
Knowledge Representation

1

Physical Symbol System Hypothesis

- Assumption of (traditional) AI work is that:
- Knowledge may be represented as “symbol structures” (essentially, complex data structures) representing pieces of knowledge (objects, concepts, facts, rules, strategies..).
 - E.g., “red” represents color red.
 - “book1” represents my book.
 - red(book1) represents fact that my book is red.
- And intelligent behavior can be achieved through manipulation of symbol structures

2

Knowledge representation languages

- Knowledge representation languages have been designed to facilitate this.
- Rather than use general C++/Java data structures, use special purpose formalisms.
- A KR language should allow you to:
 - represent adequately the knowledge you need for your problem (representational adequacy)
 - do it in a clear, precise and “natural” way.
 - allow you to reason on that knowledge, drawing new conclusions.

3

Requirements for KR language

- Representational Adequacy
- Clear syntax/semantics
- Inferential adequacy
- Inferential efficiency
- Acquisitional Efficiency
- Naturalness
- In practice no one language is perfect, and different languages are suitable for different problems.

4

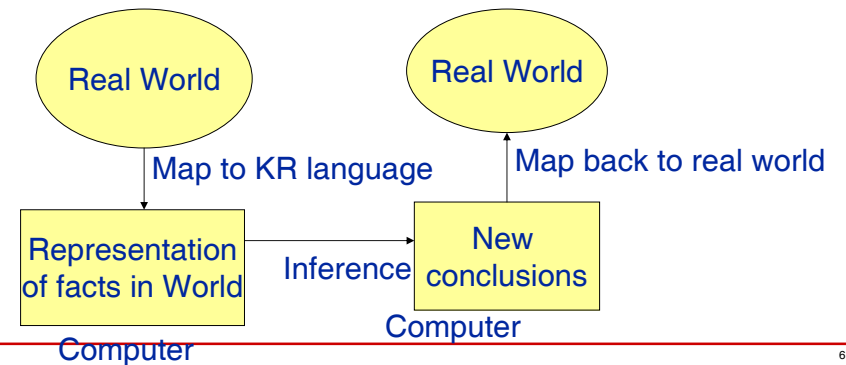
Representational adequacy

- Consider the following facts:
 - Ali believes nobody likes beans.
 - Ahmet will have to finish his assignment before he can start working on his project.
- Can all be represented as a string! But hard then to manipulate and draw conclusions.
- How do we represent these formally in a way that can be manipulated in a computer program?
- Some notations/languages only allow you to represent certain things.
 - Time, beliefs, uncertainty, all hard to represent.

5

Well-defined syntax/semantics

- Knowledge representation languages should have precise syntax and semantics.
- You must know exactly what an expression means in terms of objects in the real world.



6

Well defined syntax/semantics

- Suppose we have decided that “red1” refers to a dark red color, “car1” is my car, car2 is another..
- Syntax of language will tell you which of following is legal:
`red1(car1), red1 car1, car1(red1), red1(car1&car2)?`
- Semantics of language tells you exactly what an expression means - e.g., `Pred(Arg)` means that the property referred to by `Pred` applies to the object referred to by `Arg`. E.g., property “dark red” applies to my car.

7

What is a natural representation scheme?

- Our representation scheme should be intuitive and natural for human readers!
- Could represent the fact that **my book is red** using the notation:
 - “zzzzt -> thyv”
 - where zzzzt refers to redness, thyv refers to by book, and -> used in some way to assign properties.
- Hard to understand and manipulate

8

Inferential Adequacy

- Representing knowledge by itself is not very interesting unless you can use it to make inferences:
 - Draw new conclusions from existing facts.
 - “If its raining John never goes out” + “It’s raining today” so..
 - Come up with solutions to complex problems, using the represented knowledge.
- Inferential adequacy refers to how easy it is to draw inferences using represented knowledge.
- Representing everything as natural language strings has good representational adequacy and naturalness, but very poor inferential adequacy.

9

Inferential Efficiency

- You may be able, in principle, to make complex deductions given knowledge represented in a sophisticated language.
- But it may be just too inefficient.
- Generally the more complex the possible deductions, the less efficient will be the reasoner.
- Need representation and inference system sufficient for the task, without being hopelessly inefficient.

10

Acquisitional Efficiency

- The representation must also allow acquisition of new knowledge in a straightforward manner.

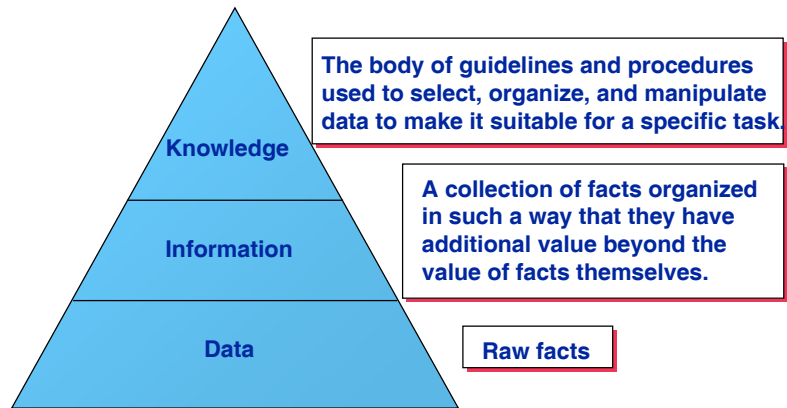
11

Knowledge Representation

- Knowledge vs. Data
- Different Knowledge Representation formalisms
 - Natural language
 - Database systems
 - Logic
 - Rules
 - Semantic nets
 - Frames

12

Information Concepts



13

Definition of Knowledge

- Knowledge is the symbolic representation of aspects of some named universe of discourse
- Universe of discourse may be the actual universe of fictional one, one in the future, or in some belief
- Examples of pieces of knowledge
 - Ali is an employee of NOY AŞ
 - All employees of NOY AŞ earn more than 100 M TL
 - All employees of NOY AŞ know that they should have a good life style.
 - Ali does not think that he has a good life style.
 - Everybody who knows that he should have a good life style is disappointed.

14

Definition of Data

- Data is the symbolic representation of simple aspects of some named universe of discourse.
- Data in this sense is a special case of knowledge.
- Examples of pieces of data:
 - Ayşe is married to Ali.
 - Ali works for NOY AŞ.
 - The average salary of NOY AŞ is 200M TL.

15

Issues in Knowledge Representation

- Are there any basic attributes of objects?
- Are there any basic relationships among objects?
- At what level should knowledge be represented?
- How should sets be represented?
- How should knowledge be accessed?

16

Knowledge Representation in Natural Language

- Expressiveness of natural languages
 - Very expressive, probably everything that can be expressed symbolically can be expressed in natural language(pictures, content of art, emotions are often hard to express)
 - Probably the most expressive knowledge representation formalism we have. Reasoning very complex, hard to model.
- Problems with natural language
 - Natural Language is often ambiguous.
 - Syntax and semantics are not fully understood.
 - There is little uniformity in the structure of sentences.

17

Knowledge Representation in Database Systems

Example:

```
person record = {   name : max 20 characters
                   age  : 3 digits in range 000-120
                   sex  : male or female
                   marital status : married, bachelor, spinster,
                                divorced, widowed, or engaged
                   first names of children : up to 10 names each max
                                           15 characters }
```

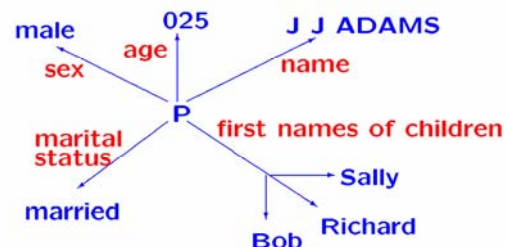
18

An Instance

A record structure:

J J ADAMS
025
male
married
Sally
Richard
Bob

or the same as **directed graph**:



Correct: "marital_status(J J ADAMS) is married"

Incorrect: "marital_status(J J ADAMS) is divorced"

19

Discussion of Database Systems

Example: Relations

R1: Person

J J ADAMS	025	male	married
P K BROWN	032	male	single
:	:	:	:

R2: Employment

XYZ	J J ADAMS
XYZ	P K BROWN
ABC	P K BROWN
:	:

R3: Parent/child

J J ADAMS	Sally
J J ADAMS	Richard
J J ADAMS	Bob
:	:

■ Properties of database systems:

- Only simple aspects of some universe of discourse. We can represent entities and relationships between entities, but not much more.
- Well-suited to efficiently represent and process large amounts of data. Reasoning=lookup.

20

Knowledge Representation in First-Order Logic

Examples: (a) $\text{man}(\text{Pat})$ (d) $\text{man}(\text{Jan}) \vee \text{woman}(\text{Jan})$
(b) $\text{married}(\text{Pat}, \text{Jan})$ (e) $\forall x \exists y [\text{person}(x) \rightarrow \text{has_mother}(x, y)]$
(c) $\forall x \forall y [[\text{married}(x, y) \wedge \text{man}(x)] \rightarrow \neg \text{man}(y)]$

Read:

logic	\vee	\wedge	\rightarrow	\neg	\forall	\exists
nat. lang.	or	and	implies	not	forall	exists

■ Properties of FOL

- Very expressive as well as unambiguous syntax and semantics
- No generally efficient procedure for processing knowledge

Correct:
 $\text{raining} \rightarrow \text{street_wet}$
 street_wet

Incorrect:
 $\text{raining} \rightarrow \text{street_wet}$
 elephant_hungry

21

LOGIC

- Logics are formal languages for representing information such that conclusions can be drawn
- Syntax defines the sentences in the language
- Semantics define the “meaning” of sentences; i.e., define truth of a sentence in a world, e.g., the language of arithmetic
 - $x + 2 > y$ is a sentence;
 - $x + 2 >$ is not a sentence
 - $x + 2 > y$ is true if the number $x + 2$ is no less than the number y
 - $x + 2 > y$ is true in a world where $x = 7$; $y = 1$
 - $x + 2 > y$ is false in a world where $x = 0$; $y = 6$

22

Propositions

- An **assertion** is a statement.
- A **proposition** is an assertion which is either true or false, but not both.
- Examples:
 - The moon is made of green cheese.
 - 3 is a prime number.
 - $x + y > 4$
 - Are you leaving?
 - Buy four of them.

23

Propositional Variable

- A **propositional variable** denotes an arbitrary proposition with an unspecified truth value.
- We will use the letters P, Q, R,... for propositional variables.

24

Logical Operators

- Propositions as well as propositional variables can be combined to form new assertions using words such as “and”, “or,” and “not”.
- Example:
 - Ahmet is 170 cm tall.
 - There are four cows in the barn.
 - Ahmet is 170 cm tall **and** there are four cows in the barn.
 - Ahmet is 170 cm tall **or** there are four cows in the barn.
 - Ahmet is **not** 170 cm tall.

25

Logical Operators

- P and Q.
- P or Q
- not P
- P and Q are called *operands*, and “and”, “or,” and “not” are called *logical operators* or *logical connectives*.

26

Propositional Form

- An assertion which contains at least one propositional variable is called *propositional form*.
- A propositional form is an expression whose truth value may not be determined until propositions are substituted for its propositional variables.
- When a logical operator is used to construct a new proposition from old ones the truth value of the new proposition depends on both the logical operator and the truth values of the original propositions.

27

Negation

- The logical operator “not” is denoted by the symbol “ \neg ” or “ \sim ”.
- If P is a proposition, then $\neg P$ is the negation of P.

P	$\neg P$
False	True
True	False

28

Conjunction

- The logical operator “and” is denoted by the symbol “ \wedge ”.
- If P and Q are propositions, then “ $P \wedge Q$ ” is a proposition called the conjunction of P and Q.

P	Q	$P \wedge Q$
0	0	0
0	1	0
1	0	0
1	1	1

29

Disjunction

- The logical operator “or” is denoted by the symbol “ \vee ”.
- “Logical or”, “inclusive or”
- If P and Q are propositions, then “ $P \vee Q$ ” is a proposition called the *disjunction* of P and Q.

P	Q	$P \vee Q$
0	0	0
0	1	1
1	0	1
1	1	1

30

Exclusive Or

- Exclusive operator is denoted by the symbol “ \oplus ”.

P	Q	$P \oplus Q$
0	0	0
0	1	1
1	0	1
1	1	0

31

Implication

- The logical operator “implies” is denoted by the symbol “ \Rightarrow ”.
- If P and Q are propositions, then “ $P \Rightarrow Q$ ” is a proposition called an implication.
- P is called the *premise*, *hypothesis* or *antecedent*.
- Q is called the *conclusion* or *consequence*.

P	Q	$P \Rightarrow Q$
0	0	1
0	1	1
1	0	0
1	1	1

32

Implication (Cont.)

- If P, then Q
- P only if Q
- P is a sufficient condition for Q
- Q is a necessary condition for P.
- Q if P.
- Q follows from P
- Q provided P
- Q is a logical consequence of P
- Q whenever P

33

Implication (Cont.)

- The converse of $P \Rightarrow Q$ is the proposition $Q \Rightarrow P$.
- The contrapositive $P \Rightarrow Q$ is the proposition $\neg Q \Rightarrow \neg P$.

34

Implication (Cont.)

CAUTION:

- The English language uses implication to assert a causal or inherent relationship between a premise and a conclusion.
- However, in the language of propositions, the premise of an implication need not be related to the conclusion in any way.
- Example
 - > Assume:
 - P represents “oranges are purple”
 - Q represents “the earth is not flat”
 - > $P \Rightarrow Q$ represents “If oranges are purple, then the earth is not flat.”

35

Equivalence

- If P and Q have the same truth values, then they are said to be logically equivalent propositions.
- The logical operator is called equivalence and is denoted by “ \Leftrightarrow ”.

P	Q	$P \Leftrightarrow Q$
0	0	1
0	1	0
1	0	0
1	1	1

36

Truth Tables

- Truth tables for individual operators can be used to construct truth tables for arbitrary complex propositional forms.
- The truth table for a propositional form specifies its truth value for every possible combination of truth values of its propositional variables.
- Each propositional variable can assume either of two values, true or false.
- Therefore, if k variables occur in a proposition, the associated truth table must describe 2^k cases.

37

Truth Table Construction Conventions

- All propositional variables occur in the leftmost column.
- Truth values assigned to the propositional variables by “counting in binary” from 0 to 2^k-1 , where k is the number of propositional variables.

38

Truth Table Example

- $[(P \wedge Q) \vee \neg R] \Leftrightarrow P$

P	Q	R	$P \wedge Q$	$\neg R$	$(P \wedge Q) \vee \neg R$	$[(P \wedge Q) \vee \neg R] \Leftrightarrow P$
0	0	0	0	1	1	0
0	0	1	0	0	0	1
0	1	0	0	1	1	0
0	1	1	0	0	0	1
1	0	0	0	1	1	1
1	0	1	0	0	0	0
1	1	0	1	1	1	1
1	1	1	1	0	1	1

39

Terminology

- A *tautology* is a propositional form whose truth value is true for all possible values of its propositional variables.
 - $P \vee \neg P$
- A *contradiction* or *absurdity* is a propositional form which is always false.
 - $P \wedge \neg P$
- A propositional form which is neither a tautology nor a contradiction is called a *contingency*.

40

Logical Identities

1. $P \Leftrightarrow (P \vee P)$ idempotence of \vee
2. $P \Leftrightarrow (P \wedge P)$ idempotence of \wedge
3. $(P \vee Q) \Leftrightarrow (Q \vee P)$ commutativity of \vee
4. $(P \wedge Q) \Leftrightarrow (Q \wedge P)$ commutativity of \wedge
5. $[(P \vee Q) \vee R] \Leftrightarrow [P \vee (Q \vee R)]$ associativity of \vee
6. $[(P \wedge Q) \wedge R] \Leftrightarrow [P \wedge (Q \wedge R)]$ associativity of \wedge
7. $\neg(P \vee Q) \Leftrightarrow (\neg P \wedge \neg Q)$ De Morgan's Laws
8. $\neg(P \wedge Q) \Leftrightarrow (\neg P \vee \neg Q)$ De Morgan's Laws

41

Logical Identities (Cont.)

9. $[P \wedge (Q \vee R)] \Leftrightarrow [(P \wedge Q) \vee (P \wedge R)]$ distributivity of \wedge over \vee
10. $[P \vee (Q \wedge R)] \Leftrightarrow [(P \vee Q) \wedge (P \vee R)]$ distributivity of \vee over \wedge
11. $(P \vee 1) \Leftrightarrow 1$
12. $(P \wedge 1) \Leftrightarrow P$
13. $(P \vee 0) \Leftrightarrow P$
14. $(P \wedge 0) \Leftrightarrow 0$
15. $(P \vee \neg P) \Leftrightarrow 1$
16. $(P \wedge \neg P) \Leftrightarrow 0$

42

Logical Identities (Cont.)

17. $P \Leftrightarrow \neg(\neg P)$ double negation
18. $(P \Rightarrow Q) \Leftrightarrow (\neg P \vee Q)$ implication
19. $(P \Leftrightarrow Q) \Leftrightarrow [(P \Rightarrow Q) \wedge (Q \Rightarrow P)]$ equivalence
20. $[(P \wedge Q) \Rightarrow R] \Leftrightarrow [P \Rightarrow (Q \Rightarrow R)]$ exportation
21. $[(P \Rightarrow Q) \wedge (P \Rightarrow \neg Q)] \Leftrightarrow \neg P$ absurdity
22. $(P \Rightarrow Q) \Leftrightarrow (\neg Q \Rightarrow \neg P)$ contrapositive

43

Example for Use of Identities

- Simplify $[(A \Rightarrow B) \vee (A \Rightarrow D)] \Rightarrow (B \vee D)$
- $[(\neg A \vee B) \vee (\neg A \vee D)] \Rightarrow (B \vee D)$ (18)
 - $[\neg A \vee (B \vee D)] \Rightarrow (B \vee D)$ (5, 3, 1)
 - $\neg[\neg A \vee (B \vee D)] \vee (B \vee D)$ (18)
 - $[A \wedge \neg(B \vee D)] \vee (B \vee D)$ (7, 17)
 - $(A \vee B \vee D) \wedge [\neg(B \vee D) \vee (B \vee D)]$ (3, 10)
 - $(A \vee B \vee D) \wedge 1$ (15)
 - $A \vee B \vee D$ (12)

44

Logical Implications

- | | |
|---|------------------------|
| 1. $P \Rightarrow (P \vee Q)$ | addition |
| 2. $(P \wedge Q) \Rightarrow P$ | simplification |
| 3. $[P \wedge (P \Rightarrow Q)] \Rightarrow Q$ | Modus Ponens |
| 4. $[(P \Rightarrow Q) \wedge \neg Q] \Rightarrow \neg P$ | Modus Tollens |
| 5. $[\neg P \wedge (P \vee Q)] \Rightarrow Q$ | disjunctive syllogism |
| 6. $[(P \Rightarrow Q) \wedge (Q \Rightarrow R)] \Rightarrow (P \Rightarrow R)$ | hypothetical syllogism |

45

PREDICATE CALCULUS

46

Predicates

- He is tall and blonde (x is tall and blonde).
- She lives in the city (x lives in y)
- These assertions are formed using variables in a “template” which expresses a property of an object or a relationship between objects.
- These templates are called **predicates**.
- Examples
 - “x + y = z” can be denoted S(x, y, z)
 - “x is a female” can be denoted female(x)
 - “x is married to y” can be denoted married(x,y)

47

Predicate Definitions

- **Predicate constant:** The symbol which denotes a specific predicate.
 - married()
- **Individual variable:** A variable which appears in the parenthesized list after a predicate.
 - x
- Consider $P(x_1, x_2, \dots, x_n)$
 - P: predicate constant
 - x_1, x_2, \dots, x_n individual variables
 - P has n arguments,
 - arity of P is n,
 - P is an n-place predicate

48

Universe of Discourse

- Values of the individual variables must be drawn from a set called the **universe of discourse**.
- If P is an n -place constant and values c_1, c_2, \dots, c_n are assigned to each of the individual variables, the result is a proposition.

49

- Suppose the universe of discourse is U .
- If the value of $P(c_1, c_2, \dots, c_n)$ is true for every choice of arguments c_1, c_2, \dots, c_n selected from U , then P is said to be **valid in the universe U** .
- If the value of $P(c_1, c_2, \dots, c_n)$ is true for some (but not necessarily all) choices of arguments selected from U , then P is said to be **satisfiable in the universe U** , and the values c_1, c_2, \dots, c_n which make $P(c_1, c_2, \dots, c_n)$ true are said to satisfy P .
- If P is not satisfiable in the universe U , then it is called **unsatisfiable in U** .

50

Predicates and Propositions

- A predicate constant with no arguments is a proposition.
- In order to change a predicate into a proposition, each individual variable of the predicate must be bound.
- Binding can be done in two ways:
 - By assigning each individual variable a value
 - By quantification of the variables

51

Assigning Values to Variables

- Consider $P(x,y)$ to be " $x+y=3$ "
- $P(1,2)$ $1+2=3$ is true
- $P(2,6)$ $2+6=3$ is false

52

Quantifiers

- Universal
- Existential
- Uniqueness

53

Universal Quantifier

- If $P(x)$ is a predicate with the individual variable x as an argument, then the assertion
“For all x , $P(x)$ ”
- which is interpreted as
“For all values of x , the assertion $P(x)$ is true,”
- is a statement in which the variable x is said to be universally quantified.
- The symbol \forall called the universal quantifier, is used to denote the phrase “for all”.
- $\forall x P(x) \Rightarrow P(c)$

54

Universal Quantification Examples

- Let $U=Z$
 - $\forall x [x < x+1]$
 - $\forall x [x = 3]$
- Let $U=Z$
- $\forall x \forall y [x + y > x]$
- What if $U=Z^+$?

55

Existential Quantifier

- If $P(x)$ is a predicate with the individual variable x as an argument, then the assertion
“For some x , $P(x)$ ”
- which is interpreted as
“There exists a value of x , for which the assertion $P(x)$ is true,”
- is a statement in which the variable x is said to be existentially quantified.
- The symbol \exists called the existential quantifier, is used to denote the phrase “there exists”.
- $P(c) \Rightarrow \exists x P(x)$

56

Existential Quantification Examples

- Let $U=Z$
 - $\exists x [x < x+1]$
 - $\exists x [x = 3]$
 - $\exists x [x = x+1]$

57

Uniqueness Quantifier

- There is one and only one element of the universe of discourse which makes a predicate true.
- The symbol $\exists!$ called the uniqueness quantifier, is used to denote the phrase “there exists a unique”.

58

Uniqueness Quantification Examples

- Let $U=Z$
 - $\exists! x [x < x+1]$
 - $\exists! x [x = 3]$
 - $\exists! x [x = x+1]$

59

Relationship between quantified variables and propositions

- Let $U=\{1, 2, 3\}$
- $\forall x P(x) \Leftrightarrow P(1) \wedge P(2) \wedge P(3)$
- $\exists x P(x) \Leftrightarrow P(1) \vee P(2) \vee P(3)$
- $\exists! x P(x) \Leftrightarrow [P(1) \wedge \neg P(2) \wedge \neg P(3)] \vee [\neg P(1) \wedge P(2) \wedge \neg P(3)] \vee [\neg P(1) \wedge \neg P(2) \wedge P(3)]$

60

Interaction of Predicates, Operators and Quantifiers

- An assertion involving predicate variables is *valid* if it is true for every universe of discourse no matter how the predicate variables are interpreted.
- An assertion is *satisfiable* if there exists a universe and some interpretation of predicate variables which makes it true.
- If an assertion is not true for any universe or interpretation, it is *unsatisfiable*.
- Valid, satisfiable and unsatisfiable assertions are the analogs of tautologies, contingencies, and contradictions in the language of propositions.

61

Equivalent Assertions

- Two assertions A_1 and A_2 are said to be logically equivalent if and only if for every universe of discourse and every interpretation of the predicate variable, A_1 is true if and only if A_2 is true.

62

Negation and Quantifiers

- $\neg \forall x P(x) \Leftrightarrow \exists x \neg P(x)$
- $\neg \exists x P(x) \Leftrightarrow \forall x \neg P(x)$
- Example:
- $\neg \exists x \forall y \forall z P(x, y, z) \Leftrightarrow \forall x \neg \forall y \forall z P(x, y, z)$
 $\Leftrightarrow \forall x \exists y \neg \forall z P(x, y, z)$
 $\Leftrightarrow \forall x \exists y \exists z \neg P(x, y, z)$

63

Scope of a quantifier

- The scope of a quantifier is the part of an assertion in which variables are bound by the quantifier.
- Examples:
 - $A \vee \forall x [P(x) \vee Q(x)]$
 - $[\forall x P(x)] \Rightarrow [\exists x Q(x)]$
 - $\forall x P(x) \vee Q(x)$

64

Quantifiers and Disjunctions and Conjunctions

- $\forall x [A(x) \vee P] \Leftrightarrow [\forall x A(x) \vee P]$
- $\forall x [A(x) \wedge P] \Leftrightarrow [\forall x A(x) \wedge P]$
- $\exists x [A(x) \vee P] \Leftrightarrow [\exists x A(x) \vee P]$
- $\exists x [A(x) \wedge P] \Leftrightarrow [\exists x A(x) \wedge P]$

- $\forall x [P(x) \vee Q(y)] \Leftrightarrow [\forall x P(x) \vee Q(y)]$

65

Distribution over \wedge

- Universal quantifier distributes over \wedge .
 - $\forall x [P(x) \wedge Q(x)] \Leftrightarrow \forall x P(x) \wedge \forall x Q(x)$
- Existential quantifier does not distribute over \wedge .

66

Distribution over \vee

- Existential quantifier distributes over \vee .
 - $\exists x [P(x) \vee Q(x)] \Leftrightarrow \exists x P(x) \vee \exists x Q(x)$
- Universal quantifier does not distribute over \vee .

67

LOGICAL INFERENCE

68

Definitions

- A **theorem** is a mathematical assertion which can be shown to be true.
- A **proof** is an argument which establishes the truth of a theorem.
- **Rules of inference** specify conclusions which can be drawn from assertions known or assumed to be true.

69

Rules of Inference

Rule of Inference	Tautological Form	Name
$\frac{}{P}$ $\therefore P \vee Q$	$P \Rightarrow (P \vee Q)$	addition
$\frac{P \wedge Q}{P}$	$(P \wedge Q) \Rightarrow P$	simplification
$\frac{P \quad P \Rightarrow Q}{Q}$	$[P \wedge (P \Rightarrow Q)] \Rightarrow Q$	Modus ponens
$\frac{\neg Q \quad P \Rightarrow Q}{\therefore \neg P}$	$[\neg Q \wedge (P \Rightarrow Q)] \Rightarrow \neg P$	Modus tollens

70

Rules of Inference (Cont.)

Rule of Inference	Tautological Form	Name
$\frac{P \vee Q \quad \neg P}{\therefore Q}$	$[(P \vee Q) \wedge \neg P] \Rightarrow Q$	Disjunctive syllogism
$\frac{P \Rightarrow Q \quad Q \Rightarrow R}{\therefore P \Rightarrow R}$	$[(P \Rightarrow Q) \wedge (Q \Rightarrow R)] \Rightarrow (P \Rightarrow R)$	Hypothetical syllogism
$\frac{P \quad Q}{\therefore P \wedge Q}$	$[P \wedge Q] \Rightarrow (P \wedge Q)$	conjunction

71

Common Fallacies

- The Fallacy of Affirming the Consequent
 - If the butler did it, he will be nervous when interrogated.
 - The butler was very nervous when he was interrogated.
 - Therefore, the butler did it.
- $$\frac{Q \quad P \Rightarrow Q}{\therefore P}$$

72

Common Fallacies (Cont.)

■ The Fallacy of Denying the Antecedent

- If the butler's hands are covered with blood, then he did it.
- The butler is impeccably groomed.
- Therefore, the butler is innocent.

$\neg P$

$P \Rightarrow Q$

$\therefore \neg Q$

73

Proof Example

Theorem:

If horses fly or cows eat artichokes, then the mosquito is the national bird. If the mosquito is the national bird, then peanut butter tastes good on hot dogs. But peanut butter tastes terrible on hot dogs. Therefore, cows don't eat artichokes.

Let

$F \Leftrightarrow$ "horses fly"

$A \Leftrightarrow$ "cows eat artichokes"

$M \Leftrightarrow$ "the mosquito is the national bird"

$P \Leftrightarrow$ "peanut butter tastes good on hot dogs"

$(F \vee A) \Rightarrow M$

$M \Rightarrow P$

$\neg P$

$\therefore \neg A$

74

Proof Example (Cont.)

■ Proof:

Assertion	Reasons
1. $(F \vee A) \Rightarrow M$	Hypothesis 1
2. $M \Rightarrow P$	Hypothesis 2
3. $(F \vee A) \Rightarrow P$	Steps 1 and 2 and hypothetical syllogism
4. $\neg P$	Hypothesis 3
5. $\neg (F \vee A)$	Steps 3 and 4 and modus tollens
6. $\neg F \wedge \neg A$	Step 5 and DeMorgan's law
7. $\neg A \wedge \neg F$	Step 6 and commutativity of \wedge
8. $\neg A$	Step 7 and simplification

75

Rules of Inference concerning Quantifiers

■ Universal Instantiation

$\forall x P(x)$

$\therefore P(c)$

■ Universal Generalization

$P(x)$

$\therefore \forall x P(x)$

■ Existential Instantiation

$\exists x P(x)$

$\therefore P(c)$

c not arbitrary!

■ Existential Generalization

$P(c)$

$\therefore \exists x P(x)$

76

Resolution

- **Resolution Principle:** If there is an axiom of the form $E_1 \vee E_2$, and there is another axiom of the form $\neg E_2 \vee E_3$, then $E_1 \vee E_3$ logically follows.
- $E_1 \vee E_3$ is called the **resolvent**.
- Resolution can be generalized to any number of disjuncts.

77

Proof by Refutation

- Assume that the negation of the theorem is TRUE.
- Show that the axioms and the assumed negation of the theorem together determine something to be TRUE that cannot be TRUE.
- Conclude that the assumed negation of the theorem cannot be TRUE since it leads to contradiction.
- Conclude that the theorem must be TRUE since the assumed negation of the theorem cannot be TRUE.

78

Resolution Example

Theorem:

Feathers(Cikcik)

$\forall x[\text{Feathers}(x) \Rightarrow \text{Bird}(x)]$

$\therefore \text{Bird}(\text{Cikcik})$

Proof:

Feathers(Cikcik) (Axiom 1)

$\neg \text{Feathers}(\text{Cikcik}) \vee \text{Bird}(\text{Cikcik})$ (axiom 2 & u.g.)

$\neg \text{Bird}(\text{Cikcik})$ (negation of conclusion)

$\text{Bird}(\text{Cikcik})$ (Resolve 1 and 2)

Nil (Resolve 3 and 4 \Rightarrow Contradiction)

79

Conversion to Clause Form

Clause form: disjunction of literals

Given

- The brick is on something that is not a pyramid.
- There is nothing that the brick is on and that is on the brick as well.
- There is nothing that is not a brick and the same thing as a brick.

$\forall x[\text{Brick}(x) \Rightarrow (\exists y[\text{On}(x,y) \wedge \neg \text{Pyramid}(y)])$

$\wedge \neg \exists y[\text{On}(x,y) \wedge \text{On}(y,x)]$

$\wedge \forall y[\neg \text{Brick}(y) \Rightarrow \neg \text{Equal}(x,y)]]$

80

Eliminate Implications

(Substitute $\neg E1 \vee E2$ for $E1 \Rightarrow E2$).

$$\begin{aligned} \forall x [\neg \text{Brick}(x) \vee (\exists y [\text{On}(x,y) \wedge \neg \text{Pyramid}(y)] \\ \wedge \neg \exists y [\text{On}(x,y) \wedge \text{On}(y,x)] \\ \wedge \forall y [\neg(\neg \text{Brick}(y)) \vee \neg \text{Equal}(x,y)])] \end{aligned}$$

81

Move negations down to atomic formulas

(Use De Morgan's laws and negation of quantifiers)

$$\begin{aligned} \forall x [\neg \text{Brick}(x) \vee (\exists y [\text{On}(x,y) \wedge \neg \text{Pyramid}(y)] \\ \wedge \forall y [\neg \text{On}(x,y) \vee \neg \text{On}(y,x)] \\ \wedge \forall y [\text{Brick}(y) \vee \neg \text{Equal}(x,y)])] \end{aligned}$$

82

Purge existential quantifiers

Use skolem functions.

$$\begin{aligned} \blacksquare \exists y [\text{On}(x,y) \wedge \neg \text{Pyramid}(y)] \\ \blacksquare y = \text{Magic}(x) \\ \blacksquare \text{On}(x, \text{Magic}(x)) \wedge \neg \text{Pyramid}(\text{Magic}(x)) \\ \blacksquare \text{Magic}(x) = \text{Support}(x) \\ \forall x [\neg \text{Brick}(x) \vee (\text{On}(x, \text{Support}(x)) \\ \wedge \neg \text{Pyramid}(\text{Support}(x))) \\ \wedge \forall y [\neg \text{On}(x,y) \vee \neg \text{On}(y,x)] \\ \wedge \forall y [\text{Brick}(y) \vee \neg \text{Equal}(x,y)]] \end{aligned}$$

83

Rename Variables

- (No two variables should be the same. Scope of universal quantifiers!)

$$\begin{aligned} \forall x [\neg \text{Brick}(x) \vee ((\text{On}(x, \text{Support}(x)) \\ \wedge \neg \text{Pyramid}(\text{Support}(x))) \\ \wedge \forall y [\neg \text{On}(x,y) \vee \neg \text{On}(y,x)] \\ \wedge \forall z [\text{Brick}(z) \vee \neg \text{Equal}(x, z)])] \end{aligned}$$

84

Move Universal Quantifiers to the left

$\forall x \forall y \forall z [$
 $\neg \text{Brick}(x) \vee ((\text{On}(x, \text{Support}(x)) \wedge \neg \text{Pyramid}(\text{Support}(x)))$
 $\wedge \neg \text{On}(x, y) \vee \neg \text{On}(y, x))$
 $\wedge \text{Brick}(z) \vee \neg \text{Equal}(x, z)]$

85

Move the disjunctions down to the literals

■ Use distribution laws

$\forall x \forall y \forall z [$
 $(\neg \text{Brick}(x) \vee (\text{On}(x, \text{Support}(x)) \wedge \neg \text{Pyramid}(\text{Support}(x))))$
 $\wedge (\neg \text{Brick}(x) \vee \neg \text{On}(x, y) \vee \neg \text{On}(y, x))$
 $\wedge (\neg \text{Brick}(x) \vee \text{Brick}(z) \vee \neg \text{Equal}(x, z))]$

86

Move the disjunctions down to the literals (Cont.)

$\forall x \forall y \forall z [$
 $(\neg \text{Brick}(x) \vee \text{On}(x, \text{Support}(x))$
 $\wedge (\neg \text{Brick}(x) \vee \neg \text{Pyramid}(\text{Support}(x)))$
 $\wedge (\neg \text{Brick}(x) \vee \neg \text{On}(x, y) \vee \neg \text{On}(y, x))$
 $\wedge (\neg \text{Brick}(x) \vee \text{Brick}(z) \vee \neg \text{Equal}(x, z))]$

87

Eliminate the conjunctions

■ Write each part of the conjunction as a separate axiom.

$\forall x [\neg \text{Brick}(x) \vee \text{On}(x, \text{Support}(x))]$
 $\forall x [\neg \text{Brick}(x) \vee \neg \text{Pyramid}(\text{Support}(x))]$
 $\forall x \forall y [\neg \text{Brick}(x) \vee \neg \text{On}(x, y) \vee \neg \text{On}(y, x)]$
 $\forall x \forall z [\neg \text{Brick}(x) \vee \text{Brick}(z) \vee \neg \text{Equal}(x, z)]$

88

Rename all the variables

$\forall x [\neg \text{Brick}(x) \vee \text{On}(x, \text{Support}(x))]$
 $\forall w [\neg \text{Brick}(w) \vee \neg \text{Pyramid}(\text{Support}(w))]$
 $\forall u \forall y [\neg \text{Brick}(u) \vee \neg \text{On}(u, y) \vee \neg \text{On}(y, u)]$
 $\forall v \forall z [\neg \text{Brick}(v) \vee \text{Brick}(z) \vee \neg \text{Equal}(v, z)]$

89

Purge the universal quantifiers

$\neg \text{Brick}(x) \vee \text{On}(x, \text{Support}(x))$
 $\neg \text{Brick}(w) \vee \neg \text{Pyramid}(\text{Support}(w))$
 $\neg \text{Brick}(u) \vee \neg \text{On}(u, y) \vee \neg \text{On}(y, u)$
 $\neg \text{Brick}(v) \vee \text{Brick}(z) \vee \neg \text{Equal}(v, z)$

90

To put Axioms into Clause Form

- Eliminate implications
- Move negations down to the atomic formulas.
- Purge existential quantifiers.
- Rename variables, if necessary.
- Move the universal quantifiers to the left.
- Move the disjunctions down to the literals.
- Eliminate the conjunctions.
- Rename variables, if necessary.
- Purge the universal quantifiers.

91

Procedure for doing Resolution Proof

- Negate the theorem to be proved and add the result to the list of axioms.
- Put the list of axioms into clause form.
- Until the empty clause, Nil, is produced or there is no resolvable pair of clauses, find the resolvable clauses, resolve them, and add the result to the list of clauses.
- If the empty clause is produced, report that the theorem is TRUE. If there are no resolvable clauses, report that the theorem is false.

92

Unification

- A variable may be replaced by a constant. $\{v1 \rightarrow C\}$
- A variable may be replaced by a variable. $\{v2 \rightarrow v3\}$
- A variable may be replaced by a function expression as long as the function expression does not contain that variable. $\{v4 \rightarrow f(\dots)\}$

93

Resolution Proof Example

On(B,A)
On(A, Table)
 \therefore Above(B, Table)

94

Proof

- Additional axioms
 $\forall x \forall y [\text{On}(x,y) \Rightarrow \text{Above}(x,y)]$
 $\forall x \forall y \forall z [\text{Above}(x,y) \wedge \text{Above}(y,z) \Rightarrow \text{Above}(x,z)]$
- Clause form:
 $\neg \text{On}(u,v) \vee \text{Above}(u,v)$
 $\neg \text{Above}(x,y) \vee \neg \text{Above}(y,z) \vee \text{Above}(x,z)$

95

- $\neg \text{On}(u,v) \vee \text{Above}(u,v)$ (1)
- $\neg \text{Above}(x,y) \vee \neg \text{Above}(y,z) \vee \text{Above}(x,z)$ (2)
- On(B,A) (3)
- On(A, Table) (4)
- $\neg \text{Above}(B, \text{Table})$ (5)

96

-
- Resolve (2) and (5) by specializing x to B and z to Table.

$\neg \text{Above}(B, y) \vee \neg \text{Above}(y, \text{Table}) \vee \text{Above}(B, \text{Table})$ (2)

$\neg \text{Above}(B, \text{Table})$ (5)

$\neg \text{Above}(B, y) \vee \neg \text{Above}(y, \text{Table})$ (6)

97

-
- Resolve (1) with (6) replacing u with y and specializing v to Table.

$\neg \text{On}(y, \text{Table}) \vee \text{Above}(y, \text{Table})$ (1)

$\neg \text{Above}(B, y) \vee \neg \text{Above}(y, \text{Table})$ (6)

$\neg \text{On}(y, \text{Table}) \vee \neg \text{Above}(B, y)$ (7)

98

-
- Resolve (1) and (7) with u specialized to B and v replaced by y.

$\neg \text{On}(B, y) \vee \text{Above}(B, y)$ (1)

$\neg \text{On}(y, \text{Table}) \vee \neg \text{Above}(B, y)$ (7)

$\neg \text{On}(B, y) \vee \neg \text{On}(y, \text{Table})$ (8)

99

-
- Resolve (3) and (8), specializing y to A

$\text{On}(B, A)$ (3)

$\neg \text{On}(B, A) \vee \neg \text{On}(A, \text{Table})$ (8)

$\neg \text{On}(A, \text{Table})$ (9)

100

How to select the right clauses to resolve?

- Confine to resolving against only the negated theorem or new clauses derived, directly or indirectly, using the negated theorem.
- Use intuition (?)

102

- Now (4) and (9) resolve to Nil

On(A, Table) (4)

¬On(A, Table) (9)

Nil (10)

101

Search Strategies

- **Unit preference strategy:** gives preference to resolutions involving the clauses with the smallest number of literals.
- **Set of support strategy:** allows only resolutions involving the negated theorem or theorem or new clauses derived, directly or indirectly, using the negated theorem.
- **Breadth-first strategy:** first resolves all possible pairs of the initial clauses, then resolves all possible pairs of the resulting set together with the initial set, level by level.

103

Properties of these strategies

- All these strategies are complete because they are guaranteed to find a proof if the theorem logically follows from the axioms.
- All strategies are subject to the combinatorial explosion problem.
- All search strategies are subject to a version of the halting problem, for search is not guaranteed to terminate unless there actually is a proof.

104

- All complete proof procedures of the first-order predicate calculus are subject to the halting problem.
- Gödel's Completeness Theorem.
- If a sentence is **true** given a set of axioms, there is a procedure that will determine this.
- If the sentence is **false**, then there is no guarantee that a procedure will ever determine this—i.e., it may never halt.
- Complete proof procedures are **semidecidable**.

105

Knowledge Representation in Knowledge-Based Systems

- Consists of
 - A knowledgebase for representing the expert knowledge
 - An inference engine

106

Rules – A Simple Example

Assume: **knowledge base** consisting of **facts** and **rules**, a **rule interpreter** to match the rule conditions against facts and means for executing the rules.

Rules:

R1:	IF: Raining, Outside(x), Has_Umbrella(x) THEN: Uses_Umbrella(x)
R2:	IF: Raining, Outside(x) NOT Has_Umbrella(x) THEN: Wet(x)
R3:	IF: Wet(x) THEN: Gets_Cold(x)
R4:	IF: Sunny, Outside(x) THEN: Gets_Sun_Tan(x)

Initial facts: Raining, Outside(john)

107

Rules – A Simple Example (Contd)

Correct:

Only one rule, R2 matches the facts with $[x \mapsto \text{john}]$, hence add **Wet(john)**

Facts after first cycle:

Raining, Outside(john), Wet(john)

Now R3 matches the facts, hence add

Gets_Cold(john)

Facts after second cycle:

Raining, Outside(john), Wet(john), Gets_Cold(john)

Incorrect:

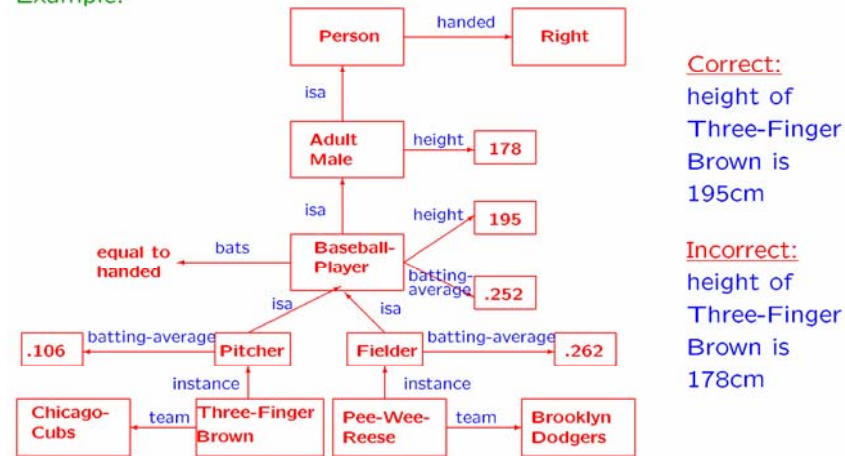
Gets_Sun_Tan(john)

Process of deriving new facts from given facts, is called **Inference**

108

Knowledge Representation in Semantic Nets

Example:



109

Semantic Networks

- Semantic networks are basically graphic depictions of knowledge that show hierarchical relationships between objects.
- A semantic network, or semantic net, is composed of nodes and links.
- One of the most interesting and useful fact's about a semantic network is that it can show inheritance.

110

Semantic Nets

- Properties
 - Allows to structure the knowledge to reflect the structure of that part of the universe which is being represented.
 - Default values (e.g. Height of a baseball player to be 195cm). Very strong representation facilities by procedural attachment.
 - Must be much refined

111

Components of a Semantic Net

- Nodes represent objects and descriptive information about those objects.
 - Objects can be any physical item such as a book, car, desk, or even a person.
- Nodes can also be concepts, events, or actions.
 - A concept might be Ohm's Law, an event such as a picnic or an election, or an action such as building a house or writing a book.
- Attributes of an object can also be used as nodes.
 - These might represent size, color, class, age, origin, or other characteristics.

112

Components of a Semantic Net

- The links or arcs show the relationships between the various objects and descriptive factors.
- Some of the most common arcs are of the is-a or has-a type.
 - Is-a, usually just designated as isa, is used to show class relationship, that is, that an object belongs to a larger class or category of objects.
 - Has-a links are used to identify characteristics or attributes of the object nodes.
- Other arcs are used for definitional purposes.

113

Frames

- A frame consists of a collection of slots which can be filled by values or pointers to other frames.

Meaning of "child's birthday party" poorly approximated by definition like "a party assembled to celebrate a birthday" with "party" defined as "people assembled for a celebration".

Children know more plus default assignments:

Child's Birthday Party	
Dress:	Sunday-Best
Present:	Must please host. Must be bought and gift-wrapped.
Games:	Hide and seek. Pin tail on donkey.
Decor:	Balloons. Favours. Crepe-paper
Party-meal:	Cake. Ice-cream. Soda. Hotdogs
Cake:	Candles. Blow-out. Wish. Sing Birthday Song.
Ice-cream:	Standard three-flavour

114

Frames

- A frame provides a means of organizing knowledge in slots that contain characteristics and attributes. In physical form, a frame is somewhat like an outline with categories and subcategories.
- Each frame describes one object.

115

Contents of a Frame

- A frame includes two basic elements: slots and facets.
- A slot is a set of attributes that describe the object represented by the frame. Each slot contains one or more facets.
- The facets (sometimes called subslots) describe some knowledge or procedures about the attribute in the slot. Facets may take many forms .

116

Facet Forms

- **Default:** This facet is used if the slot is empty, that is without any description.
- **Range:** Range indicates what kind of information can appear in a slot (e.g., integer numbers only, two decimal points, 0 to 100).
- **if added:** This facet contains procedural information or attachments. It specifies an action to be taken when a value in the slot is added (or modified). Such procedural attachments are called demons.
- **If needed:** This facet is used in a case when no slot value is given. It triggers, much like the if-added situation, a procedure that goes out and gets or computes a value.
- **Other:** Slots may contain frames, rules, semantic networks, or any type of information.

117

Frame Example

Automobile Frame

- Class of: Transportation
- Name of Manufacturer: Audi
- Origin of manufacturer: Germany
- Model: 5000 Turbo
- Type of car: Sedan
- Weight: 3300 lb.
- Wheelbase: 105.8 inches
- Number of doors: 4 (default)
- Transmission: 3-speed automatic
- Number of wheels: 4 (default)
- Engine: (Reference Engine Frame)
 - Type: In-line, overhead cam
 - Number of cylinders: 5
 - Acceleration (procedural attachment)
 - 0-60: 10.4 seconds
 - Quarter mile: 17.1 seconds, 85 mph
 - Gas mileage: 22 mpg average (procedural attachment)

118

Capabilities of Frames

- Ability to clearly document information about a domain model, for example, a plant's machines and their associated attributes
- Related ability to constrain the allowable values that an attribute can take on
- Modularity of information, permitting ease of system expansion and maintenance
- More readable and consistent syntax for referencing domain objects in the rules
- Platform for building a graphic interface with object graphics
- Mechanism that will allow us to restrict the scope of facts considered during forward or backward chaining
- Access to a mechanism that supports the inheritance of information down a class hierarchy

119

Scripts

- A script is a knowledge representation scheme similar to a frame, but instead of describing an object, the script describes a sequence of events.
- Like the frame, the script portrays a stereotyped situation.
- Unlike the frame, it is usually presented in a particular context.
- To describe a sequence of events, the script uses a series of slots containing information about the people, objects, and actions that are involved in the events.

120

Elements of a Script

- The *entry conditions* describe situations that must be satisfied before events in this script can occur or be valid.
- *Props* refer to objects that are used in the sequence of events that occur.
- *Roles* refer to the people involved in the script.
- The *result* is conditions that exist after the events in the script have occurred.
- *Track* refers to variations that might occur in a particular script.
- *Scenes* describe the actual sequence of events that occur.

121

Restaurant Script

- Track: Fast-food restaurant
- Roles:
 - Customer (C)
 - Server (S)
- Props:
 - Counter, Tray, Food, Money, Napkins, Salt, Pepper, Catsup, Straws
- Entry Conditions:
 - Customer is hungry.
 - Customer has money

122

Restaurant Script

- Scene 1: Entry
 - Customer parks car.
 - Customer enters restaurant.
 - Customer waits in line at the counter.
 - Customer reads the menu on the wall and makes a decision about what to order.
- Scene 2: Order
 - Customer gives order to server.
 - Server fills order by putting food on tray.
 - Customer pays server.

123

Restaurant Script

- Scene 3: Eating
 - Customer gets napkins, straws, salt, etc.
 - Customer takes tray to an unoccupied table.
 - Customer eats food quickly.
- Scene 3A (option): Take-out
 - Customer takes food and exits.
- Scene 4: Exit
 - Customer cleans up table.
 - Customer discards trash.
 - Customer leaves restaurant.
 - Customer drives away.

124

Restaurant Script

■ Results:

- Customer is no longer hungry.
- Customer has less money.
- Customer is happy.*
- Customer is unhappy.*
- Customer is too full.*
- Customer has upset stomach.*

125

Comparison of Knowledge Representation Methods

Knowledge Representation	Advantages	Disadvantages
Production rules	Simple syntax, easy to understand, simple interpreter, highly modular, flexible (easy to add to or modify)	Hard to follow hierarchies, inefficient for large systems, not all knowledge can be expressed as rules
Semantic networks	Easy to follow hierarchy, easy to trace associations, flexible	Meaning attached to nodes might be ambiguous, exception handling is difficult, difficult to program
Frames	Expressive power, easy to set up slots for new properties, easy to create specialized procedures, easy to include default information	Difficult to program, difficult for inference
Formal logic	Facts asserted independently of use, precision, completeness	separation of representation and processing, inefficient with large data sets

126