Forcing:
What I wish I'd been told when I was little; written up to amuse Dillon Mayhew

Thomas Forster

July 27, 2025

Years ago i decided i wanted to understand forcing properly, and i battened on Charles Morgan. (I supervised him for Part II Logic when he was a u/g at Magdalen, so i tho'rt i had the right to pester him). I told him i didn't understand it, and i needed help. He replied 'but you *do* understand it!' No! I didn't understand it, and i still don't, tho' i am getting better.

One thing i have only comparatively recently come to appreciate is that forcing is a game of two halves. The way it is narrated one would think it is a set-theoretic phenomenon. It isn't. The second half is a set-theoretic story, and it was set-theorists who first spelt it out, but it's not fundamentally a set-theoretic idea at all. What *is* true is that it was originally developed to establish the independence of the continuum hypothesis from ZFC, and in consequence a lot of the actual business that students are told about – the apparatus of names, the truth lemma – is directed towards its application to set theory.

So what is the fundamental idea? The project is to add a new object to the world in which you are operating. Perhaps the Last Common Ancestor of the ideas we are going to explore below is the concept of a convergent sequence. A convergent sequence is a sequence of things which (collaborate in the endeavour to) *approximate* the thing to which they converge. (If you want to think hard about the idea of a convergent sequence you get led – as was Moore[1] – into filters and suchlike. I think that's how filters were discovered. Later versions of this document might contain some detail about this.) There are more complex situations we encounter later which nevertheless have the same sort of flavour. Sometimes we find ourselves approximating something that isn't actually there by means of lot of things that *are* there. For example: extending a field F by adding a root for an equation with coefficients in F but no roots in F. (This example is mentioned by Cohen in his original book about the independence of CH.) An early example is the adding of ideal divisors to rings that do not have unique factorisation. In fact – casting the net a little wider – one thinks of the way in which one reconstructs (other verbs can be used too) the integers from the naturals, the rationals from the integers, the reals from the rationals and

[1] Annals of Mathematics Second Series, Vol. 38, No. 1 (Jan., 1937), pp. 39-56 (18 pages)

the complexes from the reals. It is customary to tell this story as an extended process of invention of new entities to supply – at each stage – solutions to equations which can be expressed in terms of old entities but to which the old entities do not supply solutions. One needs these new entities, but one needs them to be supplied by a sober and intelligible process.

A procedural question in all these situations is: *how do we concretise these new elements?*

There is a huge amount that can be said about this programme, but i want to restrict myself to making it plausible that forcing is just another exercise of this kind.

One needs to distinguish two things.

(i) One can introduce a new object as the set of things to which it is the answer-from-heaven. For example: ideal divisors in rings. You identify an ideal divisor with the set of things that are multiples of it. The ideal divisor is not in any sense *approximated* by its multiples. Or

(ii) one can think of the new object as the set of things that approximate it, for example a real could be identified with a rational sequence that converges to it. Since any real is the limit of lots of different sequences one identifies a real (or at least Cauchy does) with the *set of* rational sequences that converge to it. This makes it look a bit like the move that gave us ideal divisors but it's different from it beco's in (ii) we have a notion of approximation.

It is this second move that has come to be called *forcing*.

Forcing is about *approximation*. The approximants form a poset: they are partially ordered by information content. This poset is often written $\mathbb{P} = \langle P, \leq \rangle$. ("information" is a piece of slang – please do not ask for a definition!) and approximants with more information come *lower* in the poset than those with more. This might initially look perverse, but there is a good reason for it. Think of a boolean algebra of truth values. `true` (or $\mathbb{1}$ or $\top$) is the top element. It contains no information; `false` or $\bot$ is the bottom element, and contains *too much* information! The more information a proposition contains, the closer it is to being false, so the lower down.

The approximants are always collectively known as *conditions*, (which is less evocative) but we should nevertheless continue to *think* of them as approximants – even if we are to continue calling them conditions – since this terminology motivates the use to which we will put them.

A word is in order at this stage about how many conditions there will be in $\mathbb{P}$. In all interesting cases $\mathbb{P}$ will be infinite. Most (if not all, i haven't actually checked!) of the apparatus to be developed below will work even if $\mathbb{P}$ is finite, but you don't get anything interesting. So you may assume, Dear Reader, that $\mathbb{P}$ is always infinite.

1 It's time for some Definitions

There are two adjectives pertaining to posets which will be very important in what follows. They are *directed* and *separative*.

Definition 1
A poset $\langle P, \leq \rangle$ *is* **directed** *if* $(\forall x, y)(\exists z)(x \leq z \wedge y \leq z)$.
A poset is **directed-complete** *if every directed subset has a least upper bound.*

> There is a connection here with the idea of *amalgamation*, possibly familiar from Model Theory. It's a weaker condition than directedness: two elements have a common upper bound whenever there is something below both of them.

A separative poset is one that is trying its hardest not be directed. In a directed poset any two points have an upper bound. Clearly we cannot say that no two (distinct) points have an upper bound, beco's if $x \leq y$ then anything $\geq y$ is an upper bound for both. (We could say that a poset is separative if whenever $x \not\leq y \not\leq x$ then they have no common upper bound, but that is too restrictive – only trees pass that test.) What we can say is that in all other cases – where $x \not\leq y$ – there is $y' \geq y$ s.t. x and y' have no upper bound. Thus we say:

Definition 2 *A poset* $\langle X, \leq \rangle$ *is* **separative** *iff*

$$(\forall x, y \in X)(x \not\leq y \rightarrow (\exists z \geq y)(\forall w)(w \not\geq z \vee w \not\geq x))$$

We will abbreviate $(\forall w)(w \not\geq z \vee w \not\geq x)$ *("z and x have no upper bound") as* $z \perp x$*, and we will say that z and x are* **incompatible***; otherwise they are* **compatible**.

So, using $\perp$ we can rewrite the definition of separative as

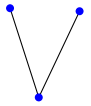
$$(\forall x, y \in X)(x \not\leq y \rightarrow (\exists y' \geq y)(y' \perp x))$$

> The way the definition of *separative* arises from an attempt to negate *directed* reminds me – for one – of the way the definition of *antisymmetric* arises from an attempt to negate *symmetric*. If you forget the definition of separative it might help to remember that it is an attempt to capture "really *really* not directed!". Notice, too, that the definition of *antisymmetric* is symmetrical in the two variables, just as the definition of *symmetric* is. However, here, the definition of separative is not symmetric in the two universally quantified variables. Is our definition equivalent to $(\forall xy)(x \not\leq y \not\leq x \rightarrow (\exists x' \geq x)(\exists y' \geq y)(x' \perp y'))$? And does it matter...? Is there a sensible notion of *infinitarily separative* where the '$\exists$' is a '$\exists^\infty$'?
>
> $$(\forall x, y \in X)(x \not\leq y \rightarrow (\exists^\infty y' \geq y)(y' \perp x))$$

Actually these two definitions should probably be described in more detail as "upwardly directed" and "upwardly separative". There are analogous notions of *downwards directed* and *downwards separative*, which are what will need here.

That being the case i will henceforth drop the 'downwards'.

Miniexercise: Is the poset in this Hasse diagram separative?



A: Yes!

Both these properties are captured by $\forall^*\exists^*\forall$ formulæ but apparently not by anything simpler. This has the effect that they are not preserved under substructure: a substructure of a separative (resp. directed) poset cannot be relied upon to be separative (resp. directed). The same goes for quotients: Item 7 below supplies an example of a directed poset with a separative quotient.

There is an important *informal* motivating intuition of compatibility lurking behind the formal notion $\perp$ in the concept of directedness seen above. The thought is that in a directed poset any two elements are in some intensional sense compatible. This is particularly pertinent when – as here – the partial order is of information content. Thus if you have a directed poset of things whose calling in life is to approximáte wombats, then this directed poset will approximáte a single wombat. (They define a *direction*, a destination towards which the process of approximation is leading). In circumstances where the existence of our desired wombat is precluded (for whatever reason) one can use the directed set of approximating prewombats as a proxy for – or a concretisation of – the wombat.

Jech [4] section 5.6 p 83 has the following exercises on separativeness.

> *3. For every partially ordered set $\langle P, \leq_P\rangle$ there exists a unique (up to isomorphism) separative partially ordered set $\langle Q, \leq_Q\rangle$ and a homomorphism $h : P \twoheadrightarrow Q$ such that for all $p, q \in P$, p, q are compatible in P iff $h(p), h(q)$ are compatible in Q.*
>
> *[Hint: Let $p \leq_Q q$ if q is compatible with each $x \leq_P p$.]*
>
> *4. Every separative partial ordering $\langle Q, \leq_Q\rangle$ can be embedded isomorphically onto a dense subset of a (unique) complete Boolean algebra.*
>
> *[Hint: Consider subsets A of Q with the following properties:*
>
> > *(a) if $y \leq_Q x \in A$, then $y \in A$;*
> > *(b) if $(\forall y \leq_Q x)(\exists z \leq_Q y)[z \in A]$, then $x \in A$.*
>
> *The family of all such sets, partially ordered by inclusion, is a complete Boolean algebra.]*
>
> [I should really write out an answer to this . . .]

. . . see also [5] pp 204ff.

I mentioned above that a separative poset is one that is as undirected as possible. So what is the rôle of the concept of separative here? You have a horde of pre-wombats whose life mission is to approximáte wombats, and they are partially ordered by information content. This horde[2] of prewombats are probably going to approximáte lots and lots of distinct wombats. If they were a directed poset they would approximáte only the one wombat; but this horde contains all prewombats and will approximáte lots of distinct wombats; the less directed the poset is, the more wombats we can approximáte. So we want the poset of prewombats to be separative.

The poset $\mathbb{P}$ of conditions is not itself directed; nevertheless it may have lots of directed *subsets*. It would be *directed-complete* if every directed subset had an infimum. The idea is that the infimum of a directed subset X of conditions is to be the concretisation of the thing that the conditions inside X are approximating. But not just any old X: the subset $\{y : y \geq x\}$ is a (downward-) directed subset but its infimum is x – which we don't need to approximáte co's we have it already. If $\mathbb{P}$ is not directed-complete there will be directed subsets which do not have infima. These directed subsets can serve as proxies for the missing infima, the things we are trying to approximáte.

Let's take a breather and record some definitions

$\mathcal{P}$ is power set. $\mathcal{P}_{\aleph_0}(x)$ is the set of finite subsets of x.

x"y is $\{x(z) : z \in y\}$. The double apostrophe here isn't a piece of punctuation or a scare quote! This notation goes back to Russell and Whitehead. We will also use it even if x is not actually a function, merely a relation, a set of ordered pairs . . . so x"y is $\{z : (\exists w \in y)(\langle z, w\rangle \in x)\}$.

We can give $\mathbb{P}$ the order topology, which has a basis of "down-sets". I think this means we can take the basic open sets to be things of the form $\{y : y \leq x\}$. Using this topological language we say that

Definition 3
A set $D \subseteq \mathbb{P}$ of conditions is **dense** *iff it meets every basic open set;*
A subset $F \subseteq \mathbb{P}$ is a **filter** *if it is upward-closed and (downwards) directed;*
A filter F is **generic** *if it meets every dense subset.*

What are filters for? Directed sets of conditions correspond to the wombats that the conditions in that set are trying to approximáte. The upward-closedness is a technical condition we will ignore for the moment. What is the significance of *generic*? A generic filter houses, for every condition p, a condition at least as strong as p. So, if we think of it as a proxy for a wombat, it partakes somehow of all approximants; it is, indeed – in a loose sense of the word – *generic*.

Is a product of separative (resp directed) posets separative (resp directed)?

[2] What is the collective noun for a plurality of prewombats. . . ? While we are about it i learnt the other day that 'horde' is a Finno-Ugric word for an *encampment* . . . as in 'asiatic hordes'.

THEOREM 1 *The set of filters on a separateive poset form a separative directed-complete family.*

Proof
(fingers crossed!)
Suppose X and Y are directed subsets of a separative poset $\mathbb{P}$, with $X \not\subseteq Y$. We seek a directed subset Z with $Z \supseteq Y$ and $Z \perp X$.
$Z \perp X$ is equivalent to $(\exists z \in Z)(\exists x \in X)(z \perp x)$. $X \not\subseteq Y$, so there is $x \in X \setminus Y$. We seek $y \in Y$ with $x \not\leq y$. If there is no such y, then everything in $X \setminus Y$ is below something in Y, contradicting $X \not\subseteq Y$ (given certain conditions on X and Y). If $x \not\leq y$ then there is $z \geq y$ s.t. $z \perp x$.
Now consider the set

$$\{z : (\exists x \in X)(\exists y \in Y)(x \not\leq y \wedge y \leq z \perp x)\}$$

of such z and take a directed subset of it. That will be the Z we seek.
That's the idea, anyway!

■

But no! Dan Turetsky says: consider $2^{\omega+1}$. It gives a counterexample ... something to do with a ω-chain and a $\omega + 1$-chain.

There may be lots of generic filters for $\mathbb{P}$. The fact that every generic filter meets every dense set of conditions tells us quite a lot about a generic filter. There are some things that will happen whatever generic filter we have. This is useful because our means of showing that $\mathbb{P}$ has a generic filter might not give us a specific one. This is where the word 'forcing' comes from: a generic filter (or the novel object corresponding to it) is not guaranteed to have any special properties beyond what $\mathbb{P}$ "forces" it to have.

To recap:

- The *separativeness* of the poset of conditions is a manifestation of the fact that the set of things that approximáte wombats might do so in lots of incompatible ways, and thereby approximáte lots of distinct wombats.
- The *directedness* of a filter is mandated by the fact that each filter is a proxy for a single wombat, so all the conditions in the filter had better be compatible.

The idea is that a poset is suitable for forcing if it is separative. If it is separative then every directed subset approximátes a different thing.

2 Let's have some examples/non-examples

1. $\mathcal{P}_{\aleph_0}(X)$, the set of finite subsets of X, partially ordered by $\supseteq$.

   This is a non-example, co's it's not separative.

2. The set of nonprincipal ideals on a fixed infinite set X partially ordered by $\supseteq$. This is a separative poset. A generic filter, or rather its sumset, is a maximal nonprincipal ideal.

3. The set of nonzero elements of a boolean algebra (equipped with the obvious partial order) is always a downwards separative poset.

4. Let A and B be two infinite sets. A member of $\mathbb{P}$ is a finite subset of $A \times B$ coding a bijection between a finite subset of A and a finite subset of B; the ordering is $\supseteq$. This is a separative poset. A generic filter, or rather its sumset, is a bijection between A and B.

5. The set of binary rationals in $[0, 1]$ (denominator is a power of 2) thought of as finite bit-strings ordered by end-extension. This is a separative poset. A generic filter gives a real number.

6. The set of closed intervals in the rationals partially ordered by reverse inclusion. This is a separative poset. A generic filter gives a real number.

7. Let K be an arbitrary field and consider[3] the polynomial ring $K[T]$ with infinitely many variables. The conditions are proper, finitely generated ideals in $K[T]$, ordered by inclusion. The union of a generic filter is an ideal Γ in $K[T]$. $K[T]/\Gamma$ is the algebraic closure of K.

8. This one has a nice naïve feel to it. A condition is the deductive closure of a consistent finite set of set existence axioms: formulæ that say "$\{x : \phi\}$ exists"... the set of conditions partially ordered of course by $\supseteq$. I don't know offhand whether this poset is separative. If we didn't have the 'consistent' requirement then the poset would of course be directed and not separative. A generic filter (or rather its sumset) will be a theory in the language of sets that decides all questions "Does $\{x : \phi\}$ exist?"

One gets the feeling that the things one can add – *adjoin* – to a poset are the kind of things one wants to add to a wombat to make it a complete wombat. You complete the rationals by adding reals.

2.1 Comments on the Examples

No comment required on the first item.

Item (2)

Consider the set of nonprincipal [nonmaximal–i don't think it makes any difference if you include maximal ones, co's it's still separative] ideals in $\mathcal{P}(\mathbb{N})$. For any $X \subseteq \mathbb{N}$ the set of ideals containing either X or $\mathbb{N} \setminus X$ is a dense set, so the generic filter F must meet it: For every $X \subseteq \mathbb{N}$, F must either have an ideal containing X or an ideal containing $\mathbb{N} \setminus X$ (but not both!).

Now consider $\bigcup F$. Ideals are $\subseteq$-downwards-closed, so it's a union of downward-closed sets, so it's downward closed. For any $X \subseteq \mathbb{N}$ it contains precisely one of X or $\mathbb{N} \setminus X$. We want it to be closed under binary union. But this is easy: F is directed (that's part of the definition of generic filter) so that will ensure

[3] Thanks to François Dorais for this most intruiging example.

that $\bigcup F$ is closed under $\cup$. Thus $\bigcup F$ is a maximal (or *prime*) ideal. And nonprincipal.

This is another illustration of how there might not be a generic filter. The existence of prime ideals is a nontrivial choice principle.

Item 4

For any $a \in A$ the set of finite maps f with a in $\mathrm{dom}(f)$ is dense and for any $b \in B$ the set of finite maps f with b in $\mathrm{range}(f)$ is dense, so a generic filter gives us a bijection between A and B. This draws our attention to the very important fact that $\mathbb{P}$ might not have a generic filter, and that one has to do some work to establish that there is one. Every case is different. This is a plot point for set theory: it can happen (and this example is a case in point) that there is a generic filter that can be seen from *outside*, but is not present *in the model that we and* $\mathbb{P}$ *are in*. More on this later, but not for the moment!

Item 6

A useful exercise at this point is to tweak this example so that the collection of generic objects give us the p-adics.

Item 7

The first thing to get straight is precisely what the poset of conditions is. Observe that the poset $\mathcal{P}_{\aleph_0}(K[T])$ of finite subsets of $K[T]$ is directed, *not* separative, so that it isn't suitable for forcing. However the poset we are invited to consider is instead the poset of finitely generated ideals under $\supset$ which is a homomorphic image (send each finite set of polynomials to the ideal that it generates) of the poset $\langle \mathcal{P}_{\aleph_0}(K[T]), \subseteq \rangle$. It is this homomorphic image that we want to be separative. Suppose I and J are finitely generated ideals with $I \not\subseteq J$. Then there is a polynomial $p \in I \setminus J$. Since an ideal is closed under multiplication by an arbitrary ring element, any ideal that contains the polynomial $\mathbb{1}$ is the whole of $K[T]$. Add the polynomial $\mathbb{1} - p$ to J to obtain an ideal J' satisfying $I \perp J'$: any fg ideal $\supseteq$ both I and J' must be the whole of $K[T]$.[4]

So it's separative.

> This provides us with an illustration of a separative poset that is a homomorphic image of a directed poset. So directedness of posets is not preserved under homomorphic images. Tells us something about the logical complexity of those two notions.

Presumably, for each polynomial f with coefficients in K, the set of ideals I s.t. $K[T]/I$ contains a root of f is dense. That seems straightforward enough. That's certainly going to help when it comes to showing that $K[T]/\Gamma$ is algebraically closed.

[4] Why is J' proper?
Suppose not, then there would be $j \in J$ and a poly $a \in K[T]$ with $a(1-p)+j=1$
Hmm. It seems that we need p to have special properties.

The generic filter F is a directed family of finitely generated ideals. A directed union of ideals is an ideal, so $\bigcup F$ is an ideal. A directed union of *proper* ideals is a *proper* ideal, so $\bigcup F$ is a proper ideal. Why is it maximal? Dan Turetsky explains: For any poly a, $\{I : a \in I \vee a + I = K[T]\}$ is a dense set. After all, if a proper ideal I does not contain a you can safely add $a + \mathbb{1}$ to it to obtain a proper ideal.

We still need to show that everything in Γ is algebraic over K.

For each p and each I consider $p(x_n)$ where x_n not mentioned in the generators of I. Then consider the ideal $(I, p(x_n))$ The idea is that it will be a proper ideal and will contain a root of p.

And you have to show that the generic ideal is maximal.

For any poly $p(x1...xk)$ and any proper ideal I (I, p) or $(I, 1 + ap)$ some a in F is proper. If (I, p) is not proper then $i + ap = \mathbb{1}$ for some a in $K[X_i...], i \in I$. So $\mathbb{1} - ap = i$, so $(I, \mathbb{1} + ap) = I$ So for every p the set of ideals that contain p or $\mathbb{1} + p$ is dense.

finally we need alg clos. So for any poly $p(y)$ (with coeffs in the original field - this is enuff beco' of Zariski) and any generic ideal I need some j s.t $(I, p(x_j))$ is proper. Dan Turetsky says:

> "For Item 7, I'm confident I could write up the first two parts in detail (the ideal is maximal and the resulting field is algebraically closed). For the last part (that the resulting field is algebraic), I don't quite see all the details, but I'm still pretty sure it works.
>
> We need a lemma like this:
>
> If K is a field, and J is a finitely generated proper ideal of $K[x_1, \ldots, x_n]$, then there is a single-variable polynomial $p(x_1)$ such that $(I, p(x_1))$ is a proper ideal.
>
> Or equivalently:
>
> If K is a field, and J is a finitely generated proper ideal of $K[x_1, \ldots, x_n]$, then there is a homomorphism $K[x_1, \ldots, x_n] \to K$ fixing K elementwise, and with J contained in the kernel.
>
> Or just more generally:
>
> If K is a field, and M is a maximal ideal of $K[x_1, \ldots, x_n]$, then $K[x_1, \ldots, x_n]/M$ is algebraic over K.
>
> This last one is Zariski's Lemma.
>
> If F and K are both fields and K is f.g over F as a ring, then K is algebraic over F.
>
> I suspect all of these are true, but I'm not sure how to prove them."

Dan sez the key is Zariski's Lemma.

The reader should not run away with the idea that this is a cute way of showing that every field K has an algebraic closure. What has been shown is that a generic filter for this forcing is (or gives rise to) the algebraic closure

of K. There is nothing in the foregoing to say that *there will be* a generic filter. Remember item 4 which won't have a generic filter unless A and B are in bijection. There is no *general* reason why a separative poset should have a generic filter: the reasons are always *ad hoc*. That is why ingenuity is required.

3 Branching Quantifiers, Forcing, Ehrenfeucht games and Omitting Types

Consider two infinite structures $\mathcal{A}$ and $\mathcal{B}$ (with carrier sets A and B) that are elementarily equivalent but perhaps of radically different sizes, so that at any rate they are not isomorphic.

There is a branching-quantifier formula that says there is an isomorphism between them. It looks something like the following.

$$\left(\begin{array}{c}\forall a \exists b' \\ \forall b \exists a'\end{array}\right)((a = a' \longleftrightarrow b = b') \wedge \cong (a, a', b, b')) \tag{1}$$

(where a and a' are constrained to live inside A and b and b' are constrained to live inside B. $\cong (a, a', b, b')$ says that the two as look like the two bs.)

There is a Hintikka game treatment of this, analogous to the standard first-order case, which makes `true` and `false` into *teams* of two players. (The number 2 comes from the width of the antichain of quantifiers, which in this case is 2!). Team `false` starts: each member instantiates one of the two universal quantifiers. Then team `true` chooses witnesses for y and w *without collaborating*. That is to say, the member of `true` who chooses a witness for 'y' is told only about `false`'s choice for 'x', and the member of `true` who chooses a witness for 'z' is told only about `false`'s choice for 'w'. Members of team `true` are not allowed to impart information to each other. The game then proceeds as an ordinary Hintikka game, with the usual rules for termination.

This *aperçu* about teams gives rise (Barwise [1]) to a family of approximants for formulæ of this kind. The easiest way to present the family is to exhibit the first two or so. The first one is

$$(\forall a \forall b \exists b' \exists a')((a = a' \longleftrightarrow b = b') \wedge \cong (a, a', b, b')$$

The second is

$$(\forall a_1 \forall b_1 \exists a_2 \exists a_2)(\forall a_3 \forall b_3 \exists b_4 \exists a_4) \bigwedge \left(\begin{array}{l} \cong (a_1, b_4, b_1, a_2) \\ \cong (a_3, b_4, b_3, a_4) \\ a_1 = a_3 \rightarrow b_2 = b_4 \\ b_1 = b_3 \rightarrow a_2 = a_4 \end{array}\right) \tag{ϕ_2}$$

The second approximant says that if `false` picks the same x (resp. z) for a second time, then `true` must reply with the same y (resp. w) as used the first time. If we constrain `true`'s play in the obvious way, by saying that if she has

ever replied with a to a challenge b she must always reply with a to any future challenges with b we arrive at a family:

$$(\forall a_1 \forall b_1 \exists b_2 \exists a_2) \dots (\forall a_{2n-1} \forall b_{2n-1} \exists b_{2n} \exists a_{2n}) \left(\begin{array}{l} \bigwedge_{i \leq n} \cong (x_i, y_i, z_i, w_i) \\ \bigwedge_{i<j \leq n} (x_i = x_j \to y_i = y_j) \\ \bigwedge_{i<j \leq n} (z_i = z_j \to w_i = w_j) \end{array} \right) \tag{ϕ_n}$$

This has got garbled: what are the xs, ys, ws and zs doing??

Let's call this formula ϕ_n. The family of approximants indexed by $\mathbb{N}$ that we elaborate was shown by Barwise to have the following features.

- For all $n \in \mathbb{N}$, $\phi \to \phi_n$ is valid. The easy way to see this is to spot that it becomes easier for team `true` to win if they are told more of `false`'s choices for the universal quantifiers, so if they can win the branching quantifier formula they can certainly win the approximants.
- $\forall n \in \mathbb{N}$ $\phi_{n+1} \to \phi_n$ is valid.
- Any first-order theory consistent with all the ϕ_n is consistent with ϕ.

One good way to think of ϕ_n is through the following game. At stage n player `false` picks a_{2n-1} and b_{2n-1}, and player `true` replies with b_{2n} and a_{2n}. Unless $\cong (x, y, z, w)$, `false` wins at once. If $x_n = x_j$ for some earlier j then y_n must equal y_j or `true` loses. Similarly if $z_n = z_j$ for some earlier j then w_n must equal w_j or `true` loses. If `true` survives these hazards she procedes to stage $n + 1$. If the game goes on for ever `true` wins. Actually we want not this exact game but a trvial modification of it: `false` picks a member of A (or of B), and `true` responds with a member of B (or A). At every position in the game `true` must ensure that $\vec{a}$ has the same properties in $\langle \mathcal{A}, \vec{a} \rangle$ that $\vec{b}$ has in $\langle \mathcal{B}, \vec{b} \rangle$, where $\vec{a}$ and $\vec{b}$ are the members of A and B selected so far. If she doesn't she loses, and that is the only way she can lose, beco's if the game goes on for ever she wins.

This is the **Ehrenfeucht game**. There is apparently a theorem to the effect that if, for each n, `true` has a strategy to stay alive for at least n moves in the Ehrenfeucht game then $\mathcal{A}$ and $\mathcal{B}$ are elementarily equivalent, and *vice versa*.

Conveniently we find that the Hintikka game for ϕ_n is the truncation of the Ehrenfeucht game to n moves, where player `true` wins the truncation if she is still in the game after n moves. So ϕ_n is true iff `true` has a strategy to stay alive for n moves. Presumably there is a neat application of the completeness theorem to show that if all the approximants are true (so that, for each n, `true` has a strategy to stay alive for at least n moves in the Ehrenfeucht game) then $\mathcal{A}$ and $\mathcal{B}$ are elementarily equivalent. We will need to set up a two-sorted language ...

What if `true` has a strategy to stay alive indefinitely? In the Ehrenfeucht game an *even position* is a sequence of `false`'s moves paired off with `true`'s replies to make a finite function. ('Even' because it is `false`'s turn to play.)

Definition 4 *Let a* **condition** p *be an even position in the Ehrenfeucht game from which* `true` *has a winning strategy.*

The set of these ps is clearly a set of forcing conditions. Order them by reverse inclusion. [**Are they downwardly separative?**] Every condition can be extended both to one that has a_i in its domain – since `false` can always play a_i – and to one that has b_i in its domain – since `false` can always play b_i. This ensures that any generic object for this set of conditions is an isomorphism between $\mathcal{A}$ and $\mathcal{B}$.

So it appears that `true` has a winning strategy in the Ehrenfeucht game iff there is a forcing extension containing an isomorphism between $\mathcal{A}$ and $\mathcal{B}$.

Now let's start on a different tack and think in a general way about Skolemisation. Without serious loss of generality we can restrict attention to formulæ with no quantifiers within the scope of a propositional connective. The skolemisation of a formula is the result of deleting all existential quantifiers and replacing each occurrence of any existentially quantified variable – 'x', say – by '$f(y_1 \dots y_n)$' where the ys are all those variables bound by universal quantifiers in whose scope that occurrence of 'x' lies. Notice the cunning with which i have given this definition of skolemisation: it applies equally well to branching quantifier formulæ!

It is standard for ordinary predicate calculus that if ϕ is satisfiable then so is its skolemisation. It is also standard that this does not depend on AC. Of course appeal to AC is the obvious way to persuade the punters but it's not the reason why the result is true. I've always had the feeling that there is something funny going on there. I think it's a minor version of the funniness associated with the Hintikka game for branching quantifier formulæ. So let's think about the Skolemisation of the formula 1.

$$(\forall a)(\forall b)((a = f(b) \longleftrightarrow b = g(a)) \wedge \cong (a, f(b), b, g(a))) \tag{2}$$

The skolemised version says there is an *isomorphism* between $\mathcal{A}$ and $\mathcal{B}$. It might not be true: $\mathcal{A}$ and $\mathcal{B}$ just might not be isomorphic. However we would like the skolemised version to be (by analogy with the kosher first-order case) at least *consistent*. Now what on earth can this mean? We have seen that apparently `true` has a winning strategy for the Ehrenfeucht game iff there is a forcing extension containing an isomorphism between $\mathcal{A}$ and $\mathcal{B}$. I have heard people say that forcing is just omitting types, so perhaps the way to get a model of the skolemised version is to use omitting types ... (?)

$\mathcal{L}(T)$ was the language for $\mathcal{A}$ and $\mathcal{B}$, and they are both models for some theory T. Let our new two-sorted language $\mathcal{L}$ be the disjoint union of two copies of $\mathcal{L}$. In one copy we put T and a name for every member of A, and in the other copy we put T and a name for every member of B. We add a new

binary predicate letter I, whose two arguments belong to different sorts. Our axioms are now things like

$$(\forall a_1)(\exists b_1)\dots(\forall a_n)(\exists b_n)(\bigwedge_{1\leq i\leq n} H(a_i,b_i)\wedge(D_A(\vec{a})\longleftrightarrow D_B(\vec{b})))$$

where $D_A(\vec{a})$ is the conjunction of all atomic formulæ concerning $a_1\dots a_n$ and $D_B(b_1\dots b_n)$ is the conjunction of all atomic formulæ concerning $b_1\dots b_n$.

For each $a\in A$ let $\Sigma(a)$ be the type

$$\{\neg I(a,b):b\in B\}$$

and for each $b\in B$ let $\Sigma(b)$ be the type

$$\{\neg I(a,b):a\in A\}$$

and then somehow contrive to omit every $\Sigma(a)$ and $\Sigma(b)$

However this can't be correct because – for example – by Macdowell-Specker every model of PA has a proper elementary end-extension, and in at least some cases they are not isomorphic even seen from outside. One can't add an isomorphism by forcing. So presumably in some cases the set of conditions does not have a generic filter. Or at least we have to exploit the fact that `true` has a strategy to stay alive indefinitely rather than merely strategies to stay alive for n moves.

(Does this have anything to do with nice embeddings as in the $\forall_\infty$ stuff?)

Forwarded by James: Cutland's Review of Barwise

MR0342370 (49 #7116)

Barwise, Jon

Back and forth through infinitary logic. Studies in model theory, pp. 5–34. MAA Studies in Math., Vol. 8, Math. Assoc. Amer., Buffalo, N.Y., 1973. 02B25 (02H10)

This is largely an expository article built around the back and forth argument invented by Cantor, and particularly the notion of partial isomorphism and its relationship with the model theory of infinitary logic. At each stage the exposition is interlaced with interesting examples and applications.

Two structures $\mathcal{A}$ and $\mathcal{B}$ are said to be partially isomorphic ($\mathcal{A}\cong_p\mathcal{B}$) if there is a non-empty set I of isomorphisms of substructures of $\mathcal{A}$ onto substructures of $\mathcal{B}$ having the following back and forth property: for every $f\in I$ and $a\in A[b\in B]$ there is a $g\in I$ with $f\subseteq g$ and $a\in\mathrm{dom}(g)[b\in\mathrm{range}(g)]$. The generalisation of the back and forth part of Cantor's original argument (which showed that any two countable dense linearly ordered sets without end points are isomorphic) is the following theorem: If $\mathcal{A}$ and $\mathcal{B}$ are countable, then $\mathcal{A}\cong\mathcal{B}$ if and only if $\mathcal{A}\cong_p\mathcal{B}$.

The author introduces the first order infinitary language $L_{\infty\omega}$ and describes the basic model-theoretic notions $\mathcal{A} \equiv_{\infty\omega} \mathcal{B}$, $\mathcal{A} \prec_{\infty\omega} \mathcal{B}$ and the associated notions $\mathcal{A} \equiv_{\infty\omega}{}^{\alpha}\mathcal{B}$ and $\mathcal{A} \prec_{\infty\omega}{}^{\alpha}\mathcal{B}$ (indicating restriction to formulæ with quantifier rank $\leq \alpha$). He goes on to expound the relationships discovered by C. R. Karp [Theory of models (Proc. Internat. Sympos., Berkeley), pp. 407–412, North-Holland, Amsterdam, 1965; MR0209132 (35 #36)], namely, that $\mathcal{A} \equiv_{\infty\omega} \mathcal{B}$ if and only if $\mathcal{A} \cong_p \mathcal{B}$, and results in the same vein for $\prec_{\infty\omega}$, $\equiv_{\infty\omega}{}^{\alpha}$ and $\prec_{\infty\omega}{}^{\alpha}$.

He then presents an exposition of Scott's theorem, based on C. C. Chang's treatment [The syntax and semantics of infinitary languages, pp. 36–63, Lecture Notes in Math., Vol. 72, Springer, Berlin, 1968; see MR0234827 (38 #3141)], using the back and forth characterisations of $\equiv_{\infty\omega}$ and $\equiv_{\infty\omega}{}^{\alpha}$ obtained by Karp. A key result is the following theorem: Let $\mathfrak{A}$ be a structure with domain A; for any $a_1 \cdots a_n \in A$ there is a formula $\varphi(v_1 \cdots v_n)$ of $L_{\infty\omega}$ with $|\varphi| \leq \max\{|A|, |L|, \aleph_0\}$ such that, for any $\mathfrak{B}$ and $b_1 \cdots b_n \in B$, $(\mathfrak{A}, a_1 \cdots a_n) \equiv_{\infty\omega} (\mathfrak{B}, b_1 \cdots b_n)$ if and only if $\mathfrak{B} = \varphi[b_1 \cdots b_n]$. Scott's theorem is easily derived (for countable $\mathfrak{A}$, with $n = 0$), as is another result of Scott characterising the strongly invariant relations on a structure as precisely those definable in $L_{\infty\omega}$. There follows an account of the work of D. W. Kueker [The syntax and semantics of infinitary languages, pp. 152–165, Lecture Notes in Math., Vol. 72, Springer, Berlin, 1968; see MR0234827 (38 #3141)] on automorphisms of countable structures, presented from a back and forth point of view.

The author shows how back and forth arguments crop up in the theory of abelian groups by giving as examples (1) the result of Kueker that an abelian group H is $\aleph_1$-free if and only if $G \cong_p H$ (where G is the free group on $\aleph_0$ generators) and (2) the generalisation of Ulm's theorem discovered by P. C. Eklof and the author [Ann. Math. Logic 2 (1970/71), no. 1, 25–68; MR0279180 (43 #4906)].

The paper concludes with an interesting section containing metamathematical, philosophical, historical and bibliographical remarks. It is shown that $\cong_p$ is the strongest possible absolute notion of isomorphism, and thus "one of which mathematicians should be aware".

The paper is clearly written and a pleasure to read; to quote the editor of the volume in which it appears, "the author has accomplished the nearly inconsistent task of writing an article useful both to those ignorant of model theory and to those expert in it".

For more complete bibliographic information about the collection in which this article appears, including the table of contents, see MR0327503 (48 #5845).

Reviewed by Nigel J. Cutland

4 Where shld this go?

Let's Skolemize ϕ_n. y_i depends on $x_1 \ldots x_i$ and $z_0 \ldots z_{i-1}$ and w_i depends on $x_1 \ldots x_i$ and $z_1 \ldots z_i$. Using $f_i(x_1 \ldots x_i, z_0 \ldots z_{i-1})$ for y_i and $g_i(x_1 \ldots x_i, z_1 \ldots z_i)$

for w_i we obtain

$$(\forall x_1 \dots x_n)(\forall z_0 \dots z_{n-1}) \begin{cases} \bigwedge\limits_{i \le n} A(x_i, f_i(x_1 \dots x_i, z_0 \dots z_{i-1}), z_i, g_i(x_1 \dots x_i, z_1 \dots z_i)) \\ \bigwedge\limits_{i<j \le n} (x_i = x_j \to f_i(x_1 \dots x_i, z_0 \dots z_{i-1}) = f_j(x_1 \dots x_j, z_0 \dots z_{j-1})) \\ \bigwedge\limits_{i<j \le n} (z_i = z_j \to g_i(x_1 \dots x_i, z_0 \dots z_i) = g_j(x_1 \dots x_j, z_0 \dots z_j)) \end{cases}$$

(ϕ_n: Skolemized)

Team `true` have the Skolem functions f_i and g_i to play with. The second and third row conjunctions say that the values of f_i depend only on its x inputs and the values of g_i depend only on the z inputs. If `true` cheat they do not respect this constraint and by the rules of the Hintikka game they lose.

5 Forcing in Set Theory

At last!! Forcing in Set Theory!!

> It would be a useful exercise to write out how to do forcing in CUS.
> Things to check:
> If i do a forcing within a forcing extension, i cld have done it in one hit?
> If I have a generic filter in $\mathbb{P} \times \mathbb{P}'$ then i can recover generic filters in $\mathbb{P}$ and $\mathbb{P}'$ ...

Set Theory is the *locus classicus* for forcing; it was for the solution of open problems in Set Theory that forcing was cooked up in the first place. Set theory has a different feel to it from a lot of the rest of mathematics, beco's in Set Theory one often has quite explicitly in mind the concept of a *model*, a toy universe within which one is working. Consider item 4; is there a bijection between A and B? Not on the bench in my workshop perhaps, but maybe if i cast my net a little wider...? So here is a potential piece of fun. Suppose i have a model $\mathfrak{M}$ of set theory in which i have two sets A and B, which – according to $\mathfrak{M}$ – are not in bijection: $\mathfrak{M}$ does not contain a bijection between A and B. However $\mathfrak{M}$ contains a forcing (item 4 above) with the property that a generic filter for it gives a bijection between A and B. Perhaps by reasoning about $\mathfrak{M}$ in the (larger) model wherein I dwell i can prove that there is – in my world – a generic filter for that forcing. Then i have shown that *in my world* there is a bijection between A and B. This is actually quite easy if $\mathfrak{M}$ is countable, and we will spell out how to do it. The hard part is show how to fit the bijection into a larger model, to build a larger model $\supset \mathfrak{M}$ in which A and B are in bijection.

It might seem obvious that if A and B live in a countable transitive model $\mathfrak{M}$ [of ZFC, to make matters calmingly concrete and specific] which believes them to be of different sizes[5] then there is a bijection between them which is

[5] There was an old man of Devizes
whose ears were of different sizes.

not present in $\mathfrak{M}$. What do we need the forcing for? We need it beco's we are trying to not only show that A and B can be made to look the same size (that follows easily from their residence in a countable transitive model, which makes them both countable – everything in a countable transitive model is countable) but to situate them inside an actual model $\mathfrak{M}' \supseteq \mathfrak{M}$ of – in this case – ZFC. To do this, it is not sufficient to have the bijection between A and B dropped on us by a passing seagull, we want to know how it was approximated from within $\mathfrak{M}$. We can then use the approximants to construct a model around it.

To construct this new model we need...

5.1 Names

We will declare

(i) a recursive datatype of names;
(ii) a recursive function that gives a name to each set;
(iii) we have a (recursive) binary valuation function that takes a name and a set of conditions and gives a set.

First the rectype of names. Names are objects in the ground model which, when evaluated with the aid of a set of conditions, denote things in the world in which we are constructing the new model. Specifically the idea will be that if we evaluate all names with respect to a nice set of conditions (plot point: a generic filter) then the things we get will comprise a model of ZFC (or whatever the base theory was that we started with). Since the aim of the exercise was to construct (by means of the generic filter) a new larger model containing the thing the conditions were approximating we don't much care what happens if we evaluate a name with respect to sets of conditions that aren't generic filters, but the process is at least well-defined.

The reason for names to be a recursive datatype is not so much that *formulæ* constitute a recursive datatype (tho' they do) but rather the fact that the relation $\in$ is wellfounded and we do a lot of things by recursion on it. The context for all this, let us not forget, is the theory of wellfounded sets.

We have a domain equation (recursive definition)

Definition 5 $\mathbf{names} =: \mathcal{P}(\mathbf{names} \times \mathbb{P})$

So a **name** is simply a set of tagged **name**s.
A tagged **name** is an ordered pair of a **name** with a condition.
So **names** is the least fixed point for $N \mapsto \mathcal{P}(N \times \mathbb{P})$.

Least fixed points support Induction and Recursion, So we can perform inductions and recursions on **names**. For example, if π is an isomorphism between two posets $\mathbb{P}_1$ and $\mathbb{P}_2$ then we can lift it to an isomorphism between

The one that was small
was of no use at all;
the other won several prizes.

names over $\mathbb{P}_1$ and names over $\mathbb{P}_2$: $\pi(n)$ is going to be n^π. (R^π is of course $\{\langle \pi(x), \pi(y)\rangle : \langle x, y\rangle \in R\}$.)

This has the startling consequence that what you get by forcing with a poset $\mathbb{P}$ depends only on the order structure of $\mathbb{P}$, and not at all on what the points are that are being ordered.

The relation:

$$a \in b\text{“}\mathbf{names}$$

between two **names** a and b is wellfounded.

Next we declare a function that will give every set a **name**:

Definition 6
The function $\check{C}$ taking values in **names** *is declared by the $\in$-recursion*

$$\check{C}(a) =: \check{C}\text{“}a \times \mathbb{P}$$

We could in fact have defined $\check{C}$ by $\check{C}(a) =: (\check{C}\text{“}a) \times \{\mathbb{1}\}$ and it would still serve its purpose of giving every set a **name**. In fact some sources (e.g. wikipædia) actually do this. Comment. . . ?

Not every name is a value of $\check{C}$: there are to be names of things that lie outside the ground model! $\check{C}$ is a total function: every set has a name, and $\check{C}(a)$ is a name of a. Use of the word 'name' in this context suggests that we are going to be able to recover a from $\check{C}(a)$ – names are supposed to point to the things they name, after all. We need a function that will recover sets from names. It will be a kind of valuation function, a binary function that takes a name and a set of conditions and evaluates the name to a set using the set of conditions.

Definition 7 *For each set F of conditions, we define a function v_F on names by the recursion*

$$v_F(a) =: v_F\text{“}(a\text{“}F)$$

'v' connotes $v$aluation. In the Set Theoretic literature one tends to see ' a^F ' rather than '$v_F(a)$'. This is a nice snappy notation, specially made up for this context, but I prefer '$v_F(a)$' to 'a^F' because we want to bring out the idea of a valuation (and the notation 'a^F' conceals the process of evaluation). We can't write '$v(F, a)$' beco's the recursion obliges us to apply v one level down as well, as in " $v_F\text{“}X$ ". And we cannot write *that* unless we first "curry" v.

> But think what we are saying by using this notation! We are saying that the difference between a function and its curried version is purely a difference in notation!
> Something to do with the fact that we can use the double apostrophe notation only with one-place functions? If f is dyadic we can write $f\text{“}(X \times X)$ but that doesn't help us here.
> When you put it like that, the S-m-n theorem is all about currying.

Notice that $a \mapsto a``F$ might not be injective, so $v_F$ might not be injective: a thing might end up with more than one name. It will turn out that the collection $v_F``\mathbf{names}$ is a model of ZF, and we notate it $\mathfrak{M}[F]$. Is this correct?

We will invoke v_F only when F is a filter (indeed a *generic* filter) but it makes sense for F any set of conditions. You don't get a *sensible* answer unless F is a generic filter, but you do get *something*.

I think [**check this**] that, for each name a, the function $\lambda X.v_X(a)$ is a $\subseteq$-monotone increasing function on $\mathcal{P}(\mathbb{P})$.

We write '$\mathfrak{M}[X]$' for $v_X``M$. (M is the carrier set of $\mathfrak{M}$.) In particular $\mathfrak{M}[\{\mathbb{1}\}] = M$ (or $\mathfrak{M}$ by abuse of notation). So what is $\mathfrak{M}[\mathbb{P}]$? Presumably it's the union of all forcing extensions. [why is the point not made in the textbooks?? Or have i misread something?]

Lemma 1 *Let F be a nonempty subset of $\mathbb{P}$. Then $v_F(\check{C}(a)) = a$.*

> (The idea is that $\check{C}(a)$ will always evaluate to a. Other names might evaluate to complex fancy useful things not in $\mathfrak{M}$, but $\check{C}(a)$ will always give you back a – whatever subset of $\mathbb{P}$ you use.)

Proof:

Expanding $v_F(\check{C}(a))$ using definition 7 we obtain

$$v_F(\check{C}(a)) = v_F``(\check{C}(a)``F)$$

and then expanding $\check{C}(a)$ on the RHS using definition 6 we obtain (A):

$$v_F(\check{C}(a)) = v_F``((\check{C}``a) \times \mathbb{P})``F \tag{A}$$

Now $F \subseteq \mathbb{P}$ so $(\check{C}``a \times \mathbb{P})``F = (\check{C}``a \times F)``F$, and the RHS of this last equation is obviously $\check{C}``a$, so the RHS of (A) will simplify to $v_F``\check{C}``a$ which is of course $(v_F \cdot \check{C})``a$. So

$$(v_F \cdot \check{C})(a) = (v_F \cdot \check{C})``a$$

which powers a proof by $\in$-induction that $v_F \cdot \check{C}(a) = a$.

■

I think all we have used in this is that F is a nonempty subset of $\mathbb{P}$. For which sets $\mathcal{C}$ is $v_{\mathcal{C}}$ a left-inverse to $\check{C}$? Is it necessary and sufficient that $\mathcal{C}$ be a nonempty subset of $\mathbb{P}$? Is this correct??

So v_X applied to a name of a thing-in-the-model always gives you back the thing named, never mind what X is. No change there. However the behaviour of v_X wrt things *not* in the range of $\check{C}$ will depend very sensitively on X. In particular if X is a generic filter interesting things can happen.

> Now **names** is a wellfounded structure (it has a recursive declaration) so there will be a transitive collapse. What is this transitive collapse? The transitive collapse of **names** is performed by $v_{\mathbb{P}}$. Presumably we get V. But if we whack **names** with v_F we get something different. Some of the things we get are not members of the model we started with, so the fact that F is a proper subset of $\mathbb{P}$ doesn't straightforwardly mean that $v_F``V$ is a proper subset of $v_{\mathbb{P}}``V$.
>
> What is $v_{\mathbb{P}}(a)$? It must be $v_{\mathbb{P}}``(a``F)$ Eurgh ...

5.1.1 A Name for the Generic Filter

There is a special name, g, which is $\check{C}\restriction\mathbb{P}$. Its significance is that it is a name for the generic filter. But first we'd better check that it really is a name. g is the set $\{\langle p, \check{C}(p)\rangle : p \in \mathbb{P}\}$. So it's a set of pairs whose first component is a name and whose second component is a condition, and that makes it a name. In fact *any* subset of the graph of $\check{C}$ will be a name.

Check that this is OK

Lemma 2 *$v_F(g) = F$ for all filters F.*

Proof:

From definition 7 we have

$$v_F(g) = v_F\text{“}(g\text{“}F)$$

Now $g\text{“}F = \check{C}\text{“}F$ so

$$v_F(g) = (v_F \cdot \check{C})\text{“}F = F.$$

The last equality holds because, by lemma 1, v_F and $\check{C}$ are inverses: $v_F \cdot \check{C} = \mathbb{1}$ ∎

This ensures that $F \in \mathfrak{M}[F]$.

Dan T tells me that it can happen that we have three names a, b and c such that there are two generic filters F_1 and F_2 with $\mathfrak{M}[F_1] \models a = b \wedge b \neq c$ and $\mathfrak{M}[F_2] \models a \neq b \wedge b = c$. It would be a useful exercise to exhibit a bundle of such things.

6 Internalising Forcing

We define a relation $\Vdash$ between conditions (on the left) and complex objects of a special kind (on the right).

In the first instance we have things like "$p \Vdash x \in y$" and "$p \Vdash x = y$". The 'p' on the left (as i have said) is a condition. The 'x' and 'y' on the right are **names** and the '$\in$ and '$=$' are symbols for two relations between names. The two relations beween names are of course not *actual* set membership and equality, but they *evaluate* to $\in$ and $=$, and that is the pretext for the overloading. The idea is going to be that $p \Vdash x \in y$ iff, for every generic filter F with $p \in F$, then $v_F(x) \in v_F(y)$.

Let's consider the simplest cases: $p \Vdash x \in y$ and $p \Vdash x = y$. Let start with this first.

It looks *prima facie* that $p \Vdash x \in y$ ought to be:

$$(\forall F)(p \in F \to v_F(y) \in v_F(y))$$

(where the letter 'F' ranges over generic filters) ... And that looks to me as if it ought to simplify to

$$\langle y, p\rangle \in x$$

That looks good, beco's it's a statement inside $\mathfrak{M}$. **However** that is not what you see in the textbooks; what you find is a mutual recursion on $\in$ and $=$. This is presumably beco's of two things: extensionality and the wellfoundedness of $\in$ alluded to above. So this feature is not really a feature of forcing, it's a consequence of our doing forcing in ZF(C).

The idea is that we will extend this definition of $p \Vdash x \in y$ and $p \Vdash x = y$ to situations where the thing to the right of the '$\Vdash$' is an object that evaluates under a v_F to a boolean, a thing with a truth-value. We read '$p \Vdash$ [stuff]' as p forces [stuff].

It will turn out that every statement is forced by some condition or other. This is striking because of the fact that a generic filter could be arbitrarily nasty and contain some very weird information. What this is going to mean is that even if the generic real that you add encodes a countable model of ZFC + existence of a measurable cardinal there is no way of decoding it to reveal the countable model.

A filter that meets every dense subset could be incredibly complex: that characterisation of 'generic' doesn't place any upper bound on how much information such an object might contain. However we do have some sort of control over what all the generic filters can agree on, and by 'we' i mean the ground model. That is to say, we can put a lower bound on the information that all forcing extensions agree on. For at least some expressions ϕ (depending on $\mathbb{P}$) we can ensure that all generic extensions satisfy ϕ. For example if our $\mathbb{P}$ is the set of finite sequences of countable ordinals then every forcing extension collapses ω_1.

(dfn of generic filter here)
$a^F =: \{b^F : (\exists p \in F)(\langle b, p\rangle \in a)\}$

> What is the base case? If a doesn't contain any ordered pairs then a^F is $\emptyset$? And if a is a piece of random rubbish that happens to contain a single pair $\langle b, p\rangle$ with $p \in P$ then a^F is $\{b\}$ iff $b \in F$, `else` it's $\emptyset$?

So what a^F turns out to be will depend on F. For each generic filter F there is a model $\mathfrak{M}[F]$ which is $\{a^F : a \in M\}$. Clearly $\mathfrak{M}[F]$ is a transitive set, by construction. What sets are in it? Now, for each $a \in M$ there is an $\check{a}$ such that for all generic F, $\check{a}^F = a$, and it is defined by $\in$-recursion:

$$\check{a} =: \{\langle \check{b}, \mathbb{1}\rangle : b \in a\}$$

[Could we take $\check{a}$ to be $\{\check{b} : b \in a\} \times P$?]

Let's simplify $\check{a}^F$ and keep fingers crossed that we get a.

$$(\check{a}^F =: \{b^F : (\exists p \in F)(\langle b, p\rangle \in \check{a})\}.$$

Now if $\langle b, p\rangle \in \check{a})$ we have $\langle b, p\rangle \in \{\langle \check{c}, \mathbb{1}\rangle : c \in a\}$, so b is $\check{c}$ for some c and $b = \mathbb{1}$. So

$$(\check{a}^F =: \{(\check{b})^F : b \in a\}.$$

and this means we can prove $(\check{a})^F = a$ by $\in$-recursion. The only fact about F that we have used is that $\mathbb{1} \in F$.

This of course has the effect that $\mathfrak{M} \subseteq \mathfrak{M}[F]$ for all F. That's not to say that all the $\mathfrak{M}[F]$ are the same. Consider the following object:

$$g =: \{\langle \check{p}, p\rangle : p \in \mathbb{P}\}$$

We check that $g^F = F$, so that $F \in M[F]$:
$g^F = \{a^F : (\exists p \in F)(\langle a, p\rangle \in \{\langle \check{p}, p\rangle : p \in \mathbb{P}\})\}$
Now if $\langle a, p\rangle \in \{\langle \check{p}, p\rangle : p \in \mathbb{P}\}$ then a must be $\check{p}$ for some $p \in \mathbb{P}$. So a^F is $(\check{p})^F$ for some $p \in \mathbb{P}$. But $(\check{p})^F$ is p, as we showed above.

If F is an $\mathfrak{M}$-generic filter on P then $\mathfrak{M}[F]$ is not only transitive (as we have seen) but is also a model of all sorts of formulæ true in $\mathfrak{M}$. Specifically if $M \models ZFC$ then $\mathfrak{M}[F] \models ZFC$.

Let us check that $\mathfrak{M}[F] \models$ pairing. Let a^F and b^F be two things in $\mathfrak{M}[F]$. If $c = \{\langle a, \mathbb{1}\rangle, \langle b, \mathbb{1}\rangle\}$ then $\mathfrak{M}[F]$ certainly believes that c is the unordered pair of a and b. (We haven't used the fact that F is a filter, let alone generic – merely the fact that $\mathbb{1} \in F$.)
Similarly we can check that $(a \cup b)^F = a^F \cup b^F$ so $\mathfrak{M}[F] \models$ binary unions.

In an earlier version i had written down:
"We say $p \Vdash \phi$ iff $(\forall$ generic $F)(p \in F \to \mathfrak{M}[F] \models \phi)$."
It's not *prima facie* evident that this relation is nonempty!
The major achievement is going to be reducing '$p \Vdash \phi$' to something that doesn't mention any Fs but talks only about the poset $\mathbb{P}$ and the syntax of ϕ. Something Dan Turetsky has just said to me makes me think that one should think of the equivalence of these two characterisations of forcing as a completeness theorem. Saying $p \Vdash \phi$ iff for every generic F with $p \in F$ we have $\mathfrak{M}[F] \models \phi$ involves the valuation function and is therefore *semantic*. If it really is a completeness theorem then the syntactic $\to$ semantic direction should be easy, and the other direction hard(er).

It turns out that this completeness theorem is called **The Truth Lemma**.

$$\mathfrak{M}[F] \models \phi \longleftrightarrow (\exists p \in F)(p \Vdash \phi)$$

$R \to L$ is immediate: it's $L \to R$ that is hard to arrange for.
Clearly the syntactic definition of $p \Vdash \phi$ is going to be by recursion on the logical structure of ϕ, so we have to start by defining $p \Vdash a = b$ and $p \Vdash a \in b$. These in turn have to be done by a recursion on $\in$.

$$p \Vdash a \in b \longleftrightarrow (\forall q \leq p)(\exists \langle c, r\rangle \in b)(\exists s \leq q, r)(s \Vdash a = c)$$

and

$$p \Vdash a = b \longleftrightarrow (\forall \langle c, r\rangle \in a)(\forall s \leq p, r)(\exists \langle d, t\rangle \in b)(\exists u \leq s, t)(u \Vdash c = d) \ldots$$

with a *vice versa* clause in the second definition.

Let's just check that if $p \Vdash a \in b$, and $p \in F$, then $\mathfrak{M}[F] \models a \in b$, which is to say, $a^F \in b^F$. Specialise q to p in the RHS of the definition, and we get $(\exists \langle c, r\rangle \in b)(\exists s \leq r, p)(s \Vdash a = c)$, which is to say that there is c s.t. $c^F \in b^F$ which some stronger condition believes to be equal to a. Well, it's a start.

Presumably we prove the truth lemma by induction on logical complexity.

References

[1] Barwise, J. Some applications of Henkin quantifiers. Israel J of Maths **25** 1976 pp 47-63.

[2] Barwise, J. On branching quantifiers in english JPL v 8 pp 47-80

[3] M. Randall Holmes "Forcing in NFU and NF" in *A Tribute to Maurice Boffa* ed Crabbé, Point, and Michaux. Supplement to the December 2001 number of the *Bulletin of the Belgian Mathematical Society*.

[4] Thomas Jech "The Axiom of Choice" North Holland. 1973

[5] Thomas Jech "Set Theory: The Third Millenium Edition" Springer 2002