

# For Friday

- None
- Program 1 due

# Program 1

- Any questions?

# Deducing Hidden Properties

breezy(Loc) :- at(agent, Loc, S), breeze(S).

Smelly(Loc) :- at(agent, Loc, S), Stench(S).

- Causal Rules

smelly(Loc2) :-

at(wumpus, Loc1, S), adjacent(Loc1, Loc2).

breezy(Loc2) :-

at(pit, Loc1, S), adjacent(Loc1, Loc2).

- Diagnostic Rules

ok(Loc2) :-

percept([none, none, G, U, C], T),

at(agent, Loc1, S), adjacent(Loc1, Loc2).

# Preferences Among Actions

- We need some way to decide between the possible actions.
- We would like to do this apart from the rules that determine what actions are possible.
- We want the desirability of actions to be based on our goals.

# Handling Goals

- Original goal is to find and grab the gold
- Once the gold is held, we want to find the starting square and climb out
- We have three primary methods for finding a path out
  - Inference (may be very expensive)
  - Search (need to translate problem)
  - Planning (which we'll discuss later)

# Wumpus World in Practice

- Not going to use situation calculus
- Instead, just maintain the current state of the world
- Advantages?
- Disadvantages?

# Inference in FOPC

- As with propositional logic, we want to be able to draw logically sound conclusions from our KB
- Soundness:
  - If we can infer  $A$  from  $B$ ,  $B$  entails  $A$ .
  - If  $B \vdash A$ , then  $B \models A$
- Complete
  - If  $B$  entails  $A$ , then we can infer  $A$  from  $B$
  - If  $B \models A$ , then  $B \vdash A$

# Inference Methods

- Three styles of inference:
  - Forward chaining
  - Backward chaining
  - Resolution refutation
- Forward and backward chaining are sound and can be reasonably efficient but are incomplete
- Resolution is sound and complete for FOPC, but can be very inefficient

# Inference Rules for Quantifiers

- The inference rules for propositional logic also work for first order logic
- However, we need some new rules to deal with quantifiers
- Let  $\text{SUBST}(q, a)$  denote the result of applying a substitution or binding list  $q$  to the sentence  $a$ .

$$\text{SUBST}(\{x/\text{Tom}, y/\text{Fred}\}, \text{Uncle}(x,y)) = \text{Uncle}(\text{Tom}, \text{Fred})$$

# Universal Elimination

- Formula:

$$\forall v a \vdash \text{SUBST}(\{v/g\}, a)$$

- Constraints:

– for any sentence,  $a$ , variable,  $v$ , and ground term,  $g$

- Example:

$$\forall x \text{ Loves}(x, \text{FOPC}) \vdash \text{Loves}(\text{Califf}, \text{FOPC})$$

# Existential Elimination

- Formula:  
 $\$v a \vdash \text{SUBST}(\{v/k\}, a)$
- Constraints:
  - for any sentence,  $a$ , variable,  $v$ , and constant symbol,  $k$ , that **doesn't** occur elsewhere in the KB (**Skolem constant**)
- Example:  
 $\exists x (\text{Owns}(\text{Mary}, x) \wedge \text{Cat}(x)) \vdash$   
 $\text{Owns}(\text{Mary}, \text{MarysCat}) \wedge \text{Cat}(\text{MarysCat})$

# Existential Introduction

- Formula:

$a \vdash \exists v \text{ SUBST}(\{g/v\}, a)$

- Constraints:

– for any sentence,  $a$ , variable,  $v$ , that does not occur in  $a$ , and ground term,  $g$ , that does occur in  $a$

- Example:

**$\text{Loves}(\text{Califf}, \text{FOPC}) \vdash \exists x \text{ Loves}(x, \text{FOPC})$**

# Sample Proof

1)  $\forall x,y(\text{Parent}(x, y) \wedge \text{Male}(x) \Rightarrow \text{Father}(x,y))$

2)  $\text{Parent}(\text{Tom}, \text{John})$

3)  $\text{Male}(\text{Tom})$

Using Universal Elimination from 1)

4)  $\forall y(\text{Parent}(\text{Tom}, y) \wedge \text{Male}(\text{Tom}) \Rightarrow \text{Father}(\text{Tom}, y))$

Using Universal Elimination from 4)

5)  $\text{Parent}(\text{Tom}, \text{John}) \wedge \text{Male}(\text{Tom}) \Rightarrow \text{Father}(\text{Tom}, \text{John})$

Using And Introduction from 2) and 3)

6)  $\text{Parent}(\text{Tom}, \text{John}) \wedge \text{Male}(\text{Tom})$

Using Modes Ponens from 5) and 6)

7)  $\text{Father}(\text{Tom}, \text{John})$

# Generalized Modus Ponens

- Combines three steps of “natural deduction” (Universal Elimination, And Introduction, Modus Ponens) into one.
- Provides direction and simplification to the proof process for standard inferences.
- Generalized Modus Ponens:

$p_1', p_2', \dots, p_n', (p_1 \wedge p_2 \wedge \dots \wedge p_n \Rightarrow q) \vdash \text{SUBST}(\theta, q)$

where  $\theta$  is a substitution such that for all  $i$

$\text{SUBST}(\theta, p_i') = \text{SUBST}(\theta, p_i)$

# Example

1)  $\forall x,y(\text{Parent}(x,y) \wedge \text{Male}(x) \Rightarrow \text{Father}(x,y))$

2)  $\text{Parent}(\text{Tom},\text{John})$

3)  $\text{Male}(\text{Tom})$

$\theta = \{x/\text{Tom}, y/\text{John}\}$

4)  $\text{Father}(\text{Tom},\text{John})$

# Canonical Form

- In order to use generalized Modus Ponens, all sentences in the KB must be in the form of Horn sentences:

$$\forall v_1, v_2, \dots, v_n p_1 \wedge p_2 \wedge \dots \wedge p_m \Rightarrow q$$

- Also called Horn clauses, where a clause is a disjunction of literals, because they can be rewritten as disjunctions with at most one non-negated literal.

$$\forall v_1, v_2, \dots, v_n \neg p_1 \vee \neg p_2 \vee \dots \vee \neg p_n \vee q$$

# Horn Clauses

- Single positive literals (facts) are Horn clauses with no antecedent.
- Quantifiers can be dropped since all variables can be assumed to be universally quantified by default.
- Many statements can be transformed into Horn clauses, but many cannot (e.g.  $P(x) \vee Q(x)$ ,  $\neg P(x)$ )

# Unification

- In order to match antecedents to existing literals in the KB, we need a pattern matching routine.
- $\text{UNIFY}(p,q)$  takes two atomic sentences and returns a substitution that makes them equivalent.
- $\text{UNIFY}(p,q)=\theta$  where  $\text{SUBST}(\theta,p)=\text{SUBST}(\theta,q)$
- $\theta$  is called a unifier

# Unification Examples

$\text{UNIFY}(\text{Parent}(x,y), \text{Parent}(\text{Tom}, \text{John})) = \{x/\text{Tom}, y/\text{John}\}$

$\text{UNIFY}(\text{Parent}(\text{Tom},x), \text{Parent}(\text{Tom}, \text{John})) = \{x/\text{John}\}$

$\text{UNIFY}(\text{Likes}(x,y), \text{Likes}(z,\text{FOPC})) = \{x/z, y/\text{FOPC}\}$

$\text{UNIFY}(\text{Likes}(\text{Tom},y), \text{Likes}(z,\text{FOPC})) = \{z/\text{Tom}, y/\text{FOPC}\}$

$\text{UNIFY}(\text{Likes}(\text{Tom},y), \text{Likes}(y,\text{FOPC})) = \text{fail}$

$\text{UNIFY}(\text{Likes}(\text{Tom},\text{Tom}), \text{Likes}(x,x)) = \{x/\text{Tom}\}$

$\text{UNIFY}(\text{Likes}(\text{Tom},\text{Fred}), \text{Likes}(x,x)) = \text{fail}$

# Same Variable

- Exact variable names used in sentences in the KB **should** not matter.
- But if **Likes(x,FOPC)** is a formula in the KB, it does not unify with **Likes(John,x)** but does unify with **Likes(John,y)**
- We can standardize one of the arguments to UNIFY to make its variables unique by renaming them.

Likes(x,FOPC)  $\rightarrow$  Likes( $x_1$  , FOPC)

UNIFY(Likes(John,x),Likes( $x_1$  ,FOPC)) = { $x_1$  /John, x/FOPC }

# Which Unifier?

- There are many possible unifiers for some atomic sentences.
  - $\text{UNIFY}(\text{Likes}(x,y),\text{Likes}(z,\text{FOPC})) =$ 
    - $\{x/z, y/\text{FOPC}\}$
    - $\{x/\text{John}, z/\text{John}, y/\text{FOPC}\}$
    - $\{x/\text{Fred}, z/\text{Fred}, y/\text{FOPC}\}$
    - .....
- UNIFY should return the most general unifier which makes the least commitment to variable values.

# How Do We Use It?

- We have two primary methods for using Generalized Modus Ponens
- We can start with the knowledge base and try to generate new sentences
  - Forward Chaining
- We can start with a sentence we want to prove and try to work backward until we can establish the facts from the knowledge base
  - Backward Chaining

# Forward Chaining

- Use modus ponens to derive **all** consequences from new information.
- Inferences cascade to draw deeper and deeper conclusions
- To avoid looping and duplicated effort, must prevent addition of a sentence to the KB which is the same as one already present.
- Must determine all ways in which a rule (Horn clause) can match existing facts to draw new conclusions.

# Assumptions

- A sentence is a renaming of another if it is the same except for a renaming of the variables.
- The composition of two substitutions combines the variable bindings of both such that:

$$\text{SUBST}(\text{COMPOSE}(\theta_1, \theta_2), p) = \text{SUBST}(\theta_2, \text{SUBST}(\theta_1, p))$$

# Forward Chaining Algorithm

procedure FORWARD-CHAIN( $KB, p$ )

if there is a sentence in  $KB$  that is a renaming of  $p$  then return

Add  $p$  to  $KB$

for each  $(p_1 \wedge \dots \wedge p_n \Rightarrow q)$  in  $KB$  such that for some  $i$ ,

UNIFY( $p_i, p$ ) =  $\theta$  succeeds do

FIND-AND-INFER( $KB, [p_1, \dots, p_{i-1}, p_{i+1}, \dots, p_n], q, \theta$ )

end

procedure FIND-AND-INFER( $KB, premises, conclusion, \theta$ )

if  $premises = []$  then

FORWARD-CHAIN( $KB, SUBST(\theta, conclusion)$ )

else for each  $p'$  in  $KB$  such that

UNIFY( $p', SUBST(\theta, FIRST(premises))$ ) =  $\theta_2$  do

FIND-AND-INFER( $KB, REST(premises), conclusion, COMPOSE(\theta, \theta_2)$ )

end

# Forward Chaining Example

Assume in KB

1)  $\text{Parent}(x,y) \wedge \text{Male}(x) \Rightarrow \text{Father}(x,y)$

2)  $\text{Father}(x,y) \wedge \text{Father}(x,z) \Rightarrow \text{Sibling}(y,z)$

Add to KB

3)  $\text{Parent}(\text{Tom}, \text{John})$

Rule 1) tried but can't "fire"

Add to KB

4)  $\text{Male}(\text{Tom})$

Rule 1) now satisfied and triggered and adds:

5)  $\text{Father}(\text{Tom}, \text{John})$

Rule 2) now triggered and adds:

6)  $\text{Sibling}(\text{John}, \text{John}) \{ x/\text{Tom}, y/\text{John}, z/\text{John} \}$

# Example cont.

Add to KB

7) Parent(Tom,Fred)

Rule 1) triggered again and adds:

8) Father(Tom,Fred)

Rule 2) triggered again and adds:

9) Sibling(Fred,Fred) { x/Tom, y/Fred, z/Fred }

Rule 2) triggered again and adds:

10) Sibling(John, Fred) { x/Tom, y/John, z/Fred }

Rule 2) triggered again and adds:

11) Sibling(Fred, John) { x/Tom, y/Fred, z/John }

# Problems with Forward Chaining

- Inference can explode forward and may never terminate.
- Consider the following:
  - $\text{Even}(x) \Rightarrow \text{Even}(\text{plus}(x,2))$
  - $\text{Integer}(x) \Rightarrow \text{Even}(\text{times}(2,x))$
  - $\text{Even}(x) \Rightarrow \text{Integer}(x)$
  - $\text{Even}(2)$
- Inference is not directed towards any particular conclusion or goal. May draw lots of irrelevant conclusions

# Backward Chaining

- Start from query or atomic sentence to be proven and look for ways to prove it.
- Query can contain variables which are assumed to be existentially quantified.

Sibling(x,John) ?

Father(x,y) ?

- Inference process should return all sets of variable bindings that satisfy the query.

# Method

- First try to answer query by unifying it to all possible facts in the KB.
- Next try to prove it using a rule whose consequent unifies with the query and then try to recursively prove all of its antecedents.
- Given a conjunction of queries, first get all possible answers to the first conjunct and then for each resulting substitution try to prove all of the remaining conjuncts.
- Assume variables in rules are renamed (standardized apart) before each use of a rule.

# Backchaining Examples

KB:

- 1)  $\text{Parent}(x,y) \wedge \text{Male}(x) \Rightarrow \text{Father}(x,y)$
- 2)  $\text{Father}(x,y) \wedge \text{Father}(x,z) \Rightarrow \text{Sibling}(y,z)$
- 3)  $\text{Parent}(\text{Tom},\text{John})$
- 4)  $\text{Male}(\text{Tom})$
- 7)  $\text{Parent}(\text{Tom},\text{Fred})$

Query:  $\text{Parent}(\text{Tom},x)$

Answers: (  $\{x/\text{John}\}, \{x/\text{Fred}\}$  )

Query:  $\text{Father}(\text{Tom}, s)$

Subgoal:  $\text{Parent}(\text{Tom}, s) \wedge \text{Male}(\text{Tom})$

$\{s/\text{John}\}$

Subgoal:  $\text{Male}(\text{Tom})$

Answer:  $\{s/\text{John}\}$

$\{s/\text{Fred}\}$

Subgoal:  $\text{Male}(\text{Tom})$

Answer:  $\{s/\text{Fred}\}$

Answers:  $(\{s/\text{John}\}, \{s/\text{Fred}\})$

Query:  $\text{Father}(f,s)$

Subgoal:  $\text{Parent}(f,s) \wedge \text{Male}(f)$

$\{f/\text{Tom}, s/\text{John}\}$

Subgoal:  $\text{Male}(\text{Tom})$

Answer:  $\{f/\text{Tom}, s/\text{John}\}$

$\{f/\text{Tom}, s/\text{Fred}\}$

Subgoal:  $\text{Male}(\text{Tom})$

Answer:  $\{f/\text{Tom}, s/\text{Fred}\}$

Answers:  $(\{f/\text{Tom}, s/\text{John}\}, \{f/\text{Tom}, s/\text{Fred}\})$

Query: Sibling(a,b)

Subgoal: Father(f,a)  $\wedge$  Father(f,b)

{f/Tom, a/John}

Subgoal: Father(Tom,b)

{b/John}

Answer: {f/Tom, a/John, b/John}

{b/Fred}

Answer: {f/Tom, a/John, b/Fred}

{f/Tom, a/Fred}

Subgoal: Father(Tom,b)

{b/John}

Answer: {f/Tom, a/Fred, b/John}

{b/Fred}

Answer: {f/Tom, a/Fred, b/Fred}

Answers: ({f/Tom, a/John, b/John}, {f/Tom, a/John, b/Fred})

{f/Tom, a/Fred, b/John}, {f/Tom, a/Fred, b/Fred})

# Incompleteness

- Rule-based inference is not complete, but is reasonably efficient and useful in many circumstances.
- Still can be exponential or not terminate in worst case.

- Incompleteness example:

$$P(x) \Rightarrow Q(x)$$

$$\neg P(x) \Rightarrow R(x) \text{ (not Horn)}$$

$$Q(x) \Rightarrow S(x)$$

$$R(x) \Rightarrow S(x)$$

- Entails  $S(A)$  for any constant  $A$  but is not inferable from modus ponens

# Completeness

- In 1930 Gödel showed that a complete inference procedure for FOPC existed, but did not demonstrate one (non-constructive proof).
- In 1965, Robinson showed a resolution inference procedure that was sound and complete for FOPC.
- However, the procedure may not halt if asked to prove a theorem that is not true, it is said to be semidecidable (a type of undecidability).
- If a conclusion  $C$  is entailed by the KB then the procedure will eventually terminate with a proof. However if it is not entailed, it may never halt.
- It does not follow that either  $C$  or  $\neg C$  is entailed by a KB (may be independent). Therefore trying to prove both a conjecture and its negation does not help.
- Inconsistency of a KB is also semidecidable.