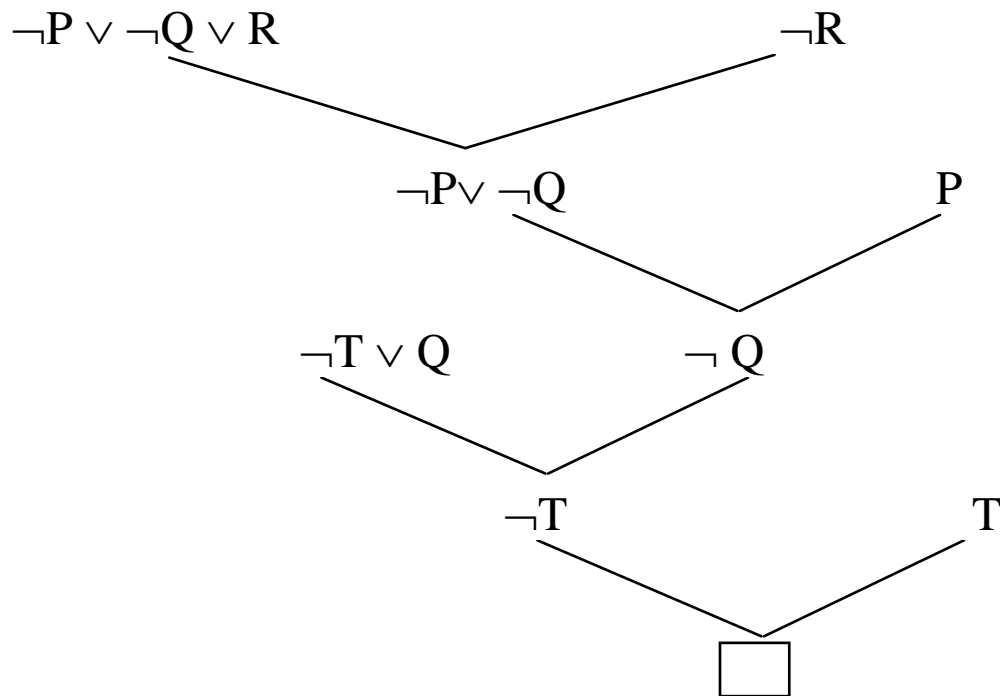


Propositional Resolution

We want to prove R.

Given Axioms	Clause Form	
P	P	(1)
$(P \wedge Q) \rightarrow R$	$\neg P \vee \neg Q \vee R$	(2)
$(S \vee T) \rightarrow Q$	$\neg S \vee Q$	(3)
	$\neg T \vee Q$	(4)
T	T	(5)



Unification

$Man(John)$	$\neg man(John)$
$man(John)$	$\neg man(table)$

$tryassassinate(Marcus, Caesar)$ $hate(Marcus, Caesar)$	whether there exists a set of substitutions that makes them identical
------------------------------------------------------------	-----------------------------------------------------------------------------

$Q(x)$	
$P(y)$	\rightarrow FAIL
$P(x)$	
$P(y)$	$\rightarrow x/y$
$P(Marcus)$	
$P(y)$	$\rightarrow Marcus/y$
$P(Marcus)$	
$P(Julius)$	\rightarrow FAIL
$P(x, x)$	
$P(y, z)$	$\rightarrow (y/x) \quad \begin{matrix} P(y, x) \\ P(y, z) \end{matrix}$
$P(y, y)$	$\rightarrow (z/y)(y/x)$
$P(y, z)$	

$f(x, x)$	expression involving a given variable
$f(g(x), g(x))$	$g(x)/x$
	will result in infinite recursion

Finding General Substitutions

Given:

hate(x, y)

hate(Marcus, z)

We could produce:

(Marcus/x, z/y)

(Marcus/x, y/z)

(Marcus/x, Caesar/y, Caesar/z)

(Marcus/x, Polonius/y, Polonius/z)

Algorithm: Unify ($L1, L2$)

1. If $L1$ or $L2$ is a variable or constant, then:
 - (a) If $L1$ and $L2$ are identical, then return NIL.
 - (b) Else if $L1$ is a variable, then if $L1$ occurs in $L2$ then return FAIL, else return $\{(L2/L1)\}$.
 - (c) Else if $L2$ is a variable, then if $L2$ occurs in $L1$ then return FAIL, else return $\{(L1/L2)\}$.
 - (d) Else return FAIL.
2. If the initial predicate symbols in $L1$ and $L2$ are not identical, then return FAIL.
3. If $L1$ and $L2$ have a different number of arguments, then return FAIL.
4. Set SUBST to NIL.
5. For $i \leftarrow 1$ to number of arguments in $L1$:
 - (a) Call Unify with the i -th argument of $L1$ and the i -th argument of $L2$, putting result in S .
 - (b) If $S = \text{FAIL}$ then return FAIL.
 - (c) If S is not equal to NIL then:
 - i. Apply S to the remainder of both $L1$ and $L2$.
 - ii. $\text{SUBST} := \text{APPEND}(S, \text{SUBST})$.
6. Return SUBST.

Resolution in Predicate Logic

Example:

1. $man(Marcus)$
2. $\neg man(x_1) \vee mortal(x_1)$

yields the substitution:

$Marcus / x_1$

So it does not yield the resolvent:

$mortal(x_1)$

The literal $man(Marcus)$ can be unified with the literal $man(x_1)$ with the substitution $Marcus / x_1$, telling us that for $x_1 = Marcus$, $\neg man(Marcus)$ is false. But we cannot simply cancel out the two man literals as we did in propositional logic and generate the resolvent $mortal(x_1)$. Clause 2 says that for a given x_1 , either $\neg man(x_1)$ or $mortal(x_1)$. So for it to be true, we can now conclude only that $mortal(Marcus)$ must be true. It is not necessary that $mortal(x_1)$ be true for all x_1 , since for some values of x_1 , $\neg man(x_1)$ might be true, making $mortal(x_1)$ irrelevant to the truth of the complete clause. So the resolvent generated by clauses 1 and 2 must be $mortal(Marcus)$, which we get by applying the result of the unification process to the resolvent.

It does yield: $mortal(Marcus)$

Resolution in Predicate Logic

Algorithm: Resolution

1. Convert all the statements of F to clause form.
2. Negate P and convert the result to clause form. Add it to the set of clauses obtained in 1.
3. Repeat until either a contradiction is found, no progress can be made, or a predetermined amount of effort has been expended.
 - (a) Select two clauses. Call these the parent clauses.
 - (b) Resolve them together. The resolvent will be the disjunction of all the literals of both parent clauses with appropriate substitutions performed and with the following exception: If there is one pair of literals $T1$ and $\neg T2$ such that one of the parent clauses contains $T1$ and the other contains $T2$ and if $T1$ and $T2$ are unifiable, then neither $T1$ nor $T2$ should appear in the resolvent. We call $T1$ and $T2$ **complementary literals**. Use the substitution produced by the unification to create the resolvent. If there is more than one pair of complementary literals, only one pair should be omitted from the resolvent.
4. If the resolvent is the empty clause, then a contradiction has been found. If it is not, then add it to the set of clauses available to the procedure.

Resolution Guidelines

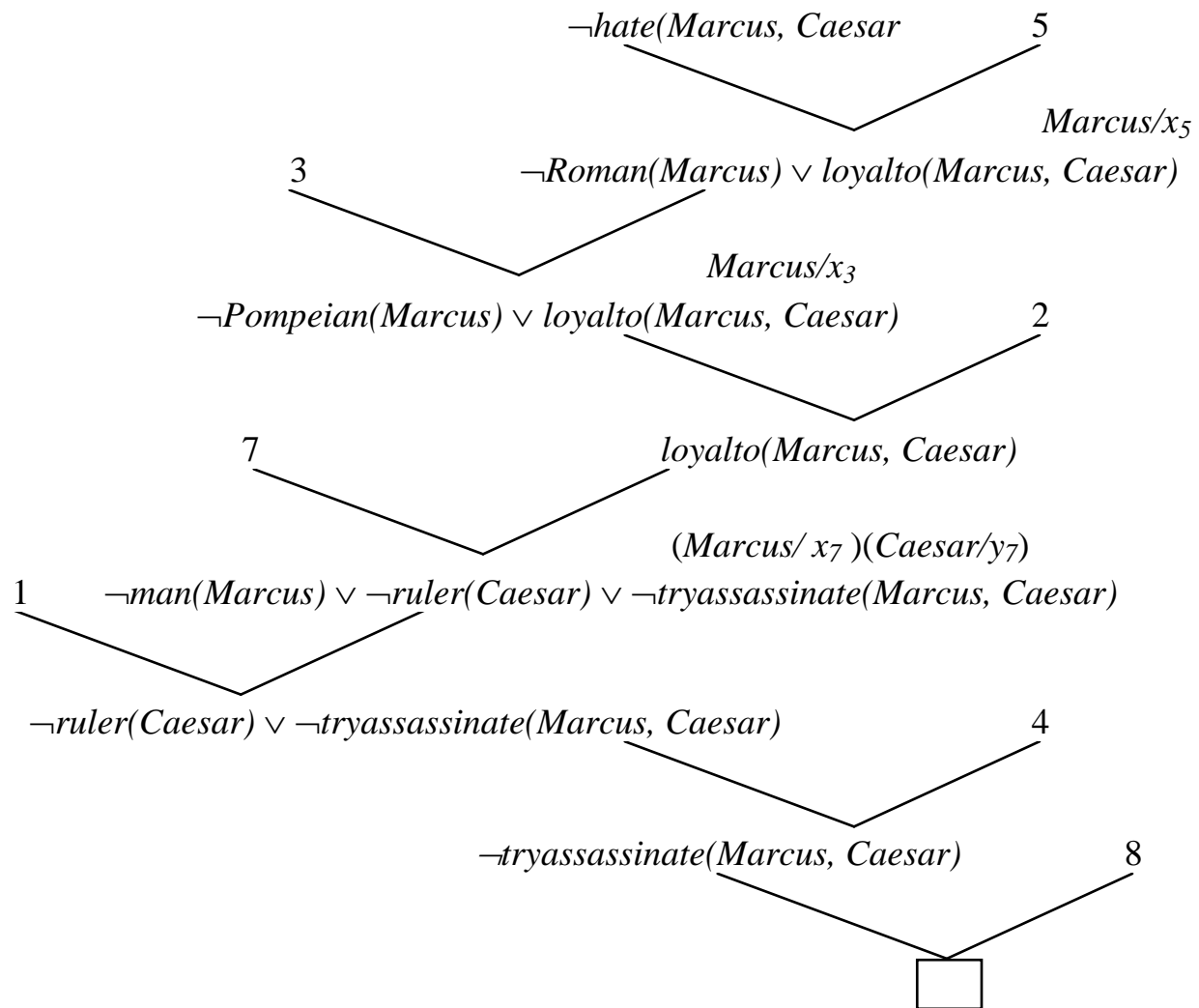
- Only resolve pairs of clauses that contain complementary literals, since only such resolutions produce new clauses that are harder to satisfy than their parents. To facilitate this, index clauses by the predicates they contain, combined with an indication of whether the predicate is negated. Then, given a particular clause, possible resolvents that contain a complementary occurrence of one of its predicates can be located directly.
- Eliminate certain clauses as soon as they are generated so that they cannot participate in later resolutions. Two kinds of clauses should be eliminated: tautologies (which can never be unsatisfied) and clauses that are subsumed by other clauses (i.e., they are easier to satisfy). For example, $P \vee Q$ is subsumed by P .)
- Whenever possible, resolve either with one of the clauses that is part of the statement we are trying to refute or with a clause generated by a resolution with such a clause. This is called the *set-of-support strategy* and corresponds to the intuition that the contradiction we are looking for must involve the statement we are trying to prove. Any other contradiction would say that the previously believed statements were inconsistent.
- Whenever possible, resolve with clauses that have a single literal. Such resolutions generate new clauses with fewer literals than larger of their parent clauses and thus are probably closer to the goal of a resolvent with zero terms. This method is called the *unit-preference strategy*.

A Predicate Logic Example

1. Marcus was a man.
Man(Marcus)
2. Marcus was a Pompeian.
Pompeian(Marcus)
3. All Pompeians were Romans.
 $\forall x: Pompeian(x) \rightarrow Roman(x)$
4. Caesar was a Ruler.
ruler(Caesar)
5. All Romans were either loyal to Caesar or hated him.
 $\forall x: Roman(x) \rightarrow loyalto(x, Caesar) \vee hate(x, Caesar)$
6. Everyone is loyal to someone.
 $\forall x: \exists y: loyalto(x, y)$
7. Persons only try to assassinate rulers they are not loyal to.
 $\forall x: \forall y: man(x) \wedge ruler(y) \wedge tryassassinate(x, y) \rightarrow$
 $\neg loyalto(x, y)$
8. Marcus tried to assassinate Caesar.
tryassassinate(Marcus, Caesar)

An Example of Resolution

1. $man(Marcus)$
 2. $Pompeian(Marcus)$
 3. $\neg Pompeian(x_3) \vee Roman(x_3)$
 4. $Ruler(Caesar)$
 5. $\neg Roman(x_5) \vee loyalto(x_5, Caesar) \vee hate(x_5, Caesar)$
 6. $loyalto(x_6, f_1(x_6))$
 7. $\neg man(x_7) \vee \neg ruler(y_7) \vee \neg tryassassinate(x_7, y_7) \vee \neg loyalto(x_7, y_7)$
 8. $tryassassinate(Marcus, Caesar)$
- Prove: $hate(Marcus, Caesar)$



Another Example of Resolution

Everyone has a parent. The parent of parent is a grandparent. Given the person John, prove that John has a grandparent.

The following sentences represent the facts and relationships in the situation above:

Everyone has a parent.

$$\forall x \exists y \mathbf{P}(x, y)$$

A parent of a parent is a grandparent.

$$\forall x \forall y \forall z \mathbf{P}(x, y) \wedge \mathbf{P}(y, z) \longrightarrow \mathbf{GP}(x, z)$$

The goal is to find a w such that $\mathbf{GP}(\text{John}, w)$ or $\exists w \mathbf{GP}(\text{John}, w)$. The negation of the goal is $\neg(\exists w \mathbf{GP}(\text{John}, w))$ or $\forall w \neg\mathbf{GP}(\text{John}, w)$.

In the process of putting the predicates above in clause form for the resolution refutation, the existential quantifier in the first predicate (everyone has a parent) requires Skolem function. This Skolem function would be the obvious function: take the given and find the parent of x . Let's call this the $\mathbf{PA}(x)$ for "find a parental ancestor for x ". For John, this would be either his father or his mother. The clause form for the predicates of this problem is:

1. $\mathbf{P}(x, \mathbf{PA}(x))$
2. $\neg\mathbf{P}(w, y) \vee \neg\mathbf{P}(y, z) \vee \mathbf{GP}(w, z)$
3. $\neg\mathbf{GP}(\text{John}, v)$

