

Knowledge

- Hard to define
- Study of knowledge – epistemology
- Concerns with nature, structure and origin of knowledge

Pyramid of the knowledge



Wisdom: using knowledge in a beneficial way

Meta knowledge: knowledge about knowledge

Knowledge: rule about using information

Information: Processed data useful for knowledge

Data: Collection of facts

Noise: No apparent information

Created with

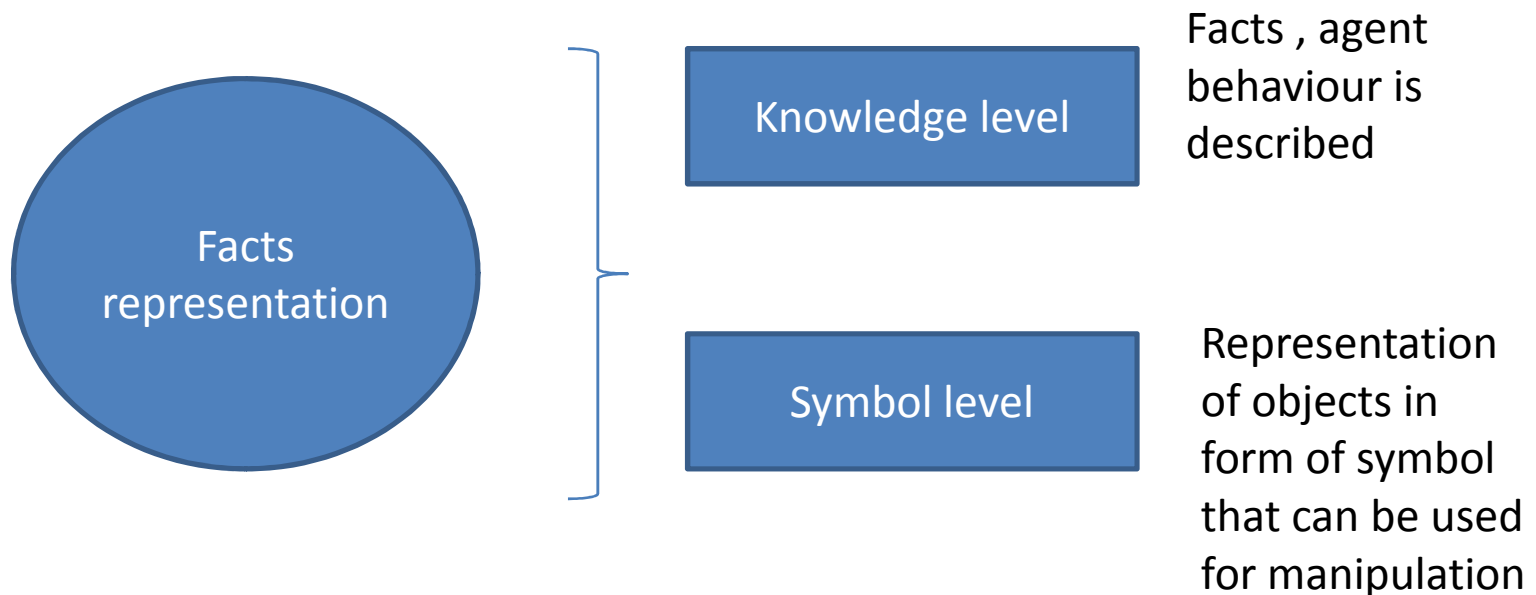


nitro^{PDF} professional

download the free trial online at nitropdf.com/professional

Representations and mapping

- This is structured as two levels



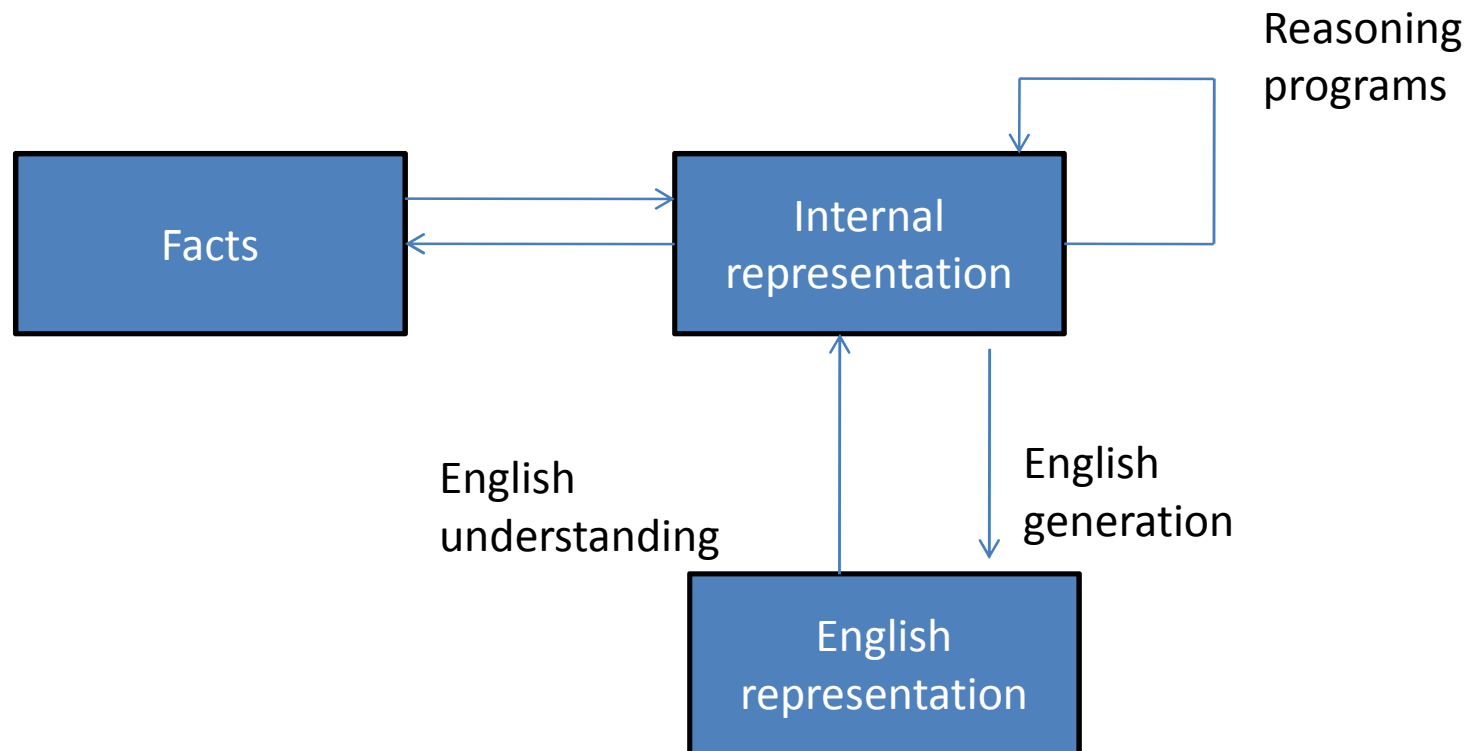
Newell model

Created with

 **nitro**^{PDF} professional

download the free trial online at nitropdf.com/professional

Representation mappings



Mapping between facts and representation

Created with

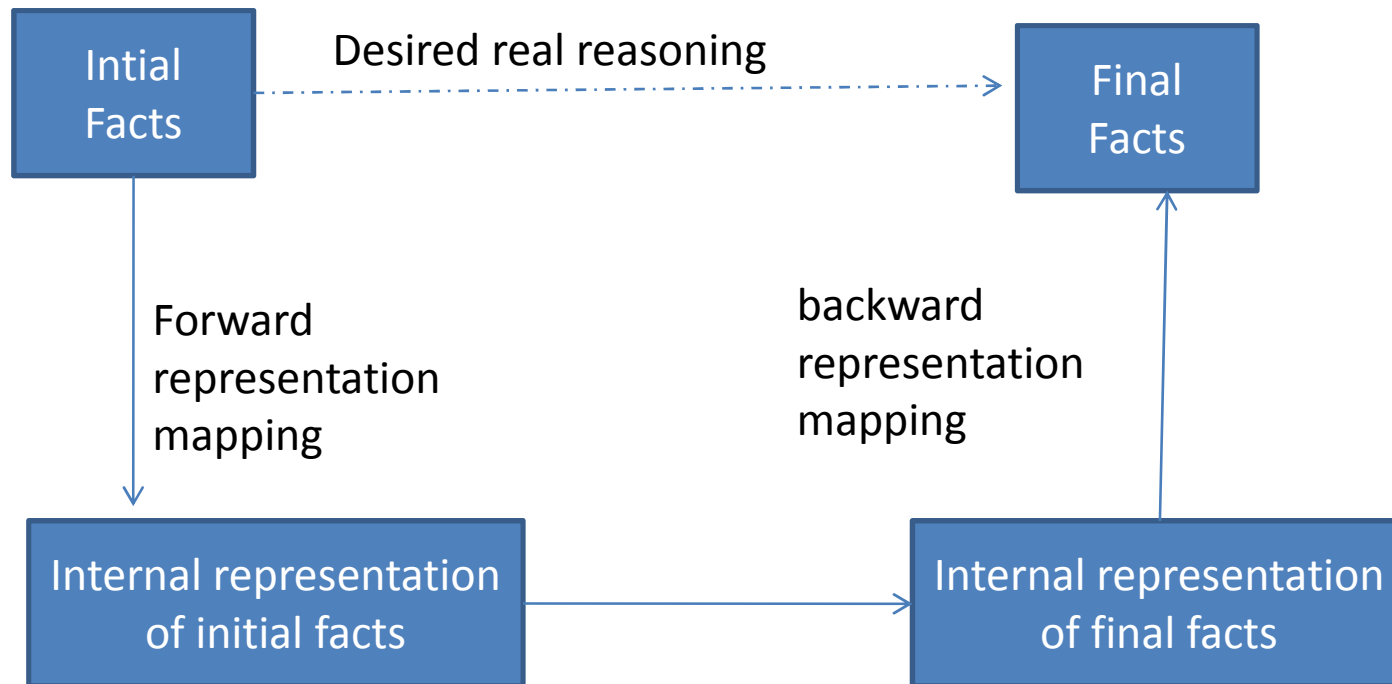
nitroPDF[®] professional

download the free trial online at nitropdf.com/professional

Example

- | • FACT | Internal Representation |
|---|--|
| • Spot is a dog | Dog(Spot) |
| • Every dog has tail | $\forall x \text{ dog}(x) \rightarrow \text{hastail}(x)$ |
| • Using deductive mechanism , a new representation is achieved | |
| – Hastail(Spot) | |
| • Using backward mapping we can then obtain the english statement | |
| – Spot has a tail | |

Representation of facts



Properties of KR

- **Representational adequacy** Ability to represent all kinds of knowledge that are needed in the domain
- **Inferential Adequacy** ability to manipulate the representational structures in such a way to derive new structures corresponding to new knowledge derived from old.
- **Inferential efficiency** ability to incorporate into the knowledge structure additional information that can focus the attention of inference mechanism in more promising direction
- **Acquisitional Efficiency**: ability to acquire new information easily. Either by direct insertion or by program itself

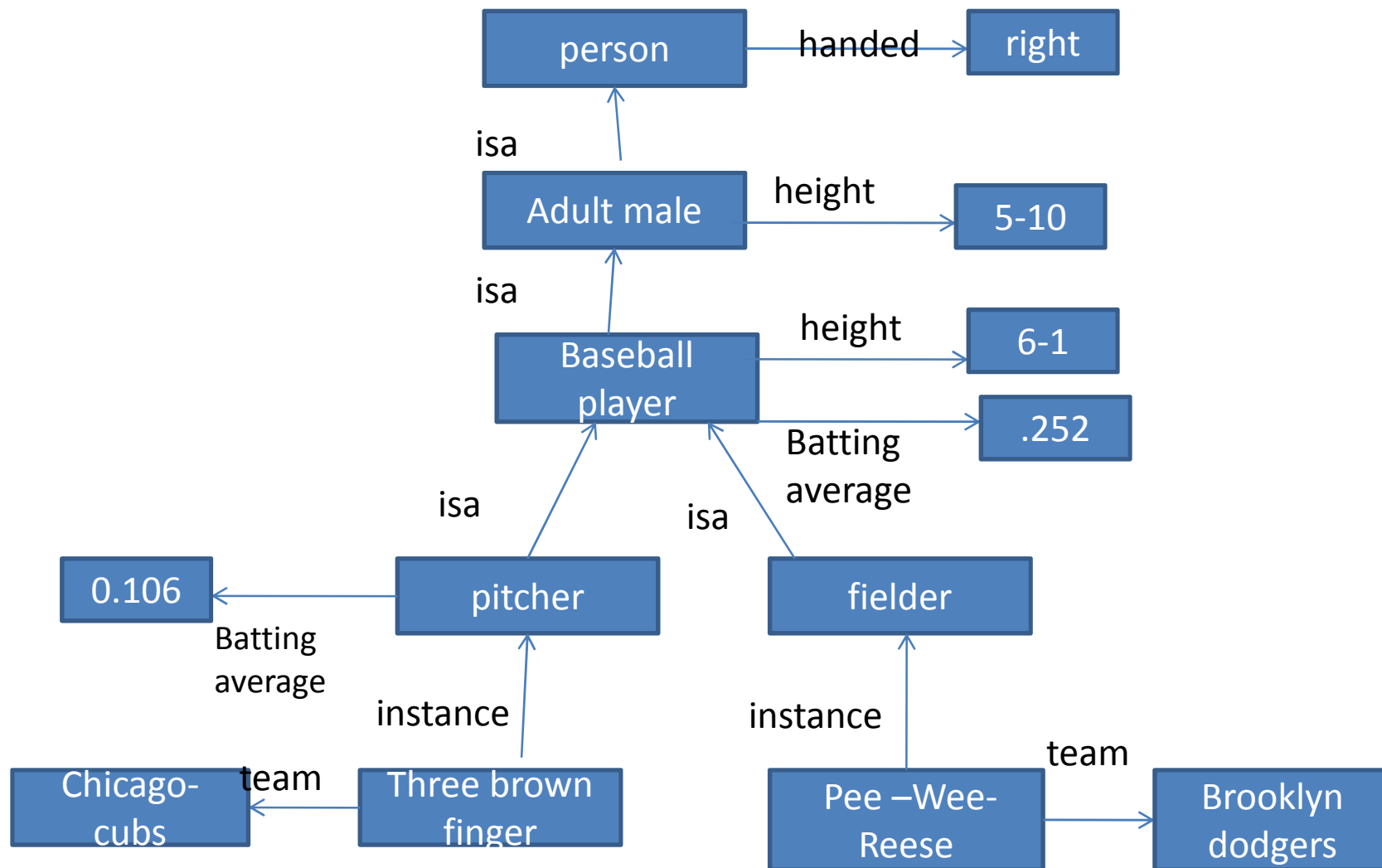
Types of knowledge

- Simple relational knowledge
 - Represented in tabular form
 - Simple
 - Weak inferential capabilities
 - Similar to databases

Player	Height	weight

Types of knowledge

- Inheritable knowledge
 - Uses objects , classes and attributes
 - Property inheritance
 - Slot and filler structure
 - Frames and semantic net
 - Isa and instance link



Inferencing

Team(Pee-Wee_Reese)=Brooklyn Dodgers

Batting_average(Three_brown_finger)=0.106

Height(Pee_Wee_Reese)=6.1

Created with

Types of knowledge

- Inferential knowledge
 - All properties or knowledge using logic
 - An efficient inferential procedure
 - Knowledge is useless if there is no inferential procedure
 - Inference in forward direction (from facts to conclusion)
 - Inference in backward direction(from conclusion to facts)
 - Inference by contradiction (resolution principle)
 - Inference rules

Types of knowledge (contd..)

- Procedural knowledge
 - Operational knowledge that specify what to do and when to do
 - Uses procedure
 - If the else constructs
 - Most common way to represent this type is to use code of a language like LISP
 - It has low score in terms of
 - Inferential adequacy
 - Acquistional efficiency
 - Most common way to represent this knowledge is production rules

Knowledge representation mechanism

- Logic
 - Propositional
 - Predicate
- Semantic net
- Frames
- Scripts
- Conceptual dependencies

Proposition logic

- Propositions are atomic sentences
- Syntax governs the combination of building blocks such as propositions and connectives
- In place of sentences term *formulae or well formed formulas (wff)* is used
- Connectives

(\sim \vee \wedge \leftrightarrow \rightarrow)

(not or and double implication implication)

Syntax of PL

- Example
 - Its raining RAIN
 - Its windy WINDY
 - Its sunny SUNNY
 - Its windy but not sunny $WINDY \wedge \sim SUNNY$
- Syntax
 - If P and Q are formulae then following are formula
 - $\sim P$
 - $(P \& Q)$
 - $(P \vee Q)$
 - $(P \rightarrow Q)$
 - $(P \leftrightarrow Q)$

Syntax of PL

- Formulae can be a compound formulae
- Precedence of operators in absence of parenthesis is
 - \sim
 - $\&$
 - \vee
 - \rightarrow
 - \leftrightarrow

Semantics of PL

- Meaning of the sentence is just true or false
- Assignment of truth value to the sentence
- An **interpretation** for the sentence or group of sentence is an assignment of truth value to each propositional symbol
 - Example $P \& \sim Q$
 - Interpretation

	P	Q	$\sim Q$	$P \& \sim Q$
• I1	t	t	f	f
• I2	t	f	t	t
- Once interpretation are given truth value of sentence can be determined

Example

- $((P \& \sim Q) \rightarrow R) \vee Q$
 - If $P=t$, $Q=f$, $R=f$
- Find truth value of sentence

Properties of sentence

- Satisfiable
 - A sentence is satisfiable if there is some interpretation for which it is true
- Contradiction
 - A sentence is contradictory (unsatisfiable) if there is no interpretation for which it is true
- Valid
 - A sentence is valid if it is true for every interpretation. They are also called tautologies
- Equivalence
 - Two sentences are equivalent if they have same truth value under every interpretation
- Logic Consequences
 - A sentence is logical consequence of other if it is satisfiable by all interpretations which satisfy the first

Examples

- A valid sentence is satisfiable
- A contradictory statement is invalid
- P is satisfiable
- $P \vee \sim P$ is Valid
- $P \& \sim P$ is contradictory
- P and $\sim(\sim P)$ are equivalent
- P is logical consequence of $(P \& Q)$

Example

- To prove $P \rightarrow Q$ is equivalent to $\sim P \vee Q$

P	Q	$\sim P$	$\sim P \vee Q$	$P \rightarrow Q$
T	T	F	T	T
T	F	F	F	F
F	T	T	T	T
F	F	T	T	T

Inference Rules

- Inference Rule
 - Provide means to perform logical proofs or deductions
 - Gives a set of sentence $S=\{s_1,s_2,\dots,s_n\}$, prove the truth of s (conclusion)
 - Syntactic methods of inference of deduction can be used
 - They do not depend on truth assignment , but on syntactic relations

Inference rules

- Modus Ponens Rule

- From P and $P \rightarrow Q$ infer Q

Example

given *Joe is intelligent*

and *Joe is intelligent \rightarrow Joe will pass exams*

Conclude *Joe will pass exam*

- Chain Rule

- From $P \rightarrow Q$ and $Q \rightarrow R$ infer $P \rightarrow R$

Inference rules

- Substitution
 - If s is valid, s' derived from s by substitution of propositions in s , then s' is also valid
 - $P \vee \sim P$ is valid then $Q \vee \sim Q$ is also valid
- Simplification
 - From $P \& Q$ infer P
- Conjunction
 - From P and from Q infer $P \& Q$
- Transposition
 - From $P \rightarrow Q$, infer $\sim Q \rightarrow \sim P$

Formal system

- A formal system is a set of axioms S and a set of inference rules L from which new statements can be logically derived. System $\langle S, L \rangle$ is also referred as knowledge base(KB)
 - Soundness
 - If $\langle S, L \rangle$ is a formal system , we say inference procedure L is sound if and only if any statement s can be derived from $\langle S, L \rangle$ is a logical consequence of $\langle S, L \rangle$
 - Completeness
 - Let $\langle S, L \rangle$ be a formal system , then inference procedure L is complete if and only if any sentence s is logically implied by $\langle S, L \rangle$ can be derived using that procedure

Syntax and semantics of FOPL

- Drawbacks of PL
 - Too “coarse” to describe properties of an object
 - Lack of structure to represent relations among two objects
 - Does not allow to construct generalized statements about classes of similar objects
 - Limitation while reasoning with the system
- FOPL
 - was developed to extend expressiveness of PL
 - Is a generalisation of PL that allows reasoning with world objects as entities , classes etc
 - Generalisation is possible by using predicates in place of proposition, use of variable , function etc

Syntax of FOPL

- In Predicate logic real world facts can be represented as *statements* written as wff
- Syntax of FOPL
 - Is determined by the allowable symbols and rules of combinations
- Semantics of FOPL
 - Are determined by the interpretation assigned to predicates instead of proposition
 - Interpretations are also assigned to constant , variables , function and arguments

Syntax of FOPL

- Connectives

\sim $\&$ \vee \rightarrow \leftrightarrow
NOT AND OR Implication double implications

- Quantifiers

- \exists there existential quantifier (there exist some)
- \forall universal quantifier (for all)

- Constants

- Fixed value term denoting number terms, words etc

- Variables

- it can assume different values over given domain x, y, z

Syntax of FOPL

- **Function**

- Function symbols denote relation define on a domain D
- They map n element to single element
- $\text{age-of}(x)$, $F((t_1, t_2, t_3, \dots, t_n))$ where t_i can be constant, variable and function

- **Predicates**

- Predicate symbol denote the relations or functional mapping from elements of domain D to values true or false
- $P(t_1, t_2, \dots, t_n)$ can take constant , variable or function as their argument
- 0-ary predicate is proposition or constant predicate
- A predicate with no variables is called **ground atom**

Syntax of FOPL

- Constant , variables and function are called terms
- Predicate is referred as atom
- Predicate with no variable is termed as ground atom
- Also atom and its negation is called literal

Semantics of FOPL

- Wffs are implemented in domain D
- When assignment of values is given to each predicate and each symbol , we say interpretation is given to wff
- $\forall x: ((A(a,x) \vee B(f(x))) \ \& \ C(x)) \rightarrow D(x)$

Given $D=\{1,2\}$

$a=2$

$f(1)=2 \ f(2)=1$

$A(2,1)=\text{true} \ A(2,2)=\text{false}$

$B(1)=\text{true} \ B(2)=\text{false}$

$C(1)=\text{true} \ C(2)=\text{false}$

$D(1)=\text{false} \ D(2)=\text{true}$

For $x=1$ find the truth value of sentence

For $x=2$ find the truth value of sentence

Properties of wff

- Valid
- Inconsistent (unsatisfiable/contradictory)
- Satisfiable
- Equivalence
- Logical consequence

Properties of wff

- Equivalent logical expression
 - $\sim(\sim F)=F$
 - $F\&G=G\&F$ Commutative
 - $(F\&G)\&H=F\&(G\&H)$ associative
 - $F\&(G\vee H)=(F\&G)\vee(F\&H)$ distributive
 - $\sim(F\&G)=\sim F\vee\sim G$ De Morgan's law
 - $F\rightarrow G=\sim F\vee G$
 - $F\leftrightarrow G=(\sim F\vee G)\&(\sim G\vee F)$
 - $\sim(\forall x)F(x)=\exists x \sim F(x)$

FOPL

- A variable of predicate is bounded if it is associated with quantifiers , otherwise free
- $\forall x (P(x) \rightarrow Q(x,y))$
 - x is bound and y is free
- Given $F_1, F_2, F_3, \dots, F_n$ as wff containing disjunct of literals only then
 - $F_1 \& F_2 \& F_3 \dots F_n$ is *conjunctive normal form (CNF)*
- Given $F_1, F_2, F_3, \dots, F_n$ as wff containing conjunct of literals only then
 - $F_1 \vee F_2 \vee F_3 \dots F_n$ is *disjunct normal form (DNF)*

Resolution principle

- It is inference procedure used in logic
- It gains its efficiency when operated on set of statement converted to standard convenient form
- Resolution produces proof by refutation
- To prove a statement , resolution attempts to show that negation of the statement produces a contradiction with known statement.

Conversion to clausal form

- Eg “ All Romans who know Marcus either hated him or think that anyone who hates anyone is crazy.”
- Such representation are complex and makes the matching processes tedious
- Formulae would be easier to work if
 - it had less embedding of components
 - Quantifiers were separated from formulae
- CNF has both the properties

Conversion to clausal form

- For resolution we need to reduce set of wff to clauses
- Clauses is set of wff in CNF with no instances of &.
- This is achieved by
 - First converting wff into CNF
 - Breaking apart each expression into clauses, one for each conjunct

Conversion to clausal form

- One of the inference programs is resolution
- Resolution requires that all statements be converted to normalized clausal form
- **Clause** : is defined as disjunction of number of literals
- **Ground clause**: clause in which no variable occurs in expression
- **Horn Clause**: clause with at least one positive literal

Conversion to clausal form

1. Remove implications : replace $p \rightarrow q$ with $\sim p \vee q$
2. Move \sim negation to innermost atom
3. Rename variable so that every bounded variable is unique- standardisation of variables
4. Move all quantifiers to left of wff without changing their relative order (prenex normal form)
5. Remove existential quantifiers (Skolemisation)

- Skolemisation

- Existential quantifier can be removed by this process
- By substituting for variable a reference to a function that produces a desired value
- A function name for is created for each replacement
 - Eg 1 Existential quantifier without universal quantifier proceeding it
 - Eg 2 Existential quantifier exist with scope of universal quantifier

Conversion to clausal form

- Drop all universal quantifiers and put the statement
- Convert into CNF
- Break into independent clause
- Standardize variable

Example-1

1. Marcus was man
2. Marcus was Pompeian
3. All Pompeian's were Roman
4. Caesar was ruler
5. All Romans were either loyal to Casar or hated him
6. Everyone is loyal to someone
7. People try to assassinate rulers they are not loyal to
8. Marcus tried to assassinate Caesar
9. All men are people

Example 2

1. Marcus was man
2. Marcus was Pompeian
3. Marcus was born in 40 AD
4. All men are mortal
5. All Pompeian died when volcano erupted in 79 AD
6. No mortal lives longer than 150 years
7. It is now 1991
8. Alive means not dead
9. If someone dies then he is dead for all times

Monkey banana problem

1. In_room(B)
2. In_room(C)
3. In_room(M)
4. Dexterous (M)
5. Tall(C)
6. $\sim \text{close}(B, \text{floor})$
7. Can_move(M, C, B)
8. Can_climb(M, C)
9. $\text{Dexterous}(X) \ \& \ \text{close}(x, y) \rightarrow \text{can-reach}(x, y)$
10. $\text{Get_on}(x, y) \ \& \ \text{under}(y, B) \ \& \ \text{tall}(y) \rightarrow \text{close}(x, B)$
11. $\text{In_room}(x) \ \& \ \text{In_room}(y) \ \& \ \text{In_room}(z) \ \& \ \text{can_move}(x, y, z) \rightarrow \text{close}(z, \text{floor}) \vee \text{under}(y, z)$
12. $\text{Can-climb}(x, y) \rightarrow \text{get_on}(x, y)$

To prove can_reach(M, B)

Where M is monkey , B is banana , C is chair

Resolution Principle

- Definition “ A syntactic inference procedure which when applied to set of clauses determine if set is unsatisfiable”
- If there is set of clause $S=\{C1,C2,....Cn\}$ and we wish to deduce D
- To show D is logical consequence of $C1\&C2\&.....\&Cn$. Negate D and add $\sim D$ to S .
- Using resolution we can show that set S is unsatisfiable by deducing contradiction
- Such proof is called proof by refutation if it yields empty clause $[]$

Resolution

- Binary Resolution: Two clauses having complementary literals are combined as disjunct to produce a single clause, after deleting the complementary literals

$$- \sim P(x,a) \vee Q(x) \quad \sim Q(b) \vee R(x)$$

$b|x$

$$\sim P(b,a) \vee R(b)$$

Unit Resolution: A number of clauses are resolved simultaneously to produce a unit clause

Linear Resolution : When each column C_i is parent to clause C_{i+1} , process is linear resolution

Created with

 **nitro**^{PDF} professional

download the free trial online at nitropdf.com/professional

Resolution Principle

- Iterative process
- 2 clauses called parent clause are compared (resolved)
- Giving a new clause : new clause represent the way two parent interact
- Example: winter V summer
- \sim winter V cold
- After resolving : summer V cold

Resolution in predicate logic

- Situation is complex because of substitution
- Theoretical basis of resolution in predicate logic is Herbrand's theorem
 - To show set of clauses S is unsatisfiable
 - Consider only interpretations over a particular set called *Herbrand Universe*
 - A set S is unsatisfiable if and only if finite subset of ground instance of S is unsatisfiable

Resolution in proposition Logic

- Convert all preposition to clausal form
- Negate P , convert it to clausal form
- Repeat till contradiction is reached or no progress made
 - Select any 2 clause- call them parent clause
 - Resolve them together
 - Resolvent = derisjunct of all literals
 - Exception case: if L and $\sim L$ exist such that $L \in \text{parent1}$ and $\sim L \in \text{parent2}$, select such pair and remove them from resolvent
 - If resolvent = empty clause \rightarrow contradiction found
 - If not, add it to set of clauses
- Example Given P
 - $(P \& Q) \rightarrow R$
 - $(S \vee T) \rightarrow Q$
 - T
- Prove R

Resolution in Predicate logic

- Given set of statement F and to prove P
- Change all F into clausal form
- Negate P , add it to set of clause
- Repeat till contradiction is reached or no progress made
 - Select any 2 clause- call them parent clause
 - Resolve them together
 - Resolvent = disjunct of all literals with proper substitution
 - Exception case: if $X1$ and $\sim X1$ exist such that $X1 \in \text{parent1}$ and $\sim X1 \in \text{parent2}$, then $X1$ and $\sim X1$ are unifiable and should not appear in resolvent
 - If resolvent = empty clause \rightarrow contradiction found
 - If not, add it to set of clauses

Choice of literals

- Only resolve pair of clauses containing complementary literal
- Eliminate certain clause like *tautologies* [which can never be satisfied]
- Eliminate clause that subsume other clause
- Resolve with one clause that is part of statement we are trying to prove- *support strategy*
- Resolve with clauses having one or few literals so that number of new literals are less- *unit preference strategy*

Unification algorithm

- Determination of contradiction is easy in proposition logic, L and $\sim L$; X and $\sim X$ contradict
- In predicate we need to map parameters of predicate being compared
- Eg $\text{dog}(\text{spot})$ and $\sim \text{dog}(\text{Pup})$ do not contradict
- Definition
 - *Recursive procedure that compare two literals and discover whether there exist a set of substitution that makes them identical is called unification algorithm.*

Unification algorithm

- Rules for unification
 - Initial predicate symbol should be same.
 - Number of arguments must be same
 - Compare each argument recursively. Arguments
 - With same parameter match
 - With same constant match
 - A variable can match another variable, constant or predicate, but predicate must not contain that variable.
 - There should be consistent substitution
 - $P(x, m)$
 - $P(y, z)$ will result in either $z | m, y | x$

Unification algorithm

- In general substitution can be given as
- $(a_1 | a_2, a_3 | a_4, \dots)(b_1 | b_2, b_3 | b_4, \dots)$
- Objective is to discover atleast one substitution that causes two literals to match.
- This is achieved in unification algorithm

UNIFICATION ALGORITHM UNIFY (L1, L2).

- Step 1: If L1 and L2 are both variables or constant then
 - If L1 and L2 are identical return nil
 - If L1 is variable and L1 occurs in L2 then return fail else return (L2|L1)
 - If L2 is a variable and L2 occurs in L1 then return FAIL else return (L1|L2)
 - Else return { FAIL }
- Step 2: If initial predicate symbols in L1 and L2 are not identical then return { FAIL }
- Step 3: If L1 and L2 have different number of argument then return {FAIL}

- Step 5: for $i=1$ to no_of_arguments
 - Call unify with i th arg of L1 and i th arg of L2, putting results in S
 - If S contain FAIL return {FAIL}
 - If S is equal to NIL then
 - Apply S to remainder of L1 and L2
 - SUBST=append (S,subst)
- Step 4: Set SUBST to NIL (SUBST contain list of substitution)

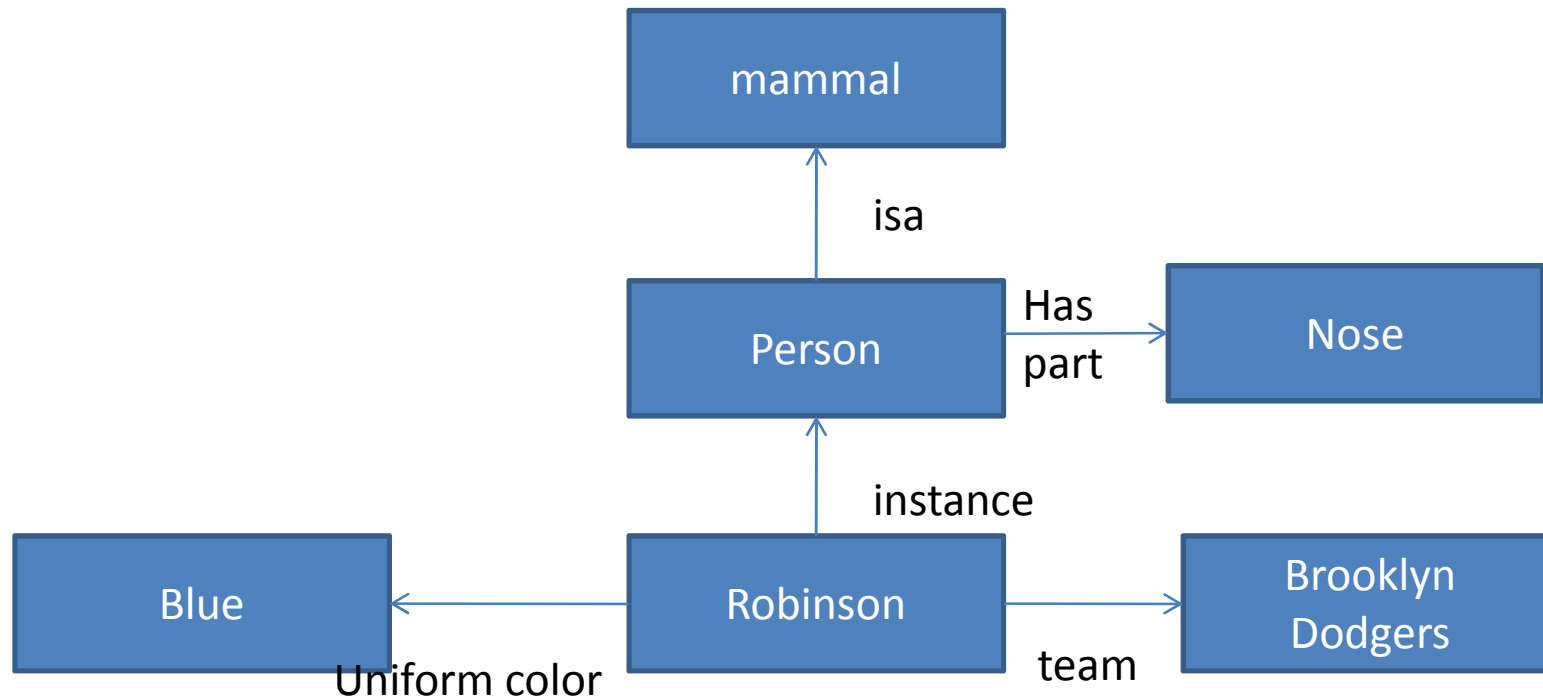
Slot and filler structure

- A device to support property inheritance
 - Isa
 - Instance links
 - Weak and strong slot and filler structure
- Here inheritance is easy because knowledge in slot and filler structure is represented as entities and set of attributes
- Two views are semantic net and frames
 - They are weak slot and filler structure as they have some limitation.

Semantic network

- Meaning of a concept comes from the ways in which it connected to other concept
- A semantic network is a structure for representing knowledge as a pattern of interconnected nodes and arcs
- Nodes represent
 - Entities
 - Attributes
 - States
 - Events
- Arc represent the relationship between nodes and label specify the type of relation that exist

Example



Semantic Network

Created with

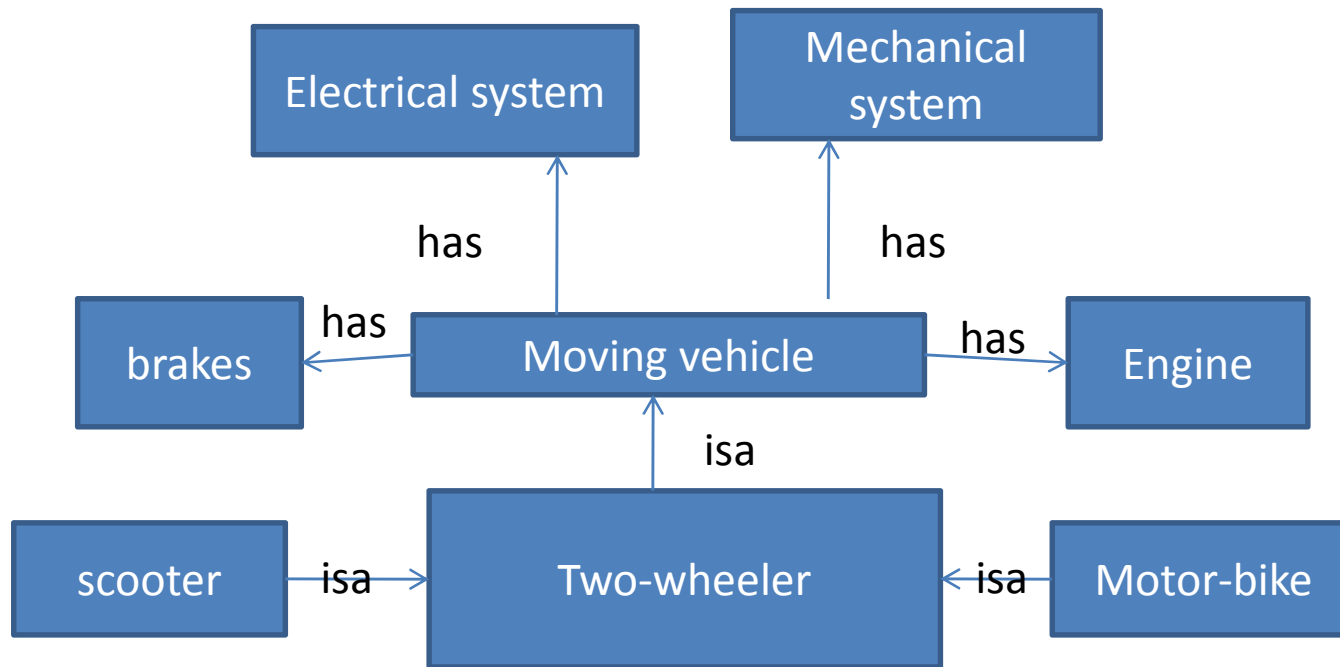
 **nitro**^{PDF} professional

download the free trial online at nitropdf.com/professional

Semantic net

- Semantic network is structure for representation of knowledge as a pattern of interconnected nodes and arcs
- It is a graphical representation of knowledge
- Nodes represent
 - Entities
 - Attributes
 - States or events
- Arcs represent
 - Relation between nodes
 - Label on arc specify the type of relation
 - It is possible to add more knowledge by linking other objects with different relations
- Also termed as **propositional net and associative net**

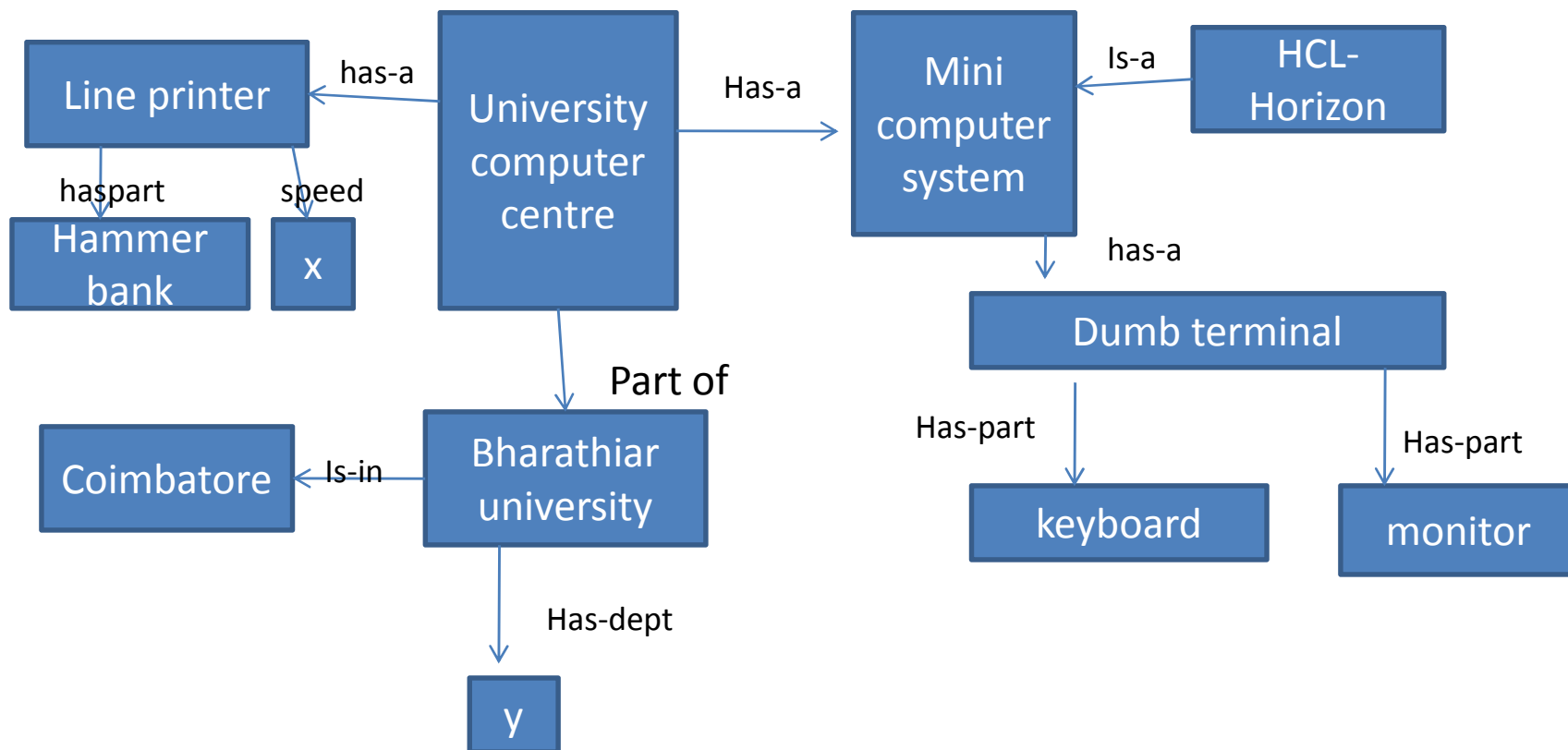
Semantic net example



Deduce : scooter is two wheeler and is a moving vehicle

Semantic net

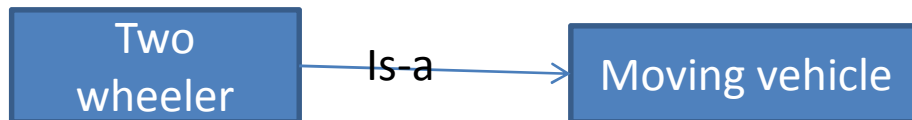
Representation of variables is also possible



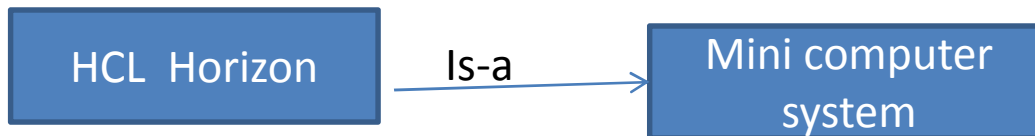
Created with

Semantic Net

- There are two types of nodes
 - Generic nodes
 - Individual or Instances nodes
- Generic node is a very general



- Individual Node



Property of is-a

- Is-a link
 - Is hierarchical in nature
 - Inheritance
- Reasoning with semantic net
 - Specify the start node
 - Traverse the other node using links
 - Till final node is reached
- What is the speed of the line printer?

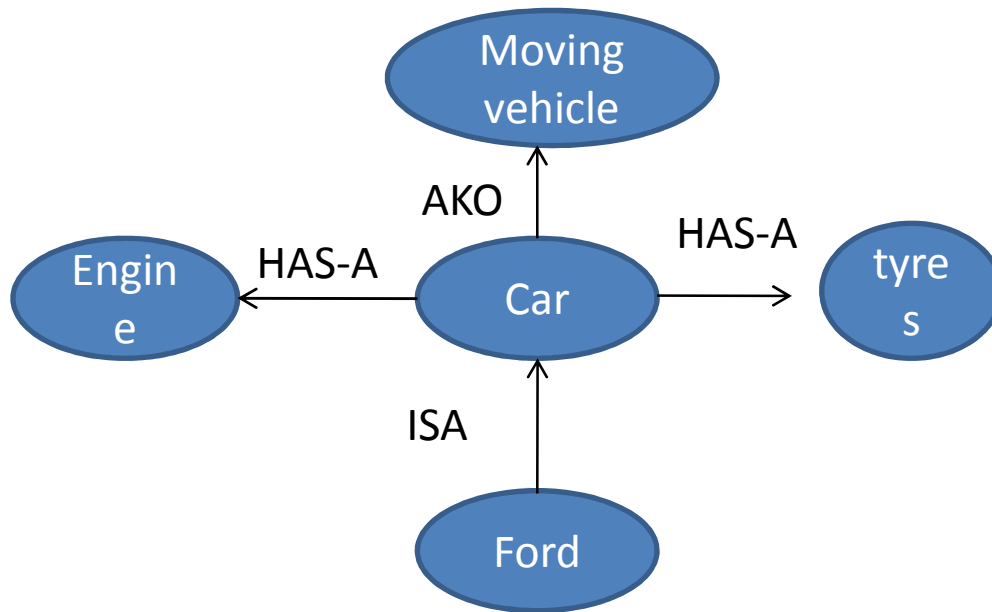
Partition Semantic Net

- Allows use of quantifiers , for all there exists
- Eg All dog bit a mail carrier

Semantic net and relation

- Is-a and a-kind-of, (ISA AKOF) are commonly used link
- Is-a is used to define the ‘instance of’ and refers to specific member of the class.
- Link AKO is used to relate one class to another, specially individual class to parent class
- AKO relates generic node to generic
- ISA relate individual to generic
- Another link HAS-A, relates subclass to a class
- It is used to relate object to part of the object
- Points opposite to AKO

Semantic Net



OAV table

Object	Attribute	Value
Car	Part	Engine
Car	Part	Tyres
Car	Type	Moving vehicle

- Problem with SN- No standard definition of link name
- Three items object attribute and value occur so frequently that it is possible to build a simplified semantic net using them
- An **object-attribute-value triple** (OAV) can be used to characterize all knowledge in semantic net

Advantage of semantic net

- Graphical
- Allows inheritance and instance representation
- Reasoning
- Can be converted into logic
- Allows variables, quantifiers

Disadvantage

- No standardization and formalization as far as notation and reasoning are concerned , but overall concept of arcs and nodes in semantic network has been standardized
- Gets large with complex representation for all and there exist

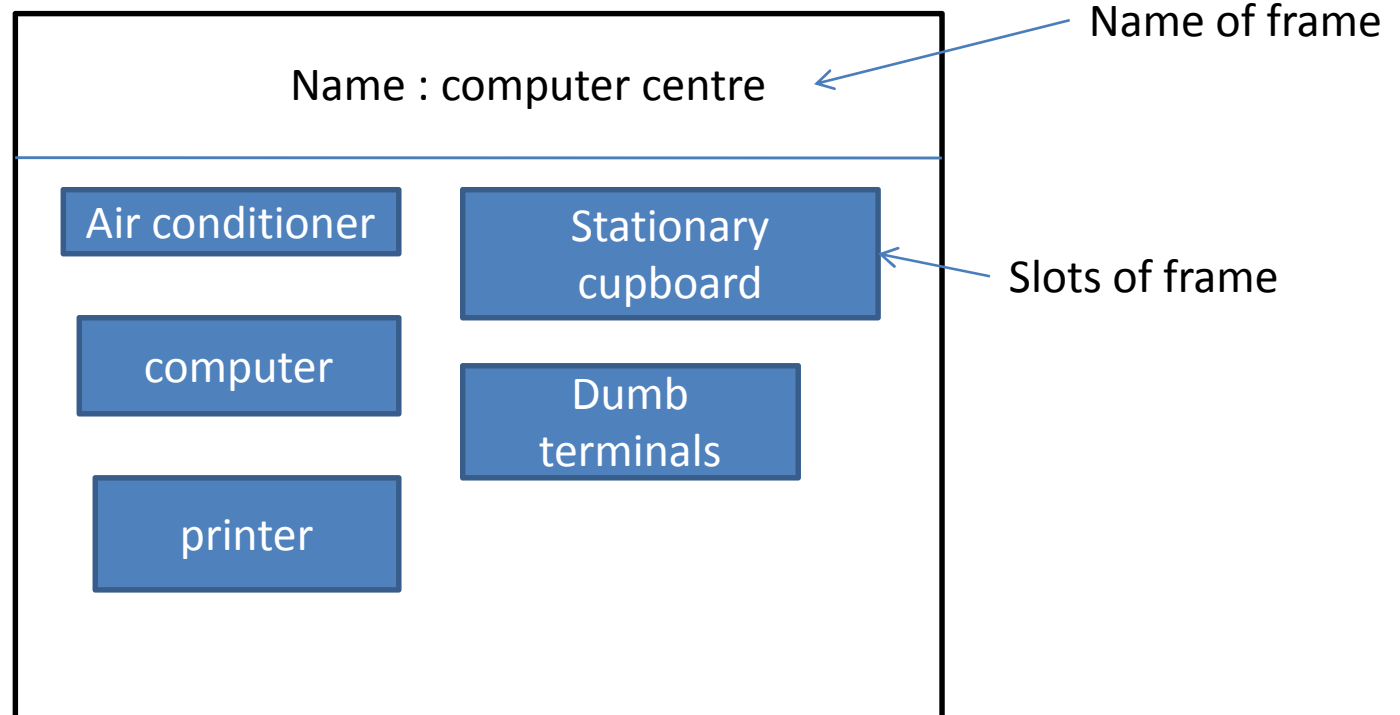
Frames

- Means of representing common sense knowledge
- Knowledge is organized into small “packets” called frame (Minsky)
- Contents of frame are certain slots which have values
- Example is computer center at our campus which may include
 - Computers, Printers, AC, UPS, Cuboard
- When we visit another computer centre, we remove some item and add other

Frames

- A frame is collection of attributes called slots and associated values that describe some entity in the world
- Entity may represent physical or abstract object.
- It is a data structure that has slots for various objects and a collection of frames consists of expectations for a given situation
- It provides facility for describing objects , facts and procedure
- It represent two types of knowledge
 - Declarative knowledge
 - Procedural knowledge

Example



A sample frame of computer system

Created with

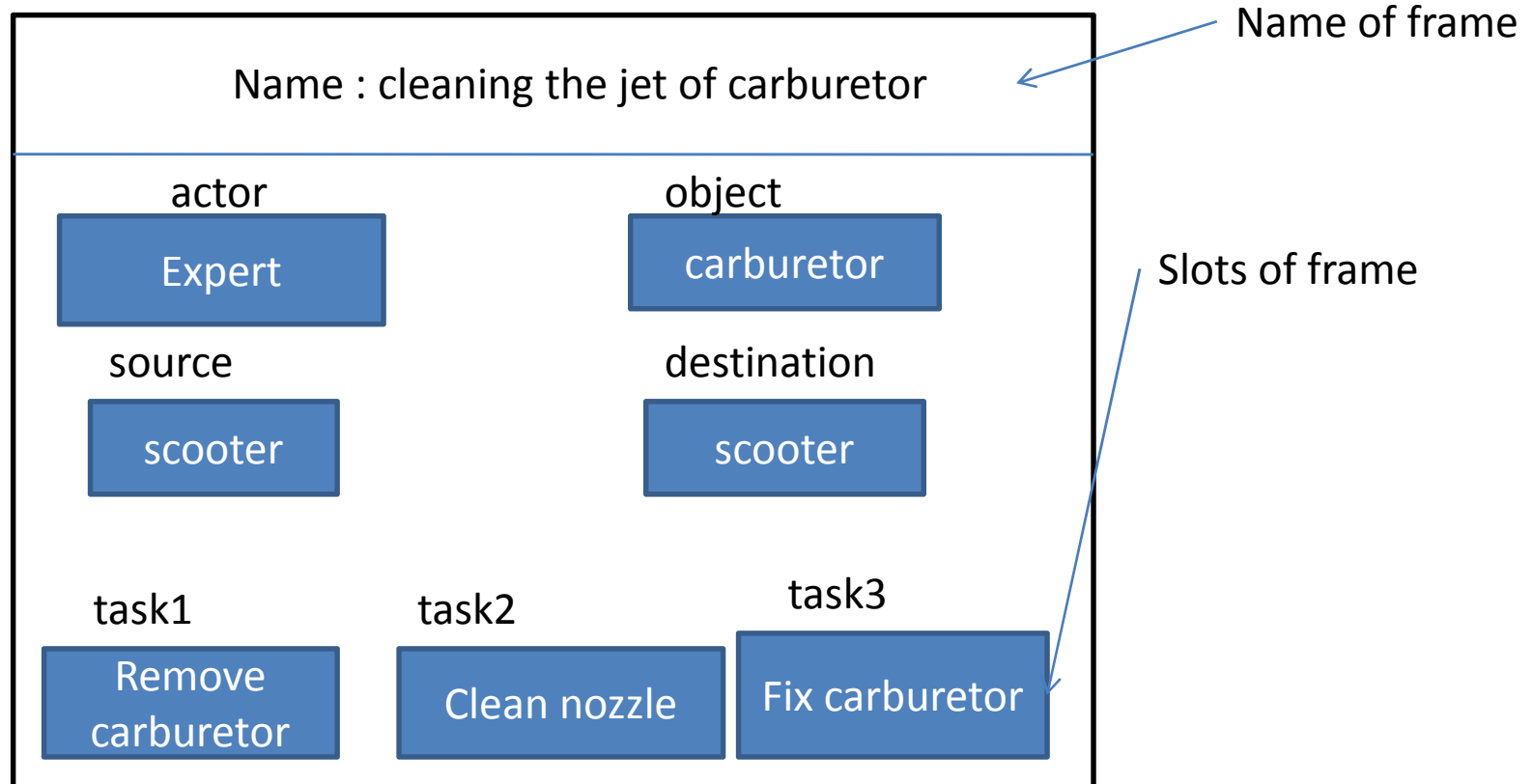
Types of frames

- **Declarative frame** : A frame that contains description about objects is called declarative or factual frame. Example : computer centre
- **Procedural frame** When slots in frame describe how to perform task then such slots are procedural and are called procedural frame
- Such frames in which procedural knowledge is embedded is also called action-procedure frame

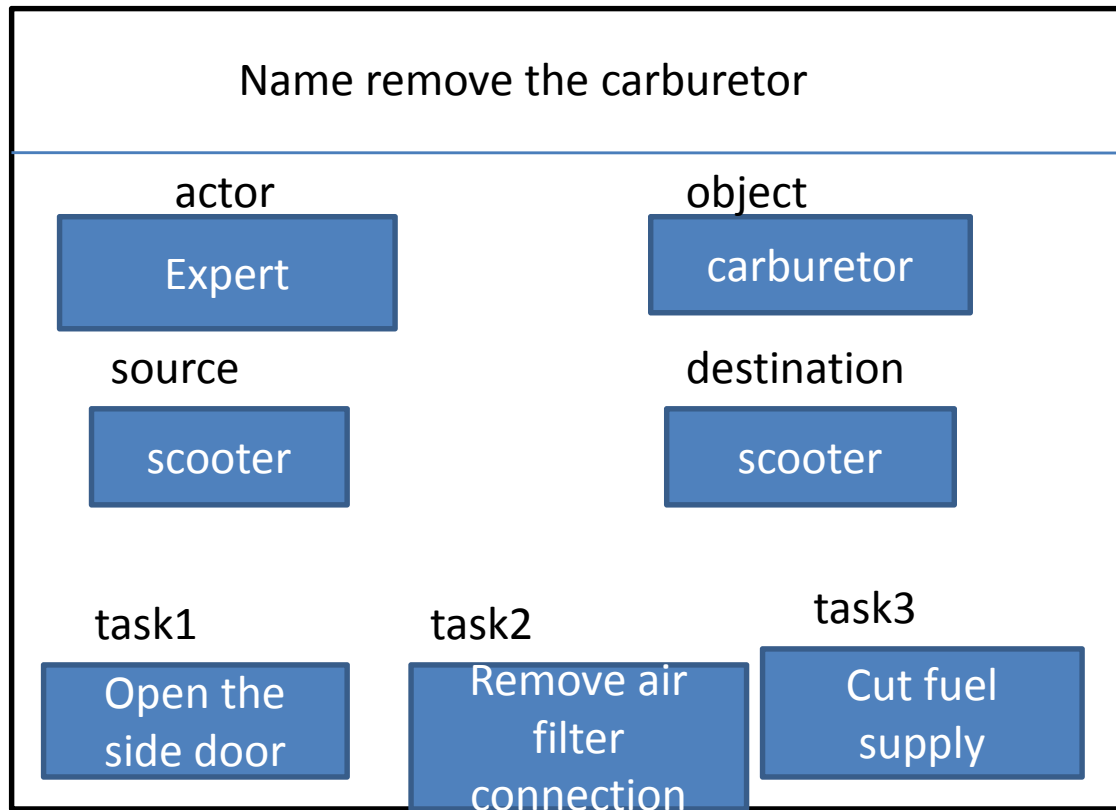
Action-procedure frame

- Action procedure frame have following slots:
 - Actor slots : hold information about who is performing the job
 - Object slots: it has information about the item to be operated upon
 - Source slots: holds information from where the action has to begin
 - Destination slots: hold information about place where action has to end
 - Task slots: generates the necessary sub frames required to perform operation

Example of procedural frame

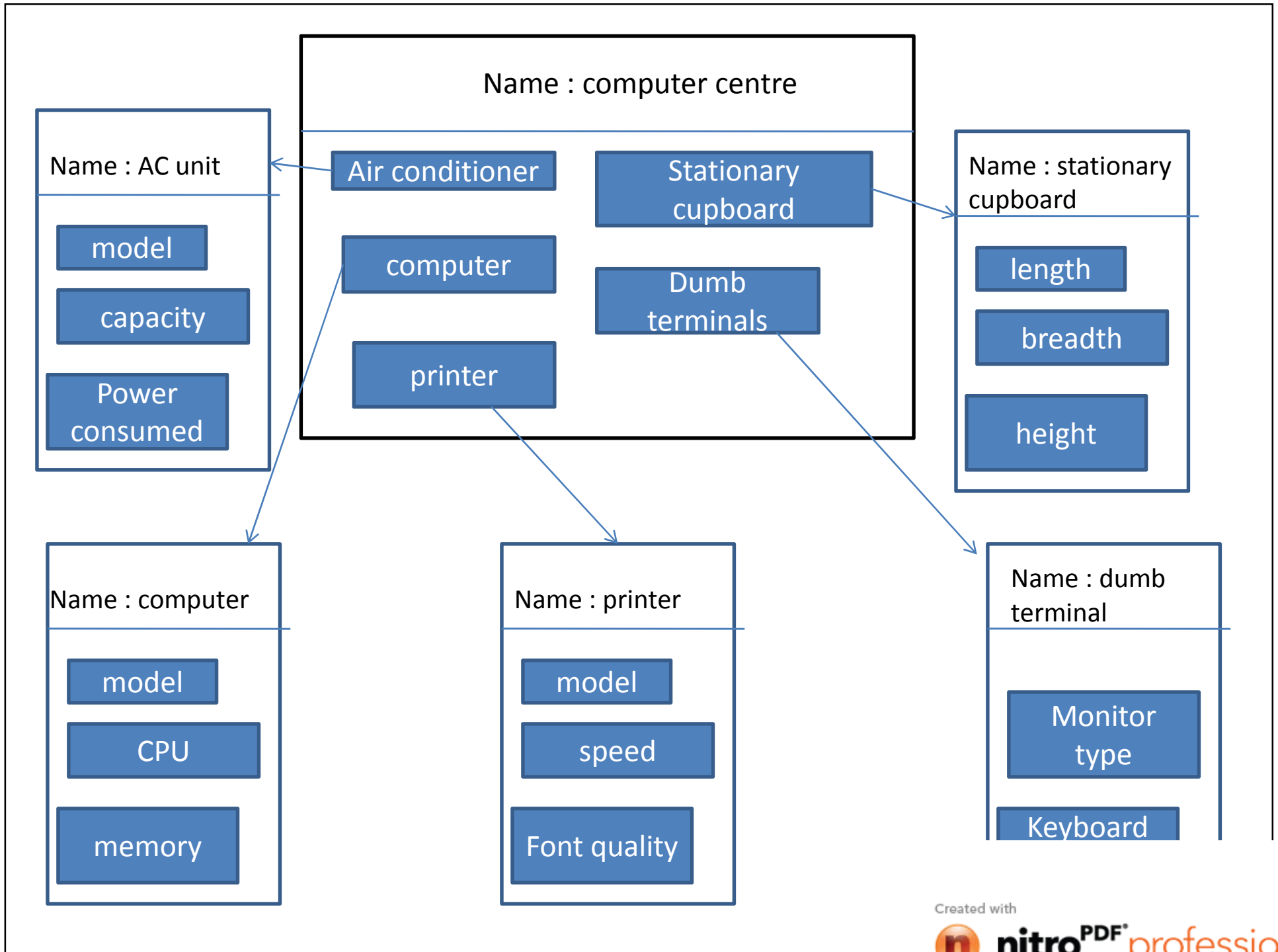


A procedural frame



Frames

- A single frame taken alone is rarely useful
- Frames system may be built from collections of frames that are connected to each other by virtue of the fact that value of an attribute of one frame may be another frame.
- It helps in breaking problem in sub task which can be described as top-down design



Frames

- Are similar to structure – field and value
- Frames – slots and value

A Car Frame	
Slots	Filler
Manufacturer	GM
Model	Chevrolet
Year	1979
Transmission	automatic
Engine	gasoline
Tires	4
Color	blue

A generic frame

Slots	Filler
Name	car
Specialization	a kind of property
Manufacturer	(GM,Ford toyota)
	if added : PROC add_model
types	Chevrolet
location	1979
Wheels	4
Transmission	automatic, manual
Engine	(gasoline, hybrid, gas)

Instance

Slots	Filler
Name	John's car
Specialization	is-a
Manufacturer	GM
types	Chevrolet
location	1979
Wheels	4
Transmission	automatic, manual
Engine	gasoline

Reasoning using frames

- It is done by instantiation
- Instantiation process begins when one given situation is matched with frames that already exist.
- Once it matches , it then fills up the slots with values which depict a particular situation and reasoning process tries to move towards the goal
- Reasoning process can be defined as filling frames slots in frame
- If value is not available it uses default value
- If different characteristics is observed values of the corresponding slots are updated

Reasoning

- Reasoning using frames allow moving from one frame to another to match the current situation
- It builds a wide network of frames there by facilitating the building of knowledge base representing knowledge about common sense

Frames as set and instances

- In this view each frame represent a class (set) or an instance(element of class)
- Isa relation is used as subset
- Instance is used as element-of
- Class represent the set hence 2 kinds of attribute
 - One from the set itself
 - Other inherits from other set

person

isa: mammal
cardinality: 6,000,000,000
*handed : right

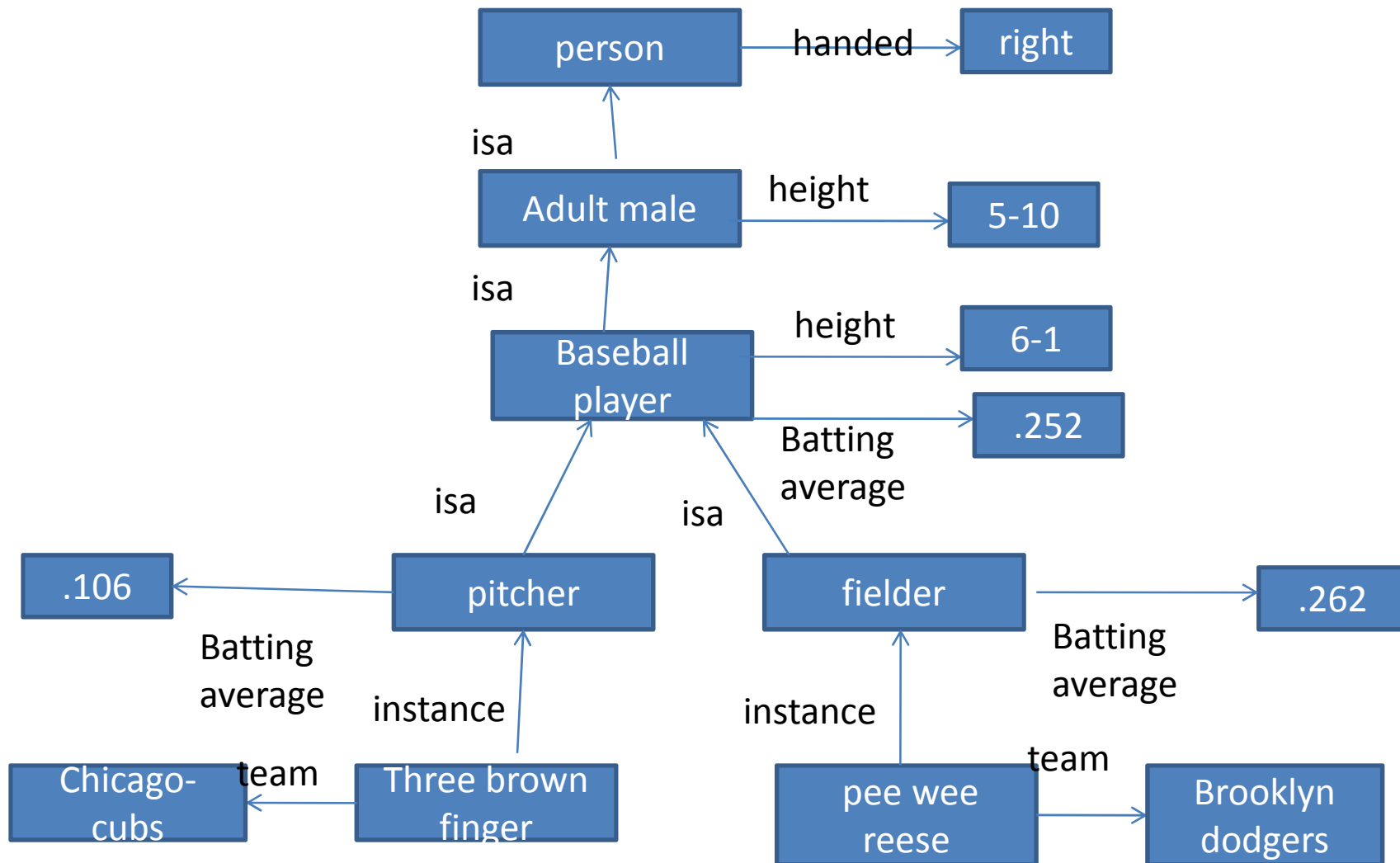
Adult male

isa: person
cardinality: 2,000,000,000
*height: 5-10

MS-baseball player

isa: adult male
cardianility: 624
*height : 6-1
*bats : equal to handed
*batting avg: 26.2
*team :
*uniform color:

Created with



Frames

- They represent general information
- Typical information may not match with general information
- Example elephant frame will define elephant with 4 legs
- If a typical elephant has 3 legs then it is not elephant at all
- Main problem
 - They allow unrestrained or alteration or cancellation of slots

Strong slot and filler structures

- Scripts
- Conceptual dependencies

Scripts

- Modified form of frames
- Help in representing stereo type of events that take place in day-to-day activities
- It contain slot with some information or default value
- Specific properties of restricted domain can be exploited with special purpose structures
- Example
 - Going to hotel
 - Going to theatre
 - Going to supermarket
 - Leaving for workplace
 - Going to bank for withdrawal of money

Scripts

- It is special case of frame structure
- Captures situation in which information is stylized
- They also have slots , in each slot we have information about slot
- It tells
 - What happens in a situation
 - What event follows
 - What role every actor play

Scripts

- Are useful because in real world there are patterns to occurrence of events
- These patterns arise because of casual relationship between events.
- These events form *casual chain* starting from entry condition and ending in result.
- If particular script is applicable in a situation then it is useful in predicting the occurrence of event not given explicitly.
- Tells the relation between events like ordering and payment are different in restaurant script.
- Before applying a script it must be activated

Activation of scripts

- Depending upon how important the script is , it can be activated in two ways
 - For fleeing script
 - One which are not central to situation
 - Are mentioned briefly
 - May be referred frequently
 - Activation – store a pointer to script so that it can be accessed later if necessary
 - Eg Susan passed her favourite restaurant on her way to museum . She enjoyed new Picasso exhibit.
 - For non fleeing script
 - Activation is full
 - Attempt to fill slots with particular objects and people involved in current situation
 - Headers can serve as an indicator that script should be activated

Activation of scripts

- Once the scripts have been activated there are variety of ways in which it can be useful for interpreting a situation like
 - Interpreting an event that has not been explicitly observed
 - Eg “ John went to restaurant last night. He ordered steak. When he paid for it he noticed that he was running out of money. He hurried home since it had started to rain.”
 - Question “ did john eat dinner last night?”
 - Ans- yes although not told explicitly
- Most of the stereotype KR structure can represent unobserved events, once they are activated ;many predictions can be made.

Activation of scripts

- Another use of script is to provide a way of building a single coherent interpretation from collection of observation.
 - Provides information as to how the events are related to each other
 - Casual chain of events
 - Eg “ Susan went out to lunch. She sat down on a table and called the waitress . The waitress brought her a menu and she ordered a hamburger.”
 - Question “ why did waitress bring Susan a menu?”
 - Ans1: Because Susan asked for it. By moving backward in casual chain
 - Ans2: So that Susan can decide what she wanted to eat. By moving forward in casual chain.

Activation of scripts

- Third way in which script is useful is that it focuses attention of unusual events
 - Eg” *John went to a restaurant. He was shown his table. He ordered a large steak. He sat there and waited for a long time. He got irritated and left.*
 - Important : place of departure
 - Typical sequence of events is interrupted
 - Hence other events which were following it cannot be interpreted . Infer :John did not made payment .
 - **Question** “ John got mad why?”
 - **Answer** Not because he was shown his table but because he waited for long time.

Components of script

- Entry condition: basic condition that must be fulfilled
- Result: what happens after script has occurred
- Props: Objects that are existing in script
- Roles: what various characters play is brought under slot (implicit or explicit)
- Track: represents a specific instance of generic pattern
- Scenes: sequence of activities are described in detail

Script : going to restaurant

Props: food

tables

menu

money

Roles: owner

customer

waiter

cashier

Entry condition

Customer is hungry

Customer has money

Owner has food

Results

Customer is not hungry

Customer has less money

Owner has more money

Owner has less food

Scene 1 entering the restaurant

Customer enter restaurant

scans the table

chooses the best food

decides to sit

goes there

occupies seat

Scene2 Ordering the food

customer ask for menu

waiter brings it

customer glances it

chooses what to eat

orders the item

Scene 3: eating food

waiter brings the food

customer eats it

Scene 4 paying the bill

ask for bill

waiter brings it

customer pays it

waiter hands over the cash to cashier

waiter brings the balance amount

customer tips him

Customer moves ou

Created with

 **nitro**^{PDF} professional

download the free trial online at nitropdf.com/professional

Advantages and disadvantages

- Advantages

- Permits one to identify what scene must have preceded, what event has taken place
- Each action or event is described to minute detail
- Gives single interpretation from variety of observation

- Disadvantages

- It is difficult to share information across scripts
- One script is true for one type of situation
- Not generalized as they represent stereo – type of information

Conceptual dependency

- It is theory of how to represent kind of knowledge about events that is usually contained in a natural language sentences.
- It is theory of NLP which mainly deals with representation of semantics of a language
- Overall goal is to represent knowledge in such a way
 - It facilitates inferences from sentences
 - Is independent of language in which the sentence were originally constructed
 - It facilitates to build question answer type of system, translation system etc.

Conceptual dependencies

- Because of these objectives CD are made of conceptual primitives which are combined to form meaning of words in a language.
- Theory was given by Shank 1973
- Unlike semantic net , CD provide both
 - structure to provide information within a sentence
 - Specific set of primitive at particular level granularity
 - Using such primitives , representation of particular piece of information can be constructed
- CD consist of two building blocks
 - CD primitives
 - CD conceptual categories from which dependency structure are built

Set of primitives

S.No	CD primitive	meaning and example
1	ATRANS	Transfer of abstract relationship (eg give)
2	PTRANS	Transfer of physical location of an object (eg go)
3	PROPEL	Application of physical force to an object (eg push)
4	MOVE	Movement of body part by its owner (eg kick)
5	GRASP	Grasping of an object by an actor (eg clutch)
6	INGEST	Ingest of an object by an animal (eg eat)
7	EXPEL	Expulsion of something from the body of an animal (eg spit)
8	MTRANS	Transfer of mental information (eg tell)
9	MBUILD	Building a anew information out of old (eg decide)
10	SPEAK	Production of sounds (eg say)
11	ATTEND	Focusing of a sense organ towards stimulus (eg listen)

Created with



download the free trial online at nitropdf.com/professional

CD building blocks

- There are four conceptual categories
 - ACTs Actions
 - PPs Picture producers, Objects
 - AAs Action aiders, modifiers of action
 - PAs Picture aiders , modifiers of PPs
- Dependency structures are conceptualizations
- They can serve as component of large CD structures
- Dependencies among conceptualization correspond to semantic relations of underlying concept.
- Some of the rules of CD as given by Shank is as follows:

Rules of CD`s

- Rule 1: relation between an actor and event he/she causes. Two way dependencies as none can be considered primary

– PP ↔ ACT John ^p ↔ PTRANS John runs.

- Rule 2: Describes relations between a PP and PA

– PP ↔ PA John ↔ height(>avg) John is tall.

- Rule 3: relation between two PPs

– PP ↔ PP John ↔ doctor John is doctor

- Rule 4: relation between a PP and an attribute that has been already predicate of it

PP boy A nice boy.
 ↑ ↑
 PA nice

- Rule 5: describes relation between two PPs, one which provide some kind of information about other .three types of such information is
 - Possession (POSS-BY)
 - Location (LOC)
 - Physical containment (CONT)

PP
↑
PP

Dog
↑ poss-by
John

Rules of CD

- Rule 6: describes relation between ACT and PP that is **object** of ACT
- Rule 7: describes relation between an ACT and the **source and recipient**
- Rule 8: describes relation between an ACT and the **instrument** with which it is performed
- Rule 9: describes relationship between an ACT and **its physical source and destination**
- Rule 10: represent relation between a PP and a state in which it **started and a state in which it ended**

Rules of CD

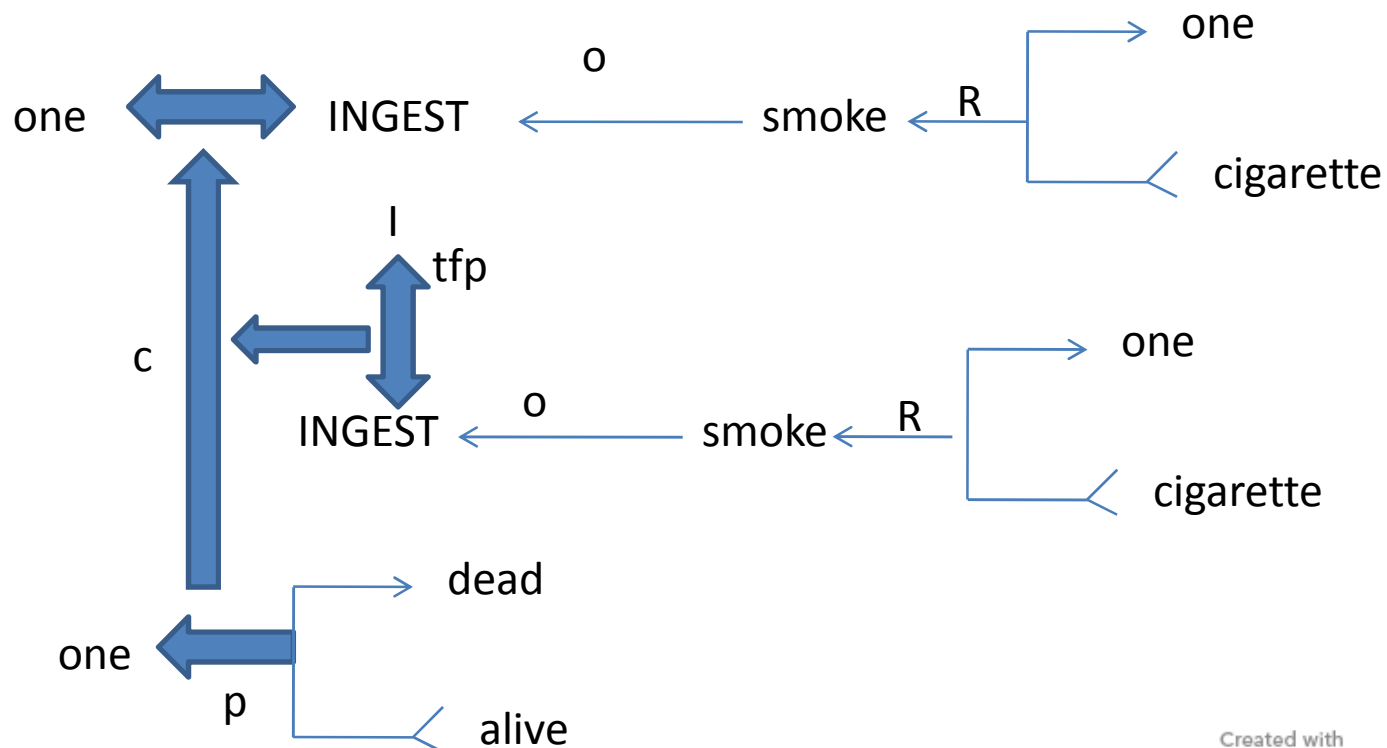
- Rule 11 describe relation between one concept and another that causes it. Arrow indicates the **dependency of one concept on another**.
- Rule 12: Describe relation between conceptualization and the **time** at which the event it describes occurs
- Rule 13: describes conceptualization another that is **time of first** .
- Rule 14 : describes the relation between a concept and **place** at which it occurred

Language information

- CD can be modified to incorporate information indicated in a language tense , mood , aspect of verb form
 - Past p
 - Future f
 - Transition t
 - Start transition ts
 - Finish transition tf
 - Continuing k
 - Interrogative ?
 - Negative /
 - Conditional c

Example

- Example for use of these tenses:
 - Since smoking can kill you , I stopped.



Inference

- There are 3 ways in which representing knowledge using conceptual CD facilitates reasoning with knowledge
 - Fewer inference rules are required
 - Many inference are contained in the representation itself
 - Initial structure that is built to represent the information contained in one sentence will have holes that need to be filled

Examples

- Birds flew.
- Bill is programmer.
- Susan gave box of candy to Joe.
- Charlie drove pickup fast.
- John met with an accident on road.