CPE/CSC 481: Knowledge-Based Systems

Franz J. Kurfess

Computer Science Department
California Polytechnic State University
San Luis Obispo, CA, U.S.A.

Usage of the Slides

- these slides are intended for the students of my CPE/CSC 481 “Knowledge-Based Systems” class at Cal Poly SLO
  - if you want to use them outside of my class, please let me know (fkurfess@calpoly.edu)
- I usually put together a subset for each quarter as a “Custom Show”
  - to view these, go to “Slide Show => Custom Shows”, select the respective quarter, and click on “Show”
  - in Apple Keynote (.key files), I use the “Skip” feature to achieve similar results
- To print them, I suggest to use the “Handout” option
  - 4, 6, or 9 per page works fine
  - Black & White should be fine; there are few diagrams where color is important

Overview
Logic and Reasoning

- Motivation
- Objectives
- Knowledge and Reasoning
  - logic as prototypical reasoning system
  - syntax and semantics
  - validity and satisfiability
  - logic languages
- Reasoning Methods
  - propositional and predicate calculus
  - inference methods
- Reasoning in Knowledge-Based Systems
  - shallow and deep reasoning
  - forward and backward chaining
  - rule-based systems
  - alternative inference methods
  - meta-knowledge
- Important Concepts and Terms
- Chapter Summary

Motivation

- without reasoning, knowledge-based systems would be practically worthless
  - derivation of new knowledge
  - examination of the consistency or validity of existing knowledge
- reasoning in KBS can perform certain tasks better than humans
  - reliability, availability, speed
  - also some limitations
    - common-sense reasoning
    - complex inferences

Objectives

- be familiar with the essential concepts of logic and reasoning
  - sentence, operators, syntax, semantics, inference methods
- appreciate the importance of reasoning for knowledge-based systems
  - generating new knowledge
  - explanations
- understand the main methods of reasoning used in KBS
  - shallow and deep reasoning
  - forward and backward chaining
- evaluate reasoning methods for specific tasks and scenarios
- apply reasoning methods to simple problems

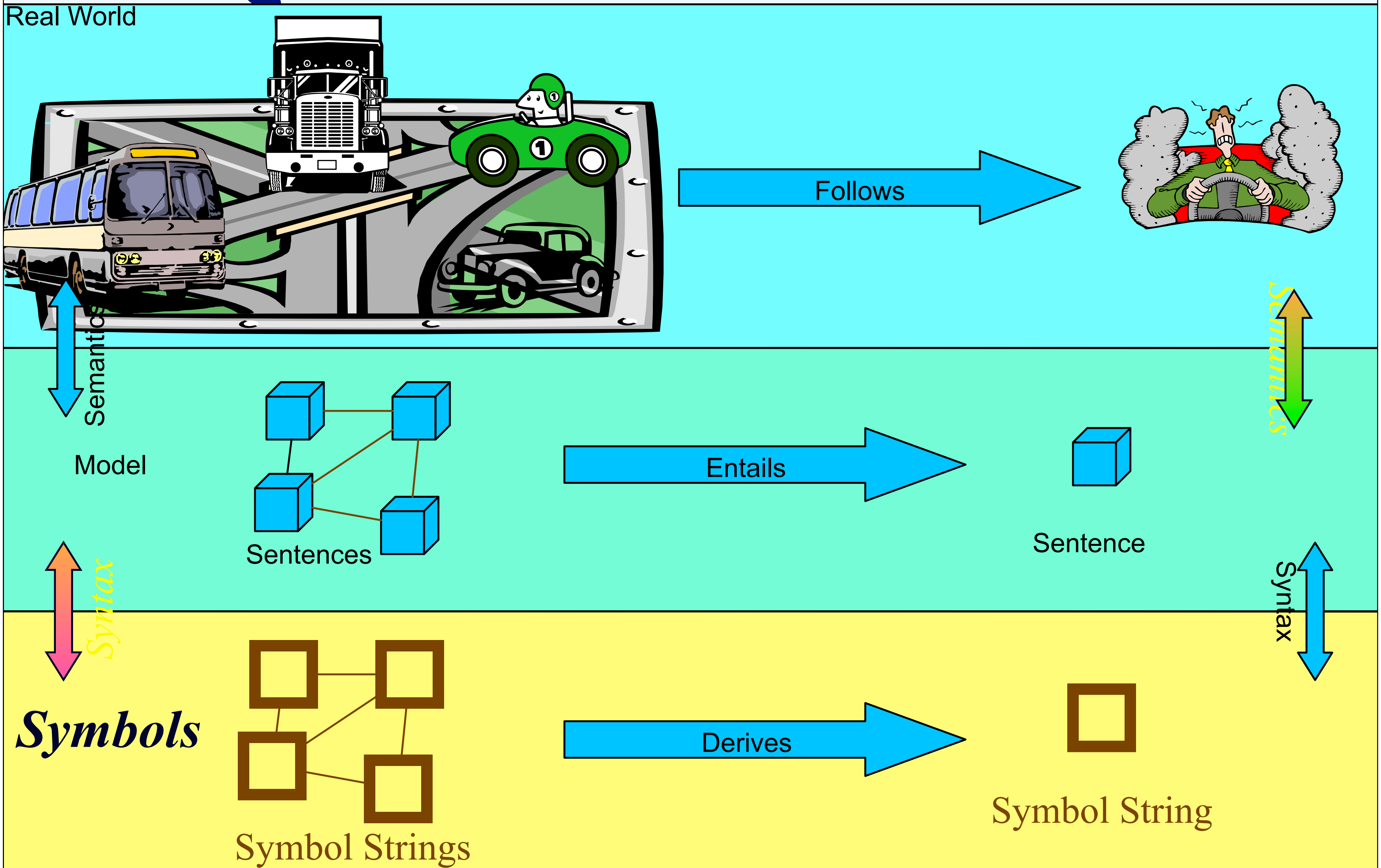
Knowledge Representation Languages

- syntax
  - sentences of the language that are built according to the syntactic rules
  - some sentences may be nonsensical, but syntactically correct
- semantics
  - refers to the facts about the world for a specific sentence
  - interprets the sentence in the context of the world
  - provides meaning for sentences
- languages with precisely defined syntax and semantics can be called logics

Sentences and the Real World

- syntax
  - describes the principles for constructing and combining sentences
    - e.g. BNF grammar for admissible sentences
    - inference rules to derive new sentences from existing ones
- semantics
  - establishes the relationship between a sentence and the aspects of the real world it describes
  - can be checked directly by comparing sentences with the corresponding objects in the real world
    - not always feasible or practical
  - compositional semantics
    - complex sentences can be checked by examining their individual parts

Diagram: Sentences and the Real



Introduction to Logic

- expresses knowledge in a particular mathematical notation

  `All birds have wings --> ¥x. Bird(x) -> HasWings(x)`

- rules of inference
  - guarantee that, given true facts or premises, the new facts or premises derived by applying the rules are also true

  `All robins are birds --> ¥x Robin(x) -> Bird(x)`

- given these two facts, application of an inference rule gives:

  `¥x Robin(x) -> HasWings(x)`

Logic and Knowledge

- rules of inference act on the superficial structure or syntax of the first two sentences
  - doesn't say anything about the meaning of birds and robins
  - could have substituted mammals and elephants etc.
- major advantages of this approach
  - deductions are guaranteed to be correct to an extent that other representation schemes have not yet reached
  - easy to automate derivation of new facts
- problems
  - computational efficiency
  - uncertain, incomplete, imprecise knowledge

Summary of Logic Languages

- propositional logic
  - facts
  - true/false/unknown
- first-order logic
  - facts, objects, relations
  - true/false/unknown
- temporal logic
  - facts, objects, relations, times
  - true/false/unknown
- probability theory
  - facts
  - degree of belief [0..1]
- fuzzy logic
  - degree of truth
  - degree of belief [0..1]

Propositional Logic

- Syntax
- Semantics
- Validity and Inference
- Models
- Inference Rules
- Complexity

Syntax

- symbols
  - logical constants True, False
  - propositional symbols P, Q, …
  - logical connectives
    - conjunction $\wedge$, disjunction $\vee$,
    - negation $\neg$,
    - implication $\Rightarrow$, equivalence $\Leftrightarrow$
  - parentheses (,)
- sentences
  - constructed from simple sentences
  - conjunction, disjunction, implication, equivalence, negation

BNF Grammar Propositional Logic

Sentence → *AtomicSentence* | *ComplexSentence*

AtomicSentence → `True` | `False` | `P` | `Q` | `R` | `...`

ComplexSentence → (*Sentence*)

| *Sentence Connective Sentence*

| ¬ *Sentence*

Connective → ∧ | ∨ | ⇒ | ⇔

ambiguities are resolved through precedence ¬ ∧ ∨ ⇒ ⇔ or parentheses

e.g. ¬ *P* ∨ *Q* ∧ *R* ⇒ *S* is equivalent to (¬ *P*) ∨ (*Q* ∧ *R*)) ⇒ *S*

Semantics

- interpretation of the propositional symbols and constants
  - symbols can be any arbitrary fact
    - sentences consisting of only a propositional symbols are satisfiable, but not valid
  - the constants True and False have a fixed interpretation
    - True indicates that the world is as stated
    - False indicates that the world is not as stated
- specification of the logical connectives
  - frequently explicitly via truth tables

Validity and Satisfiability

- a sentence is valid or necessarily true if and only if it is true under all possible interpretations in all possible worlds
  - also called a tautology
  - since computers reason mostly at the syntactic level, valid sentences are very important
    - interpretations can be neglected
- a sentence is satisfiable iff there is some interpretation in some world for which it is true
- a sentence that is not satisfiable is unsatisfiable
  - also known as a contradiction

Truth Tables for Connectives

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
False	False	True	False	False	True	True
False	True	True	False	True	True	False
True	False	False	False	True	False	False
True	True	False	True	True	True	True

Validity and Inference

- truth tables can be used to test sentences for validity
  - one row for each possible combination of truth values for the symbols in the sentence
  - the final value must be True for every sentence

Propositional Calculus

- properly formed statements that are either True or False
- syntax
  - logical constants, True and False
  - proposition symbols such as P and Q
  - logical connectives: and ^, or V, equivalence <=>, implies => and not ~
  - parentheses to indicate complex sentences
- sentences in this language are created through application of the following rules
  - True and False are each (atomic) sentences
  - Propositional symbols such as P or Q are each (atomic) sentences
  - Enclosing symbols and connective in parentheses yields (complex) sentences, e.g., (P ^ Q)

Complex Sentences

- Combining simpler sentences with logical connectives yields complex sentences
  - conjunction
    - sentence whose main connective is and: P ^ (Q V R)
  - disjunction
  - sentence whose main connective is or: A V (P ^ Q)
  - implication (conditional)
    - sentence such as (P ^ Q) => R
    - the left hand side is called the premise or antecedent
    - the right hand side is called the conclusion or consequent
    - implications are also known as rules or if-then statements
  - equivalence (biconditional)
    - (P ^ Q) <=> (Q ^ P)
  - negation
    - the only unary connective (operates only on one sentence)
    - e.g., ~P

Syntax of Propositional Logic

- A BNF (Backus-Naur Form) grammar of sentences in propositional logic

```
Sentence -> AtomicSentence | ComplexSentence

  AtomicSentence -> True | False | P | Q | R | ...

  ComplexSentence -> (Sentence)
                   | Sentence Connective Sentence
                   | ~Sentence

  Connective -> ^ | V | <=> | =>
```

Semantics

- propositions can be interpreted as any facts you want
  - e.g., P means "robins are birds", Q means "the wumpus is dead", etc.
- meaning of complex sentences is derived from the meaning of its parts
  - one method is to use a truth table
  - all are easy except P => Q
    - this says that if P is true, then I claim that Q is true; otherwise I make no claim;
    - P is true and Q is true, then P => Q is true
    - P is true and Q is false, then P => Q is false
    - P is false and Q is true, then P => Q is true
    - P is false and Q is false, then P => Q is true

Inference Rules

- more efficient than truth tables

Modus Ponens

- eliminates =>

  (X => Y), X

  ———————————

  Y

  - If it rains, then the streets will be wet.
  - It is raining.
  - Infer the conclusion: The streets will be wet.
    - (affirms the antecedent)

Modus tollens

$$\frac{(X \Rightarrow Y),\ \sim Y}{\neg X}$$

- If it rains, then the streets will be wet.
- The streets are not wet.
- Infer the conclusion: It is not raining.

- NOTE: Avoid the fallacy of affirming the consequent:
  - If it rains, then the streets will be wet.
  - The streets are wet.
  - cannot conclude that it is raining.

  - If Bacon wrote Hamlet, then Bacon was a great writer.
  - Bacon was a great writer.
  - cannot conclude that Bacon wrote Hamlet.

Syllogism

- chain implications to deduce a conclusion

$$\frac{(X \Rightarrow Y),\ (Y \Rightarrow Z)}{(X \Rightarrow Z)}$$

More Inference Rules

- and-elimination
- and-introduction
- or-introduction
- double-negation elimination
- unit resolution

Resolution

$$\frac{(X \lor Y),\ (\sim Y \lor Z)}{(X \lor Z)}$$

- basis for the inference mechanism in the Prolog language and some theorem provers

Complexity issues

- truth table enumerates 2n rows of the table for any proof involving n symbol
  - it is complete
  - computation time is exponential in n
- checking a set of sentences for satisfiability is NP-complete
  - but there are some circumstances where the proof only involves a small subset of the KB, so can do some of the work in polynomial time
  - if a KB is monotonic (i.e., even if we add new sentences to a KB, all the sentences entailed by the original KB are still entailed by the new larger KB), then you can apply an inference rule locally (i.e., don't have to go checking the entire KB)

Inference Methods 1

- deduction sound
  - conclusions must follow from their premises; prototype of logical reasoning
- induction unsound
  - inference from specific cases (examples) to the general
- abduction unsound
  - reasoning from a true conclusion to premises that may have caused the conclusion
- resolution sound
  - find two clauses with complementary literals, and combine them
- generate and test unsound
  - a tentative solution is generated and tested for validity
  - often used for efficiency (trial and error)

Inference Methods 2

- default reasoning unsound
  - general or common knowledge is assumed in the absence of specific knowledge
- analogy unsound
  - a conclusion is drawn based on similarities to another situation
- heuristics unsound
  - rules of thumb based on experience
- intuition unsound
  - typically human reasoning method
- nonmonotonic reasoning unsound
  - new evidence may invalidate previous knowledge
- autoepistemic unsound
  - reasoning about your own knowledge

Predicate Logic

- new concepts (in addition to propositional logic)
  - complex objects
    - terms
  - relations
    - predicates
    - quantifiers
  - syntax
  - semantics
  - inference rules
  - usage

Objects

- distinguishable things in the real world
  - people, cars, computers, programs, ...
- frequently includes concepts
  - colors, stories, light, money, love, ...
- properties
  - describe specific aspects of objects
    - green, round, heavy, visible,
  - can be used to distinguish between objects

Relations

- establish connections between objects
- relations can be defined by the designer or user
  - neighbor, successor, next to, taller than, younger than, …
- functions are a special type of relation
  - non-ambiguous: only one output for a given input

CAL POLY

Syntax

- also based on sentences, but more complex
  - sentences can contain terms, which represent objects
- constant symbols: A, B, C, Franz, Square1,3, …
  - stand for unique objects (in a specific context)
- predicate symbols: Adjacent-To, Younger-Than, ...
  - describes relations between objects
- function symbols: Father-Of, Square-Position, …
  - the given object is related to exactly one other object

CAL POLY

Semantics

- provided by interpretations for the basic constructs
  - usually suggested by meaningful names
- constants
  - the interpretation identifies the object in the real world
- predicate symbols
  - the interpretation specifies the particular relation in a model
  - may be explicitly defined through the set of tuples of objects that satisfy the relation
- function symbols
  - identifies the object referred to by a tuple of objects
  - may be defined implicitly through other functions, or explicitly through tables

BNF Grammar Predicate Logic

Sentence → *AtomicSentence*

| *Sentence Connective Sentence*

| *Quantifier Variable, ... Sentence*

| ¬ *Sentence* | (*Sentence*)

AtomicSentence → *Predicate(Term, ...)* | *Term = Term*

Term → *Function(Term, ...)* | *Constant* | *Variable*

Connective → ∧ | ∨ | ⇒ | ⇔

Quantifier → ∀ | ∃

Constant → *A, B, C,* X_1, X_2, *Jim, Jack*

Variable → *a, b, c,* x_1, x_2, *counter, position*

Predicate → *Adjacent-To, Younger-Than,*

Function → *Father-Of, Square-Position, Sqrt, Cosine*

ambiguities are resolved through precedence or parentheses

Terms

- logical expressions that specify objects
- constants and variables are terms
- more complex terms are constructed from function symbols and simpler terms, enclosed in parentheses
  - basically a complicated name of an object
- semantics is constructed from the basic components, and the definition of the functions involved
  - either through explicit descriptions (e.g. table), or via other functions

Unification

- an operation that tries to find consistent variable bindings (substitutions) for two terms
  - a substitution is the simultaneous replacement of variable instances by terms, providing a “binding” for the variable
  - without unification, the matching between rules would be restricted to constants
  - often used together with the resolution inference rule
  - unification itself is a very powerful and possibly complex operation
    - in many practical implementations, restrictions are imposed
      - e.g. substitutions may occur only in one direction (“matching”)

Atomic Sentences

- state facts about objects and their relations
- specified through predicates and terms
  - the predicate identifies the relation, the terms identify the objects that have the relation
- an atomic sentence is true if the relation between the objects holds
  - this can be verified by looking it up in the set of tuples that define the relation

CAL POLY

Complex Sentences

- logical connectives can be used to build more complex sentences
- semantics is specified as in propositional logic

Quantifiers

- can be used to express properties of collections of objects
  - eliminates the need to explicitly enumerate all objects
- predicate logic uses two quantifiers
  - universal quantifier $\forall$
  - existential quantifier $\exists$

Universal Quantification

- states that a predicate P is holds for all objects x in the universe under discourse
  $\forall x\ P(x)$
- the sentence is true if and only if all the individual sentences where the variable x is replaced by the individual objects it can stand for are true

Existential Quantification

- states that a predicate P holds for some objects in the universe
  $\exists x \; P(x)$
- the sentence is true if and only if there is at least one true individual sentence where the variable x is replaced by the individual objects it can stand for

Horn clauses or sentences

- class of sentences for which a polynomial-time inference procedure exists
  - $P1 \wedge P2 \wedge \ldots \wedge Pn \Rightarrow Q$
  - where Pi and Q are non-negated atomic sentences
- not every knowledge base can be written as a collection of Horn sentences
- Horn clauses are essentially rules of the form
  - If $P1 \wedge P2 \wedge \ldots \wedge Pn$ then Q

Reasoning in Knowledge-Based Systems

- shallow and deep reasoning
- forward and backward chaining
- alternative inference methods
- metaknowledge

Shallow and Deep Reasoning

- shallow reasoning
  - also called experiential reasoning
  - aims at describing aspects of the world heuristically
  - short inference chains
  - possibly complex rules
- deep reasoning
  - also called causal reasoning
  - aims at building a model of the world that behaves like the "real thing"
  - long inference chains
  - often simple rules that describe cause and effect relationships

Examples Shallow and Deep Reasoning

- shallow reasoning

```
IF a car has
   a good battery
   good spark plugs
   gas
   good tires
  THEN the car can move
```

- deep reasoning

```
IF the battery is good
THEN there is electricity

IF there is electricity AND
   good spark plugs
THEN the spark plugs will fire

IF the spark plugs fire AND
   there is gas
THEN the engine will run

IF the engine runs AND
   there are good tires
THEN the car can move
```

Forward Chaining

- given a set of basic facts, we try to derive a conclusion from these facts
- example: What can we conjecture about Clyde?

```
IF elephant(x) THEN mammal(x)
IF mammal(x) THEN animal(x)
elephant (Clyde)
```

modus ponens:

```
IF p THEN q
p
---------------
q
```

unification:

find compatible values for variables

Forward Chaining Example

```
IF elephant(x) THEN mammal(x)
IF mammal(x) THEN animal(x)
elephant(Clyde)
```

unification: find compatible values for variables

modus ponens:

```
IF p THEN q
p
-----------
q
```

```
IF elephant(   x   ) THEN mammal(   x   )
```

```
elephant (Clyde)
```

Forward Chaining Example

```
IF elephant(x) THEN mammal(x)
IF mammal(x) THEN animal(x)
elephant(Clyde)
```

unification: find compatible values for variables

modus ponens:

```
IF p THEN q
p
-----------
q
```

```
IF elephant(Clyde) THEN mammal(Clyde)
```

```
elephant (Clyde)
```

Forward Chaining Example

```
IF elephant(x) THEN mammal(x)
IF mammal(x) THEN animal(x)
elephant(Clyde)
```

unification:
find compatible values for variables

modus ponens:

```
IF p THEN q
p
q
```

```
IF mammal(  x  ) THEN animal(  x  )
```

```
IF elephant(Clyde) THEN mammal(Clyde)
```

```
elephant (Clyde)
```

Forward Chaining Example

```
IF elephant(x) THEN mammal(x)
IF mammal(x) THEN animal(x)
elephant(Clyde)
```

unification:
find compatible values for variables

modus ponens:

```
IF p THEN q
p
-----------
q
```

```
IF mammal(Clyde) THEN animal(Clyde)
IF elephant(Clyde) THEN mammal(Clyde)
elephant (Clyde)
```

Forward Chaining Example

```
IF elephant(x) THEN mammal(x)
IF mammal(x) THEN animal(x)
elephant(Clyde)
```

unification:
find compatible values for variables

modus ponens:

```
IF p THEN q
p
-----------
q
```

```
animal(  x  )
IF mammal(Clyde) THEN animal(Clyde)
IF elephant(Clyde) THEN mammal(Clyde)
elephant (Clyde)
```

CAL POLY

Forward Chaining Example

```
IF elephant(x) THEN mammal(x)
IF mammal(x) THEN animal(x)
elephant(Clyde)
```

unification:

find compatible values for variables

modus ponens:

```
IF p THEN q
p
```

```
q
```

```
animal(Clyde)
IF mammal(Clyde) THEN animal(Clyde)
IF elephant(Clyde) THEN mammal(Clyde)
elephant (Clyde)
```

Backward Chaining

- try to find supportive evidence (i.e. facts) for a hypothesis
- example: Is there evidence that Clyde is an animal?

```
IF elephant(x) THEN mammal(x)
IF mammal(x) THEN animal(x)
elephant (Clyde)
```

modus ponens:

```
IF p THEN q
p
```

```
q
```

unification:

find compatible values for variables

Backward Chaining Example

```
IF elephant(x) THEN mammal(x)
IF mammal(x) THEN animal(x)
elephant(Clyde)
```

unification: find compatible values for variables

modus ponens:

```
IF p THEN q
p
-----------
q
```

```
animal(Clyde)
```

?

```
IF mammal(  x  ) THEN animal(  x  )
```

Backward Chaining Example

```
IF elephant(x) THEN mammal(x)
IF mammal(x) THEN animal(x)
elephant(Clyde)
```

unification:
find compatible values for variables

modus ponens:

```
IF p THEN q
p
---
q
```

```
animal(Clyde)
```

?

```
IF mammal(Clyde) THEN animal(Clyde)
```

Backward Chaining Example

```
IF elephant(x) THEN mammal(x)
IF mammal(x) THEN animal(x)
elephant(Clyde)
```

unification:
find compatible values for variables

modus ponens:

```
IF p THEN q
p
q
```

```
animal(Clyde)
```

?

```
IF mammal(Clyde) THEN animal(Clyde)
```

?

```
IF elephant(   x   ) THEN mammal(   x   )
```

Backward Chaining Example

```
IF elephant(x) THEN mammal(x)
IF mammal(x) THEN animal(x)
elephant(Clyde)
```

unification:
find compatible values for variables

modus ponens:
```
IF p THEN q
p
-----------
q
```

```
animal(Clyde) ?
IF mammal(Clyde) THEN animal(Clyde) ?
IF elephant(Clyde) THEN mammal(Clyde)
```

Backward Chaining Example

```
IF elephant(x) THEN mammal(x)
IF mammal(x) THEN animal(x)
elephant(Clyde)
```

unification: find compatible values for variables

modus ponens:

```
IF p THEN q
p
-----------
q
```

```
animal(Clyde)
IF mammal(Clyde) THEN animal(Clyde)
IF elephant(Clyde) THEN mammal(Clyde)
elephant ( x )
```

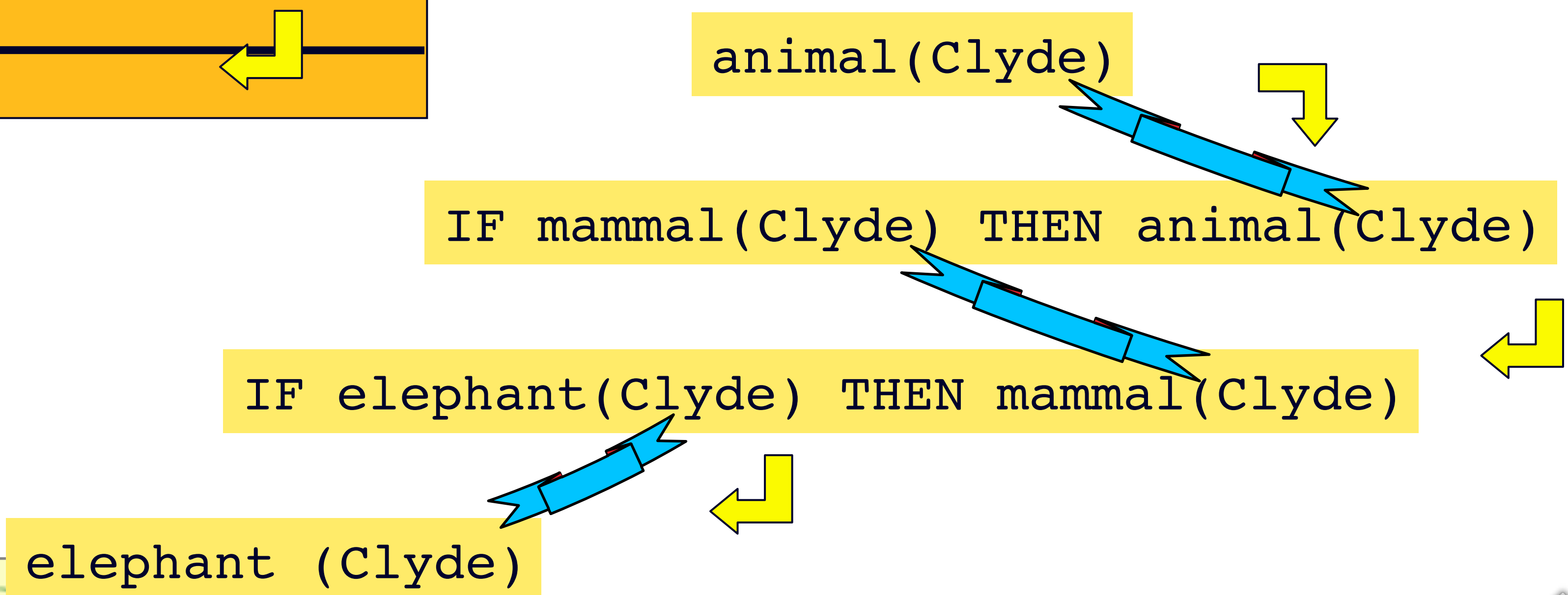
Backward Chaining Example

```
IF elephant(x) THEN mammal(x)
IF mammal(x) THEN animal(x)
elephant(Clyde)
```

unification: find compatible values for variables

modus ponens:

```
IF p THEN q
p
---
q
```



Forward vs. Backward Chaining

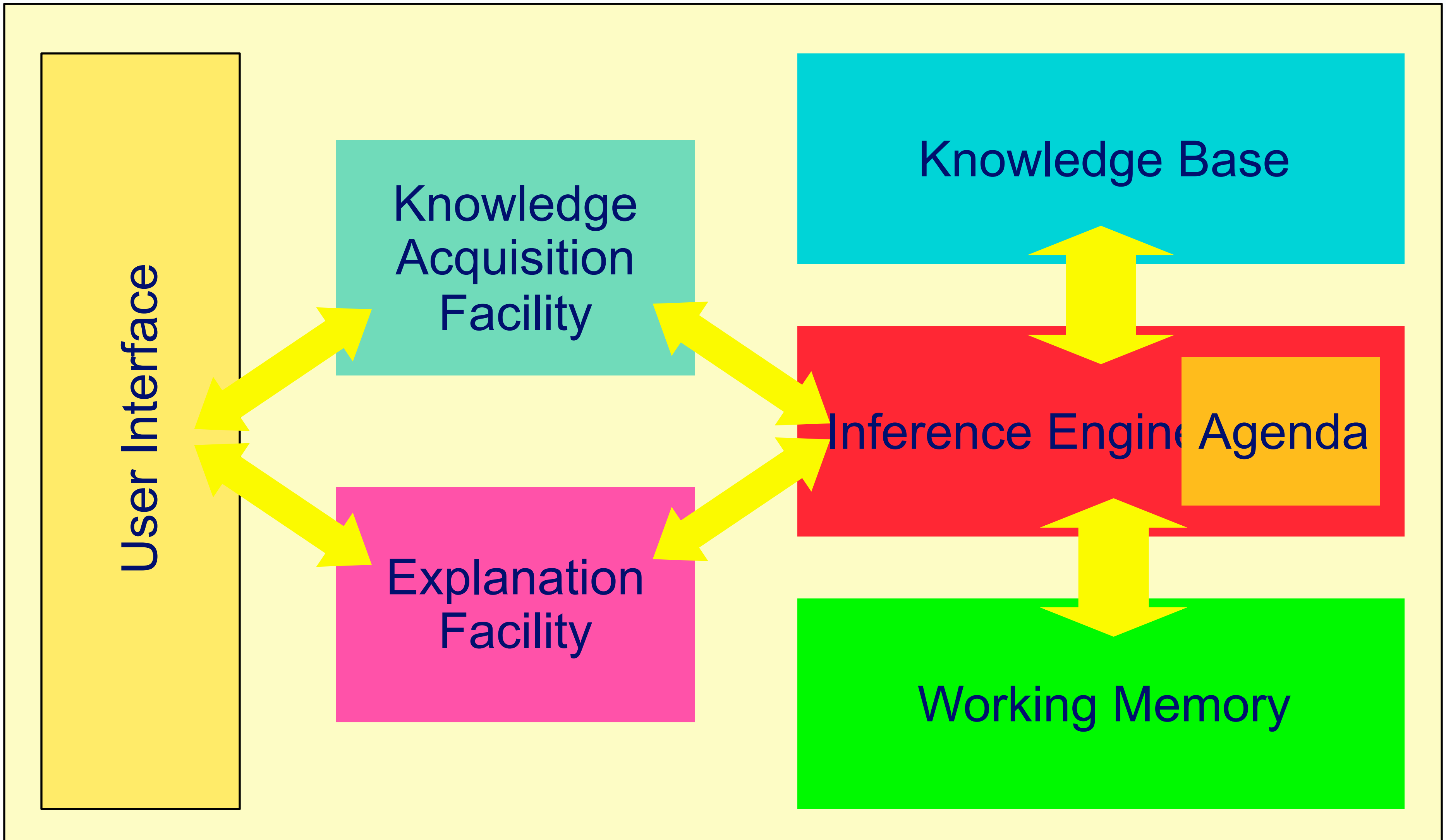
Forward Chaining	*Backward Chaining*
planning, control	diagnosis
data-driven	goal-driven (hypothesis)
bottom-up reasoning	top-down reasoning
find possible conclusions supported by given facts	find facts that support a given hypothesis
similar to breadth-first search	similar to depth-first search
antecedents (LHS) control evaluation	consequents (RHS) control evaluation

Reasoning in Rule-Based Systems

ES Elements

- knowledge base
- inference engine
- working memory
- agenda
- explanation facility
- knowledge acquisition facility
- user interface

ES Structure



Rule-Based ES

- knowledge is encoded as IF … THEN rules
  - these rules can also be written as production rules
- the inference engine determines which rule antecedents are satisfied
  - the left-hand side must "match" a fact in the working memory
- satisfied rules are placed on the agenda
- rules on the agenda can be activated ("fired")
  - an activated rule may generate new facts through its right-hand side
  - the activation of one rule may subsequently cause the activation of other rules

Example Rules

IF … THEN Rules

```
Rule: Red_Light
   IF     the light is red
   THEN   stop
Rule: Green_Light
   IF     the light is green
   THEN   go
```

antecedent (left-hand-side)

consequent (right-hand-side)

Production Rules

```
the light is red ==> stop

the light is green ==> go
```

antecedent (left-hand-side)

consequent (right-hand-side)

MYCIN Sample Rule

Human-Readable Format

IF the stain of the organism is gram negative

AND the morphology of the organism is rod

AND the aerobiocity of the organism is gram anaerobic

THEN the there is strongly suggestive evidence (0.8)

that the class of the organism is enterobacteriaceae

MYCIN Format

```
IF (AND (SAME CNTEXT GRAM GRAMNEG)
        (SAME CNTEXT MORPH ROD)
        (SAME CNTEXT AIR AEROBIC)
THEN (CONCLUDE CNTEXT CLASS ENTEROBACTERIACEAE
     TALLY .8)
```

[Durkin 94, p. 133]

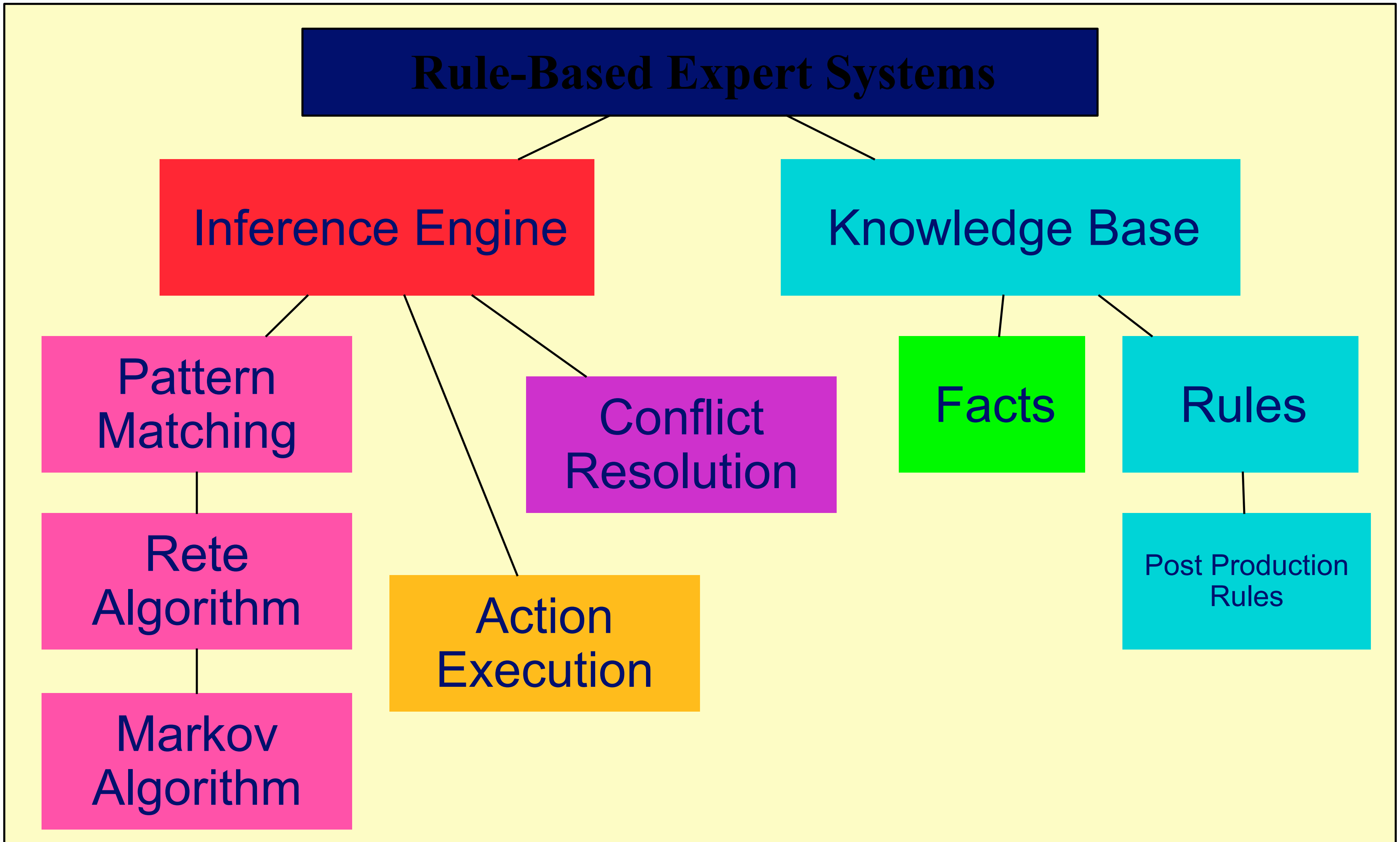
Inference Engine Cycle

- describes the execution of rules by the inference engine
  - conflict resolution
    - select the rule with the highest priority from the agenda
  - execution
    - perform the actions on the consequent of the selected rule
    - remove the rule from the agenda
  - match
    - update the agenda
      - add rules whose antecedents are satisfied to the agenda
      - remove rules with non-satisfied agendas
- the cycle ends when no more rules are on the agenda, or when an explicit stop command is encountered

Forward and Backward Chaining

- different methods of rule activation
  - forward chaining (data-driven)
    - reasoning from facts to the conclusion
    - as soon as facts are available, they are used to match antecedents of rules
    - a rule can be activated if all parts of the antecedent are satisfied
    - often used for real-time expert systems in monitoring and control
    - examples: CLIPS, OPS5
  - backward chaining (query-driven)
    - starting from a hypothesis (query), supporting rules and facts are sought until all parts of the antecedent of the hypothesis are satisfied
    - often used in diagnostic and consultation systems
    - examples: EMYCIN

Foundations of Expert Systems



Post Production Systems

- production rules were used by the logician Emil L. Post in the early 40s in symbolic logic
- Post's theoretical result
  - any system in mathematics or logic can be written as a production system
- basic principle of production rules
  - a set of rules governs the conversion of a set of strings into another set of strings
    - these rules are also known as rewrite rules
    - simple syntactic string manipulation
    - no understanding or interpretation is required
    - also used to define grammars of languages
      - e.g. BNF grammars of programming languages

Emil Post

- 20th century mathematician
- worked in logic, formal languages
  - truth tables
  - completeness proof of the propositional calculus as presented in Principia Mathematica
  - recursion theory
    - mathematical model of computation similar to the Turing machine
- not related to Emily Post ;-)

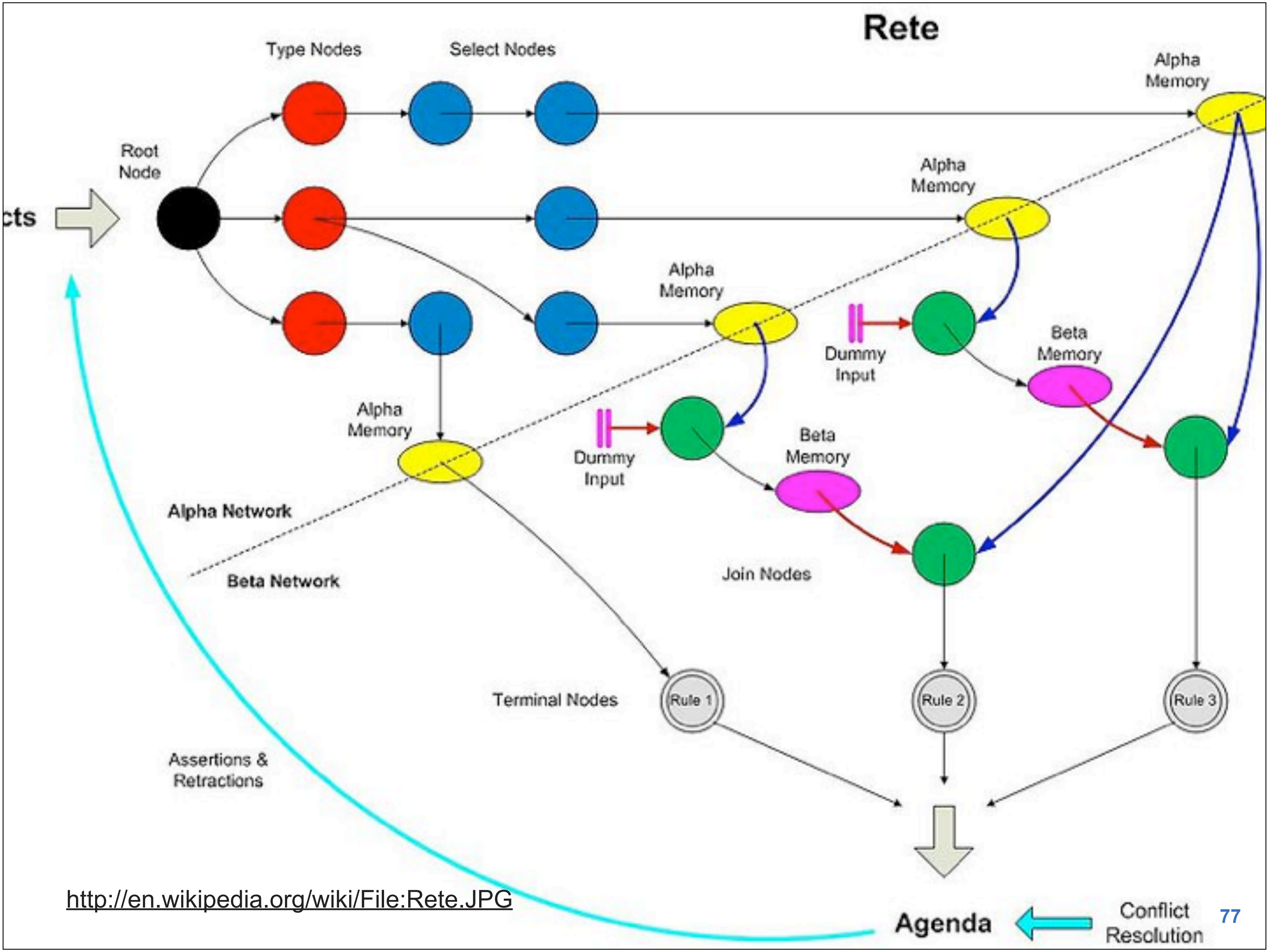
http://en.wikipedia.org/wiki/Emil_Post

Markov Algorithms

- in the 1950s, A. A. Markov introduced priorities as a control structure for production systems
  - rules with higher priorities are applied first
  - allows more efficient execution of production systems
  - but still not efficient enough for expert systems with large sets of rules
  - he is the son of Andrey Markov, who developed Markov chains

Rete Algorithm

- developed by Charles L. Forgy in the late 70s for CMU's OPS (Official Production System) shell
  - stores information about the antecedents in a network
  - in every cycle, it only checks for changes in the networks
  - this greatly improves efficiency



http://en.wikipedia.org/wiki/File:Rete.JPG

Alternative Inference Methods

Alternative Inference Methods

- theorem proving
  - emphasis on mathematical proofs, not so much on performance and ease of use
- probabilistic reasoning
  - integrates probabilities into the reasoning process
- fuzzy reasoning
  - enables the use of ill-defined predicates

Metaknowledge

- deals with “knowledge about knowledge”
  - e.g. reasoning about properties of knowledge representation schemes, or inference mechanisms
  - usually relies on higher order logic
    - in (first order) predicate logic, quantifiers are applied to variables
    - second-order predicate logic allows the use of quantifiers for function and predicate symbols
      - equality is an important second order axiom
        - two objects are equal if all their properties (predicates) are equal
    - may result in substantial performance problems

Important Concepts and Terms

- and operator
- atomic sentence
- backward chaining
- existential quantifier
- expert system shell
- forward chaining
- higher order logic
- Horn clause
- inference
- inference mechanism
- If-Then rules
- implication
- knowledge
- knowledge base
- knowledge-based system
- knowledge representation
- matching
- meta-knowledge
- not operator
- or operator
- predicate logic
- propositional logic
- production rules
- quantifier
- reasoning
- rule
- satisfiability
- semantics
- sentence
- symbol
- syntax
- term
- validity
- unification
- universal quantifier

CAL POLY

Summary Reasoning

- reasoning relies on the ability to generate new knowledge from existing knowledge
  - implemented through inference rules
    - related terms: inference procedure, inference mechanism, inference engine
- computer-based reasoning relies on syntactic symbol manipulation (derivation)
  - inference rules prescribe which combination of sentences can be used to generate new sentences
  - ideally, the outcome should be consistent with the meaning of the respective sentences ("sound" inference rules)
- logic provides the formal foundations for many knowledge representation schemes
  - rules are frequently used in expert systems