1 Resolution

Resolution

Algorithm Overview

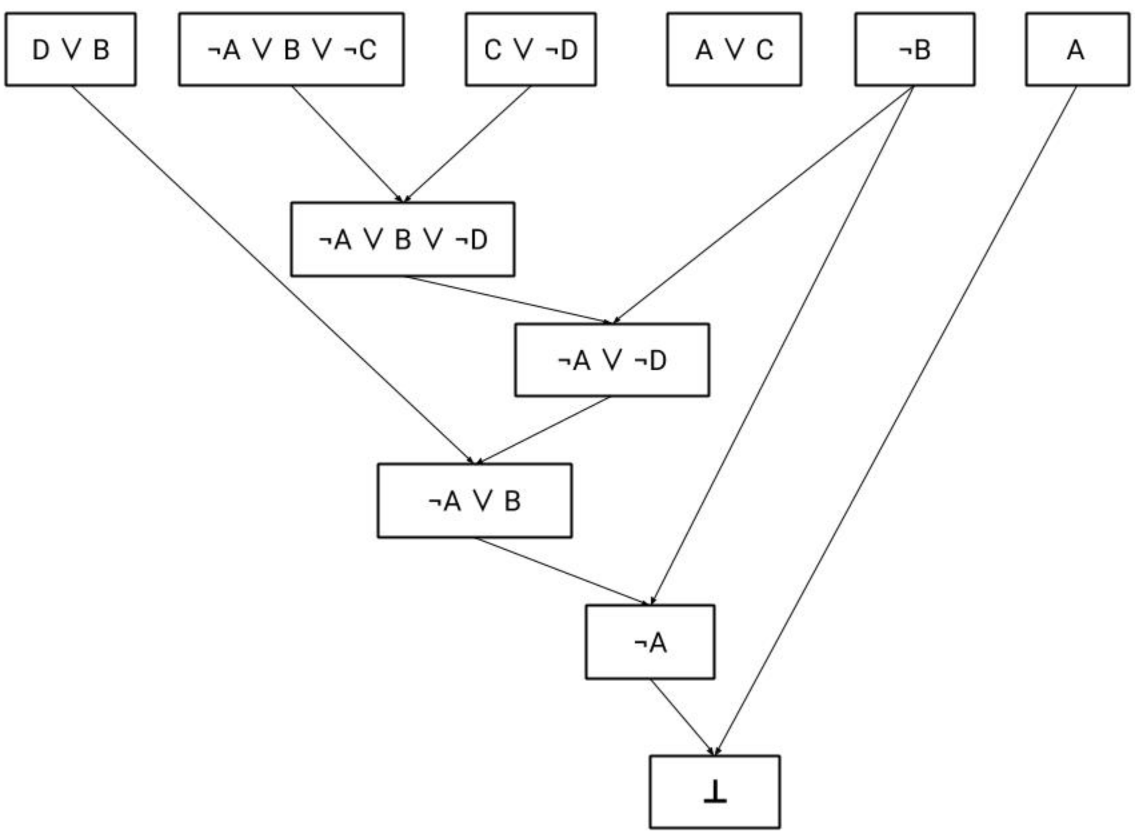
function PL-RESOLUTION?(KB, α) returns true or false

We want to prove that KB entails α

In other words, we want to prove that we cannot satisfy (KB and **not** α)

1. Start with a set of CNF clauses, including the KB as well as $\neg\alpha$
2. Keep resolving pairs of clauses until
   - A. You resolve the empty clause
     - Contradiction found!
     - KB $\wedge \neg\alpha$ cannot be satisfied
     - Return true, KB entails α
   - B. No new clauses added
     - Return false, KB does not entail α

From the knowledge base below, show ¬A must be true.



The resolution algorithm involves iteratively applying the general resolution rule to derive new clauses which will (hopefully) lead to a contradiction.
The general resolution rule has the following format:

$$\frac{a_1 \vee \cdots \vee a_m \vee b \qquad \neg b \vee c_1 \vee \ldots c_n}{a_1 \vee \cdots \vee a_m \vee c_1 \vee \ldots c_n}$$

We can show that this works by considering the unit clause case - resolving $(a \vee b) \wedge (\neg b \vee c)$. By the law of excluded middle, exactly one of the following cases must hold:
Case 1: b is true
In this case, c must be true for the clause $(\neg b \vee c)$ to evaluate to true.

Case 2: b is false
In this case, a must be true for the clause $(a \vee b)$ to evaluate to true.

Thus, a or c must be true for our original conjunction to be true.
We can inductively use this logic to show that the general resolution rule applies for arbitrary $m, n \in \mathbb{N}$.

A couple things to note:

- Applying the resolution rule isn't "reducing" or removing anything from our current knowledge base - rather, we're generating more clauses and adding them to our KB.
- It's not necessary to resolve all possible pairs of clauses to reach a contradiction. As we can see above, we only needed to derive $\neg P_{1,2}$ to reach a contradiction.
- We can only eliminate one pair of literals at a time. To demonstrate this, consider the following pair of clauses: $(A \vee P \vee Q) \wedge (B \vee \neg P \vee \neg Q)$. We can only resolve this pair to get $(A \vee P \vee B \vee \neg P)$ and $(A \vee Q \vee B \vee \neg Q)$ separately (which both end up evaluating to True).
  What we cannot do is derive $(A \vee B)$. Consider the model $A =$ False, $B =$ False, $P =$ True, $Q =$ False.

2 Forward chaining

In this section, we will be proving a statement using forward chaining.

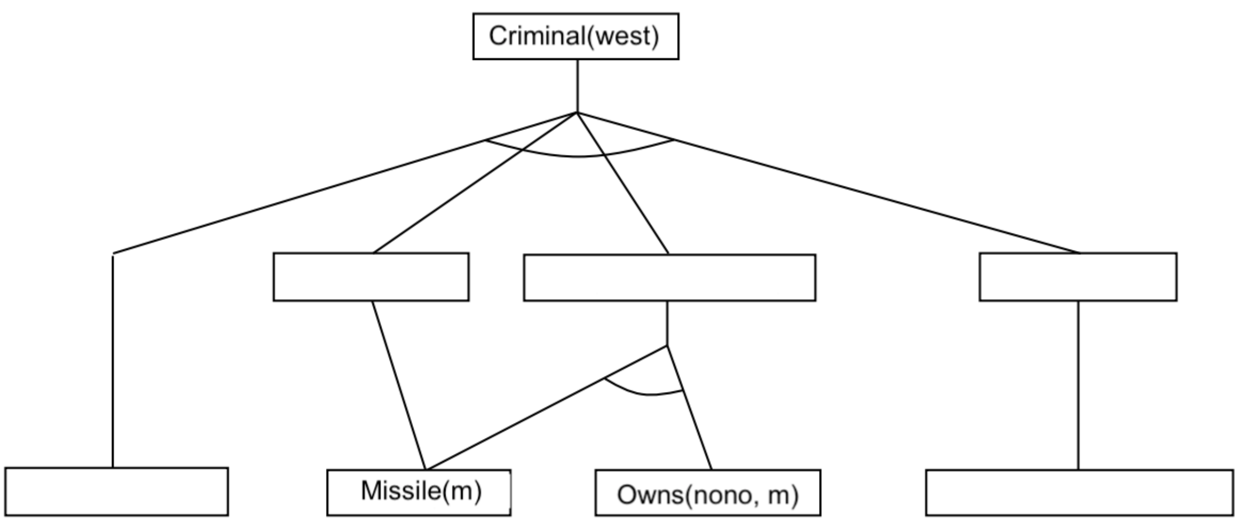
There is currently a war going on and the United States is desperate to round up all the criminals. We want to determine whether Colonel West is a criminal. Let's start with what we know.

We know that it is a crime for an American to sell weapons to hostile nations. The country Nono is an enemy of America. Furthermore, we know that Nono has some missiles, all of which were sold to it by Colonel West, who is American.

(a) Represent your knowledge base using first order logic. You can use the following function predicates: American(x), Criminal(x), Hostile(x), Missile(x), Weapon(x), Enemy(x,y), Owns(x,y), Sells(x,y,z).

1. ____________ $\wedge$ ____________ $\wedge$ ____________ $\wedge$ ____________ $\Rightarrow$ Criminal(x)
2. Missile(x) $\Rightarrow$ ____________
3. Missile(m)
4. Owns(nono, m)
5. Missile(x) $\wedge$ ____________ $\Rightarrow$ Sells(west, x, nono)
6. Enemy(x, america) $\Rightarrow$ ____________
7. ____________
8. ____________

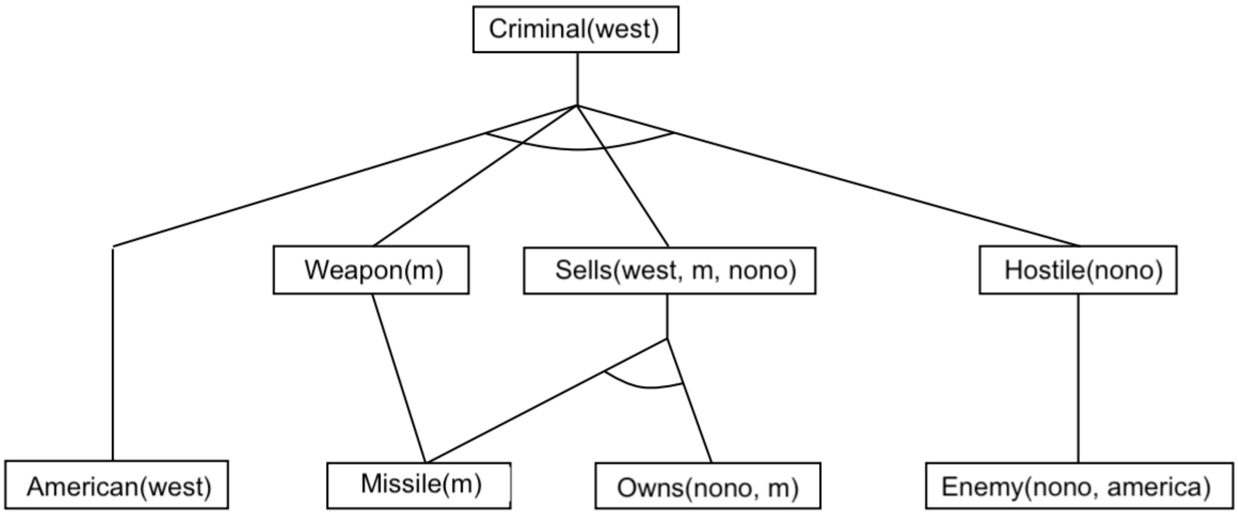
(b) Fill in the blanks below using your knowledge base to prove that Colonel West is a criminal.



(a) Represent your knowledge base using first order logic.

1. American(x) $\wedge$ Weapon(y) $\wedge$ Sells(x,y,z) $\wedge$ Hostile(z) $\Rightarrow$ Criminal(x)
2. Missile(x) $\Rightarrow$ Weapon(x)
3. Missile(m)
4. Owns(nono, m)
5. Owns(nono, x) $\wedge$ Missile(x) $\Rightarrow$ Sells(west, x, nono)
6. Enemy(x, America) $\Rightarrow$ Hostile(x)
7. American(west)
8. Enemy(nono, america)

(b) Fill in the blanks below using your knowledge base to prove that Colonel West is a criminal.


Criminal(west)
Weapon(m)
Sells(west, m, nono)
Hostile(nono)
American(west)
Missile(m)
Owns(nono, m)
Enemy(nono, america)

3 First-Order Logic

(a) Write in first-order logic the assertion that every key will eventually be lost forever, using only the following vocabulary:

- $Key(x), x$ is a key
- $Sock(x), x$ is a sock
- $Pair(x, y), x$ and y are a pair
- Now is the current time
- $Before(t_1, t_2)$ represents that time t_1 comes before t_2
- $Lost(x, t)$ represents that object x is lost at time t

$\forall k\ Key(k) \Rightarrow [\exists t_0\ Before(Now, t_0) \wedge \forall t\ Before(t_0, t) \Rightarrow Lost(k, t)]$

(b) Write in first-order logic the assertion that at least one of every pair of socks will eventually be lost forever.

$\forall s_1, s_2\ Sock(s_1) \wedge Sock(s_2) \wedge Pair(s_1, s_2) \Rightarrow$
$[\exists t_1\ Before(Now, t_1) \wedge \forall t\ Before(t_1, t) \Rightarrow Lost(s_1, t)] \vee$
$[\exists t_2\ Before(Now, t_2) \wedge \forall t\ Before(t_2, t) \Rightarrow Lost(s_2, t)]$

Notice that the disjunction allows for both socks in the pair to be lost, as the English sentence implies.

(c) Write out the vocabulary you would use to represent the following sentence in first-order logic.

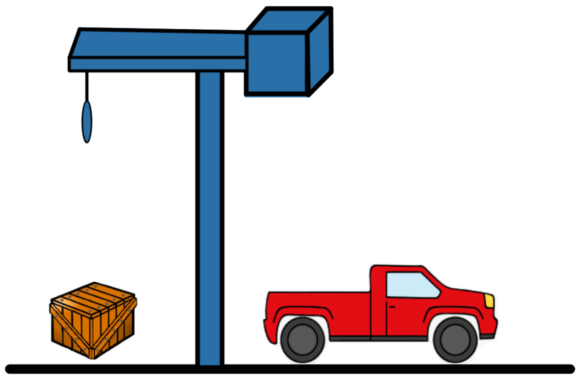
- Everyone who takes 15-281 loves Pacman.

  Student(x), x is a student
  In281(x), x is in 15-281
  Loves(x, y), x loves y
  Pacman(x), x is a Pacman

4 Symbolic Planning - Crate Problem

In the Crane problem, you are given a crane, a package and a truck. The package starts on the left, the truck on the right, and the crane faces the left. The goal of this is to load the package onto the truck and have the crane be facing the left.

The crane can swing between left and right, with or without a payload, and it can pick up the crate if it is on the same side. The crate can only be loaded onto the truck using the crane.

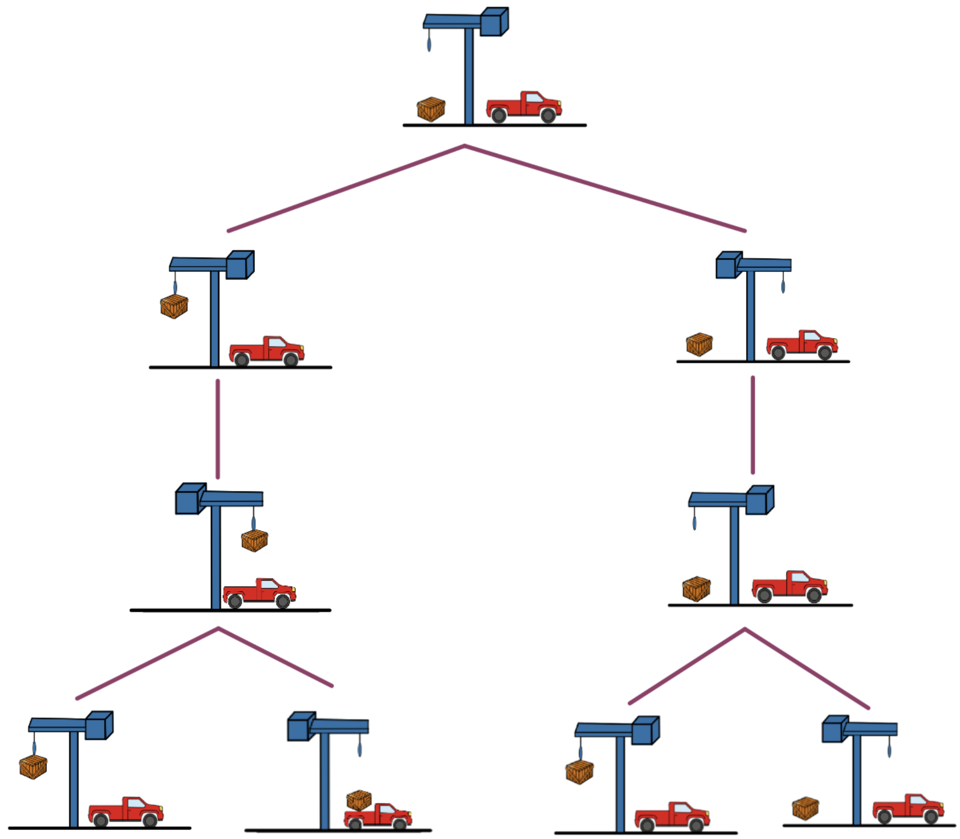


(a) Conceptual Question: What is the difference between linear and non-linear planning? When are they the same?

Linear planning: We keep a stack of unnachieved goals and solve each one, one at a time, adding back goals that we violate along the way.
Non-Linear planning: Maintain a set of unnachieved goals and search all interleavings of these goals adding a goal back to the set if a later change makes it violated.

Linear planning and non-linear planning are equivalent when there is one goal because there is only one possible "interleaving" of the goals so linear and non-linear planning will have the same approach.

(b) Draw the planning graph for the first 3 moves. You may use pictures instead of propositions.

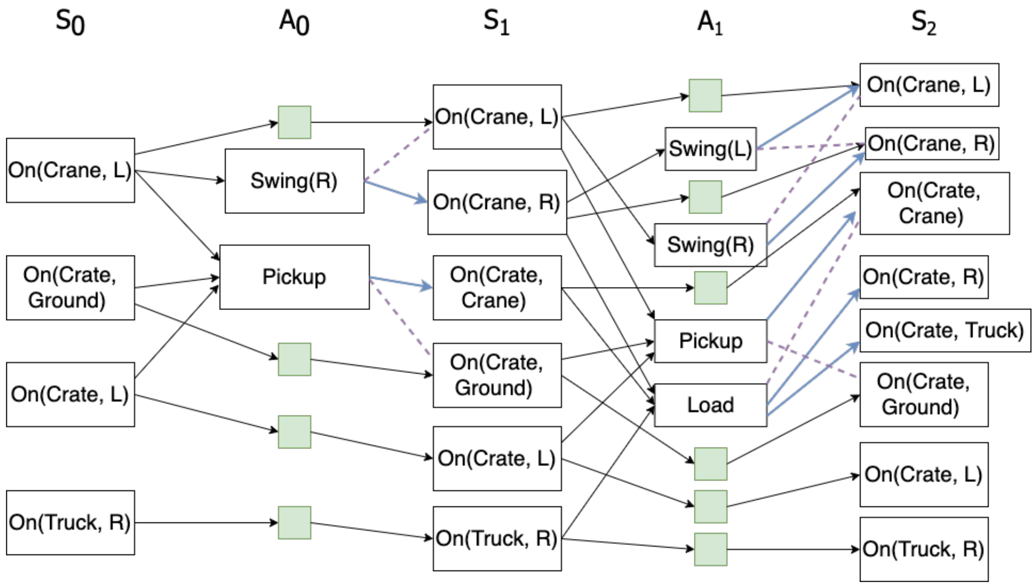


(c) Formulate the crate problem as a symbolic plan.

See provided sample code

(d) Draw the first two levels of the Graph Plan graph.

In the following diagram, the blue lines represent the propositions added as the result of an action and the dotted purple lines represent the propositions deleted at the result of that action. The green squares in the action levels represent no-op's.



(e) Identify the exclusive actions in your graph and determine which type of mutex each is.

In the level A_0, Swing(R) and Pickup interfere with each other. In level A_1, one example would be Swing(L) and Swing(R) being inconsistent.