LECTURE: 08
ARTIFICIAL INTELLIGENCE
CSE 365
COURSE INSTRUCTOR: AMIT KUMAR DAS

LOGIC

FIRST ORDER LOGIC (FOL)

- First-order logic (FOL) models the world in terms of
  - **Objects:** which are things with individual identities
  - **Properties** : of objects that distinguish them from other objects
  - **Relations**: that hold among sets of objects
  - **Functions:** which are a subset of relations where there is only one "value" for any given "input".

- Examples:
  - Objects: Student, Professor, Car, Person…
  - Properties: blue, oval, even, large, …
  - Relations: Brother-of, mother-of, bigger-than, Likes…
  - Functions: father(), brother(), best-friend()…

SYNTAX OF FOL

- **Constant symbols,** which represent individuals in the world
  - Meena
  - 3
  - Green
- **Function symbols,** which map individuals to individuals
  - brother-of(Meena) = raju
  - color-of(Sky) = Blue
- **Predicate symbols,** which map individuals to truth values
  - greater(5,3)
  - green(Grass)
  - color(Grass, Green)

SYNTAX OF FOL CONT…

- **Variable symbols**
  - x, y, a, b,…
- **Connectives**
  - not ($\neg$), and ($\wedge$), or ($\vee$), implies ($\rightarrow$), if and only if (biconditional $\leftrightarrow$)
- **Quantifiers**
  - Universal $\forall$x : For all x
  - Existential $\exists$x : For some x or at least one x

SOME DEFINITIONS IN FOL

- **Term**: a constant symbol or a variable symbol, or a function.

- **Atomic sentence:** predicate of n terms,

  $$predicate\ (term_1, \ldots, term_n)$$

- **Complex sentence:** is formed from atomic sentences connected by the logical connectives.

  Example: $\neg P$, $P \vee Q$, $P \wedge Q$, $P \rightarrow Q$, $P \leftrightarrow Q$, where P and Q are atomic sentences.

QUANTIFIERS

- **Universal quantification**
  - $(\forall x)P(x)$ means that P holds for **all** values of x in the domain associated with that variable
  - E.g., $\forall x\ (\text{dolphin}(x) \rightarrow \text{mammal}(x))$

- **Existential quantification**
  - $(\exists x)P(x)$ means that P holds for **some** value of x in the domain associated with that variable
  - E.g., $\exists x\ (\text{mammal}(x) \land \text{lays-eggs}(x))$

QUANTIFIERS (CONT…)

- **Universal quantifiers are often used with "implies" to form "rules":**

  $(\forall x)$ student(x) $\rightarrow$ smart(x) means "All students are smart"

- Universal quantification is *rarely* used to make blanket statements about every individual in the world:

  $(\forall x)$student(x)$\wedge$smart(x) means "Everyone in the world is a student and is smart"

- **Existential quantifiers are usually used with "and" to specify a list of properties about an individual:**

  $(\exists x)$ student(x) $\wedge$ smart(x) means "There is a student who is smart"

- A common mistake is to represent this English sentence as the FOL sentence:

  $(\exists x)$ student(x) $\rightarrow$ smart(x)

  - But what happens when there is a person who is *not* a student?

QUANTIFIER SCOPE

- Switching the order of universal quantifiers *does not* change the meaning:
  - $(\forall x)(\forall y)P(x,y) \leftrightarrow (\forall y)(\forall x)\ P(x,y)$
- Similarly, you can switch the order of existential quantifiers:
  - $(\exists x)(\exists y)P(x,y) \leftrightarrow (\exists y)(\exists x)\ P(x,y)$
- Switching the order of universal and existential *does* change meaning:
  - Everyone likes someone: $(\forall x)(\exists y)$ likes(x,y)
  - Someone is liked by everyone: $(\exists y)(\forall x)$ likes(x,y)

NEGATION OF QUANTIFIERS

We can relate sentences involving $\forall$ and $\exists$ using De Morgan's laws:

$$(\forall x)\ \neg P(x) \leftrightarrow \neg(\exists x)\ P(x)$$

$$\neg(\forall x)\ P \leftrightarrow (\exists x)\ \neg P(x)$$

REPRESENTING FACTS USING FOL

- **Lucy is a professor.**

  is-prof(Lucy)

- **All professors are people.**

  $\forall$x (is-prof(x) $\rightarrow$ is-person(x))

- **All Deans are professors.**

  $\forall$x (is-dean(x)$\rightarrow$ is-prof(x))

- **All professors consider the dean a friend or don't know him.**

$\forall$ x ($\exists$ y (is-prof(x) $\wedge$ is-dean(y) $\rightarrow$ is-friend-of(y,x) $\vee$ knows(x, y)))

- **Everyone is a friend of someone.**

  $\forall$ x ($\exists$ y (is-friend-of(y, x)))

REPRESENTING FACTS USING FOL(CONT)

- Every gardener likes the sun.
  - $(\forall x)$ gardener(x) => likes(x, Sun)
- You can fool some of the people all of the time.
  - $(\exists x)(\forall t)$ (person(x) ^ time(t)) => can-fool(x,t)
- You can fool all of the people some of the time.
  - $(\forall x)(\exists t)$ (person(x) ^ time(t) => can-fool(x,t)
- All purple mushrooms are poisonous.
  - $(\forall x)$ (mushroom(x) ^ purple(x)) => poisonous(x)
- No purple mushroom is poisonous.
  - ~$(\exists x)$ purple(x) ^ mushroom(x) ^ poisonous(x)
- or, equivalently,
  - $(\forall x)$ (mushroom(x) ^ purple(x)) => ~poisonous(x)

SUBSTITUTION

- Variables in the sentences can be substituted with terms.

  (terms = constants, variables, functions)
- Thus, Substitution Is a mapping from variables to terms asa we have already seen in inference.

  $\{x_1/t_1, x_2/t_2, \ldots\}$

Example:

- *SUBST*({*x*/*Sam*, *y*/*Pam*}, *Likes*(*x*, *y*))=*Likes*(*Sam*, *Pam*)

- *SUBST* ({*x*/*z*, *y*/*fatherof*(*John*)}, *Likes*(*x*, *y*))

  =*Likes*(*z*, *fatherof*(*John*))

INFERENCE RULES

Remember the inference rules for propositional logic:

RULE	PREMISE	CONCLUSION
Modus Ponens	$A, A \rightarrow B$	B
And Introduction	A, B	$A \wedge B$
And Elimination	$A \wedge B$	A
Double Negation	$\neg\neg A$	A
Unit Resolution	$A \vee B, \neg B$	A
Resolution	$A \vee B, \neg B \vee C$	$A \vee C$

These rules are still valid for FOL

QUANTIFIED INFERENCE RULES

- Universal Elimination:

$$\frac{\forall x\, P(x)}{P(A)}$$

– substitutes a variable with **a constant symbol**

Example:

$\forall x$ *Likes*(*x*, *IceCream*)

SUBST{x/Ben}:

Likes(*Ben*, *IceCream*)

QUANTIFIED INFERENCE RULES

- Existential Elimination:

$$\frac{\exists x\, P(x)}{P(A)}$$

- – substitutes a variable with **a constant symbol** that does not appear elsewhere in the Knowledge Base

Example:

$$\exists x\ \mathit{Kill}(x, \mathit{Victim})$$

SUBST{x/Murderer}:

$$\mathit{Kill}(\mathit{Murderer}, \mathit{Victim})$$

QUANTIFIED INFERENCE RULES

- **Universal Generalization:**

$$\frac{P(A)}{\forall x\, P(x)}$$

Example:

- If "byte carries 4bits of information", then **any** byte carries 4 bits of information.
- If "a single line passes through two points", then a single line passes through **any** two points.

QUANTIFIED INFERENCE RULES

- **Existential Generalization:**

$$\frac{P(A)}{\exists x\, P(x)}$$

Example:

- If you can access a UNIX system with a particular password, then a valid password exists for that UNIX machine.
- If water boils over 100°C then there exists a temperature over which water boils.

GENERALIZED MODUS PONENS

$$\frac{p_1{}',\ p_2{}',\ \ldots,\ p_n{}',\ (p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q)}{q\sigma}$$

where $p_i{}'\sigma = p_i\sigma$ for all i

E.g. $p_1{}' =$ Faster(Bob,Pat)

$p_2{}' =$ Faster(Pat,Steve)

$p_1 \wedge p_2 \Rightarrow q = Faster(x, y) \wedge Faster(y, z) \Rightarrow Faster(x, z)$

$\sigma = \{x/Bob, y/Pat, z/Steve\}$

$q\sigma = Faster(Bob, Steve)$

INFERENCE

1. It is illegal for every students to copy music."
2. "Joe is a student."
3. "Every student copies music."

   Is Joe a criminal?

1) $\forall x\ \exists y$ student(x) $\wedge$ Music(y) $\wedge$ Copies(x,y) $\Rightarrow$ Criminal(x)
2) student (Joe)
3) $\forall x\ \exists y$ student(x) $\wedge$ Music(y) $\wedge$ Copies(x,y)
4) student(joe)$\wedge$ Music(somemusic) $\wedge$ Copies(joe, somemusic)$\Rightarrow$Criminal(joe)
   [From 1 by Universal Elimination]
5) $\exists y$ student(Joe) $\wedge$ Music(y) $\wedge$ Copies(Joe,y)
   [From 3 by Universal Elimination]
6) student(Joe) $\wedge$ Music(somemusic) $\wedge$ Copies(joe, somemusic)
   [From 5 by Existential Elimination]
7) Criminal (Joe) [From 4 and 6 by Modus Ponens]

UNIFICATION

- **Unification:** takes two similar sentences and computes the substitution that makes them look the same.
- If there is no such substitution, then UNIFY should return *fail.*

  UNIFY (*p*, *q*) = θ such that *SUBST*(θ , *p*) = *SUBST* (θ , *q*)

- Θ is called unifier.

Examples:

- *UNIFY* (*Knows* (*John*, *x*), *Knows* (*John*, *Jane*)) = {*x*/*Jane*}
- *UNIFY*(*Knows*(*John*, *x*), *Knows*(*y*, *Ann*)) = {*x*/*Ann*, *y*/*John*}
- *UNIFY* (*Knows* (*John* , *x*), *Knows* (*y*, *MotherOf*(*y*))) =

  {x/MotherOf(John), y/John}

UNIFICATION

- *UNIFY* (*Knows* (*John*, *x*), *Knows* (*x*, *Elizabeth*)) = fail

- *UNIFY (Parents (x, father(x), mother(Jane)),*
  Parents(Bill, father(y), mother(y))) = fail.

- *Same variable can not be replaced by two constants.*
- A variable can never be replaced by a term containing that variable. For example, **x/f(x)** is illegal. This is known as **"occurs check“.**

HORN CLAUSE

- A **Horn clause** is a clause that has **at most one positive literal**.

Examples:

$P;\ P \vee \neg Q;\ \neg P \vee \neg Q;\ \neg P \vee \neg Q \vee R;\ P \rightarrow Q$

- Conjunctive Normal Form(CNF):
  - Conjunction of disjunction of literals
- All of the following formulas are **in CNF**:

$$\neg A \wedge (B \vee C)$$

$$(A \vee B) \wedge (\neg B \vee C \vee \neg D) \wedge (D \vee \neg E)$$

$$A \wedge B$$

$$A \vee B$$

CNF

- The following formulas are **not in CNF**:

$$(A \wedge B) \vee C$$

$$A \wedge (B \vee (D \wedge E)).$$

$$\neg(B \vee C)$$

KNOWLEDGE BASE IN INFERENCE

- Knowledge Base
  - Stores knowledge used by the system, usually represented in a formal logical manner

- Inference System
  - Defines how existing knowledge may be used to derive new knowledge

FORWARD & BACKWARD CHAINING

- **forward chaining:** We start with the sentences in the knowledge base and generate new conclusions that in turn can allow more inferences to be made.
  - *data-driven*
  - Automatic, unconscious processing
  - May do lots of work that is irrelevant to the goal

FORWARD CHAINING EXAMPLE

- 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 EXAMPLE

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 EXAMPLE

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]

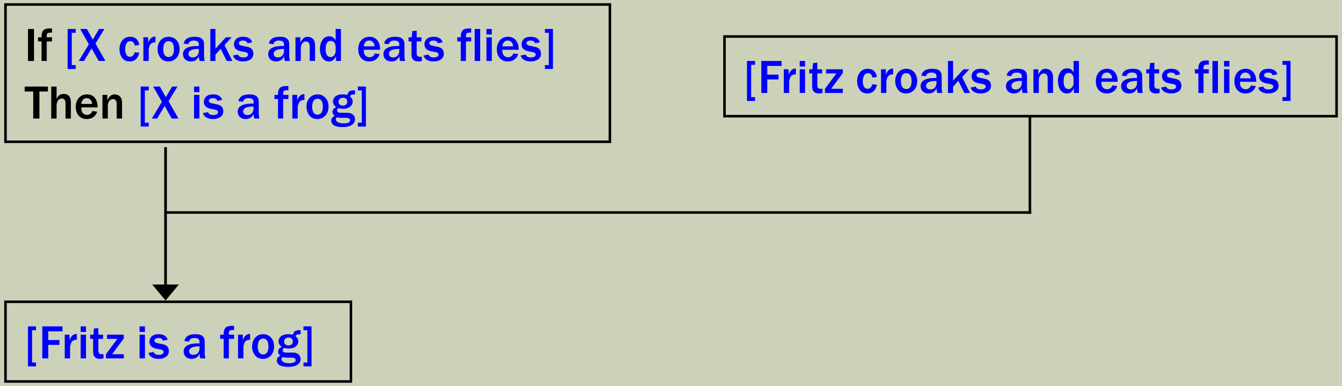
If [X is a canary]
Then [X is colored yellow]

[Fritz croaks and eats flies]

Goal

[Fritz is colored Y]?

FORWARD CHAINING EXAMPLE



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]

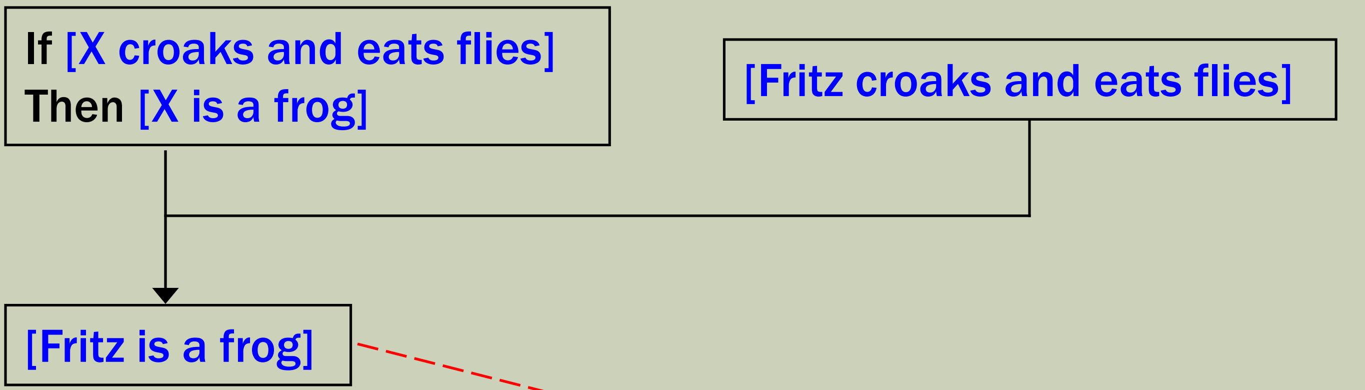
If [X is a canary]
Then [X is colored yellow]

[Fritz croaks and eats flies]

Goal

[Fritz is colored Y]?

FORWARD CHAINING EXAMPLE



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]

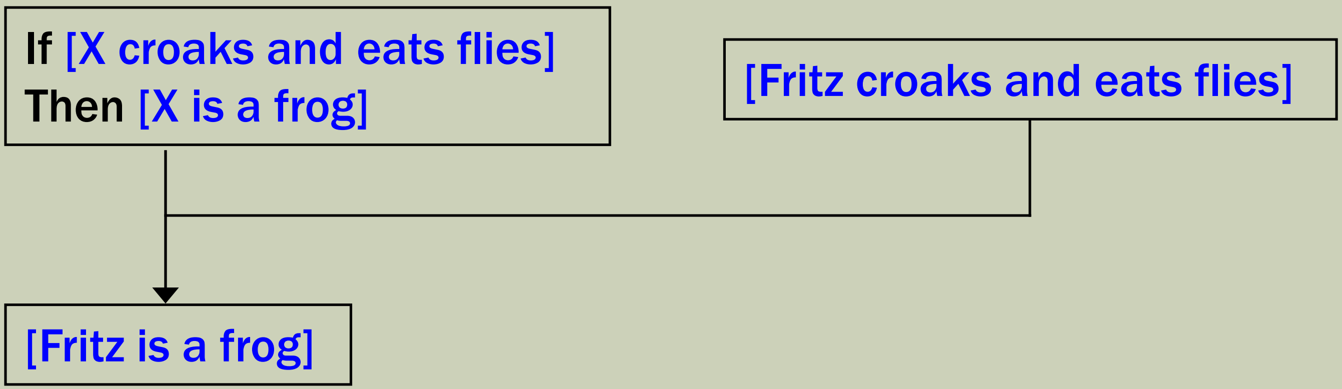
[Fritz croaks and eats flies]

[Fritz is a frog]

Goal

[Fritz is colored Y]?

FORWARD CHAINING EXAMPLE



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]

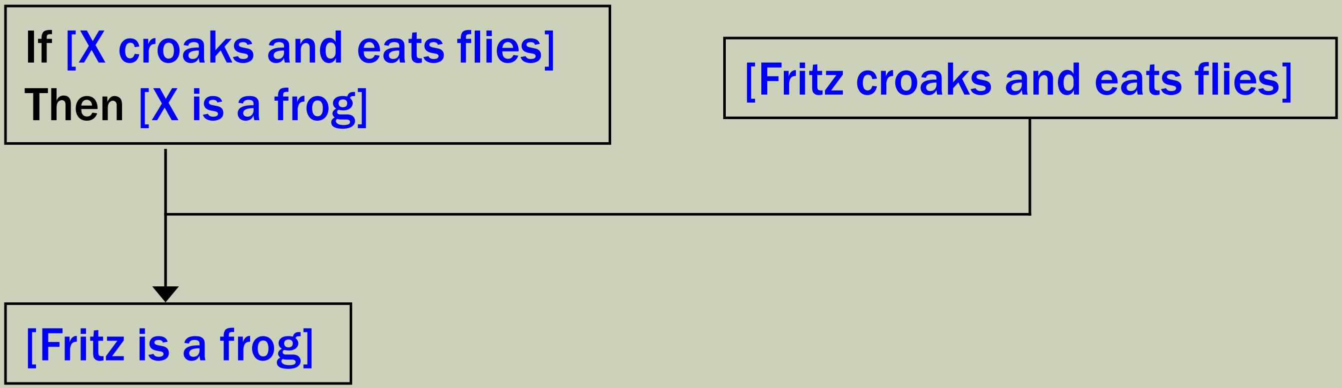
[Fritz croaks and eats flies]

[Fritz is a frog]

Goal

? [Fritz is colored Y]?

FORWARD CHAINING EXAMPLE



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]

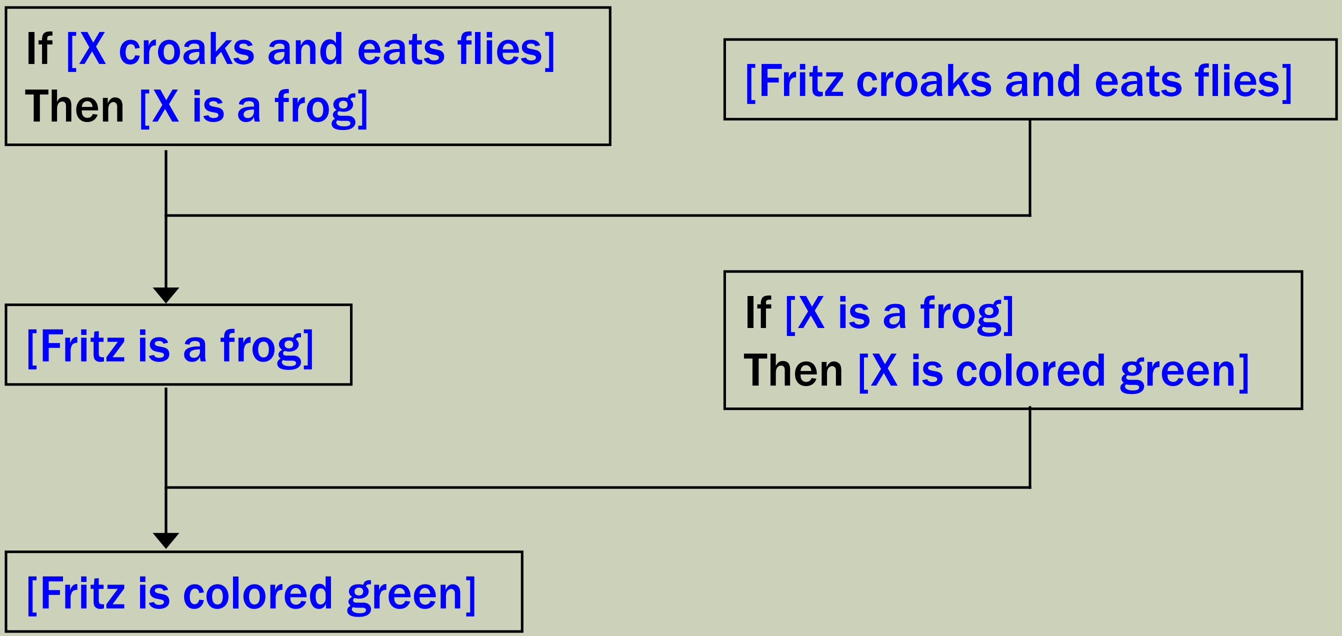
[Fritz croaks and eats flies]

[Fritz is a frog]

Goal

[Fritz is colored Y]?

FORWARD CHAINING EXAMPLE



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]

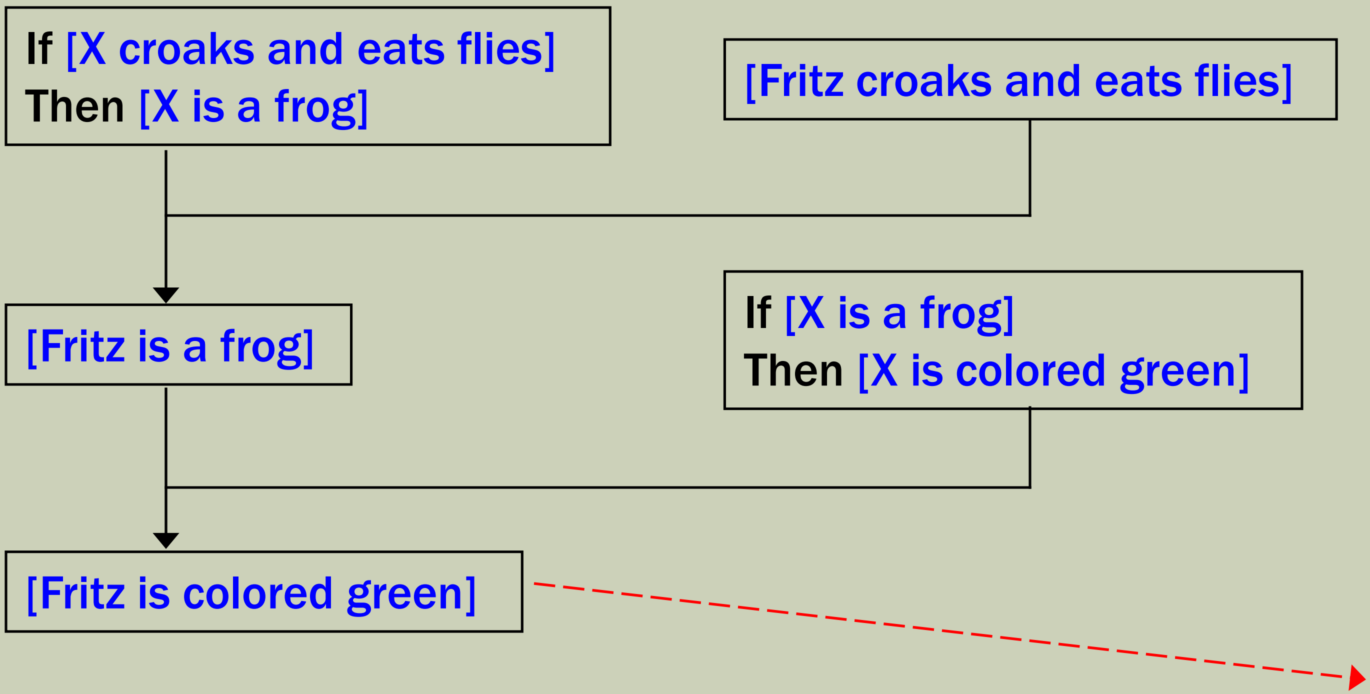
[Fritz croaks and eats flies]

[Fritz is a frog]

Goal

[Fritz is colored Y]?

FORWARD CHAINING EXAMPLE



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]

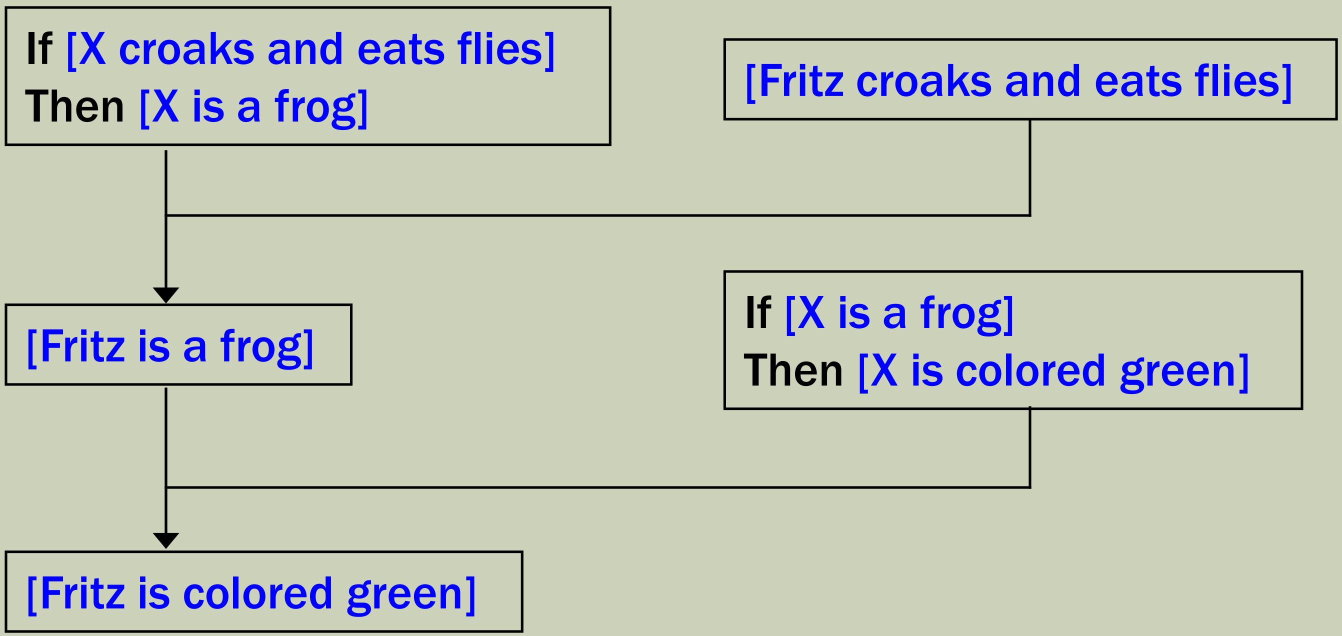
[Fritz is a frog]

[Fritz is colored green]

Goal

[Fritz is colored Y]?

FORWARD CHAINING EXAMPLE



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]

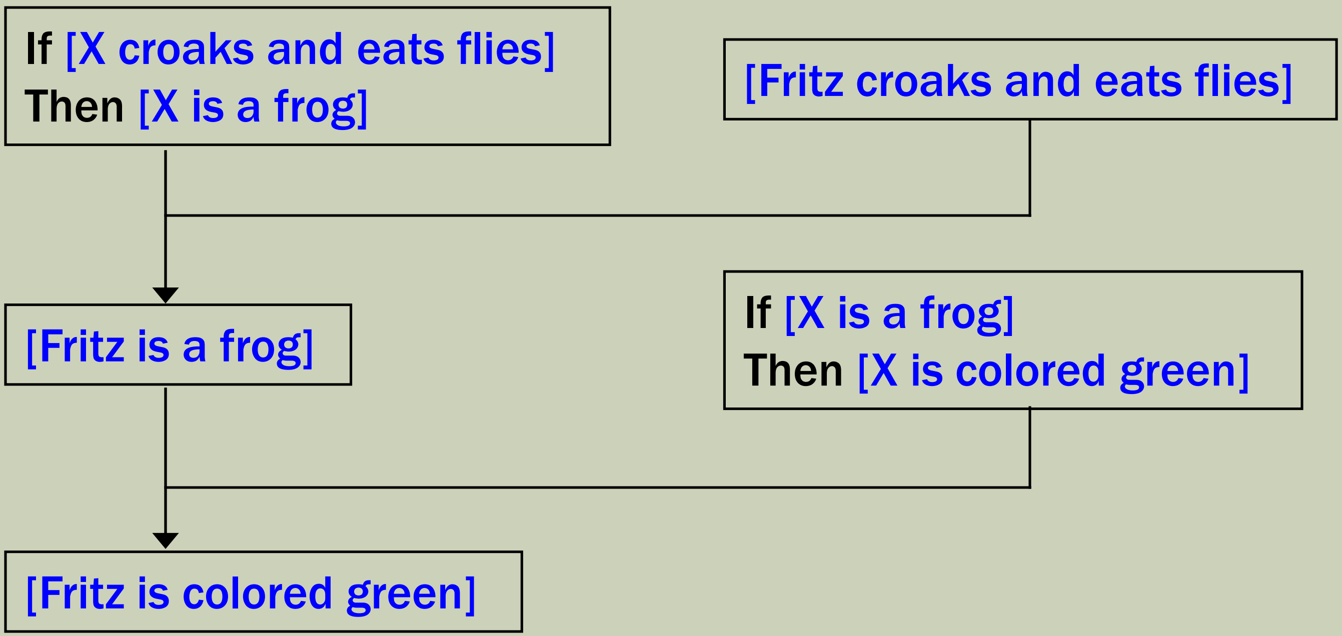
[Fritz is a frog]

[Fritz is colored green]

Goal

? [Fritz is colored Y]?

FORWARD CHAINING EXAMPLE



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]

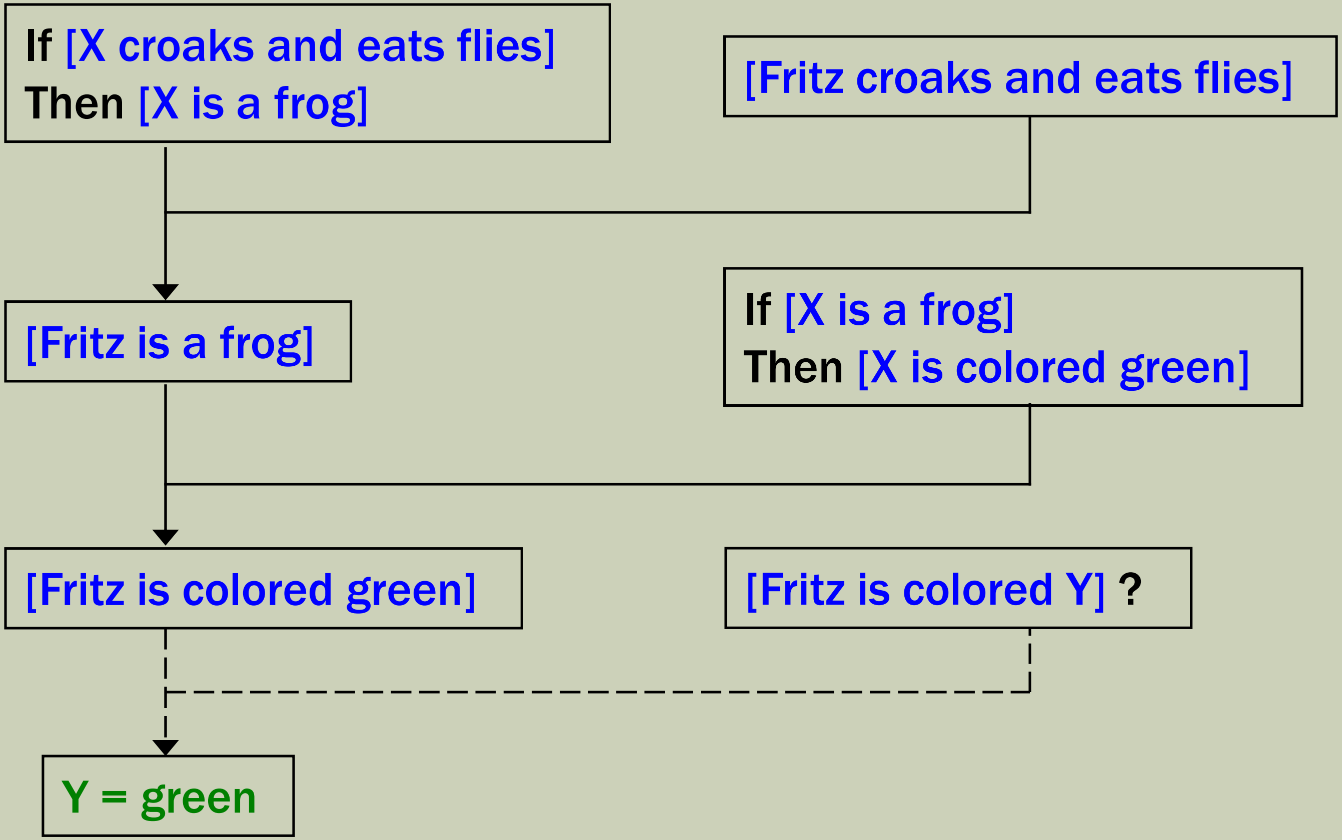
[Fritz is a frog]

[Fritz is colored green]

Goal

[Fritz is colored Y]?

FORWARD CHAINING EXAMPLE



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]?

BACKWARD CHAINING

- **Backward chaining:** We start with something we want to prove, find implication sentences that would allow us to conclude it, and then attempt to establish their premises in turn.
  - goal-driven
  - Where are my keys? How do I get to my next class

BACKWARD CHAINING EXAMPLE

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 EXAMPLE

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]

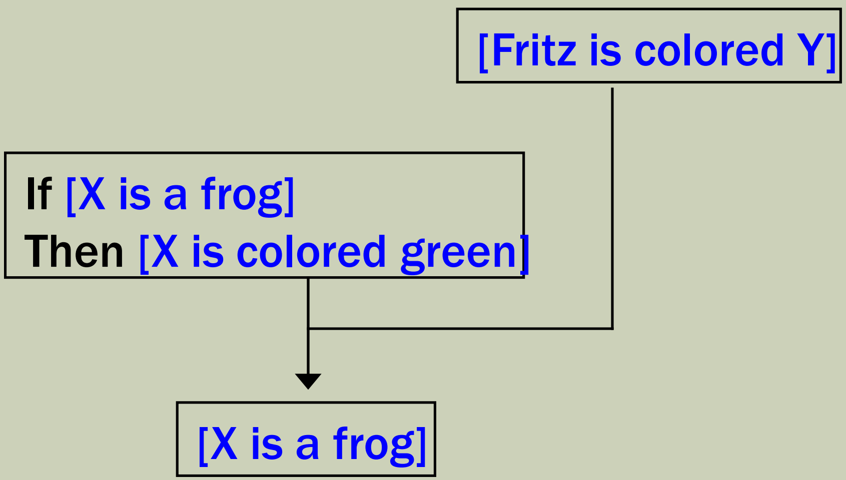
If [X is a canary]
Then [X is colored yellow]

[Fritz croaks and eats flies]

Goals

[Fritz is colored Y]?

BACKWARD CHAINING EXAMPLE



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]

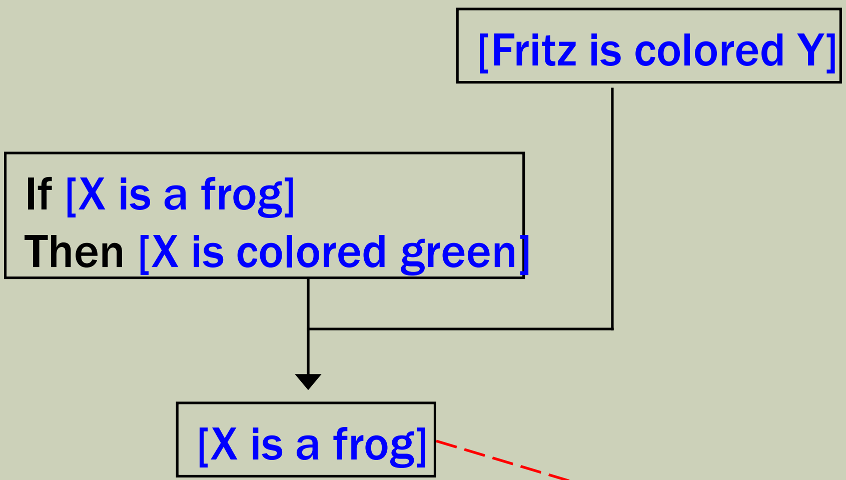
If [X is a canary]
Then [X is colored yellow]

[Fritz croaks and eats flies]

Goals

[Fritz is colored Y]?

BACKWARD CHAINING EXAMPLE



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]

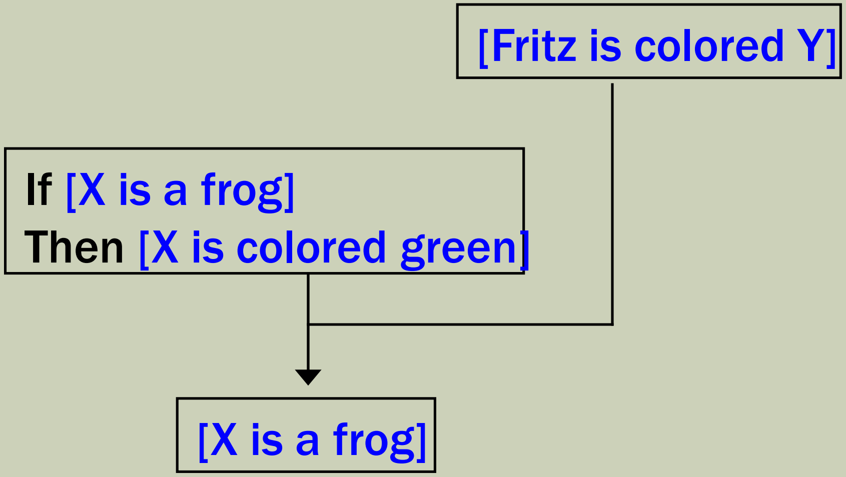
[Fritz croaks and eats flies]

Goals

[Fritz is colored Y]?

[X is a frog]

BACKWARD CHAINING EXAMPLE



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]

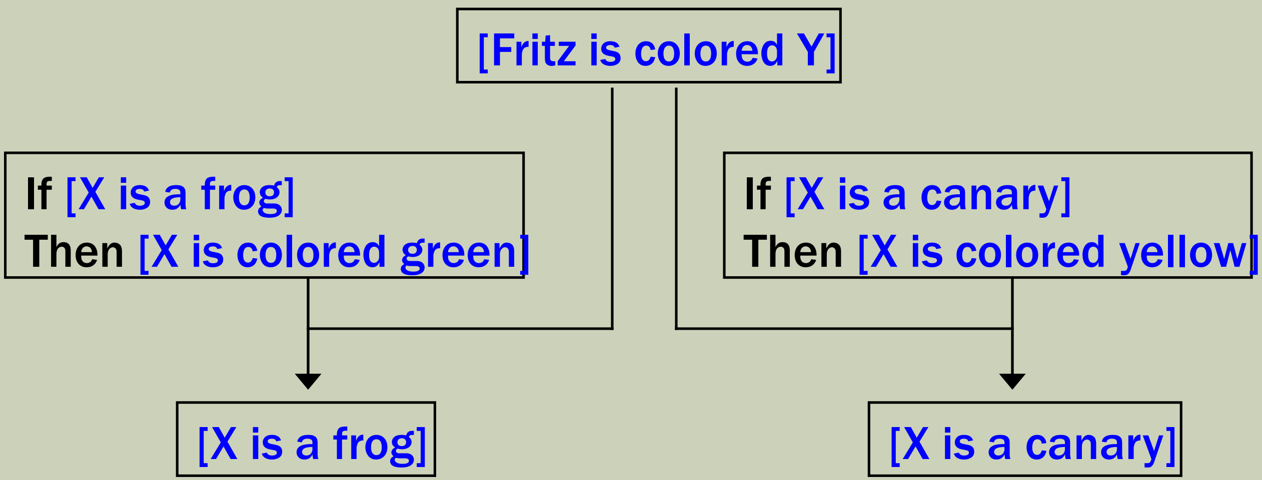
[Fritz croaks and eats flies]

Goals

[Fritz is colored Y]?

[X is a frog]

BACKWARD CHAINING EXAMPLE



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]

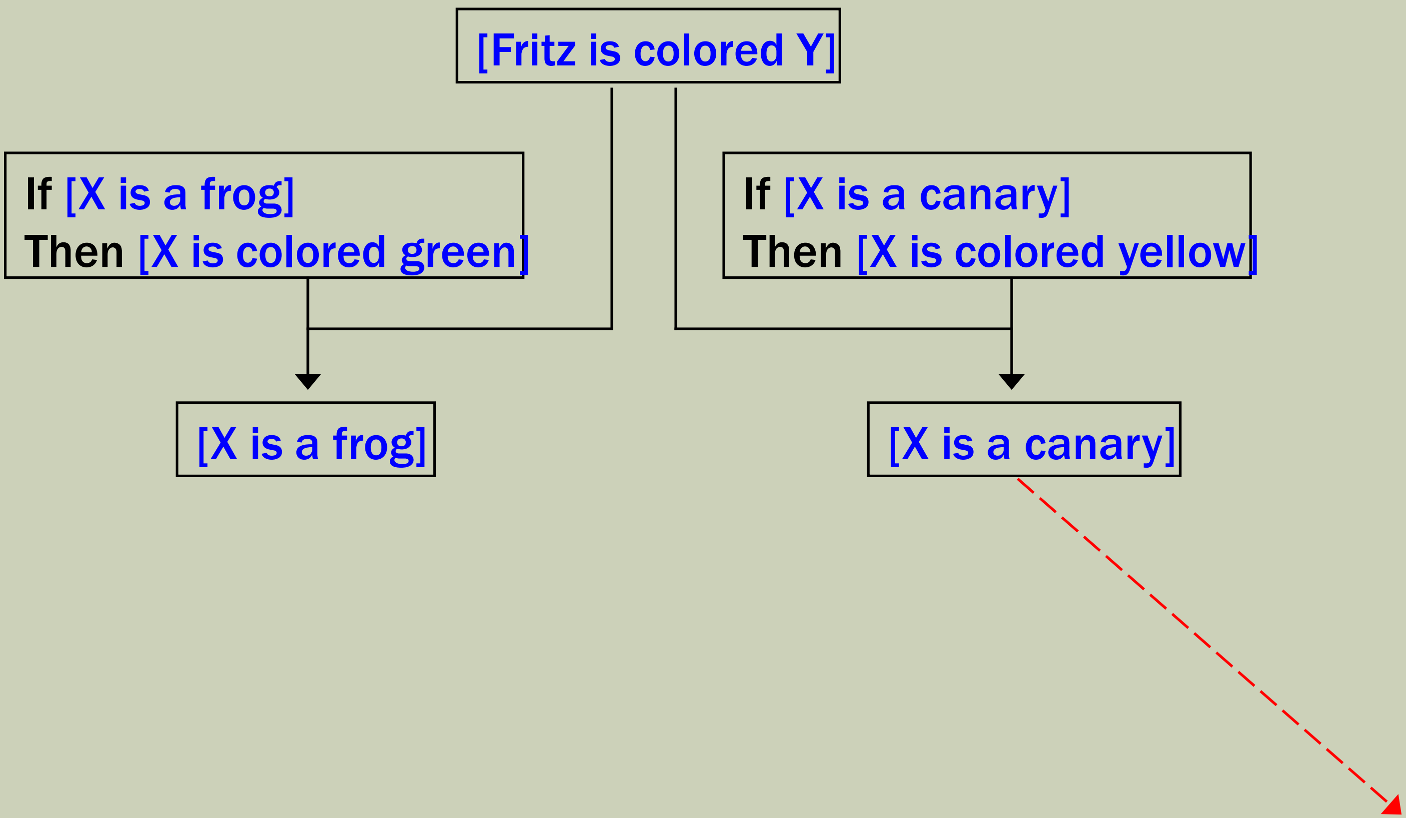
[Fritz croaks and eats flies]

Goals

[Fritz is colored Y]?

[X is a frog]

BACKWARD CHAINING EXAMPLE



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]

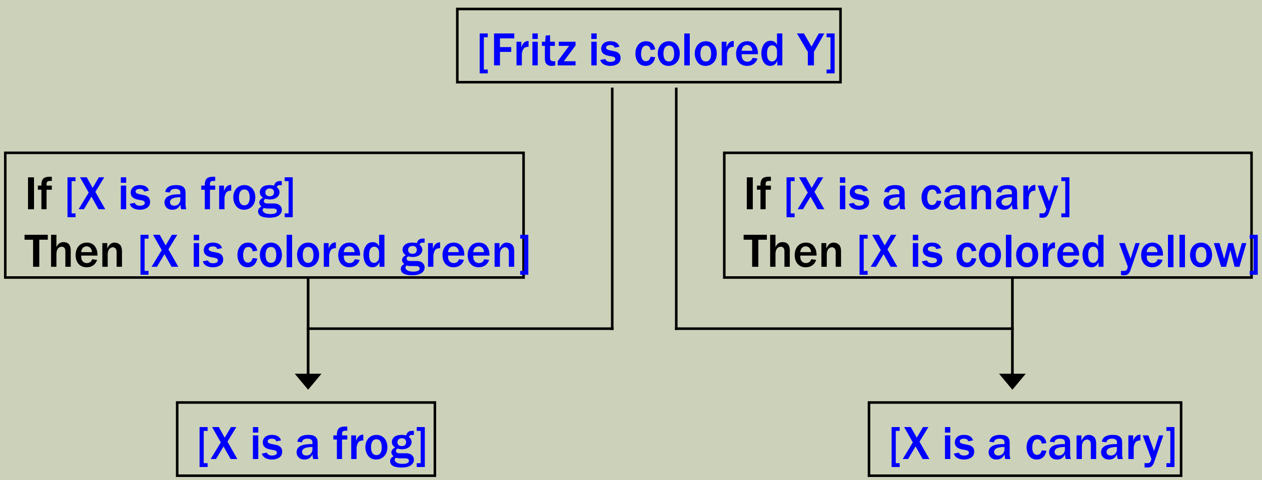
Goals

[Fritz is colored Y]?

[X is a frog]

[X is a canary]

BACKWARD CHAINING EXAMPLE



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]

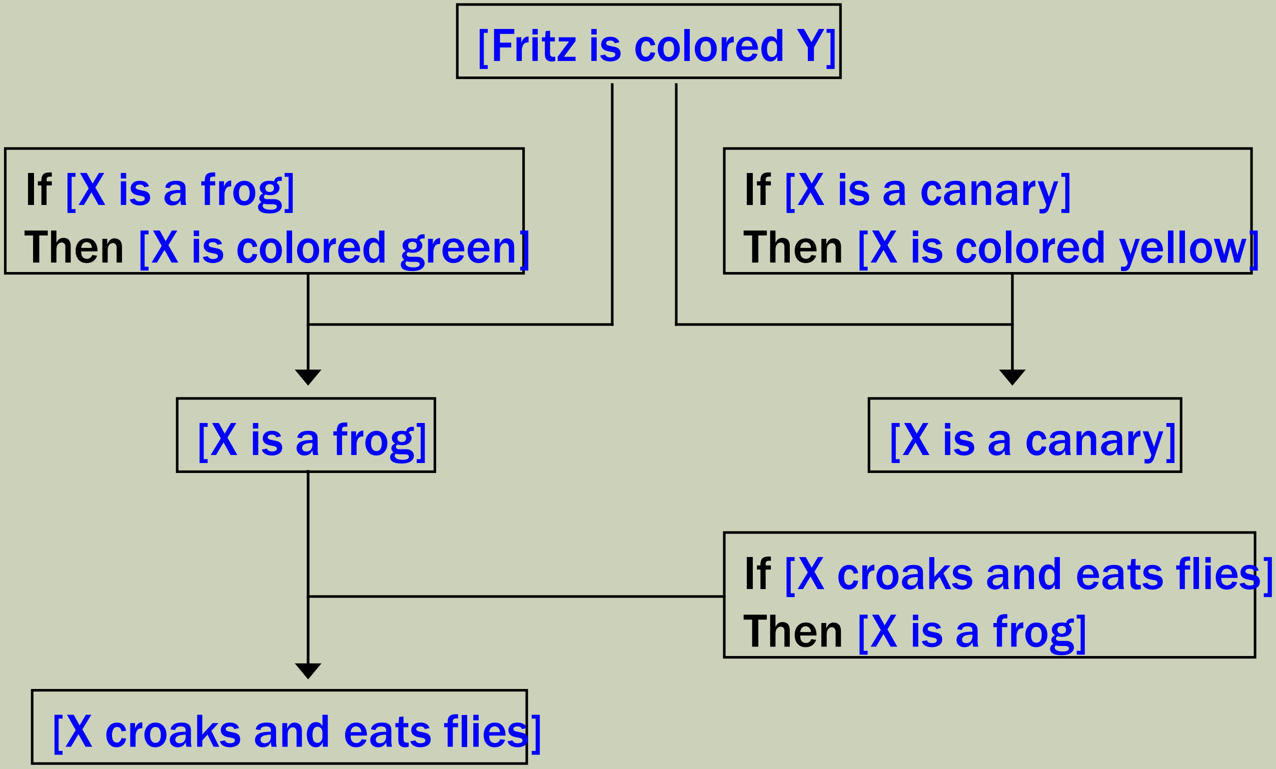
Goals

[Fritz is colored Y]?

[X is a frog]

[X is a canary]

BACKWARD CHAINING EXAMPLE



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]

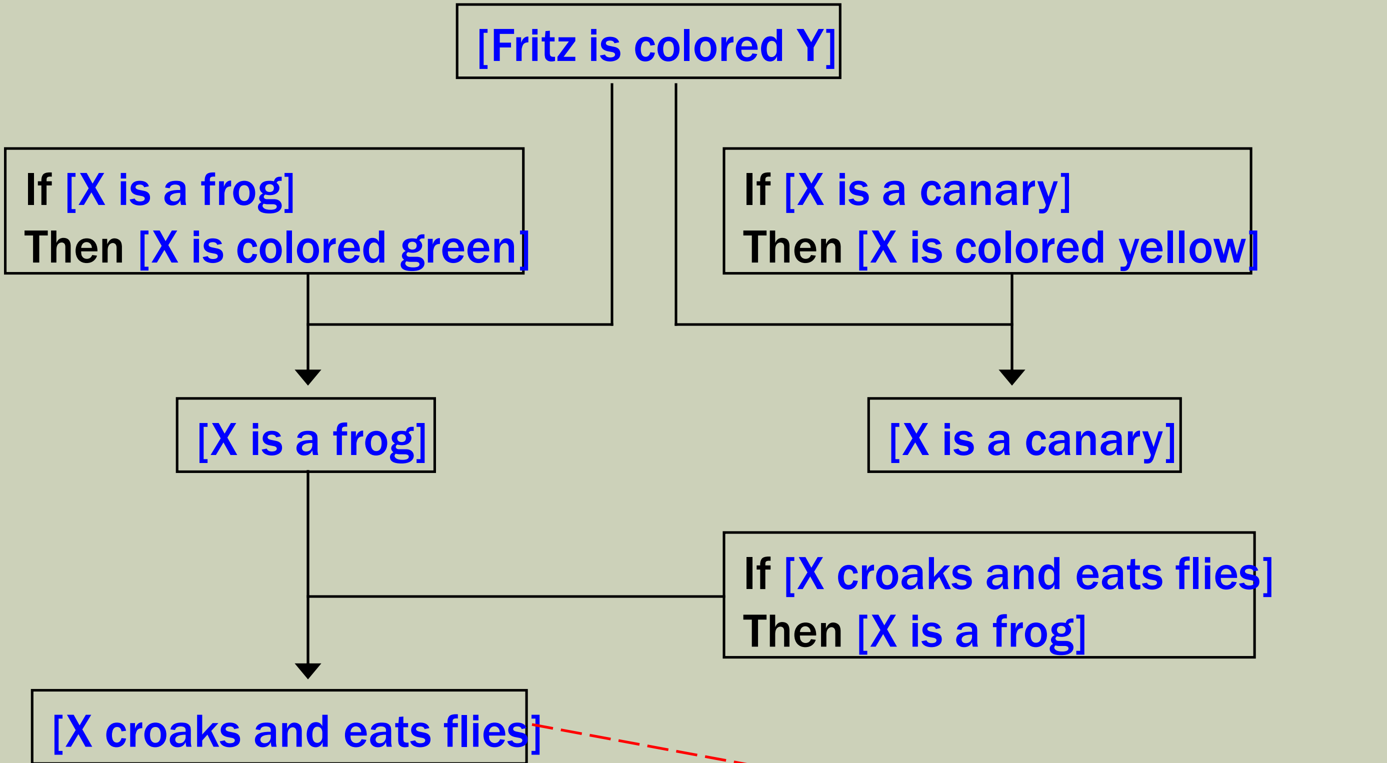
Goals

[Fritz is colored Y]?

[X is a frog]

[X is a canary]

BACKWARD CHAINING EXAMPLE



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

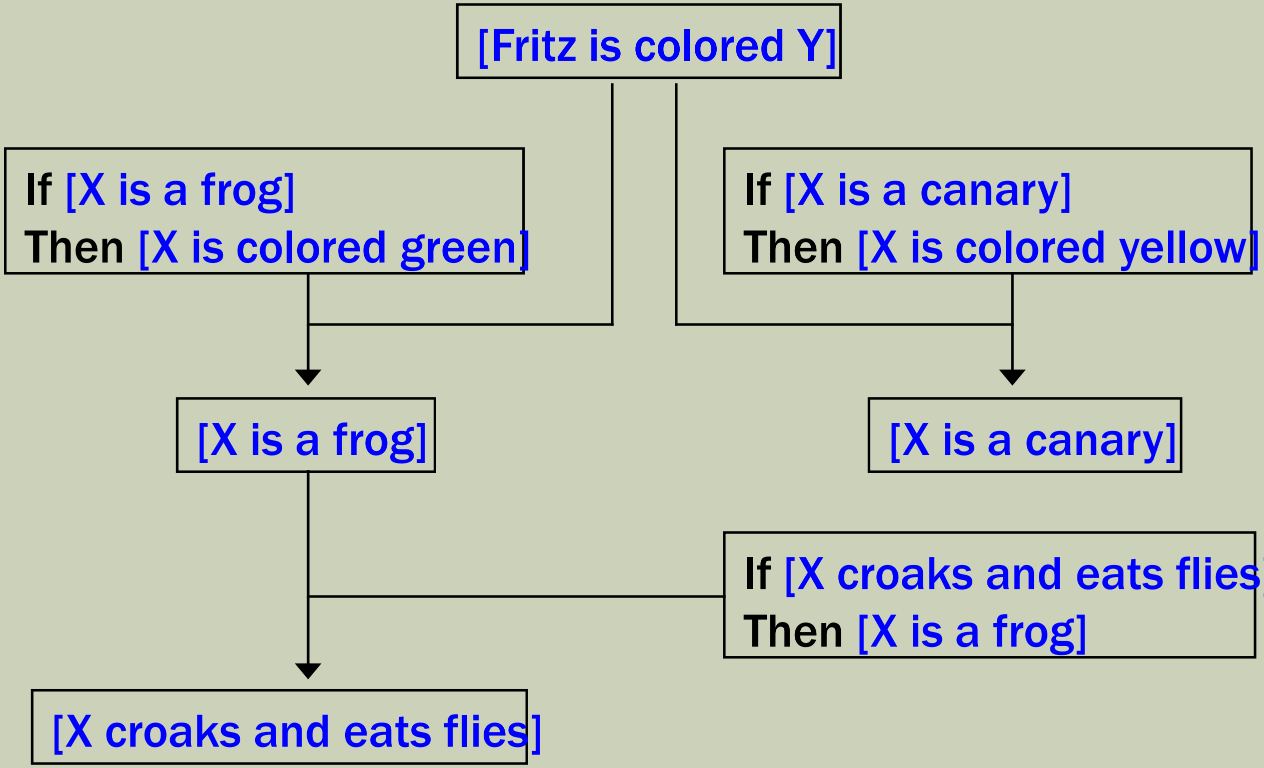
[Fritz is colored Y]?

[X is a frog]

[X is a canary]

[X croaks and eats flies]

BACKWARD CHAINING EXAMPLE



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

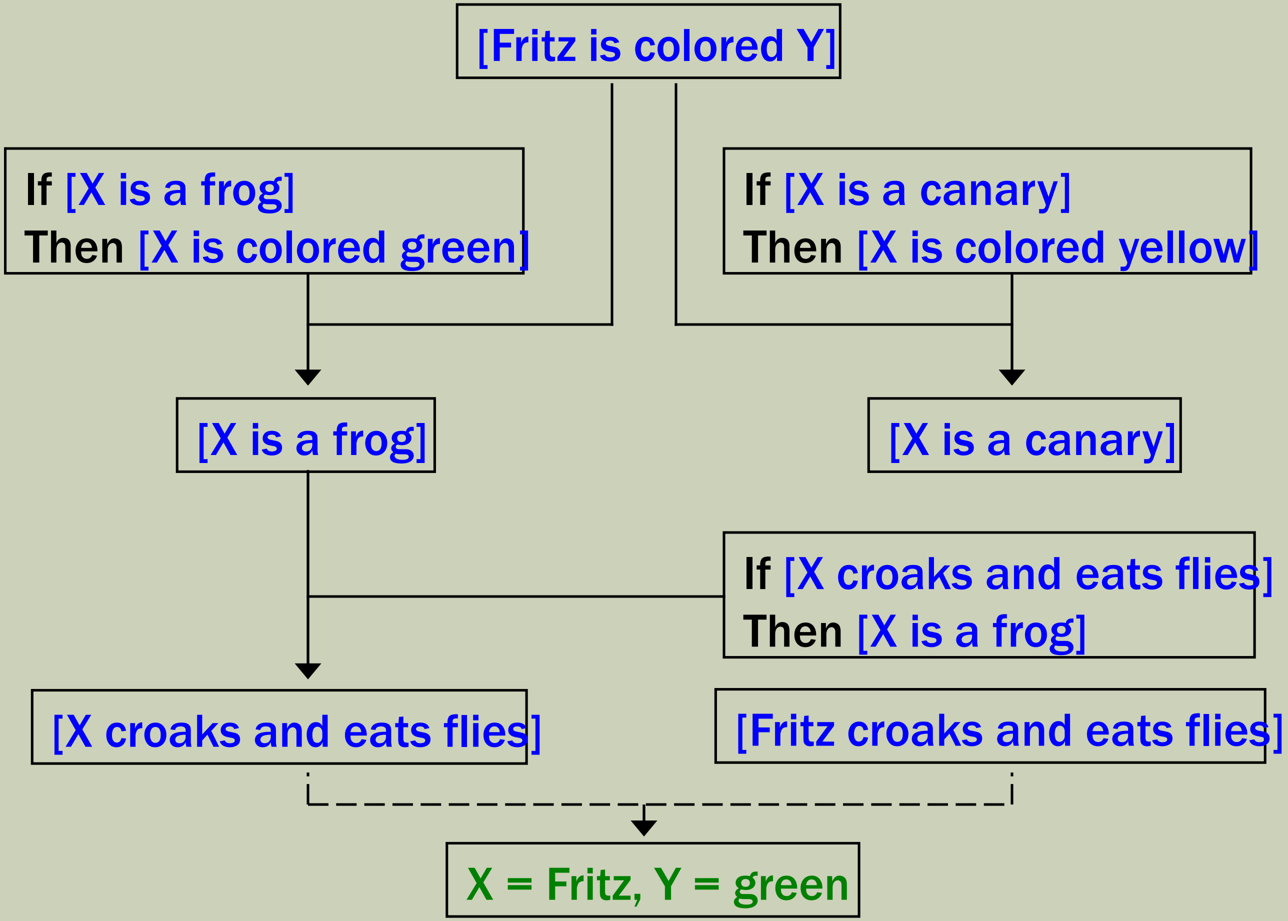
[Fritz is colored Y]?

[X is a frog]

[X is a canary]

[X croaks and eats flies]

BACKWARD CHAINING EXAMPLE



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

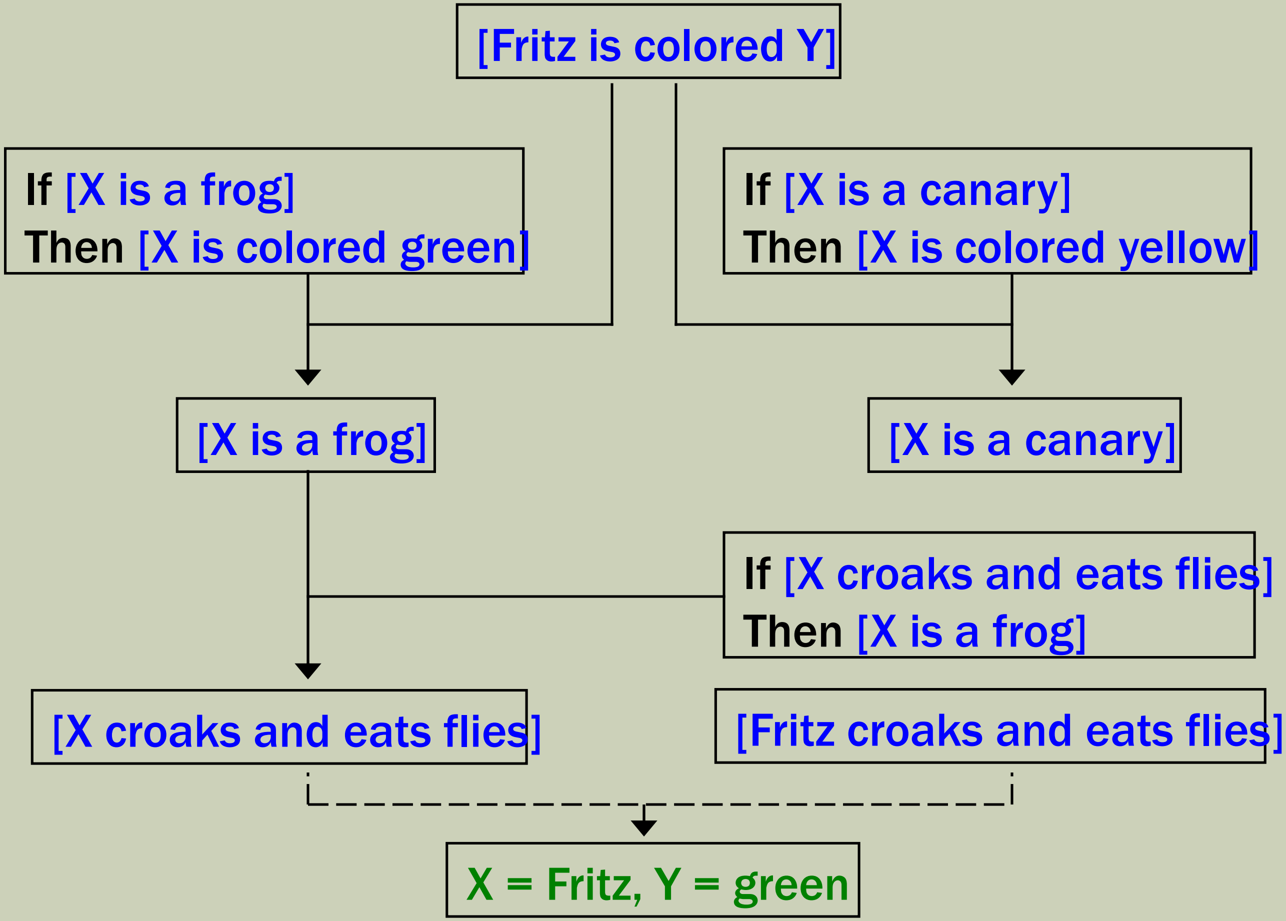
[Fritz is colored Y]?

[X is a frog]

[X is a canary]

[X croaks and eats flies]

BACKWARD CHAINING EXAMPLE



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

- S_1: for each x_1,y_1 child(x_1,y_1) ⇒ parent(y_1,x_1)
- S_2: for each x_2,y_2 parent(x_2,y_2) ∧ female(x_2) ⇒ mother(x_2,y_2)
- S_3: child(Lisa,Homer)
- S_4: child(Lisa,Marge)
- S_5: female(Marge)
- Goal: mother(x_0,Lisa)

BACKWARD CHAINING