

Computing logical consequences

- •We want procedures for computing logical consequences that can be implemented in our programs.
- This would allow us to reason with our knowledge Represent the knowledge as logical formulas
- Apply procedures for generating logical consequences
- These procedures are called proof procedures.

Proof Procedures

- •Interesting, proof procedures work by simply manipulating formulas. They do not know or care anything about interpretations.
- •Nevertheless they respect the semantics of interpretations!
- •We will develop a proof procedure for firstorder logic called resolution.
 - Resolution is the mechanism used by PROLOG

CSE3402 Winter 2010 Fahiem Bacchus & Yves Les

3

Properties of Proof Procedures

CSE3402 Winter 2010 Fahiem Bacchus & Yves Lesperar

- •Before presenting the details of resolution, we want to look at properties we would like to have in a (any) proof procedure.
- •We write $KB \vdash f$ to indicate that f can be proved from KB (the proof procedure used is implicit).



Soundness

 $\blacksquare \ KB \vdash f \rightarrow \ KB \vDash f$

i.e all conclusions arrived at via the proof procedure are correct: they are logical consequences.

Completeness

 $\blacksquare \ KB \vDash f \rightarrow \ KB \vdash f$

i.e. every logical consequence can be generated by the proof procedure.

• Note proof procedures are computable, but they might have very high complexity in the worst case. So completeness is not necessarily achievable in practice.

CSE3402 Winter 2010 Fahiem Bacchus & Yves Lesp

5

Resolution

•Clausal form.

- Resolution works with formulas expressed in clausal form.
- A literal is an atomic formula or the negation of an atomic formula. dog(fido), ¬cat(fido)
- A clause is a disjunction of literals:
 - \neg owns(fido,fred) $\lor \neg$ dog(fido) \lor person(fred)
 - We write (¬owns(fido,fred), ¬dog(fido), person(fred))

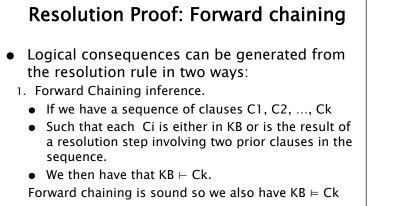
CSE3402 Winter 2010 Fahiem Bacchus & Yves Lespe

■ A clausal theory is a conjunction of clauses.

Resolution **Resolution Rule for Ground Clauses** •The resolution proof procedure consists of only Prolog Programs one simple rule: Prolog programs are clausal theories. From the two clauses ■ However, each clause in a Prolog program is Horn. • (P, Q1, Q2, ..., Qk) ■ A horn clause contains at most one positive literal. ● (¬P, R1, R2, ..., Rn) • The horn clause • We infer the new clause $\neg q1 \lor \neg q2 \lor ... \lor \neg qn \lor p$ is equivalent to • (Q1, Q2, ..., Qk, R1, R2, ..., Rn) $q1 \land q2 \land ... \land qn \Rightarrow p$ ■ Example: and is written as the following rule in Prolog: • (\neg largerThan(clyde,cup), \neg fitsIn(clyde,cup) • (fitsIn(clyde,cup)) p:-q1,q2,...,qn $\Rightarrow \neg$ largerThan(clyde,cup) 7 CSE3402 Winter 2010 Fahiem Bacchus & Yves L CSE3402 Winter 2010 Fahiem Bacchus & Yves Lest

2

6



9

11

Resolution Proof: Refutation proofs

- 2. Refutation proofs.
 - We determine if KB \vdash f by showing that a contradiction can be generated from KB $\Lambda \neg f$.
- In this case a contradiction is an empty clause ().
- We employ resolution to construct a sequence of clauses C1, C2, ..., Cm such that

CSE3402 Winter 2010 Fahiem Bacchus & Yves Lespera

- Ci is in KB $\Lambda \neg f$, or is the result of resolving two previous clauses in the sequence.
- Cm = () i.e. its the empty clause.

Resolution Proof: Refutation proofs

CSE3402 Winter 2010 Fahiem Bacchus & Yves Lesperan

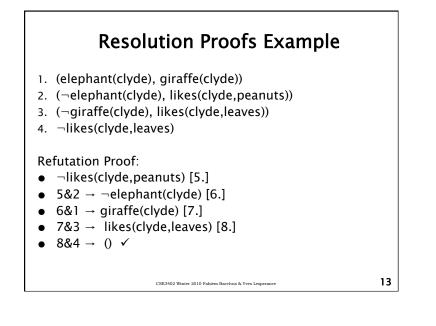
- •If we can find such a sequence C1, C2, ..., Cm=(), we have that
 - KB \vdash f.
 - Furthermore, this procedure is sound so • KB ⊨ f
- •And the procedure is also complete so it is capable of finding a proof of any f that is a logical consequence of KB. I.e.
 - \bullet If KB $\vDash f$ then we can generate a refutation from KB A $\neg f$

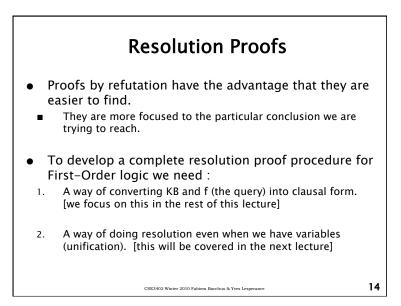
Resolution Proofs Example
Want to prove likes(clyde,peanuts) from:

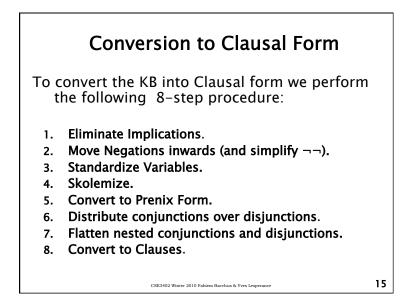
(elephant(clyde), giraffe(clyde))
(-elephant(clyde), likes(clyde,peanuts))
(-giraffe(clyde), likes(clyde,leaves))
(-likes(clyde,leaves)

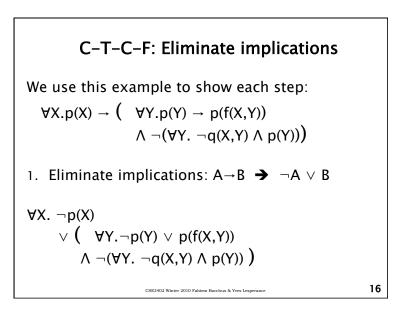
Swad → -giraffe(clyde) [5.]
Swad → -giraffe(clyde) [6.]
Swad → -likes(clyde,peanuts) [7.] ✓

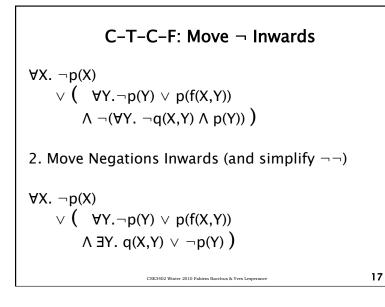
3

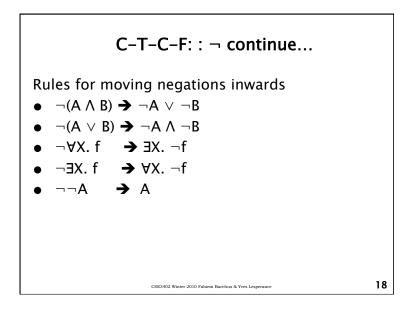




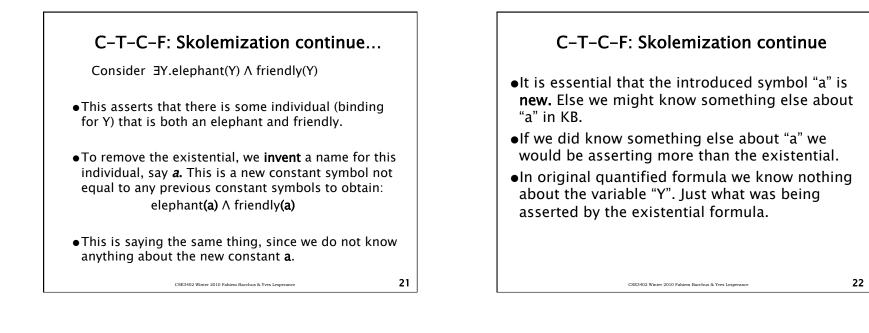








C-T-C-F: Standardize Variables $\forall X. \neg p(X)$ $\lor (\forall Y. \neg p(Y) \lor p(f(X,Y))$ $\land \exists Y.q(X,Y) \lor \neg p(Y))$ 3. Standardize Variables (Rename variables so that each quantified variable is unique) $\forall X. \neg p(X)$ $\lor (\forall Y. (\neg p(Y) \lor p(f(X,Y)))$ $\land \exists Z.q(X,Z) \lor \neg p(Z))$ $\begin{array}{l} \textbf{C-T-C-F: Skolemize} \\ \forall X. \neg p(X) \\ \lor \left(\quad \forall Y. \neg p(Y) \lor p(f(X,Y)) \\ \land \exists Z.q(X,Z) \lor \neg p(Z) \right) \\ \textbf{4. Skolemize (Remove existential quantifiers by introducing <u>new function symbols</u>).} \\ \forall X. \neg p(X) \\ \lor \left(\forall Y. \neg p(Y) \lor p(f(X,Y)) \\ \land q(X,g(X)) \lor \neg p(g(X)) \right) \end{array}$



C-T-C-F: Skolemization continue

Now consider $\forall X \exists Y$. loves(X,Y).

- This formula claims that for every X there is some Y that X loves (perhaps a different Y for each X).
- Replacing the existential by a new constant won't work
 ∀X.loves(X,a).

Because this asserts that there is a **particular** individual "a" loved by every X.

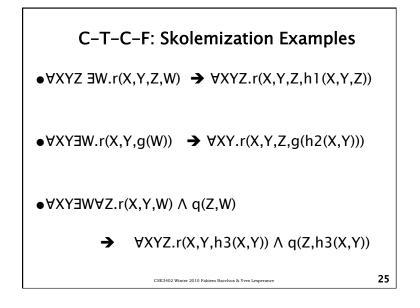
•To properly convert existential quantifiers scoped by universal quantifiers we must use **functions** not just constants.

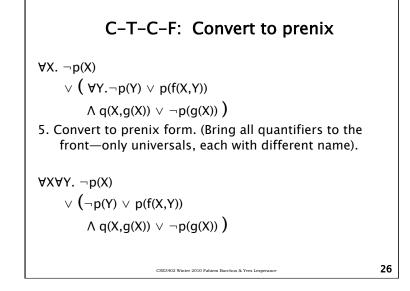
CSE3402 Winter 2010 Fahiem Bacchus & Yves Lesp

23

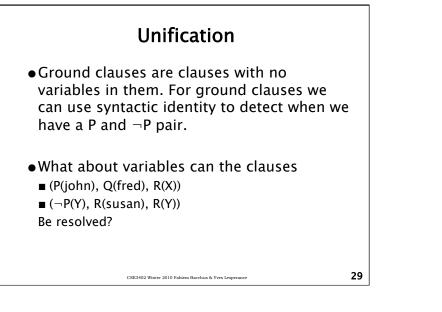
C-T-C-F: Skolemization continue

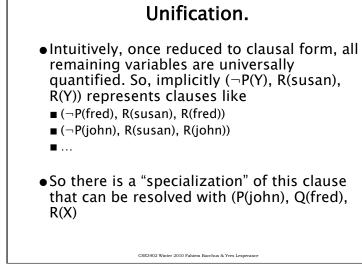
- •We must use a function that mentions every universally quantified variable that scopes the existential.
- In this case X scopes Y so we must replace the existential Y by a function of X
 ∀X. loves(X,g(X)).
 where g is a new function symbol.
- This formula asserts that for every X there is some individual (given by g(X)) that X loves. g(X) can be different for each different binding of X.

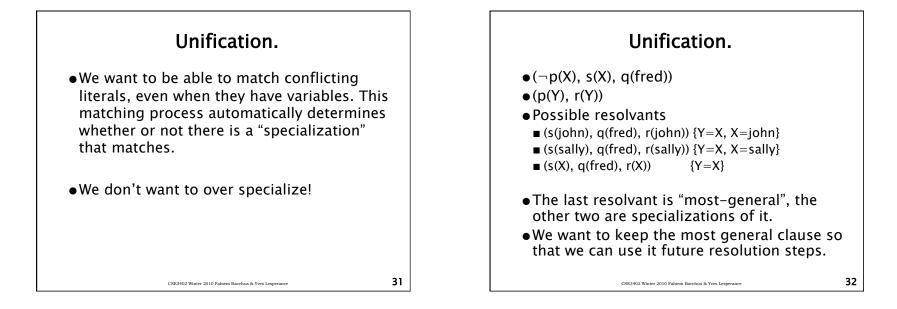


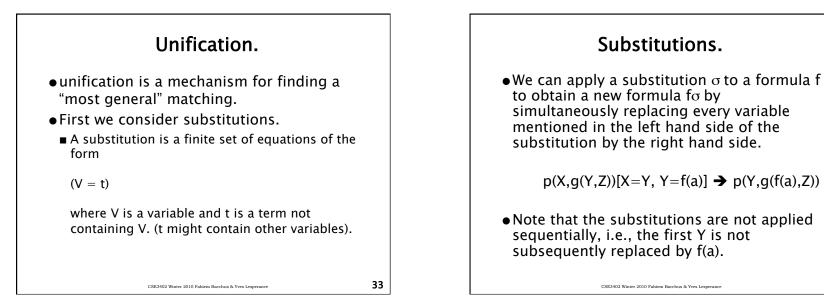


C-T-C-F: Conjunctions over disjunctions $\begin{array}{l} \forall X \forall Y. \neg p(X) \\ \lor (\neg p(Y) \lor p(f(X,Y)) \\ \land q(X,g(X)) \lor \neg p(g(X))) \end{array} \\ \begin{array}{l} 6. \ Conjunctions \ over \ disjunctions \\ A \lor (B \land C) \twoheadrightarrow (A \lor B) \land (A \lor C) \end{array} \\ \begin{array}{l} \forall XY. \ \neg p(X) \lor \neg p(Y) \lor p(f(X,Y)) \\ \land \neg p(X) \lor q(X,g(X)) \lor \neg p(g(X)) \end{array} \end{array}$ C-T-C-F: flatten & convert to clauses 7. Flatten nested conjunctions and disjunctions. $(A \lor (B \lor C)) \Rightarrow (A \lor B \lor C)$ 8. Convert to Clauses (remove quantifiers and break apart conjunctions). $\forall XY. \neg p(X) \lor \neg p(Y) \lor p(f(X,Y))$ $A \neg p(X) \lor q(X,g(X)) \lor \neg p(g(X))$ a) $\neg p(X) \lor \neg p(Y) \lor p(f(X,Y))$ b) $\neg p(X) \lor q(X,g(X)) \lor \neg p(g(X))$









Substitutions. We can compose two substitutions. θ and σ to obtain a new substition θσ.

Let $\theta = \{X_1 = s_1, X_2 = s_2, ..., X_m = s_m\}$ $\sigma = \{Y_1 = t_1, Y_2 = t_2, ..., Y_k = s_k\}$

To compute $\theta\sigma$

1.
$$S = \{X_1 = s_1\sigma, X_2 = s_2\sigma, ..., X_m = s_m\sigma, Y_1 = t_1, Y_2 = t_2, ..., Y_k = s_k\}$$

we apply σ to each RHS of θ and then add all of the equations of $\sigma.$

CSE3402 Winter 2010 Fahiem Bacchus & Yves Les

35

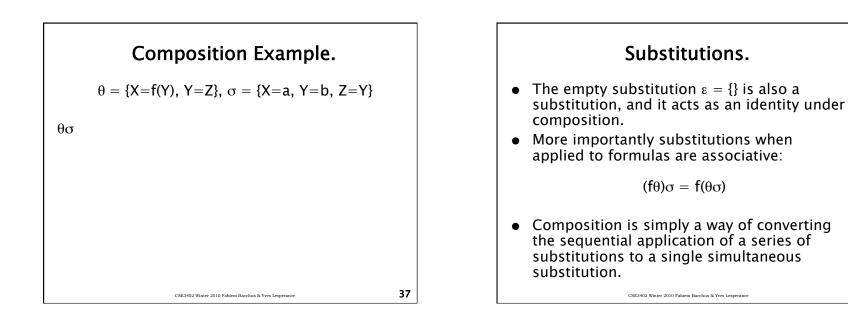
Substitutions.

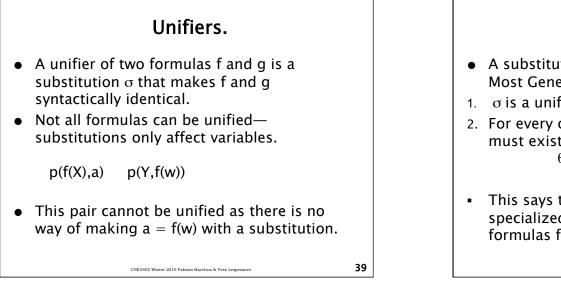
- 1. $S = \{X_1 = s_1\sigma, X_2 = s_2\sigma, ..., X_m = s_m\sigma, Y_1 = t_1, Y_2 = t_2, ..., Y_k = s_k\}$
- 2. Delete any identities, i.e., equations of the form V=V.
- 3. Delete any equation $Y_i = s_i$ where Y_i is equal to one of the X_i in θ .

SE3402 Winter 2010 Fahiem Bacchus & Yves L

The final set S is the composition $\theta\sigma$.

34

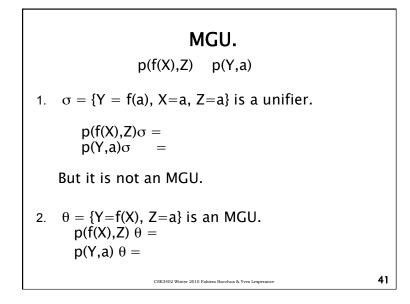


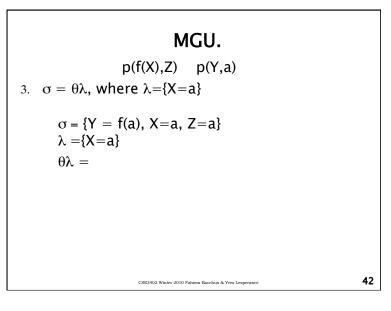


MGU.

- A substitution σ of two formulas f and g is a Most General Unifier (MGU) if
- 1. σ is a unifier.
- 2. For every other unifier θ of f and g there must exist a third substitution λ such that $\theta = \sigma \lambda$
- This says that every other unifier is "more specialized than σ . The MGU of a pair of formulas f and g is unique up to renaming.

38





MGU.

- The MGU is the "least specialized" way of making clauses with universal variables match.
- We can compute MGUs.
- Intuitively we line up the two formulas and find the first sub-expression where they disagree. The pair of subexpressions where they first disagree is called the disagreement set.
- The algorithm works by successively fixing disagreement sets until the two formulas become syntactically identical.

43

MGU.

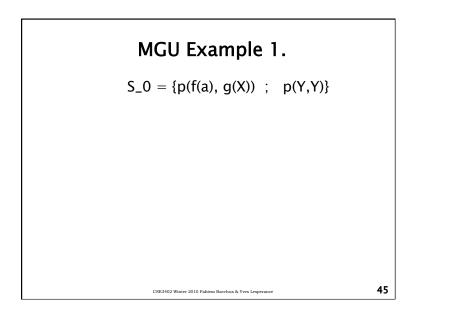
To find the MGU of two formulas f and g.

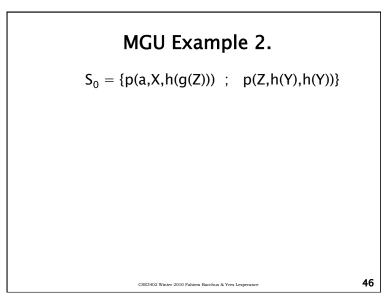
1.
$$k = 0; \sigma_0 = \{\}; S_0 = \{f,g\}$$

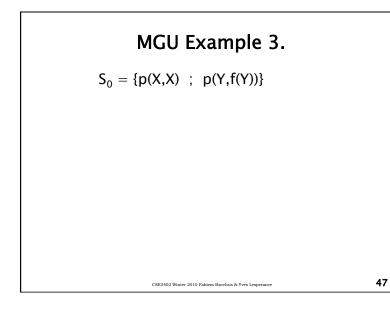
- 2. If S_k contains an identical pair of formulas stop, and return σ_k as the MGU of f and g.
- 3. Else find the disagreement set $D_k = \{e_1, e_2\}$ of S_k
- 4. If $e_1 = V$ a variable, and $e_2 = t$ a term not containing V (or vice-versa) then let $\sigma_{k+1} = \sigma_k \{V=t\}$ (Compose the additional substitution) $S_{k+1} = S_k \{V=t\}$ (Apply the additional substitution) k = k+1GOTO 2

CSE3402 Winter 2010 Fabiert Bacchus & Year Lee

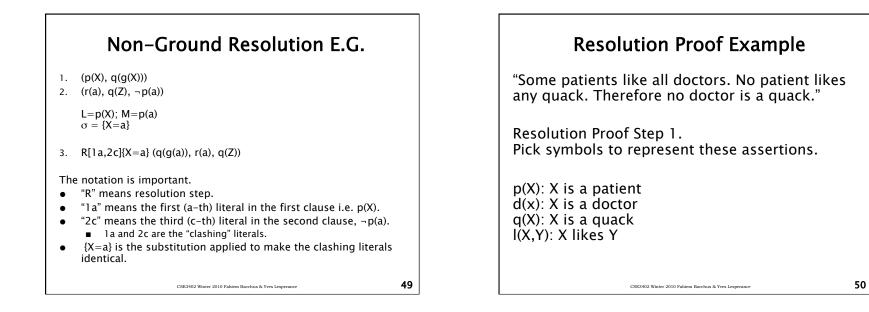
5. Else stop, f and g cannot be unified.







<section-header>Non-Ground Resolution• essolution of non-ground clauses. From the
two clauses
(μ, Q1, Q2, ..., Qk)
(¬M, R1, R2, ..., Rn)Where there exists σ a MGU for L and M.We infer the new clause
(Q1σ, ..., Qkσ, R1σ, ..., Rnσ)



Resolution Proof Example

Resolution Proof Step 2.

Convert each assertion to a first-order formula.

CSE3402 Winter 2010 Fahiem Bacchus & Yves Lesp

1. Some patients like all doctors.

F1.

Resolution Proof Example

CSE3402 Winter 2010 Fahiem Bacchus & Yves Lesp

2. No patient likes any quack

F2.

3. Therefore no doctor is a quack. Query.

51

Resolution	Proof	Example
------------	-------	---------

CSE3402 Winter 2010 Fahiem Bacchus & Yves Lesperance

Resolution Proof Step 3. Convert to Clausal form.

F1.

F2.

Negation of Query.

53

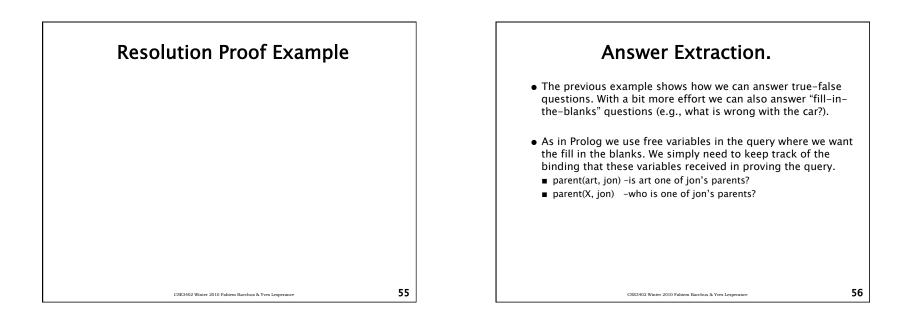
Resolution Proof Example

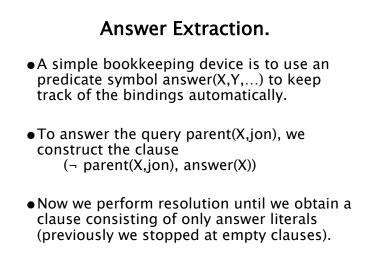
CSE3402 Winter 2010 Fahiem Bacchus & Yves Lesperance

Resolution Proof Step 4.
Resolution Proof from the Clauses.
1. p(a)
2. (¬d(Y), l(a,Y))
3. (¬p(Z), ¬q(R), ¬l(Z,R))
4. d(b)

5. q(b)

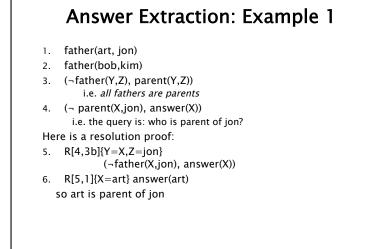




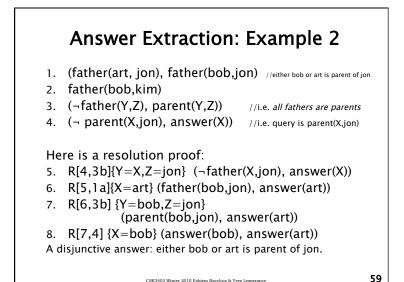


CSE3402 Winter 2010 Fahiem Bacchus & Yves Lesperanc

57



CSE3402 Winter 2010 Fahiem Bacchus & Yves Lesperan



Factoring (optional)1. (p(X), p(Y))// $\forall X.\forall Y. \neg p(X) \Rightarrow p(Y)$ 2. $(\neg p(V), \neg p(W))$ // $\forall V.\forall W. p(V) \Rightarrow \neg p(W)$ • These clauses are intuitively contradictory, but following the strict rules of resolution only we obtain:2. PUL-2c1(X, V) (r(V) = r(V))

3. R[1a,2a](X=V) (p(Y), ¬p(W)) Renaming variables: (p(Q), ¬p(Z))

4. R[3b,1a](X=Z) (p(Y), p(Q))

No way of generating empty clause! Factoring is needed to make resolution complete, without it resolution is incomplete!

SE3402 Winter 2010 Fahiem Bacchus & Yves Les

60

