$$C : Human(x) \leftarrow Human(father(x))$$

$$D : Human(y) \leftarrow Human(father(father(y)))$$

With a little thought (let us not get too entangled in the species problem here), you should be able to convince yourself that $C \models D$. But you will find it impossible to find a substitution θ that will make $C\theta \subseteq D$. What makes the difference to the propositional case? The difference between implication and subsumption in first-order logic arises because of self-recursive clauses of the kind shown: a short, but influential paper by Georg Gottlob shows that it is indeed only the self-recursive case that results in the difference.

1.4.7 Subsumption Lattice over Atoms

The subsumption relation is an example of a quasi-order. Let us take the simple case of definite clauses with a single literal (that is, atoms). Consider the set $\mathcal{A}$ of all atoms in some language, and $\mathcal{A}^+ = \mathcal{A} \cup \{\top, \bot\}$. Let the binary relation $\succeq$ be such that:

- $\top \succeq \mathbf{l}$ for all $\mathbf{l} \in \mathcal{A}^+$
- $\mathbf{l} \succeq \bot$ for all $\mathbf{l} \in \mathcal{A}^+$
- $\mathbf{l} \succeq m$ iff there is a substitution θ such that $\mathbf{l}\theta = m$, for $\mathbf{l}, \mathbf{m} \in \mathcal{A}$

We will represent a list of elements $e_1, \ldots, e_n$ as the(as the language Prolog does) by $[e_1, \ldots, e_n]$, and let $\mathbf{l} = Mem(x, [x, y])$ and $\mathbf{m} = Mem(1, [1, 2])$ then $\mathbf{l} \succeq \mathbf{m}$ with $\theta = \{x/1, y/2\}$. It is easy to see that $\succeq$ is a quasi-order over $\mathcal{A}^+$: clearly $\mathbf{l} \succeq \mathbf{l}$, with the empty substitution $\theta = \emptyset$ (that is, $\succeq$ is reflexive). Now, let $\mathbf{l} \succeq \mathbf{m}$ and $\mathbf{m} \succeq \mathbf{l}$. That is, there are some substitutions θ_1 and θ_2 such that $\mathbf{l}\theta_1 = \mathbf{m}$ and $\mathbf{m}\theta_2 = \mathbf{l}$. That is, $(\mathbf{l}\theta_1) \circ \theta_2 = n$. With $\theta = \theta_1 \circ \theta_2$ it follows that $\mathbf{l} \succeq \mathbf{l}$.

Since $\succeq$ is a quasi-order, we know a partial ordering must result from the partition of $\mathcal{A}^+$ into a set of equivalence classes $\mathcal{A}^+_E$. In fact, the partitions are $\{[\top]\}, \{[\bot]\}, X_1, \ldots$ where $[\mathbf{l}]$ denotes all atoms that are alphabetic variants[28] of $\mathbf{l}$. That is, if $\mathbf{l}, \mathbf{m} \in X_i$ then there are substitutions μ and σ s.t. $\mathbf{l}\mu = \mathbf{m}$ and $\mathbf{m}\sigma = \mathbf{l}$. That is, $\succeq$ is a partial ordering over the set of equivalence classes of atoms $(\mathcal{A}^+_E)$. $(Mem(x_1, [x_1, y_1]), Mem(x_2, [x_2, y_2]) \ldots$ are examples of members of an equivalence class.)

Recall that the difference between subsumption and implication in first-order logic arose with the appearance of self-recursive clauses. Since there is no possibility of this with atoms in first-order logic, subsumption and implication are equivalent, and we can see that logical implication (*models*) over atoms is also a quasi-order over atoms.

[28] Two atoms are subsume-equivalent iff they are variants. This is not true for clauses in general.

As soon as we have a quasi-order, we can effectively construct a partial-order over equivalence classes. So, the quasi-order of subsumption over atoms results in a partial order over equivalence classes of atoms. In fact, $\mathcal{A}_E^+$ is a lattice with the binary operations $\sqcap$ and $\sqcup$ defined on elements of $\mathcal{A}_E^+$ as follows (here, we have used $[\cdot]$ to represent an equivalence class):

- $[\bot] \sqcap [\mathbf{l}] = [\bot]$, and $[\top] \sqcap [\mathbf{l}] = [\mathbf{l}]$
- If $\mathbf{l}_1, \mathbf{l}_2 \in \mathcal{A}$ have a most general unifier (see page 78) θ then $[\mathbf{l}_1] \sqcap [\mathbf{l}_2] = [\mathbf{l}_1\theta] = [\mathbf{l}_2\theta]$.

  This can be proved as follows. Let $[\mathbf{u}] \in \mathcal{A}_E^+$ such that $[\mathbf{l}_1] \succeq [u]$ and $[\mathbf{l}_2] \succeq [\mathbf{u}]$, then we need to show that $[\mathbf{l}_1\theta] \succeq [\mathbf{u}]$. If $[\mathbf{u}] = [\bot]$, this is obvious. If $[\mathbf{u}]$ is conventional, then there are substitutions σ_1 and σ_2 such that $[\mathbf{l}_1\sigma_1] = [\mathbf{u}] = [\mathbf{l}_2\sigma 2]$. Here we can assume σ_1 only acts on variables in $\mathbf{l}_1$, and σ_2 only acts on variables in $\mathbf{l}_2$. Let $\sigma = \sigma_1 \cup \sigma_2$. Notice that σ is a unifier for $\{[\mathbf{l}_1], [\mathbf{l}_2]\}$. Since θ is an mgu for $\{[\mathbf{l}_1\sigma_1], [\mathbf{l}_2\sigma_2]\}$, there is a γ such that $\theta\gamma = \sigma$. Now $[\mathbf{l}_1\theta\gamma] = [\mathbf{l}_1\sigma] = [\mathbf{l}_1\sigma_1] = [\mathbf{u}]$, so $[\mathbf{l}_1\theta] \succeq [\mathbf{u}]$.
- If $\mathbf{l}_1, \mathbf{l}_2 \in \mathcal{A}$ do not have a most general unifier θ then $[\mathbf{l}_1] \sqcap [\mathbf{l}_2] = [\bot]$.

  Since $\mathbf{l}_1$ and $\mathbf{l}_2$ are not unifiable, there is no conventional atom $\mathbf{u}$ such that $[\mathbf{l}_1] \succeq [\mathbf{u}]$ and $[\mathbf{l}_2] \succeq [\mathbf{u}]$. Hence $[\mathbf{l}_1] \sqcap [\mathbf{l}_2] = [\bot]$.
- $[\bot] \sqcup [\mathbf{l}] = [\mathbf{l}]$, and $[\top] \sqcup [\mathbf{l}] = [\top]$
- If $\mathbf{l}_1$ and $\mathbf{l}_2$ have an "anti-unifier" $\mathbf{m}$ then $[\mathbf{l}_1] \sqcup [\mathbf{l}_2] = [\mathbf{m}]$; otherwise $[\mathbf{l}_1] \sqcup [\mathbf{l}_2] = [\top]$

Henceforth, we will drop the square backets $[\cdot]$ to denote equivalence classes and will instead implicitly assume their presence. The "anti-unifier" in the join operation is not something we have come across before, and needs some explanation. To get started, let us look at the atom $Mem(1, [1, 2])$. The list $[1, 2]$ written out in long-hand is really a term composed of the constants 1, 2 and the empty list, which we will denote by the constant nil. That is, $[1, 2]$ is really the term $list(1, list(2, list(nil)))$, where $list$ is a function, and $Mem(1, [1, 2])$ is really $Mem(1, list(1, list(2, nil)))$. Now, we can devise a "term-place" notation to identify the occurrence of each term in any atom. In $Mem(1, list(1, list(2, nil)))$, the 1 is a term that occurs in two places: in the first argument (or "place") of Mem, and as the first argument of the second place of Mem. We can denote these two occurrences as $(1, \langle 1\rangle)$ and $(1, \langle 2, 1\rangle)$. Similarly, we can encode the occurrences of other terms: $(2, \langle 2, 2, 1\rangle)$ and $(nil, \langle 2, 2, 2\rangle)$.

You should convince yourself that the occurrence of every term t in an atom can indeed be represented by the pair (t, p), where p is a sequence of places. We now have all we need to be able to describe the anti-unification algorithm for a pair of literals with the same predicate symbol (adapted from Plotkin, 1970):

Input: A pair of atoms $\mathbf{l}_1$ and $\mathbf{l}_2$ with the same predicate symbol

Output: $\mathbf{l}_1 \sqcup \mathbf{l}_2$

1. Let $\mathbf{l} = \mathbf{l}_1$ and $\mathbf{m} = \mathbf{l}_2$, $\theta = \emptyset$, $\sigma = \emptyset$
2. If $\mathbf{l} = \mathbf{m}$ return $\mathbf{l}$ and stop.
3. Try to find terms t_1 and t_2 that have the same (leftmost) place in $\mathbf{l}$ and $\mathbf{m}$ respectively, such that $t_1 \neq t_2$ and either t_1 and t_2 begin with different function symbols, or at least one of them is a variable.
4. If there is no such t_1, t_2, return $\mathbf{l}$ and stop.
5. Choose a variable x that does not occur in either $\mathbf{l}$ or $\mathbf{m}$ and wherever t_1 and t_2 occur in the same place in $\mathbf{l}$ and $\mathbf{m}$, replace each of them by x
6. Set θ to $\theta \cup \{x/t_1\}$ and σ to $\sigma \cup \{x/t_2\}$
7. Go to Step 3

The Table 1.1 shows the progressive construction of lubs starting with terms and culminating in literals.

Lub	Definition	Examples
lub of terms $lub(t_1, t_2)$	1. $lub(t, t) = t$, 2. $lub(f(s1, \ldots, s_n), f(t_1, \ldots, t_n)) = f(lub(s_1, t_1), \ldots, lub(s_n, t_n))$, 3. $lub(f(s_1, \ldots, s_m), g(t_1, \ldots, t_n)) = V$, where $f \neq g$, and V is a variable which represents $lub(f(s_1, \ldots, s_m), g(t_1, \ldots, t_n))$, 4. $lub(s, t) = V$, where $s \neq t$ and at least one of s and t is a variable; in this case, V is a variable which represents $lub(s, t)$.	• $lub([a, b, c], [a, c, d]) = [a, X, Y]$. • $lub(f(a, a), f(b, b)) = f(lub(a, b), lub(a, b)) = f(V, V)$ where V stands for $lub(a, b)$. • When computing lggs one must be careful to use the same variable for multiple occurrences of the lubs of subterms, *i.e.*, $lub(a, b)$ in this example. This holds for lubs of terms, atoms and clauses alike.
lub of atoms $lub(\mathbf{a}_1, \mathbf{a}_2)$	1. $lub(P(s_1, \ldots, s_n), P(t_1, \ldots, t_n)) = P(lub(s_1, t_1), \ldots, lub(s_n, t_n))$, if atoms have the same predicate symbol P, 2. $lub(P(s_1, \ldots, s_m), Q(t_1, \ldots, t_n))$ is undefined if $P \neq Q$.	
lub of literals $lub(\mathbf{l}_1, \mathbf{l}_2)$	1. if $\mathbf{l}_1$ and $\mathbf{l}_2$ are atoms, then $lub(\mathbf{l}_1, \mathbf{l}_2)$ is computed as defined above, 2. if both $\mathbf{l}_1$ and $\mathbf{l}_2$ are negative literals, $\mathbf{l}_1 = \overline{\mathbf{a}_1}$, $\mathbf{l}_2 = \overline{\mathbf{a}_2}$, then $lub(\mathbf{l}_1, \mathbf{l}_2) = lub(\overline{\mathbf{a}_1}, \overline{\mathbf{a}_2}) = \overline{lub(\mathbf{a}_1, \mathbf{a}_2)}$, 3. if $\mathbf{l}_1$ is a positive and $\mathbf{l}_2$ is a negative literal, or vice versa, $lub(\mathbf{l}_1, \mathbf{l}_2)$ is undefined.	• $lub(Parent(ann, mary), Parent(ann, tom)) = Parent(ann, X)$. • $lub(Parent(ann, mary), \overline{Parent(ann, tom)}) = undefined$. • $lub(Parent(ann, X), Daughter(mary, ann)) = undefined$.

Table 1.1: Table showing progressive definitions of lubs, starting with terms and culminating in literalss.

Let us look at an example of constructing the anti-unifier of $Mem(1, [1, 2])$ and $Mem(2, [2, 4]$. That is, $\mathbf{l}_1 = Mem(1, list(1, list(1, list(2, nil))))$ and $\mathbf{l}_2 = Mem(2, list(2, list(2, list(4, nil))))$. You should be able to work through the

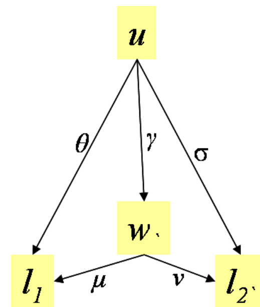

Figure 1.13: Illustration of the proof of theorem 22.

steps of the algorithm to find that it terminates with $\mathbf{l} = \mathbf{m} = Mem(x, list(x, list(y, nil)))$ with $\theta = \{x/1, y/2\}$ and $\sigma = \{x/2, y/4\}$.

But is the procedure correct? That is, does it really return a lub of a pair of atoms $\mathbf{l}_1$ and $\mathbf{l}_2$? Suppose the procedure returned an atom $\mathbf{l}$, and let $\theta = \{x_1/s_1, \ldots, x_k/s_k\}$ and $\sigma = \{x_1/t_1, \ldots, x_k/t_k\}$. That is $\mathbf{l}\theta = \mathbf{l}_1$ and $\mathbf{l}\sigma = \mathbf{l}_2$. Now suppose there is some other atom $\mathbf{l}'$ such that $\mathbf{l}' \succeq \mathbf{l}_1$ and $\mathbf{l}' \succeq \mathbf{l}_2$. Then, we have to show that $\mathbf{l}' \succeq \mathbf{l}$ for any such $\mathbf{l}'$.

The proof of this is a bit laborious and will dealt with subsequently. The truth of the next lemma is easy to see:

Theorem 21 *After each iteration of the Anti-Unification Algorithm, there are terms $s_1, \ldots, s_i$ and $t_1, \ldots, t_i$ such that:*

1. *$\theta = \{z_1/s_1, \ldots, z_i/s_i\}$ and $\sigma = \{z_1/t_1, \ldots, z_i/t_i\}$.*
2. *$\mathbf{l}\theta = \mathbf{l}_1$ and $\mathbf{m}\sigma = \mathbf{l}_2$.*
3. *For every $1 \leq j \leq i$, s_j and t_j differ in their first symbol.*
4. *There are no $1 \leq j, k \leq i$ such that $j \neq k$, $s_j = s_k$ and $t_j = t_k$.*

Theorem 22 *Let $\mathbf{l}_1$ and $\mathbf{l}_2$ be two atoms with the same predicate symbol. Then the Anti- Unification Algorithm with $\mathbf{l}_1$ and $\mathbf{l}_2$ as inputs returns $\mathbf{l}_1 \sqcup \mathbf{l}_2$.*

Proof: It is easy to see that the algorithm terminates after a finite number of steps, for any $\mathbf{l}_1$, $\mathbf{l}_2$. Let $\mathbf{u}$ be the atom that the algorithm returns, and let $\theta = \{z_1/s_1, \ldots, z_i/s_i\}$ and $\sigma = \{z_1/t_1, \ldots, z_i/t_i\}$ be the final values of θ and σ in the computation of $\mathbf{u}$ (so $\mathbf{u}$ equals the final values of $\mathbf{l}$ and $\mathbf{m}$ in the execution of the algorithm). Then $\mathbf{u}\theta = \mathbf{l}_1$ and $\mathbf{u}\sigma = \mathbf{l}_2$ by Theorem 21, part 2. Suppose $\mathbf{v}$ is an atom such that $\mathbf{v} \succeq \mathbf{l}_1$ and $\mathbf{v} \succeq \mathbf{l}_2$. In order to show that $\mathbf{u} = \mathbf{l}_1 \sqcup \mathbf{l}_2$, we have to prove $\mathbf{v} \succeq \mathbf{u}$.

Let $\mathbf{w} = \mathbf{u} \sqcap \mathbf{v}$, which exists by the proof on page 82. Then $\mathbf{u} \succeq \mathbf{w}$ and $\mathbf{v} \succeq \mathbf{w}$. Since $\mathbf{w} = \mathbf{u} \sqcap \mathbf{v}$, $\mathbf{u} \succeq \mathbf{l}_1$ and $\mathbf{v} \succeq \mathbf{l}_1$, we must have $\mathbf{w} \succeq \mathbf{l}_1$. Similarly

$\mathbf{w} \succeq \mathbf{l}_2$. Thus there are substitutions γ, μ, ν, such that $\mathbf{u}\gamma = \mathbf{w}$, $\mathbf{l}_1 = \mathbf{w}\mu = \mathbf{u}\gamma\mu$ and $\mathbf{l}_2 = \mathbf{w}\nu = \mathbf{u}\gamma\nu$. Then $\mathbf{u}\theta = \mathbf{l}_1 = \mathbf{u}\gamma\mu$ and $\mathbf{u}\sigma = \mathbf{l}_2 = \mathbf{u}\gamma\nu$ (see Figure 1.13 for illustration). Hence, if x is a variable occurring in $\mathbf{u}$, then $x\theta = x\gamma\mu$ and $x\sigma = x\gamma\nu$.

We will now show that $\mathbf{u}$ and $\mathbf{w} = \mathbf{u}\gamma$ are variants, by showing that γ is a renaming substitution for $\mathbf{u}$. Suppose it is not. Then γ maps some variable x in $\mathbf{u}$ to a term that is not a variable, or γ unifies two distinct variables x, y in $\mathbf{u}$.

Suppose x is a variable in $\mathbf{u}$, such that $x\gamma = t$, where t is a term that is not a variable. If x is not one of the z_j's, then $x\gamma\mu = x\theta = x$, contradicting the assumption that $x\gamma = t$ is not a variable. But on the other hand, if x equals some z_j, then $t\mu = x\gamma\mu = x\theta = s_j$ and $t\nu = x\gamma\nu = x\sigma = t_j$. Then s_j and t_j would both start with the first symbol of t, contradicting theorem 21, part 3. So this case leads to a contradiction.

Suppose x and y are distinct variables in $\mathbf{u}$ such that γ unifies x and y. Then,

1. If neither x nor y is one of the z_j's, then $x\gamma\mu = x\theta = x \neq y = y\theta = y\gamma\mu$, contradicting $x\gamma = y\gamma$

2. If x equals some z_j and y does not, then $x\gamma\mu = x\theta = s_j$ and $x\gamma\nu = x\sigma = t_j$, so $x\gamma\mu \neq x\gamma\nu$ by theorem 21, part 3. But $y\gamma\mu = y\theta = y = y\sigma = y\gamma\nu$, contradicting $x\gamma = y\gamma$.

3. Similarly for the case where y equals some z_j and x does not.

4. If $x = z_j$ and $y = z_k$, then $j \neq k$, since $x \neq y$. Furthermore, $s_j = x\theta = x\gamma\mu = y\gamma\mu = y\theta = s_k$ and $t_j = x\sigma = x\gamma\nu = y\gamma\nu = y\sigma = t_k$. But this contradicts theorem 21, part 4.

Thus, the assumption that γ unifies two variables in $\mathbf{u}$ also leads to a contradiction. Thus γ is a renaming substitution for $\mathbf{u}$, and hence $\mathbf{u}$ and $\mathbf{w}$ are variants. Finally, since $\mathbf{v} \succeq \mathbf{w}$, we have $\mathbf{v} \succeq \mathbf{u}$. $\Box$

It is however a more straightforward matter to see the following:

Theorem 23 *If $\mathcal{A}_E^+$ is a set of equivalance classes of atoms $\mathcal{A}$ augmented by the two elements $\top$ and $\bot$, then for any pair of elements $\mathbf{l}_1, \mathbf{l}_2 \in \mathcal{A}_E^+$, $\mathbf{l}_1 \sqcup \mathbf{l}_2$ always exists.*

Proof: The possibilities for each of $\mathbf{l}_1$ and $\mathbf{l}_2$ are that they are either: (1) some variant of $\top$; (2) some variant of $\bot$; or (3) an atom from S. It can be verified that $\mathbf{l}_1 \sqcap \mathbf{l}_2$ is defined in all 9 cases.

1. If $\mathbf{l}_1 = \top$ or $\mathbf{l}_2 = \top$, then $\mathbf{l}_1 \sqcup \mathbf{l}_2 = \top$. If $\mathbf{l}_1 = \bot$, then $\mathbf{l}_1 \sqcup \mathbf{l}_2 = \mathbf{l}_2$. If $\mathbf{l}_2 = \bot$, then $\mathbf{l}_1 \sqcup \mathbf{l}_2 = \mathbf{l}_1$.

2. If $\mathbf{l}_1$ and $\mathbf{l}_2$ are conventional atoms with the same predicate symbol, $\mathbf{l}_1 \sqcup \mathbf{l}_2$ is given by the Anti-Unification Algorithm.