- Here the antecedent is $\forall x\ \exists y\ R(x, y)$; replacing y by the Skolem function f yields the clause $\{R(x,\ f(x))\}$.
- The negation of the consequent is $\neg(\exists y\ \forall x\ R(x, y))$, which becomes $\forall y\ \exists x\ \neg R(x, y)$. Replacing x by the Skolem function g yields the clause $\{\neg R(g(y), y)\}$.

Observe that $R(x,\ f(x))$ and $R(g(y), y)$ are not unifiable because of the occurs check. And so it should be, because the original formula is not a theorem!

Exercise 31 For each of the following pairs of terms, give a most general unifier or explain why none exists. Do not rename variables prior to performing the unification.

$f(g(x), z)$	$f(y, h(y))$
$j(x, y, z)$	$j(f(y, y),\ f(z, z),\ f(a, a))$
$j(x, z, x)$	$j(y,\ f(y), z)$
$j(f(x), y, a)$	$j(y, z, z)$
$j(g(x), a, y)$	$j(z, x,\ f(z, z))$

10 Applications of Unification

By means of unification, we can extend resolution to first-order logic. As a special case we obtain Prolog. Other theorem provers are also based on unification. Other applications include polymorphic type checking for the language ML.

10.1 Binary resolution

We now define the binary resolution rule with unification:

$$\frac{\{B, A_1, \ldots, A_m\} \quad \{\neg D, C_1, \ldots, C_n\}}{\{A_1, \ldots, A_m, C_1, \ldots, C_n\}\sigma} \qquad \text{provided } B\sigma = D\sigma$$

As before, the first clause contains B and other literals, while the second clause contains $\neg D$ and other literals. The substitution σ is a unifier of B and D (almost always a *most general* unifier). This substitution is applied to all remaining literals, producing the conclusion.

The variables in one clause are renamed before resolution to prevent clashes with the variables in the other clause. Renaming is sound because the scope of each variable is its clause. Resolution is sound because it takes an instance of each clause — the instances are valid, because the clauses are universally valid —

and then applies the propositional resolution rule, which is sound. For example, the two clauses

$$\{P(x)\} \quad \text{and} \quad \{\neg P(g(x))\}$$

yield the empty clause in a single resolution step. This works by renaming variables — say, x to y in the second clause — and unifying $P(x)$ with $P(g(y))$. Forgetting to rename variables is fatal, because $P(x)$ cannot be unified with $P(g(x))$.

10.2 Factoring

In the general case, the resolution rule must perform *factoring*. This uses additional unifications to identify literals in the same clause. Factoring can make the clause $\{P(x, b), P(a, y)\}$ behave like the clause $\{P(a, b)\}$, since $P(a, b)$ is the result of unifying $P(x, b)$ with $P(a, y)$.

The factoring unifications are done at the same time as the unification of the complementary literals in the two clauses. The binary resolution rule with factoring is

$$\frac{\{B_1, \ldots, B_k, A_1, \ldots, A_m\} \quad \{\neg D_1, \ldots, \neg D_l, C_1, \ldots, C_n\}}{\{A_1, \ldots, A_m, C_1, \ldots, C_n\}\sigma}$$

where σ is the most general substitution such that

$$B_1\sigma = \cdots = B_k\sigma = D_1\sigma = \cdots = D_l\sigma.$$

Resolution with factoring is *refutation complete*: it will find a contradiction if there is one. Showing this is difficult.

The search space is huge: resolution with factoring can be applied in many different ways, every time. Modern resolution systems use highly complex heuristics to limit the search. Typically they only perform resolutions that can lead (perhaps after several steps) to very short clauses, and they discard the intermediate clauses produced along the way. Dozens of flags and parameters influence their operation.

Example 36 Let us prove $\forall x\, \exists y\, \neg(P(y, x) \leftrightarrow \neg P(y, y))$.

Negate and expand the $\leftrightarrow$, getting

$$\neg\forall x\, \exists y\, \neg((\neg P(y, x) \vee \neg P(y, y)) \wedge (\neg\neg P(y, y) \vee P(y, x)))$$

Its negation normal form is

$$\exists x\, \forall y\, ((\neg P(y, x) \vee \neg P(y, y)) \wedge (P(y, y) \vee P(y, x)))$$

Skolemization yields

$$(\neg P(y,a) \vee \neg P(y,y)) \wedge (P(y,y) \vee P(y,a))$$

The clauses are

$$\{\neg P(y,a), \neg P(y,y)\} \qquad \{P(y,y), P(y,a)\}$$

Note that $\neg P(a,a)$ is an instance of the first clause and that $P(a,a)$ is an instance of the second, contradiction. This is a one-step proof! But it involves both resolution and factoring, since the 2-literal clauses must collapse to singleton clauses.

Example 37 Let us prove $\exists x\,[P \rightarrow Q(x)] \wedge \exists x\,[Q(x) \rightarrow P] \rightarrow \exists x\,[P \leftrightarrow Q(x)]$. The clauses are

$$\{P, \neg Q(b)\} \quad \{P, Q(x)\} \quad \{\neg P, \neg Q(x)\} \quad \{\neg P, Q(a)\}$$

A short resolution proof follows. The complementary literals are underlined:

Resolve	$\{P, \underline{\neg Q(b)}\}$	with	$\{P, \underline{Q(x)}\}$	getting $\{P\}$
Resolve	$\{\neg P, \underline{\neg Q(x)}\}$	with	$\{\neg P, \underline{Q(a)}\}$	getting $\{\neg P\}$
Resolve	$\{\underline{P}\}$	with	$\{\underline{\neg P}\}$	getting $\square$

Exercise 32 Show the steps of converting $\exists x\,[P \rightarrow Q(x)] \wedge \exists x\,[Q(x) \rightarrow P] \rightarrow \exists x\,[P \leftrightarrow Q(x)]$ into clauses. Then show two resolution proofs different from the one shown above.

Exercise 33 Is the clause $\{P(x,b), P(a,y)\}$ logically equivalent to the unit clause $\{P(a,b)\}$? Is the clause $\{P(y,y), P(y,a)\}$ logically equivalent to $\{P(y,a)\}$? Explain both answers.

10.3 Prolog clauses

Prolog clauses, also called Horn clauses, have at most one positive literal. A *definite* clause is one of the form

$$\{\neg A_1, \ldots, \neg A_m, B\}$$

It is logically equivalent to $(A_1 \wedge \cdots \wedge A_m) \rightarrow B$. Prolog's notation is

$$B \leftarrow A_1, \ldots, A_m.$$