MODULE 3:KNOWLEDGE AND REASONING

ARTIFICIAL INTELLIGENCE

KNOWLEDGE BASED AGENTS

A knowledge-based agent includes a knowledge base and an inference system.

A knowledge base is a set of representations of facts of the world.

Each individual representation is called a sentence.

The sentences are expressed in a knowledge representation language.

The agent operates as follows:

1. It TELLs the knowledge base what it perceives.

2. It ASKs the knowledge base what action it should perform.

3. It performs the chosen action.

KB agent execute TELL using propositional logic or FOL(First order Logic)/prediction logic

KB agent execute ASK using resolution, forward chaining and backward chaining

WUMPUS WORLD

PEAS description

1.Performance measure:

+1000 for grabbing gold,-1000 for falling in pit or killed by wumpus, -1 for each action(move) and -10 for using arrow

2.Environment:

Cave with 4*4 grid of rooms

Agent enter at(1,1) and there no wampus/pit at (1,1)

3.Action/Actuators:

Move forward, Turn right 90 ,turn left 90, grab gold, shot wumpus (using arrow once)

4.Sensors :

Breeze-rooms adjacent to pits

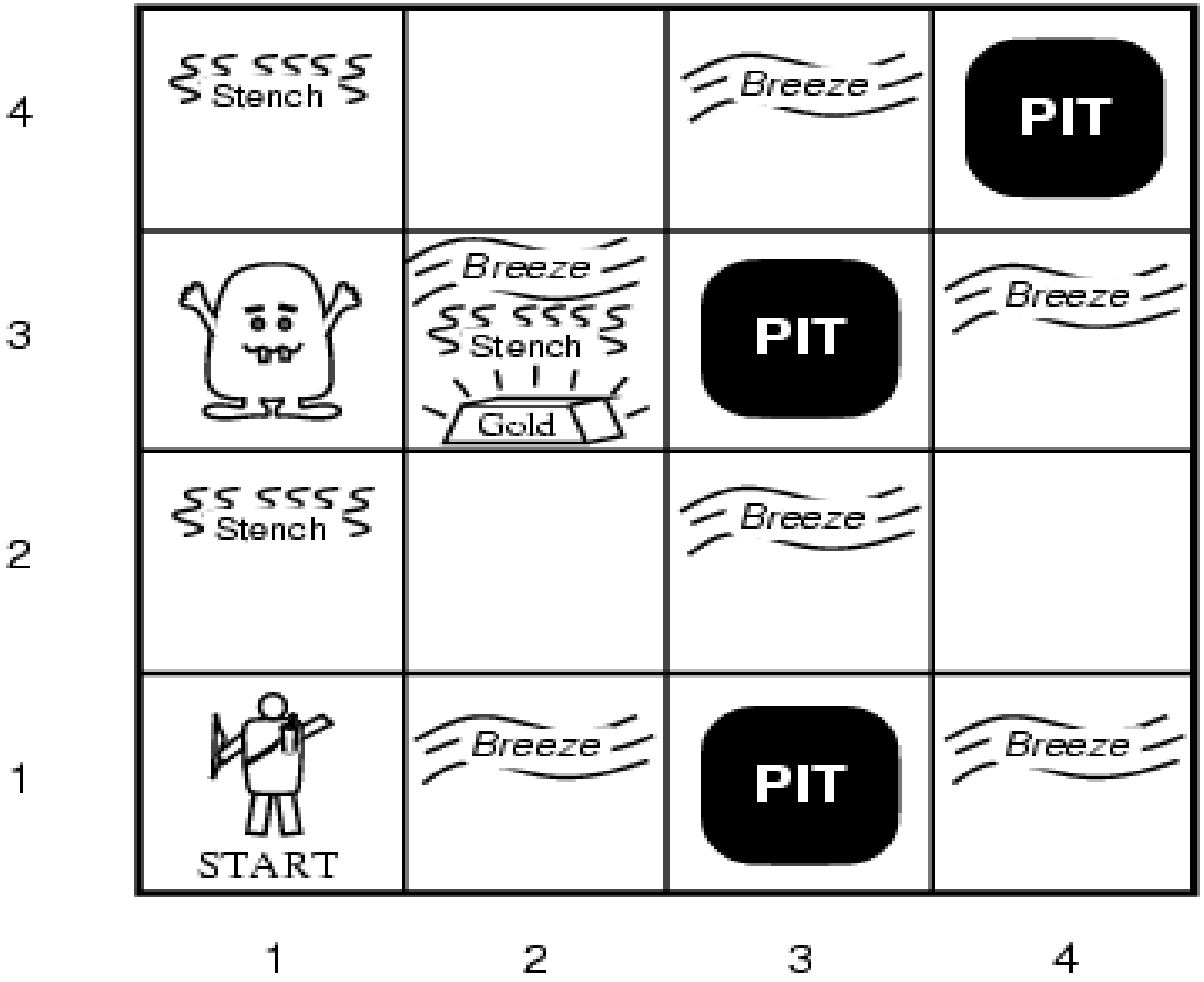
Stench-room adjacent to wumpus

Glitter-room with gold

Scream-wampus dead

Bump-wall/obstacles

WUMPUS WORLD



LOGIC

We need a language to represent domain (possible world) facts

Logic is one such language to represent knowledge in KB

When we talk of any language it must have well defined syntax (grammar or rules) so inference machine can interpret it and semantics help to understand meaning so inference machine can take decision.

If a sentence follows logically from another sentence it is called as entailment.

LOGIC

A declarative statement which is either true or false but not both is called statement or proposition in logic.

Example:

-Amit is Intelligent (Atomic Proposition) P

-Amit is Hardworking (Atomic Proposition)Q

-If Amit is intelligent and Amit is hardworking then Amit top exams (Compound Proposition)

[(P ^ Q)->R]

PROPOSITIONAL LOGIC

Syntax:

The Atomic sentences consists of a single proposition symbol. Each symbol stands for a proposition that can be true or false.

Complex sentences are constructed from simpler sentences, using parentheses and logical connectives.

There are five connectives in common use:

¬ (NOT), ∧ (AND), ∨ (OR), →(Implies), ↔(If and only if)

PROPOSITIONAL LOGIC

Semantics:

For complex sentences, we have 5 rules, which holds for any subsentences P and Q in any model m.

1. ¬ P is true iff P is false in m
2. P ∧ Q is true iff both P and Q are true in m
3. P ∨ Q is true iff either P or Q is true in m
4. P → Q is true unless P is true and Q is false in m
5. P ↔ Q is true iff P and Q are both true or both false in m

FIRST ORDER LOGIC

1. First-order logic is formal logical system used in mathematics, philosophy, linguistic and computer science.

Syntax

User defines these primitives as follows:

1. Constant symbols (i.e., the “individuals” in the world) eg: Ram, 4.
2. Function symbols (mapping individuals to individuals) eg: color of (Asoka Chakra) = Blue.
3. Predicate symbols (mapping from individuals to truth values) eg: greater (5,3), blue(sky).

First-order logic supplies these primitives:

Variable symbols: *x*, *y*.

Connectives: ⌐, ∧, ∨, →, ↔

Quantifiers: universal and existential.

FIRST ORDER LOGIC

Quantifiers

1. Universal quantification corresponds to conjunction (and) in this $(\forall x)\ P(x)$ means that P holds for all values of x in the domain associated with that variable.
2. Existential quantification corresponds to disjunction (or) in that $(\exists x)\ P(x)$ means that P holds for some value of x in the domain associated with that variable.
3. Universal quantifiers are usually used with “implies” to form “IF–THEN rules”.
4. Existential quantifiers are usually used with “and” to specify a list of properties or facts about an individual.
5. Switching the order of universal quantifiers does not change the meaning:$(\forall x)\ (\forall y)\ P\ (x, y)$ is logically equivalent to $(\forall y)\ (\forall x)\ P\ (x, y)$. Similarly, the order of existential quantifiers can be switched.
6. Switching the order of universals and existential does change the meaning.

RESOLUTION

This is most popular method to find inference /reasoning

It is based on proof by contradiction i.e Assume exactly opposite what is asked to prove and then prove your assumption is wrong.

It begin by contradicting the fact to be proved and proceed till KB return NULL or Empty clause

It needs sentences in CNF(Conjuctive normal form)

RESOLUTION

Steps for converting Logical sentence to CNF (Conjuctive Normal Form)

1. Remove Bicondition
2. Remove implication
3. Move negation inside
4. Do Unification
5. No ∧ is permitted in CNF

FORWARD CHAINING

1. Forward chaining is a data driven method of deriving a particular goal from a given knowledge base and set of inference rules
2. Inference rules are applied by matching facts to the antecedents of consequence relations in the knowledge base
3. The application of inference rules results in new knowledge (from the consequents of the relations matched), which is then added to the knowledge base
4. Inference rules are successively applied to elements of the knowledge base until the goal is reached

FORWARD CHAINING

Knowledge Base:

If [X croaks and eats flies] Then [X is a frog]

If [X chirps and sings] Then [X is a canary]

If [X is a frog] Then [X is colored green]

If [X is a canary] Then [X is colored yellow]

[Fritz croaks and eats flies]

Goal:

[Fritz is colored Y]?

FORWARD CHAINING

Knowledge Base
If [X croaks and eats flies]
Then [X is a frog]

If [X chirps and sings]
Then [X is a canary]

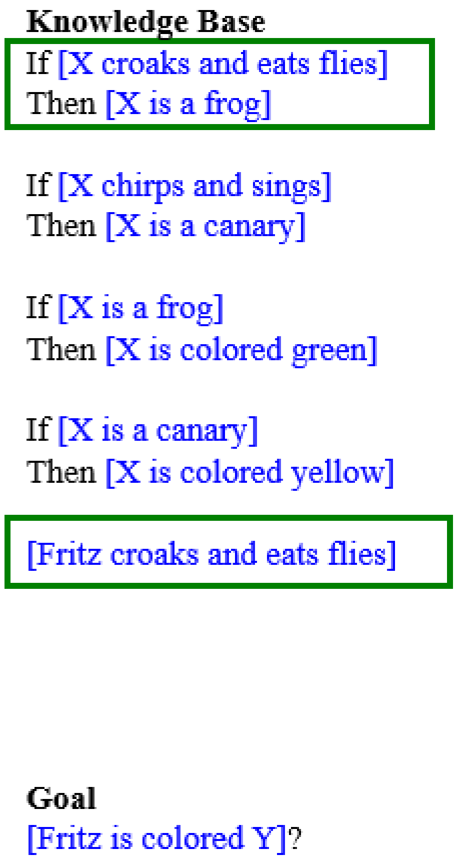
If [X is a frog]
Then [X is colored green]

If [X is a canary]
Then [X is colored yellow]

[Fritz croaks and eats flies]

Goal
[Fritz is colored Y]?

FORWARD CHAINING



Knowledge Base

If [X croaks and eats flies]
Then [X is a frog]

If [X chirps and sings]
Then [X is a canary]

If [X is a frog]
Then [X is colored green]

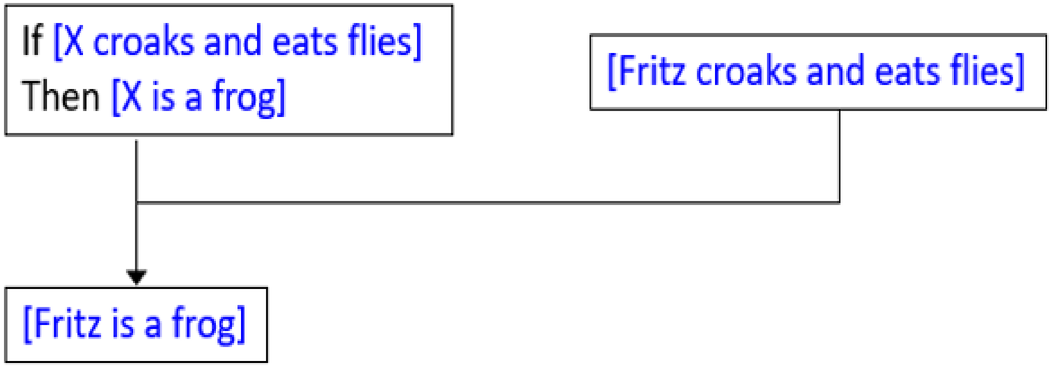
If [X is a canary]
Then [X is colored yellow]

[Fritz croaks and eats flies]

Goal

[Fritz is colored Y]?

FORWARD CHAINING



Knowledge Base

If [X croaks and eats flies]
Then [X is a frog]

If [X chirps and sings]
Then [X is a canary]

If [X is a frog]
Then [X is colored green]

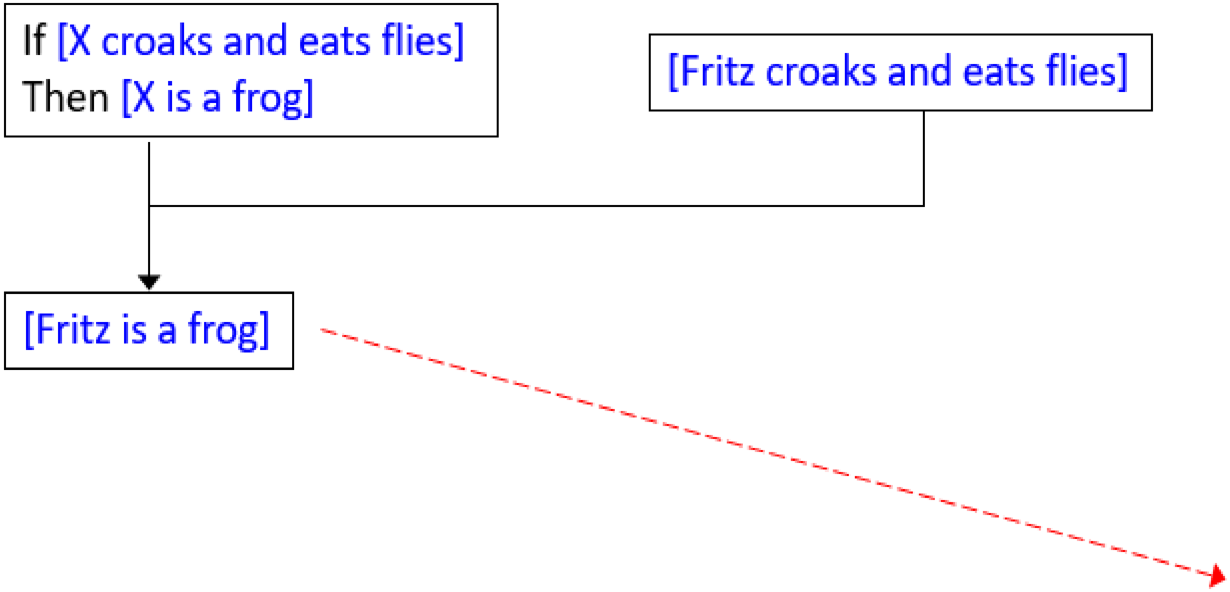
If [X is a canary]
Then [X is colored yellow]

[Fritz croaks and eats flies]

Goal

[Fritz is colored Y]?

FORWARD CHAINING



If [X croaks and eats flies]
Then [X is a frog]

[Fritz croaks and eats flies]

[Fritz is a frog]

Knowledge Base

If [X croaks and eats flies]
Then [X is a frog]

If [X chirps and sings]
Then [X is a canary]

If [X is a frog]
Then [X is colored green]

If [X is a canary]
Then [X is colored yellow]

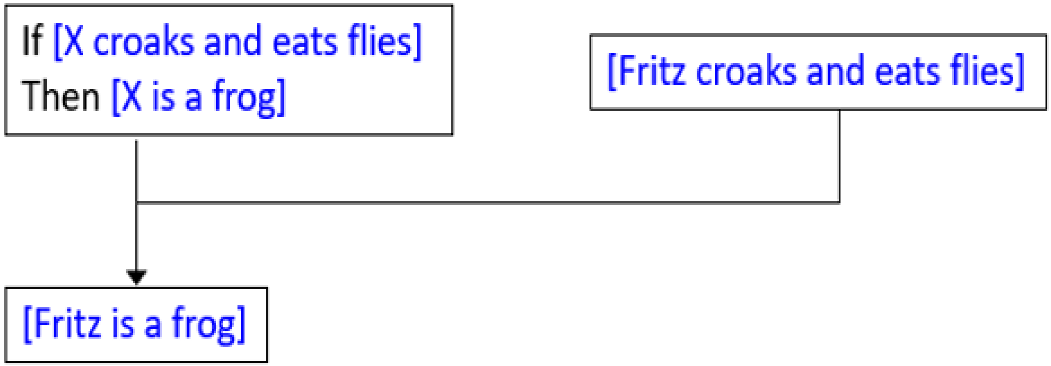
[Fritz croaks and eats flies]

[Fritz is a frog]

Goal

[Fritz is colored Y]?

FORWARD CHAINING



Knowledge Base

If [X croaks and eats flies]
Then [X is a frog]

If [X chirps and sings]
Then [X is a canary]

If [X is a frog]
Then [X is colored green]

If [X is a canary]
Then [X is colored yellow]

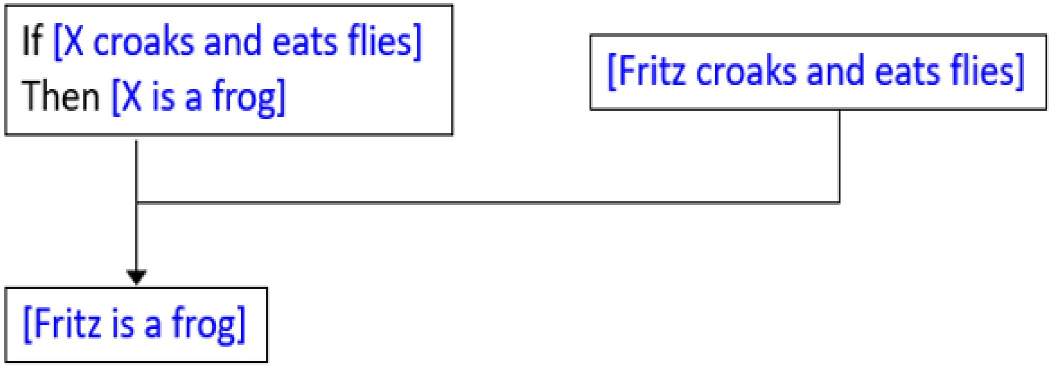
[Fritz croaks and eats flies]

[Fritz is a frog]

Goal

? [Fritz is colored Y]?

FORWARD CHAINING



If [X croaks and eats flies]
Then [X is a frog]

[Fritz croaks and eats flies]

[Fritz is a frog]

Knowledge Base

If [X croaks and eats flies]
Then [X is a frog]

If [X chirps and sings]
Then [X is a canary]

If [X is a frog]
Then [X is colored green]

If [X is a canary]
Then [X is colored yellow]

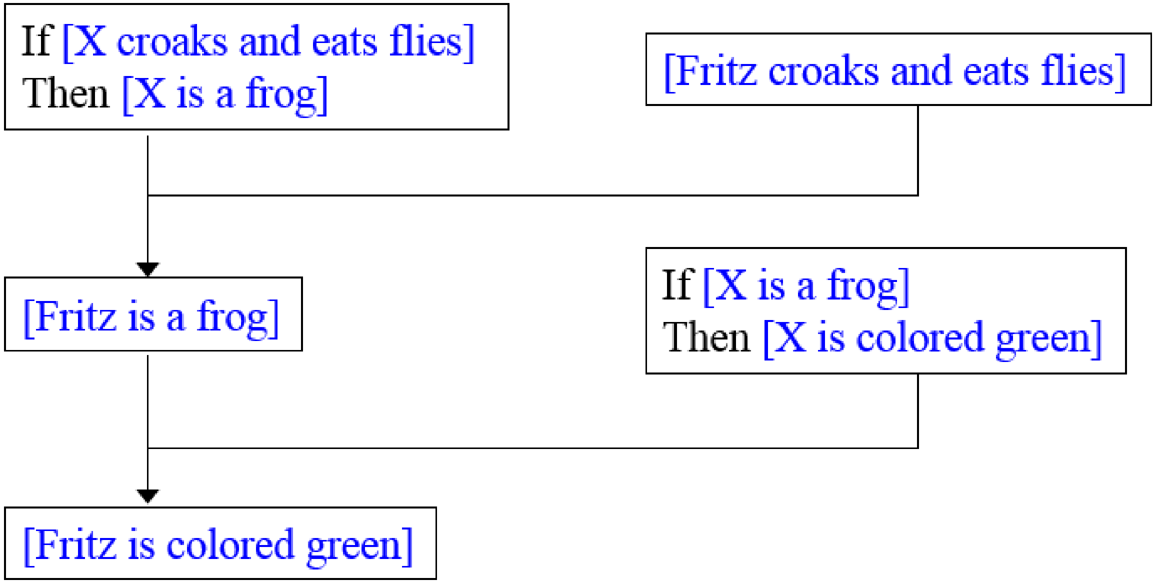
[Fritz croaks and eats flies]

[Fritz is a frog]

Goal

[Fritz is colored Y]?

FORWARD CHAINING



If [X croaks and eats flies]
Then [X is a frog]

[Fritz croaks and eats flies]

[Fritz is a frog]

If [X is a frog]
Then [X is colored green]

[Fritz is colored green]

Knowledge Base

If [X croaks and eats flies]
Then [X is a frog]

If [X chirps and sings]
Then [X is a canary]

If [X is a frog]
Then [X is colored green]

If [X is a canary]
Then [X is colored yellow]

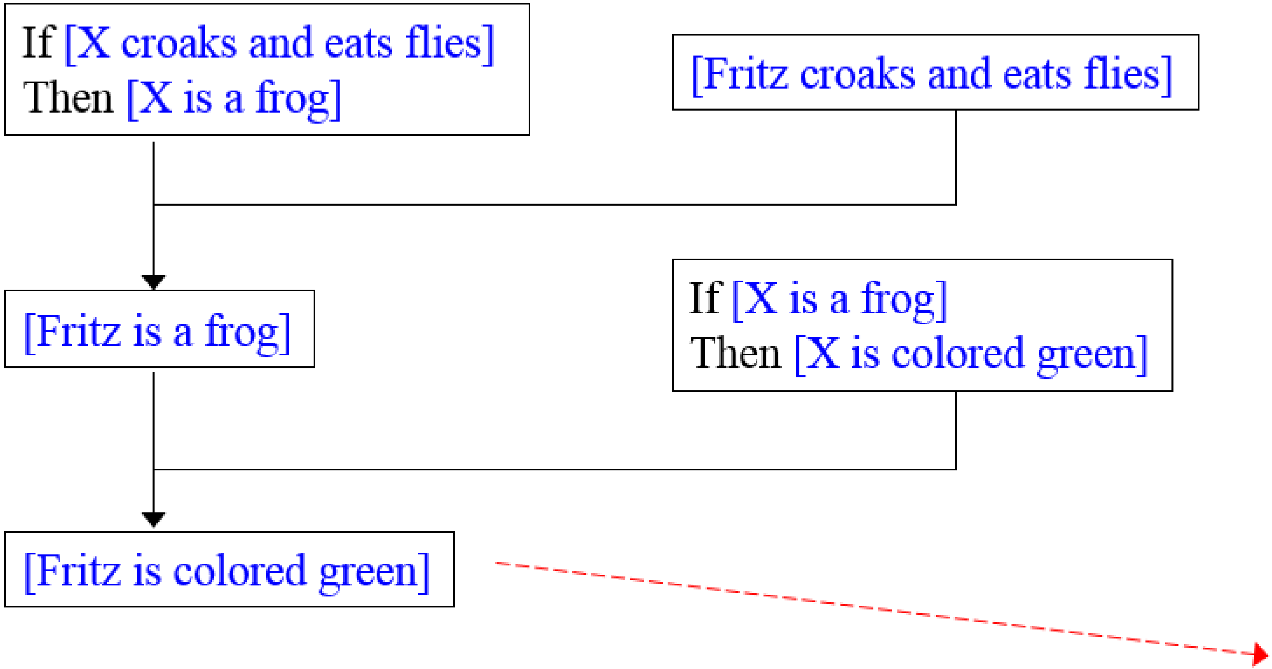
[Fritz croaks and eats flies]

[Fritz is a frog]

Goal

[Fritz is colored Y]?

FORWARD CHAINING



If [X croaks and eats flies]
Then [X is a frog]

[Fritz croaks and eats flies]

[Fritz is a frog]

If [X is a frog]
Then [X is colored green]

[Fritz is colored green]

Knowledge Base

If [X croaks and eats flies]
Then [X is a frog]

If [X chirps and sings]
Then [X is a canary]

If [X is a frog]
Then [X is colored green]

If [X is a canary]
Then [X is colored yellow]

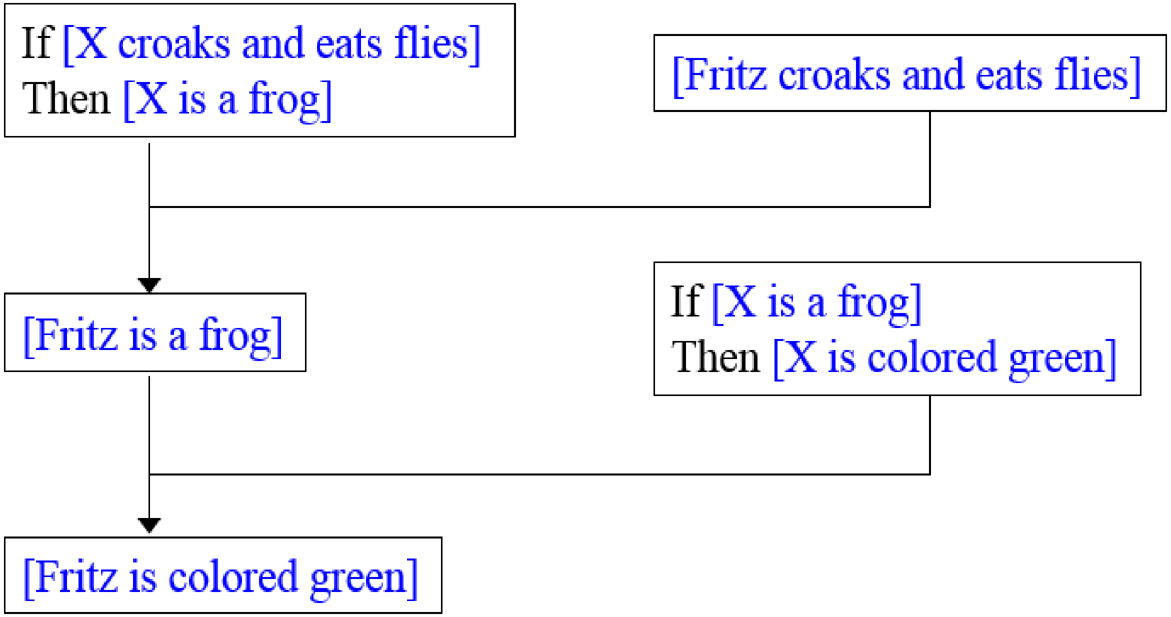
[Fritz croaks and eats flies]

[Fritz is a frog]

[Fritz is colored green]

Goal
[Fritz is colored Y]?

FORWARD CHAINING



If [X croaks and eats flies]
Then [X is a frog]
[Fritz croaks and eats flies]
[Fritz is a frog]
If [X is a frog]
Then [X is colored green]
[Fritz is colored green]

Knowledge Base

If [X croaks and eats flies]
Then [X is a frog]

If [X chirps and sings]
Then [X is a canary]

If [X is a frog]
Then [X is colored green]

If [X is a canary]
Then [X is colored yellow]

[Fritz croaks and eats flies]

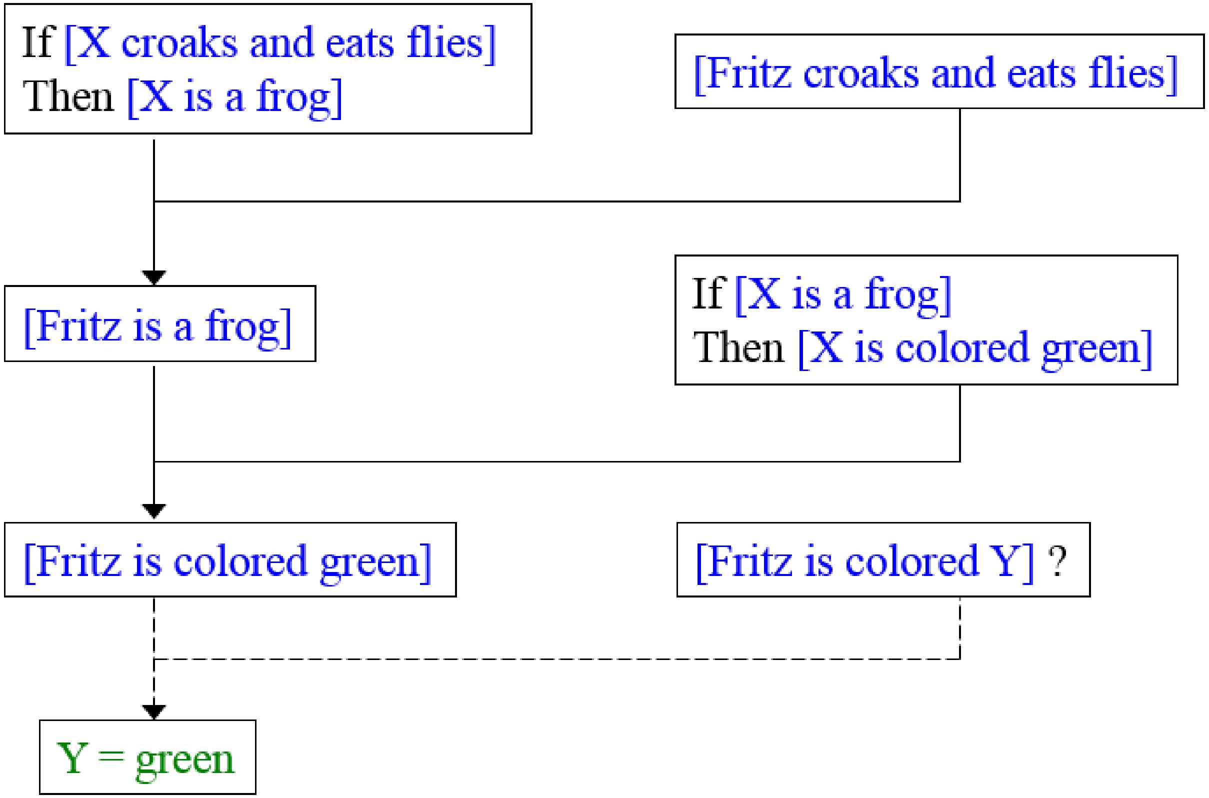
[Fritz is a frog]

[Fritz is colored green]

Goal

[Fritz is colored Y]?

FORWARD CHAINING



If [X croaks and eats flies]
Then [X is a frog]

[Fritz croaks and eats flies]

[Fritz is a frog]

If [X is a frog]
Then [X is colored green]

[Fritz is colored green]

[Fritz is colored Y] ?

Y = green

Knowledge Base

If [X croaks and eats flies]
Then [X is a frog]

If [X chirps and sings]
Then [X is a canary]

If [X is a frog]
Then [X is colored green]

If [X is a canary]
Then [X is colored yellow]

[Fritz croaks and eats flies]

[Fritz is a frog]

[Fritz is colored green]

Goal

[Fritz is colored Y]?

BACKWORD CHAINING

Backward chaining is a goal driven method of deriving a particular goal from a given knowledge base and set of inference rules

Inference rules are applied by matching the goal of the search to the consequents of the relations stored in the knowledge base

When such a relation is found, the antecedent of the relation is added to the list of goals (and not into the knowledge base, as is done in forward chaining)

Search proceeds in this manner until a goal can be matched against a fact in the knowledge base

BACKWARD CHAINING

Knowledge Base

If [X croaks and eats flies]
Then [X is a frog]

If [X chirps and sings]
Then [X is a canary]

If [X is a frog]
Then [X is colored green]

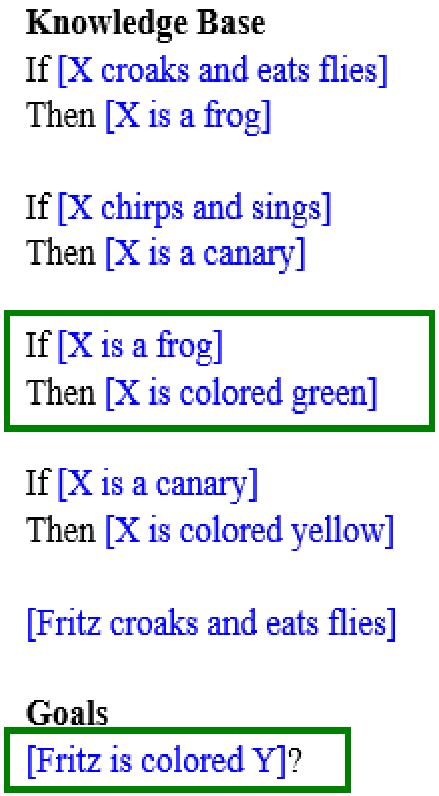
If [X is a canary]
Then [X is colored yellow]

[Fritz croaks and eats flies]

Goal

[Fritz is colored Y]?

BACKWARD CHAINING



Knowledge Base

If [X croaks and eats flies]
Then [X is a frog]

If [X chirps and sings]
Then [X is a canary]

If [X is a frog]
Then [X is colored green]

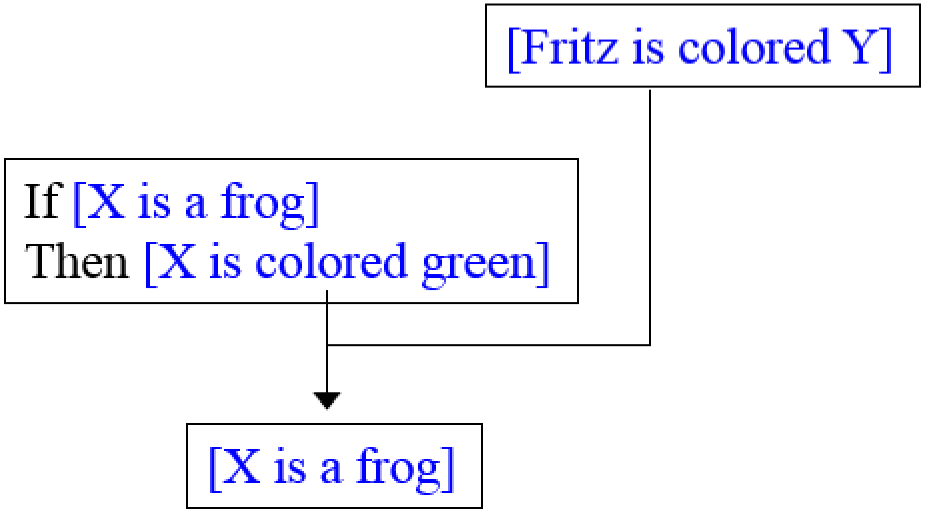
If [X is a canary]
Then [X is colored yellow]

[Fritz croaks and eats flies]

Goals

[Fritz is colored Y]?

BACKWARD CHAINING



[Fritz is colored Y]

If [X is a frog]
Then [X is colored green]

[X is a frog]

Knowledge Base

If [X croaks and eats flies]
Then [X is a frog]

If [X chirps and sings]
Then [X is a canary]

If [X is a frog]
Then [X is colored green]

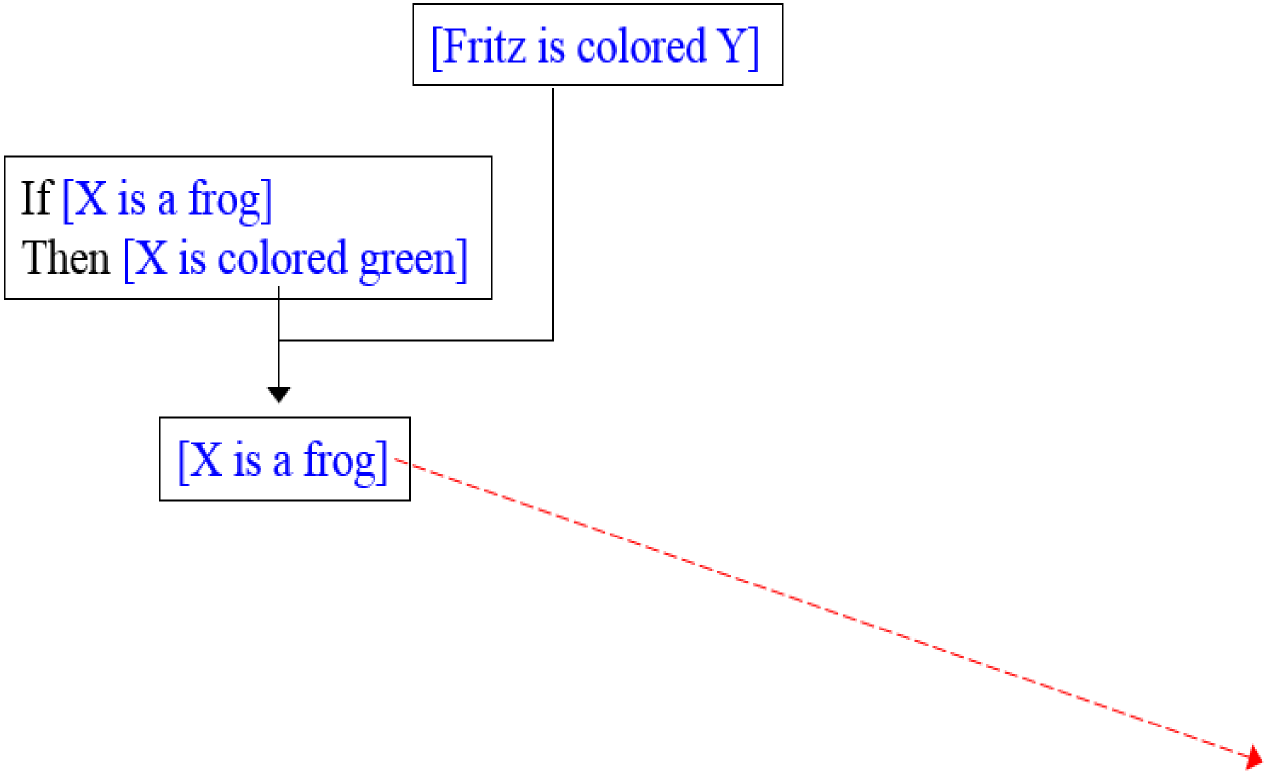
If [X is a canary]
Then [X is colored yellow]

[Fritz croaks and eats flies]

Goals

[Fritz is colored Y]?

BACKWARD CHAINING



[Fritz is colored Y]

If [X is a frog]
Then [X is colored green]

[X is a frog]

Knowledge Base

If [X croaks and eats flies]
Then [X is a frog]

If [X chirps and sings]
Then [X is a canary]

If [X is a frog]
Then [X is colored green]

If [X is a canary]
Then [X is colored yellow]

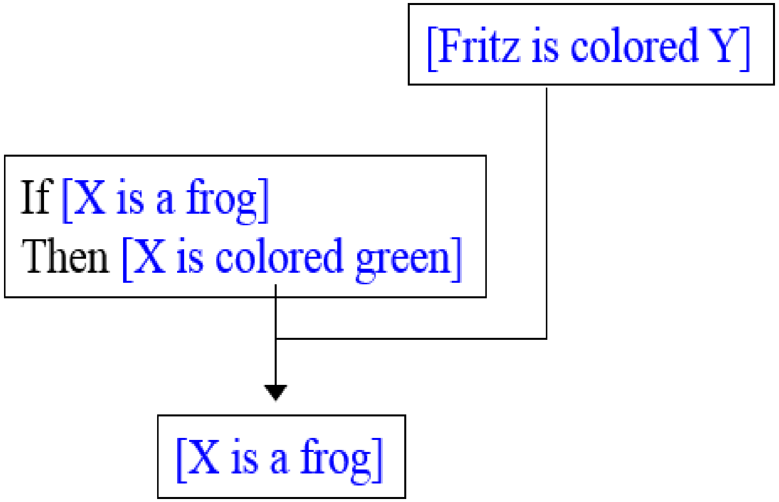
[Fritz croaks and eats flies]

Goals

[Fritz is colored Y]?

[X is a frog]

BACKWARD CHAINING



Knowledge Base

If [X croaks and eats flies]
Then [X is a frog]

If [X chirps and sings]
Then [X is a canary]

If [X is a frog]
Then [X is colored green]

If [X is a canary]
Then [X is colored yellow]

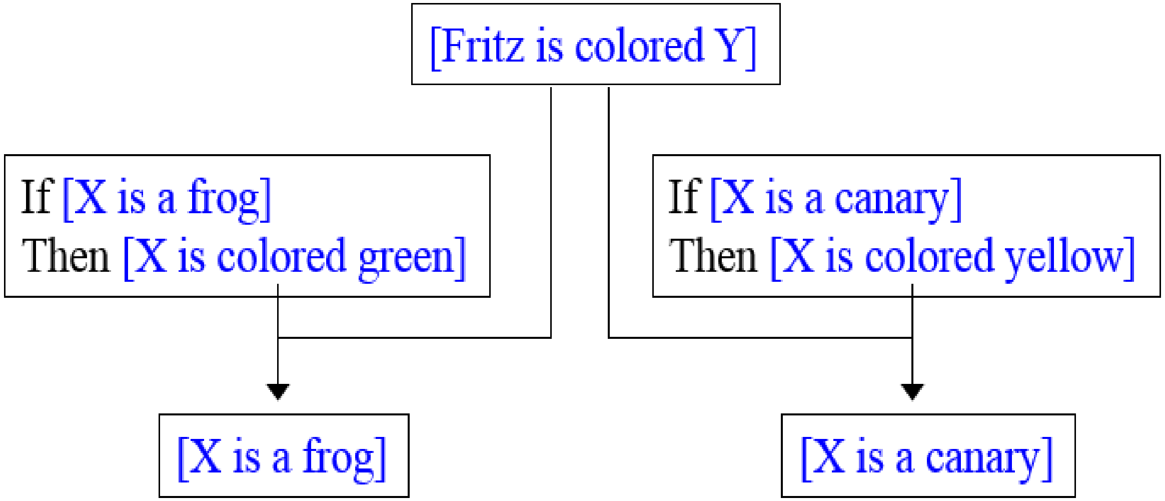
[Fritz croaks and eats flies]

Goals

[Fritz is colored Y]?

[X is a frog]

BACKWARD CHAINING



[Fritz is colored Y]

If [X is a frog]
Then [X is colored green]

If [X is a canary]
Then [X is colored yellow]

[X is a frog]

[X is a canary]

Knowledge Base

If [X croaks and eats flies]
Then [X is a frog]

If [X chirps and sings]
Then [X is a canary]

If [X is a frog]
Then [X is colored green]

If [X is a canary]
Then [X is colored yellow]

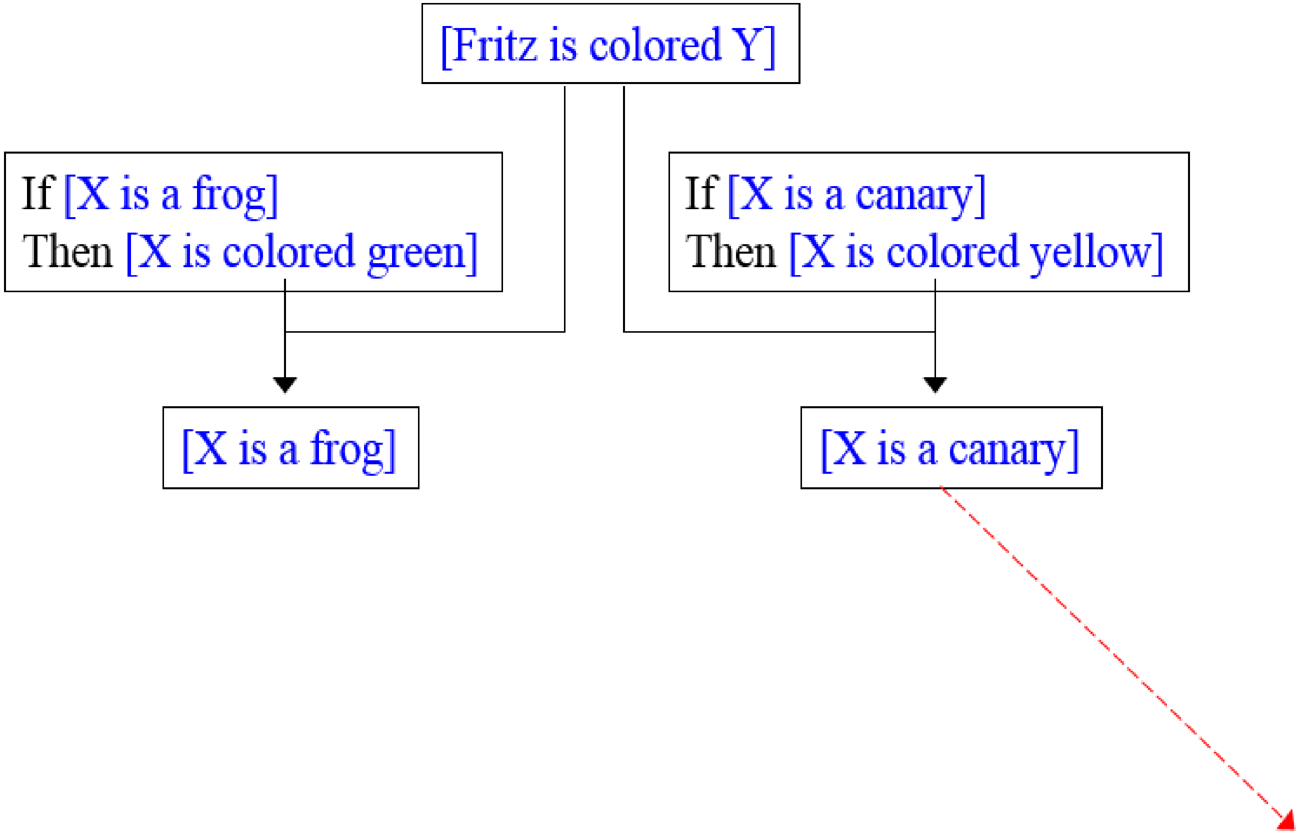
[Fritz croaks and eats flies]

Goals

[Fritz is colored Y]?

[X is a frog]

BACKWARD CHAINING



[Fritz is colored Y]

If [X is a frog]
Then [X is colored green]

If [X is a canary]
Then [X is colored yellow]

[X is a frog]

[X is a canary]

Knowledge Base

If [X croaks and eats flies]
Then [X is a frog]

If [X chirps and sings]
Then [X is a canary]

If [X is a frog]
Then [X is colored green]

If [X is a canary]
Then [X is colored yellow]

[Fritz croaks and eats flies]

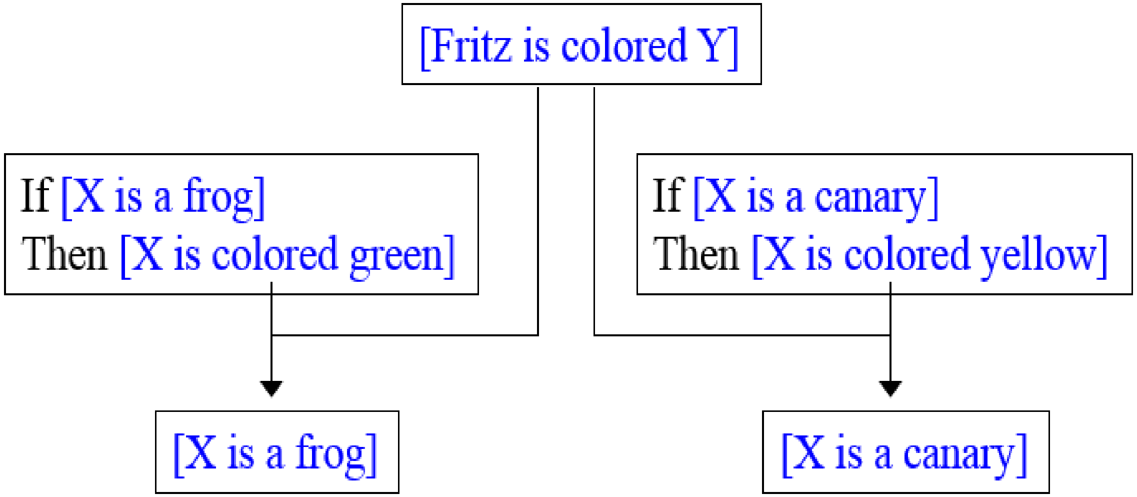
Goals

[Fritz is colored Y]?

[X is a frog]

[X is a canary]

BACKWARD CHAINING



Knowledge Base

If [X croaks and eats flies]
Then [X is a frog]

If [X chirps and sings]
Then [X is a canary]

If [X is a frog]
Then [X is colored green]

If [X is a canary]
Then [X is colored yellow]

[Fritz croaks and eats flies]

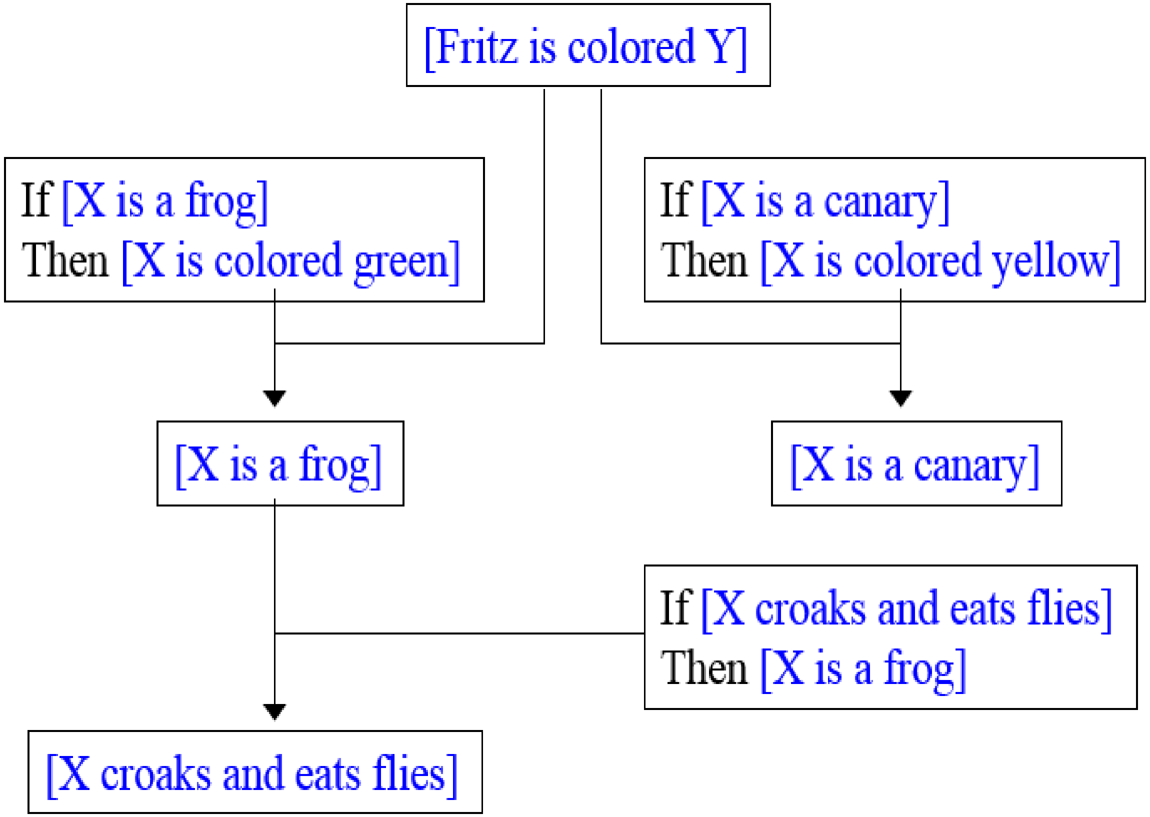
Goals

[Fritz is colored Y]?

[X is a frog]

[X is a canary]

BACKWARD CHAINING


[Fritz is colored Y]
If [X is a frog]
Then [X is colored green]
If [X is a canary]
Then [X is colored yellow]
[X is a frog]
[X is a canary]
If [X croaks and eats flies]
Then [X is a frog]
[X croaks and eats flies]

Knowledge Base

If [X croaks and eats flies]
Then [X is a frog]

If [X chirps and sings]
Then [X is a canary]

If [X is a frog]
Then [X is colored green]

If [X is a canary]
Then [X is colored yellow]

[Fritz croaks and eats flies]

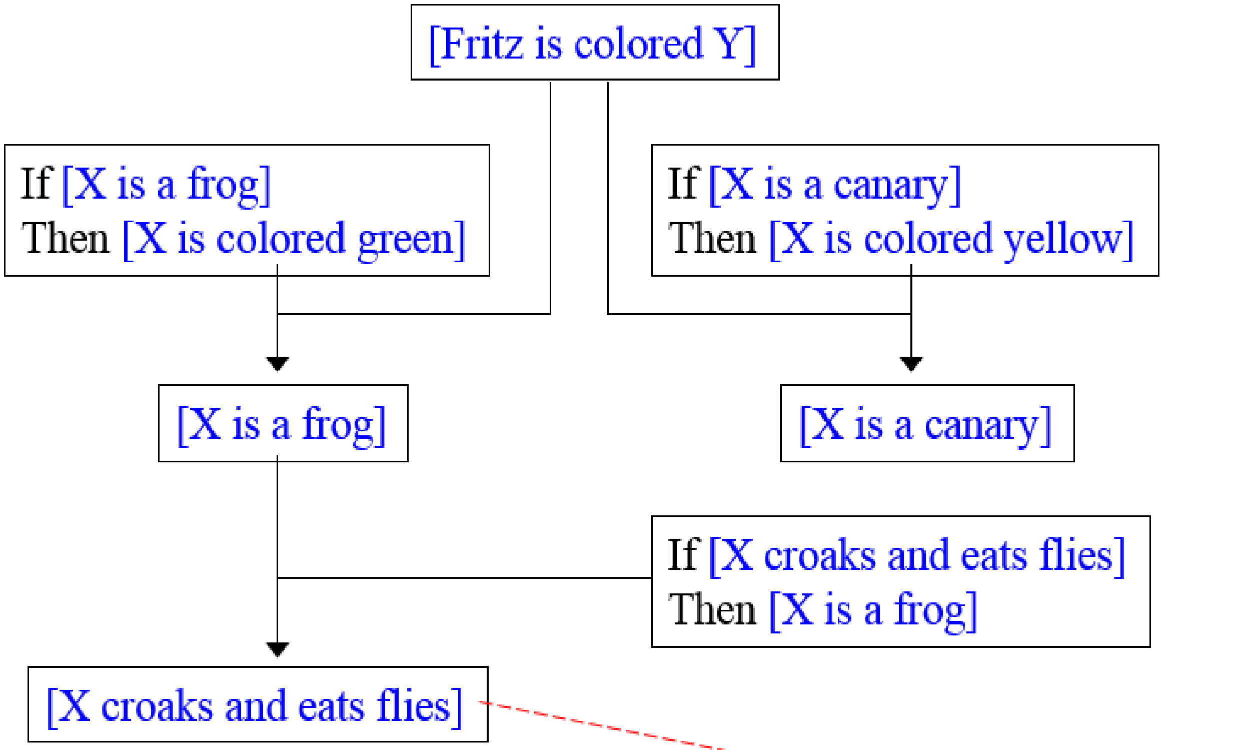
Goals

[Fritz is colored Y]?

[X is a frog]

[X is a canary]

BACKWARD CHAINING


[Fritz is colored Y]
If [X is a frog]
Then [X is colored green]
If [X is a canary]
Then [X is colored yellow]
[X is a frog]
[X is a canary]
If [X croaks and eats flies]
Then [X is a frog]
[X croaks and eats flies]

Knowledge Base

If [X croaks and eats flies]
Then [X is a frog]

If [X chirps and sings]
Then [X is a canary]

If [X is a frog]
Then [X is colored green]

If [X is a canary]
Then [X is colored yellow]

[Fritz croaks and eats flies]

Goals

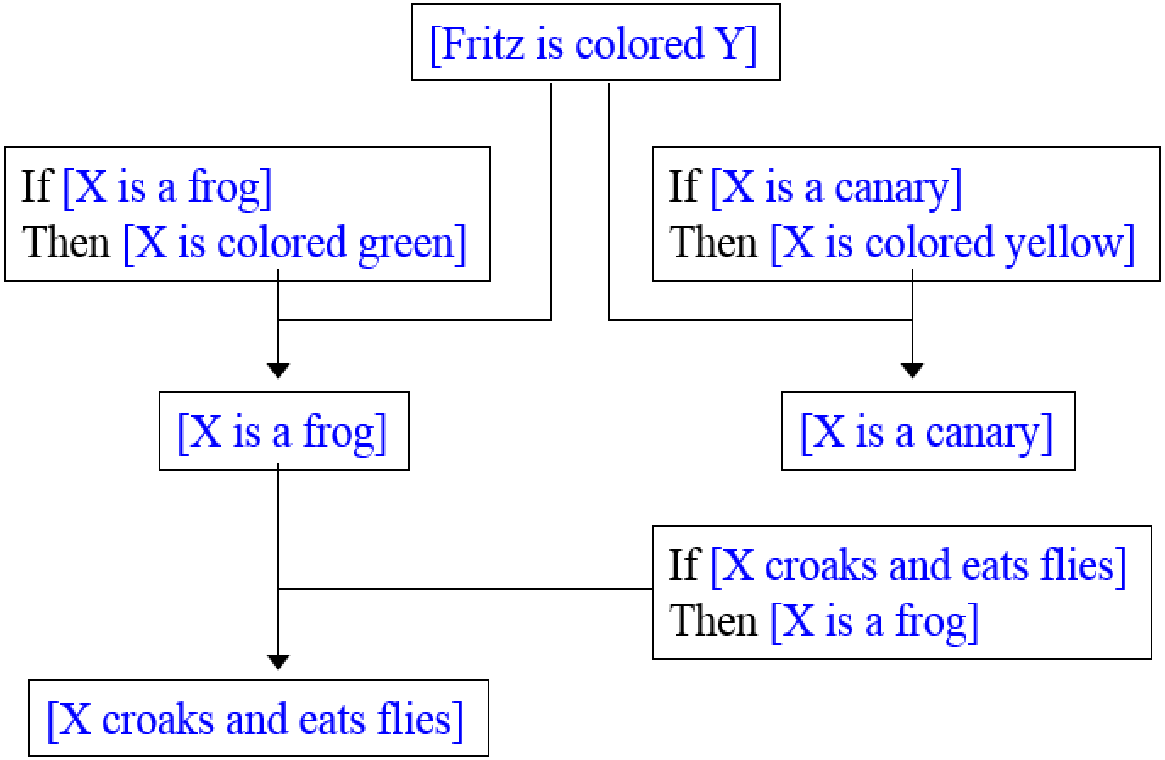
[Fritz is colored Y]?

[X is a frog]

[X is a canary]

[X croaks and eats flies]

BACKWARD CHAINING


[Fritz is colored Y]
If [X is a frog]
Then [X is colored green]
If [X is a canary]
Then [X is colored yellow]
[X is a frog]
[X is a canary]
If [X croaks and eats flies]
Then [X is a frog]
[X croaks and eats flies]

Knowledge Base

If [X croaks and eats flies]
Then [X is a frog]

If [X chirps and sings]
Then [X is a canary]

If [X is a frog]
Then [X is colored green]

If [X is a canary]
Then [X is colored yellow]

[Fritz croaks and eats flies]

Goals

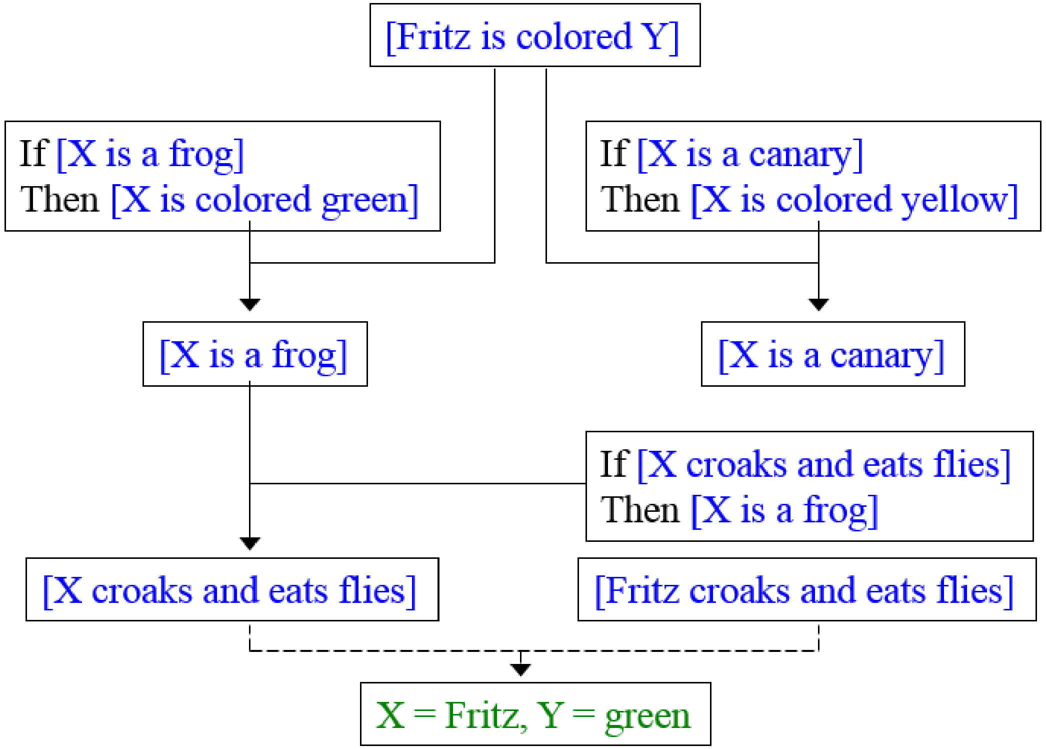
[Fritz is colored Y]?

[X is a frog]

[X is a canary]

[X croaks and eats flies]

BACKWARD CHAINING


[Fritz is colored Y]
If [X is a frog]
Then [X is colored green]
If [X is a canary]
Then [X is colored yellow]
[X is a frog]
[X is a canary]
If [X croaks and eats flies]
Then [X is a frog]
[X croaks and eats flies]
[Fritz croaks and eats flies]
X = Fritz, Y = green

Knowledge Base

If [X croaks and eats flies]
Then [X is a frog]

If [X chirps and sings]
Then [X is a canary]

If [X is a frog]
Then [X is colored green]

If [X is a canary]
Then [X is colored yellow]

[Fritz croaks and eats flies]

Goals

[Fritz is colored Y]?

[X is a frog]

[X is a canary]

[X croaks and eats flies]