Year

Fred
Instance-Of: Male
Gender: *Male*
Age:
Unit: *Year*

KIF Translation

(Template-Slot-Value Gender Male-Person Male)
(Template-Facet-Value Unit Age Male-Person Unit)
(Instance-Of Fred Male-Person)

Inferences

(Gender Fred Male)
(Unit Age Fred Year)

Value-Type Facet

- A type restriction on the values of a slot of a frame E.g.,

  Fred
  Favorite-Food: Broccoli Salsa
  Value-Type: Human-Food

- (=> (Value-Type ?S ?F ?C) (Class ?C)
  (=> (holds ?S ?F ?V) (Instance-Of ?V ?C))))
- (=> (Template-Facet-Value Value-Type ?S ?F ?C) (Class ?C)
  (=> (Template-Slot-Value ?S ?F ?V)
  (Instance-Of ?V ?C))))
- (= (Class-Of @i)
  (setofall (list ?x) (list-member ?x (listof @i))))

Example Class Frame and Instance

- Male Person
  - *Gender: Male
    - Value-Type: Gender
  - *Age
    - Unit: Year
    - Value-Type: Integer
  - *Parent:
    - Value-Type: Person

- Fred
  - Instance-Of: Male-Person
  - Gender: *Male*
    - Value-Type: *Gender*
  - Age:
    - Unit: *Year*
    - Value-Type: *Integer*
  - Parent: Earl
    - Value-Type: *Person*
- Earl
  - Instance-Of: *Person*

Example Value-Type Inferences

KIF Translation

(Template-Facet-Value Value-Type Gender Male-Person Gender)
(Template-Facet-Value Value-Type Age Male-Person Integer)
(Template-Facet-Value Value-Type Parent Male-Person Person)

Axioms

```
(=> (Template-Facet-Value Value-Type ?S ?F ?C) (Class ?C)
        (=> (Template-Slot-Value ?S ?F ?V) (Instance-Of ?V ?C))))
(=> (Value-Type ?S ?F ?C) (Class ?C)
        (=> (holds ?S ?F ?V) (Instance-Of ?V ?C))))
```

Example inferences

(Instance-Of Male Gender)
(=> (Template-Slot-Value Age Male-Person ?V) (Integer ?V))
(Instance-Of Person Earl)

Cardinality Facets

Specifies the number of values a slot may have at a frame

```
(=> (Slot-Cardinality ?S ?F ?N)
        (=  (Cardinality (setofall ?V (holds ?S ?F ?V)))  ?N))
(=> (Template-Facet-Value Slot-Cardinality ?S ?F ?N)
        (=< (Cardinality
                (setofall ?V (Template-Slot-Value ?S ?F ?V))
            ?N)))
```

Similar definitions for -

- Minimum-Slot-Cardinality
  At least that number of values
- Maximum-Slot-Cardinality
  At most that number of values

Example Class Frame and Instance

- Male Person
  - *Gender: Male
    - Value-Type: Gender
    - Slot-Cardinality: 1
  - *Age
    - Unit: Year
    - Value-Type: Integer
    - Slot-Cardinality: 1
  - *Parent:
    - Value-Type: Person
    - Slot-Cardinality: 2

- Fred
  - Instance-Of: Male-Person
  - Gender: *Male*
    - Value-Type: *Gender*
    - Slot-Cardinality: *1*
  - Age:
    - Unit: *Year*
    - Value-Type: *Integer*
    - Slot-Cardinality: *1*
  - Parent:
    - Value-Type: *Person*
    - Slot-Cardinality: *2*

Example Cardinality Inferences

KIF Translation

(Template-Facet-Value Slot-Cardinality Gender Male-Person 1)
(Template-Facet-Value Slot-Cardinality Age Male-Person 1)
(Template-Facet-Value Slot-Cardinality Parent Male-Person 2)

Axioms

(=> (Template-Facet-Value Slot-Cardinality ?S ?F ?C)
(=< (Cardinality (setofall ?V (Template-Slot-Value ?S ?F ?V)) ?N)))
(=> (Slot-Cardinality ?S ?F ?N)
(= (Cardinality (setofall ?V (holds ?S ?F ?V))) ?N))

Example inferences

(= (Cardinality (setofall ?V (holds Parent Fred ?V))) 2)
(=> (Slot-Cardinality ?S ?F 1) (holds ?S ?F ?X) (holds ?S ?F ?Y)
(= ?X ?Y))

Inverse Facet

- Specifies slots that are inverse relations of a slot S relative to the values of S at a specific frame.

E.g.: Person
*Parent:
Inverse: Child

- Axiom
  - (=> (Inverse ?S1 ?F ?S2)
    (and (Slot ?S2)
    (=> (holds ?S1 ?F ?V) (holds ?S2 ?V ?F))))

Example Class Frame and Instance

- Male-Person
  *Gender: Male
  Value Type: Gender
  Slot Cardinality: 1
  *Age
  Unit: Year
  Value Type: Integer
  Slot Cardinality: 1
  *Parent:
  Value Type: Person
  Slot Cardinality: 2
  Inverse: Child

- Fred
  Instance Of: Male Person
  Gender: *Male*
  Value Type: *Gender*
  Slot Cardinality: *1*
  Age:
  Unit: *Year*
  Value Type: *Integer*
  Slot Cardinality: *1*
  Parent: Earl
  Value Type: *Person*
  Slot Cardinality: *2*
  Inverse: *Child*
- Earl
  Child: *Fred*

Slot Chains

Slot Chains

(<=> (Slot-Chain-Value (listof ?S1 ?S2 @Sn) ?F ?Vn)
(exists ?V1 (and (holds ?S1 ?F ?V1)
(Slot-Chain-Value (listof ?S2 @Sn) ?V1 ?Vn))))
(<=> (Slot-Chain-Value (listof ?S) ?F ?V) (holds ?S ?F ?V))

Subset-Of-Values

(=> (Subset-Of-Values ?S1 ?F ?S2)
(=> (holds ?S1 ?F ?V) (holds ?S2 ?F ?V)))
(=> (Subset-Of-Values ?S ?F (listof @Sn))
(=> (holds ?S ?F ?V) (Slot-Chain-Value (listof @Sn) ?F ?V)))

Example
Person
*Child
Subset-Of-Values: (Parent Grandchild)

Same-Values (analogous to Subset-Of-Values)

Example Class Frame and Instance

- Male Person
  *Gender: Male
    Value Type: Gender
    Slot Cardinality: 1
  *Age
    Unit: Year
    Value Type: Integer
    Slot Cardinality: 1
  *Parent:
    Value Type: Person
    Slot Cardinality: 2
    Inverse: Child
    Subset Of Values:
      (Child Grandparent)

- Fred
  Instance Of: Male Person
  …
  Parent: Earl
    Value Type: *Person*
    Slot Cardinality: *2*
    Inverse: *Child*
    Subset Of Values:
      (Child Grandparent)
  Child: David
- Earl
  Child: *Fred*
- David
  Grandparent: *Earl*
  Parent: *Fred*

Decompositions

A decomposition is a set of subclasses of a class

E.g., {Adult Student} is a decomposition of People

```
(<=> (Decomposition ?c ?s)
     (and (class ?c) (set ?s) (=> (member ?x ?s) (Subclass-Of ?x ?c))))
```

E.g., Person
Decomposition: (set-of Adult Student)

Exhaustive decomposition

- All instances of the class are also instances of one of the subclasses in the decomposition
-
```
(<=> (Exhaustive-Decomposition ?c ?s)
     (and (Decomposition ?c ?s)
          (=> (Instance-Of ?x ?c)
              (exists ?sub (and (member ?sub ?s)
                                (Instance-Of ?x ?sub))))))
```

E.g., Person
Exhaustive-Decomposition: (set-of Adult Student Preschool-Child)

Decompositions

Disjoint decomposition

- An object cannot be an instance of more than one of the subclasses in the decomposition
- Axiom:
```
(<=> (Disjoint-Decomposition ?c ?s)
     (and (Decomposition ?c ?s)
          (=> (member ?x ?s) (member ?y ?s) (/= ?x ?y) (Instance-Of ?i ?x)
              (not (Instance-Of ?i ?y)))))
```

E.g., Person
Disjoint-Decomposition: (set-of Adult Preschool-Child)
Vehicle
Disjoint-Decomposition: (set-of Automobile Bicycle)

Partition

- An exhaustive disjoint decomposition
- Axiom:
```
(<=> (Partition ?c ?s)
     (and (Exhaustive-Decomposition ?c ?s) (Disjoint-Decomposition ?c ?s)))
```

E.g., Person
Disjoint-Decomposition: (set-of Adult Child)
Thing
Partition: (set-of Tangible-Object Intangible-Object)

Frame Based Representation Languages The End