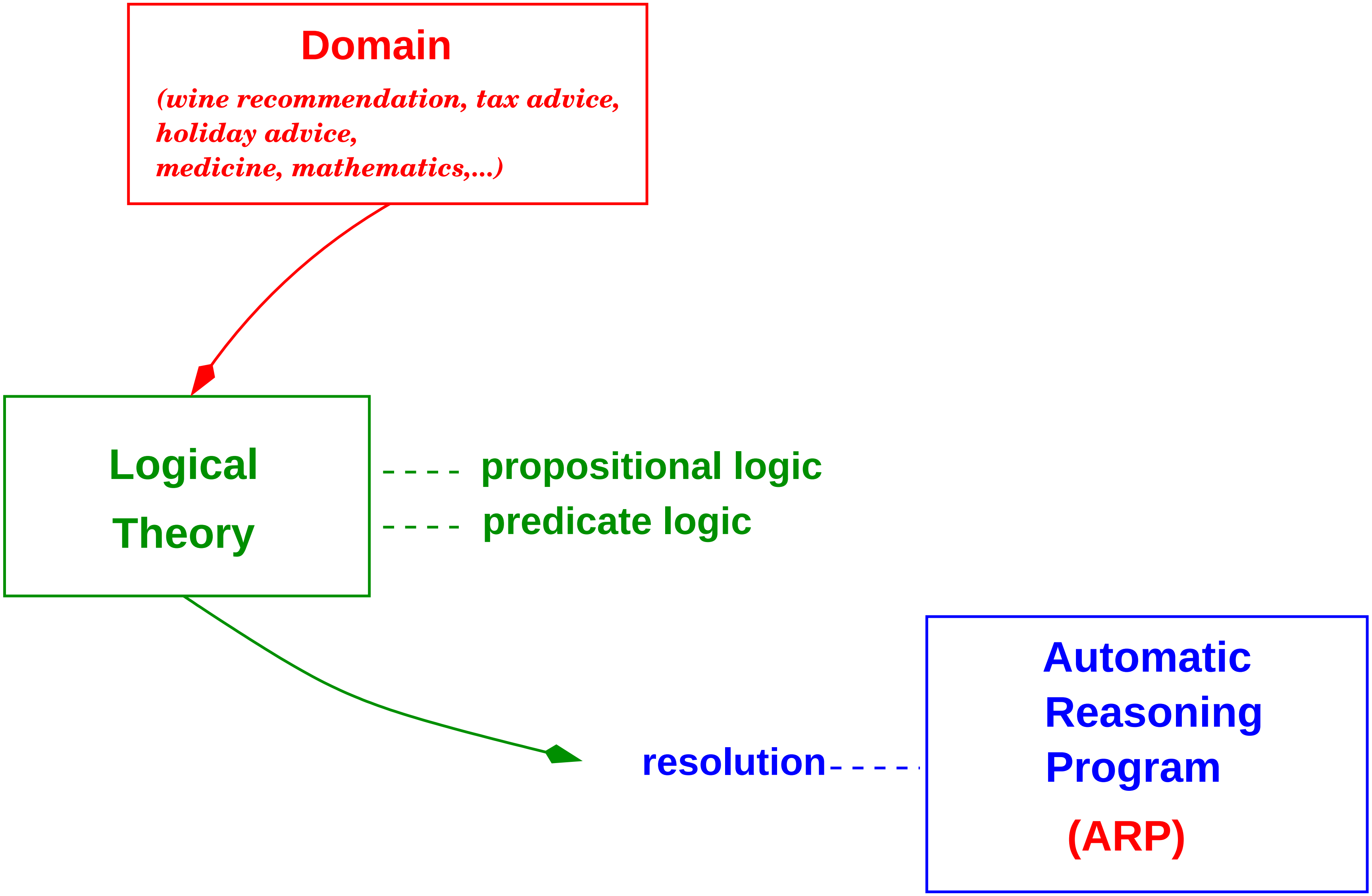
Logic and Resolution

Representation and Reasoning

KR&R using Logic



Goals for Today

- Refresh your memory about logic
- Make sure everyone understands the notation
- Learn the basic method for automated reasoning systems: resolution

$\Rightarrow$ forms the basis for a large part of the course

Warm-up Quiz

Which of the following statements are true?

(a) IsHuman $\rightarrow$ IsMammal

(b) $\models$ IsHuman $\rightarrow$ IsMammal

(c) $\models \neg($IsMammal $\rightarrow$ IsHuman$)$

(d) IsHuman $\not\models$ IsMammal

(e) IsHuman $\rightarrow$ IsMammal $\models$ IsMammal $\rightarrow$ IsHuman

(f) IsHuman $\rightarrow$ IsMammal $\models$ $\neg$IsMammal $\rightarrow$ $\neg$IsHuman

(g) $\exists x($IsHuman$(x)) \models$ IsHuman$($child$($Mary$))$

(h) $\forall x(P(x) \vee \neg Q(a)) \models Q(a) \rightarrow P(b)$

(i) $\forall x(P(x) \vee Q(x)) \wedge \forall y(\neg P(f(y)) \vee R(y)) \models \forall z(Q(f(z)) \vee R(z))$

Logic Concepts

- If a formula φ is true under a given interpretation M, one says that that M satisfies φ, $M \models \varphi$
- A formula is satisfiable if there is some interpretation under which it is true
- Otherwise, it is unsatisfiable (inconsistent)
- A formula is valid (a tautology), denoted by $\models \varphi$, if it is true in every interpretation

$$\text{for all } M\colon M \models \varphi$$

- A formula φ is entailed (or is a logical consequence) by a conjunction of formulas (sometimes called a theory) Γ, denoted by $\Gamma \models \varphi$, if

$$\text{for all } M\colon M \models \Gamma \text{ then } M \models \varphi$$

Proposition Logic

- Well-formed formulas $\mathcal{F}$: *constants* $\Box$, *propositional symbols* (atoms) P, *negation* $\neg\varphi$, *disjunction* $(\varphi \vee \psi)$, *conjunction* $(\varphi \wedge \psi)$, *implication* $(\varphi \rightarrow \psi)$
- Interpreted with a function w:

$$w : \mathcal{F} \rightarrow \{\textit{true}, \textit{false}\}$$

  such that for w holds:
  - $w(\neg\varphi) = \textit{true}$ if and only if $w(\varphi) = \textit{false}$
  - $w(\varphi \wedge \psi) = \textit{true}$ iff $w(\varphi) = \textit{true}$ and $w(\psi) = \textit{true}$
  - etc.
- Equivalently, we could restrict ourselves to an assignment of truth to the propositional symbols
- If w is a model of φ then this is denoted by $\boxed{w \models \varphi}$

Propositional Logic: Example

"Because the classroom was small (S) and many students subscribed to the course (M), there was a shortage of chairs (C)"

Formally: $((S \wedge M) \rightarrow C)$

- If $w(S) = w(M) = w(C) =$ ***true***, then $w \vDash ((S \wedge M) \rightarrow C)$,
- Also: $((S \wedge M) \wedge ((S \wedge M) \rightarrow C)) \vDash C$; other notation: $\{S, M, (S \wedge M) \rightarrow C\} \vDash C$
- If $w'(S) = w'(M) =$ ***true*** and $w'(C) =$ ***false***:

$$\{S, M, \neg C, (S \wedge M) \rightarrow C\} \vDash \Box$$

- Short notation: remove some brackets ...

Deduction Rules

- Instead of truth assignments (interpretations) we can focus only the syntax (the form ...)

  Examples:
  - modus ponens: $\frac{\varphi, \varphi \rightarrow \psi}{\psi}$
  - $\wedge$-introduction rule: $\frac{\varphi, \psi}{\varphi \wedge \psi}$
- Replace $\models$ by syntactical manipulation (derivation) $\vdash$

  Examples: Given: P, Q and $(P \wedge Q) \rightarrow R$, then
  - $\{P, Q\} \vdash P \wedge Q$ ($\wedge$-introduction)
  - $\{P, Q, (P \wedge Q) \rightarrow R\} \vdash R$ ($\wedge$-introduction and modus ponens)

Deduction Concepts

- A deductive system S is a set of axioms and rules of inference for deriving theorems
- A formula φ can be deduced by a set of formulae Γ if φ can be proven using a deduction system S, written as $\Gamma \vdash_S \varphi$
- A deductive system S is sound if

$$\Gamma \vdash_S \varphi \Rightarrow \Gamma \models \varphi$$

- A deductive system S is complete if

$$\Gamma \models \varphi \Rightarrow \Gamma \vdash_S \varphi$$

- A deductive system S is refutation-complete if

$$\Gamma \models \Box \Rightarrow \Gamma \vdash_S \Box$$

Resolution

- Reason only with formulas in its clausal form:

$$L_1 \vee L_2 \vee \cdots \vee L_n$$

with L_i a literal, i.e., an atom or a negation of an atom; if $n = 0$, then it is $\Box$ (empty clause)

- Complementary literals: L and L', such that $L \equiv \neg L'$

- Resolution (rule) $\mathcal{R}$ (J.A. Robinson, 1965):

$$\frac{C \vee L, \quad C' \vee \neg L}{D}$$

with C, C' clauses, and D the (binary) resolvent equal to $C \vee C'$ (repeating literals may be removed)

Examples Resolution

- Given $V = \{P \vee Q \vee \neg R, U \vee \neg Q\}$, then

$$\frac{P \vee Q \vee \neg R, \quad U \vee \neg Q}{P \vee U \vee \neg R}$$

  so $V \vdash_{\mathcal{R}} P \vee U \vee \neg R$ by applying the resolution rule $\mathcal{R}$ once
- Given $V = \{\neg P \vee Q, \neg Q, P\}$, then $\frac{\neg P \vee Q, \quad \neg Q}{\neg P}$ and $\frac{\neg P, \quad P}{\square}$ so $V \vdash_{\mathcal{R}} \square$
- If $V \vdash_{\mathcal{R}} \square$ then it will hold that V is inconsistent and the derivation will then be called a refutation
  - $\mathcal{R}$ is sound
- If $V \nvdash_{\mathcal{R}} \square$, then V is consistent
  - $\mathcal{R}$ is refutation-complete

Motivation for Resolution

- Proving unsatisfiability is enough, because:

$$\Gamma \models \varphi \Leftrightarrow \Gamma \cup \{\neg\varphi\} \models \Box$$

- If a theory Γ is inconsistent, then resolution will eventually terminate with a derivation such that

$$\Gamma \vdash_{\mathcal{R}} \Box$$

- For first-order logic, resolution may not terminate for consistent theories...

Many applications:

- Mathematics: Robbins' conjecture
- Proving that medical procedures are correct
- Logic programming

Soundness Resolution

- Theorem: resolution is sound (so, $V \vdash_{\mathcal{R}} C \Rightarrow V \models C$)

  Proof (sketch). Suppose $C_1 = L \vee C_1'$ and $C_2 = \neg L \vee C_2'$, so using resolution we find:

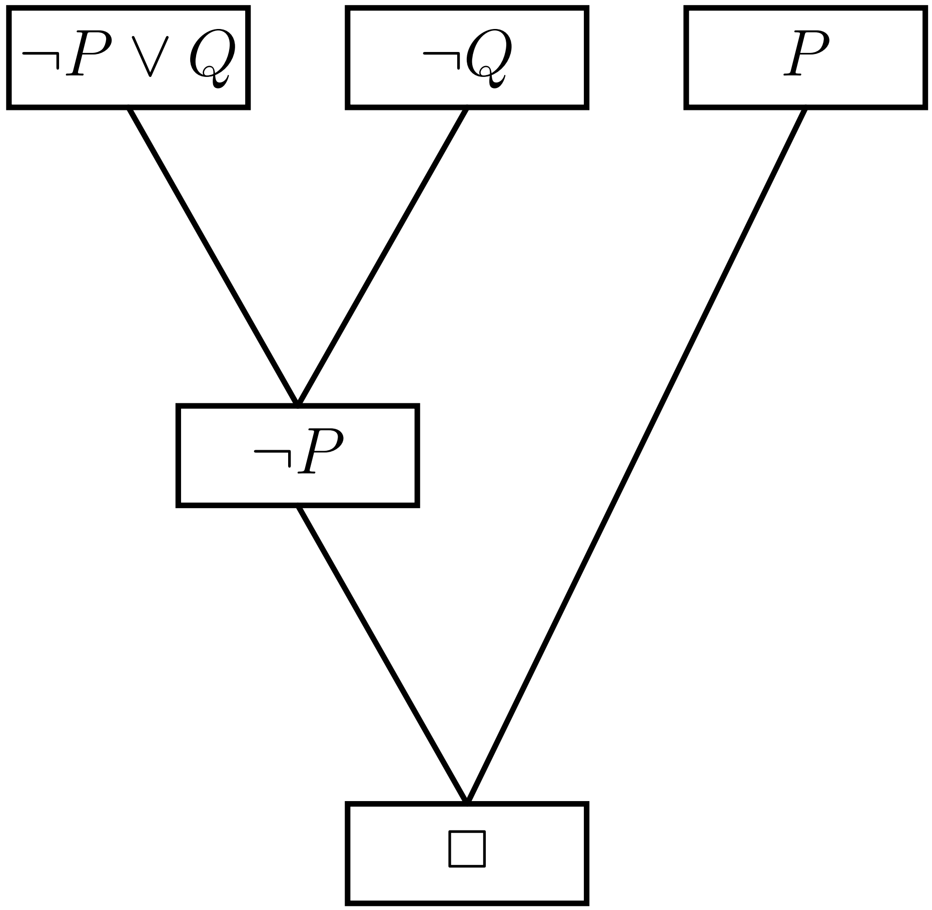
  $$\{C_1, C_2\} \vdash_{\mathcal{R}} D$$

  with D equal to $C_1' \vee C_2'$. We thus need to prove:

  $$w \models (C_1 \wedge C_2) \;\Rightarrow\; w \models D$$

  for every w. It holds that either L or $\neg L$ is true in w. Suppose L is true, then C_2' must be true, so D is true Similar for when $\neg L$ is true.

Resolution (Refutation) Tree



Given $V = \{\neg P \vee Q, \neg Q, P\}$, then $V \vdash_{\mathcal{R}} \square$
Note: resolution trees are *not* unique

SLD Resolution

- Horn clause: clause with maximally one positive literal $\neg A_1 \vee \cdots \vee \neg A_m \vee B$, also denoted by $B \leftarrow A_1, \ldots, A_m$
- SLD resolution (for Horn clauses):

$$\frac{\leftarrow B_1, \ldots, B_n, \qquad B_i \leftarrow A_1, \ldots, A_m}{\leftarrow B_1, \ldots, B_{i-1}, A_1, \ldots, A_m, B_{i+1}, \ldots, B_n}$$

- SLD derivation: a sequence $G_0, G_1, \ldots$ and $C_1, C_2, \ldots$

Exercise:

$$\Gamma = \{R \leftarrow T, T \leftarrow, P \leftarrow R\}$$

Prove P from Γ using SLD resolution.

First-order Logic

- Allow the representation of entities (also called objects) and their properties, and relations among such entities
- More expressive than propositional logic
- Distinguished from propositional logic by its use of quantifiers
  - Each interpretation of first-order logic includes a domain of discourse over which the quantifiers range
- Additionally, it covers predicates
  - Used to represent either a property or a relation between entities
- Basis for many other representation formalisms

First-order Logic: Syntax

Well-formed formulas are build up from:

- Constants: denoted by $a, b, \ldots$ (or sometimes names such as 'Peter' and 'Martijn')
- Variables: denoted by $x, y, z \ldots$
- Functions: maps (sets of) objects to other objects, e.g. *father*, *plus*, $\ldots$
- Predicates: resemble a function that returns either *true* or *false*: *Brother-of*, *Bigger-than*, *Has-color*, $\ldots$
- Quantifiers: allow the representation of properties that hold for a collection fo objects. Consider a variable x,
  - Existential: $\exists x$, 'there is an x'
  - Universal: $\forall x$, 'for all x'
- Logical connectives and auxiliary symbols

First-order Logic: Interpretations

Formulas are interpreted by a variable assignment v and I based on a structure

$$S = (D, \{f_i\}_i, \{P_j\}_j)$$

consisting of

- A domain of discourse D (typically non-empty)
- f_i is a function $D^n \to D$ for some n
- P_j is a relation, i.e., $P_j \subseteq D^n$ or $P_j : D^n \to \{\textit{true}, \textit{false}\}$ for some n

Then:

- v maps all variables to a $d \in D$
- I maps all n-ary functions/predicates in the language to n-ary functions/relations in the structure

First-order Logic: Example Model

- A simple structure S could consist of:
  - D the set of natural numbers, $D = \{0, 1, 2, \ldots\}$
  - 2-ary function '$+$' (regular addition)
  - 2-ary relationship '$>$' (regular greater than)
- A function symbol f/2 can be interpreted as '$+$'

$$I(f) = +$$

- The constant $\perp$ can be interpreted by the constant 0

$$I(\perp) = 0$$

- The predicate P could mean $>$, i.e., $P(x, y)$ means 'x is greater than y'

$$I(P) = >$$

First-order Logic: Truth

- A predicate is true if the interpretation of the predicate evaluates to 'true' (in the structure)
- Logical connectives are interpreted just like in proposition logic
- $\forall x \varphi(x)$ is true if φ is true for all variable assignments
- $\exists x \varphi(x)$ is true if φ is true for some variable assignments

Example

- Consider the formula $\forall x \exists y \exists z P(f(y, z), x)$
- Given the structure S, this formula is clearly true
- Note, however, that this would not be the case if we had, for instance, interpreted P as 'less than'

First-order Clausal Form

First-order resolution only uses clauses

$$\forall x_1 \dots \forall x_s (L_1 \vee \dots \vee L_m) \text{ written as } L_1 \vee \dots \vee L_m$$

$\Rightarrow$ we will translate formulas in predicate logic to a clausal normal form:

1. Convert to negation normal form: eliminate implications and move negations inwards
2. Make sure each bound variable has a unique name
3. Skolemize: replace $\exists x$ by terms with function symbols of previously universally quanified variables
   $\forall x \exists y P(x, y)$ becomes $\forall x P(x, f(x))$
4. Put it into a conjunctive normal form by using the distributive laws and put the quantifiers up front

Each conjunct is now a clause

Skolemisation: Underlying Idea

What you have is that,

$$\forall x\Big(g(x) \vee \exists y R(x,y)\Big) \Rightarrow \forall x\Big(g(x) \vee R(x, f(x))\Big)$$

where $f(x)$ is the (Skolem) function that maps x to y

- "For every x there is a y s.t. $R(x,y)$" is converted into "There is a function f mapping every x into a y s.t. for every x $R(x, f(x))$ holds"
- $\forall x \exists y R(x,y)$ is satisfied by a model M iff
  - For each possible value for x there is a value for y that makes $R(x,y)$ true
  - which implies: there exists a function f s.t. $y = f(x)$ such that $R(x,y)$ holds

Skolemization: Example

- Given a formula
  $\exists x \text{Father}(x, \text{amalia}) \wedge \neg \exists x \exists y (\text{Father}(x, y) \wedge \neg \text{Parent}(x, y))$

1. Move negations inwards:

$$\exists x \text{Father}(x, \text{amalia}) \wedge \forall x \forall y (\neg \text{Father}(x, y) \vee \text{Parent}(x, y))$$

2. Make variable names unique:

$$\exists z \text{Father}(x, \text{amalia}) \wedge \forall x \forall y (\neg \text{Father}(x, y) \vee \text{Parent}(x, y))$$

3. Skolemize (suggestively replacing z by 'alex')

$$\text{Father}(\text{alex}, \text{amalia}) \wedge \forall x \forall y (\neg \text{Father}(x, y) \vee \text{Parent}(x, y))$$

4. To clausal normal form:

$$\forall x \forall y (\text{Father}(\text{alex}, \text{amalia}) \wedge (\neg \text{Father}(x, y) \vee \text{Parent}(x, y))$$

Resolution and First-order Logic

- Problem: given

  $S = \{\forall x \forall y(\text{Father}(\text{alex}, \text{amalia}) \wedge (\neg\text{Father}(x, y) \vee \text{Parent}(x, y))\}$

  We know $S \vDash \text{Parent}(\text{alex}, \text{amalia})$
- Extract the clauses (for resolution):

  $S' = \{\text{Father}(\text{alex}, \text{amalia}), \neg\text{Father}(x, y) \vee \text{Parent}(x, y)\}$
- Solution: substitute x with 'alex' and substitute y with 'amalia'

  $\Rightarrow$ substitution $\sigma = \{\text{alex}/x, \text{amalia}/y\}$
- Application of resolution:

$$\frac{\text{Father}(\text{alex}, \text{amalia}), \{\neg\text{Father}(x, y) \vee \text{Parent}(x, y)\}\sigma}{\text{Parent}(\text{alex}, \text{amalia})}$$

so $S' \vdash_{\mathcal{R}} \text{Parent}(\text{alex}, \text{amalia})$

Substitution

- A substitution σ is a finite set of the form $\sigma = \{t_1/x_1, \ldots, t_n/x_n\}$, with x_i a variabele and t_i a term; $x_i \neq t_i$ and $x_i \neq x_j$, $i \neq j$
- $E\sigma$ is an expression derived from E by simultaneously replacing all occurences of the variables x_i by the terms t_i. $E\sigma$ is called an instantiation
- If $E\sigma$ does not contain variables, then $E\sigma$ is called a ground instance

Examples for $C = P(x, y) \vee Q(y, z)$:

- $\sigma_1 = \{a/x, b/y\}$, $\sigma_2 = \{y/x, x/y\}$: $C\sigma_1 = P(a, b) \vee Q(b, z)$ en $C\sigma_2 = P(y, x) \vee Q(x, z)$
- $\sigma_3 = \{f(y)/x, g(b)/z\}$: $C\sigma_3 = P(f(y), y) \vee Q(y, g(b))$

Making Things Equal

- Compare $\neg\text{Father}(\text{alex}, \text{amalia})$ and $\neg\text{Father}(x, y)$. What are the differences and the similarities?
  - complementary sign
  - the same predicate symbol ('Father')
  - constant 'alex' versus variable x en constant 'amalia' versus variable y

  Make things equal through substitution

$$\sigma = \{\text{alex}/x, \text{amalia}/y\}$$

- Compare $P(x, f(x))$ and $\neg P(g(a), f(g(a)))$; after removing the sign, make them equal with

$$\sigma = \{g(a)/x\}$$

- Making things syntactically equal = unification

Unification

- Let $\theta = \{t_1/x_1, \ldots, t_m/x_m\}$ and $\sigma = \{s_1/y_1, \ldots, s_n/y_n\}$, then the composition, denoted by $\theta \circ \sigma$ or $\theta\sigma$, is defined by:

$$\{t_1\sigma/x_1, \ldots, t_m\sigma/x_m, s_1/y_1, \ldots, s_n/y_n\}$$

  where each element $t_i\sigma/x_i$ is removed for which $x_i = t_i\sigma$ and also each element s_j/y_j for which $y_j \in \{x_1, \ldots, x_m\}$

- A substitution σ is called a unifier of E and E' if $E\sigma = E'\sigma$; E and E' are then called unifiable

- A unifier θ of expressions E en E' is called the most general unifier (mgu) if and only if for each unifier σ of E and E' there exists a substitution λ such that $\sigma = \theta \circ \lambda$
  $\Rightarrow$ derive expressions which are as general as possible (with variables)

Examples Unifiers

Consider the following logical expressions

$$R(x, f(a, g(y)))$$

and

$$R(b, f(z, w))$$

Some possible unifiers:

- $\sigma_1 = \{b/x, a/z, g(c)/w, c/y\}$
- $\sigma_2 = \{b/x, a/z, f(a)/y, g(f(a))/w\}$
- $\sigma_3 = \{b/x, a/z, g(y)/w\}$ (mgu)

Note that:

- $\sigma_1 = \sigma_3 \circ \{c/y\}$
- $\sigma_2 = \sigma_3 \circ \{f(a)/y\}$

Resolution in Predicate Logic

- Consider: $\{C_1 = P(x) \vee Q(x), C_2 = \neg P(f(y)) \vee R(y)\}$;
  $P(x)$ and $P(f(y))$ are *not* complementary, but they are unifiable, for example $\sigma = \{f(a)/x, a/y\}$
- Result: $C_1\sigma = P(f(a)) \vee Q(f(a))$
  en

  $$C_2\sigma = \neg P(f(a)) \vee R(a)$$

  $P(f(a))$ en $\neg P(f(a))$ are complementary

  $$\{C_1\sigma, C_2\sigma\} \vdash_{\mathcal{R}} Q(f(a)) \vee R(a)$$

- Using the mgu $\theta = \{f(y)/x\}$

  $$\{C_1\theta, C_2\theta\} \vdash_{\mathcal{R}} Q(f(y)) \vee R(y)$$

Resolution Rule for First-order Logic

- Notation: if L is a literal, dan $[L]$ is the atom
- Given the following two clauses $C_1 = C_1' \vee L_1$ and $C_2 = C_2' \vee L_2$, with L_1 an atom, and L_2 negated
- Suppose $[L_1]\sigma = [L_2]\sigma$, with σ an mgu
- Binary resolution rule $\mathcal{B}$ for predicate logic:

$$\frac{(C_1' \vee L_1)\sigma,\ (C_2' \vee L_2)\sigma}{C_1'\sigma \vee C_2'\sigma}$$

$C_1'\sigma \vee C_2'\sigma$ is binary resolvent, and

$$\{C_1, C_2\} \vdash_{\mathcal{B}} C_1'\sigma \vee C_2'\sigma$$

Resolution: Summary

In summary, this is what occurs,

- Find two clauses containing the same predicate, where such predicate is negated in one clause but not in the other
- Perform a unification on the two complementary predicates
  - If the unification fails, you might have made a bad choice of predicates
    Go back to the previous step and try again
- Discard the unified predicates, and combine the remaining ones from the two clauses into a new clause, also joined by the or-operator

Schubert's Steamroller

Wolves, foxes, birds, caterpillars, and snails are animals, and there are some of each of them

Also there are some grains, and grains are plants

Every animal either likes to eat all plants or all animals much smaller than itself that like to eat some plants

Caterpillars and snails are much smaller than birds, which are much smaller than foxes, which are in turn much smaller than wolves

Wolves do not like to eat foxes or grains, while birds like to eat caterpillars but not snails

Caterpillars and snails like to eat some plants

Prove there is an animal that likes to eat a grain-eating animal

Representation

- Wolfs are animals: $\forall x(\mathit{Wolf}(x) \rightarrow \mathit{Animal}(x))$
- There are wolfs: $\exists x \mathit{Wolf}(x)$
- Every animal either likes to eat all plants or all animals much smaller than itself that like to eat some plants

$$\begin{aligned}\forall x(\mathit{Animal}(x) \rightarrow \quad & (\forall y(\mathit{Plant}(y) \rightarrow \mathit{Eats}(x,y))) \\ & \vee(\forall z(\mathit{Animal}(z) \wedge \mathit{Smaller}(z,x) \\ & \qquad \wedge(\exists u(\mathit{Plant}(u) \wedge \mathit{Eats}(z,u))) \\ & \quad \rightarrow \mathit{Eats}(x,z))))\end{aligned}$$

- Caterpillars are smaller than birds

$$\forall x \forall y(\mathit{Caterpillar}(x) \wedge \mathit{Bird}(y) \rightarrow \mathit{Smaller}(x,y))$$

- etc.

Applying an ARP (Prover9)

```
============================== PROOF =================================

% Proof 1 at 0.02 (+ 0.00) seconds.
% Length of proof is 100.
% Level of proof is 47.
% Maximum clause weight is 20.
% Given clauses 229.

...

25 -Wolf(x) | animal(x).  [clausify(1)].
26 -Fox(x) | animal(x).  [clausify(2)].
27 -Bird(x) | animal(x).  [clausify(3)].
29 -Snail(x) | animal(x).  [clausify(5)].
30 -Grain(x) | plant(x).  [clausify(6)].
31 Wolf(c1).  [clausify(7)].
32 Fox(c2).  [clausify(8)].
33 Bird(c3).  [clausify(9)].
```

Continuation ...

```
282 -animal(c3) | eats(c3,f3(c2,c3)) | -animal(c5)
   | eats(c3,c5).  [resolve(278,a,99,b)].
283 -animal(c3) | eats(c3,f3(c2,c3)) | eats(c3,c5).
   [resolve(282,c,56,a)].
284 eats(c3,f3(c2,c3)) | eats(c3,c5).  [resolve(283,a,54,a)].
287 eats(c3,c5) | eats(c1,c6) | eats(c1,c2) | -animal(c2)
   | -animal(c3).  [resolve(284,a,224,e)].
297 eats(c3,c5) | eats(c1,c6) | eats(c1,c2) | -animal(c2).
  [resolve(287,e,54,a)].
298 eats(c3,c5) | eats(c1,c6) | eats(c1,c2).  [resolve(297,d,53,a)].
302 eats(c1,c6) | eats(c1,c2) | -Bird(c3) | -Snail(c5).
  [resolve(298,a,49,c)].
305 eats(c1,c6) | eats(c1,c2) | -Bird(c3).  [resolve(302,d,35,a)].
306 eats(c1,c6) | eats(c1,c2).  [resolve(305,c,33,a)].
310 eats(c1,c2) | -Wolf(c1) | -Grain(c6).  [resolve(306,a,48,c)].
313 eats(c1,c2) | -Grain(c6).  [resolve(310,b,31,a)].
314 eats(c1,c2).  [resolve(313,b,36,a)].
319 -Wolf(c1) | -Fox(c2).  [resolve(314,a,47,c)].
321 -Fox(c2).  [resolve(319,a,31,a)].
322 $F.  [resolve(321,a,32,a)].
============================== end of proof ==========================
```