LOGIC-BASED MODELS

Dr. Norshuhani Zamin

Adapted from:

1. Stanford slides (overview)
2. DLSU previous lecture slides (overview)
3. UTP slides (propositional, predicate, unification, resolution)
4. Slideplayer.com (forward, backward chaining)

Logic-Based Models

- Logical AI involves **representing knowledge of an agent's world, its goals and the current situation by sentences in logic**.
- The agent decides what to do by inferring that a certain action or course of action is appropriate to achieve the goals.
- Roles of logic in AI:
  - (1) logic as a basis for computation
  - (2) logic for learning from a combination of data and knowledge
  - (3) logic for reasoning about the behavior of machine learning systems

Logic-Based Models

$$\begin{tikzcd} \text{World} & \text{Aspects of the real World} \arrow[r, "\textit{follows}"] \arrow[d, "\textit{semantics}"] & \text{Aspects of the real World} \arrow[d, "\textit{semantics}"] \\ \text{Representation} & \text{sentences} \arrow[r, "\textit{entails}"] & \text{sentences} \end{tikzcd}$$

Represent knowledge about the world

Reason with that knowledge

Language

- Natural Language (informal):
  - A number that is divisible by 2 and generates a remainder of 0 is called an even number.
- Programming Language (formal):
  - Python: def even(x): return x % 2 == 0
  - C++: bool even(int x) { return x % 2 == 0; }
- Logical Language (formal):
  - First-order-logic: $\forall x.Even(x) \rightarrow Divides(x, 2)$

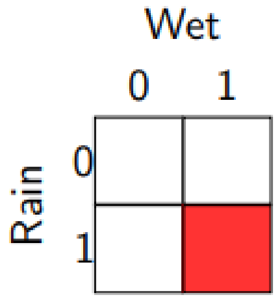
Language

- Natural Language (informal):
  - A number that is divisible by 2 and generates a remainder of 0 is called an even number.
- Programming Language (formal):
  - Python: def even(x): return x % 2 == 0
  - C++: bool even(int x) { return x % 2 == 0; }
- Logical Language (formal):
  - Propositional Logic: $\forall x.Even(x) \rightarrow Divides(x, 2)$

NOTE: Propositional logic is less expressive when the world consists many objects but English is very much expressive. Why not use English for our representation?
Answer: Natural language like English is ambiguous and not compositional

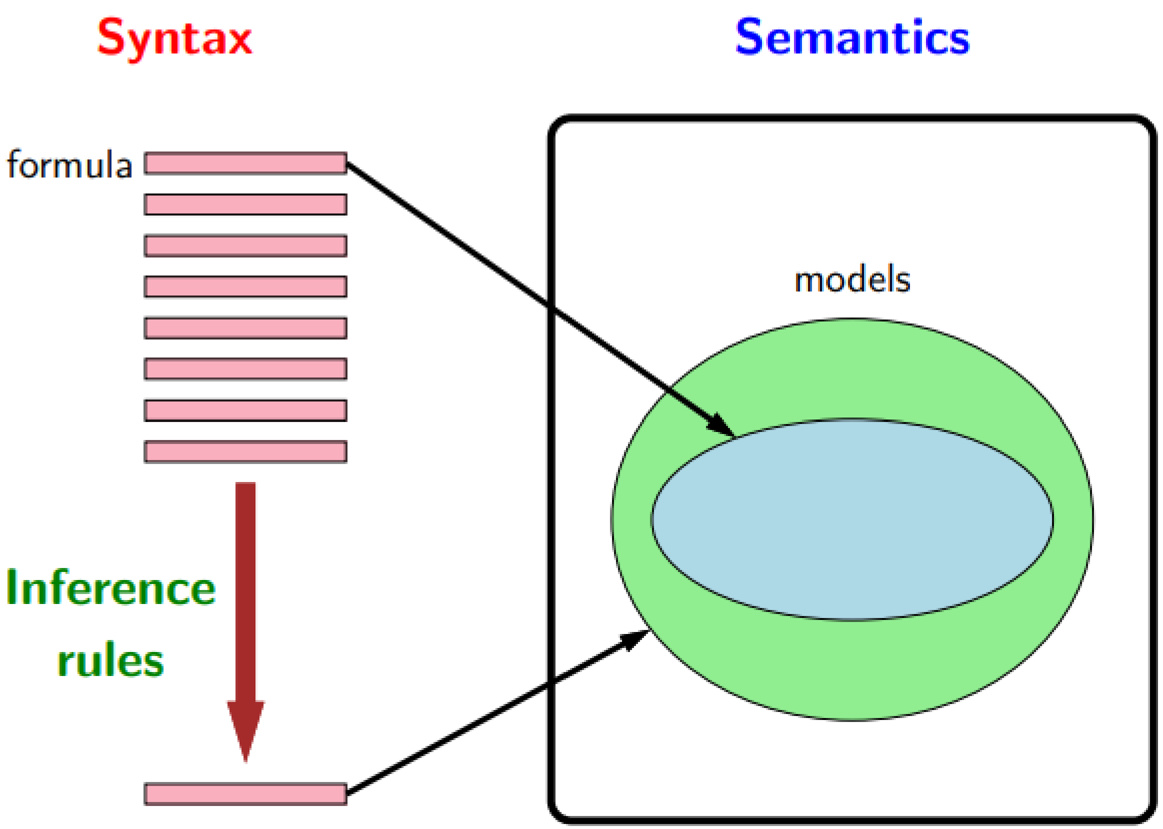
Basic of logic

- Syntax: defines a set of valid formulas (Formulas)
  - Example: Rain ∧ Wet
- Semantics: for each formula, specify a set of models (assignments / configurations of the world)
  - Example:



- Inference rules: given f, what new formulas g can be added that are guaranteed to follow?
  - Example: from Rain ∧ Wet, derive Rain

Basic of logic



Syntax: what are valid expressions in the language?
Semantics: what do these expressions mean?

Examples:

Different syntax, same semantic:
2+3 == 3+2

Same syntax, different semantic:
3 / 2 (Python ver 2.7) 6 <> 3 / 2 (Python ver 3)

Propositional Calculus

CONTENTS

- Arithmetic vs. Logic
- Propositional Calculus
  - Precedence and Associativity
  - Truth Table
  - Evaluation
  - Derivation
  - Deduction
  - Theorem Proving
- References

What can you say about the following..

Ashraff is teaching

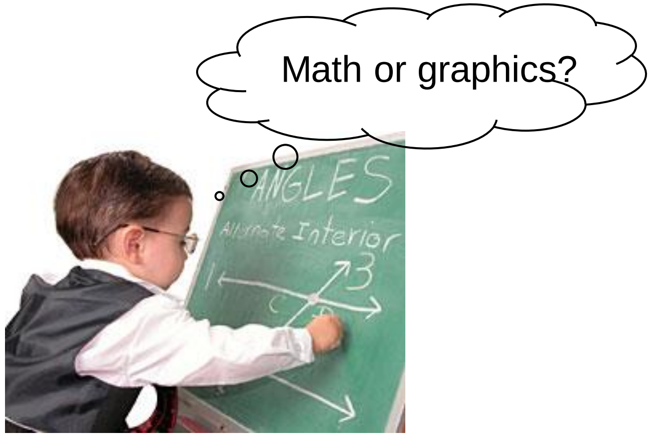
He teaches either discrete math or computer graphics

He does not teach both courses at the same time

Discrete math has many students

Computer graphics has very few students

He is teaching a course with many students this time



If my glasses were on the kitchen table, then I saw it at breakfast

I was reading the newspaper in the living room or I was reading the newspaper in the kitchen

If I were reading the newspaper in the living room, then my glasses is on the coffee table

I did not see my glasses at breakfast

If I were reading my book in bed, then my glasses is on the bed table

If I were reading the newspaper in the kitchen, then my glasses is on the kitchen table

Arithmetic vs. Logic

	ARITHMETIC	LOGIC
Basic	Variables: x,y,z,.. that are 1,2,3,..N	Propositions: p,q,r,s,.. that are either T or F
Evaluation	Operators: +, -, x, / E.g. $3 \times 4 = 12$	Connectives: ~, V, Λ, **→, ↔, ≡** E.g. $p \wedge q \rightarrow q$ i.e. $T \wedge F \rightarrow F$
Derivation	Equating from one formula to another E.g. 3x4+15/3x2=22	Equating from one propositions to another E.g. $\sim(\sim p \wedge q) \wedge (p \vee q) \equiv p$
Deduction	Solving simultaneous <u>equations</u> i.e. given a set of equations, find the values for variables.	Solving simultaneous <u>arguments</u> i.e. given a set of premises / hypothesis, find the conclusion.
Quantifying		Expand the above to include for every ∀ and some ∃ objects.

Propositional Calculus

- Proposition is a statement/sentence that is either True or False but NOT both.
    - “2 + 2 = 4” is a True proposition.
    - “2 + 2 = 5” is a False proposition.
    - “Aliff is a UTP student” is a True proposition.
    - “He is a teacher” is not a proposition.
    - “Why are you late?” is not a proposition.
    - “Get me a cup of coffee” is not a proposition.
    - “Ameer is 20 years old. He is a teacher.” is a True proposition.
    - “All dogs can fly. K9 is a dog.” is a True proposition.
- Propositional Calculus is a method of manipulating symbols and sentences.
- A sentence represented symbolically is known as well-formed formulae (wff). Simple proposition and compound proposition are wffs.

Propositional Calculus

- Simple proposition is a single proposition without connectives.
- Compound proposition is constructed from one or more simple propositions using connectives:
    - Assume simple proposition p as “Apple is fruit” is True
    - Assume simple proposition q as “Apple is red” is True

$\sim p$	Negation of p (i.e. NOT p)
$p \wedge q$	Conjunction of p and q (i.e. p AND q)
$p \vee q$	Disjunction of p and q (i.e. p OR q)
$p \rightarrow q$	Conditional if p then q (i.e. ~p V q)
$p \leftrightarrow q$	Bi-conditional p if and only if q (i.e. ((~p V q) Λ (~q V p)))

Propositional Calculus:

Precedence and Associativity

- Precedence is to determine the order in which different connectives in a compound proposition are evaluated.
- Associativity is to determine the order in which connectives with the same precedence in a compound proposition are evaluated.
- Propositional vs. programming:

precedence ↑

Programming		Propositional	
Operators	Associativity	Connectives	Associativity
* / %	Left	~	
+ -	Left	V Λ	Left
< <= > >=	Left	→ ↔	Right
== !=	Left	≡	
=	Right		

precedence ↑

- Consider the following compound proposition:
  - ~p Λ q Λ ~ r → r V s → q Λ s

Propositional Calculus:

Truth Table

- Truth table is used to determine the truth value of a compound proposition under all possible assignments of truth value to its simple propositions.
  - The truth value of a simple proposition is directly assigned:
    - Let p represents “It is raining” be true.
    - Let q represents “Ali stays at home” be true.
  - The truth value of a compound proposition is calculated from the truth value of each of the simple proposition:
    - What is the truth value of $p \rightarrow q$?
- Truth table is a way to go about evaluating compound propositions.
- Suitable if the number of simple propositions in the compound proposition is small (limit to <= 3).

Propositional Calculus:

Truth Table

Negation

p	$\sim p$
F	T
T	F

Conjunction

p	q	$p \wedge q$
F	F	F
F	T	F
T	F	F
T	T	T

Disjunction

P	q	$p \vee q$
F	F	F
F	T	T
T	F	T
T	T	T

Conditional

p	q	$p \to q$
F	F	T
F	T	T
T	F	F
T	T	T

Bi-conditional

p	q	$p \leftrightarrow q$
F	F	T
F	T	F
T	F	F
T	T	T

Propositional Calculus:

Evaluation

- Methods to construct truth value of compound statement:
  - Step 1: Resolve ambiguity of propositions (use the precedence and associativity rules)
  - Step 2: Extract simple proposition and place them on the LHS of the table.
  - Step 3: Enumerate all possible truth assignments to the simple propositions.
  - Step 4: Fill the table from “small expressions” to “big expressions”
- Example:

$\sim p \lor \sim q \rightarrow p \land r$

$= (\sim p) \lor (\sim q) \rightarrow p \land r$

$= ((\sim p) \lor (\sim q)) \rightarrow (p \land r)$

$= (((\sim p) \lor (\sim q)) \rightarrow (p \land r))$

Note:
There are 3 propositions (p,q,r) giving 2^3 possible truth value i.e. 8.

p	**q**	**r**	**~p**	**~q**	**~p V ~q**	**p Λ r**	**~p V ~q → p Λ r**
F	F	F	T	T	T	F	**F**
F	F	T	T	T	T	F	**F**
F	T	F	T	F	T	F	**F**
F	T	T	T	F	T	F	**F**
T	F	F	F	T	T	F	**F**
T	F	T	F	T	T	T	**T**
T	T	F	F	F	F	T	**T**
T	T	T	F	F	F	T	**T**

Propositional Calculus:
Derivation

- Derivation is used to find out <u>equivalence</u> of a proposition equation (proofing).
- Some definitions:
  - Tautology: A compound proposition is a tautology if it is true for <u>every</u> possible assignment to its simple proposition.
  - Contradiction: A compound proposition is a contradiction if it is false for <u>every</u> possible assignment to its simple proposition.

Propositional Calculus:

Derivation

- Logical equivalence laws are used to proof / verify proposition equations.
- The list of logical equivalence laws:

NOTE: p and q are simple propositions, t is tautology and c is contradiction.

- Commutative Laws
  - $p \vee q \equiv q \vee p$
  - $p \wedge q \equiv q \wedge p$
- Associative Laws
  - $(p \vee q) \vee r \equiv p \vee (q \vee r)$
  - $(p \wedge q) \wedge r \equiv p \wedge (q \wedge r)$
- Distributive Laws
  - $p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$
  - $p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$
- DeMorgan's Laws
  - $\sim(p \vee q) \equiv \sim p \wedge \sim q$
  - $\sim(p \wedge q) \equiv \sim p \vee \sim q$
- Contrapositive Laws
  - $p \rightarrow q \equiv \sim q \rightarrow \sim p$
- Negative Law
  - $p \vee \sim p \equiv t$
  - $p \wedge \sim p \equiv c$
- Double Negative Laws
  - $\sim(\sim p) \equiv p$
- Idempotent Laws
  - $p \vee p \equiv p$
  - $p \wedge p \equiv p$
- Universal Bound Laws
  - $p \vee t \equiv t$
  - $p \wedge c \equiv c$
- Identity Laws
  - $p \wedge t \equiv p$
  - $p \vee c \equiv p$

Propositional Calculus:

Derivation

- Use the various logical equivalence laws to verify the following logical equation:

$\sim(\sim p \land q) \land (p \lor q) \equiv p$

Answer:

$\sim(\sim p \land q) \land (p \lor q)$

$\equiv (\sim(\sim p) \lor \sim q) \land (p \lor q)$ by DeMorgan's Law

$\equiv (p \lor \sim q) \land (p \lor q)$ by Double Negative Law

$\equiv p \lor (\sim q \land q)$ by Distributive Law

$\equiv p \lor (q \land \sim q)$ by Commutative Law

$\equiv p \lor c$ by Negative Law

$\equiv p$ by Identity Law

Propositional Calculus:

Deduction

- Deduction is used to draw conclusion / inference from the given premises / hypothesis.
- Consider the following statement:
  - If Socrates is a man, then Socrates is mortal.
  - If Malaysia is a country, then Malaysia has a national day.
  - If Garfield is a cat, then Garfield can talk.
- All the above is the same as:

$$\begin{array}{l} p \rightarrow q \\ p \\ \therefore\ q \end{array} \quad \equiv \quad (p \rightarrow q) \land p \Rightarrow q$$

This is called an argument which has 2 premises, $(p \rightarrow q)$ and p, and 1 conclusion, q.

- An argument is a conditional of the form:

  $p_1 \land p_2 \land p_3 \land .. \; p_n \rightarrow r$ where $n \geq 0$ and each p_i not necessary be a simple proposition.

Propositional Calculus:

Deduction

- Acceptability of a deduction is based from the validity of the argument.
- An argument is valid when it is a tautology.

  (TIP: check the truth table when the final column contains only T)
- Examples:

$(p \rightarrow q) \wedge (p) \Rightarrow (\sim q)$

p	q	$\sim q$	$p \rightarrow q$	$(p \rightarrow q) \wedge p$	$((p \rightarrow q) \wedge p) \rightarrow (\sim q)$
F	F	T	T	F	T
F	T	F	T	F	T
T	F	T	F	F	T
T	T	F	T	T	F

$(p \rightarrow q) \wedge (p) \Rightarrow (q)$

p	q	$p \rightarrow q$	$(p \rightarrow q) \wedge p$	$((p \rightarrow q) \wedge p) \rightarrow q$
F	F	T	F	T
F	T	T	F	T
T	F	F	F	T
T	T	T	T	T

Propositional Calculus:

Deduction

- Consider the case that we have to prove:
  $(p \rightarrow r \vee s) \wedge (u \wedge \sim s \rightarrow t) \wedge (r \rightarrow t \wedge u) \Rightarrow \sim((p \vee q) \vee (\sim s \rightarrow q))$
- Complex arguments require rules of inference to derive statements until we obtain its conclusion:

Method of Affirming	$p \rightarrow q$ p $\therefore q$
Method of Denying	$p \rightarrow q$ $\sim q$ $\therefore \sim p$
Disjunctive Addition	p $\therefore p \vee q$
Conjunctive Simplification	$p \wedge q$ $\therefore p$
Conjunctive Addition	p q $\therefore p \wedge q$

Disjunctive Syllogism	$p \vee q$ $\sim q$ $\therefore p$
Hypothetical Syllogism	$p \rightarrow q$ $q \rightarrow r$ $\therefore p \rightarrow r$
Proof by division into cases	$p \vee q$ $p \rightarrow q$ $q \rightarrow r$ $\therefore r$
Rule of Contradiction	$\sim p \rightarrow c$ $\therefore p$

Propositional Calculus:

Now lets solve the problem you saw earlier:

1. If my glasses are on the kitchen table, then I saw them in breakfast
2. I was reading the newspaper in the living room or I was reading the newspaper in the kitchen
3. If I was reading the newspaper in the living room, then my glasses are on the coffee table
4. I did not see my glasses at breakfast
5. If I was reading my book in bed, then my glasses are on the bed table
6. If I was reading the newspaper in the kitchen, then my glasses are on the kitchen table

Step 1: Represent the above statements symbolically.

p: My glasses are on the kitchen table

q: I saw them in breakfast

r: I were reading the newspaper in the living room

s: I were reading the newspaper in the kitchen

t: My glasses are on the coffee table

u: I were reading my book in bed

v: My glasses are on the bed table

Step 2: Using the symbols, converts the statement into propositions.

1. $p \rightarrow q$	4. $\sim q$
2. $r \vee s$	5. $u \rightarrow v$
3. $r \rightarrow t$	6. $s \rightarrow p$

Step 3: Construct arguments and do deduction using Rules of Inference (DIY).

What have we learnt?

- What is / is not a proposition?
- Operators possible with propositions.
- Propositions in proof / logic using truth table.
- Short-cut in proof without truth table.
  - Logical Equivalence Law
  - Rules of Inference
- Propositional calculus is suitable for:
  - solving state-space problems.
  - solving path-finding problems.
  - proving theorems / algorithms.
  - solving inference problems in language understanding.

Predicate Calculus

CONTENTS

- Concept of Abstraction
- Predicate Calculus
  - Objects and Variables
  - Syntax
  - Quantified Predicate
  - Evaluation
  - Negation Equivalence
  - Unification
  - Resolution Refutation
- References

What can you say about the following..

All academic staff in UTP are teaching in each regular semester.

Ashraff is an academic staff in UTP.

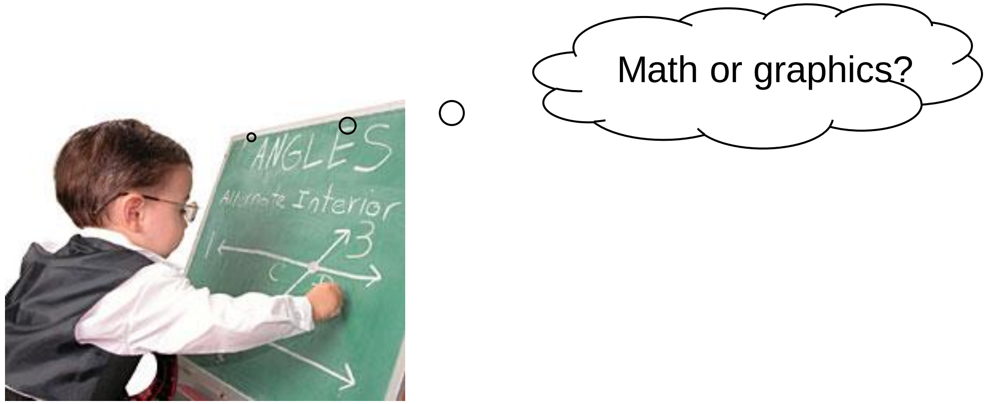
If Ashraff teaches, he can only teach discrete math or computer graphics.

Most, but not all, academic staff in UTP teach one course per semester.

All 3 credit courses have many students.

Very few specialized courses have many students.

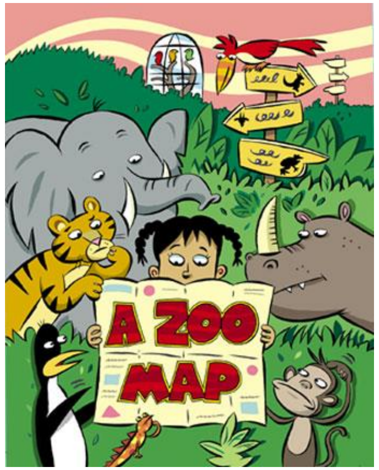
Ashraff is teaching a course with many students this time.



Concept of Abstraction

Apple is good for our health.
Pear is good for our health.
Banana is good for our health.
Strawberry is good for our health.
….

Jojo went to the zoo to see giraffes.
Jojo went to the zoo to see elephants.
Jojo went to the zoo to see tigers.
Jojo went to the zoo to see monkeys.
….



If the class is noisy then I will stop lecturing.
If Akmal talks, then the class is noisy.
If Farhana talks, then the class is noisy.
If Yin May talks, then the class is noisy.
If Thayalan talks, then the class is noisy.
If Jimmy talks, then the class is noisy.

Concept of Abstraction

Apple is good for our health.

Pear is good for our health.

Banana is good for our health.

Strawberry is good for our health.

….

For all type of fruits, they are good for our health.

Jojo went to the zoo to see giraffes.

Jojo went to the zoo to see elephants.

Jojo went to the zoo to see tigers.

Jojo went to the zoo to see monkeys.

….

Jojo went to the zoo to see all animals.

If the class is noisy then I will stop lecturing.

If Akmal talks, then the class is noisy.

If Farhana talks, then the class is noisy.

If Yin May talks, then the class is noisy.

If Thayalan talks, then the class is noisy.

If Jimmy talks, then the class is noisy.

If the class is noisy, then I will stop lecturing.

If some students talks, then the class is noisy.

Predicate Calculus:

Objects and Variables

- Variables are used to save effort in repeating the same proposition for different objects.
- Objects are physical or abstract entity referred either directly or indirectly:
  - Dr. Nini is the lecturer of CSINTSY → physical + direct
  - Dr. Nini is the lecturer of programming courses → physical + indirect
  - The set of positive integers → abstract + indirect
  - 10 → physical + direct
  - x
  - y

  → abstract + indirect

Predicate Calculus:

Syntax

- Consider these propositions:
  - Liza is a student at UTP.
  - Amirul is a student at UTP.
  - Wei Han is a student at UTP.
  - Davuth is a student at UTP.
  - Nurul is a student at UTP.

- The above propositions can be written in general as:

Variable → | X | is a student at utp | ← Predicate

- Good practice:

Use upper case to represent predicate variables and lower case letter / sentence to represent predicate and constant:

  - X: variable (student)
  - p: "is a student in utp"

p(X): X is a student at UTP.

  - X: variable (student)
  - Y: variable (university)
  - student_at : "is a student at"

student_at(X,Y): X is a student at Y.

Predicate Calculus:

Syntax

- By substituting its variables with relevant constant, predicates are reusable, meaningful and different.
  - p(liza) represents “Liza is the student at UTP”.
  - p(amirul) represents “Amirul is the student at UTP”.
  - student_at(wei han, cambridge) represents “Wei Han is the student at Cambridge“.
  - student_at(davuth, mit) represents “Davuth is the student at MIT”.
  - student_at(nurul, utp) represents “Nurul is the student at UTP”.
- Predicate is a proposition with variables that does not refer to any specific object.
- Predicate is either True or False but NOT both.
- Quantifiers are used in predicate to indicate whether some or all objects exhibit a particular property:
  - ∀: Universal quantifier to indicate all, every and each.
  - ∃: Existential quantifier to indicate some and a few.

Predicate Calculus:

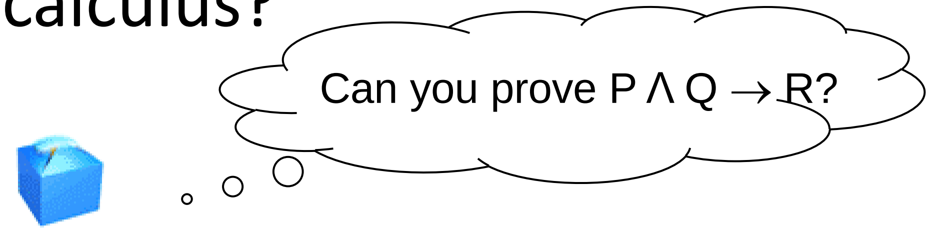
Evaluation

- Remember propositional calculus?

  P: All men are mortal

  Q: Socrates is a man

  ∴ R: Socrates is mortal



We can’t. No rules in propositional calculus will prove the above arguments.

- Need a more expressive calculus: Predicate Calculus
  - Step 1: Define all variables

    X: object(man)

    mortal: “is mortal”

    man: “is a man”

  - Step 2: Construct predicates and prove

    $\forall X \text{ man}(X) \rightarrow \text{mortal}(X)$

    man(socrates)

    ∴ mortal(socrates)

Predicate Calculus:

Quantified Predicate

- $\forall X\ p(X)$
  - **True** if p(X) is true for <u>all X.</u>
  - **False** if p(X) is false for <u>at least one X.</u>
    - *All students in this class understand the topic so far.*

      True if everyone understand the topic.

      False if we can find someone who are still lost.

- $\exists X\ p(X)$
  - **True** if p(X) is true for <u>at least one X.</u>
  - **False** if p(X) is false for <u>all X.</u>
    - *Some students in this class understand the topic so far.*

      True if there is at least one student who understand the topic.

      False if everyone do not understand the topic.

Predicate Calculus:

Quantified Predicate

- Typically $\rightarrow$ is the main connective with $\forall$.
  - Be careful about using $\wedge$ with $\forall$:
    - $\forall X$ student(X,utp) $\wedge$ smart(X)

      which means “Every student is at UTP and every student is smart”.

      This shows that the set of smart students do not relied on the set of all UTP students.
- Typically $\wedge$ is the main connective with $\exists$.
  - Be careful about using $\rightarrow$ with $\exists$:
    - $\exists X$ student(X,utp) $\rightarrow$ smart(X)

      which means “There exist at least one student in UTP who is smart”.

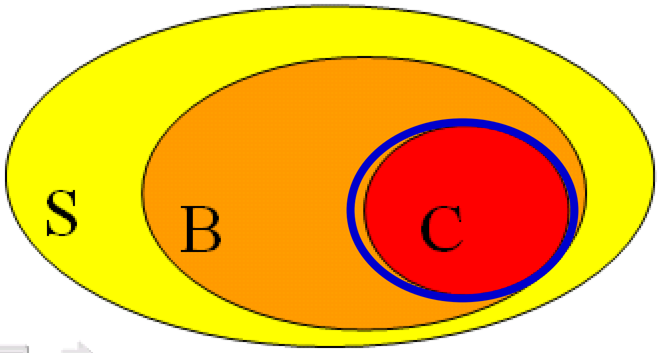
      Implication is true as long as there is only 1 student in UTP is smart.

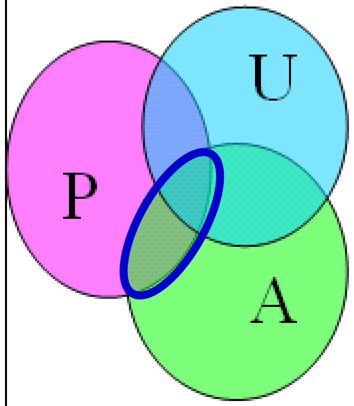
Predicate Calculus:

Quantified Predicate

- Euler Diagrams (a.k.a Vann Diagrams) helps to tell the truth set for the given predicates:

All college students are brilliant.	Some poets are unsuccessful.
All brilliant people are scientists.	Some athletes are unsuccessful.
∴ All college students are scientists.	∴ Some poets are athletes.





Predicate Calculus:
Quantified Predicate

- Nested quantified predicate is a predicate with multiple quantifiers.
- The position of quantifiers in the predicate carries different semantic (meaning):

  $\forall X\ \exists Y$ is_taken_by(X,Y) $\not\equiv$ $\exists X\ \forall Y$ is_taken_by(X,Y)

  where X = course and Y = student

  Consider:

  - *Every taught course is taken by at least one student.*
  - *Some taught courses is taken by all students.*

Predicate Calculus:

Try these..

Convert the following statements into their predicates:

- *All elephants like peanuts.*
- *Someone wrote the Deep Blue.*
- *All countries have a national day.*
- *If everyone at the UTP, then everyone is smart.*
- *Everybody loves somebody.*
- *There is flying dog.*

Evaluate the following arguments:

1. *Every computing student likes logic.*
   John is a computing student.
   Therefore, John likes logic.
2. *All dogs can fly.*
   K9 is a dog.
   Therefore, K9 can fly.

Translate the following predicates into English:

1. $\forall$X, Y father_of(X, Y) $\wedge$ female(Y) $\rightarrow$ daughter_of(Y, X)
2. $\exists$S student(S) $\wedge$ at(S, utp) $\rightarrow$ cis(S)
3. $\forall$X man(X) $\wedge$ handsome(X) $\vee$ rich(X) $\rightarrow$ love(siti, X)

Predicate Calculus:

Answers

Convert the following statements into their predicates:

- *All elephants like peanuts.*
  $\forall X$ elephant(X) $\rightarrow$like_peanut(X) **or**
  $\forall X$ elephant(X) $\rightarrow$like (X, peanut)
- *Someone wrote the Deep Blue.*
  $\exists X$ write(X, deep blue)
- *All countries have a national day.*
  $\forall X$ country(X) $\rightarrow$ has_national_day(X)
- *If everyone at the UTP, then everyone is smart.*
  $\forall S$ at(S, UTP) $\rightarrow$ smart(S)
- *Everybody loves somebody.*
  $\forall X\ \exists Y$ human(X) $\wedge$ human(Y) $\rightarrow$ love(X,Y)
- *There is no flying dog.*
  ~$\exists X$ dog(X) $\wedge$ fly(X) **or** $\forall X$ dog(X) $\rightarrow$ ~fly(X)

Predicate Calculus:

Answers

Evaluate the following arguments:

Every computing student likes logic.
John is a computing student.
Therefore, John likes logic.

Let comp_student: “is a computing student”
Let logic: “likes logic”

$\forall$S comp_student(S) $\rightarrow$ logic(S)
comp_student(john)
$\therefore$ logic(john)

* The given argument is **true.**

All dogs can fly.
K9 is a dog.
Therefore, K9 can fly.

Let dog: is a dog
Let fly: can fly

$\forall$D dog(D) $\rightarrow$ fly(D)
dog(k9)
$\therefore$ fly(k9)

* The given argument is true.

Predicate Calculus:

Negation Equivalance

- Logical equivalence laws for negation of universal and existential quantifiers is used to find the equivalent negation statement of a given statement.

  1. $\sim\exists X\ p(X) \equiv \forall X \sim p(X)$
  2. $\sim\forall X\ p(X) \equiv \exists X \sim p(X)$

- Examples:

  1. Nobody likes tax = Everybody do not like tax

     $\sim\exists X\ like\_tax(X) \equiv \forall X \sim like\_tax(X)$ * by law **#1**
  2. Not everybody play soccer = Somebody do not play soccer

     $\sim\forall X\ play(X,soccer) \equiv \exists X \sim play(X,soccer)$ * by law **#2**

Predicate Calculus:
Negation Equivalence

Consider the following statements:

- None of you understand the topic so far.
- All students have not visited Papua New Guinea.
- All American do not eat cheese burger.
- There is no honest politician in the world.

Predicate Calculus:
Unification

- Unification is a process that computes the appropriate substitution.
- The objective of unification is to find a substitution that will allow 2 statements to appear similar, thus will make the resolution refutation (inference) of the 2 statements possible.
- In unification, variables can be replaced within the same predicate with any of the following:
  - Other variables
  - Constants
  - Function expressions

Predicate Calculus:

Unification - Examples

- Unify p(a,X) and p(a,b)
  - Answer: {X/b}
- Unify p(a,X) and p(Y,b)
  - Answer: {Y/a, X/b}
- Unify p(a,X) and p(Y,f(Y))
  - Answer: {Y/a, X/f(a)}
- Unify p(a,X) and p(X,b)
  - Answer: Unification failure.
  - Justification: If {X/a} then p(a,a) $\neq$ p(a,b)
- Unify p(a,b) and p(X,X)
  - Answer: Unification failure.
  - Justification: If {X/a} then p(a,b) $\neq$ p(a,a)

Predicate Calculus:

Resolution Refutation

- Resolution refutation is an inference technique used for predicate calculus. It involves 5 main steps:
  1. Translate each statements into predicate form.
  2. Translate each predicate into clausal form.
     - **A. Eliminate the $\rightarrow$ and $\leftrightarrow$ connectives:**

       $p \rightarrow q \equiv \sim p \vee q$

       $p \wedge q \rightarrow r \equiv \sim p \vee \sim q \vee r$

       $p \wedge q \equiv \begin{array}{l} p \\ q \end{array}$

       $p \leftrightarrow q \equiv (\sim p \vee q) \wedge (\sim q \vee p)$
     - **B. Reduce the scope of negation:**

       $\sim(\sim p) \equiv p$

       $\sim(p \vee q) \equiv \sim p \wedge \sim q$

       $\sim(p \wedge q) \equiv \sim p \vee \sim q$

       $\sim\exists X\ p(X) \equiv \forall X \sim p(X)$

       $\sim\forall X\ p(X) \equiv \exists X \sim p(X)$
  3. Eliminate existential and universal quantifier.
  4. Negate the goal.
  5. Draw the resolution graph. Find contradiction and apply unification.

Predicate Calculus:

Resolution Refutation – Working Example 1

A. Eliminate the → and ↔ connectives:

$p \rightarrow q \equiv \sim p \vee q$

$p \wedge q \rightarrow r \equiv \sim p \vee \sim q \vee r$

$p \wedge q \equiv p$

q

$p \leftrightarrow q \equiv (p \rightarrow q) \vee (q \rightarrow p)$

B. Reduce the scope of negation:

$\sim(\sim p) \equiv p$

$\sim(p \vee q) \equiv \sim p \wedge \sim q$

$\sim(p \wedge q) \equiv \sim p \vee \sim q$

All dogs are animal. Fido is a dog. All animals will die. Prove that Fido will die.

Step 1

1. $\forall X$ dog(X) → animal(X).
2. dog(fido).
3. $\forall X$ animal (X) → die(X).

Step 2

1. $\forall X$ ~dog(X) V animal(X).
2. dog(fido).
3. $\forall X$ ~animal(X) V die(X).

Step 3

1. ~dog(X) V animal(X). √
2. dog(fido). √
3. ~animal(X) V die(X). √

Step 4

4. ~die(fido) √

Step 5

~~~die(fido)~~   ~animal(X) V~~die(X)~~

{X/fido}

~~~animal(fido)~~   ~dog(fido) V~~animal(fido)~~

~~dog(fido)~~   ~~~dog(fido)~~

○

Yes, Fido will die.

NOTE: Graph terminate when goal is resolved and no further progress can be made. Sometimes there can be some unused clauses which are not related to prove the goal.

Predicate Calculus:

Resolution Refutation – Working Example 2

Exciting Life

All people who are not poor and are smart are happy. Those people who read are not stupid. John can read and is wealthy. Happy people have exciting lives. Can anyone be found with an exciting life?

Predicate Calculus:

Resolution Refutation – Working Example 2

"Exciting Life"

1. All people who are not poor and are smart are happy.
   - $\forall X(\neg poor(X) \wedge smart(X) \rightarrow happy(X))$
2. Those people who read are not stupid.
   - $\forall Y(read(Y) \rightarrow smart(Y))$
   - {assume $\forall X(smart(X) \equiv \neg stupid(X))$}
3. John can read and is wealthy.
   - $read(john) \wedge \neg poor(john)$
   - {assume $\forall Y(wealthy(Y) \equiv \neg poor(Y))$}
4. Happy people have exciting lives.
   - $\forall Z(Happy(Z) \rightarrow exciting(Z))$
5. Negate the conclusion.
   - "Can anyone be found with an exciting life?"

   $\exists W \ (exciting(W))$

A. Eliminate the $\rightarrow$ and $\leftrightarrow$ connectives:

$p \rightarrow q \equiv {\sim}p \vee q$

$p \wedge q \rightarrow r \equiv {\sim}p \vee {\sim}q \vee r$

$p \wedge q \equiv p$

q

$p \leftrightarrow q \equiv (p \rightarrow q) \vee (q \rightarrow p)$

B. Reduce the scope of negation:

${\sim}({\sim}p) \equiv p$

${\sim}(p \vee q) \equiv {\sim}p \wedge {\sim}q$

${\sim}(p \wedge q) \equiv {\sim}p \vee {\sim}q$

${\sim}\exists X\ p(X) \equiv \forall X\ {\sim}p(X)$

${\sim}\forall X\ p(X) \equiv \exists X\ {\sim}p(X)$

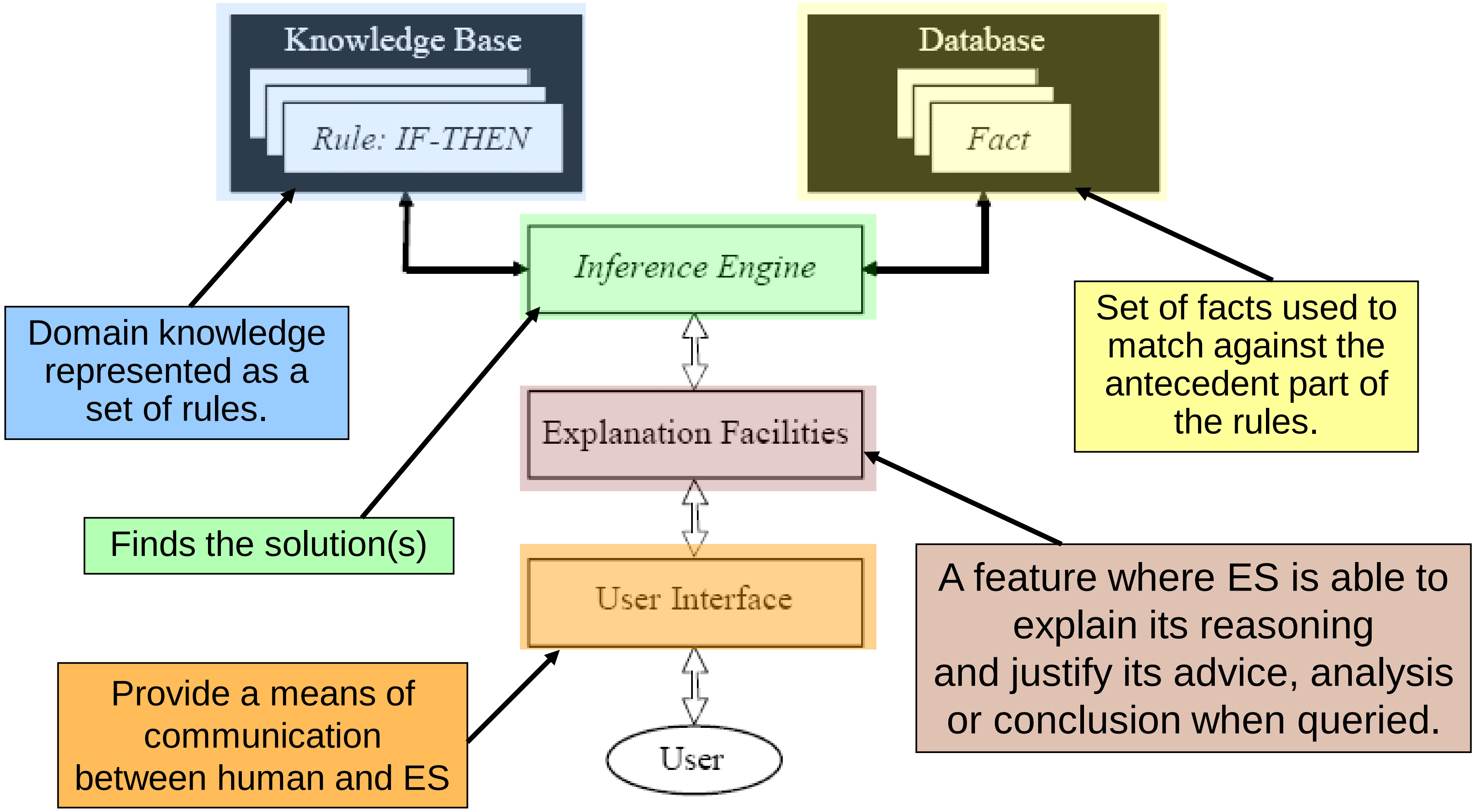
Clause forms of "exciting life"

1. $poor(X) \vee \neg smart(X) \vee happy(X)$
2. $\neg read(Y) \vee smart(Y)$
3. $read(john)$

   $\neg poor(john)$
4. $\neg happy(Z) \vee exciting(Z)$
5. $\neg exciting(W)$

Rule-Based Expert System (ES)

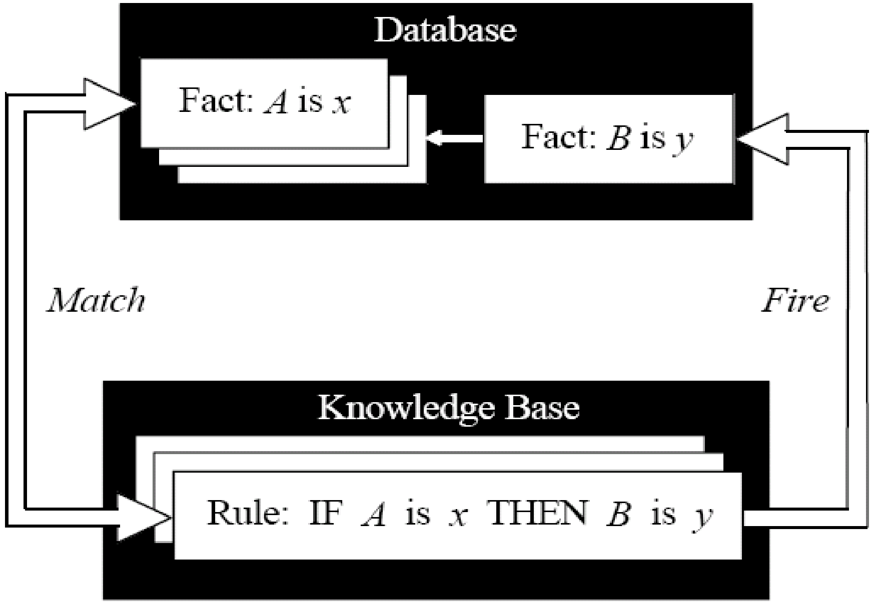
Rule-Based ES Architecture



Rule-Based ES

Inferencing method

- Domain knowledge is represented by a <u>set of IF-THEN</u> rules and data is represented by a <u>set of facts.</u>
- The inference engine compares each rule stored in the knowledge base with facts contained in the database.
- When the antecedent part of the rule matches a fact, the rule is FIRED and its consequent part is executed.
- The fired rule may change the set of facts by adding a <u>new fact</u> to the database.



Rule-Based ES

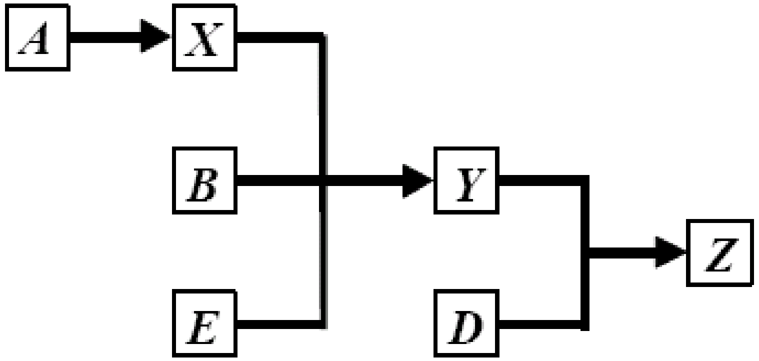
Inferencing method

- The matching of the rule antecedent parts to the facts in DB produces an inference chain:

Rule 1: IF Y is true
AND D is true
THEN Z is true

Rule 2: IF X is true
AND B is true
AND E is true
THEN Y is true

Rule 3: IF A is true
THEN X is true



Rule-Based ES

Inferencing method

- Example:

Database

larger (house, car)

larger (car, cat)

larger (house, cat)

MATCHED!

$\forall X, Y, Z$ larger $(X, Y) \wedge$ larger $(Y, Z) \rightarrow$ larger (X, Z)

FIRED!

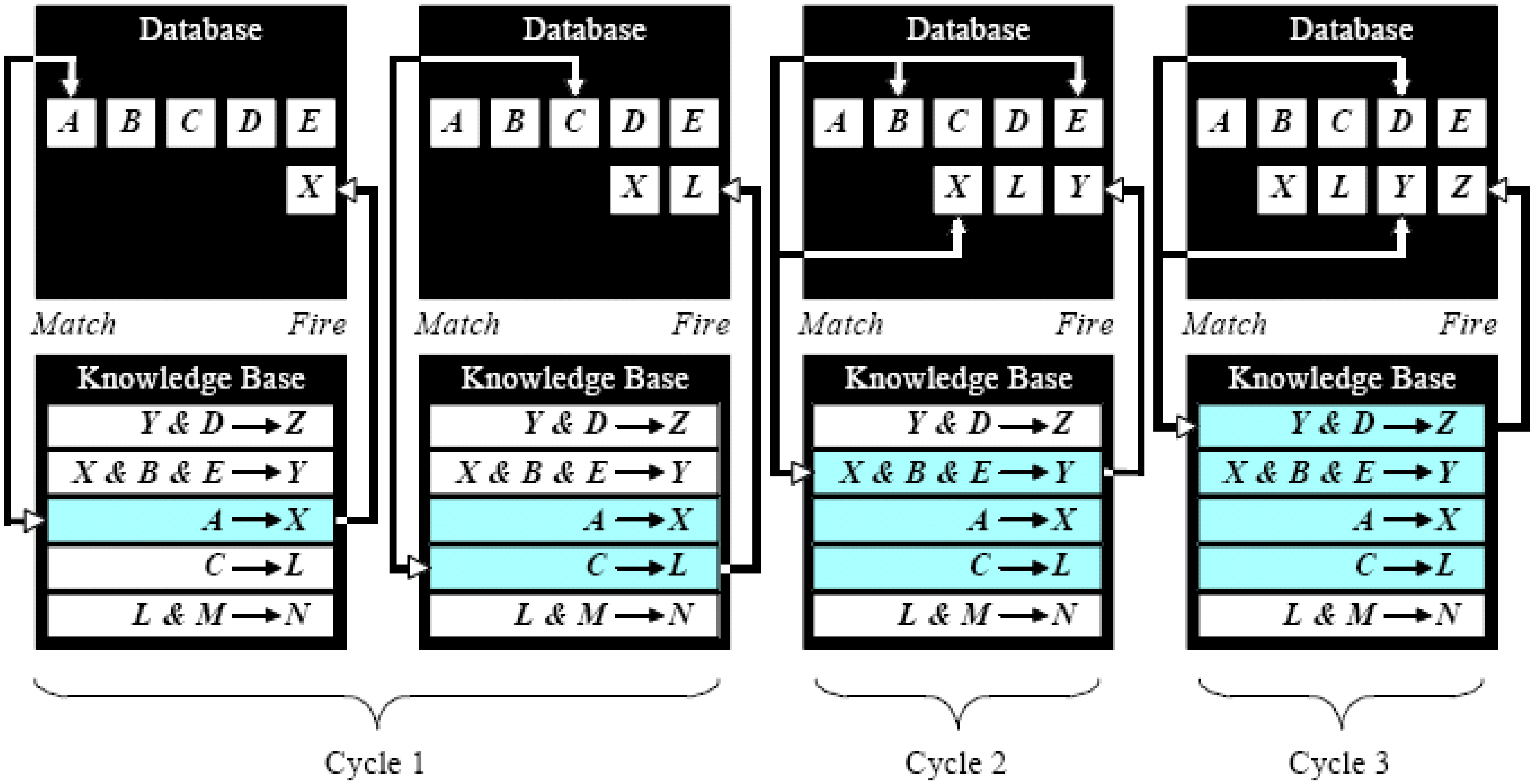
Rule-Based ES

Forward Chaining Inferencing

- Forward Chaining:
  - Also known as the **data-driven reasoning**.
  - The reasoning starts from the known data and proceeds forward with that data.
  - Any rule can be executed only once.
  - When fired, the rule adds a new fact to the database.
  - The match-fire cycle stops when no further rules can be fired.

Rule-Based ES

Forward Chaining Inferencing



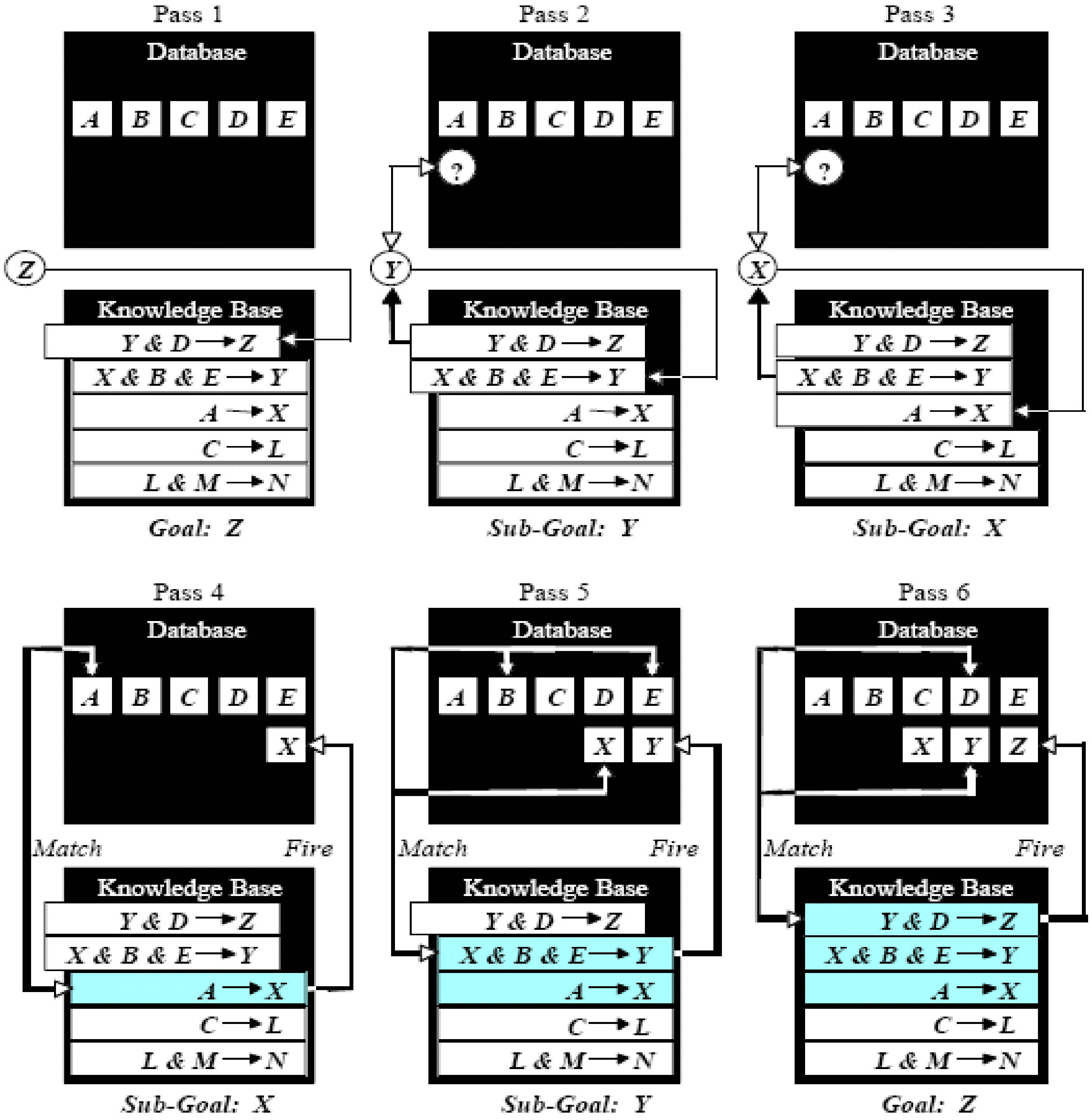
Rule-Based ES

Backward Chaining Inferencing

- Backward Chaining:
  - Also known as the **goal-driven reasoning**.
  - ES has the goal and the inference engine attempts to find the evidence to prove it.
  - First, the knowledge base is searched to find rules that might have the desired solution.
    - Such rules must have the goal in their consequent parts.
    - If such a rule is found and its antecedent part matches data in the database, then the rule is fired and the goal is proved.

Rule-Ba

Backward C



Rule-Based ES

Forward or Backward Chaining?

- If an expert first needs to gather some information and then tries to infer from it whatever can be inferred, choose the forward chaining inference engine.
  - Example: DENDRAL
- However, if your expert begins with a hypothetical solution and then attempts to find facts to prove it choose the backward chaining inference engine.
  - Example: MYCIN
- There are many ES shells use a combination of both techniques. However, the common inference method is backward chaining.

EXAMPLE:

You are given a set of rules. Assume that an investor has USD 10000 and she is 25 years old. Using forward chaining inference technique, advice whether she should get a pension plan or not.
RULE 1: IF a person has USD 10000 to invest AND has a degree THEN the person should invest in real estate
RULE 2: IF a person’s annual income >= USD 50000 AND has a degree THEN the person should invest in retirement planning
RULE 3: IF a person is younger than 30 AND is investing in real estate THEN the person should also invest in retirement planning
RULE 4: IF a person is younger than 30 AND older than 22 THEN the person has a degree
RULE 5: IF a person wants to invest in retirement planning THEN the retirement planning is the pension plan