Truth Maintenance

willie

Inference Engine

When a fact or rule is added to the KB

Use forward-chaining to infer new facts

Incremental approach allows us simplify how we match facts to rule

- Infer new rules

Infer New Facts

Start KB with a couple of facts

(barks Hershey)

(isa Dog Mammal)

(isa Mammal Animal)

Then add a rule

((barks ?d)) -> (inst ?d Dog) called Unification

Forward-chain to infer new facts

- Unify (barks ?d) with (barks Hershey)

  {?d/Hershey}
- Make substitutions

  ((barks Hershey)) -> (inst Hershey Dog)
- Add new fact to KB

More Complex Rules

If rules have multiple conditions, need to satisfy all before inferring new fact

Start KB with a couple of facts

(isa Dog Mammal)

(isa Mammal Animal)

Then add a rule

((isa?x ?y) (isa ?y ?z)) -> (isa ?x ?z)

Forward-chain to infer new rule

Infer New Rules

Start with first sentence in left-hand side (lhs) of rule

(isa ?x ?y)

Try to unify it with each fact in KB

(isa Dog Mammal)

When a substitution is found

- Make substitutions

  ((isa Dog Mammal) (isa Mammal ?z)) -> (isa Dog ?z)
- Create new rule with rest of rule

  ((isa Mammal ?z)) -> (isa Dog ?z)
- Add new rule to KB

Repeat

Added new rule

((isa Mammal ?z)) -> (isa Dog ?z)

Unify first sentence of LHS with facts in the KB

(isa Mammal Animal)

{?z/Animal}

Make substitutions

((isa Mammal Animal)) -> (isa Dog Animal)

Add new fact to KB

(isa Dog Animal)

Keep repeating

((isa?x ?y) (isa ?y ?z)) -> (isa ?x ?z)

(isa Dog Mammal)

(isa Mammal Animal)

(isa Animal LivingThing)

(isa LivingThing Thing)

...

(isa Dog Thing)

FC Algorithm For New Fact

```
Forward_chain(fact):
  For each rule in KB
    Get first statement in lhs of rule
    Unify fact with first statement
    If substitution found
      If lhs is only one statement
        Create new fact using the substitution on rhs
        Add new fact to kb
      Else
        Create new rule with rest of lhs and rhs
        Add new rule to kb
```

FC Algorithm For New **Rule**

```
Forward_chain(rule):
  For each fact in KB
    Get first statement in lhs of rule
    Unify fact with first statement
    If substitution found
      If lhs is only one statement
        Create new fact using the substitution on rhs
        Add new fact to kb
      Else
        Create new rule with rest of lhs and rhs
        Add new rule to kb
```

Truth Maintenance

Bookkeeping to track how facts came to be believed

Track which facts and rules support other facts and rules

Asserted facts and rules are assumed to be true

- No internal support for them

Everything else must have support

If some fact is inferred, then it is supported by the rule and fact used to infer the fact

Similarly, if some rule is inferred, then it is supported by the rule and fact used to infer the rule

TRUTH MAINTENANCE

Every fact/rule not asserted must have support

Arrows indicate support

Asserted facts and rules in the KB

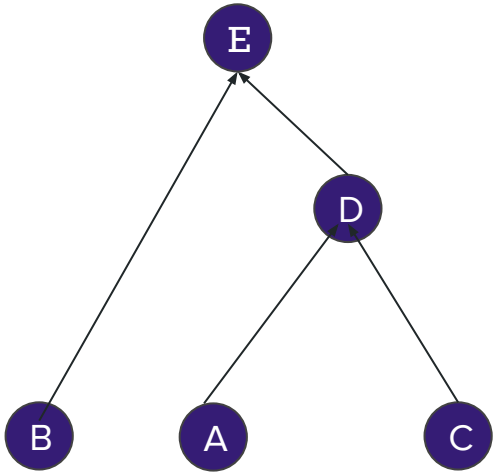
- A (isa Dog Mammal)
- B (isa Mammal Animal)
- C ((isa?x ?y) (isa ?y ?z)) -> (isa ?x ?z)

FC to infer

- E (isa Dog Animal)

via

- D ((isa Mammal ?z)) -> (isa Dog ?z)



“The Law says that it is a crime for an American to sell weapons to hostile nations. The country Nono, an enemy of America, has some missiles , and all of its missiles were sold to it by Colonel West, who is American”

1: ((american ?x) (weapon ?y) (sells ?x ?y ?z) (hostile ?z)) -> (criminal ?x)

A: (owns Nono M1)

B: (missile M1)

2: ((missile ?x)) -> (weapon ?x)

3: ((missile ?x) (owns Nono ?x)) -> (sells West ?x Nono)

4: ((enemy ?x America)) -> (hostile ?x)

C: (american West)

D: (enemy Nono America)

Assert each of the following statements into a KB
What facts and rules get inferred?

1: ((american ?x) (weapon ?y) (sells ?x ?y ?z) (hostile ?z)) -> (criminal ?x)

A: (owns Nono M1)

B: (missile M1)

2: ((missile ?x)) -> (weapon ?x)

3: ((missile ?x) (owns Nono ?x)) -> (sells West ?x Nono)

4: ((enemy ?x America)) -> (hostile ?x)

C: (american West)

D: (enemy Nono America)

Assert each of the following statements into a KB
What facts and rules get inferred?

1: ((american ?x) (weapon ?y) (sells ?x ?y ?z) (hostile ?z)) -> (criminal ?x)

A: (owns Nono M1)

B: (missile M1)

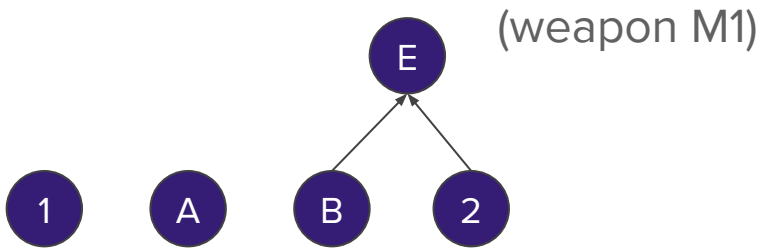
2: ((missile ?x)) -> (weapon ?x)

3: ((missile ?x) (owns Nono ?x)) -> (sells West ?x Nono)

4: ((enemy ?x America)) -> (hostile ?x)

C: (american West)

D: (enemy Nono America)



1: ((american ?x) (weapon ?y) (sells ?x ?y ?z) (hostile ?z)) -> (criminal ?x)

A: (owns Nono M1)

B: (missile M1)

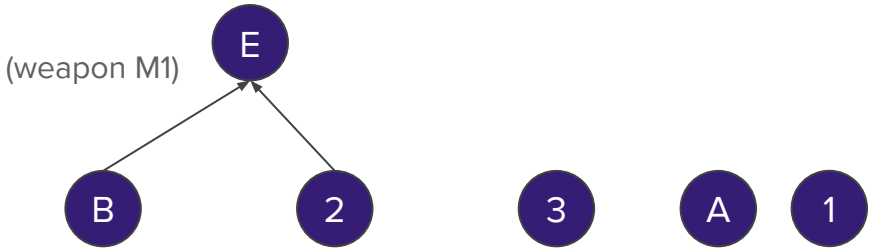
2: ((missile ?x)) -> (weapon ?x)

3: ((missile ?x) (owns Nono ?x)) -> (sells West ?x Nono)

4: ((enemy ?x America)) -> (hostile ?x)

C: (american West)

D: (enemy Nono America)



1: ((american ?x) (weapon ?y) (sells ?x ?y ?z) (hostile ?z)) -> (criminal ?x)

A: (owns Nono M1)

B: (missile M1)

2: ((missile ?x)) -> (weapon ?x)

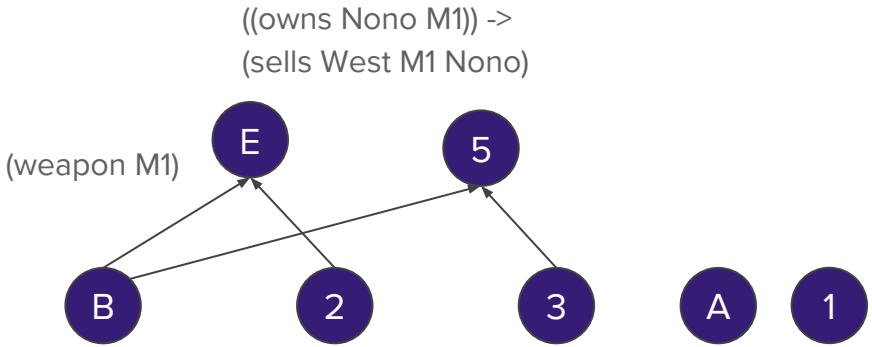
3: ((missile ?x) (owns Nono ?x)) -> (sells West ?x Nono)

4: ((enemy ?x America)) -> (hostile ?x)

C: (american West)

D: (enemy Nono America)

add 5 b/c we only want to look at the item on the lhs,
and once that matches we can add more items to the list



1: ((american ?x) (weapon ?y) (sells ?x ?y ?z) (hostile ?z)) -> (criminal ?x)

A: (owns Nono M1)

B: (missile M1)

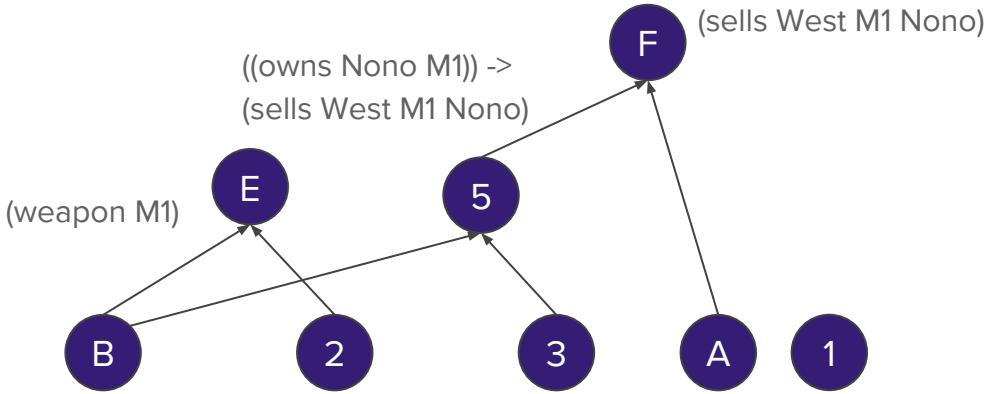
2: ((missile ?x)) -> (weapon ?x)

3: ((missile ?x) (owns Nono ?x)) -> (sells West ?x Nono)

4: ((enemy ?x America)) -> (hostile ?x)

C: (american West)

D: (enemy Nono America)



1: ((american ?x) (weapon ?y) (sells ?x ?y ?z) (hostile ?z)) -> (criminal ?x)

A: (owns Nono M1)

B: (missile M1)

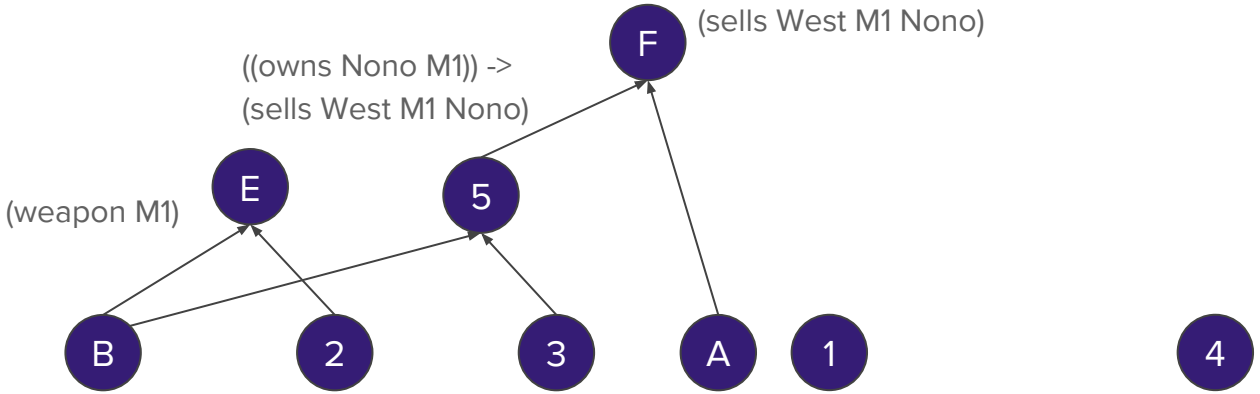
2: ((missile ?x)) -> (weapon ?x)

3: ((missile ?x) (owns Nono ?x)) -> (sells West ?x Nono)

4: ((enemy ?x America)) -> (hostile ?x)

C: (american West)

D: (enemy Nono America)



1: ((american ?x) (weapon ?y) (sells ?x ?y ?z) (hostile ?z)) -> (criminal ?x)

A: (owns Nono M1)

B: (missile M1)

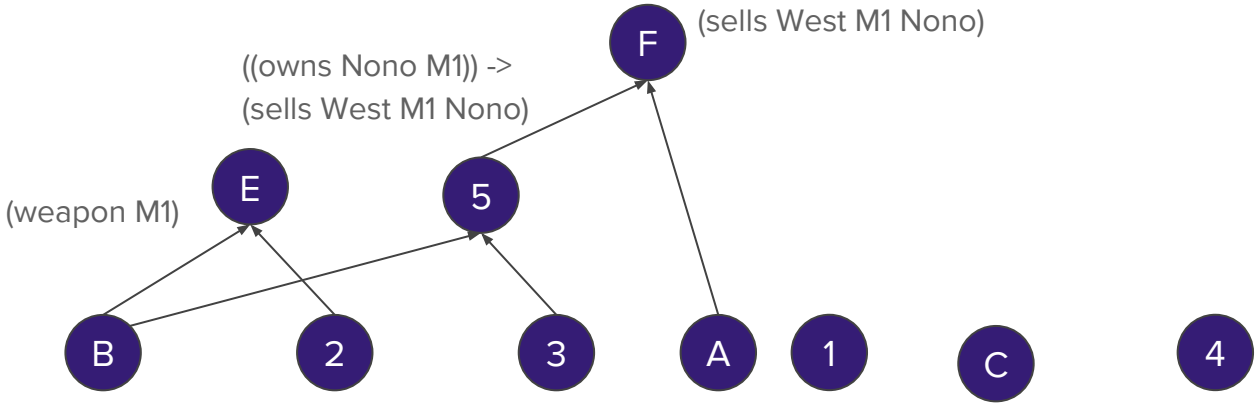
2: ((missile ?x)) -> (weapon ?x)

3: ((missile ?x) (owns Nono ?x)) -> (sells West ?x Nono)

4: ((enemy ?x America)) -> (hostile ?x)

C: (american West)

D: (enemy Nono America)



1: ((american ?x) (weapon ?y) (sells ?x ?y ?z) (hostile ?z)) -> (criminal ?x)

A: (owns Nono M1)

B: (missile M1)

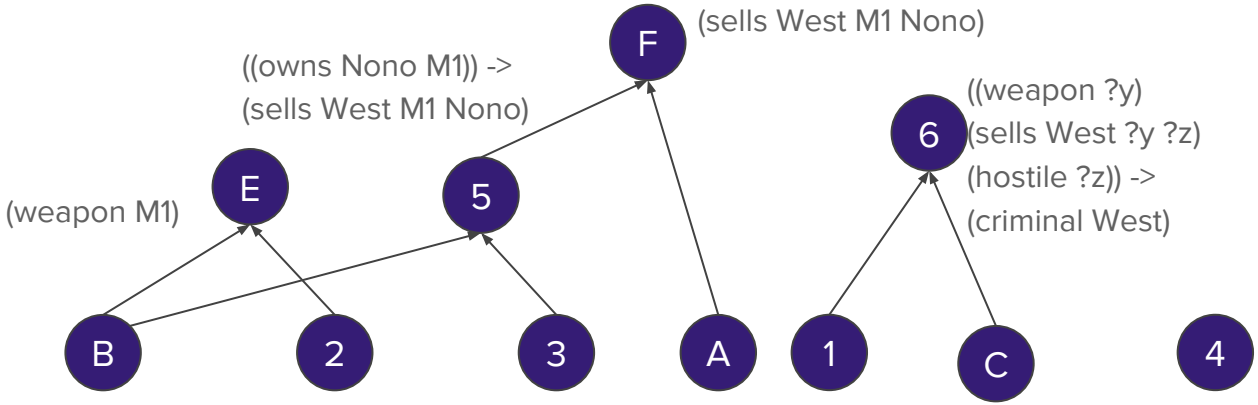
2: ((missile ?x)) -> (weapon ?x)

3: ((missile ?x) (owns Nono ?x)) -> (sells West ?x Nono)

4: ((enemy ?x America)) -> (hostile ?x)

C: (american West)

D: (enemy Nono America)



1: ((american ?x) (weapon ?y) (sells ?x ?y ?z) (hostile ?z)) -> (criminal ?x)

A: (owns Nono M1)

B: (missile M1)

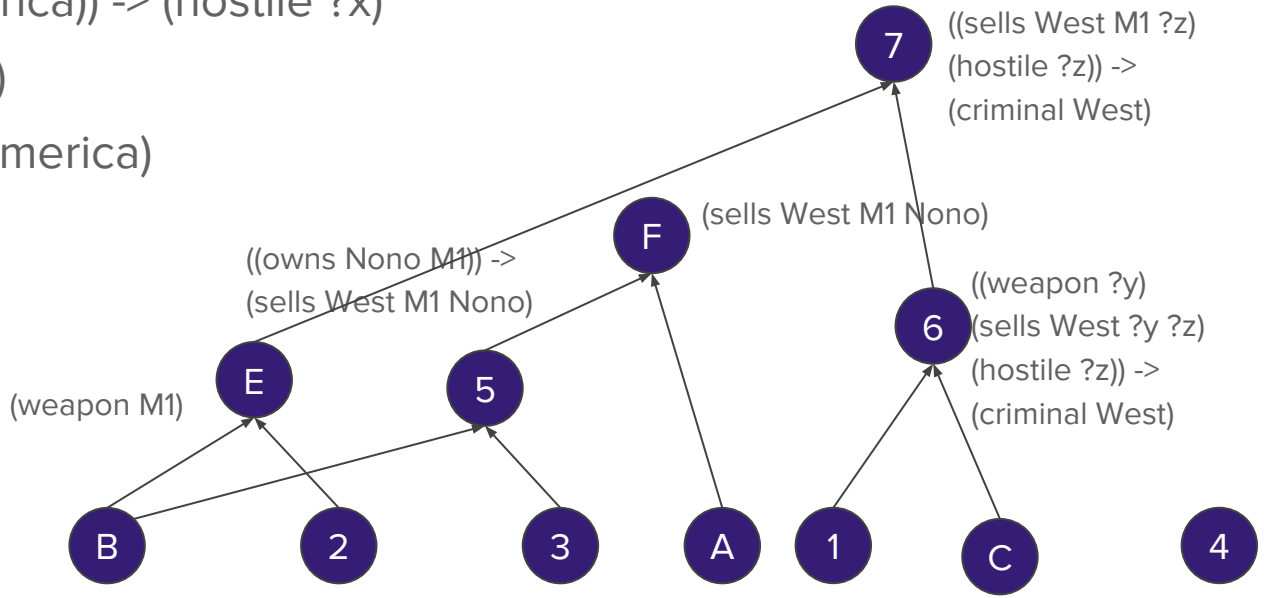
2: ((missile ?x)) -> (weapon ?x)

3: ((missile ?x) (owns Nono ?x)) -> (sells West ?x Nono)

4: ((enemy ?x America)) -> (hostile ?x)

C: (american West)

D: (enemy Nono America)



1: ((american ?x) (weapon ?y) (sells ?x ?y ?z) (hostile ?z)) -> (criminal ?x)

A: (owns Nono M1)

B: (missile M1)

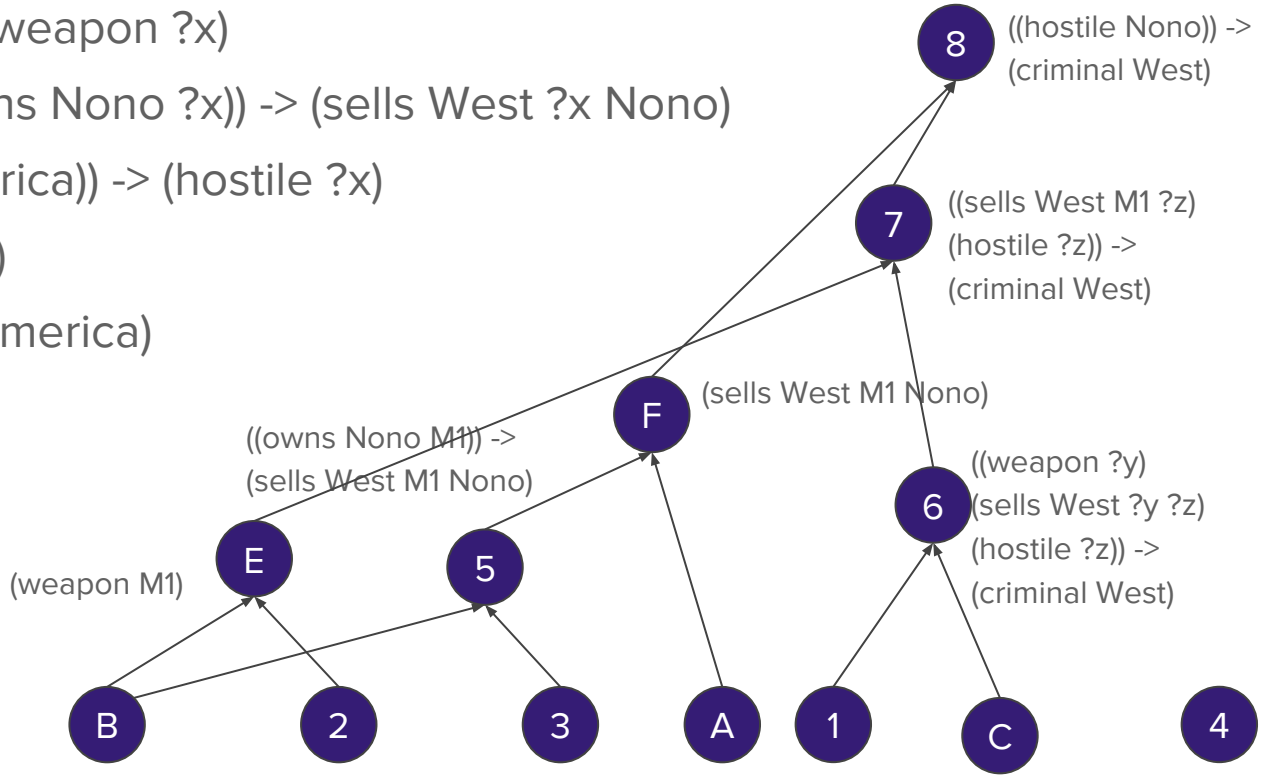
2: ((missile ?x)) -> (weapon ?x)

3: ((missile ?x) (owns Nono ?x)) -> (sells West ?x Nono)

4: ((enemy ?x America)) -> (hostile ?x)

C: (american West)

D: (enemy Nono America)



1: ((american ?x) (weapon ?y) (sells ?x ?y ?z) (hostile ?z)) -> (criminal ?x)

A: (owns Nono M1)

B: (missile M1)

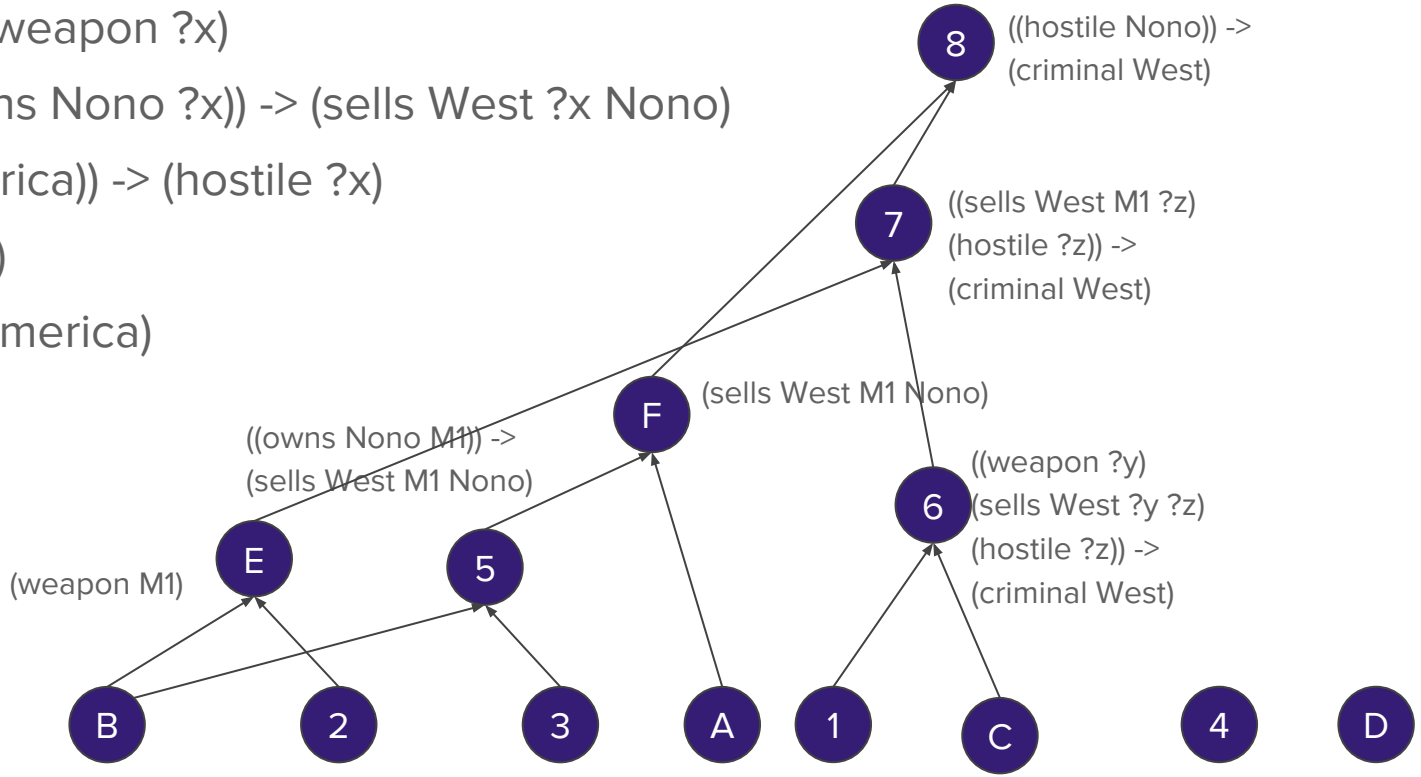
2: ((missile ?x)) -> (weapon ?x)

3: ((missile ?x) (owns Nono ?x)) -> (sells West ?x Nono)

4: ((enemy ?x America)) -> (hostile ?x)

C: (american West)

D: (enemy Nono America)



1: ((american ?x) (weapon ?y) (sells ?x ?y ?z) (hostile ?z)) -> (criminal ?x)

A: (owns Nono M1)

B: (missile M1)

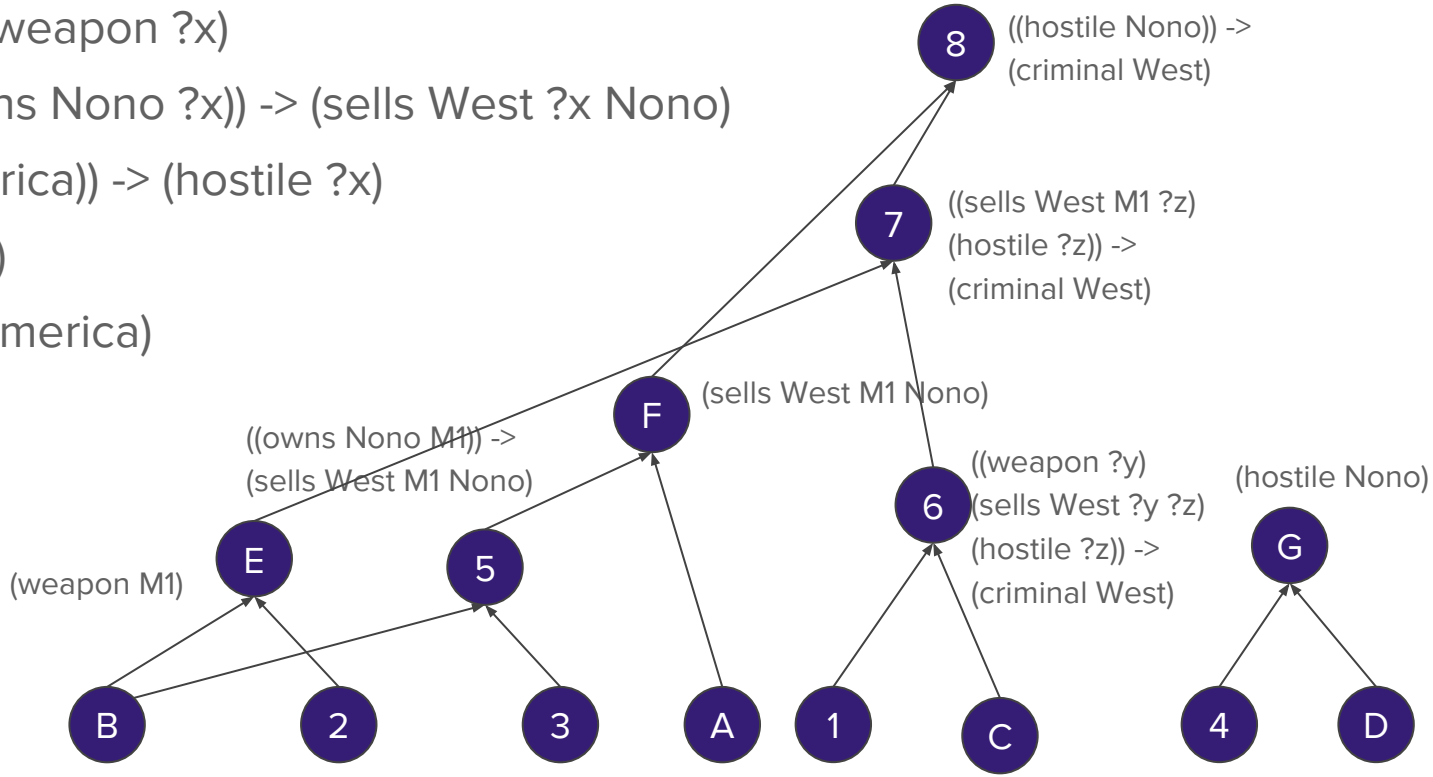
2: ((missile ?x)) -> (weapon ?x)

3: ((missile ?x) (owns Nono ?x)) -> (sells West ?x Nono)

4: ((enemy ?x America)) -> (hostile ?x)

C: (american West)

D: (enemy Nono America)



1: ((american ?x) (weapon ?y) (sells ?x ?y ?z) (hostile ?z)) -> (criminal ?x)

A: (owns Nono M1)

B: (missile M1)

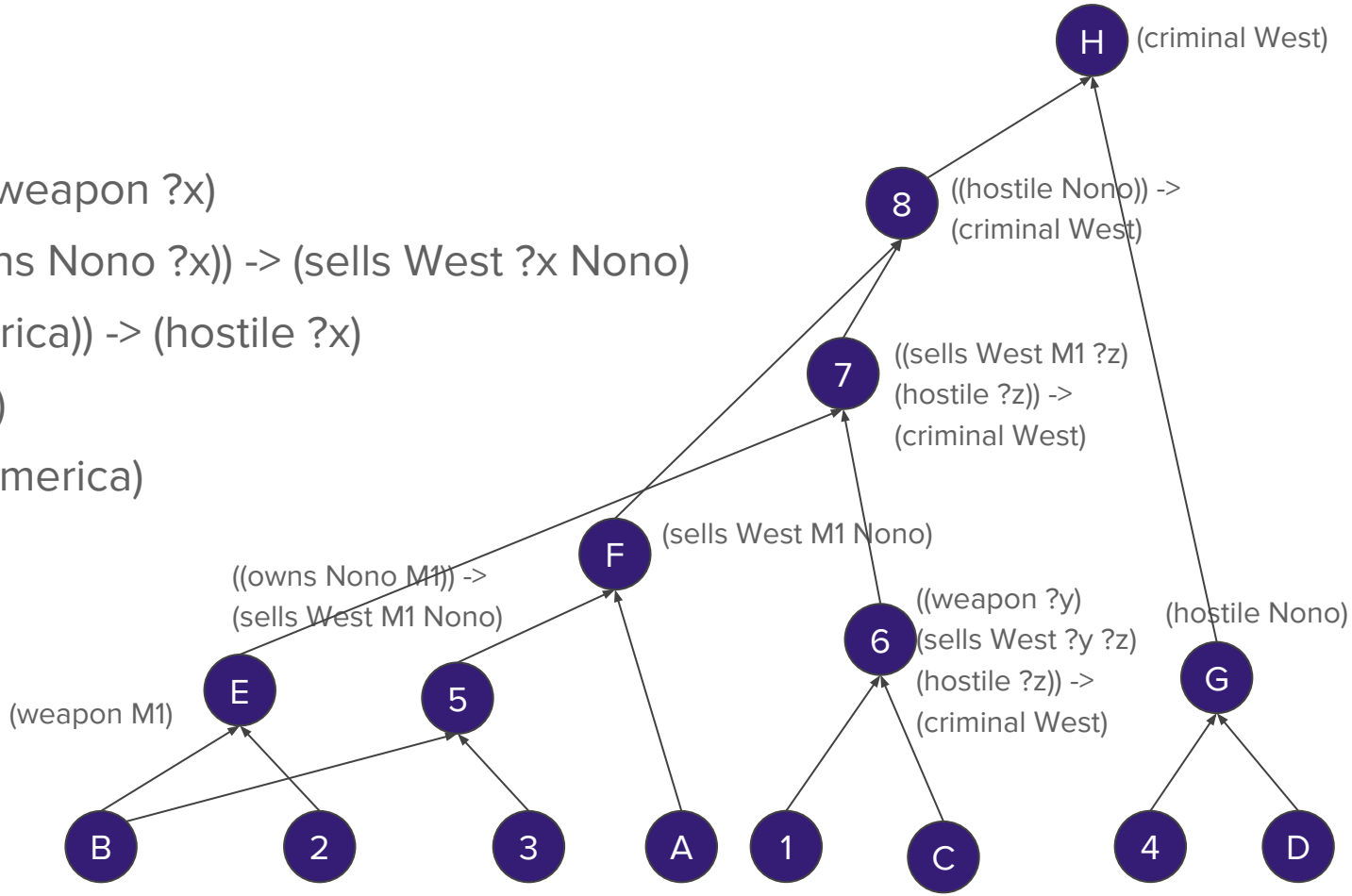
2: ((missile ?x)) -> (weapon ?x)

3: ((missile ?x) (owns Nono ?x)) -> (sells West ?x Nono)

4: ((enemy ?x America)) -> (hostile ?x)

C: (american West)

D: (enemy Nono America)



****How do we get an explanation for an answer?****

Why?

How do we know a Dog is an Animal?

(isa Dog Animal)

If we didn't have visibility, we wouldn't be able to inspect and figure out what wasn't true.

Explanation

We know

All Dogs are Mammals

All Mammals are Animals

All things that are Mammals are Animals

Retract Facts

Tracking truths also helps when retracting facts

When retracting a fact, not only is that fact no longer believed to be true

But everything it supports also may no longer be true

Retracting Facts – Example

Asserted 1: ((pred A) (pred B)) -> (pred C)
2: ((pred C) (pred D)) -> (pred E)
A: (pred A)

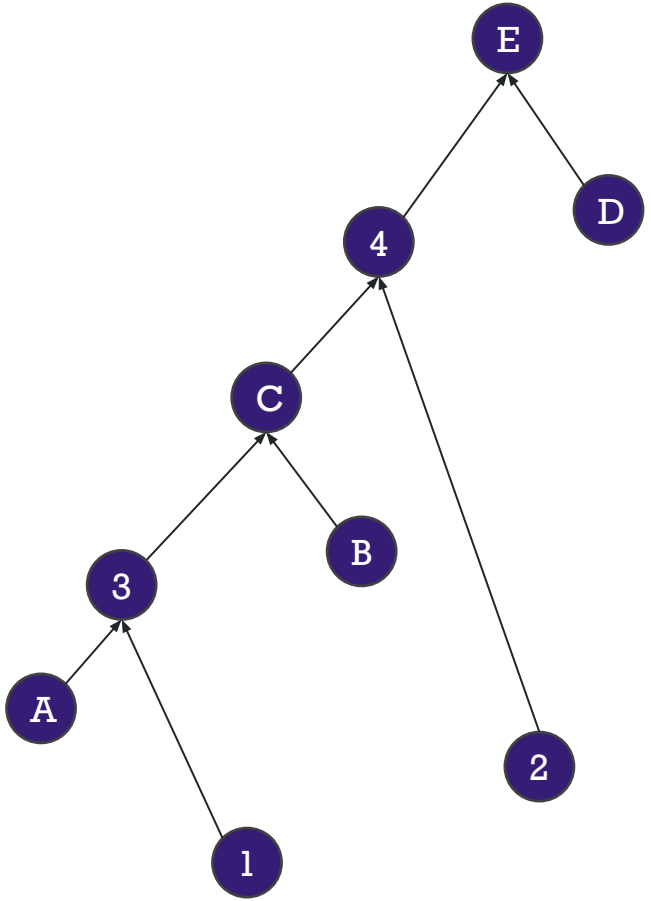
Inferred 3: ((pred B)) -> (pred C)

Asserted B: (pred B)

Inferred C: (pred C)
4: ((pred D)) -> (pred E)

Asserted D: (pred D)

Inferred E: (pred E)



Retracting Facts – Example

Asserted 1: ((pred A) (pred B)) -> (pred C)
2: ((pred C) (pred D)) -> (pred E)
A: (pred A)

Inferred 3: ((pred B)) -> (pred C)

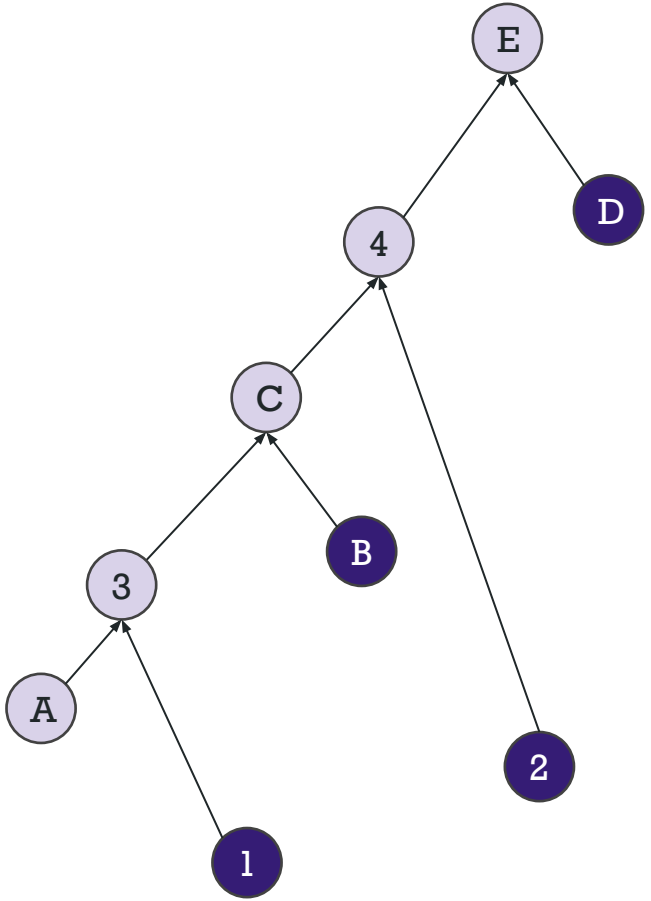
Asserted B: (pred B)

Inferred C: (pred C)
4: ((pred D)) -> (pred E)

Asserted D: (pred D)

Inferred E: (pred E)

Retracted (pred A)



Retracting Facts – Additional Support

Suppose C had some additional support.

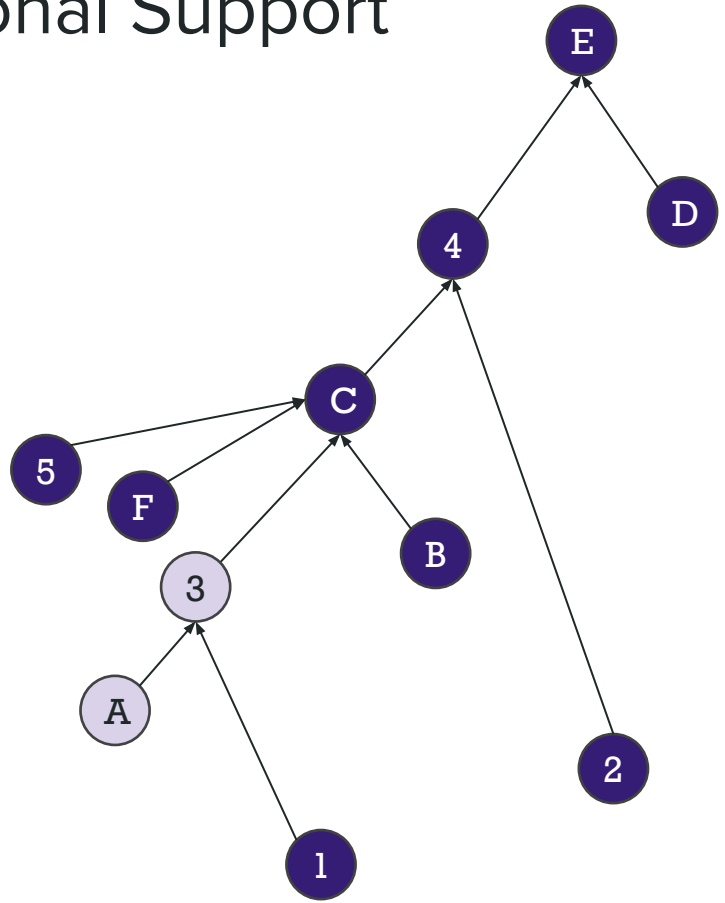
Asserted F: (pred F)
5: ((pred F)) -> (pred C)

Inferred C: (pred C)

Retracted (pred A)

C is not retracted
Other support from F and 5

4 and E remain supported



RETRACTING FACTS

- Checking for other support not always enough

- If breathes fire and flies then evil
- Retract nosliw is a dragon
- Retract nosliw is evil, but it has other support, don’t retract it
  - Uh oh
  - Phew, it still gets retracted

RETRACTING FACTS

- Checking for other support not always enough
- Consider the following:

  A: (isa Nosliw Dragon)

  B: (evil Dragon)

  C: (breathes_fire Nosliw)

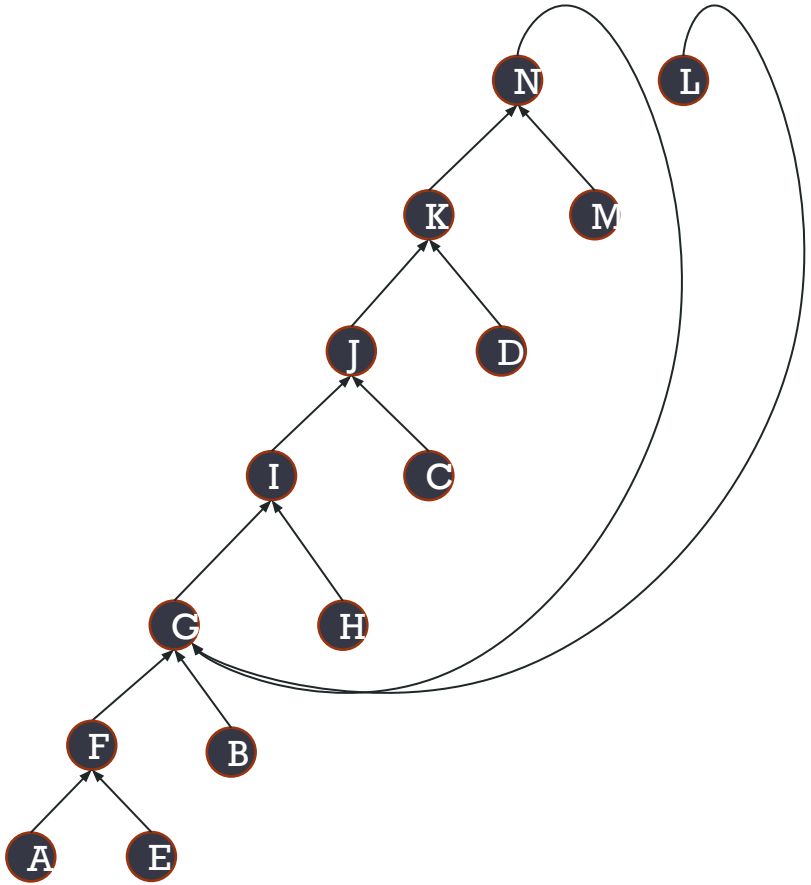
  D: (near Nosliw HappyDale)

  E: ((isa ?x ?y) (evil ?y) -> (evil ?x))

  H: ((evil Nosliw) (breathes_fire Nosliw)
  (near Nosliw HappyDale)) -> (destroyed HappyDale)

  L: (happy Nosliw)

  M: ((destroyed ?v) (happy ?d)) -> (evil ?d)



Retracting Facts - Circular Logic

1 ((wags-tail ?x)) -> (dog ?x)

2 ((tail ?x) (barks ?x)) -> (dog ?x)

A (tail Hershey)

B (barks Hershey)

4 ((dog ?x) (happy ?x)) -> (wags-tail ?x)

D (happy Hershey)

Retracting Facts - Circular Logic

1 ((wags-tail ?x)) -> (dog ?x)

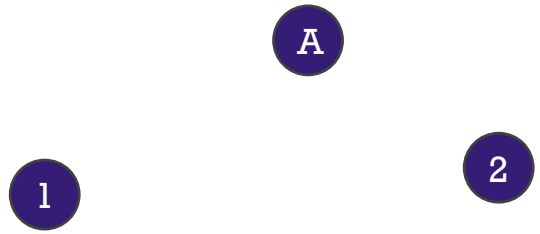
2 ((tail ?x) (barks ?x)) -> (dog ?x)

A (tail Hershey)

B (barks Hershey)

4 ((dog ?x) (happy ?x)) -> (wags-tail ?x)

D (happy Hershey)



Retracting Facts - Circular Logic

1 ((wags-tail ?x)) -> (dog ?x)

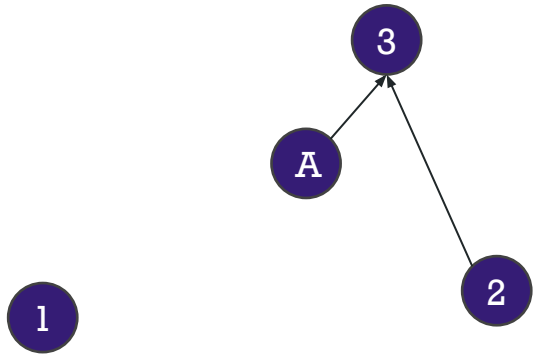
2 ((tail ?x) (barks ?x)) -> (dog ?x)

A (tail Hershey)

B (barks Hershey)

4 ((dog ?x) (happy ?x)) -> (wags-tail ?x)

D (happy Hershey)



Retracting Facts - Circular Logic

1 ((wags-tail ?x)) -> (dog ?x)

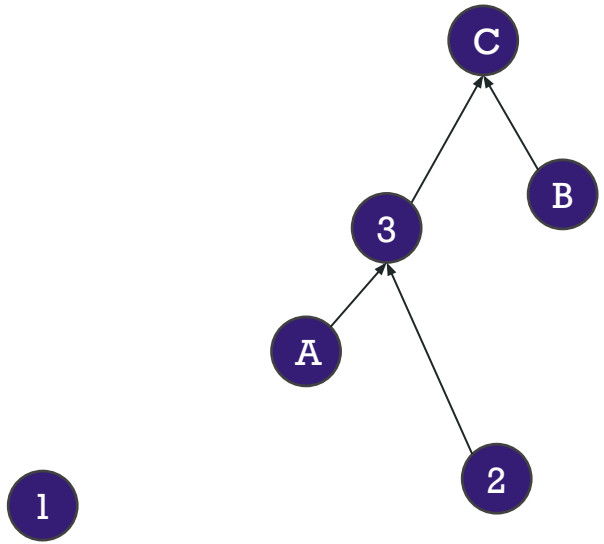
2 ((tail ?x) (barks ?x)) -> (dog ?x)

A (tail Hershey)

B (barks Hershey)

4 ((dog ?x) (happy ?x)) -> (wags-tail ?x)

D (happy Hershey)



Retracting Facts - Circular Logic

1 ((wags-tail ?x)) -> (dog ?x)

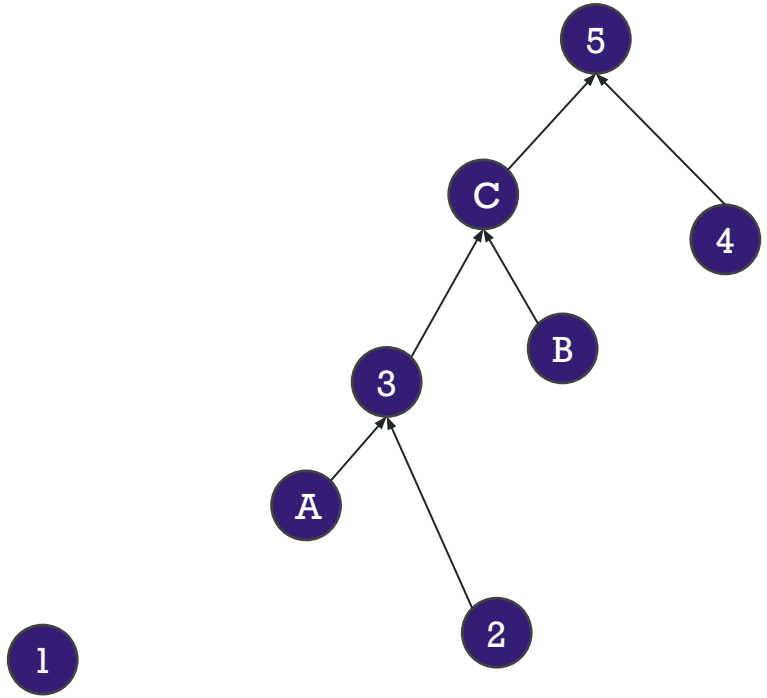
2 ((tail ?x) (barks ?x)) -> (dog ?x)

A (tail Hershey)

B (barks Hershey)

4 ((dog ?x) (happy ?x)) -> (wags-tail ?x)

D (happy Hershey)



Retracting Facts - Circular Logic

1 ((wags-tail ?x)) -> (dog ?x)

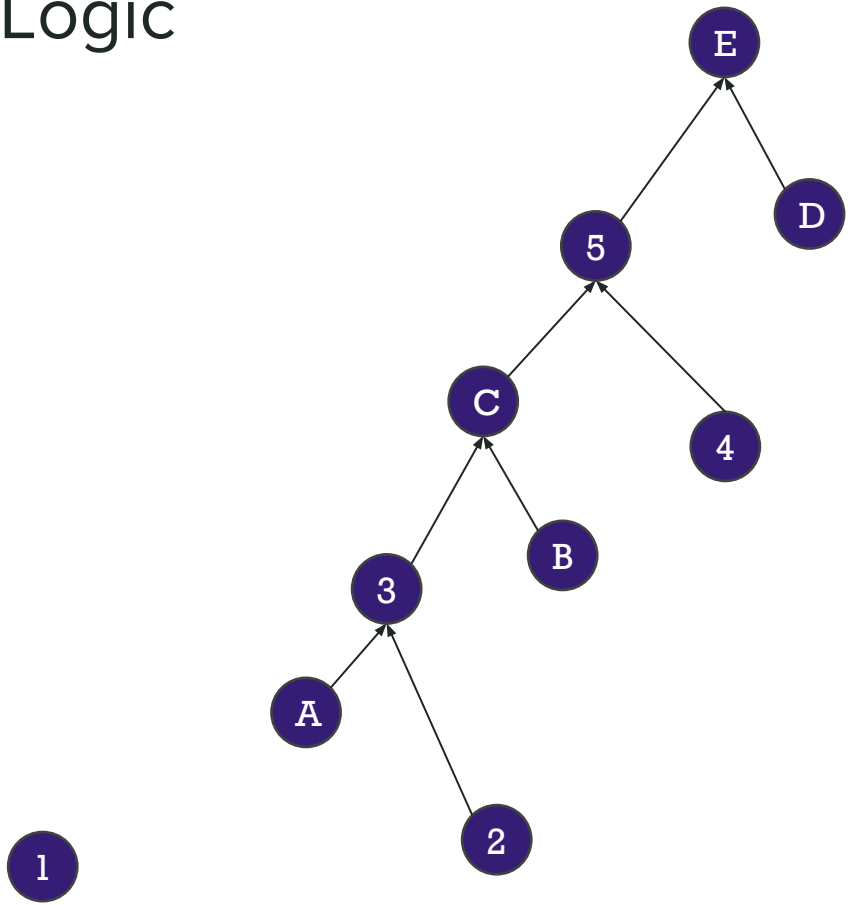
2 ((tail ?x) (barks ?x)) -> (dog ?x)

A (tail Hershey)

B (barks Hershey)

4 ((dog ?x) (happy ?x)) -> (wags-tail ?x)

D (happy Hershey)



Retracting Facts - Circular Logic

1 ((wags-tail ?x)) -> (dog ?x)

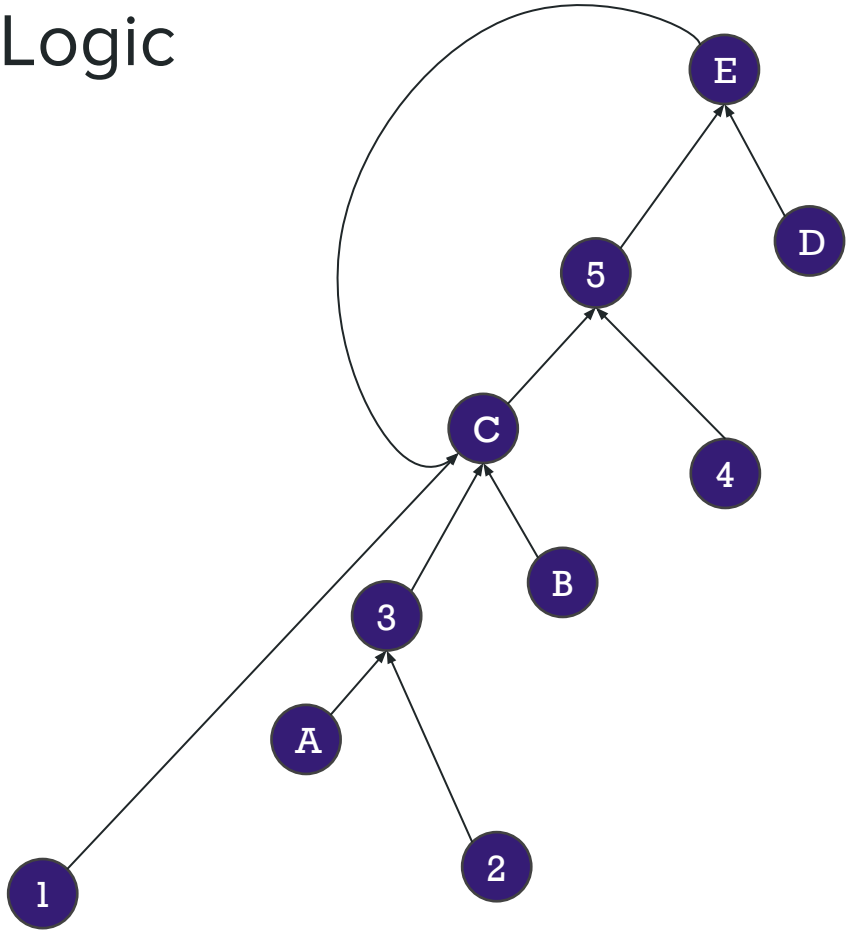
2 ((tail ?x) (barks ?x)) -> (dog ?x)

A (tail Hershey)

B (barks Hershey)

4 ((dog ?x) (happy ?x)) -> (wags-tail ?x)

D (happy Hershey)



Retracting Facts - Circular Logic

1 ((wags-tail ?x)) -> (dog ?x)

2 ((tail ?x) (barks ?x)) -> (dog ?x)

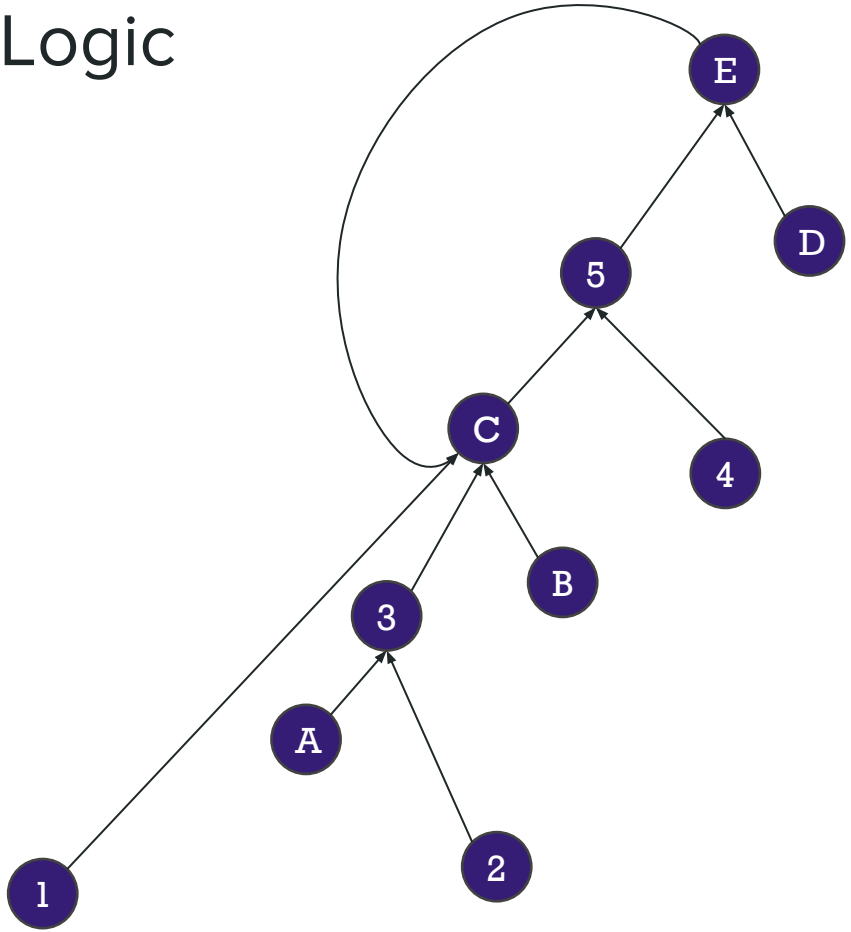
A (tail Hershey)

B (barks Hershey)

4 ((dog ?x) (happy ?x)) -> (wags-tail ?x)

D (happy Hershey)

Retract: (tail Hershey)



Retracting Facts - Circular Logic

1 ((wags-tail ?x)) -> (dog ?x)

2 ((tail ?x) (barks ?x)) -> (dog ?x)

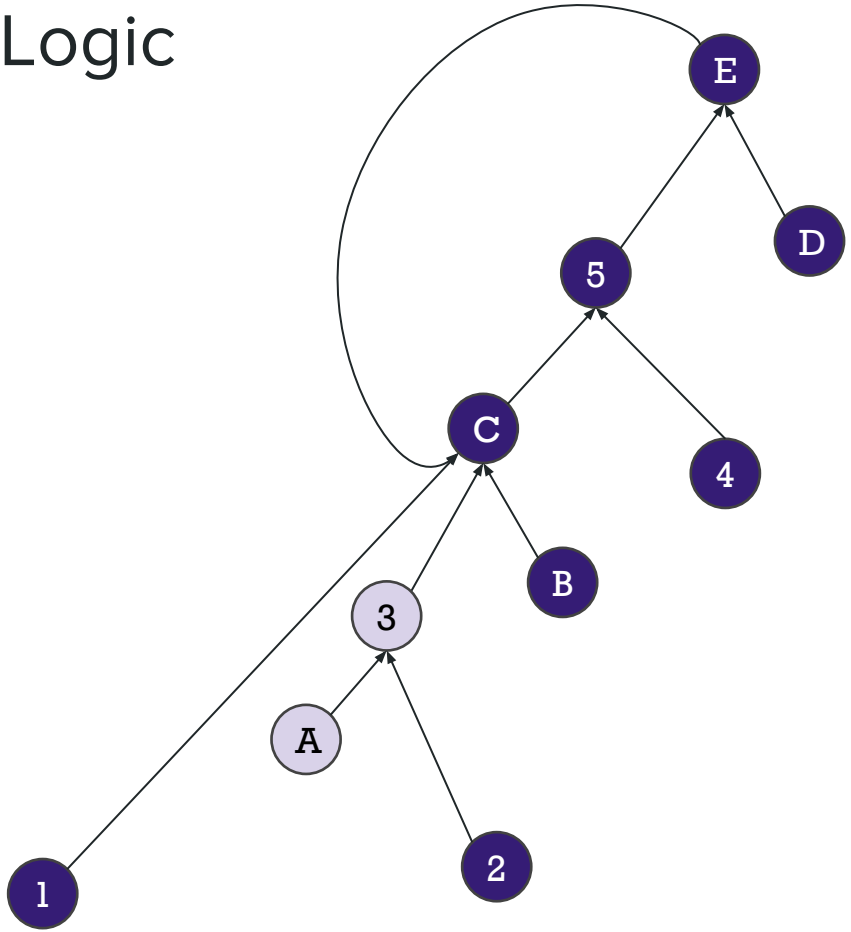
A (tail Hershey)

B (barks Hershey)

4 ((dog ?x) (happy ?x)) -> (wags-tail ?x)

D (happy Hershey)

Retract: (tail Hershey)



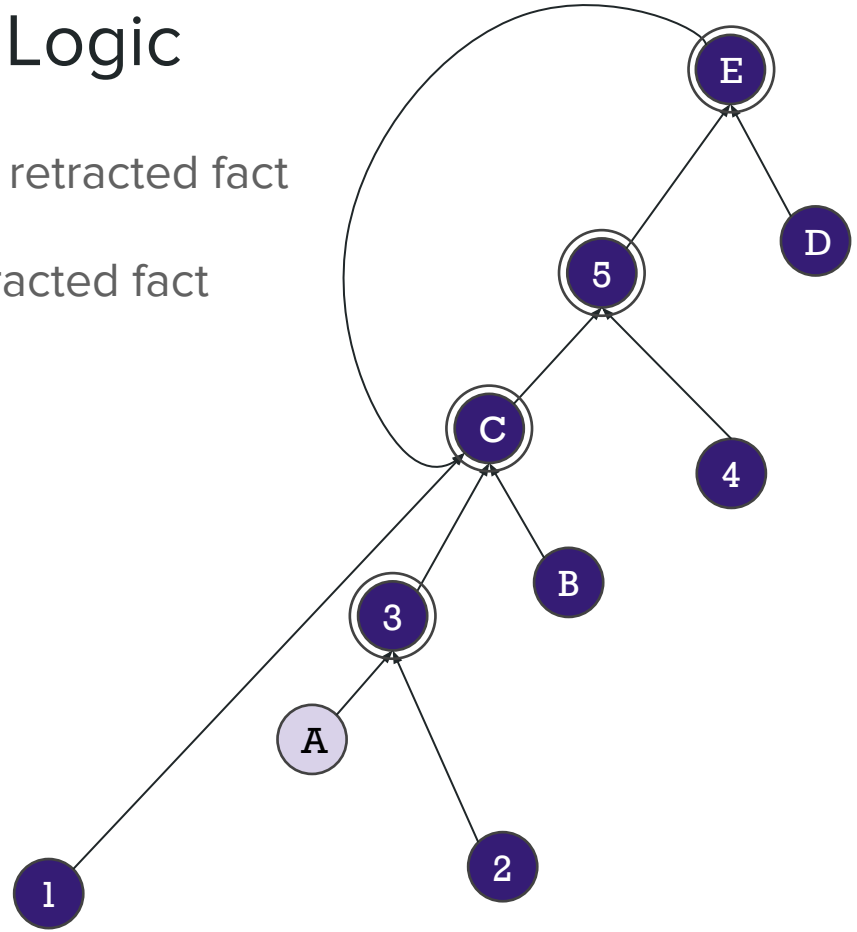
Retracting Facts - Circular Logic

Mark everything that is supported by the retracted fact

Trace back through again, starting at retracted fact

Retract anything that

- Has no other support
- Has no other unmarked support



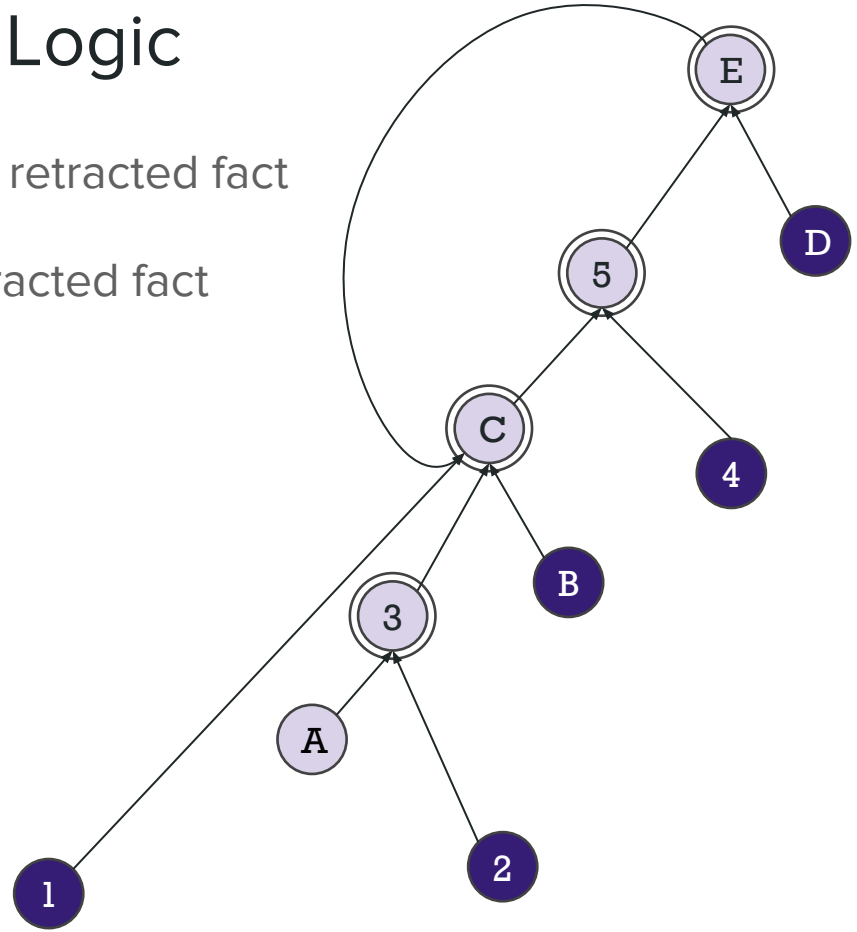
Retracting Facts - Circular Logic

Mark everything that is supported by the retracted fact

Trace back through again, starting at retracted fact

Retract anything that

- Has no other support
- Has no other unmarked support



Retracting Facts - Supported Facts

1 ((wags-tail ?x)) -> (dog ?x)

2 ((tail ?x) (barks ?x)) -> (dog ?x)

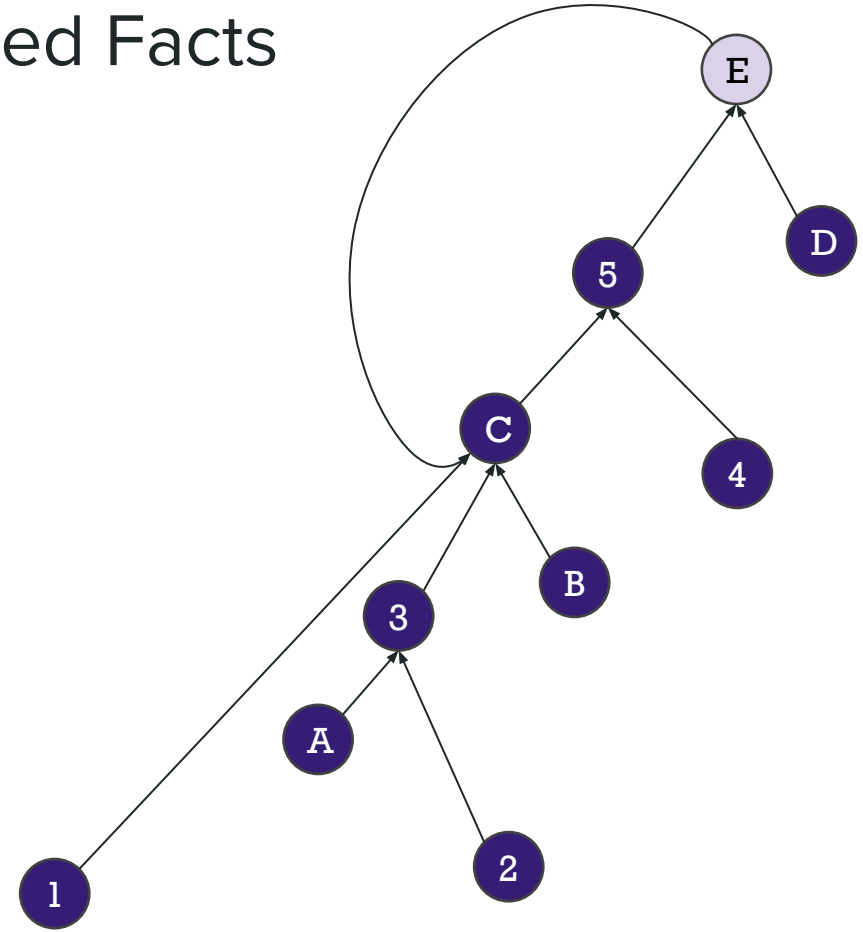
A (tail Hershey)

B (barks Hershey)

4 ((dog ?x) (happy ?x)) -> (wags-tail ?x)

D (happy Hershey)

Retract: (wags-tail Hershey)



Retracting Facts - Supported Facts

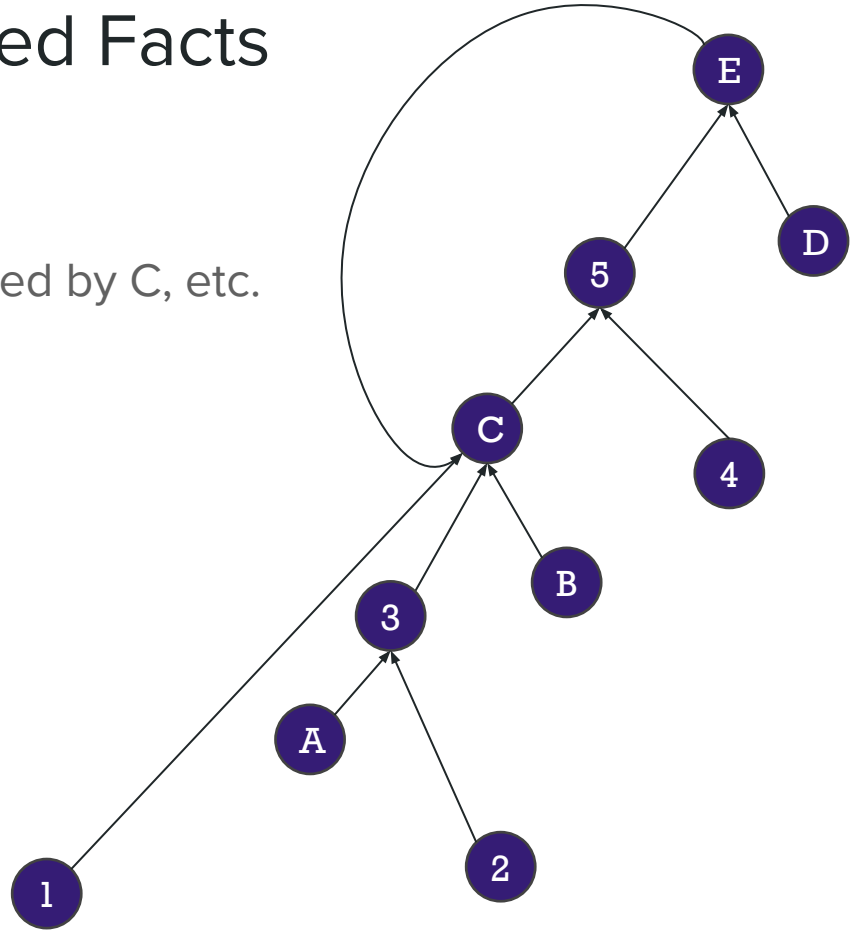
Retract E: (wags-tail Hershey)

But E is supported by 5, which is supported by C, etc.

Support indicates E should be True

Retracting it creates an **inconsistency**

Retract only assertions (leafs)



Knowledge

Search

Chapters 3 & 4