Expert Systems
Knowledge Based Systems

Example Areas of Use

- Medical diagnosis
  - **Disease identification**

Example Areas of Use – 2

- Medical diagnosis
  - **Disease identification**
- Natural resource exploration
  - **Analyzing geological data**

Example Areas of Use

- ◊ Medical diagnosis
  - » **Disease identification**
- ◊ Natural resource exploration
  - » **Analyzing geological data**
- ◊ Customizing complex equipment
  - » **Computer systems**

Properties

- ◊ Behaves like an expert in a narrow area

Properties – 2

- Behaves like an expert in a narrow area
- Has a knowledge base of the information in the area

Properties – 3

◊ Behaves like an expert in a narrow area

◊ Has a knowledge base of the information in the area

◊ Ability to explain its behaviour

Properties – 4

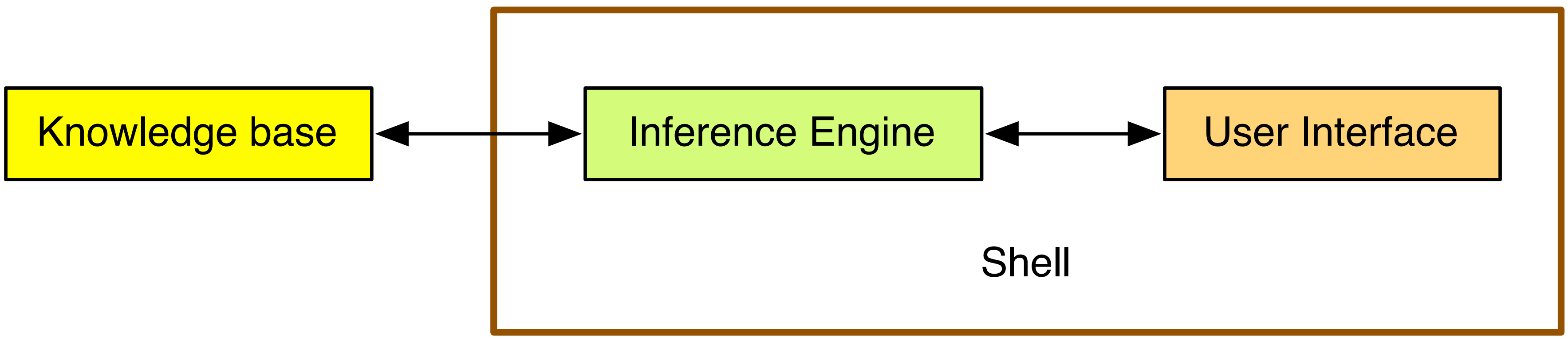
◊ Behaves like an expert in a narrow area

◊ Has a knowledge base of the information in the area

◊ Ability to explain its behaviour

◊ Ability to deal with uncertain data

Structure



If … then … else … rules

◊ The most popular form of knowledge representation

◊ Typical types of rules
  - » **If condition P holds then conclude C**
  - » **If situation S exists then do action A**
  - » **If conditions P and Q hold then conditions C1 and C2 cannot hold**

If … then … else … examples

◊ **See Figures 15.2, 15.3 & 15.4 in Bratko**

Kitchen leak example

◊ Figure 15.5 in Bratko

» **How do you read the graph?**

Kitchen leak example – 2

- ◊ Figure 15.5 in Bratko
  - » **Can see how if…then…else rules can represent the graph on the left hand side**
  - » **Note the use of AND / OR for inputs**
    - > **Arc represents AND of inputs**
    - > **No arc represents OR of inputs**

Properties of if…then…else rules

- Moduarity
  - » **Each rule or group of rules encapsulates a part of the domain**

Properties of if…then…else rules – 2

- Moduarity
  - **Each rule or group of rules encapsulates a part of the domain**
- Incrementabiity
  - **Add / delete rules as needed**

Properties of if…then…else rules – 3

- Moduarity
  - » **Each rule or group of rules encapsulates a part of the domain**
- Incrementabiity
  - » **Add / delete rules as needed**
- Modifiability
  - » **Can modify small parts of the knowledge as needed**

Properties of if…then…else rules – 4

- Moduarity
  - **Each rule or group of rules encapsulates a part of the domain**
- Incrementabiity
  - **Add / delete rules as needed**
- Modifiability
  - **Can modify small parts of the knowledge as needed**
- Supports transparency
  - **Relatively easy to explain and guide system's behaviour**

Probabilistic behaviour

- Can extend rule syntax to include probability information
  - **If condition A then**
    conclude C with probability P
  - **See Figure 15.3 in Bratko**

Inference Engine

- With if…then…else rules there are two ways of making inferences.
- What are they?

Inference Engine – 2

- With if…then…else rules there are two ways of making inferences.
- What are they?
  - Backward chaining
  - Forward chaining

Backward chaining

» **What is backward chaining?**

Backward chaining – 2

» What is backward chaining?

> The way Prolog works from conclusions to the base facts, the confirmed facts, the evidence

> See pages 348..349

Problems with backward chaining

- What are the problems with backward chaining?

Problems with backward chaining – 2

» **What are the problems with backward chaining?**

> **Syntax is not suitable of people unfamiliar with Prolog**

> **Cannot distinguish knowledge base from the rest of the system**

Problems with backward chaining – 3

- What are the problems with backward chaining?
  - Syntax is not suitable of people unfamiliar with Prolog
  - Cannot distinguish knowledge base from the rest of the system

- How can we overcome these problems?

Problems with backward chaining – 4

- What are the problems with backward chaining?
  - Syntax is not suitable of people unfamiliar with Prolog
  - Cannot distinguish knowledge base from the rest of the system
- How can we overcome these problems?
  - Customize the syntax with new operators
    - Bottom of page 349

Problems with backward chaining – 5

» **What do we need to do?**

Problems with backward chaining – 6

» **What do we need to do?**

> **Build an inference engine for the new rules**

– **Figure 15.6 and program text**

Forward chaining

- What is forward chaining?

Forward chaining – 2

- **What is forward chaining?**
  - **Work from the base facts, the confirmed facts, the evidence, to the conclusion**

Forward chaining – 3

- What is forward chaining?
  - Work from the base facts, the confirmed facts, the evidence, to the conclusion

- What do we need to do?

Forward chaining – 4

» **What is forward chaining?**

> **Work from the base facts, the confirmed facts, the evidence, to the conclusion**

» **What do we need to do?**

> **Build a forward chained inference engine**

– **Figure 15.7 and program text**

Forward vs backward chaining

◊ Abstract view

Backward chaining <---------

input information → … → derived information

-------> Forward chaining

See Figure 15.5

Forward vs backward chaining – 2

- More concrete views
  - **data → … → goals**
  - **evidence → … → hypotheses**
  - **findings, observations → … → explanations, diagnosis**
  - **manifestations → … → diagnoses, causes**

Which is better

- ◊ Forward chaining?
- ◊ Backward chaining?

Which is better

◊ Depends upon the problem

Which is better – 2

- Depends upon the problem
  - **Check if a hypothesis is true**
    - **Work backward**

Which is better – 3

◊ Depends upon the problem

» **Check if a hypothesis is true**

> **Work backward**

» **Many hypotheses, cannot choose**

> **Work forward**

Which is better – 3

◊ Depends upon the problem

» **Check if a hypothesis is true**

> **Work backward**

» **Many hypotheses, cannot choose**

> **Work forward**

◊ Forward is better when

» **Accumulating evidence, data**

Shape heuristic

- When input information is sparse relative to derived information
- Work forward or backward?

Shape heuristic – 2

- When input information is sparse relative to derived information
- Work forward or backward?
  - Use forward chaining

Shape heuristic – 3

» When input information is sparse relative to derived information

» Work forward or backward?

  > Use forward chaining

» When input information is rich relative to derived information

» Work forward or backward?

Shape heuristic – 5

- When input information is sparse relative to derived information
- Work forward or backward?
  - Use forward chaining

- When input information is rich relative to derived information
- Work forward or backward?
  - Use backward chaining

Reality

◊ Do we work forward or backward?

Reality – 2

- As tasks get more complex
  - **Better to interleave forward and backward chaining**

Reality Example

◊ Doctor uses initial observations, evidence to form hypothesis

Reality Example – 2

- ◊ Doctor uses initial observations, evidence to form hypothesis
  - » **Forward direction**

Reality Example – 3

- ◊ Doctor uses initial observations, evidence to form hypothesis
  - » **Forward direction**
- ◊ Pursues most likely hypothesis

Reality Example – 4

- Doctor uses initial observations, evidence to form hypothesis
  - **Forward direction**
- Pursues most likely hypothesis
  - **Backward direction to find if there is confirming evidence**

Reality Example – 5

- ◊ Doctor uses initial observations, evidence to form hypothesis
  - » **Forward direction**
- ◊ Pursues most likely hypothesis
  - » **Backward direction to find if there is confirming evidence**
- ◊ Can lead to gathering more evidence

Reality Example – 6

- ◊ Doctor uses initial observations, evidence to form hypothesis
  - » **Forward direction**
- ◊ Pursues most likely hypothesis
  - » **Backward direction to find if there is confirming evidence**
- ◊ Can lead to gathering more evidence, need new hypothesis
  - » **Forward direction, again**

Reality Example 2

- Top page 353

Generating explanations

- **There are two types of explanation**
  What are they?

Generating explanations

» There are two types of explanation
What are they?

> How

> Why

Generating explanations – how

- How did you find the answer?
  - **What do you present?**

Generating explanations – how – 2

- How did you find the answer?
  - **What do you present?**
    - **Typically present a path trace**

Generating explanations – how – 3

◊ How did you find the answer?

» **What do you present?**

> **Typically present a path trace**

» **There is a problem in the kitchen, which was concluded from the hall being wet and the bathroom dry**

> **And**

» **No water came from the outside, which was concluded from the window being closed**

Proof tree

- The how answer is to print out the proof tree
  - **Top page 354, Figure 15.8**
    - **Given program text**
    - **Compare with Figure 15.6**
      - **backward chained interpreter**

Proof tree – 2

- ◊ The how answer is to print out the proof tree
  - » **Top page 354, Figure 15.8**
    - > **Given program text**
    - > **Compare with Figure 15.6**
      - – **backward chained interpreter**
- ◊ The main work is printing the result in a readable format

Generating explanations – why

- The why explanation is required during the reasoning process
  - **What do you present?**

Generating explanations – why – 2

- The why explanation is required during the reasoning process
  - » **What do you present?**
    - > **The system asks the user for information**

Generating explanations – why – 3

- The why explanation is required during the reasoning process
  - **The system asks the user for information**
  - **The user may ask why**

Generating explanations – why – 4

◊ The why explanation is required during the reasoning process

» **The system asks the user for information**

» **The user may ask why**

» **The system then gives an explanation**

Generating explanations – why – 5

◊ The why explanation is required during the reasoning process

» **The system asks the user for information**

» **The user may ask why**

» **The system then gives an explanation**

> **Pages 354 .. 355**

Generating explanations – why – 5

◊ The why explanation is required during the reasoning process

» **The system asks the user for information**

» **The user may ask why**

» **The system then gives an explanation**

> **Pages 354 .. 355**

> **Figure 15.9 and program text**

On introducing uncertainty

◊ Chapter 15 introduces an ad hoc way of dealing with probabilities

On introducing uncertainty – 2

◊ Chapter 15 introduces an ad hoc way of dealing with probabilities

» **We will not look at these methods**

On introducing uncertainty – 3

◊ Chapter 15 introduces an ad hoc way of dealing with probabilities

» **We will not look at these methods**

» **We defer to Chapter 16 where we handle probabilities in a proper mathematical way**

On introducing uncertainty – 4

- ◊ Chapter 15 introduces an ad hoc way of dealing with probabilities
  - » **We will not look at these methods**
  - » **We defer to Chapter 16 where we handle probabilities in a proper mathematical way**
    - > **Use Bayesian networks**

On introducing uncertainty – 5

◊ Chapter 15 introduces an ad hoc way of dealing with probabilities

» **We will not look at these methods**

» **We defer to Chapter 16 where we handle probabilities in a proper mathematical way**

> **Use Bayesian networks**

> **A sound and correct way of dealing with probability and uncertainty**

On introducing uncertainty – 6

◊ Chapter 15 introduces an ad hoc way of dealing with probabilities

» **We will not look at these methods**

» **We defer to Chapter 16 where we handle probabilities in a proper mathematical way**

> **Use Bayesian networks**

> **A sound and correct way of dealing with probability and uncertainty**

> **Modern approach**

Semantic networks & frames

◊ Structure facts so as to elicit information

Semantic networks & frames – 2

- Structure facts so as to elicit information

- Introduce concepts that were adapted by object-oriented programming

Semantic networks & frames – 3

◊ Structure facts so as to elicit information

◊ Introduce concepts that were adapted by object-oriented programming

◊ Amounts to adopting a particular style of programming and organizing a program

Semantic networks & frames – 4

◊ Structure facts so as to elicit information

◊ Introduce concepts that were adapted by object-oriented programming

◊ Amounts to adopting a particular style of programming and organizing a program

  » **Requires discipline**

    > **The programming environment does not directly support the style**

Semantic networks

- Consist of
  - **Entities**

Semantic networks – 2

◊ Consist of

» **Entities**

» **Relations between Entities**

Semantic networks – 3

◊ Consist of

» **Entities**

» **Relations between Entities**

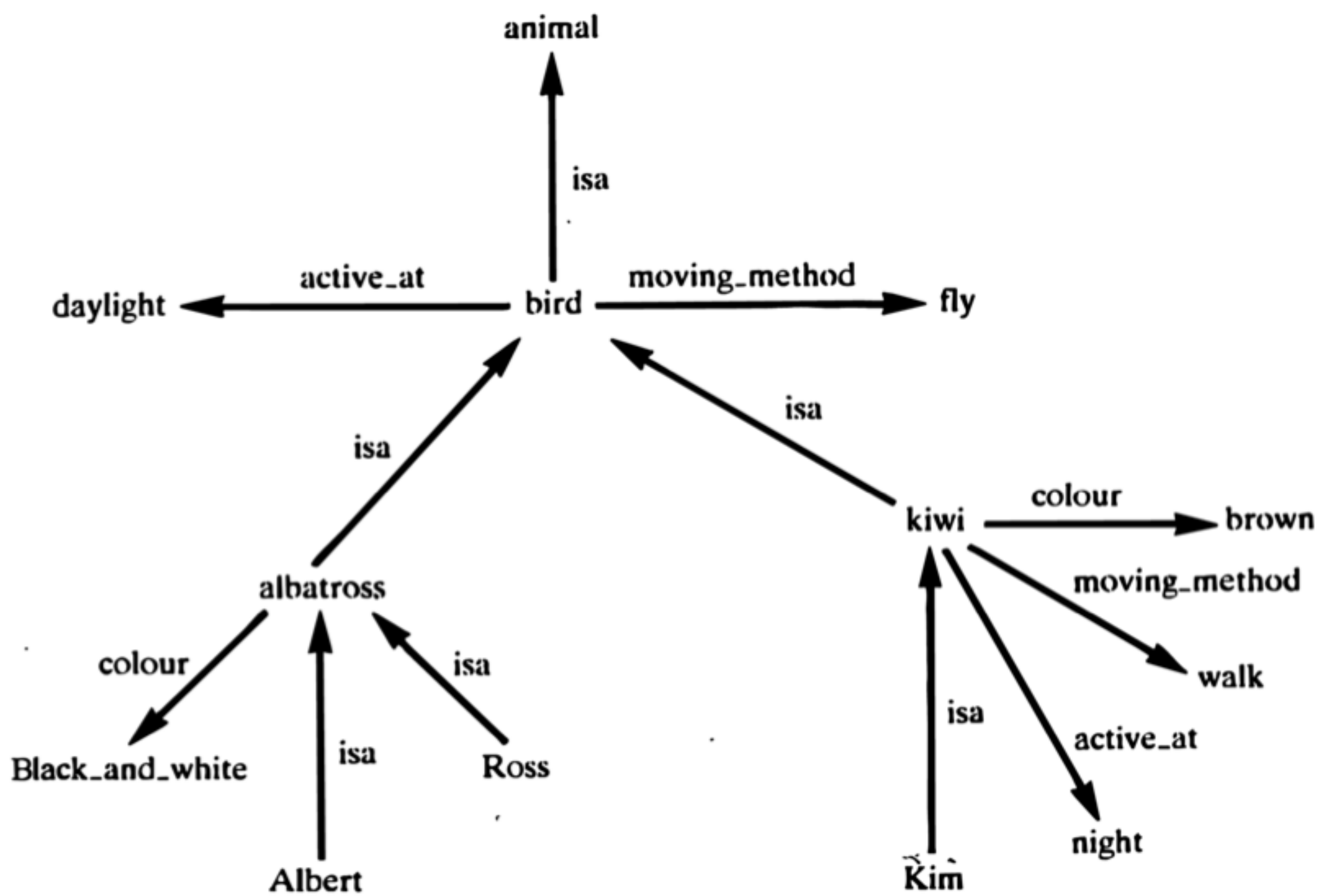
» **Some similarity with Entity-Relation model in databases**

Semantic networks – 4

- Consists of
  - **Entities**
  - **Relations between Entities**
  - **Some similarity with Entity-Relation model in databases**
  - **A graph representation is used**

Example semantic network

◊ Page 361 Bratko, E4



Semantic method representation

- The graph is represented in Prolog as a set of facts
  - » **Functor a relation name**
  - » **Arguments are the entities at the head and tail of a relation**

moving_method (kiwi , walk)
colour (kiwi , brown)
is_a (‘Kim’ , kiwi)
active_at (kiwi , night)

Inheritance representation

◊ Inheritance can be represented as a custom rule for each relationship

```
moving_method ( X , Method ) :-
        is_a ( X , SuperX ) ,
        moving_method ( SuperX , Method) .
```

Inheritance representation – 2

◊ Inheritance can be represented as a custom rule for each relationship

```
moving_method ( X , Method ) :-
        is_a ( X , SuperX ) ,
        moving_method ( SuperX , Method) .
```

◊ Cumbersome to use extensively

Inheritance representation – 3

- Inheritance can be represented as a custom rule for each relationship

```
moving_method ( X , Method ) :-
        is_a ( X , SuperX ) ,
        moving_method ( SuperX , Method) .
```

- Cumbersome to use extensively
  - **Need a separate rule for each relation**

Inheritance representation – 2

- ◊ A better way is to have a general rule for is_a based on facts
  - » **Argument to fact is a compound term**
    - > **relation_name (Arg1 , Arg2)**

```
fact ( Fact ) :- Fact , ! .

fact ( Fact ) :-
        Fact =.. [ Relation , Arg1 , Arg2 ] ,
        is_a ( Arg1 , SuperArg ) ,
        SuperFact =.. [ Relation , SuperArg , Arg2 ] ,
        fact ( SuperFact ) .
```

Frames

◊ A frame is data structure whose components are slots

Frames – 2

◊ A frame is data structure whose components are slots

◊ Slots have a name and a value

Frames – 3

◊ A frame is data structure whose components are slots

◊ Slots have a name and a value

FRAME: bird
a_kind_of: animal
moving_method: fly
active_at: daylight

Frames – 4

- ◊ A frame is data structure whose components are slots
- ◊ Slots have a name and a value
  - » **The value can be**
    - > **What?**

Frames – 5

◊ A frame is data structure whose components are slots

◊ Slots have a name and a value

  » **The value can be**

    > **Simple values**

Frames – 6

◊ A frame is data structure whose components are slots

◊ Slots have a name and a value

» **The value can be**

> **Simple values**

> **References to other frames**

Frames – 7

- A frame is data structure whose components are slots

- Slots have a name and a value
  - **The value can be**
    - **Simple values**
    - **References to other frames**
    - **Procedures to compute the slot value**

Frames – 8

- A frame is data structure whose components are slots

- Slots have a name and a value
  - **The value can be**
    - **Simple values**
    - **References to other frames**
    - **Procedures to compute the slot value**
    - **Unfilled**

Frames – 9

◊ A frame is data structure whose components are slots

◊ Slots have a name and a value

» **The value can be**

> **Simple values**

> **References to other frames**

> **Procedures to compute the slot value**

> **Unfilled**

– **How would they be filled?**

Frames – 10

- ◊ A frame is data structure whose components are slots

- ◊ Slots have a name and a value
  - » **The value can be**
    - > **Simple values**
    - > **References to other frames**
    - > **Procedures to compute the slot value**
    - > **Unfilled**
      - – **They are filled by inference**

Frame representation in Prolog

- The frame name is the functor for a set of predicates.
- The arguments of the predicate are
  - **The slot name**
  - **The slot value**

```
bird ( a_kind_of , animal ).
bird ( moving_method , fly ).
bird ( active_at , daylight ).
```

FRAME: bird

a_kind_of:	animal
moving_method:	fly
active_at:	daylight

Frame inheritance

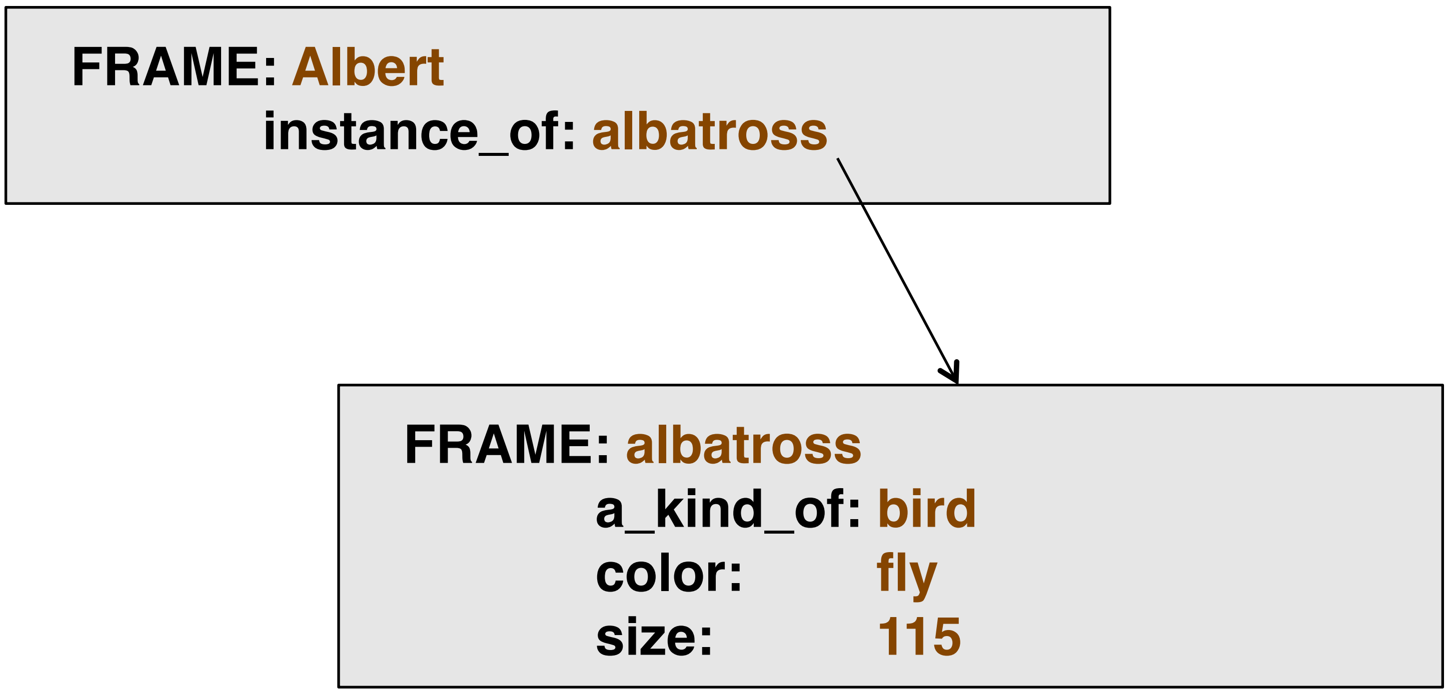
◊ Inheritance is shown by having a slot value being the name of a frame

```
FRAME: bird
        a_kind_of:          animal
        moving_method:      fly
        active_at:          daylight
```

```
FRAME: albatross
        a_kind_of:  bird
        color:      fly
        size:       115
```

Frame instance

- A frame instance references the frame of which it is an instance



Frame instance – 2

- ◊ A frame instance references a frame
- ◊ Can override slot values

FRAME: Albert
    instance_of: albatross
    size: 120

FRAME: albatross
    a_kind_of: bird
    color: fly
    size: 115

Frame example

◊ Figure 15.12

◊ Program text pages 365 .. 366