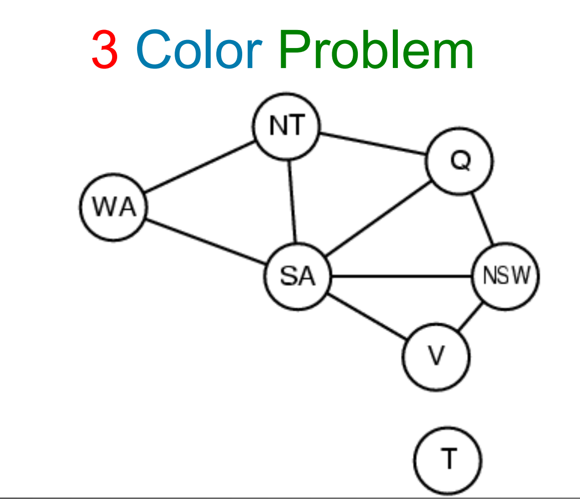
CSCI561 SPRING 2015 WEEK 9 DISCUSSION

Prof Tejada stejada@usc.edu

EFFICIENCY OF FORWARD CHAINING

- Is quite inefficient if you check every rule against the known facts on every cycle of execution
  - Assume N rules, each with M conditions; plus K facts
  - Matching against KB can require time complexity $N*K^M$

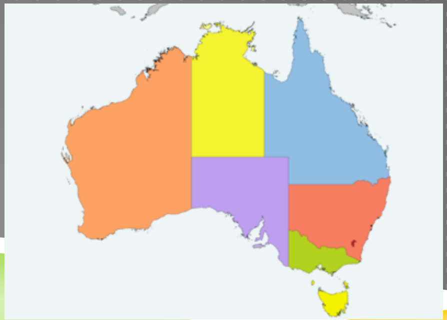


Diff(*wa*, *nt*) ∧ *Diff*(*wa*, *sa*) ∧ *Diff*(*nt*, *q*) ∧ *Diff*(*nt*, *sa*) ∧ *Diff*(*q*, *nsw*) ∧ *Diff*(*q*, *sa*) ∧ *Diff*(*nsw*, *v*) ∧ *Diff*(*nsw*, *sa*) ∧ *Diff*(*v*, *sa*) ⇒ *Colorable*()

Diff(*Red*, *Blue*) *Diff*(*Blue*, *Red*)
Diff (*Red*, *Green*) *Diff*(*Green* ,*Red*)
Diff(*Blue*, *Green*) *Diff*(*Green*, *Blue*)

1 rule, 9 conditions, 6 facts

Time = $1*6^9 \approx 10M$



EXAMPLE 9.4

- For each pair of atomic sentences, give the most general unifier if it exists:

a) P(A,B,B), P(x,y,z).

- {x/A, y/B, z/B} (or some permutation of this).
- b) Q(y,G(A,B)),Q(G(x,x),y).
- No unifier (x cannot bind to both A and B).
- c) Older(Father(y),y),Older(Father(x),John).
- {y/John,x/John}.
- d) Knows(Father(y),y), Knows(x,x).
  - No unifier (prevents unification of y with Father(y)).

EXAMPLE 9.6

- Write down logical representations for the following sentences, suitable for use with Generalized Modus Ponens:

a) Horses, cows, and pigs are mammals.

- $Horse(x) \Rightarrow Mammal(x)$
- $Cow(x) \Rightarrow Mammal(x)$
- $Pig(x) \Rightarrow Mammal(x)$
- b) An offspring of a horse is a horse.
- $Offspring(x,y) \land Horse(y) \Rightarrow Horse(x)$
- c) Bluebeard is a horse.
- $Horse(Bluebeard)$

EXAMPLE 9.6

- Write down logical representations for the following sentences, suitable for use with Generalized Modus Ponens:
- d) Bluebeard is Charlie's parent.
- Parent(Bluebeard,Charlie)
- e) Offspring and parent are inverse relations.
- Offspring(x,y) ⇒ Parent(y,x)
  Parent(x,y) ⇒ Offspring(y,x)
- f) Every mammal has a parent.
- Mammal(x) ⇒ Parent(G(x), x) (here G is a Skolem function).

EXAMPLE 9.7

- These questions concern issues with substitution and Skolemization.

a) Given the premise $\forall x\ \exists y\ P(x,y)$, it is not valid to conclude that $\exists q\ P(q,q)$. Give an example of a predicate P where the first is true but the second is false.

- Let P(x,y) be the relation "x is less than y" over the integers. Then $\forall x\ \exists y\ P(x,y)$ is true but $\exists x\ P(x,x)$ is false.

EXAMPLE 9.7

- These questions concern issues with substitution and Skolemization.
- b) Suppose that an inference engine is incorrectly written with the occurs-check omitted, so that it allows a literal like $P(x,F(x))$ to be unified with $P(q,q)$.
- Show that such an inference engine will allow the conclusion $\exists q\ P(q,q)$ to be inferred from the premise $\forall x\ \exists y\ P(x,y)$.
- Converting the premise $P(x, Sk0(x))$
- Converting the negated goal $\neg P(q,q)$.
- If the two formulas can be unified $\{q/Sk0(x), x/Sk0(x)\}$
- Then these resolve to the null clause.

EXAMPLE 9.7

- These questions concern concern issues with substitution and Skolemization.
- c) Suppose that a procedure that converts first-order logic to clausal form incorrectly Skolemizes $\forall x\ \exists y\ P(x,y)$ to $P(x,Sk0)$
  - replaces y by a Skolem constant rather than a Skolem function of x.
- Show that an inference engine that uses such a procedure will likewise allow $\exists q\ P(q,q)$ to be inferred from the premise $\forall x\ \exists y\ P(x,y)$.
- If the premise is represented as $P(x,Sk0)$ and
- The negated goal is $\neg P(q, q)$
- Then can be resolved to the null clause under substitution $\{q/Sk0, x/Sk0\}$.

EXAMPLE 9.7

- These questions concern concern issues with substitution and Skolemization.
- d) A common error among students is to suppose that, in unification, one is allowed to substitute a term for a Skolem constant instead of for a variable. For instance, they will say that the formulas P (Sk1) and P (A) can be unified under the substitution {Sk1/A}.
- Give an example where this leads to an invalid inference.
  - Suppose you are given the premise $\exists$ x Cat(x) and you wish to prove Cat(Socrates).
  - Converting the premise to Cat(Sk1).
  - If this unifies with Cat(Socrates) with substitution {Sk1/Socrates}
  - Then you can resolve this with the negated goal ¬Cat(Socrates) to give the null clause.

EXAMPLE 9.13

- In this exercise, use the sentences you wrote in Exercise 9.6 to answer a question by using a backward-chaining algorithm.
- $Horse(x) \Rightarrow Mammal(x)$
- $Cow(x) \Rightarrow Mammal(x)$
- $Pig(x) \Rightarrow Mammal(x)$
- $Offspring(x,y) \land Horse(y) \Rightarrow Horse(x)$
- $Horse(Bluebeard)$
- $Parent(Bluebeard, Charlie)$
- $Offspring(x,y) \Rightarrow Parent(y,x)$

$Parent(x,y) \Rightarrow Offspring(y,x)$

- $Mammal(x) \Rightarrow Parent(G(x), x)$

EXAMPLE 9.13

- Draw the proof tree generated by an exhaustive backward-chaining algorithm for the query ∃h Horse(h), where clauses are matched in the order given.

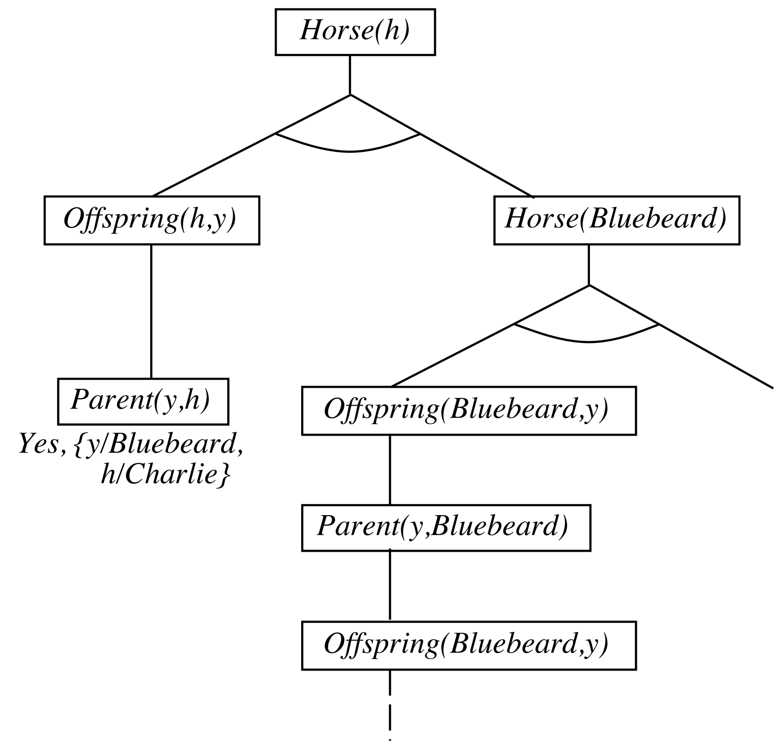


Figure S9.1 Partial proof tree for finding horses.

EXAMPLE 9.13

- What do you notice about this domain?
- We get an infinite loop because of rule b, $Offspring(x, y) \land Horse(y) \Rightarrow Horse(x)$.
- The specific loop appearing in the figure arises because of the ordering of the clauses— it would be better to order Horse(Bluebeard) before the rule from b.
- However, a loop will occur no matter which way the rules are ordered if the theorem-prover is asked for all solutions.

EXAMPLE 9.13

- How many solutions for h actually follow from your sentences?
- One should be able to prove that both Bluebeard and Charlie are horses.

- Can you think of a way to find all of them?
- Whenever a "looping" goal occurs, suspend the attempt to prove that subgoal.
- Continue with all other branches of the proof for the supergoal, gathering up the solutions. Then use those solutions (suitably instantiated if necessary) as solutions for the suspended subgoal, continuing that branch of the proof to find additional solutions if any.
- E.g. Offspring(Bluebeard,y) is a repeated goal and would be suspended.
- Since no other way to prove it exists, that branch will terminate with failure.