Logic and Proof Exercises

Thomas Forster

February 11, 2018

Contents

0.1 Two Definitions

Definition 1 Input Resolution
One of the resolving clauses should always be from the input (i.e. from the knowledge base or the negated query).

Complete for Horn clauses but not in general. (In fact we will see below two examples where Input Resolution doesn't deliver the goods, namely exercises 4 and 2)

Definition 2 Linear Resolution
A generalization of Input Resolution. Allows resolutions of clauses P and Q if P is in the input or is an ancestor of Q in the proof tree.

I think it's complete. I inadvertently fibbed to some of you earlier: the definition i gave of linear resolution was the definition of *input* resolution. I told you (correctly) that Linear Resolution is complete, but i gave you the wrong definition, so that the thing i gave you was not complete.

Chapter 1

tf's Exercises

1.1 Completeness of Propositional Logic

Exercise 1 *Using the sequent rules in Prof Paulson's notes, show that every truth-table tautology A "has a sequent proof" in the sense that there is a proof of the sequent* $\vdash \phi$

1.2 Excluded Middle

Excluded middle is the principle $A \vee \neg A$ that says there are only two truth-values. One might expect it to be equivalent to

$$\neg(\neg(A \longleftrightarrow B) \wedge \neg(B \longleftrightarrow C) \wedge \neg(A \longleftrightarrow C)) \qquad (\textit{Tertium non datur} \text{ (variant form)})$$

but actually this variant form is weaker, and is constructively correct.

Exercise 2
Provide sequent, natural deduction and resolution proofs of the above variant form of Tertium not datur. *Establish that there is no input resolution proof of it.*

There are extra points for supplying proofs that respect the constraints of constructive logic, so in particular your sequent proof should only ever have one formula on the right.

(I have a constructive sequent proof of the variant form of *Tertium non datur* but it is in WORD and the turnstile doesn't print. The usual bribes are available for a nice LaTeX-ed proof.)

1.3 Wellfounded Relations

Exercise 3
Provide demonstrations, using sequent calculus, natural deduction or resolution that

$$\forall x(\forall y(R(y,x) \to \phi(y)) \to \phi(x)) \to \forall z(\phi(z)) \textit{ and } (\forall xy)(R'(x,y) \to R(x,y))$$

together imply

$$\forall x(\forall y(R'(y,x) \to \phi(y)) \to \phi(x)) \to \forall z(\phi(z)).$$

Maria Gorinova of Clare has kindly provided two model answers: a sequent calculus proof, and a resolution proof. Daniel Spencer (also of Clare) has provided a natural deduction proof.

1.4 Graphs

A graph is a set of vertices with undirected edges, and no loops at vertices. It is *connected* if one can get from any vertex to any other vertex by following edges. The *complement* of a graph is what you think it is: it has the same vertices, but a complementary set of edges.

Exercise 4
Use propositional resolution to show that a graph and its complement cannot both be disconnected.

Your first thought might be to try to do it in first-order logic, but that is actually quite hard—and i haven't managed to get it to come out yet. Doing it in propositional logic requires some preliminary thought. I have a proof, but i haven't managed to find a proof using linear resolution—tho' i know there must be one. My proof is not compliant with the restrictions surrounding *Input Resolution*. In fact it's not hard to show that there is no proof thus compliant:

Exercise 5
Show that there is no input resolution proof for exercise 4.

It ought to be possible to do Exercise 4 in first-order logic as well as in propositional logic, but i haven't managed to get a model answer. (This possibility came up beco's Nick Spooner (top of 1a in 2011) misread the then current version of exercise 4 as an instruction to find a resolution proof in first-order logic!) In this setting you have a binary relation $E(x,y)$ to say there is an edge between x and y, a binary relation $P(x,y)$ to say there is a path between x and y in G, and a binary relation $Q(x,y)$ to say there is a path between x and y in $\overline{G}$.

Exercise 6
Using the above vocabulary write down an expression of first-order logic that says that a graph and its complement cannot both be disconnected.

The exercise should continue "Then prove it by any method of your choice: natural deduction, sequent calculus or resolution." However, as of Lent term 2016/7 i haven't got a resolution proof (which annoys me, beco's i should) and i haven't attempted the other two—yet. They look a bit hard to do by hand, tho' with a proof assistant it should be fairly easy.

1.5 Skolemisation

The two formulæ in this section are related to the fact that Skolemisation is in some sense logically conservative. The reader might like to look long and hard at the formulæ below with a view to seeing what they mean.

Exercise 7
Prove the two following formulæ by resolution or sequent calculus.

1.

$$(\forall x \exists y)R(x,y) \to (\forall x \exists y)(\forall x' \exists y')(R(x,y) \land R(x',y') \land (x = x' \to y = y'))$$

2.

$$(\forall x \exists y) R(x, y) \to (\forall x \exists y)(\forall x' \exists y')(\forall x'' \exists y'') \bigwedge \left(\begin{array}{l} R(x, y) \\ R(x', y') \\ R(x'', y'') \\ x = x' \to y = y' \\ x = x'' \to y = y'' \\ x' = x'' \to y' = y' \end{array} \right) \tag{1.1}$$

In the proof by resolution you will need resolution rules for functions and equality, such as

$$\{\neg(x = y), f(x) = f(y)\}$$

I have sequent proofs but—annoyingly—no resolution proofs at this stage.

1.6 Berkeley's Master Argument for Idealism

The following text is a celebrated argument by Bishop Berkeley which purports to show that nothing exists unconceived. It's a fairly delicate exercise in formalisation.

> HYLAS : What more easy than to conceive of a tree or house existing by itself, independent of, and unperceived by any mind whatsoever. I do at present time conceive them existing after this manner.
>
> PHILONOUS : How say you, Hylas, can you see a thing that is at the same time unseen?
>
> HYLAS : No, that were a contradiction.
>
> PHILONOUS : Is it not as great a contradiction to talk of *conceiving* a thing which is *unconceived*?
>
> HYLAS : It is
>
> PHILONOUS : This tree or house therefore, which you think of, is conceived by you?
>
> HYLAS : How should it be otherwise?
>
> PHILONOUS : And what is conceived is surely in the mind?
>
> HYLAS : Without question, that which is conceived exists in the mind.
>
> PHILONOUS How then came you to say, you conceived a house or a tree existing independent and out of all mind whatever?
>
> HYLAS That was I own an oversight ...

The exercise here is to formalise the above conversation and construct a natural deduction proof that everything is conceived (as Berkeley wants) and perhaps even a sequent calculus proof. This has been discussed in print by my friend Graham Priest, and this treatment draws heavily on his.

You may, if you wish to think through this exercise very hard, try to work out what new syntactic gadgets one needs to formalise this argument, but i don't recommend it. The best thing is to use the gadgetry Priest introduced.

Priest starts off by distinguishing, very properly, between **conceiving objects** and **conceiving propositions**. Accordingly in his formalisation he will have *two* devices. One is a sentence operator T which is syntactically a modal operator and a predicate τ whose intended interpretation is that $\tau(x)$ iff x is conceived. $T\phi$ means that the proposition ϕ is being entertained. (*By whom* is good question: is the point of the argument that for every object there is someone who conceives it? or that everybody thinks about every object?)

At this point you could, if you like, work out your own natural deduction rules. Here are the rules Priest came up with.

$$\frac{\phi \to \psi}{T(\phi) \to T(\psi)}$$

which says something to the effect that T distributes over conditionals. Priest calls this "affixing". The other rule is one that tells us that if we conceive an object to have some property ϕ then we conceive it.

$$\frac{T(\phi(x))}{\tau(x)}$$

Let us call it the **mixed rule**.

Now it's your turn to do some work.

Exercise 8

1. *Devise a natural deduction proof of $(\forall x)(\tau(x))$, or of $(\forall x)((\tau(x) \to \bot) \to \bot)$. You are allowed to use the undischarged premiss Tp where p is an arbitrary propositional letter. You may wish to use a natural deduction version of the law of excluded middle. My model answer doesn't, and accordingly i prove only that $(\forall x)((\tau(x) \to \bot) \to \bot)$. You might try to prove $(\exists x)(\tau(x) \to \bot) \to \bot$ as well.*

   At this point you could, if you like, work out your own sequent calculus rules. Here are the rules i came up with.

   $$\frac{\Gamma, A \vdash \Delta, B}{\Gamma, TA \vdash \Delta, TB}$$

   and

   $$\frac{\Gamma \vdash \Delta, T(\phi(x))}{\Gamma \vdash \Delta, \tau(x)}$$

2. *Prove the sequent $Tp \vdash (\forall x)(\tau(x))$*

3. *Prove that a premiss of the form Tp really is needed.*

1.7 A Special Kind of Poset

Exercise 9

(i) What is an antichain in a poset?

(ii) Show that the theory of posets in which every element belongs to a unique maximal antichain can be axiomatised by adding to the theory of posets either of:

A: $(\forall xyz)(z > x \not\leq y \not\leq x \to z < y)$; or

B: $(\forall xyz)(z > x \not\leq y \not\leq x \to z > y)$.

(iii) Show that A and B are equivalent, by natural deduction, sequent calculus or resolution.

I have no sample answers at this stage. The usual bribes are available.

This is of some interest, since "posets in which every element belongs to a unique maximal antichain" is *prima facie* second order (it appears to be Σ^2_2, with two second order quantifiers) but it is in fact first order. The question of whether or not a given *prima facie* second-order theory is in fact first order is presumably unsolvable.

Chapter 2

Answers to tf's Exercises

2.1 Exercise 1 (Completeness of Propositional Calculus)

We take a sequent $\Gamma \vdash \Delta$ to be saying that every valuation that satisfies every formula in Γ also satisfies at least one formula in Δ. That way the sequent $\vdash \phi$ is the allegation that ϕ is (semantically) valid—a truth-table tautology. We say a sequent is *correct* iff the allegation it makes is correct.

We want to show that if ϕ is a truth-table tautology, then there is a proof of the sequent $\vdash \phi$. We will obtain this as a special case of a result that says that if a sequent $\Gamma \vdash \Delta$ is correct then there is a sequent proof of it.

The idea is that if $\Gamma \vdash \Delta$ is a correct sequent then it can be obtained by means of a sequent rule from another correct sequent $\Gamma' \vdash \Delta'$ which is simpler in some sense where 'simpler than' is a wellfounded relation. Clearly the relation we want is: $\Gamma' \vdash \Delta'$ is *one-step* simpler than $\Gamma \vdash \Delta$ if Γ' is a proper subset of Γ or is obtained from Γ by replacing a formula in Γ by one or more of its immediate subformulæ. Either that or $\Gamma' = \Gamma$ and Δ' is obtained similarly from Δ. 'Simpler' is now the transitive closure of one-step simpler. Observe that 'simpler' is wellfounded. Observe too that a sequent that cannot be simplified further contains atomic formulæ only. Any correct sequent that contains only atomic formuæ is an initial sequent and so has a sequent proof.

To obtain the result we need to show not only that the correctness of the upper sequent(s) in any instance of a sequent rule guarantees the correctness of the lower sequent (which is the whole point, after all) but also that in each [instance of a] rule, if the lower sequent is correct then all the upper sequents are correct too. Mostly this is pretty obvious, so let's try a hard one.

$$\frac{\Gamma \vdash \Delta, A \qquad \Gamma, B \vdash \Delta}{\Gamma, A \to B \vdash \Delta}$$

Suppose the lower sequent is correct. We want to ensure that the two upper sequents, $\Gamma \vdash \Delta, A$ and $\Gamma, B \vdash \Delta$ are correct too. Let's take them in turn

$\Gamma \vdash \Delta, A$:

> Let v be a valuation that satisfies Γ; does it also satisfy A? If it does, we are home and dry. If it doesn't, then it satisfies $\neg A$ and therefore $A \to B$—in which case the lower sequent will tell us that it satisfies something in Δ.

$\Gamma, B \vdash \Delta$:

> Let v be a valuation that satisfies Γ and B. If it satisfies B it will also satisfy $A \to B$; but then the correctness of the lower sequent tells us that it satisfies something in Δ.

This enables us to prove that every truth-table tautology has a sequent proof, as follows. Let ϕ be a truth-table tautology; we start with the sequent $\vdash \phi$ and apply the rules backwards until each branch reaches either a basic sequent or a sequent in which all formulæ are atomic. But then, because the backward-chaining preserves correctness, any such sequent must be correct. And any correct sequent in which all formulæ are atomic must be a basic sequent.

I suspect that we need to say something about confluence.

2.2 Exercise 3 on Wellfoundedness

Every exercise has a subtext. (How else do exercise-setters dream up exercises?) The subtext here is that wellfoundedness is what supports induction. If you discard ordered pairs from a wellfounded relation the result is still wellfounded—and still supports induction.

Resolution and sequent proofs supplied by Maria Gorinova of Clare (1B 2013/4); natural deduction proof supplied by Daniel Spencer of Clare (1B 2014/5).

2.2.1 A Resolution Proof from Maria Gorinova

Let
$A = \forall x(\forall y(R(y,x) \to \phi(y)) \to \phi(x)) \to \forall z(\phi(z))$,
$B = \forall xy(R'(x,y) \to R(x,y))$ and
$C = \forall x(\forall y(R'(y,x) \to \phi(y)) \to \phi(x)) \to \forall z(\phi(z))$.
We want to prove $A \wedge B \to C$.

The resolution proof starts by translating A, B, C and $\neg(A \wedge B \to C)$ into NNF, skolemizing, renaming some of the variables and removing the universal quantifiers.

$$\begin{aligned}\neg(A \wedge B \to C) &\simeq \neg(\neg(A \wedge B) \vee C)\\ &\simeq A \wedge B \wedge \neg C\end{aligned}$$

Thus, the set of clauses of $(\neg(A \wedge B \to C)$ is $cl(A) \cup cl(B) \cup cl(\neg C)$ (where $cl(E)$ is the set of clauses of the expression E).

$$\begin{aligned}
A &= \forall x(\forall y(R(y,x) \to \phi(y)) \to \phi(x)) \to \forall z(\phi(z)) && \\
&\simeq \neg\forall x(\neg\forall y(\neg R(y,x) \vee \phi(y)) \vee \phi(x)) \vee \forall z(\phi(z)) && (\to \text{ elim x3})\\
&\simeq \exists x\neg(\exists y\neg(\neg R(y,x) \vee \phi(y)) \vee \phi(x)) \vee \forall z(\phi(z)) && (\neg \text{ pull x2})\\
&\simeq \exists x(\neg\exists y(\neg\neg R(y,x) \wedge \neg\phi(y)) \wedge \neg\phi(x)) \vee \forall z(\phi(z)) && (\text{De Morgan x2})\\
&\simeq \exists x(\forall y\neg(R(y,x) \wedge \neg\phi(y)) \wedge \neg\phi(x)) \vee \forall z(\phi(z)) && (\neg \text{ pull})\\
&\simeq \exists x(\forall y(\neg R(y,x) \vee \neg\neg\phi(y)) \wedge \neg\phi(x)) \vee \forall z(\phi(z)) && (\text{De Morgan})\\
&\simeq \exists x(\forall y(\neg R(y,x) \vee \phi(y)) \wedge \neg\phi(x)) \vee \forall z(\phi(z)) && \\
&\simeq (\forall y(\neg R(y,a) \vee \phi(y)) \wedge \neg\phi(a)) \vee \forall z(\phi(z)) && (\text{skolemization})\\
&\simeq (\forall y(\neg R(y,a) \vee \phi(y)) \vee \forall z(\phi(z))) \wedge (\neg\phi(a) \vee \forall t(\phi(t)) && (\text{distributive, [t/z]})\\
&\simeq (\neg R(y,a) \vee \phi(y) \vee \phi(z)) \wedge (\neg\phi(a) \vee \phi(t)) && (\forall \text{ rem})
\end{aligned}$$

$\Rightarrow cl(A) = \{\{\neg R(y,a), \phi(y), \phi(z)\},\ \{\neg\phi(a), \phi(t)\}\}$

$$\begin{aligned}
B &= \forall xy(R'(x,y) \to R(x,y)) \\
&\simeq \forall xy(\neg R'(x,y) \vee R(x,y)) && (\to \text{ elim}) \\
&\simeq \forall uv(\neg R'(u,v) \vee R(u,v)) && (\text{[u/x, v/y]}) \\
&\simeq \neg R'(u,v) \vee R(u,v) && (\forall\text{rem})
\end{aligned}$$

$\Rightarrow cl(B) = \{\{\neg R'(u,v), R(u,v)\}\}$

$$\begin{aligned}
\neg C &\simeq \neg(\forall x(\forall y(R'(y,x) \to \phi(y)) \to \phi(x)) \to \forall z(\phi(z))) \\
&\simeq \forall x(\neg\forall y(\neg R'(y,x) \vee \phi(y)) \vee \phi(x)) \wedge \neg\forall z(\phi(z)) && (\to \text{ elim x3 and De Morgan}) \\
&\simeq \forall x(\exists y\neg(\neg R'(y,x) \vee \phi(y)) \vee \phi(x)) \wedge \exists z\neg(\phi(z)) && (\neg \text{ pull x2}) \\
&\simeq \forall x(\exists y(\neg\neg R'(y,x) \wedge \neg\phi(y)) \vee \phi(x)) \wedge \exists z\neg(\phi(z)) && (\text{De Morgan}) \\
&\simeq \forall x(\exists y(R'(y,x) \wedge \neg\phi(y)) \vee \phi(x)) \wedge \exists z\neg(\phi(z)) \\
&\simeq \forall x(R'(f(x),x) \wedge \neg\phi(f(x))) \vee \phi(x)) \wedge \neg\phi(b) && (\text{skolemization}) \\
&\simeq \forall x((R'(f(x),x) \vee \phi(x)) \wedge (\neg\phi(f(x)) \vee \phi(x))) \wedge \neg\phi(b) && (\text{distributive}) \\
&\simeq \forall x(R'(f(x),x) \vee \phi(x)) \wedge \forall w(\neg\phi(f(w)) \vee \phi(w)) \wedge \neg\phi(b) && (\text{distributive, [w/x]}) \\
&\simeq (R'(f(x),x) \vee \phi(x)) \wedge (\neg\phi(f(w)) \vee \phi(w)) \wedge \neg\phi(b) && (\forall \text{ rem})
\end{aligned}$$

$\Rightarrow cl(\neg C) = \{\{R'(f(x),x), \phi(x)\},\ \{\neg\phi(f(w)), \phi(w)\},\ \{\neg\phi(b)\}\}$

Therefore the clauses of $\neg(A \wedge B \to C)$ are $\{\neg R(y,a), \phi(y), \phi(z)\},\ \{\neg\phi(a), \phi(t)\},$ $\{\neg R'(u,v), R(u,v)\}, \{R'(f(x),x), \phi(x)\},\ \{\neg\phi(f(w)), \phi(w)\},\ \{\neg\phi(b)\}$ and now we can use the resolution rule on these to get the empty clause:

$$\begin{tikzcd}
\{\neg R(y,a),\phi(y),\phi(z)\} \arrow[d] & \{\neg R'(u,v),R(u,v)\} \arrow[dl, "{[y/u,a/v]}"] \\
\{\phi(y),\phi(z)\,\neg R'(y,a)\} \arrow[d, "{[f(a)/y]}"] & \{R'(f(x),x),\phi(x)\} \arrow[dl, "{[a/x]}"'] \\
\{\phi(f(a)),\phi(z),\phi(a)\} \arrow[d] & \{\neg\phi(f(w)),\phi(w)\} \arrow[dl, "{[a/w]}"'] \\
\{\phi(z),\phi(a)\} \arrow[d, "{[b/z]}"] & \{\neg\phi(a),\phi(t)\} \arrow[dl, "{[b/t]}"'] \\
\{\phi(b)\} \arrow[d] & \{\neg\phi(b)\} \arrow[dl] \\
\square &
\end{tikzcd}$$

□

2.2.2 A Sequent Proof from Daniel Spencer

Let

$$\begin{aligned}
XY &= \forall xy(R'(x,y)\to R(x,y)),\\
Y &= \forall y(R(y,x)\to\phi(y)),\\
Y' &= \forall y(R'(y,x)\to\phi(y))\ and\\
Y'_z &= \forall y(R'(y,z)\to\phi(y)).
\end{aligned}$$

We want to prove

$$(\forall x(Y\to\phi(x))\to\forall z(\phi(z)))\wedge XY\to(\forall x(Y'\to\phi(x))\to\forall z(\phi(z)))$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{R'(y,x),R'(y,z)\vdash\phi(x),\phi(y),\phi(z),R(y,x),R'(y,x)}\,(b)\qquad\dfrac{}{R(y,x),R'(y,x),R'(y,z)\vdash\phi(x),\phi(y),\phi(z),R(y,x)}\,(b)}{R'(y,x)\to R(y,x),R'(y,x),R'(y,z)\vdash\phi(x),\phi(y),\phi(z),R(y,x)}\,(\to l)}{\forall xy(R'(x,y)\to R(x,y)),R'(y,x),R'(y,z)\vdash\phi(x),\phi(y),\phi(z),R(y,x)}\,(2\times\forall l)}{XY,R'(y,x),R'(y,z)\vdash\phi(x),\phi(y),\phi(z),R(y,x)}\,(expand)}{(****)}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{(****)}{XY, R'(y,x), R'(y,z) \vdash \phi(x), \phi(y), \phi(z), R(y,x)} \qquad \dfrac{}{XY, \phi(y), R'(y,x), R'(y,z) \vdash \phi(x), \phi(y), \phi(z)}\,(b)}{XY, R(y,x) \to \phi(y), R'(y,x), R'(y,z) \vdash \phi(x), \phi(y), \phi(z)}\,(\to l)}{XY, \forall y(R(y,x) \to \phi(y)), R'(y,x), R'(y,z) \vdash \phi(x), \phi(y), \phi(z)}\,(\forall l)}{XY, Y, R'(y,x), R'(y,z) \vdash \phi(x), \phi(y), \phi(z)}\,(expand)}{XY, Y, \vdash \phi(x), \phi(z), R'(y,x) \to \phi(y), R'(y,z) \to \phi(y)}\,(2\times \to r)}{XY, Y, \vdash \phi(x), \phi(z), \forall y(R'(y,x) \to \phi(y)), \forall y(R'(y,z) \to \phi(y))}\,(2 \times \forall r)}{XY, Y, \vdash \phi(x), \phi(z), Y', Y'_z}\,(expand)}{(***)}$$

$$\dfrac{\dfrac{\dfrac{(***)}{XY, Y, \vdash \phi(x), \phi(z), Y', Y'_z} \qquad \dfrac{}{XY, Y, \phi(x), \phi(z) \vdash \phi(x), \phi(z)}\,(b)}{XY, Y, Y'_z \to \phi(z) \vdash \phi(x), \phi(z), Y'}\,(\to l)}{(**)}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{(**)}{XY, Y, Y'_z \to \phi(z) \vdash \phi(x), \phi(z), Y'} \qquad \dfrac{}{XY, Y, \phi(x), Y'_z \to \phi(z) \vdash \phi(x), \phi(z)}\,(b)}{XY, Y, Y' \to \phi(x), Y'_z \to \phi(z) \vdash \phi(x), \phi(z)}\,(\to l)}{XY, Y, \forall x(Y' \to \phi(x)) \vdash \phi(x), \phi(z)}\,(2 \times \forall l)}{XY, \forall x(Y' \to \phi(x)) \vdash \phi(z), Y \to \phi(x)}\,(\to r)}{XY, \forall x(Y' \to \phi(x)) \vdash \phi(z), \forall x(Y \to \phi(x))}\,(\forall r)}{(*)}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{(*)}{XY, \forall x(Y' \to \phi(x)) \vdash \phi(z), \forall x(Y \to \phi(x))} \qquad \dfrac{\dfrac{}{\phi(z), XY, \forall x(Y' \to \phi(x)) \vdash \phi(z)}\,(b)}{\forall z(\phi(z)), XY, \forall x(Y' \to \phi(x)) \vdash \phi(z)}\,(\forall l)}{\forall x(Y \to \phi(x)) \to \forall z(\phi(z)), XY, \forall x(Y' \to \phi(x)) \vdash \phi(z)}\,(\to r)}{\forall x(Y \to \phi(x)) \to \forall z(\phi(z)), XY, \forall x(Y' \to \phi(x)) \vdash \forall z(\phi(z))}\,(\forall r)}{\forall x(Y \to \phi(x)) \to \forall z(\phi(z)), XY \vdash \forall x(Y' \to \phi(x)) \to \forall z(\phi(z))}\,(\to r)}{\forall x(Y \to \phi(x)) \to \forall z(\phi(z)) \wedge XY \vdash \forall x(Y' \to \phi(x)) \to \forall z(\phi(z))}\,(\wedge l)}{\vdash ((\forall x(Y \to \phi(x)) \to \forall z(\phi(z))) \wedge XY) \to (\forall x(Y' \to \phi(x)) \to \forall z(\phi(z)))}\,(\to r)$$

□

2.2.3 A Natural Deduction Proof from Daniel Spencer

I assume that there are the elements x' and y' within the domain

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{[\forall y(R(y,x')\to\phi(y))]}{R(y',x')\to\phi(y')}\,\forall\mathrm{E} \qquad \dfrac{[R'(y',x')] \qquad \dfrac{\dfrac{(\forall xy)(R'(x,y)\to R(x,y))}{(\forall y)(R'(x',y)\to R(x',y))}\,\forall\mathrm{E}}{R'(x',y)\to R(x',y)}\,\forall\mathrm{E}}{R(y',x')}}{\phi(y')}}{R'(y',x')\to\phi(y')}\,\to\mathrm{I}}{\forall y(R'(y,x')\to\phi(y))}\,\forall\mathrm{I} \qquad \dfrac{[\forall x(\forall y(R'(y,x)\to\phi(y))\to\phi(x))]}{\forall y(R'(y,x')\to\phi(y))\to\phi(x')}\,\forall\mathrm{E}}{\phi(x')}\,\to\mathrm{E}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\phi(x')}{\forall y(R(y,x')\to\phi(y))\to\phi(x')}\,\to\mathrm{I}}{\forall x(\forall y(R(y,x)\to\phi(y))\to\phi(x))}\,\forall\mathrm{I} \qquad \forall x(\forall y(R(y,x)\to\phi(y))\to\phi(x))\to\forall z(\phi(z))}{\forall z(\phi(z))}\,\to\mathrm{E}}{\forall x(\forall y(R'(y,x)\to\phi(y))\to\phi(x))\to\forall z(\phi(z))}\,\to\mathrm{I}$$

2.3 Exercise 4 on Graphs

Use resolution to prove that a graph and its complement cannot both be disconnected.

Suppose G is a graph such that G and $\overline{G}$ are both disconnected. We will derive a contradiction by resolution.

If G is disconnected then there are vertices a and b which are not connected in G. If $\overline{G}$ is disconnected then there are vertices c and d which are disconnected in $\overline{G}$. It is our earnest hope that we can obtain a contradiction by considering just these four vertices. If that doesn't work we'll have to try something else.

Let us have six propositional letters: ab, ac, ad, bc, bd, cd. The intended interpretation is that ab (for example) means that the edge ab belongs to the edge set of G.

First consider G. The first thing we know is that the edge ab is **not** in the edge set of G, whence we have the clause $\{\neg ab\}$. Since we know that a and b are disconnected in G, we cannot allow any indirect paths from a to b. There are two possible lengths of indirect path involving 1 or 2 indirect vertices. (It will turn out that we can get our desired contradiction without considering indirect paths that are longer, but we don't know that yet, and are just hoping for the best!) This tells us that $(\neg ac \vee \neg bc)$, or, in resolution jargon, $\{\neg ac, \neg bc\}$. Similarly we may add $\{\neg ad, \neg bd\}$. The paths involving 2 vertices are $acdb$ and $adcb$. We already know that the path cd must be present (since it cannot be in $\overline{G}$). Therefore we may add the clauses $\{\neg ac, \neg bd\}$ and $\{\neg ad, \neg cb\}$.

Now consider $\overline{G}$. This cannot have cd, so we can add $\{cd\}$ (this is not negated, since we are now considering edges that are not in $\overline{G}$, and hence must be in G). Similarly, we cannot have any indirect connections from c to d, so we cannot have the paths cad, cbd, $cabd$ and $cbad$. Since we know that ab cannot be in the graph, we can write these as the clauses: $\{ac, ad\}$, $\{bc, bd\}$, $\{ac, bd\}$ and $\{bc, ad\}$. Note that the last two do not contain ab since we know that we **must** have $\neg ab$ by choice of a and b.

So now we have a set of clauses representing the conditions that need to be satisfied if both G and $\overline{G}$ are to be disconnected. To recapitulate, these are:

$$\begin{gathered}\{\neg ab\}\ \{\neg ac, \neg bc\}\ \{\neg ad, \neg bd\}\ \{\neg ad, \neg bc\}\ \{\neg ac, \neg bd\} \\ \{cd\}\ \{ac, ad\}\ \{bc, bd\}\ \{ac, bd\}\ \{ad, bc\}\end{gathered}$$

Now we may combine these clauses (carefully) using resolution – the choice of clauses to resolve is crucial, since it is very easy to end up with many useless clauses of the form $\{A, \neg A\}$.

$$\dfrac{\dfrac{\{\neg ac, \neg bc\} \qquad \{bc, bd\}}{\{\neg ac, bd\}} \qquad \{\neg ac, \neg bd\}}{\{\neg ac\}} \tag{2.1}$$

$$\dfrac{\dfrac{\{\neg ad, \neg bd\} \qquad \{bd, bc\}}{\{\neg ad, bc\}} \qquad \{\neg ad, \neg bc\}}{\{\neg ad\}} \tag{2.2}$$

Now that we have two literal clauses, we can use them to derive a contradiction:

$$\frac{\dfrac{\{ac, ad\} \qquad \{\neg ac\}}{\{ad\}} \qquad \{\neg ad\}}{\bot} \tag{2.3}$$

We have derived the empty clause.

Rob Thatcher (whose proof this is) comments: "Note that only six of the original 10 clauses were required for the resolution – partly because the two simple clauses are disjoint from the remaining eight (due to the way we initially wrote down the clauses – in a sense they have already been used in a resolution). The remaining two clauses are superfluous: they do not provide any information beyond the original eight. The clauses found when considering the 3-stage path in $\overline{G}$ are complementary to those found when considering similar paths in G, and hence if resolved in any way with them, give useless clauses of the form $\{A, \neg A\}$".

Evidently we are not going to be able to use the singleton clauses, since the literals within them appear nowhere else. In fact, Rob uses only these six clauses:

$$\{\neg ac, \neg bc\}\ \{\neg ad, \neg bd\}\ \{\neg ad, \neg bc\}\ \{\neg ac, \neg bd\}\ \{ac, ad\}\ \{bc, bd\}$$

If you think about it for a bit you will see that this cannot give us the empty clause if we use only input resolution. Once we reach a singleton clause we will only ever have singleton clauses, since our input clauses are all doubletons!

2.3.1 Doing it in first-order logic

Here is another discussion provoked by another answer, submitted by Nick Spooner (top of 1a in 2011). It is incomplete, so you might not wish to read it . . . unless, that is, you are planning to finish off the work i started here. It goes without saying that the usual bribes are offered.

Let us write $E(x, y)$ to say that x and y are joined by a G-edge. There are no loops at vertices so we need

$$(\forall x)(\neg E(x, x)) \tag{9}$$

We write $P(x, y)$ to say that x and y are connected in G.

$$(\forall xy)(P(x, y) \longleftrightarrow (E(x, y) \vee (\exists z)(P(x, z) \wedge E(y, z)))) \tag{10}$$

To capture "x is connected to y in $\overline{G}$" we will need a predicate $Q(x, y)$ and an analogue of (10). . .

$$(\forall xy)(Q(x, y) \longleftrightarrow ((\neg E(x, y) \wedge x \neq y) \vee (\exists z)(Q(x, z) \wedge \neg E(y, z) \wedge y \neq z))) \tag{11}$$

This looks slightly different from the (10); this is because the edge relation of $\overline{G}$ is not the complement of the edge relation of G!

Finally we have to say that G and $\overline{G}$ are not both disconnected.

$$(\forall xy)(P(x, y)) \vee (\forall xy)(Q(x, y)) \tag{12}$$

Of course there is another way. You could invent a predicate letter E' for the edge relation in $\overline{G}$, and adopt an axiom to say that precisely one of $\{E(x, y), x = y, E'(x, y)\}$ holds.

And i'm still not 100% satisfied that we don't need to worry about E (and, if we adopt it, E') being symmetric.

2.3.2 A Sequent Proof

This challenges us to prove the sequent

$$(\forall xy)(P(x,y) \longleftrightarrow (E(x,y) \lor (\exists z)(P(x,z) \land E(y,z)))), (\forall x)(\neg E(x,x)), (\forall xy)(Q(x,y) \longleftrightarrow ((\neg E(x,y) \land x \neq y) \lor (\exists z)(Q(x,z) \land \neg E(y,z) \land y \neq z))) \ \vdash \ (\forall xy)(P(x,y)) \lor (\forall xy)(Q(x,y))$$

2.3.3 A Resolution Proof

It may be easier as a resolution exercise, which is what Mr Spooner took it to be.

If we skolemise (11) we obtain

$$P(x,y) \longleftrightarrow (E(x,y) \lor (P(x,f(x,y)) \land E(y,f(x,y)))) \tag{13}$$

giving the conjunction

$$(P(x,y) \to (E(x,y) \lor (P(x,f(x,y)) \land E(y,f(x,y))))) \land ((E(x,y) \lor (P(x,f(x,y)) \land E(y,f(x,y)))) \to P(x,y)) \tag{14}$$

which we can process successively into

$$(\neg P(x,y) \lor (E(x,y) \lor (P(x,f(x,y)) \land E(y,f(x,y))))) \land (\neg(E(x,y) \lor (P(x,f(x,y)) \land E(y,f(x,y)))) \lor P(x,y)) \tag{15}$$

$$(\neg P(x,y) \lor (E(x,y) \lor (P(x,f(x,y)) \land E(y,f(x,y))))) \land (\neg E(x,y) \land (\neg P(x,f(x,y)) \lor \neg E(y,f(x,y)))) \lor P(x,y) \tag{16}$$

$$(\neg P(x,y) \lor E(x,y) \lor (P(x,f(x,y)) \land E(y,f(x,y)))) \land (\neg E(x,y) \land (\neg P(x,f(x,y)) \lor \neg E(y,f(x,y)))) \lor P(x,y)) \tag{17}$$

With a view to mortal comprehension we should try reading the right-hand conjunct as $(A \land (B \lor C)) \lor D$, where A is $\neg E(x,y)$, $B \lor C$ is $\neg P(x,f(x,y)) \lor \neg E(y,f(x,y))$ and D is $P(x,y)$ and we can distribute to get $(A \lor D) \land (B \lor C \lor D)$ which is

$$(\neg E(x,y) \lor P(x,y)) \land (\neg P(x,f(x,y)) \lor \neg E(y,f(x,y)) \lor P(x,y))$$

which is in CNF and gives the two clauses

$$\{\neg E(x,y),\ P(x,y)\} \text{ and } \{\neg P(x,f(x,y)), \neg E(y,f(x,y)), P(x,y)\}$$

which we can delete from work-in-progress leaving only the left conjunct

$$\neg P(x,y) \lor E(x,y) \lor (P(x,f(x,y)) \land E(y,f(x,y))) \tag{17}$$

which will give two clauses

$$\{\neg P(x,y), E(x,y), E(y,f(x,y))\} \text{ and } \{\neg P(x,y), E(x,y), P(x,f(x,y))\}$$

So, from skolemising (11) we obtain the clauses

$\{\neg E(x,y),\ P(x,y)\}$;
$\{\neg P(x,f(x,y)), \neg E(y,f(x,y)), P(x,y)\}$;
$\{\neg P(x,y), E(x,y), E(y,f(x,y))\}$;
and $\{\neg P(x,y), E(x,y), P(x,f(x,y))\}$

Now we need to skolemise (12)

$$(\forall xy)(Q(x,y) \longleftrightarrow ((\neg E(x,y) \wedge x \neq y) \vee (\exists z)(Q(x,z) \wedge \neg E(y,z) \wedge y \neq z))) \tag{12}$$

$$(Q(x,y) \longleftrightarrow ((\neg E(x,y) \wedge x \neq y) \vee (Q(x,g(x,y)) \wedge \neg E(y,g(x,y)) \wedge y \neq g(x,y)))) \tag{18}$$

$$(Q(x,y) \to ((\neg E(x,y) \wedge x \neq y) \vee (Q(x,g(x,y)) \wedge \neg E(y,g(x,y)) \wedge y \neq g(x,y)))) \wedge (((\neg E(x,y) \wedge x \neq y) \vee (Q(x,g(x,y)) \wedge \neg E(y,g(x,y)) \wedge y \neq g(x,y))) \to Q(x,y)) \tag{19}$$

$$(\neg Q(x,y) \vee ((\neg E(x,y) \wedge x \neq y) \vee (Q(x,g(x,y)) \wedge \neg E(y,g(x,y)) \wedge y \neq g(x,y)))) \wedge (\neg((\neg E(x,y) \wedge x \neq y) \vee (Q(x,g(x,y)) \wedge \neg E(y,g(x,y)) \wedge y \neq g(x,y))) \vee Q(x,y)) \tag{20}$$

Abbreviating $\neg E(x,y)$ to A, $x \neq y$ to B, $Q(x,g(x,y))$ to C, $\neg E(y,g(x,y))$ to D,$y \neq g(x,y))$ to E and $Q(x,y)$ to F the right-hand conjunct becomes

$$\neg((A \wedge B) \vee (C \wedge D \wedge E)) \vee F$$

$$(\neg(A \wedge B) \wedge \neg(C \wedge D \wedge E)) \vee F$$

$$((\neg A \vee \neg B) \wedge (\neg C \vee \neg D \vee \neg E)) \vee F$$

$$(\neg A \vee \neg B \vee F) \wedge (\neg C \vee \neg D \vee \neg E \vee F)$$

Giving two clauses

$$\{\neg E(x,y), x = y, Q(x,y)\} \text{ and } \{Q(x,g(x,y)), \neg E(y,g(x,y)), y = g(x,y), Q(x,y)\}$$

The left-hand conjunct is

$$\neg Q(x,y) \vee (\neg E(x,y) \wedge x \neq y) \vee (Q(x,g(x,y)) \wedge \neg E(y,g(x,y)) \wedge y \neq g(x,y))$$

which will give six clauses

$$\begin{array}{l}
\{\neg Q(x,y), \neg E(x,y), Q(x,g(x,y))\}; \\
\{\neg Q(x,y), \neg E(x,y), \neg E(y,g(x,y))\}; \\
\{\neg Q(x,y), \neg E(x,y), y \neq g(x,y)\}; \\
\{\neg Q(x,y), x \neq y, Q(x,g(x,y))\}; \\
\{\neg Q(x,y), x \neq y, \neg E(y,g(x,y))\}; \\
\{\neg Q(x,y), x \neq y, y \neq g(x,y)\}
\end{array}$$

Finally we need clauses to say that both G and $\overline{G}$ are disconnected. $(\exists xy)\neg P(x,y)$ and $(\exists xy)\neg Q(x,y)$ which, on skolemisation, gives $\{\neg P(a,b)\}$ and $\{\neg P(c,d)\}$ giving us the grand total

$\{\neg E(x,x)\}$;	from (9)
$\{\neg E(x,y),\ P(x,y)\}$;	from (10)
$\{\neg P(x,f(x,y)),\ \neg E(y,f(x,y)),\ P(x,y)\}$;	from (10)
$\{\neg P(x,y),\ E(x,y),\ E(y,f(x,y))\}$;	from (10)
$\{\neg P(x,y),\ E(x,y),\ P(x,f(x,y))\}$;	from (10)
$\{Q(x,g(x,y)),\ \neg E(y,g(x,y)),\ y=g(x,y),\ Q(x,y)\}$;	from (11)
$\{\neg E(x,y),\ x=y,\ Q(x,y)\}$;	from (11)
$\{\neg Q(x,y),\ \neg E(x,y),\ Q(x,g(x,y))\}$;	from (11)
$\{\neg Q(x,y),\ \neg E(x,y),\ \neg E(y,g(x,y))\}$;	from (11)
$\{\neg Q(x,y),\ \neg E(x,y),\ y\neq g(x,y)\}$;	from (11)
$\{\neg Q(x,y),\ x\neq y,\ Q(x,g(x,y))\}$;	from (11)
$\{\neg Q(x,y),\ x\neq y,\ \neg E(y,g(x,y))\}$;	from (11)
$\{\neg Q(x,y),\ x\neq y,\ y\neq g(x,y)\}$;	from (11)
$\{\neg P(a,b)\}$;	from (12)
$\{\neg Q(c,d)\}$;	from (12)

So far so good!

2.4 Exercise 7

2.4.1 A Resolution Proof of Part 1

$$(\forall x\exists y)R(x,y)\rightarrow(\forall x\exists y)(\forall x'\exists y')(R(x,y)\wedge R(x',y')\wedge(x=x'\rightarrow y=y'))$$

Negate:

$$(\forall x\exists y)R(x,y)\wedge(\exists x\forall y\exists x'\forall y')\neg(R(x,y)\wedge R(x',y')\wedge(x=x'\rightarrow y=y'))$$

$$\forall x\exists yR(x,y)\wedge(\exists x\forall y\exists x'\forall y')(\neg(R(x,y)\vee\neg R(x',y')\vee(x=x'\wedge y\neq y'))$$

$$\forall x\exists yR(x,y)\wedge(\exists x\forall y\exists x'\forall y')((\neg R(x,y)\vee\neg R(x',y')\vee x=x')\wedge(\neg R(x,y)\vee\neg R(x',y')\vee y\neq y'))$$

Reletter, just to be safe:

$$\forall x\exists yR(x,y)\wedge(\exists x\forall u\exists x'\forall y')((\neg R(x,u)\vee\neg R(x',y')\vee x=x')\wedge(\neg R(x,u)\vee\neg R(x',y')\vee u\neq y'))$$

Skolemise . . . $y\mapsto f(x)$; x in the negated consequent $\mapsto a$; $x'\mapsto g(u)$, and reletter 'y'' to 'z' for neatness' sake.

$$\forall xR(x,f(x))\wedge(\forall u\forall z)((\neg R(a,u)\vee\neg R(g(u),z)\vee a=g(u))\wedge(\neg R(a,u)\vee\neg R(g(u),z)\vee u\neq z))$$

Reletter 'u' back to 'y':
This gives us the clauses

(1) $\{R(x,f(x)\}$,

(2) $\{\neg R(a,y),\ \neg R(g(y),z),\ a=g(y)\}$, and

(3) $\{\neg R(a,y),\ \neg R(g(y),z),\ y\neq z\}$

We can resolve (1) with (2) by binding $x \mapsto a$ and $y \mapsto f(a)$ getting

(4) $\{\neg R(g(f(a)), z),\ a = g(f(a))\}$,

and of course we can resolve (1) with (3) by binding $x \mapsto a$ and $y \mapsto f(a)$ getting

(5) $\{\neg R(g(f(a)), z),\ f(a) \neq z\}$.

Now we can resolve (1) with (4) by binding $x \mapsto g(f(a))$ and $z \mapsto f(x)$ (and therefore to $f(g(f(a)))$ getting

(6) $\{a = g(f(a))\}$,

and we can resolve (1) with (5) binding $x \mapsto g(f(a))$ and $z \mapsto f(x)$ (and therefore to $f(g(f(a)))$ getting

(7) $\{f(a) \neq f(g(f(a)))\}$.

at which point it becomes clear that you need some rules for dealing with functions and equality, such as

(8) $\{\neg(x = y), f(x) = f(y)\}$

We can resolve (8) with (6), binding $x \mapsto a$ and $y \mapsto g(f(a))$, getting

(9) $\{f(a) = f(g(f(a)))\}$

which immediately resolves with (7) to give the empty clause.
I don't know if there are proofs using linear resolution only or input resolution only.

2.4.2 A Resolution Proof of Part 2

$$(\forall x \exists y) R(x, y) \to (\forall x \exists y)(\forall x' \exists y')(\forall x'' \exists y'') \bigwedge \left(\begin{array}{l} R(x, y) \\ R(x', y') \\ R(x'', y'') \\ x = x' \to y = y' \\ x = x'' \to y = y'' \\ x' = x'' \to y' = y' \end{array} \right) \tag{2.4}$$

This looks too large to do by hand, tho' i have no doubt that a suitable program will chew it up and spit it out in seconds. My attempt is incomplete, so i repeat the *caveat* on p. 17: you might not wish to read it ... unless, that is, you are planning to finish off the work i started here. It goes without saying that the usual bribes are offered.

Negate

$$(\forall x \exists y) R(x, y) \wedge (\exists x \forall y)(\exists x' \forall y')(\exists x'' \forall y'') \bigvee \neg \left(\begin{array}{l} R(x, y) \\ R(x', y') \\ R(x'', y'') \\ x = x' \to y = y' \\ x = x'' \to y = y'' \\ x' = x'' \to y' = y' \end{array} \right) \tag{2.5}$$

Peel off the antecedent to obtain the clause $\{R(x, f(x))\}$, and import the '$\neg$'

$$(\exists x \forall y)(\exists x' \forall y')(\exists x'' \forall y'') \bigvee \left(\begin{array}{l} \neg R(x, y) \\ \neg R(x', y') \\ \neg R(x'', y'') \\ \neg(x = x' \to y = y') \\ \neg(x = x'' \to y = y'') \\ \neg(x' = x'' \to y' = y') \end{array} \right) \tag{2.6}$$

$$(\exists x \forall y)(\exists x' \forall y')(\exists x'' \forall y'') \bigvee \left(\begin{array}{l} \neg R(x, y) \\ \neg R(x', y') \\ \neg R(x'', y'') \\ x = x' \wedge y \neq y' \\ x = x'' \wedge y \neq y'' \\ x' = x'' \wedge y' \neq y' \end{array} \right) \tag{2.7}$$

Skolemise: $x \mapsto a$, $x' \mapsto g(y)$, $x'' \mapsto h(y, y')\ldots$

$$\bigvee \left(\begin{array}{l} \neg R(a, y) \\ \neg R(g(y), y') \\ \neg R(h(y, y'), y'') \\ a = g(y) \wedge y \neq y' \\ a = h(y, y'') \wedge y \neq y'' \\ g(y) = h(y, y') \wedge y' \neq y'' \end{array} \right) \tag{2.8}$$

We now distribute, and obtain 8 conjuncts

Let's abbreviate '$\neg R(a, y) \vee \neg R(g(y), y') \vee \neg R(h(y, y'), y'')$' to '$D$' *pro tem.* to save space. . .

$D \vee a = g(y) \vee a = h(y, y'') \vee g(y) = h(y, y')$

$D \vee a = g(y) \vee a = h(y, y'') \vee y' \neq y''$

$D \vee a = g(y) \vee y \neq y'' \vee g(y) = h(y, y')$

$D \vee a = g(y) \vee y \neq y'' \vee y' \neq y''$

$D \vee y \neq y' \vee a = h(y, y'') \vee g(y) = h(y, y')$

$D \vee y \neq y' \vee a = h(y, y'') \vee y' \neq y''$

$D \vee y \neq y' \vee y \neq y'' \vee g(y) = h(y, y')$

$D \vee y \neq y' \vee y \neq y'' \vee y' \neq y''$

giving us the clauses

(1) $\{\neg R(a, y),\ \neg R(g(y), y'),\ \neg R(h(y, y'), y''),\ a = g(y),\ a = h(y, y''),\ g(y) = h(y, y')\}$

(2) $\{\neg R(a, y),\ \neg R(g(y), y'),\ \neg R(h(y, y'), y''),\ a = g(y),\ a = h(y, y''),\ y' \neq y''\}$

(3) $\{\neg R(a, y),\ \neg R(g(y), y'),\ \neg R(h(y, y'), y''),\ a = g(y),\ y \neq y'',\ g(y) = h(y, y')\}$

(4) $\{\neg R(a, y),\ \neg R(g(y), y'),\ \neg R(h(y, y'), y''),\ a = g(y),\ y \neq y'',\ y' \neq y''\}$

(5) $\{\neg R(a, y),\ \neg R(g(y), y'),\ \neg R(h(y, y'), y''),\ y \neq y',\ a = h(y, y''),\ g(y) = h(y, y')\}$

(5) $\{\neg R(a, y),\ \neg R(g(y), y'),\ \neg R(h(y, y'), y''),\ y \neq y',\ a = h(y, y''),\ y' \neq y''\}$

(6) $\{\neg R(a, y),\ \neg R(g(y), y'),\ \neg R(h(y, y'), y''),\ y \neq y',\ y \neq y'',\ g(y) = h(y, y')\}$

(7) $\{\neg R(a, y),\ \neg R(g(y), y'),\ \neg R(h(y, y'), y''),\ y \neq y',\ y \neq y'',\ y' \neq y''\}$
plus of course
(8) $\{R(x, f(x))\}$

Naturally for the three function letters f g and h we will need the analogues of clause (8) from the first part of this exercise, namely

(9) $\{\neg(x = y), f(x) = f(y)\}$

(10) $\{\neg(x = y), \neg(u = v), g(x, u) = g(y, v)\}$

(11) $\{\neg(x = y), \neg(u = v), h(x, u) = h(y, v)\}$

Now this is not as scary as it might look. Clearly, we have to resolve each of the first eight recently obtained clauses with $\{R(x, f(x))\}$ **thrice**, and we generate the same bindings on each occasion:

At the first pass $x \mapsto a$ and $y \mapsto f(a)$.
At the second pass $x \mapsto g(f(a))$ and $y' \mapsto f(g(f(a)))$.
At the third pass $x \mapsto h(y, y')$ which by then has become $h(f(a), f(g(f(a))$, so $y'' \mapsto f(h(f(a), f(g(f(a)))))$

This gives us eight clauses, obtained from our original eight by deleting the disjuncts in D and binding $y' \mapsto f(g(f(a)))$ and $y'' \mapsto f(h(f(a), f(g(f(a)))))$
$\{a = g(f(a)),\ a = h(f(a), f(h(f(a), f(g(f(a)))))),\ f(g(f(a))) = f(h(f(a), f(g(f(a)))))\}$
$\{a = g(f(a)),\ a = h(f(a), f(h(f(a), f(g(f(a)))))),\ f(a) \neq f(g(f(a)))\}$
$\{a = g(f(a)),\ f(a) \neq f(h(f(a), f(g(f(a))))),\ f(g(f(a))) = f(h(f(a), f(g(f(a)))))\}$
$\{a = g(f(a)),\ f(a) \neq f(h(f(a), f(g(f(a))))),\ f(a) \neq f(g(f(a)))\}$
$\{f(a) \neq f(g(f(a))),\ a = h(f(a), f(h(f(a), f(g(f(a)))))),\ f(g(f(a))) = f(h(f(a), f(g(f(a)))))\}$
$\{f(a) \neq f(g(f(a))),\ a = h(f(a), f(h(f(a), f(g(f(a)))))),\ f(a) \neq f(g(f(a)))\}$
$\{f(a) \neq f(g(f(a))),\ f(a) \neq f(h(f(a), f(g(f(a))))),\ f(g(f(a))) = f(h(f(a), f(g(f(a)))))\}$
$\{f(a) \neq f(g(f(a))),\ f(a) \neq f(h(f(a), f(g(f(a))))),\ f(a) \neq f(g(f(a)))\}$

If we now regard $a = g(f(a)$ and $f(a) \neq f(g(f(a)))$ as contradictories (using clause 9) we can cut the first formula in the latest list against the fifth, the second agains the sixth, the third against the seventh and the fourth against the eighth to obtain

$\{a = h(f(a), f(h(f(a), f(g(f(a)))))),\ f(g(f(a))) = f(h(f(a), f(g(f(a)))))\}$
$\{a = h(f(a), f(h(f(a), f(g(f(a)))))),\ f(a) \neq f(g(f(a)))\}$
$\{f(a) \neq f(h(f(a), f(g(f(a))))),\ f(g(f(a))) = f(h(f(a), f(g(f(a)))))\}$
$\{f(a) \neq f(h(f(a), f(g(f(a))))),\ f(a) \neq f(g(f(a)))\}$

I was expecting to be able to work the same trick again but i seem to have made a transcription error.

Thinking aloud...

The four clauses are of the form
$a = b \vee c = d,\ a = b \vee f(a) \neq c,\ f(a) \neq d \vee c = d,\ f(a) \neq d \vee f(a) \neq c.$
The conjunction of these four is equivalent to

$(a = b \wedge f(a) \neq d) \vee (c = d \wedge f(a) \neq c)$
In the first case we have

$a = h(f(a), f(h(f(a), f(g(f(a)))))) \wedge f(a) \neq f(h(f(a), f(g(f(a)))))$

The second conjunct implies $a \neq h(f(a), f(g(f(a)))))$. However the first conjunct tells us that $a = h(f(a), f(h(f(a), f(g(f(a))))))$, so the two second components of the inputs to h must differ, whence $f(h(f(a), f(g(f(a))))) \neq f(g(f(a)))$, whence $h(f(a), f(g(f(a))) \neq g(f(a))$.

In the second case we have
$f(g(f(a))) = f(h(f(a), f(g(f(a))))) \wedge f(a) \neq f(g(f(a)))$
and the second conjunct implies $a \neq g(f(a)$.
$f(a) \neq f(g(f(a))) = f(h(f(a), f(g(f(a))))))$ so
$f(a) \neq f(h(f(a), f(g(f(a)))))$ so
$a \neq h(f(a), f(g(f(a))))$ so
trails off rather ...

2.5 Exercise 8

2.5.1 A Natural Deduction Proof

The natural deduction proof i favour looks like this:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{[p]^1 \qquad [\tau(x) \to \bot]^2}{\tau(x) \to \bot}\text{ identity rule}}{p \to (\tau(x) \to \bot)} \to\text{-int (1)}}{Tp \to T(\tau(x) \to \bot)}\text{ Affixing} \qquad Tp}{T(\tau(x) \to \bot)} \to\text{-elim}}{\tau(x)}\text{ Mixed Rule} \qquad [\tau(x) \to \bot]^2}{\bot} \to\text{-elim}}{(\tau(x) \to \bot) \to \bot} \to\text{-int (2)}}{(\forall x)((\tau(x) \to \bot) \to \bot)}\ \forall\text{-int} \tag{2.9}$$

2.5.2 A Sequent Proof

The shortest sequent calculus proof i can find is the following.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{p, \tau(x) \ \vdash \ \tau(x)}{p \ \vdash \ \neg\tau(x), \tau(x)}\ \neg\text{ R}}{Tp, \ \vdash \ T(\neg\tau(x)), \tau(x)}\text{ Affixing}}{Tp, \ \vdash \ \tau(x), \tau(x)}\text{ Mixed rule}}{Tp \ \vdash \ \tau(x)}\text{ contraction-R}}{Tp \ \vdash \ (\forall x)(\tau(x))}\ \forall\text{ R} \tag{2.10}$$

Some formulæ are highlighted in red; in each case the highlighted formula is the eigenformula of the rule being aplied to the sequent that contains that formula. That proof is classical. Here is a constructive proof:

$$\begin{array}{cl} p, \neg\tau(a) \;\vdash\; \neg\tau(a) & \\ \hline Tp, \neg\tau(a) \;\vdash\; T(\neg\tau(a)) & \text{Affixing} \\ \hline Tp, \neg\tau(a) \;\vdash\; \tau(a) & \text{Mixed rule} \\ \hline Tp, \neg\tau(a), \neg\tau(a) \;\vdash & \neg\ \text{L} \\ \hline Tp, \neg\tau(a) \;\vdash & \text{Contract-L} \\ \hline Tp \;\vdash\; \neg\neg\tau(a) & \neg\ \text{R} \\ \hline Tp \;\vdash\; (\forall x)(\neg\neg\tau(x)) & \forall\ \text{R} \end{array} \tag{2.11}$$

2.5.3 “Prove that a premiss of the form Tp really is needed.”

Notice that since neither the affixing rule nor the mixed rule have anything like $T\phi$ as a conclusion, we can obtain models of this calculus in which Tp is always false (The modal logicians express this by saying that T is a *falsum* operator) and $\tau(x)$ is always false. Accordingly we cannot expect to be able to prove that even one thing is τ without some extra premisses.

Chapter 3

Discussion Answers to Old Tripos Questions

3.1 1993:3:3

Must write this out properly at some point. Try to prove the nonexistence of the Russell class by resolution. You need what Larry calls *factoring*... which seems to be a kind of contraction

Derive a contradiction from

$$(\exists y)(\forall x)(x \in y \longleftrightarrow x \notin x)$$

Skolemising gives $x \in a \longleftrightarrow x \notin x$ which is a conjunction of $x \in a \to x \notin x$ and $x \notin x \to x \in a$ whence two clauses

$$\{x \notin a \vee x \notin x\} \text{ and } \{x \in x \vee x \in a\}$$

Clearly you have to bind 'x' $\mapsto$ 'a', whereupon the first clause becomes $\{a \notin a \vee a \notin a\}$ and the second becomes $\{a \in a \vee a \in a\}$.
At this point you have to contract the two occurrences in each clause to obtain singleton clauses $\{a \notin a\}$ and $\{a \in a\}$.
[BAD JOIN]

1. $\{\neg P(x), Q(x)\}$
2. $\{\neg P(x), \neg Q(x), P(fx)\}$.
3. $\{P(b)\}$
4. $\{\neg P(f^4x)\}$.

   1 and 2 resolve to give
5. $\{\neg P(x), P(fx)\}$. First reletter this to get
6. $\{\neg P(w), P(fw)\}$

   Resolve 5 and 6 by unifying $w \to fx$, cut against $P(fx)$ to get
7. $\{\neg P(x), P(f^2x)\}$. First reletter this to get

8. $\{\neg P(w), P(f^2w)\}$.

   Resolve 7 and 8 by unifying $w \to f^2x$ cut against $P(f^2x)$ to get

9. $\{\neg P(x), P(f^4x)\}$. Resolve with 3 to get

10. $\{P(f^4b)\}$. Resolve with 4 to get the empty clause

11. $\{\}$.

That's the clever way to do it. I think what PROLOG does is something more like this. It cuts 4 against 2 to get $\{\neg P(f^3x), \neg Q(f^3x)\}$ and cuts against 1 to get $\{\neg P(f^3x)\}$.

Then repeat until you get $\{\neg P(x)\}$ which you can cut against 3. The point is that at each stage PROLOG only ever cuts the current goal clause against something it was given to start with. That way it has only a linear search for a cut at each stage instead of a quadratic one. I'm not sure what sort of relettering PROLOG does, and whether it can make copies of clauses, and reletter one to cut against the other as above. It certainly only ever does linear resolution.

(a) How long does it take?
(b)
(c)
(d)

$$\neg[(\forall y \exists x)\neg(p(x,y) \longleftrightarrow \neg(\exists z)(p(x,z) \wedge p(z,x)))]$$

Rewrite to get rid of the biconditional

$$\neg[(\forall y \exists x)\neg(p(x,y) \vee \neg(\exists z)(p(x,z) \wedge p(z,x)) \wedge p(x,y) \vee \neg(\exists z)(p(x,z) \wedge p(z,x))]$$

push in $\neg$

$$[(\exists y \forall x)\neg(p(x,y) \vee \neg(\exists z)(p(x,z) \wedge p(z,x)) \wedge p(x,y) \vee \neg(\exists z)(p(x,z) \wedge p(z,x))]$$

3.2 1996:5:10

$$(\forall z)(\exists x)(\forall y)((P(y) \to Q(z)) \to (P(x) \to Q(x)))$$

Given that the decision problem for first-order logic is undecidable, you haven't much chance of finding a proof of something or a convincing refutation of it unless you postpone work on it until you have a feel for what it is saying.

First we notice that as long as there is an x s.t. $Q(x)$ we can take that element to be a witness to the '$\exists x$' no matter what z is. This is because the truth of '$Q(x)$' ensures the truth of the whole conditional. On the other hand even if *nothing* is Q we are still OK as long as nothing is P—because the falsity of '$P(x)$' ensures the truth of the consequent of the main conditional. There remains the case where $(\forall x)(\neg Q(x))$ and $(\exists x)(P(x))$. But it's easy to check that in that case the whole conditional comes out true too.

So we can approach the search for a sequent calculus proof confident that there is one to be found.

Clearly the only thing we can do with

$$\vdash (\forall z)(\exists x)(\forall y)((P(y) \to Q(z)) \to (P(x) \to Q(x)))$$

is a $\forall$-R getting

$$\vdash (\exists x)(\forall y)((P(y) \to Q(a)) \to (P(x) \to Q(x)))$$

(I have relettered ‘z’ to ‘a’ for no particular reason). We could have got this by ∃-R by replacing ‘a’ by ‘x’ so that it came from

$$\vdash (\forall y)((P(y) \to Q(a)) \to (P(a) \to Q(a)))$$

but this doesn’t appear to be valid. So we presumably have to keep an extra copy of ‘$\vdash (\exists x)(\forall y)((P(y) \to Q(a)) \to (P(x) \to Q(x)))$’ and we got it from

$$\vdash (\exists x)(\forall y)((P(y) \to Q(a)) \to (P(x) \to Q(x))), \quad (\forall y)((P(y) \to Q(a)) \to (P(a) \to Q(a)))$$

which came by ∀-R from

$$\vdash (\exists x)(\forall y)((P(y) \to Q(a)) \to (P(x) \to Q(x))), \quad ((P(b) \to Q(a)) \to (P(a) \to Q(a)))$$

This obviously came from an ∃-R:

$$\vdash (\forall y)((P(y) \to Q(a)) \to (P(b) \to Q(b))), \quad ((P(b) \to Q(a)) \to (P(a) \to Q(a)))$$

. . . where i’m assuming the ‘x’ came from the ‘b’ we’ve already seen.

and this must’ve come from a ∀-R with a new variable:

$$\vdash (P(c) \to Q(a)) \to (P(b) \to Q(b))), \quad ((P(b) \to Q(a)) \to (P(a) \to Q(a)))$$

and now we’ve got all the quantifiers out of the way and have only the propositional rules to worry about: pretty straightforward from here. Four applications of →-R take us to

$$P(c) \to Q(a),\ P(b) \to Q(a),\ P(b),\ P(a) \vdash Q(b),\ Q(a)$$

and if we break up the ‘$P(b) \to Q(a)$’ on the left we get the two initial sequents:

$$P(c) \to Q(a),\ P(b),\ P(a),\ \underline{Q(a)} \vdash Q(b),\ Q(a)$$

and

$$P(c) \to Q(a),\ P(b),\ P(a) \vdash \underline{P(b)},\ Q(b),\ Q(a)$$

. . . where i have underlined the two formulæ that get glued together by the →-L rule.

This gives us the proof

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{P(c) \to Q(a), P(b), P(a), Q(a) \;\vdash\; Q(b), Q(a), \underline{P(b)} \qquad \dfrac{P(b), P(a), Q(a) \;\vdash\; Q(b), Q(a), \underline{P(c)} \qquad \underline{Q(a)}, P(b), P(a), Q(a) \;\vdash\; Q(b), Q(a)}{P(c) \to Q(a), P(b), P(a), \underline{Q(a)} \;\vdash\; Q(b), Q(a)} \to \mathrm{L}}{P(c) \to Q(a), P(b) \to Q(a), P(b), P(a) \;\vdash\; Q(b), Q(a)} \to \mathrm{L}}{P(c) \to Q(a), P(b) \to Q(a), P(b) \;\vdash\; Q(b), P(a) \to Q(a)} \to \mathrm{R}}{P(c) \to Q(a), (P(b) \to Q(a) \;\vdash\; P(b) \to Q(b), P(a) \to Q(a)} \to \mathrm{R}}{(P(c) \to Q(a)) \;\vdash\; (P(b) \to Q(b))), ((P(b) \to Q(a)) \to (P(a) \to Q(a)))} \to \mathrm{R}}{\vdash\; (P(c) \to Q(a)) \to (P(b) \to Q(b))), \quad ((P(b) \to Q(a)) \to (P(a) \to Q(a)))} \to \mathrm{R}}{\vdash\; (\forall y)((P(y) \to Q(a)) \to (P(b) \to Q(b)), ((P(b) \to Q(a)) \to (P(a) \to Q(a)))} \forall\ \mathrm{R}}{\vdash\; (\exists x)(\forall y)((P(y) \to Q(a)) \to (P(x) \to Q(x)), ((P(b) \to Q(a)) \to (P(a) \to Q(a)))} \exists\ \mathrm{R}}{\vdash\; (\exists x)(\forall y)((P(y) \to Q(a)) \to (P(x) \to Q(x)), (\forall y)((P(y) \to Q(a)) \to (P(b) \to Q(b))), ((P(b) \to Q(a)) \to (P(a) \to Q(a)))} \forall\ \mathrm{R}}{\vdash\; (\exists x)(\forall y)((P(y) \to Q(a)) \to (P(x) \to Q(x))), ((P(b) \to Q(a)) \to (P(a) \to Q(a)))} \exists\ \mathrm{R}}{\vdash\; (\forall z)(\exists x)(\forall y)((P(y) \to Q(z)) \to (P(x) \to Q(x)))} \forall\ \mathrm{R} \tag{3.1}$$

3.3 1996:6:10

Davis-Putnam: This procedure has three main steps:

1. Delete tautological clauses;
2. Delete unit clauses $\{A\}$ and remove $\neg A$ from all clauses. This is safe because a unit clause $\{A\}$ can be satisfied only if $A \mapsto$ **true** and once that is done A does not need to be considered further.
3. Delete any formula containing pure literals. (If a literal appears always positively or always negatively we can send it to **true** or to **false** without compromising later efforts to find an interpretation of the formula).

If a point is reached where none of the rules above can be applied, a variable is selected arbitrarily for a **case split** and the proof proceeds along both resulting clause sets. We will be happy if *either* resolves to the empty clause. This algorithm terminates because each case split removes a literal.

In this example we have no tautological clauses or pure literals, so we start with a case split, arbitrarily selecting P to split. If P is true, our clauses are $\{R\}, \{\neg R\}$. We delete unit clause $\{R\}$, and then delete $\neg R$ from all clauses; we are left with the empty clause, which constitutes a refutation of the clause set (the empty disjunction), so the formula is valid. The P false case proceeds similarly, with Q for R.

Resolution: There is only one rule of inference in resolution:

$$\frac{\{B, A\} \quad \{\neg B, C\}}{\{A, C\}}$$

The algorithm terminates because as soon as a point is reached where the rule cannot be applied, the clause set is established as satisfiable. Repeatedly applying this rule to the given clause set:

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

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

$$\frac{\{R, \neg Q\} \quad \{\neg R, \neg Q\}}{\Box}$$

The empty clause ($\Box$) is a contradiction: we have refuted the clause set and so proved the original formula.

For the second half ...

Let A^* represent the formula A, converted into polynomial representation. First we note that in arithmetic mod 2, $x^2 \equiv x$, as $0^2 = 0 \equiv 0$ and $1^2 = 1 \equiv 1$, and all integers are congruent to 0 or 1 modulo 2. Now $(\neg A)^*$ is $1+A^*$, $(A \wedge B)^*$ is $A^* \cdot B^*$, $(A \vee B)^*$ becomes $A^*+B^*+A^*B^*$, $A \rightarrow B$ is $1+A^*+A^*B^*$, and $A \leftrightarrow B$ is $((1-A^*)+B^*) \cdot ((1-B^*)+A^*)$, which simplifies to $1+2A^*B^*-A^*-B^*$ and thence to $1+A^*+B^*$. Recursively applying these rules to any formula will convert it to equivalent polynomial form. $(A \wedge B) \leftrightarrow (B \wedge A)$ translates into $1 + 2(A^*B^*)^2 - A^*B^* - B^*A^* = 1$, hence the formula is a tautology. $A \leftrightarrow A$ translates into 1. $1 \leftrightarrow A$ translates into $1 + 2A^* - A^* - 1 = A^*$. So if we adopt the notation $(A \leftrightarrow A)^n$, to represent formulae of the given type where $\leftrightarrow$ appears n times, we get: $(A \leftrightarrow A)^n = 1$ (n odd) or A (n even), $n \geq 0$. This works for $n = 0$, which is just the formula A.

3.4 1998:6:10

Clause 1 tells us that if x pees on itself it pees on a. Clause 2 tells us that if x does *not* pee on itself then it pees on $f(a)$. This drops a broad hint that perhaps a is $\{x : x \in x\}$ and $f(a)$ is $\{x : x \notin x\}$. Clause 3 tells us that nothing pees on both a and $f(a)$—which is starting to look good. Now ask whether or not $P(f(a), f(a))$? Well, $P(f(a), f(a)) \rightarrow P(f(a), a)$ by clause 1. Then use clause 3 to infer $\neg P(f(a), f(a))$ whence $\neg P(f(a), f(a))$. But then clause 2 tells us that $P(f(a), f(a))$ after all.

The final part.

Three clauses:

$$\{\neg P(x, x), P(x, a)\}, \quad \{P(x, x), P(x, f(a))\} \quad \{\neg P(y, f(x)), \neg P(y, x)\}$$

I think this is Russell's paradox. If we take $P(x, y)$ to be $x \in y$; a to be the complement of the Russell class, and f to be complementation then all three causes (universally quantified) come out true—at least on the assumption that there is a Russell class!

In the third clause make the substitutions a/x and $f(a)/y$ to obtain

$$\{\neg P(f(a), f(a)), \neg P(f(a), a)\}$$

In the first clause make the substitution $f(a)/x$ to obtain

$$\{\neg P(f(a), f(a)), P(f(a), a)\}$$

and resolve these two on $P(f(a), a)$ to get

$$\{\neg P(f(a), f(a))\}.$$

In the second clause make the substitution $f(a)/x$ to get

$$\{P(f(a), f(a)), P(f(a), f(a))\}$$

which is of course

$$\{P(f(a), f(a))\}$$

There is a subtlety here that I should understand but don't. We can resolve $\{\neg P(f(a), f(a))\}$ with $\{P(x, x), P(x, f(a))\}$ with the substitution $f(a)/x$ to get...Do we get $\{P(f(a), f(a))\}$ (in which case we can resolve with $\{\neg P(f(a), f(a))\}$ again to get the empty clause) or do we get the empty clause straight off (because the two formulæ in the second clause coalesce after substitution)? (Isn't that what they call *factoring*??) My guess is that we are supposed to do the latter, because if we do the former the result apparently can't be turned into a linear resolution.

3.5 2002:6:11

(a) If we do things properly and fail to discover a contradiction then the original formula is not a tautology. If we neglect to negate then the negation of the original formula is not a tautology, so the answer is: A is consistent.

(b) We have derived a contradiction from something that has been negated. So the thing that had been negated is valid. That thing is the skolemised version of A. So the skolemised version of A is valid. Doesn't seem to tell us anything. Skolemisation preserves satisfiability.

(c) If $\neg A$ is refutable, then it shouldn't matter which variable you choose for a case split: you should get the empty clause every time. OTOH if he means by the question that if you split on p say, and the clauses arising from p resolve to the empty clause but the clauses arising from $\neg p$ don't, then the formula is consistent.

3.6 2005:5:9

3.6.1 part (a)

In order to prove the following formula by resolution, what set of clauses should be submitted to the prover? Justify your answer briefly

$$\forall x[(P(x) \vee Q) \to \neg R(x)] \wedge \forall x[(Q \to \neg S(x)) \to P(x) \wedge R(x)] \to \forall x S(x)$$

(Just to remind myself what I'm doing ...) If ϕ is satisfiable, so is $sk(\phi)$, the skolemisation of ϕ. So if we want to prove ϕ we investigate $\neg\phi$ and hope that it isn't satisfiable. However if it *is* satisfiable, so is $sk(\neg\phi)$. So we put $sk(\neg\phi)$ into clause form and hope to find a contradiction.

The negation of ϕ is the conjunction of

$$\forall x[(P(x) \vee Q) \to \neg R(x)] \tag{1}$$

$$\forall x[(Q \to \neg S(x)) \to P(x) \wedge R(x)] \tag{2}$$

$$\exists x \neg S(x) \tag{3}$$

On skolemising we obtain

$$(P(x) \vee Q) \to \neg R(x) \tag{1}$$

$$(Q \to \neg S(y)) \to P(y) \wedge R(y) \tag{2}$$

$$\neg S(a) \tag{3}$$

(1) gives us the two clauses $\{\neg P(x), \neg R(x)\}$ and $\{\neg Q, \neg R(x)\}$;
(3) obviously gives us the clause $\{\neg S(a)\}$
(2) gives us the two expressions $(Q \to \neg S(y)) \to P(y)$ and $(Q \to \neg S(y)) \to P(y)R(y)$. The first becomes

$$\neg(Q \to \neg S(y)) \vee P(y)$$

which becomes

$$(Q \wedge S(y)) \vee P(y)$$

which becomes

$$(Q \vee P(y)) \wedge (S(y) \vee P(y))$$

which gives us the two clauses

$$\{Q, P(y)\} \text{ and } \{S(y), P(y)\}.$$

The second differs from the first only in having 'R' instead of 'S' and so we get the two clauses

$$\{Q, R(y)\} \text{ and } \{S(y), R(y)\}.$$

3.6.2 part (b)

The third clause says that P is transitive, and the first clause says that P is irreflexive. P is starting to look like $<$ on the naturals. It looks even more like that when you reflect that clause two makes sense if you think of f as successor.

3.6.3 part (c)

Resolve $\{Q(a)\}$ with $\{\neg Q(a), P(a), \neg R(y), \neg Q(y)\}$ to obtain

$$\{P(a), \neg R(y), \neg Q(y)\}.$$

Next resolve this with $\{\neg P(a)\}$ to obtain

$$\{\neg R(y), \neg Q(y)\}.$$

Next resolve this with $\{R(b)\}$ to obtain

$$\{\neg Q(b)\}.$$

Next resolve this with $\{\neg S(b), \neg R(b), Q(b)\}$ to obtain

$$\{\neg R(b), \neg S(b)\}.$$

Next resolve this with $\{S(b)\}$ to obtain

$$\{\neg R(b)\}.$$

Resolve this with $\{R(b)\}$ to obtain the empty clause.

And it was Input Resolution all the way!!

3.7 2009:6:8

We are given four clauses:

$\{\neg R(x,a), R(x,x)\}$ $\{\neg R(x,x), R(x,a)\}$
$\{\neg R(y, f(x)), \neg R(y,x)\}$ $\{R(y,x), R(y, f(x))\}$

The two clauses in the top row say $(\forall x)(R(x,a) \longleftrightarrow R(x,x))$; the two clauses in the second row say $(\forall xy)(R(y,x) \longleftrightarrow \neg R(x, f(x)))$

The way to crack this (and i admit i can't think of a reason why this should be obvious to you) is to spot the similarity of the first line to the definition of the Russell class. If you think of $R(x,y)$ as saying $x \in y$ then the two clauses in the first row simply say that a is the set $\{x : x \in x\}$. This is not a paradoxical object (try it!) but its complement is the Russell class which most definitely is paradoxical.

Now that we have decided that R is $\in$ the clauses in the second row make sense too: $f(x)$ is clearly just the complement of x. Now, as remarked, the complement of a (which is $f(a)$) is the Russell class. We obtain a contradiction by asking whether or not it is a member of itself. Clearly these clauses are going to resolve to the empty clause—resolution is complete, after all—all that remains is to do it.

How do we obtain a contradiction from the assumption that the Russell class is a set? If we can remember how the proof works it will make it easier to see how to resolve these clauses to the empty clause. We show that if the Russell class is a member of itself then it isn't, and if it isn't then it is. Clearly, substituting '$f(a)$' into these formulæ is going to help, and formulæ like '$R(f(a), a)$' (which says that the Russell class is not a member of itself) will loom large.

Let's resolve $\{\neg R(x, a), R(x, x)\}$ with $\{R(y, x), R(y, f(x))\}$. First reletter all the 'x's in the first clause to 'z's (to prevent us all going crazy) getting $\{\neg R(z, a), R(z, z)\}$. Then we can unify with $z \mapsto y$ and $x \mapsto a$, and resolve to obtain $\{R(y, a), R(y, f(a))\}$.

Similarly we can resolve $\{\neg R(x, x), R(x, a)\}$ with $\{\neg R(y, f(x)), \neg R(y, x)\}$ to obtain $\{\neg R(y, y), \neg R(y, f(a))\}$. This is clearly going to give us $\neg R(f(a), f(a))$, so we should look for something that will give us $R(f(a), f(a))$

Clearly the correct thing to do is to copy Ricky Jones' answer (which he arrived at without all the psychology)
He numbers the clauses:

(i) $\{\neg R(x, x), R(x, a)\}$

(ii) $\{\neg R(x, a), R(x, x)\}$

(iii) $\{\neg R(y, f(x)), \neg R(y, x)\}$

(iv) $\{R(y, x), R(y, f(x))\}$

He says: in (i) do $x \mapsto f(a)$ and in (iv) do $y \mapsto f(a)$. Then cut to obtain $\{R(f(a), a)\}$.

Then (ii) do $x \mapsto f(a)$ and in (iii) do $y \mapsto f(a)$ and resolve to obtain $\{\neg R(f(a), a)\}$.

3.8 2010:6:6

Part (b)

$$\frac{\Gamma \vdash \Delta, A, B \qquad \Gamma, A, B \vdash \Delta}{\Gamma \vdash \Delta, A \oplus B}$$

If we consider the case where Γ and Δ are both empty the sequent rule becomes

$$\frac{\vdash A, B \qquad A, B \vdash}{\vdash A \oplus B}$$

Now one thing one can infer from the top line is A `XOR` B, since the left upper sequent says at least one of A and B is true, while the right upper sequent says at least one of A and B is false. Could it be anything stronger? $A \wedge B$ for example? $\neg A \wedge \neg B$? These are the only candidates—$A \oplus B$ has to be one of the 16 boolean binary connectives, and it's easy to check that nothing else will do.

So $A \oplus B$ is A `XOR` B.

To discover what the rule on the left is for A `XOR` B one could try looking for a proof of the sequent $\Gamma, \neg A \to B, \neg B \to A \vdash \Delta$, and seeing what sequents one finds at the top of one's tree once one runs out of connectives. Another approach is to observe that A `XOR` B is symmetric in 'A' and 'B' so the upper sequents in a `XOR`$-L$ must be preserved by swapping 'A' and 'B'. One tries various things like

$$\frac{\Gamma \vdash \Delta, A, B \qquad \Gamma, A, B \vdash \Delta}{\Gamma,\ A \texttt{ XOR } B \vdash \Delta}$$

but i think what one wants is

$$\frac{\Gamma, A \vdash \Delta, B \qquad \Gamma, B \vdash \Delta, A}{\Gamma,\ A \texttt{ XOR } B \vdash \Delta}$$

and it's a simple matter to check that this is truth-preserving. To do this, assume the upper sequents, assume Γ and assume that precisely one of A, B is true. If it's A that is true we invoke the left upper sequent and infer that either B is true or something in Δ is true. Well, *ex hypothesi* it ain't B wot is true, so it must be something in Δ (as desired); OTOH if it's B that is true we invoke the right upper sequent and infer that either A is true or something in Δ is true. Well, *ex hypothesi* it ain't A wot is true, so it must be something in Δ (as desired).

I think that requires quite a lot of work for 6 marks, but perhaps i'm getting soft in my old age. (It certainly took me more than 6 minutes.)

3.9 2016, Paper 6, Question 5

An answer from Dmitry Kazhdan

3.9.1 Part (a)

- A literal is an atomic formula, or its negation.
- A clause is a disjunction of literals: $\neg K_1 \lor \neg K_2 \lor ... \lor \neg K_m \lor L_1 \lor ... \lor L_n$
- Written as a set: $\{\neg K_1, \neg K_2, ..., A_1, A_n\}$.
- A clause is true just when one of its literals is true. Thus the empty clause indicates a contradiction.
- One way of proving theorems is to take the formula, negate it, convert it to CNF and convert it to a set of clauses.
- A method must then attempt to find a contradiction in this set of clauses in order to prove that they are unsatisfiable, i.e. to refute the set of clauses. This will imply that the original formula is valid.

The DPLL algorithm can either refute a set of clauses or exhibit a model if one exists. The algorithm presented here is one for propositional logic only.
Given a set of clauses:

1) Delete all tautological clauses containing $\{P, \neg P ...\}$.
2) Perform unit propagation: for each unit clase { L },
—Delete all clauses containing L.
—Delete $\neg L$ from all clauses.
—Keep note that L is assumed to be true.
3) Delete all clauses containing pure literals, i.e. a literal that does not appear in negated form in any clause. Keep note that L is assumed to be true.
4) If the empty clause is generated, then the refutation was successful.
If all clauses have been deleted, then the set is satisfiable. In that case exhibit a model:
—Use the set of recorded assignments to assign values to the literals.
—Literals without any associated assignments can take on any value.

5) If none of the above apply, perform a case split on some literal L, and recursively apply the algorithm to the two subcases: L and $\neg L$. The set is satisfiable iff one of the sub-cases is satisfiable.

Applied to the set: $\{P, Q, \neg R\}, \quad \{\neg P, R\}, \quad \{\neg Q\}, \quad \{P, R\}$
Unit propagation of $\{\neg Q\}$ (keep track that $Q \mapsto 0$) : $\{P, \neg R\}, \quad \{\neg P, R\}, \quad \{P, R\}$
Case split on P:
Case P is true (keep track that $P \mapsto 1$):
Delete all clauses containing P and delete $\neg P$ from all clauses: $\{R\}$
Unit propagation of $\{R\}$ (keep track that $R \mapsto 1$): all clauses deleted.
Hence for $P \mapsto 1, \quad Q \mapsto 0, \quad R \mapsto 1$, we have a model satisfying the clauses.

3.9.2 Part (b)

i) Clauses: $\{\neg P(x), Q(x,x)\}, \{\neg Q(x,y), \neg Q(y,x), R(x,y)\}, \{\neg R(x,y), \neg R(y,x)\}, \{P(a), P(b)\}$

Derive the following:

$$\dfrac{\dfrac{\{\neg P(x), Q(x,x)\} \qquad \{P(a), P(b)\}}{\{Q(a,a), P(b)\}} \qquad \dfrac{\dfrac{\{\neg Q(x,y), \neg Q(y,x), R(x,y)\}}{\{\neg Q(a,a), R(a,a)\}} \qquad \dfrac{\{\neg R(x,y), \neg R(y,x)\}}{\{\neg R(a,a)\}}}{\{\neg Q(a,a)\}}}{\{P(b)\}}$$

Also, derive:

$$\dfrac{\dfrac{\{\neg P(x), Q(x,x)\}}{\{\neg P(b), Q(b,b)\}} \qquad \dfrac{\dfrac{\{\neg Q(x,y), \neg Q(y,x), R(x,y)\}}{\{\neg Q(b,b), R(b,b)\}} \qquad \dfrac{\{\neg R(x,y), \neg R(y,x)\}}{\{\neg R(b,b)\}}}{\{\neg Q(b,b)\}}}{\{\neg P(b)\}}$$

Using these to obtain:

$$\dfrac{\{\neg P(b)\} \qquad \{P(b)\}}{\Box}$$

ii) Clauses: $\{P(x), Q(x)\}, \{\neg P(x), Q(f(x))\}, \{P(x), \neg Q(f(x))\}, \{\neg P(x), \neg Q(x)\}$

Consider the interpretation, where the language L has predicates { P, Q }, and has a function f. The constants used by the language are not specified in the question. However, they are not used in the clauses and hence are immaterial. The interpretation does not mention them.

Consider the interpretation I:

Domain = $\{a, b\}$

$I[P](a) \to 1$, $I[Q](a) \to 0$, $I[f](a) \to b$, $I[P](b) \to 0$, $I[Q](b) \to 1$, $I[f](b) \to a$.

This interpretation satisfies: $\{P(x), Q(x)\}, \{\neg P(x), Q(f(x))\}, \{P(x), \neg Q(f(x))\}, \{\neg P(x), \neg Q(x)\}$.

Thus it is a suitable model.

3.10 2015, Paper 6, Question 5

An answer from Dmitry Kazhdan

3.10.1 Part (a)

Resolution has a main resolution ground rule, which takes two clauses and creates a new one:

$$\frac{\{B, A_1, ..., A_m\} \qquad \{\neg B, C_1, ..., C_m\}}{\{A_1, ..., A_m, C_1, ..., C_m\}}$$

This works because if two clauses have the same literal, with one of them having it in negated form, then either one of the sets of remaining literals in the two clauses must be true.

In FOL we also get the binary resolution rule:

$$\frac{\{B, A_1, ..., A_m\} \qquad \{\neg D, C_1, ..., C_m\}}{\{A_1, ..., A_m, C_1, ..., C_m\}\sigma} \; if \; B\sigma \; = \; D\sigma$$

Here σ is a substitution, unifying B and D. It is applied to all remaining literals.
This rule works because it takes an instance of each clause. The instances are valid because the clauses are assumed to be universally valid. The propositional ground rule is then applied.

Finally, there is also the factoring rule:

$$\frac{\{B_1, ..., B_k, A_1, ..., A_m\}}{\{B_1, A_1, ..., A_m\}\sigma} \; if \; B_1\sigma \; = \; ... \; = \; B_k\sigma$$

Factoring takes a clause and unifies some literals with it. It works for the same reason of universal validity as the previous rule. Factoring is necessary, because resolution by itself tends to make clauses longer and longer. Our goal, however, is to reach the empty clause. Thus if every clause has at least two literals, then the only way to reach the empty clause is with the use of factoring, because otherwise any resolution will generate a new clause that is at least two literals in length as well.

3.10.2 Part (b)

i) Clauses: $\{P, \neg Q(a), \neg Q(b), R(a)\}, \{\neg P, Q(x), R(b)\}, \{\neg R(b), \neg R(x)\}$
Assume we have a language L which has predicates { P, Q, R }, no functions and constant symbols { a, b }.
Now, consider an interpretation I with domain { a, b }:

$I[a] = a, I[b] = b, I[P] \to 0, I[R](a) \to 1, I[R](b) \to 0$. The other bits of the interpretation do not matter. The bits mentioned are enough to produce a model that will satisfy all of the clauses.

ii)
Clauses: $\{\neg P(x, y), Q(x, y, f(x, y))\}, \{\neg R(y, z), Q(a, y, z)\}, \{R(y, z), \neg Q(a, y, z)\}$, $\{P(x, g(x)), Q(x, g(x), z)\}, \{\neg R(x, y), \neg Q(x, w, z)\}$

Resolve:

$$\dfrac{\dfrac{\{\neg P(x_1, y_1), Q(x_1, y_1, f(x_1, y_1))\} \qquad \{R(y_2, z_1), \neg Q(a, y_2, z_1)\}}{\{\neg P(a, y_1), R(y_1, f(a, y_1))\}}\,[a/x_1, y_1/y_2, f(a, y_1)/z_1]}{\{\neg P(a, g(a)), R(g(a), f(a, g(a)))\}}$$

$$\dfrac{\dfrac{\{P(x_1, g(x_1)), Q(x_1, g(x_1), z_1)\} \qquad \{R(y_1, z_2), \neg Q(a, y_1, z_2)\}}{\{P(a, g(a)), R(g(a), z_1))\}}\,[a/x_1, g(a)/y_1, z_1/z_2]}{\{P(a, g(a)), R(g(a), f(a, g(a)))\}}$$

$$\dfrac{\{P(a, g(a)), R(g(a), f(a, g(a)))\} \qquad \{\neg P(a, g(a)), R(g(a), f(a, g(a)))\}}{\{R(g(a), f(a, g(a)))\}}$$

$$\dfrac{\{P(x_1, g(x_1)), Q(x_1, g(x_1), z_1)\} \qquad \{\neg R(x_2, y_1), \neg Q(x_2, w_1, z_2)\}}{\{P(x_1, g(x_1)), \neg R(x_1, y_1)\}}\,[x_1/x_2, g(x_1)/w_1, z_1/z_2]$$

$$\dfrac{\{\neg P(x_1, y_1), Q(x_1, y_1, f(x_1, y_1))\} \qquad \{\neg R(x_2, y_2), \neg Q(x_2, w_1, z_1)\}}{\{\neg P(x_1, y_1), \neg R(x_1, y_2)\}}\,[x_1/x_2, y_1/w_1, f(x_1, y_1)/z_1]$$

$$\dfrac{\dfrac{\{\neg P(x_1, y_1), \neg R(x_1, y_2)\} \qquad \{P(x_2, g(x_2)), \neg R(x_2, y_3)\}}{\{\neg R(x_1, y_2), \neg R(x_1, y_3)\}}\,[x_1/x_2, g(x_1)/y_1]}{\{\neg R(g(a), f(a, g(a)))\}}$$

$$\dfrac{\{\neg R(g(a), f(a, g(a)))\} \qquad \{R(g(a), f(a, g(a)))\}}{\Box}$$

Hence clauses always inconsistent.

3.11 2016, Paper 6, Question 6

An answer from Dmitry Kazhdan

3.11.1 Part (a)

-SMT refers to the problem of determining whether a first-order formula is satisfiable with respect to some logical theory.
-Some applications include: planning and constraint solving, hardware and software verification.
-SMT typically uses decision procedures for theories, and uses DPLL methods for logical reasoning.

3.11.2 Part (b)

Fourier-Motzkin variable elimination is a decision procedure for real or rational linear arithmetic. It deals with conjunctions of linear constraints over the reals or rationals:

$$\bigwedge_{i=1}^{m} \sum_{j=1}^{n} a_{ij}x_j \leq b_i$$

Variables are eliminated in succession until a contradiction is reached, or a trivial constraint remains.

To eliminate a variable, every combination of its lower bound with an upper bound is formed. This is done by rewriting all constraints that have the variable x in the form $x \leq$`expr` or $-x \leq$ `expr` and then summing all the ones with opposite signs pairwise. This generates a new set of constraints that don't contain x. The constraints in the initial set without x are kept as well. If x has only one bound, then constraints containing it can be disregarded. All variables are eliminated this way.

Contraints: $x + z \geq 5, \quad y + z \geq 5, \quad y - 2z \geq -2, \quad x + y + z \leq 7$

Begin with x: $-x \leq z - 5, \quad x \leq 7 - y - z$
Form new set of constraints: $0 \leq 2 - y, \quad y + z \geq 5, \quad y - 2z \geq -2$
Eliminate y: $y \leq 2, \quad -y \leq z - 5, \quad -y \leq 2 - 2z$

Form new set of constraints: $0 \leq z - 3, \quad 0 \leq 4 - 2z$
Eliminate z: $-z \leq -3, \quad z \leq 2$

Form new set of constraints: $0 \leq -1$

All variables have been eliminated, and the inequality obtained is certainly not true.
Hence we can conclude that the constraints are unsatisfiable.

3.12 2015, Paper 6, Question 6

An answer from Dmitry Kazhdan

3.12.1 Part (a), (i)

Converting A to DNF form yields: $C_1 \vee C_2 \vee ... \vee C_m$, where every C_i is a conjunction of one or more literals. For A to hold, at least one of C_i must hold.
We can thus adopt an iterative approach, checking satisfiability of every C_i, for $1 \leq i \leq m$.
The first such C_i will give us a model. If none of them are satisfiable, then A is unsatisfiable.
Every C_i is of the form $L_1 \wedge ... \wedge L_n$, where every L_i is a literal.
If $\neg C_i$ is a tautology, then C_i is unsatisfiable. Hence we can apply DPLL to C_i.
This can be done by applying DPLL to the set of clauses: $\{L_1\}, \{L_2\}, ..., \{L_n\}$ consisting of literals of C_i.
Note that all clauses will be unit clauses, hence no case splitting is necessary. It will just be iterative unit propagation application.
If a model is returned, then C_i is satisfiable, and then so is A.
Doing this for every C_i will allow us to exhibit a model if there is one.

ii) In case of a BDD, the truth of A is represented by binary decisions (if-then-else expressions) over the propositional letters.

A BDD is a directed, acyclic graph sharing identical subtrees.
With a BDD, if A is unsatisfiable, then the BDD will just be a 0.
Otherwise, it will contain at least one path from the root node to a leaf containing a 1.
Following this path from the root to this leaf and mapping the propositional letters to appropriate truth values, based on the branch followed, will give us a model of when A holds.
Letters excluded from the tree can be set to arbitrary values.

-DNF is easier to store and handle in a computer, because BDDs require handling of binary trees and their traversal.
-Bad choice of variable ordering can lead to exponentially-sized BDDs.
-However, canonical forms of BDDs allow faster checking of equivalence of expressions.
-BDD representation is also often more compact.

3.12.2 Part (b)

i) Statement: $[\forall x\, \exists y\, Q(x,y)] \vdash \exists y Q(y,y)$

Consider the following interpretation:
Language L has the predicate set { Q }. We do not care about its functions or constants, since they are not used.
Consider an interpretation I with domain = { a, b }, and the following predicate mappings:
$I[Q](a,b) \to 1, \quad I[Q](b,a) \to 1, \quad I[Q](a,a) \to 0, \quad I[Q](b,b) \to 0$

This is a suitable falsifying interpretation.

ii) Statement: $[\forall x\,(P(x) \to \neg P(x))] \wedge [\exists y\, P(y)] \to \exists y\, Q(y)$

Using sequent calculus:

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{P(y) \vdash Q(z),\, P(y)}\,(\mathit{basic}) \qquad \dfrac{\dfrac{}{P(y) \vdash Q(z),\, P(y)}\,(\mathit{basic})}{\neg P(y),\, P(y) \vdash Q(z)}\,(\neg l)}{P(y) \to \neg P(y),\; P(y) \vdash Q(z)}\,(\to l)}{\forall x\,(P(x) \to \neg P(x)),\; P(y) \vdash Q(z)}\,(\forall l)}{\forall x\,(P(x) \to \neg P(x)),\; P(y) \vdash \exists y\, Q(y)}\,(\exists r)}{\forall x\,(P(x) \to \neg P(x)),\, \exists y\, P(y) \vdash \exists y\, Q(y)}\,(\exists l)}{[\forall x\,(P(x) \to \neg P(x))] \wedge [\exists y\, P(y)] \vdash \exists y\, Q(y)}\,(\wedge l)}{\vdash [\forall x\,(P(x) \to \neg P(x))] \wedge [\exists y\, P(y)] \to \exists y\, Q(y)}\,(\to r)
$$

iii)
Statement: $\Box(A \vee B) \to (\Diamond\Box\neg A \to \Diamond\Box B)$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A \vdash B,\ A}\,(basic) \qquad \dfrac{}{B \vdash B,\ A}\,(basic)}{(A \lor B) \vdash B,\ A}\,(\lor l)}{(A \lor B),\ \neg A \vdash B}\,(\neg l)}{\Box(A \lor B),\ \neg A \vdash B}\,(\Box l)}{\Box(A \lor B),\ \Box\neg A \vdash B}\,(\Box l)}{\Box(A \lor B),\ \Box\neg A \vdash \Box B}\,(\Box r)}{\Box(A \lor B),\ \Box\neg A \vdash \Diamond\Box B}\,(\Diamond r)}{\Box(A \lor B),\ \Diamond\Box\neg A \vdash \Diamond\Box B}\,(\Diamond l)}{\Box(A \lor B) \vdash \Diamond\Box\neg A \to \Diamond\Box B}\,(\to r)}{\vdash \Box(A \lor B) \to (\Diamond\Box\neg A \to \Diamond\Box B)}\,(\to r)$$

Chapter 4

Answers to Larry's exercises

Some of these answers are by Dave Tonge. Not all (most of them are by me) and some of his have been mutilated by me.

4.0.1 Exercise 1 (p 5)

Is the formula $A \to \neg A$ satisfiable? Is it valid?

The case where A is false satisfies. The case where A is true does not satisfy. Therefore the expression is satisfiable but not valid.

4.0.2 Exercise 3 (p 5)

Each of the following formulae is satisfiable but not valid. Exhibit a truth assignment that makes the formula true and another truth assignment that makes the formula false.
$P \to Q$

True for $P =$ true and $Q =$ true. False for $P =$ true and $Q =$ false.

$P \vee Q \to P \wedge Q$

True for $P = Q =$ true. False for $P =$ true and $Q =$ false.

$\neg(P \vee Q \vee R)$

True for $P = Q = R = false$. False otherwise.

$\neg(P \wedge Q) \wedge \neg(Q \vee R) \wedge (P \vee R)$

True for $P =$ true and $Q = R =$ false. False for $P = Q = R =$ true.

4.0.3 Exercise 4 (p 5)

Convert each of the following propositional formulæ into Conjunctive Normal Form and also into Disjunctive Normal Form. For each formula, state whether it is valid, satisfiable, or unsatisfiable; justify each answer.

(i) $(P \to Q) \wedge (Q \to P)$

To obtain CNF we first eliminate → to get $(\neg P \vee Q) \wedge (\neg Q \vee P)$.

To obtain DNF we first eliminate → to get $(\neg P \vee Q) \wedge (\neg Q \vee P)$. Push in conjunctions to get $(\neg P \wedge (\neg Q \vee P)) \vee (Q \wedge (\neg Q \vee P))$. And again to get $(\neg P \wedge P) \vee (\neg P \wedge \neg Q) \vee (Q \wedge \neg Q) \vee (Q \wedge P)$. Remove those which are obviously false to get $(\neg P \wedge \neg Q) \vee (P \wedge Q)$.

This formula is satisfiable—it is satisfied when $P = Q$.

(ii) $((P \wedge Q) \vee R) \wedge (\neg((P \vee R) \wedge (Q \vee R)))$

Both CNF and DNF require one to push in negations to get

$$((P \wedge Q) \vee R) \wedge (\neg(P \vee R) \vee \neg(Q \vee R))$$

and then

$$((P \wedge Q) \vee R) \wedge ((\neg P \wedge \neg R) \vee (\neg Q \wedge \neg R)).$$

To get CNF push in disjunctions to get

$$(P \vee R) \wedge (Q \vee R) \wedge (\neg P \vee \neg Q) \wedge (\neg P \vee \neg R) \wedge (\neg R \vee \neg Q) \wedge (\neg R \vee \neg R)$$

which is

$$(P \vee R) \wedge (Q \vee R) \wedge (\neg P \vee \neg Q) \wedge (\neg R \vee \neg Q) \wedge \neg P \wedge \neg R.$$

To get DNF push in conjunctions to get

$$(P \wedge Q \wedge \neg R \wedge \neg R) \vee (P \wedge Q \wedge \neg Q \wedge \neg R) \vee (R \wedge \neg P \wedge \neg R) \vee (R \wedge \neg Q \wedge \neg R).$$

The formula is unsatisfiable—if you look at it in DNF each conjunct has an atom in both negated and unnegated form so all conjuncts must be false so the whole disjunction is always false.

(iii) $\neg(P \vee Q \vee R) \vee ((P \wedge Q) \vee R)$

Both CNF and DNF require one to push in negations to get $(\neg P \wedge \neg Q \wedge \neg R) \vee ((P \wedge Q) \vee R)$.

To get CNF we need to push in disjunctions to get
$(\neg P \wedge \neg Q \wedge \neg R) \vee ((P \vee R) \wedge (Q \vee R))$
then
$((\neg P \wedge \neg Q \wedge \neg R) \vee (P \vee R)) \wedge ((\neg P \wedge \neg Q \wedge \neg R) \vee (Q \vee R))$
and then
$(\neg P \vee P \vee R)) \wedge (\neg Q \vee P \vee R) \wedge (\neg R \vee P \vee R) \wedge (\neg P \vee Q \vee R) \wedge (\neg Q \vee Q \vee R) \wedge (\neg R \vee Q \vee R)$ which might as well be $(\neg Q \vee P \vee R) \wedge (\neg P \vee Q \vee R)$.

To get DNF we don't have to do much except expand brackets to $(\neg P \wedge \neg Q \wedge \neg R) \vee (P \wedge Q) \vee R$.

This is satisfiable—it is only false for $P = R =$ false, $Q =$ true and $P =$ true, $Q = R =$ false.

(iv) $\neg(P \vee Q \to R) \wedge (P \to R) \wedge (Q \to R)$

Both CNF and DNF need one to get rid of →s to give

$$\neg(\neg(P \vee Q) \vee R) \wedge (\neg P \vee R) \wedge (\neg Q \vee R).$$

Push in negations to get $((P \vee Q) \wedge \neg R) \wedge (\neg P \vee R) \wedge (\neg Q \vee R)$.

We would appear to have the CNF already—$(P \vee Q \vee \neg R) \wedge (\neg P \vee R) \wedge (\neg Q \vee R)$.

To get DNF we need to push in conjunctions to get $(P \vee Q \vee \neg R) \wedge (\neg P \vee \neg Q) \wedge (\neg P \vee R) \wedge (R \vee \neg Q) \wedge R$. Again to give $(P \vee Q \vee R) \wedge (P \vee R) \wedge (\neg P \vee Q \vee R) \wedge (Q \vee R) \wedge (\neg P \vee \neg Q \vee \neg R)$.

This is satisfiable—for example in the case $P = Q =$ false, $R =$ true but it can be false as it is when $P = Q = R =$ false.

4.0.4 Exercise 9 (p 11)

Describe a formula that is true in every domain that contains at least m elements. Write down a formula that is true in every domain that contains at most most m elements.

At least m:

$$(\exists x_1 \ldots x_m)(\bigwedge_{k \neq j} (x_j \neq x_k)^{1}$$

[1] The temptation to write this as: $(\exists a_1 \ldots a_m)(\forall j, k < m)(k \neq j \to a_j \neq a_k)$ must be resisted. This is *not* correct, since the subscripts on the variables are not themselves variables and cannot be bound.

An answer for the next is obviously obtainable by increasing m by one and negating!

At most m:

$$(\forall x_1 \ldots x_{m+1})(\bigvee_{i \neq j < m} x_j = x_i)$$

Many readers find the following more natural

$$(\exists x_1 \ldots x_m)(\forall y)(\bigvee_{1 \leq i \leq m} y = x_i)$$

This formula is logically more complicated (it has an alternation of quantifiers) but is shorter.

A brief question to ask yourself: how rapidly does the formula grow with n?

Important to make the point that the string i wrote above is not a first-order formula. For one thing it exploits the fact that variables have internal structure; first-order logic will not let us do this. How should we think of it? As a syntactic proto-object that evaluates to the formula? As a description of a formula? Add water/light blue touch-paper? One way of thinking of it is as a *program* that *evaluates to* a first-order formula.

4.0.5 Exercise 10 (p 11)

Let $\sim$ be a 2-place predicate symbol, which we write using infix notation: for instance, $x \sim y$ rather than $\sim (x, y)$. Consider the following axioms:

$$(\forall x) \qquad x \sim x \tag{4.1}$$

$$(\forall xy) \qquad (x \sim y \to y \sim x) \tag{4.2}$$

$$(\forall xyz) \qquad (x \sim y \wedge y \sim z \to x \sim z) \tag{4.3}$$

Let the universe be the set of natural numbers, $\mathbb{N} = \{0, 1, 2, \cdots\}$*. Which axioms hold if the interpretation of* $\sim$ *is...*

1. *the empty relation,* ϕ*?*

   (1) does not hold. (2), (3) hold.

   Notice that the empty relation on an empty set **is** *reflexive!!*

2. *the universal relation,* $\{(x, y) : x, y \in \mathbb{N}\}$*?*

   (1), (2), (3) all hold.

3. *the relation* $\{(x, x) : x \in \mathbb{N}\}$*?*

   (1), (2), (3) all hold.

4. *the relation* $\{(x, y) : x, y \in \mathbb{N} \wedge x + y\text{is even}\}$

   (1), (2), (3) all hold.

5. *the relation* $\{(x, y) : x, y \in \mathbb{N} \wedge x + y = 100\}$*?*

   (1), (3) do not hold. (2) holds.

6. *the relation* $\{(x, y) : x, y \in \mathbb{N} \wedge x = y \ (mod\, 16)\}$*?*

   (1), (2), (3) all hold.

4.0.6 Exercise 11 (p 11)

Taking $\sim$ and R as 2-place relation symbols, consider the following axioms:

$$\begin{array}{rl}
(\forall x) & \neg R(x,x) \\
(\forall xy) & \neg(R(x,y) \wedge R(y,x)) \\
(\forall xyz) & (R(x,y) \wedge R(y,z) \rightarrow R(x,z)) \\
(\forall xy) & (R(x,y) \vee x = y \vee R(y,x)) \\
(\forall xz) & (R(x,z) \rightarrow (\exists y)(R(x,y) \wedge R(y,z)))
\end{array}$$

Exhibit two interpretations that satisfy axioms 1-3 and falsify axioms 4 and 5. Exhibit two interpretations that satisfy axioms 1-4 and falsify axiom 5. Exhibit two interpretations that satisfy axioms 1-5. Consider only interpretations that make = denote the equality relation.

1-3 true, 4 and 5 false.

Domain is sets of natural numbers. $\hat{R}$ is $\subseteq$ or $\subset$ (strict subset) or $\supseteq$ or $\supset$ (strict superset).

Another cute example (due to Loretta He) is: Domain is $\mathbb{N}$, and the relation is $\{\langle x,y\rangle : x+1<y\}$. The converse of this relation will do too, of course.

1-4 true, 5 false.

Domain is natural numbers. $\hat{R}$ is $<$ or $>$ or $\leq$ or $\geq$.

1-5 true.

Domain is real numbers. $\hat{R}$ is $<$ or $>$ or $\leq$ or $\geq$.

4.0.7 Exercise 12 (p 13)

Verify these equivalences by appealing to the truth definition for first order logic.

There are too many of these, so I'll just do the infinitary de Morgan law $\mathcal{M}_V \models \neg((\forall x)A) = ((\exists x)\neg A)$.

To show this we have to show that $\mathcal{M}_V \models \neg((\forall x)A)$ is equivalent to $\mathcal{M}_V \models ((\exists x)\neg A)$.

The first half becomes: for all $m \in M$ such that $\mathcal{M}_{V\{m/x\}} \models A$ does not hold. The second half becomes there exists an $m \in M$ for which $\mathcal{M}_{V\{m/x\}} \models A$ does not hold.

These two are plainly equivalent for if the first one does not hold then there there will not exist an m for which the second holds. Similarly, if the second is true then the first will not hold for all ms (for it won't hold for the m given by the first).

4.0.8 Exercise 13 (p 13)

Explain why the following are not equivalences. Are they implications? In which direction?

$((\forall x)A) \vee ((\forall x)B)? =?(\forall x)(A \vee B)$

$((\exists x)A) \wedge ((\exists x)B)? =?(\exists x)(A \wedge B)$

First one: The RHS could be true if A were true and B were false for a particular x. Thus, B would not be true for all x. There might be another x for which A were false and B true. Thus A is not true for all x. Thus the LHS can be false although the RHS is true, so the two statements are not equivalent. However, there is a left-to-right implication.

Second: The x for which A might not be the same as the x for which B. Therefore the RHS will could be false even if the LHS is true. The two statements are not equivalent. However, the RHS implies the LHS.

4.0.9 Exercise 15 (p 13)

Third part:

$$
\begin{array}{rcll}
P(f(a)) \to P(f(f(a))), P(a) \to P(f(a)), P(a) & \vdash & P(f(f(a)))) & \\ \hline
(\forall x)(P(x) \to P(f(x))), P(a) \to P(f(a)), P(a) & \vdash & P(f(f(a)))) & \forall\ \text{L} \\ \hline
(\forall x)(P(x) \to P(f(x))), P(a) & \vdash & P(f(f(a)))) & \forall\ \text{L} \\ \hline
(\forall x)(P(x) \to P(f(x))) & \vdash & P(a) \to P(f(f(a)))) & \to \text{R} \\ \hline
(\forall x)(P(x) \to P(f(x))) & \vdash & (\forall x)(P(x) \to P(f(f(x)))) & \forall\ \text{R}
\end{array}
\tag{4.4}
$$

. . . after which two →-L will do it. I've omitted them to save space.

4.0.10 Exercise 17 (p 13)

Third part:

$$
\begin{array}{rcll}
P(a), P(b) & \vdash & P(a) \wedge P(b), P(b) & \\ \hline
P(a), P(b) & \vdash & P(a) \wedge P(b), P(a) \wedge P(b) & \wedge\ \text{R} \\ \hline
P(a), P(b) & \vdash & P(a) \wedge P(b), P(a) \wedge P(b) & \wedge\ \text{R} \\ \hline
P(b) & \vdash & P(a) \to P(a) \wedge P(b), P(a) \wedge P(b) & \to \text{R} \\ \hline
 & \vdash & P(a) \to P(a) \wedge P(b), P(b) \to P(a) \wedge P(b) & \to \text{R} \\ \hline
 & \vdash & (\exists x)(P(x) \to P(a) \wedge P(b)), P(b) \to P(a) \wedge P(b) & \exists\ \text{R} \\ \hline
 & \vdash & (\exists x)(P(x) \to P(a) \wedge P(b)), (\exists x)(P(x) \to P(a) \wedge P(b)) & \exists\ \text{R} \\ \hline
 & \vdash & (\exists x)(P(x) \to P(a) \wedge P(b)) & \text{contraction-R}
\end{array}
\tag{4.5}
$$

The first couple of lines have become garbled, but you get the idea.

4.0.11 Exercise 21 (p 17)

Prove Peirce's Law using resolution

We have to negate and eliminate →s first.

$$
\begin{gathered}
\neg(\neg(\neg(\neg P \vee Q) \vee P) \vee P) \\
\neg(\neg((P \wedge \neg Q) \vee P) \vee P) \\
\neg((\neg(P \wedge \neg Q) \wedge \neg P) \vee P) \\
\neg(((\neg P \vee Q) \wedge \neg P) \vee P) \\
(\neg((\neg P \vee Q) \wedge \neg P) \wedge \neg P) \\
((\neg(\neg P \vee Q) \wedge P) \wedge \neg P) \\
(((P \wedge \neg Q) \wedge P) \wedge \neg P) \\
P \wedge P \wedge \neg P \wedge \neg Q \\
P \wedge \neg P \wedge \neg Q
\end{gathered}
$$

This gives us the clauses $\{P\}$, $\{\neg P\}$ and $\{\neg Q\}$. Resolving $\{P\}$ with $\{\neg P\}$ gives a contradiction ($\{\}$). We have assumed the negation of Peirce's law and derived a contradiction, thus proving the law.

4.0.12 Exercise 22 (p 17)

"Prove $(P \land Q \to R) \land (P \lor Q \lor R) \to ((P \leftrightarrow Q) \to R)$ by resolution. Show the steps of converting the formula into clauses."

We have to negate and remove $\to$s first.

$$\neg((P \land Q \to R) \land (P \lor Q \lor R) \to ((P \leftrightarrow Q) \to R))$$
$$\neg(\neg((\neg(P \land Q) \lor R) \land (P \lor Q \lor R)) \lor (\neg((\neg P \lor Q) \land (\neg Q \lor P)) \lor R))$$
$$((\neg P \lor \neg Q \lor R) \land (P \lor Q \lor R)) \land \neg((P \land \neg Q) \lor (\neg P \land Q) \lor R)$$
$$(\neg P \lor \neg Q \lor R) \land (P \lor Q \lor R) \land (\neg(P \land \neg Q) \land \neg(\neg P \land Q) \land \neg R)$$
$$(\neg P \lor \neg Q \lor R) \land (P \lor Q \lor R) \land (\neg P \lor Q) \land (P \lor \neg Q) \land \neg R$$

This gives us the clauses

$\{\neg P, \neg Q, R\}$, $\{P, Q, R\}$, $\{\neg P, Q\}$, $\{\neg Q, P\}$, $\{\neg R\}$.

If we resolve the last one with the first two we get $\{\neg P, \neg Q\}$ and $\{P, Q\}$. This gives us all four possible clauses starring P and Q so we will certainly get the empty clause. ($\{\}$). We have assumed the negation of the theorem and derived a contradiction. This proves the theorem.

(This was supposed to be linear resolution)

4.0.13 Exercise 23 (p 17)

Using linear resolution, prove that $(P \land Q) \to (R \land S)$ follows from $(P \to R) \land (Q \to S)$ and $R \land P \to S$.

The two assumed formulæ and the negated conclusion give us the clauses $\{\neg P, R\}$, $\{\neg Q, S\}$ and $\{\neg R, \neg P, S\}$. We need to resolve these with the clauses given by the negation of the formula we are trying to prove. These clauses are $\{P\}$, $\{Q\}$ and $\{\neg R, \neg S\}$.

Take $\{\neg R, \neg S\}$ and resolve with $\{\neg R, \neg P, S\}$ to get $\{\neg R, \neg P\}$.

Resolve the result with $\{\neg P, R\}$ to get $\{\neg P\}$.

Resolve the result with $\{P\}$ to get a contradiction ($\{\}$). This proves the formula.

4.0.14 Exercise 24 (p 17)

Convert these axioms to clauses, showing all steps.

$\texttt{rain} \land (\texttt{windy} \lor \neg\texttt{umbrella}) \to \texttt{wet}$,
$\texttt{winterstorm} \to \texttt{storm} \land \texttt{cold}$,
$\texttt{wet} \land \texttt{cold} \to \texttt{miserable}$
and
$\texttt{storm} \to \texttt{rain} \land \texttt{windy}$.

Then prove $\texttt{winterstorm} \to \texttt{miserable}$ by resolution.

First we need to construct all our definite clauses.

$$\begin{aligned}
& (\texttt{rain} \wedge (\texttt{windy} \vee \neg\texttt{umbrella}) \rightarrow \texttt{wet} \\
\textit{expand } \wedge \quad & ((\texttt{rain} \wedge \texttt{windy}) \vee (\texttt{rain} \wedge \neg\texttt{umbrella})) \rightarrow \texttt{wet} \\
\textit{remove } \rightarrow \quad & \neg((\texttt{rain} \wedge \texttt{windy}) \vee (\texttt{rain} \wedge \neg\texttt{umbrella})) \vee \texttt{wet} \\
& (\neg(\texttt{rain} \wedge \texttt{windy}) \wedge \neg(\texttt{rain} \wedge \neg\texttt{umbrella})) \vee \texttt{wet} \\
& ((\neg\texttt{rain} \vee \neg\texttt{windy}) \wedge (\neg\texttt{rain} \vee \neg\texttt{umbrella})) \vee \texttt{wet} \\
& (\neg\texttt{rain} \vee \neg\texttt{windy} \vee \texttt{wet}) \wedge (\neg\texttt{rain} \vee \neg\texttt{umbrella} \vee \texttt{wet}) \\
\text{clauses are} \quad & \{\neg\texttt{rain}, \neg\texttt{windy},\ \texttt{wet}\} \textit{ and } \{\neg\texttt{rain}, \neg\texttt{umbrella}, \texttt{wet}\}
\end{aligned}$$

$$\begin{aligned}
& \texttt{winterstorm} \rightarrow \texttt{storm} \wedge \texttt{cold} \\
\textit{remove} \rightarrow \quad & \neg\texttt{winterstorm} \vee (\texttt{storm} \wedge \texttt{cold}) \\
\textit{expand } \wedge \quad & (\neg\texttt{winterstorm} \wedge \texttt{storm}) \wedge (\neg\texttt{winterstorm} \wedge \texttt{cold}) \\
\text{clauses are} \quad & \{\neg\texttt{winterstorm}, \texttt{storm}\} \textit{ and } \{\neg\texttt{winterstorm},\ \texttt{cold}\}
\end{aligned}$$

$$\begin{aligned}
& \texttt{wet} \wedge \texttt{cold} \rightarrow \texttt{miserable} \\
\textit{remove} \rightarrow \quad & \neg(\texttt{wet} \wedge \texttt{cold}) \vee \texttt{miserable} \\
\textit{push in } \neg \quad & \neg\texttt{wet} \vee \neg\texttt{cold} \vee \texttt{miserable} \\
\text{clauses are} \quad & \{\neg\texttt{wet}, \neg\texttt{cold},\ \texttt{miserable}\}
\end{aligned}$$

$$\begin{aligned}
& \texttt{storm} \rightarrow \texttt{rain} \wedge \texttt{windy} \\
\textit{remove} \rightarrow \quad & \neg\texttt{storm} \vee (\texttt{rain} \wedge \texttt{windy}) \\
\textit{expand}\wedge \quad & (\neg\texttt{storm} \vee \texttt{rain}) \wedge (\neg\texttt{storm} \vee \texttt{windy}) \\
\text{clauses are} \quad & \{\neg\texttt{storm},\ \texttt{rain}\} \textit{ and } \{\neg\texttt{storm},\ \texttt{windy}\}
\end{aligned}$$

In order to prove `winterstorm` → `miserable` we have to assume its negation and derive a contradiction. So, let's find out the clauses that would give us.

$$\begin{aligned}
& \texttt{winterstorm} \rightarrow \texttt{miserable} \\
\textit{negate} \quad & \neg(\texttt{winterstorm} \rightarrow \texttt{miserable}) \\
\textit{remove} \rightarrow \quad & \neg(\neg\texttt{winterstorm} \vee \texttt{miserable}) \\
\textit{push in}\neg \quad & \texttt{winterstorm} \wedge \neg\texttt{miserable} \\
\text{clauses are} \quad & \{\texttt{winterstorm}\} \textit{ and } \{\neg\texttt{miserable}\}
\end{aligned}$$

Now, using all the clauses we have gathered we need to use resolution to get a contradiction.

Resolve $\{\texttt{winterstorm}\}$ with $\{\neg\texttt{winterstorm}, \texttt{storm}\}$ to give $\{\texttt{storm}\}$.
Resolve $\{\texttt{winterstorm}\}$ with $\{\neg\texttt{winterstorm}, \texttt{cold}\}$ to give $\{\texttt{cold}\}$.
Resolve $\{\texttt{storm}\}$ with $\{\neg\texttt{storm}, \texttt{windy}\}$ to give $\{\texttt{windy}\}$.
Resolve $\{\texttt{storm}\}$ with $\{\neg\texttt{storm}, \texttt{rain}\}$ to give $\{\texttt{rain}\}$.
Resolve $\{\texttt{rain}\}$ with $\{\neg\texttt{rain}, \neg\texttt{windy}, \texttt{wet}\}$ to give $\{\neg\texttt{windy}, \texttt{wet}\}$.
Resolve $\{\texttt{windy}\}$ with $\{\neg\texttt{windy}, \texttt{wet}\}$ to give $\{\texttt{wet}\}$.
Resolve $\{\texttt{wet}\}$ with $\{\neg\texttt{wet}, \neg\texttt{cold}, \texttt{miserable}\}$ to give $\{\neg\texttt{cold}, \texttt{miserable}\}$.
Resolve $\{\texttt{cold}\}$ with $\{\neg\texttt{cold}, \texttt{miserable}\}$ to give $\{\texttt{miserable}\}$.
Resolve $\{\texttt{miserable}\}$ with $\{\neg\texttt{miserable}\}$ to give a contradiction ($\{\}$). This proves the theorem by contradiction of the negated theorem.

4.0.15 Exercise 25 (p 21)

Consider a first-order language with 0 and 1 as constant symbols, with - as a 1-place function symbol and + as a 2-place function symbol, and with = as a 2 place predicate symbol.

(a) Describe the Herbrand Universe for this language.

$$\begin{array}{rcl}
\mathcal{C} & = & \{0, 1\} \\
\mathcal{F}_1 & = & \{-\} \\
\mathcal{F}_2 & = & \{+\} \\
\mathcal{F}_n(n > 2) & = & \phi \\
\mathcal{P}_1 & = & \phi \\
\mathcal{P}_2 & = & \{=\} \\
\mathcal{P}_n(n > 2) & = & \phi \\
H_0 & = & \{0, 1\} \\
H_1 & = & \{0, 1, -(0), -(1)\} \\
H & = & \{0, 1, -(0), -(1), -(-(0)), -(-(1)), +(0, 0), +(0, 1), +(1, 0), \\
 & & +(1, 1), +(0, -(0)), +(0, -(1)), +(-(0), 0), +(-(1), 1) \cdots\} \\
HB & = & \{= (0, 0), = (0, 1), = (1, 0), = (1, 1), = (-(0), -(0)), \\
 & & = (-(1), -(0)), = (+(0, 1), +(-(1), -(0))), \cdots\}
\end{array}$$

points
gram-
his set

(b) The language can be interpreted by taking the integers for the universe and giving 0, 1, -, + and = their usual meanings over the integers. What do those symbols denote in the corresponding Herbrand interpretation?

In the interpretation = is interpreted by the set of all ordered pairs formed from two expressions α and β such that the result of putting an equals sign between α and β is a theorem of the theory we have in mind. + similarly is the set of all ordered triples of expressions $\langle\alpha, \beta, \gamma\rangle$ such that the result of putting a '+' sign and an '=' sign between them in the obvious way gives an expression that is a theorem of, again, whatever the theory is that we have in mind. Interpretations for the others are defined similarly.

4.0.16 Exercise 26 (p 21)

"For each of the following pairs of terms, give a most general unifier or explain why none exists."

(i) $f(g(x), z)$ and $f(y, h(y))$

$f(g(x), h(g(x)))$ is the most general unifier.

(ii) $j(x, y, z)$ and $j(f(y, y), f(z, z), f(a, a))$

$j(f(f(f(a, a), f(a, a)), f((a, a), f(a, a))), f(f(a, a), f(a, a)), f(a, a))$ is the most general unification.

(iii) $j(x, z, x)$ and $j(y, f(y), z)$

Any unification requires us to bind 'x" 'y' and 'z' all to the same thing and that 'z' be bound to '$f(y)$'. This violates the occurs check.

(iv) $j(f(x), y, a)$ and $j(y, z, z)$

Any unification requires that 'y' and 'z' both be bound to 'a' and that 'y' be bound to '$f(x)$'. This violates the occurs check.

(v) $j(g(x), a, y)$ and $j(z, x, f(z, z))$

$j(g(a), a, f(g(a), g(a)))$ is the most general unification.

4.0.17 Exercise 27 (p 24)

. . . will appear here, fashionably late.

Convert the following formula into clauses, showing your working. Then present two resolution proofs different from the shown in example 33 above

$$(\exists x)[P \to Q(x)] \wedge (\exists x)[Q(x) \to P]. \to .(\exists x)(P \longleftrightarrow Q(x))$$

First thing to do is negate:

$$(\exists x)[P \to Q(x)] \wedge (\exists x)[Q(x) \to P] \wedge \neg(\exists x)(P \longleftrightarrow Q(x))$$

$$(\exists x)[P \to Q(x)] \wedge (\exists x)[Q(x) \to P] \wedge (\forall x)(\neg P \longleftrightarrow Q(x))$$

This is a conjunction of three things. The first two give us clauses without too much trouble, so let's get them out of the way:

$$\{\neg P, Q(a)\} \text{ and } \{\neg Q(b), P\}$$

. . . leaving us free to concentrate on extracting clause(s) (i think there will be two) from

$$(\forall x)(\neg P \longleftrightarrow Q(x))$$

$$(\forall x)((\neg P \to Q(x)) \wedge (Q(x) \to \neg P)$$

skolemize

$$((\neg P \to Q(x)) \wedge (Q(x) \to \neg P)$$

giving clauses

$$\{P, Q(x)\} \text{ and } \{\neg Q(x), \neg P\}.$$

So our clauses are

1 $\{\neg P, Q(a)\}$
2 $\{\neg Q(b), P\}$
3 $\{P, Q(x)\}$ and
4 $\{\neg Q(x), \neg P\}$.

4.0.18 Exercise 28 (p 24)

(i) Are $\{P(x,b), P(a,y)\}$ and $\{P(a,b)\}$ equivalent?

$\{P(x,b), P(a,y)\}$ is $(\forall xy)(P(x,b) \vee P(a,y))$. Instantiating '$x$' to '$a$' and '$y$' to '$b$' we infer $P(a,b)$. The converse is obviously not going to be provable: take a universe just containing a and b and decide that $P(a,b)$ is true but all other atomics are false.

(ii) Are $\{P(y,y), P(y,a)\}$ and $\{P(y,a)\}$ equivalent?

This one looks uncannily like a tarted up version of Russell's paradox. Perhaps i just have a nasty suspicious mind. Let's rewrite P as $\in$ (and as infix) and the question then becomes

> Are $\{y \in y,\ y \in a\}$ and $\{y \in a\}$ equivalent?

which are $(\forall y)(y \notin y \to y \in a)$ and $(\forall y)(y \in a)$ and it's already much clearer what is going on. They are obviously not equivalent, but it might be an idea to cook up a small finite countermodel making the first true and the second false. Try the model that contains just a and b. $a \in a$ but $b \notin a$. We want $b \notin a$ in order to make $(\forall y)(y \in a)$ false. Since $b \notin a$ we'd better have $b \in b$ lest $(\forall y)(y \notin y \to y \in a)$ fail. Is a a member of b? It seems we can jump either way.

4.0.19 Exercise 29 (p 24)

Definite clauses

If we resolve two nonempty definite clauses we get a nonempty definite clause. So, by induction, the only things we can deduce from nonempty definite clauses are other nonempty definite clauses. Since no definite clause is empty, we cannot deduce the empty disjunction, which is to say we cannot deduce the false!

Toby Miller has a rather cute observation. If we have a hatful of definite clauses, consider the valuation that makes true every variable that has a positive occurrence. Every clause has at least one variable with a positive occurrence and so is rendered true! Clever Toby.

Dmitri Kazhdan says that the same goes for any set of clauses every one of which contains *at least one* positive literal. I suppose he must be right. I wonder if that has any significance....

4.0.20 Exercise 30 (p 24)

Convert these formulæ into clauses, showing each step: negating the formula, eliminating → and ↔, moving the quantifiers, Skolemizing, dropping the universal quantifiers and converting the matrix into CNF.

(i)

$$\begin{array}{rl}
 & (\forall x)(\exists y)R(x,y)) \to ((\exists y)(\forall x)R(x,y)) \\
\textit{negate and remove } \to & ((\forall x)(\exists y)R(x,y)) \wedge \neg((\exists y)(\forall x)R(x,y)) \\
\textit{move quantifiers} & ((\forall x)(\exists y)R(x,y)) \wedge (\forall y)(\neg(\forall x)R(x,y)) \\
 & ((\forall x)(\exists y)R(x,y)) \wedge ((\forall y)(\exists x)\neg R(x,y)) \\
\textit{skolemise and clause} & \{R(x,f(x))\},\ \{\neg R(g(x),x)\}
\end{array}$$

(ii)

$$\begin{array}{rl} & ((\exists y)(\forall x)R(x,y)) \to ((\forall x)(\exists y)R(x,y)) \\ \textit{negate and remove } \to & ((\exists y)(\forall x)R(x,y)) \wedge \neg((\forall x)(\exists y)R(x,y)) \\ & ((\exists y)(\forall x)R(x,y)) \wedge ((\exists x)(\forall y)\neg R(x,y)) \\ \textit{skolemise and clause} & \{R(x,a)\},\ \{\neg R(b,x)\} \end{array}$$

(iii)

$$\begin{array}{rl} & (\exists x)(\forall yz)((P(y) \to Q(z)) \to (P(x) \to Q(x))) \\ \textit{negate and remove } \to & \neg(\exists x)(\forall yz)((P(y) \wedge \neg Q(z)) \vee \neg P(x) \vee Q(x)) \\ & (\forall x)(\exists yz)\neg((P(y) \wedge \neg Q(z)) \vee \neg P(x) \vee Q(x)) \\ & (\forall x)(\exists yz)((\neg P(y) \vee Q(z)) \wedge P(x) \wedge \neg Q(x)) \\ \textit{skolemise and clause} & \{\neg P(f(x)), Q(g(x))\},\ \{P(x)\},\ \{\neg Q(x)\} \end{array}$$

(iv) This is the hardest!

Negate

$$(\exists y)(\forall x)[R(x,y) \longleftrightarrow \neg(\exists z)(R(x,z) \wedge R(z,x))]$$

$$(\exists y)(\forall x)[(R(x,y) \to \neg(\exists z)(R(x,z) \wedge R(z,x))) \wedge (\neg(\exists z)(R(x,z) \wedge R(z,x)) \to R(x,y))]$$

import '¬' past '∃' and a few other minor changes

$$(\exists y)(\forall x)[(R(x,y) \to (\forall z)(\neg R(x,z) \vee \neg R(z,x))) \wedge ((\exists z)(R(x,z) \wedge R(z,x)) \vee R(x,y))]$$

Import $\forall x$:

$$(\exists y)[(\forall x)(R(x,y) \to (\forall z)(\neg R(x,z) \vee \neg R(z,x))) \wedge (\forall x)((\exists z)(R(x,z) \wedge R(z,x)) \vee R(x,y))]$$

I think we can at this stage safely replace '$\exists y$' with a constant:

$$[(\forall x)(R(x,a) \to (\forall z)(\neg R(x,z) \vee \neg R(z,x))) \wedge (\forall x)((\exists z)(R(x,z) \wedge R(z,x)) \vee R(x,a))]$$

Let's now deal with these two conjuncts separately. The left-hand conjunct is

$$(\forall x)(R(x,a) \to (\forall z)(\neg R(x,z) \vee \neg R(z,x)))$$

and we can pull the '$\forall z$' to the front to get

$$(\forall x)(\forall z)(R(x,a) \to (\neg R(x,z) \vee \neg R(z,x)))$$

which is

$$(\forall x)(\forall z)(\neg R(x,a) \vee \neg R(x,z) \vee \neg R(z,x)))$$

which gives us the clause $\{\neg R(x,a), \neg R(x,z), \neg R(z,x)\}$. The right-hand conjunct is

$$(\forall x)((\exists z)(R(x,z) \wedge R(z,x)) \vee R(x,a))$$

and again we can pull the z quantifier out to get

$$(\forall x)(\exists z)((R(x,z) \wedge R(z,x)) \vee R(x,a))$$

Now we can distribute the $\vee$ over the $\wedge$ getting

$$(\forall x)(\exists z)((R(x,z) \vee R(x,a)) \wedge (R(z,x) \vee R(x,a)))$$

giving us the two clauses $\{R(x,f(x)), R(x,a)\}$ and $\{R(f(x),x), R(x,a)\}$

4.0.21 Exercise 33 (p 24)

Find a refutation from the following set of clauses using resolution with factoring.

$$\{\neg P(x,a), \neg P(x,y), \neg P(y,x)\} \quad (1)$$
$$\{P(x,f(x)), P(x,a)\}, \quad (2)$$
$$\{P(f(x),x), P(x,a)\} \quad (3)$$

Binding both 'y' and 'x' to 'a' in clause 1 we get

$$\{\neg P(a,a)\} \quad (4)$$

(with factoring). We can resolve (4) with both clauses 2 and 3 (separately) to get

$$\{P(a,f(a))\} \quad (5)$$

and

$$\{P(f(a),a)\}. \quad (6)$$

(5) resolves with 1 (bind 'y' to 'a' and 'x' to '$f(a)$') to give

$$\{\neg P(f(a),a), \neg P(f(a),a)\} \quad (7)$$

which reduces by factoring to

$$\{\neg P(f(a),a)\}$$

and this resolves with (6) to the empty clause.

4.0.22 Exercise 34 (p 24)

Prove the following formulae by resolution, showing all steps of the conversion into clauses. Remember to negate first!

(i) $(\forall x)(P \vee Q(x)) \to (P \vee (\forall x)Q(x))$

$$\begin{array}{rl}
 & (\forall x)(P \vee Q(x)) \to (P \vee (\forall x)Q(x)) \\
negate\ and\ remove\ \to & \neg(\neg((\forall x)(P \vee Q(x))) \vee (P \vee (\forall x)Q(x))) \\
move\ quantifiers & ((\forall x)(P \vee Q(x))) \wedge \neg((\forall x)(P \vee Q(x))) \\
 & ((\forall x)(P \vee Q(x))) \wedge ((\exists x)\neg(P \vee Q(x))) \\
 & ((\forall x)(P \vee Q(x))) \wedge ((\exists x)(\neg P \wedge \neg Q(x))) \\
skolemise\ and\ clause & \{P, Q(x)\},\ \{\neg P\},\ \{\neg Q(a)\}
\end{array}$$

Resolving the three clauses together gives a contradiction. Therefore the negation of the formula is inconsistent. Therefore the formula is proven.

(ii) $(\exists xy)(P(x,y) \to (\forall vw)P(v,w))$

$$\begin{array}{rl}
 & (\exists xy)(P(x,y) \to (\forall vw)P(v,w)) \\
negate\ and\ remove\ \to & \neg((\exists xy)(\neg P(x,y) \vee (\forall vw)P(v,w)) \\
move\ quantifiers & (\forall xy)(\exists vw)(P(x,y) \wedge \neg P(v,w)) \\
skolemise\ and\ clause & \{P(x,y)\},\ \{\neg P(f(x,y), g(x,y))\}
\end{array}$$

These two clauses resolve to the empty clause when x takes on the value of $f(x,y)$ and y the value of $g(x,y)$. Thus the formula is proven.

4.1 Fourier-Motzkin

We are dealing with equations and inequations over $\Re$ or **Q**. The key common feature is that both these orderings are *dense*.

Key thought: **quantifier-elimination**[2].

Beco's $<_\Re$ and $<_Q$ are dense, '$(\exists x)(y < x < w)$' is equivalent to '$y < w$'. This means that if we can somehow manipulate a formula into a form where the only way a variable 'x' occurs is in a subformula '$(\exists x)(A < x < B)$' where A and B are [possibly complex] terms **not** containing 'x' then we can rewrite the formula as '$A < B$', and we have got rid of a variable and a quantifier. [The discussion is slightly different if we write things with '$\leq$' instead of '$<$', and a little attention is required if you are not to trip yourself up. However the idea is the same.]

So the challenge is to take a complicated package of equations and inequations, with numerical constants and numerical variables, all wrapped up inside some existential quantifiers with perhaps some function symbols, and get rid of as many of those quantifiers as we can.

We start by putting the stuff inside the quantifiers into disjunctive normal form. Reflect that '$\exists$' commutes with '$\vee$' so we can "import" all our existential quantifiers and thereby represent/rewrite our original formula as a disjunction of things like '$(\exists x_1 \ldots x_n)A$' where A is a conjunction of equations and inequations. Inequations? Well no need for them because things like $a \neq b$ and $\neg(a \leq b)$ can be written as boolean combinations (without $\neg$) of things like $a < b$: $a \neq b$ is $a < b \ \vee \ b < a$, and $\neg(a \leq b)$ is of course just $a > b$.

[2] Try Wikipædia.

4.1.1 Exercise 35 (p 27)

We are given the inequalities

$$3x \geq y;\ x \geq 0;\ y \geq z;\ z \leq 1;\ z \geq 0.$$

which is to say that we are asked to examine

$$(\exists z)(\exists x)(\exists y)(3x \geq y \ \wedge\ x \geq 0 \ \wedge\ y \geq z \ \wedge\ z \leq 1 \ \wedge\ z \geq 0)$$

which we can rearrange to

$$(\exists z)(\exists x)([(\exists y)(3x \geq y \ \wedge\ y \geq z)] \ \wedge\ z \leq 1 \ \wedge\ z \geq 0))$$

Observe that the stuff in square brackets—'$(\exists y)(3x \geq y \geq z)$'—is the same as '$3x \geq z$', so we can rewrite it to discard 'y' obtaining

$$(\exists z)(\exists x)(3x \geq z \ \wedge\ z \leq 1 \ \wedge\ z \geq 0)$$

which we can rearrange to

$$(\exists x)[(\exists z)(max(1, 3x) \ \geq\ z \geq 0]$$

so the z disappears, and all we are saying is that $max(1, 3x)$ is +ve... which appears to me to be saying nothing at all.

4.1.2 Exercise 36 (p 27)

We are given

$$x \geq^{(1)} z;\ \ y \geq^{(2)} 2z;\ \ z \geq^{(3)} 0;\ x + y \leq^{(4)} z$$

where we superscript the clauses for easy reference.

That is to say we are contemplating the formula

$$(\exists x)(\exists y)(\exists z)(x \geq^{(1)} z \ \wedge\ y \geq^{(2)} 2z \ \wedge\ z \geq^{(3)} 0 \ \wedge\ x + y \leq^{(4)} z)$$

(2) becomes $z \leq y/2$, so we can rearrange to get

$$(\exists x)(\exists y)(\exists z)(0 \leq z \ \wedge\ x + y \leq z \ \wedge z \leq y/2 \ \wedge z \leq x)$$

which is simply to say

$$(\exists x)(\exists y)(\exists z)(\max(0,\ x + y) \leq z \leq \min(y/2,\ x))$$

and $(\exists z)(A \leq z \leq B)$ is equivalent to $A \leq B$ so we get

$$(\exists x)(\exists y)(\max(0,\ x + y) \leq \min(y/2,\ x))$$

Think about $\max(0,\ x + y)$. There are three cases to consider: $0 = x + y$, $0 < x + y$ and $0 > x + y$, so we take them in order:

(i) $0 = x + y$

x and y might both be zero. But what other possibilities are there? $x = -y$ looks like a starter. But it implies that $y/2$—and therefore y and x—are nonnegative; one is minus the other so they'd both have to be 0 as before.

So $x = y = 0$ is a possibility.

(ii) $0 < x + y$.

This implies that both x and $y/2$—and therefore y—are +ve. But if x and y are both +ve then $x + y \leq y/2$ isn't possible, so $x + y$ must be -ve. This takes us to case (iii).

(iii) $x + y < 0$.

But $0 < y/2$ implies that y is +ve, and $0 < x$ means that x is +ve, and these can't be true if $x + y < 0$.

So we conclude that $x = y = 0$

4.1.3 Exercise 37 (p 27)

$$(\exists x)(\exists y)(\exists z)(x \leq 2y \wedge\ x \leq y + 3\ \wedge\ z \leq x\ \wedge\ 0 \leq z\ \wedge\ y \leq 4x)$$

What is 'z' doing? Nothing: it's sitting between 0 and x. We can simplify to get

$$(\exists x)(\exists y)(x \leq 2y\ \wedge\ x \leq y + 3\ \wedge\ 0 \leq x\ \wedge\ y \leq 4x)$$

Now $y \leq 4x$ is the same as $y/4 \leq x$ so we have

$$(\exists x)(\exists y)(x \leq 2y\ \wedge\ x \leq y + 3\ \wedge\ 0 \leq x\ \wedge\ y/4 \leq x)$$

which we can rearrange to

$$(\exists y)(\exists x)(0 \leq x\ \wedge\ y/4\ \leq\ x\ \wedge\ \ x \leq y + 3\ \wedge\ x \leq 2y)$$

and the stuff within the scope of the '$(\exists y)$' is of the form '$(\exists x)(A \leq x \leq B)$' with '$x$' not free in A or B, and we know what to do with that—we can discard x to get

$$(\exists y)(max(0, y/4) \leq min(2y, y + 3)),$$

which i think just says that $y \geq 0$.

4.1.4 Exercise 42 (p 30)

Let `blob` and `blah` be two operator strings, and suppose we are given the equivalence

$$\texttt{blob}\ \phi \longleftrightarrow \texttt{blah}\ \phi$$

This biconditional holds for all ϕ so we can substitute $\neg\phi$ for ϕ:

$$\texttt{blob}\neg\phi \longleftrightarrow \texttt{blah}\neg\phi$$

But $A \longleftrightarrow B$ is equivalent to $(\neg A) \longleftrightarrow (\neg B)$, so the last formula is equivalent to

$$\neg\texttt{blob}\neg\phi \longleftrightarrow\ \neg\texttt{blah}\neg\phi$$

But this is precisely the dual of the equivalence we started with, and it holds for all ϕ.

4.1.5 Exercise 43 (p 30)

An answer from Dmitry Kazhdan

Prove the sequents: $\Diamond(A \vee B) \vdash \Diamond A, \Diamond B$ and $\Diamond A \vee \Diamond B \vdash \Diamond(A \vee B)$

i)

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{B \vdash A, B}\,(basic) \qquad \dfrac{}{A \vdash A, B}\,(basic)}{(A \vee B) \vdash A, B}\,(\vee l)}{(A \vee B) \vdash A, \Diamond B}\,(\Diamond r)}{(A \vee B) \vdash \Diamond A, \Diamond B}\,(\Diamond r)}{\Diamond(A \vee B) \vdash \Diamond A, \Diamond B}\,(\Diamond l)$$

ii)

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A \vdash A, B}\,(basic)}{A \vdash (A \vee B)}\,(\vee r)}{A \vdash \Diamond(A \vee B)}\,(\Diamond r)}{\Diamond A \vdash \Diamond(A \vee B)}\,(\Diamond l) \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{B \vdash A, B}\,(basic)}{B \vdash (A \vee B)}\,(\vee r)}{B \vdash \Diamond(A \vee B)}\,(\Diamond r)}{\Diamond B \vdash \Diamond(A \vee B)}\,(\Diamond l)}{\Diamond A \vee \Diamond B \vdash \Diamond(A \vee B)}\,(\vee l)$$

Thus: $\Diamond A \vee \Diamond B \vdash \Diamond(A \vee B)$ and $\Diamond(A \vee B) \vdash \Diamond A, \Diamond B$.
This allows us to conclude that: $\Diamond A \vee \Diamond B \simeq \Diamond(A \vee B)$

4.1.6 Exercise 44 (p 30)

An answer from Dmitry Kazhdan

Prove: $\Diamond(A \to B), \Box A \vdash \Diamond B$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A \vdash B, A}\,(basic) \qquad \dfrac{}{B, A \vdash B}\,(basic)}{(A \to B), A \vdash B}\,(\to l)}{(A \to B), \Box A \vdash B}\,(\Box l)}{(A \to B), \Box A \vdash \Diamond B}\,(\Diamond r)}{\Diamond(A \to B), \Box A \vdash \Diamond B}\,(\Diamond l)$$

4.1.7 Exercise 45 (p 30)

An answer from Dmitry Kazhdan

Prove: $\Box(A \wedge B) \simeq \Box A \wedge \Box B$

i) Prove: $\Box(A \wedge B) \vdash \Box A \wedge \Box B$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A, B \vdash A}\,(basic)}{(A \wedge B) \vdash A}\,(\wedge l)}{\Box(A \wedge B) \vdash A}\,(\Box l)}{\Box(A \wedge B) \vdash \Box A}\,(\Box r) \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{A, B \vdash B}\,(basic)}{(A \wedge B) \vdash B}\,(\wedge l)}{\Box(A \wedge B) \vdash B}\,(\Box l)}{\Box(A \wedge B) \vdash \Box B}\,(\Box r)}{\Box(A \wedge B) \vdash \Box A \wedge \Box B}\,(\wedge r)$$

ii) Prove: $\Box A \wedge \Box B \vdash \Box(A \wedge B)$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A, B \vdash B}\,(basic) \qquad \dfrac{}{A, B \vdash A}\,(basic)}{A, B \vdash (A \wedge B)}\,(\wedge r)}{\Box A, B \vdash (A \wedge B)}\,(\Box l)}{\Box A, \Box B \vdash (A \wedge B)}\,(\Box l)}{\Box A, \Box B \vdash \Box(A \wedge B)}\,(\Box r)}{\Box A \wedge \Box B \vdash \Box(A \wedge B)}\,(\wedge l)$$

4.1.8 Exercise 46 (p 30)

An answer from Dmitry Kazhdan

Prove: $\Box\Diamond\Box A, \Box\Diamond\Box B \vdash \Box\Diamond\Box(A \wedge B)$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A, B \vdash A}\,(basic) \qquad \dfrac{}{A, B \vdash B}\,(basic)}{A, B \vdash (A \wedge B)}\,(\wedge r)}{\Box A, B \vdash (A \wedge B)}\,(\Box l)}{\Box A, \Box B \vdash (A \wedge B)}\,(\Box l)}{\Box A, \Box B \vdash \Box(A \wedge B)}\,(\Box r)}{\Box A, \Box B \vdash \Diamond\Box(A \wedge B)}\,(\Diamond r)}{\Box A, \Diamond\Box B \vdash \Diamond\Box(A \wedge B)}\,(\Diamond l)}{\Box A, \Box\Diamond\Box B \vdash \Diamond\Box(A \wedge B)}\,(\Box l)}{\Diamond\Box A, \Box\Diamond\Box B \vdash \Diamond\Box(A \wedge B)}\,(\Diamond l)}{\Box\Diamond\Box A, \Box\Diamond\Box B \vdash \Diamond\Box(A \wedge B)}\,(\Box l)}{\Box\Diamond\Box A, \Box\Diamond\Box B \vdash \Box\Diamond\Box(A \wedge B)}\,(\Box r)$$

4.1.9 Exercise 47 (p 32)

An answer from Dmitry Kazhdan

i) Prove: $(\exists y\, \forall x\, R(x,y)) \to (\forall x\, \exists y\, R(x,y))$
Negate to obtain: $(\exists y\, \forall x\, R(x,y)) \wedge \neg(\forall x\, \exists y\, R(x,y))$
Simplify: $(\exists y\, \forall x\, R(x,y)) \wedge (\exists x\, \forall y\, \neg R(x,y))$
Skolemize: $(\forall x\, R(x,a)) \wedge (\forall y\, \neg R(b,y))$
Prove:

$$\dfrac{\dfrac{\dfrac{\dfrac{x \mapsto b,\ y \mapsto a}{R(x,a), \neg R(b,y) \vdash}\,(basic)}{R(x,a), \forall y\, \neg R(b,y) \vdash}\,(\forall l)}{\forall x\, R(x,a), \forall y\, \neg R(b,y) \vdash}\,(\forall l)}{(\forall x\, R(x,a)) \wedge (\forall y\, \neg R(b,y)) \vdash}\,(\wedge l)$$

ii) Prove: $(P(a,b) \vee\ \exists z\, P(z,z)) \to \exists xy\, P(x,y)$
Negate to obtain: $(P(a,b) \vee\ \exists z\, P(z,z)) \wedge \neg(\exists xy\, P(x,y))$
Simplify: $(P(a,b) \vee\ \exists z\, P(z,z)) \wedge (\forall xy\, \neg P(x,y))$
Skolemize: $(P(a,b) \vee\ P(c,c)) \wedge (\forall xy\, \neg P(x,y))$. Prove:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{x \mapsto a,\ y \mapsto b,\ z \mapsto c,\ r \mapsto c}{P(a,b),\ \neg P(x,y), \neg P(z,r) \vdash}\,(basic) \qquad \dfrac{x \mapsto a,\ y \mapsto b,\ z \mapsto c,\ r \mapsto c}{P(c,c),\ \neg P(x,y), \neg P(z,r) \vdash}\,(basic)}{P(a,b) \vee\ P(c,c),\ \neg P(x,y), \neg P(z,r) \vdash}\,(\vee l)}{P(a,b) \vee\ P(c,c),\ \neg P(x,y), \forall y\, \neg P(z,y) \vdash}\,(\forall l)}{P(a,b) \vee\ P(c,c),\ \neg P(x,y), \forall xy\, \neg P(x,y) \vdash}\,(\forall l)}{P(a,b) \vee\ P(c,c), \forall y\, \neg P(x,y), \forall xy\, \neg P(x,y) \vdash}\,(\forall l)$$

$$\dfrac{\dfrac{\cdots}{P(a,b) \vee\ P(c,c), \forall xy\, \neg P(x,y) \vdash}\,(\forall l)}{(P(a,b) \vee\ P(c,c)) \wedge (\forall xy\, \neg P(x,y)) \vdash}\,(\wedge l)$$

iii) Prove: $(\exists x\, P(x) \to Q) \to \forall x\, (P(x) \to Q)$
Negate: $(\exists x\, P(x) \to Q) \wedge \neg(\forall x\, (P(x) \to Q))$
Simplify: $(\neg(\exists x\, P(x)) \vee Q) \wedge \exists x\, \neg(P(x) \to Q)$
Simplify: $(\forall x\, \neg P(x) \vee Q) \wedge \exists x\, (P(x) \wedge \neg Q)$
Skolemize: $(\forall x\, \neg P(x) \vee Q) \wedge P(a) \wedge \neg Q$
Prove:

$$\begin{prooftree}
\AxiomC{$x \mapsto a$}
\RightLabel{$(basic)$}
\UnaryInfC{$\neg P(x),\, P(a),\, \neg Q \vdash$}
\RightLabel{$(\forall l)$}
\UnaryInfC{$\forall x\, \neg P(x),\, P(a),\, \neg Q \vdash$}
\AxiomC{$x \mapsto a$}
\RightLabel{$(basic)$}
\UnaryInfC{$Q,\, P(a),\, \neg Q \vdash$}
\RightLabel{$(\vee l)$}
\BinaryInfC{$(\forall x\, \neg P(x) \vee Q),\, P(a),\, \neg Q \vdash$}
\RightLabel{$(\wedge l)$}
\UnaryInfC{$(\forall x\, \neg P(x) \vee Q),\, P(a) \wedge \neg Q \vdash$}
\RightLabel{$(\wedge l)$}
\UnaryInfC{$(\forall x\, \neg P(x) \vee Q) \wedge P(a) \wedge \neg Q \vdash$}
\end{prooftree}$$

4.2 Some leftovers from old versions of the file of Larry's questions

4.2.1 Exercise ??

Negate and convert $(A_1 \wedge \cdots \wedge A_k) \rightarrow B$ **to CNF**
Negate to give $\neg((A_1 \wedge \cdots \wedge A_k) \rightarrow B)$
Eliminate → to give $\neg(\neg(A_1 \wedge \cdots \wedge A_k) \vee B)$
Push in negations $(A_1 \wedge \cdots \wedge A_k) \wedge \neg B$
Remove parentheses to give $A_1 \wedge \cdots \wedge A_k \wedge \neg B$

Convert $M \rightarrow K \wedge P$ **to clausal form.**
Split into two formulae, $M \rightarrow K$ and $M \rightarrow P$.
Eliminate →s to give $\neg M \vee K$ and $\neg M \vee P$.
Convert to clauses $\{\neg M, K\}$ and $\{\neg M, P\}$.

Another Exercise

This question seems to have vanished from later editions. It's obviously Russell's paradox. Prove: $\neg(\exists x)(\forall y)(R(y,x) \leftrightarrow \neg R(y,y))$

$$\begin{array}{rl} & \neg(\exists x)(\forall y)(R(y,x) \leftrightarrow \neg R(y,y)) \\ \textit{negate and remove} \leftrightarrow & (\exists x)(\forall y)((\neg R(y,x) \vee \neg R(y,y)) \wedge (R(y,y) \vee R(y,x))) \\ \textit{skolemise and clause} & \{\neg R(x,a), \neg R(x,x)\}, \ \ \{R(x,x), R(x,a)\} \end{array}$$

If we resolve these two clauses together we get a contradiction. Thus formula is proven.

Former Exercise 26: Dual Skolemisation

Let $\mathcal{L}$ be a language, and let $\Psi : \mathcal{L} \rightarrow \mathcal{L}$ be a map such that, for all formulæ ϕ, $\Psi(\phi)$ is satisfiable iff ϕ is. (Skolemisation is an example). We will now show that the map $\lambda\phi.\neg(\Psi(\neg\phi))$ preserves validity.

ϕ is valid iff
$\neg\phi$ is not satisfiable iff
$\Psi(\neg\phi)$ is not satisfiable iff
$\neg(\Psi(\neg\phi))$ is valid.

Now all we have to check is that if Ψ is skolemisation then $\lambda\phi.\neg(\Psi(\neg\phi))$ is dual skolemisation *à la Larry Paulson*. Take a formula in prenex normal form, as it might be

$$\forall x \exists y \forall z \phi$$

where ϕ is anything without quantifiers. Negate it to get

$$\exists x \forall y \exists z \neg\phi$$

and skolemize to get

$$\neg\phi[(f(y))/z, a/x]$$

and negate again to get

$$\phi[(f(y))/z, a/x]$$

. . . which is exactly what you would have got if you dual skolemised the formula we started with.

I'm not entirely sure what use this is. I suppose one might use it in the following circumstances. You want to prove ϕ from Γ. You (i) **dual**-skolemise ϕ and convert to **disjunctive** normal form; (ii) negate Γ and convert to **disjunctive** normal form. Then you write clauses which are of course conjunctions not disjunctions and resolving to the empty clause means you have established the truth of $\neg\Gamma \vee \phi$. Two things to think about: (i) is there a problem about relettering clauses? (ii) Did we really need to dual skolemise?

Skolemisation

The way to understand what skolemisation is doing is something like this: skolemisation is supposed to preserve satisfiability not truth. Truth is a property of a formula in an environment, but satisfiability is a property of the formula itself. Skolemisation is something you do to a formula not to a formula-in-an-environment. (Any logical manipulation on a formula of course—such as conversion to CNF—is also (strictly) something you do to a (naked) formula rather than to a formula-in-an-environment, but with conversion to CNF you can carry the environment along with you if you like). Ideally we should introduce a bit of model theory at this point, and it's actually quite easy to do—at least if you are a compsci who is used to abstract datatypes and object-oriented programming . . . so here goes! The formula ϕ you are trying to skolemise has various function symbols and relation symbols in it, and interpretations for it are objects of a certain type—a type of objects equipped with operations of the right arity to correspond to the function letters and relation symbols of your formula ϕ. When you skolemise ϕ you are obtaining from ϕ a formula in a language with more gadgets, appropriate to objects of a slightly more complicated type—a type whose objects have an extra gadget. The reason why skolemisation is so smooth and well-behaved is that it is very easy to obtain an object of the enhanced type from the object of the orginal ϕ-type by decorating it with a function. (This is what the axiom of choice tells us, if you have ever heard of the axiom of choice and wondered what it was). So if ϕ is satisfiable it is because there is some object of the ϕ-type that it describes. Simply decorate that object with the skolem function and you have an object of the enhanced type that corresponds to the skolemisation of ϕ.

Notice that Skolemisation preserves satisfiability but not validity. $(\forall x)(\exists y)(x = y)$ is valid but its skolemisation (which is $(\forall x)(x = f(x))$) is not valid! Indeed

Remark 3 *Except in trivial cases, the skolemised version of a formula is never valid*

Proof:

Suppose ϕ' is the skolemisation of ϕ. Then each *new* function letter (arising from Skolemisation) in ϕ' appears only in connection with one variable. If the function letter 'f' arose by skolemising the variable 'y' which is in the scope of 'x' then, in ϕ', 'f' only ever appears succeded by 'x'. If we have a proof of ϕ' we can replace all occurrences of '$f(x)$' in it by a new constant, and then use UG on that constant. So if ϕ' were valid, then the formula obtained from ϕ by replacing all existential quantifiers by universal quantifers would be valid too. ■

The fact that Skolemisation preserves satisfiability is well known to the *cognoscenti*; and the rest of us can consult—for example—the entry by Avigad and Zach on the epsilon calculus in the Stanford online Encyclopædia of Philosophy to be found at https://plato.stanford.edu/entries/epsilon-calculus/index.html. As a gesture in the direction of making this note self-contained, a sketch follows.

We will illustrate with a simple two-quantifier case. Our proof system will be sequent calculus; we will outline a proof of the contrapositive: if $\forall x\phi(x, f(x))$ is not satisfiable, then neither is $\forall x\exists y\phi(x, y)$.

Suppose we have a proof of

$$\vdash \exists x\neg\phi(x, f(x)) \tag{2}$$

There might be more than one application of $\exists$-R with some contraction on the right but we will have got this from something like

$$\vdash \neg\phi(x_1, f(x_1)), \quad \neg\phi(x_2, f(x_2)), \quad \neg\phi(x_n, f(x_n)) \ldots \tag{4.6}$$

where the various x_i are not necessarily variables but might be complex terms. Now any sequent proof of (2) can be transformed into a proof of

$$\vdash \neg\phi(x_1, z_1), \quad \neg\phi(x_2, z_2), \quad \neg\phi(x_n, z_n), \ldots \tag{4.7}$$

simply by replacing '$f(x_1)$', '$f(x_2)$', '$f(x_n)$' etc throughout by fresh variables z_i. (Since we know nothing about f it must destroy all information about its argument.) We can then do some $\forall$-R on the z_i to obtain

$$\vdash \forall y \neg\phi(x_1, y), \quad \forall y \neg\phi(x_2, y), \quad \forall y \neg\phi(x_n, y), \ldots \tag{4.8}$$

and further $\exists$-R and contraction-on-the-right to get

$$\vdash \exists x \forall y \neg\phi(x, y) \tag{4.9}$$

So if the skolemised version of the formula was refutable then the original formula was refutable, which is what we wanted.

Postscript

Both skolemisation and dual skolemisation are maps from one language to another language that preserve something. In one case *satisfiability* and in the other case *validity*. In this context we can anticipate a map used in the complexity theory course. Take a formula in conjunctive normal form (so it's a conjunction of disjunctions). In general the individual conjuncts may have lots of literals in them, and be something like $(p \vee q \vee \neg r \vee s)$. This conjunct has four literals in it. Now consider the result of replacing this conjunction by the (conjunction of the) two conjuncts $(p \vee q \vee t) \wedge (\neg t \vee \neg r \vee s)$, where '$t$' is a new variable not present in the original formula. The new formula is satisfiable iff the original one was. (Any valuation v satisfying $(p \vee q \vee t) \wedge (\neg t \vee \neg r \vee s)$ restricts to a valuation satisfying $(p \vee q \vee \neg r \vee s)$—we just discard the information about what v does to 't'.)

Also (although the new formula has more conjuncts) we have replaced longer conjuncts by shorter conjuncts. You will see later why this is a useful trick.

Former Exercise 26

Verify that $\circ$ is associative and has [] for an identity.

To show associativity we need to show that $(\phi \circ \theta) \circ \sigma = \phi \circ (\phi \circ \theta)$.

If we consider ϕ, θ and σ as functions $f(x)$, $g(x)$ and $h(x)$ which map literals to their substituted values then we get the composition

$$\begin{aligned} \lambda x.((f \circ g) \circ h) &= \lambda x.(f \circ (g \circ h)) \\ \lambda x.((\lambda y.f(g(x)))h) &= \lambda x.(f(g \circ h)) \\ \lambda x.(f(g(h(x)))) &= \lambda x.f(g(h(x))) \end{aligned}$$

Which says that they are the same. This relies on our functions returning the literal given as an argument in cases where no substitution has been defined.

To show that [] is the identity we need to consider it as a function g which maps all the argument literals to themselves, without substitution. ϕ remains the function f as before.

$$\begin{aligned} f \circ g &= f \\ g(f) &= f \\ f &= f \end{aligned}$$

4.2.2 Exercise ??

Find a refutation from the following set of clauses using linear resolution

$$\{P(f(x,y)), \neg Q(x), \neg R(y)\}, \quad \{\neg P(v)\}, \quad \{\neg R(z), Q(g(z))\}, \quad \{R(a)\}.$$

Unify 'x' ↦ '$g(z)$' and cut $\{P(f(x,y)), \neg Q(x), \neg R(y)\}$ against $\{\neg R(z), Q(g(z))\}$ with cut formula $Q(g(z))$ to give $\{P(f(g(z),z)), \neg R(z)\}$. Unify '$z$' ↦ '$a$' and cut $\{R(a)\}$ against $\{P(f(g(z),z)), \neg R(z)\}$ to give $\{P(f(g(a),a))\}$. Unify 'v' ↦ '$f(g(a),a)$' and cut the result against $\{\neg P(v)\}$ to give a refutation ($\{\}$).

Chapter 5

Unification

5.1 Some ML code for unification

This code comes from a dialect of ML known as HOL. All terms are regarded as curried: operator applied to operand. Thus HOL would regard $f(x, y)$ as $f(x)$ applied to y. `rev_itlist` iteratively applies a list of functions to an arguments to obtain a values. Thus `apply_subst` successively applies a list of substitutions to a term. A substitution is a pair of terms. `@` concatenates two lists.

```
let apply_subst l t = rev_itlist (\pair term.subst[pair]term) l t;;

% Find a substitution to unify two terms (lambda-terms not dealt with) %

letrec find_unifying_subst t1 t2 =
 if t1=t2
  then []
 if is_var t1
  then if not(mem t1 (frees t2)) then [t2,t1] else fail
 if is_var t2
  then if not(mem t2 (frees t1)) then [t1,t2] else fail
 if is_comb t1 & is_comb t2
  then
   (let rat1,rnd1 = dest_comb t1
    and rat2,rnd2 = dest_comb t2
    in
    let s = find_unifying_subst rat1 rat2
    in s@find_unifying_subst(apply_subst s rnd1)(apply_subst s rnd2)
   )else fail;;
```

This currying corresponds to a determination—when unifying (for example)—‘$f(a, b, f(x))$’ with ‘$f(x, y, w)$’—to detect $x \mapsto a$ and then do that to the third argument of the first occurrence of ‘f’ so that it becomes ‘$f(a)$’ before we get there. This finesses questions about simultaneous *versus* consecutive execution of substitution.

5.1.1 Unification: an illustration

In the two axioms.

1. $(\forall xy)(x > y \to Sx > Sy)$
2. $(\forall w)(Sw > 0)$

'S' is the successor function: $S(x) = x + 1$. (Remember that $\mathbb{N}$ is the recursive datatype built up from 0 by means of the successor function.)

Now suppose we want to use PROLOG-style proof with resolution and unification to find a z such that $z > S0$. We turn 1 and 2 into clauses getting $\{\neg(x > y), Sx > Sy\}$ and $\{Sw > 0\}$, and the (negated) goal clause $\{\neg(z > S0)\}$.

The idea now is to refute this negated goal clause. Of course we can't refute it, beco's there are indeed some z of which this clause holds, but we might be able to refute some instances of it, and this is where unification comes in.

$z > S0$ will unify with $Sx > Sy$ generating the bindings $z \mapsto Sx$ and $y \mapsto 0$. We apply these bindings to the two clauses clauses $\{\neg(x > y), Sx > Sy\}$ and $\{\neg(z > S0)\}$, obtaining $\{\neg(x > S0), Sx > S0\}$ and $\{\neg(Sx > S0)\}$. These two resolve to give $\{\neg(x > 0)\}$. Clearly the substitution $x \mapsto Sw$ will enable us to resolve $\{\neg(x > 0)\}$ (which has become $\{\neg(Sw > 0)\}$) with $\{Sw > 0\}$ to resolve to give the empty clause. *En route* we have generated the bindings $z \mapsto Sx$ and $x \mapsto Sw$, which compose to give $z \mapsto SSw$, which tells us that the successor of the successor of any number is bigger than the successor of 0 as desired. Notice that the answer given by this binding ($z \mapsto SSw$) is the most general possible response to "find me something $> S0$". This is because the unification algorithm finds the most general answer.

The idea is this: We are trying to find a witness to $(\exists x)(A(x))$. Assume the negation of this, and try to refute it. In the course of refuting it we generate bindings that tell us what the witnesses are.

5.1.2 Higher-order Unification

Unification in first-order logic is well-behaved. For any two complex terms t_1 and t_2 if there is any unifier at all there is a most general unifier which is unique up to relettering. This doesn't hold for higher-order logic where there are function variables. It's pretty clear what you have to do if you want to unify $f(3)$ and 6: you replace f by something like

`if` $x = 3$ `then` 6 `else don't-care`

(which one might perhaps write $(\epsilon f)(f(3) = 6)$).

However what happens if you are trying to unify $f(3)$ and $g(6)$? You want to bind 'f' to

`if` $x = 3$ `then` $g(6)$ `else don't-care` (A)

but then you also want to bind 'g' to

`if` $x = 6$ `then` $f(3)$ `else don't-care` (B)

and you have a vicious loop of substitutions. There are restricted versions that work, and there was even a product called `Q-PROLOG` ('Q' for Queensland) that did something clever. I've long ago forgotten.

I find in my notes various ways of coping with this, one using ϵ terms. One can have an epsilon term which is is a pair of things satisfying (A) and (B):

$$(\epsilon p)(\exists h_1, h_2)(p = \langle h_1, h_2\rangle \wedge h_1(3) = h_2(6))$$

so that we bind 'f' to '$\mathtt{fst}(p)$' and 'g' to '$\mathtt{snd}(p)$'.