A Summer Project

Thomas Forster

February 12, 2022

Yuanyuan Shen ¡ys564@cam.ac.uk¿, Oli Wilshaw ¡osw26@cam.ac.uk¿, Alexander Darby ¡atd39@cam.ac.uk¿, Daniel Turaev ¡dt493@cam.ac.uk¿ Vaios-Rafail Michalakis vrm27@cam.ac.uk Callum Hobbis ¡ch869@cam.ac.uk¿

weakly stratified arises from stratified in a way one should explain. What arises from acyclic in the saame way?

Calliope has shown that the class of stratifiable formulae is not context free but that the corresponding typed language is context free. I don't think he looked at the complements of these languages and as far as i know these questions are open Set of acyclic formulae not CF? Set of cyclic formulae CF?

What about the set of formulae that are stratifiable-mod-n? Stratified-mod-n? The project concerns the following topics:

(i) Proof Theory (specifically cut-proofs and the subformula relation);
(ii) Formal language theory (specifically context-free languages);
(iii) Rewrite systems.

(i) isn't taught. You pick it up behind the bike-sheds; (ii) is taught in the Maths Tripos at Part II (Languages and Automata); and in the Computer Science tripos (in 1a DM and 1b Computation Theory); (iii) isn't taught.

But all this stuff can be picked up quite easily. Wikipædia is quite good for this sort of thing. I can suss out more detailed references if called upon to.

If there is a unifying theme it is a background (well, my background!) in Set Theory. Have i ever bent your ear about NF? It's my particular baby, so naturally it is a source (for me) of fun questions. However the questions that follow do not require the investigator to know much about NF: they arise *from* NF not *within* NF, as it were.

There are connections here with *unification* in resolution theorem-proving. But none of you are compscis so that might be a bridge too far. But if you like the idea i can provide you with some briefing.

Contents

1 Some Syntactic Concepts

1.1 Stratifiable and weakly stratifiable formulæ

1.1.1 Stratifiable formulæ

NF trades on the idea of a **stratifiable** formula of the language of set theory. A formula of the language of set theory is stratifiable iff you can decorate the variables within it with integers in such a way that

(i) every occurrence of any one variable received the same decoration;

(ii) if '$x = y$' is a subformula then 'x' and 'y' get the same decoration, and

(iii) if '$x \in y$' is a subformula then the decoration on 'y' is (decoration-on-'x') + 1.

Such a decoration of a formula is a *stratification*. We say that it *stratifies* the variables in the formula. If we have a stratification that stratifies the *bound* variables of ϕ (but perhaps not the free variables) we say that ϕ is *weakly stratifiable*. Weakly-stratifiable turns out to be a surprisingly important notion. Anyway the idea behind NF is that if you insist that in any comprehension axiom giving you $\{x : \phi\}$ then ϕ has to be stratifiable then—on the face of it at least—you don't get the Russell class. That's the idea, anyway.

1.1.2 Weakly Stratifiable

Weakly stratified sounds like an annoying complication, and in some sense it is, but we need it, and for two reasons.

(i) Consider the axiom

$$(\forall x, y)(\exists z)(\forall w)(w \in z \longleftrightarrow (w \in x \vee w = y))$$

This is a stratifiable expression, and is an axiom of NF. It says that $x \cup \{y\}$ always exists. So, in particular, $x \cup \{x\}$ always exists, but this isn't stratifiable. Now consider what a proof that $x \cup \{x\}$ always exists must look like. We get a proof that $x \cup \{y\}$ always exists (it's an axiom, as we have seen). So we do a $\forall$-elimination (two, in fact) to get that $x \cup \{x\}$ exists, and we then do a $\forall$-introduction to deduce that $(\forall x)(X \cup \{x\}$ exists) But this proof has a maximal formula (or, if we do it in sequent calculus, a cut-proof) and that is an *infelicity*. The solution is to have $(\forall x)(\exists z)(\forall w)(w \in z \longleftrightarrow (w \in x \vee w = x))$ as an axiom. Even tho' it isn't stratifiable! No it isn't, but it is *weakly stratified*.

(ii) The other reason why we need weak stratification concerns the subformula property for cut-free proofs, of which more below.

1.2 Acyclic Formulæ

The concept of an acyclic formula of the language of set theory is a comparatively new one, but it's elementary. It was invented by an Iraqi neurologist by the name of Zuhair Abdul Ghafoor Al-Johar who is also an amateur set-theorist. It has turned out to be surprisingly fruitful. Here is the definition: take your formula

ϕ, shake out all the variables into a little pile on your desk, and use them as vertices of a graph G_ϕ. You put an edge between two vertices if there is an atomic subformula of ϕ in which they both appear. You then ask if the graph G_ϕ is cyclic. Does it have loops? If it doesn't, you say the formula is acyclic. The axioms of empty set, of pair set and of unions are all acyclic. Notice that acyclic implies stratifiable but not *vice versa*.

Very much to everyone's surprise it turned out that comprehension for acyclic formulæ was equivalent to comprehension for stratifiable formulæ. That was proved in [**?**]. Even more to everyone's surprise is the news that every stratifable formula is equivalent (modulo some really Noddy Set Theory) to an acyclic one. This was proved by Nathan Bowler, but he has never published it. I have asked him for permission to include as an appendix to the final version of this document the pdf of his proof wot he sent me. I do not understand it. One item of this project is to understand this proof and explain it to me! This is the topic of section **??**.

1.3 Stratification-mod-n

There is a more relaxed notion of stratification which uses integers mod n rather then integers. For example '$x \in y \wedge y \in x$' is stratifiable mod 2. All the discussion about subformulæ that one has in connection with stratifiability can be had here too. I am guessing that there are theorems of FOL that are stratifiable-mod-2 but have no cut-free proof in which every formula is stratifiable-mod-2. Probably not *terribly* interesting but one never knows. In any case it's virgin territory.

2 The Subformula Relation

[**?**] might be a good place to start looking.

The concept of a subformula is more important than you might think. Typically it is wellfounded, and lots of theorems are proved by induction on it, and lots of functions from formulæ to formulæ—such as translations from one language to another—are defined by recursion on it.

There is also the subtly different notion of subformula in play in proof theory, where we say that cut-free proofs have the *subformula property* for cut-free proofs. In a cut-free proof every formula that appears is a subformula of the thing being proved—subformula in a subtly different sense. For example, in this sense '$\phi(t)$' is a subformula of '$(\exists x)\phi(x)$' for any term t, however large. (If you look at the sequent rules for the quantifiers you see that we have to define 'subformula' in this more inclusive way). The subformula property for cut-free proof means there is no cut-free proof of the `false`. So: *if* you can prove, for a theory T, that every theorem has a cut-free proof, *then* you have proved that T is consistent. Lots of fun to be had there. Get straight the various concepts of subformula, what they are and why you need them. This will inform your study of proof theory and (if you are so inclined) resolution proofs (`PROLOG` etc. But only if you are so inclined!

A natural question to ask about a class of formulæ is "is it closed under subformula?"

A subformula of a weakly stratifiable formula is weakly stratifiable, but not all subformulæ of stratifiable formulæ are stratifiable. For example '$x \in y$' is stratifiable, but it has the substitution instance (and therefore subformula) '$x \in x$' which is not stratifiable. The cut-elimination theorem for first-order logic says that every theorem has a cut-free proof. Every stratifiable theorem has a cut-free proof, but it may be that every cut-free proof of it has an unstratifiable formula in it. Examples worth thinking about are the following FOL tautologies "If no member of x is the universal set then neither is x"; "If x is the universal set then there is something that it is a member of".

However every *weakly* stratifiable formula has a cut-free proof wherein every formula is weakly stratifiable. This is because the class of weakly stratifiable formulæ is closed under subformula.

Someone needs to look at how 'acyclic' and 'weakly acyclic' behave in this regard. Presumably the situation is analogous, but no-one has ever sat down and worked it out. That would be a good thing to think through.

(Come to think of it i don't know off the top of my head what the correct definition of 'weakly acyclic' would have to be. It would have to meet the analogues of the two concerns which the concept of 'weakly stratifiable' address.)

3 Context-free?

This section is rather a jumble at the moment

The set of stratifable formulæ and the set of acyclic formulæ are presumably neither of them context-free. This is a specialisation of an old question about whether or not the language of first-order logic is context-free. There is some stackexchange traffic on this, not much of it helpful. Of course if the signature of the languages is infinite then it's not CF, but there may be useful things one can say in the finite case. And the case i am interested in is the language of set theory which has only two predicate letters and no constants.

There are various languages we encounter in this setting, and—to the best of my knowledge—nobody has applied themselves to the question of whether or not they—any of them—are context-free. I rather expect none of them are, and that it would be fairly easy to prove. It can do no harm to work through these questions and write up something sensible.

Is the language of set theory context-free?
Is the language of stratifiable formulæ of set theory context-free?
Is the language of weakly stratifiable formulæ of set theory context-free?
Is the language of acyclic formulæ of set theory context-free?

and so on.

Of course a stratifiable formula is not the same as a stratified formula; a stratified formula is equipped with a decorations, a *stratification*. Is the language of stratif**ied** formulæ of set theory context-free? I'm guessing that adding the decorations doesn't affect the yes/no answer, but one never knows. Ditto weakly stratified too of course.

Work with the first-order language of set theory, augmented with terms. A term is either a variable or a set abstract $\{x : \phi(\vec{y})\}$ (parameters allowed). The reader is assumed to know what a stratifiable formula is. A substitution instance of a formula is the result of replacing an occurrence of a free variable in it by a term.

It is not 100% clear to me that the set of stratifiable formulæ of this language is context-free, so that is my first question. I have an awful feeling that it might not be context-free.

Fix a stratifiable formula with at least one free variable (o/w there is nothing to do) and consider:

(i) the class of substitution-instances of it; and
(ii) the class of stratifiable substitution-instances of it.

I'm guessing that (i) is a context-free language, but that (ii) might not be. Or perhaps both are context-free, and in order for this difference to show up one has to consider the formulæ obtainable from a *set* of stratifiable formulæ. That is to say, fix the language as above, and let Γ be a context-free set of formulæ all of whose members are stratifiable. Consider

(i) the class of substitution instances of members of Γ; and
(ii) the class of stratifiable substitution-instances of members of Γ.

I'm guessing that (i) is a context-free language, but that (ii) might not be.

4 Set Abstraction and Rewriting

Start with the language of set theory, with $\in$ and $=$. Add some apparatus of set abstraction, namely the ability to form—recursively—terms of the sort $\{x : \phi\}$ for any expression ϕ of the language. (There is no stratification restriction at this stage of the narrative).

In this connection pause to take on board the *aperçu* of Randall's that any closed formula ϕ is equivalent to the comprehension axiom giving $\{x : \neg\phi \wedge x \notin x\}$. For all i know it may even work when ϕ has free variables as long as 'x' is not free in ϕ.

Let me write this out ....

If ϕ is true (where 'x' is not free in ϕ) then $\{x : \neg\phi \wedge x \notin x\}$ is $\{x : \bot\}$ which exists.

If ϕ is false then $\{x : \neg\phi \wedge x \notin x\}$ is $\{x : x \notin x\}$ which does not exist.

So ϕ is equivalent to the existence axiom for $\{x : \neg\phi \wedge x \notin x\}$.

So (and i think this works constructively, even) every sentence is equivalent to a parameter-free comprehension axiom. What happens if we allow parameters?

The following rewrite rules spring to mind:

Input		**Output**	**Name of rule**
$\Psi(\{x:\phi\})$	$\Longrightarrow$	$(\exists y)(\Psi(y) \wedge ((\forall x)(x \in y \longleftrightarrow \phi) \wedge$ $(\forall w)((\forall x)(x \in y \longleftrightarrow \phi) \rightarrow w = y)))$	(King of France)
$\Psi(\{x:\phi\})$	$\Longrightarrow$	$(\forall w)((\forall x)(x \in w \longleftrightarrow \phi) \rightarrow \Psi(w))$	(King of France)$'$
$t \in \{x:\psi\}$	$\Longrightarrow$	$[t/x]\psi$	$\in$-elimination
$\{x:\phi\} = \{x:\psi\}$	$\Longrightarrow$	$\phi \longleftrightarrow \psi$	$=$-elimination

$[t/x]\psi$ is of course the result of substituting t for x in ϕ.

There may be other rewrite rules that i haven't thought of but that you will think of as you investigate this. That would be fun!

The King of France makes an appearance here because the output of the rule which i have given that name matches Russell's analysis of "The king of France is bald": "there is a king of france, all kings of france are identical and every king of France is bald".

The obvious motivation is that, for any model $\mathfrak{M}$, all these rules should preserve truth and falsity in $\mathfrak{M}$.

Clearly the King-of-France rule is good in this sense. In contrast the (King of France)$'$ rule might not preserve falsity. However the (King of France)$'$ rule is well-behaved for those $\mathfrak{M}$ that satisfy comprehension for ϕ whenever $\Psi(\{x:\phi\})$ appears in the input.

Naturally one wants to know which combinations of these rules give terminating, confluent rewrite systems with normal forms. We start with some easy observations

- The rewrite system (King of France) plus $=$-elimination is not confluent. Consider

  '$\{x : x \notin x\} = \{x : x \notin x\}$'. If you rewrite this using (King of France) then you get something that asserts the existence of the Russell class—in plain language, you get the `false`; OTOH if you rewrite it using $=$-elimination then you get the `true`. So *that* system of rules is not going to be confluent![1] ■

- The rule of $\in$-elimination by itself gives a non-terminating rewrite system, because of $\{x : x \in x\} \in \{x : x \in x\}$).

We need to think about whether or not the (King of France) rule by itself is confluent and terminating. There may be more than one occurrence of '$\{x:\phi\}$' in $\Psi(\{x:\phi\})$ and that means that (King of France) can be applied in more than

[1] There is also the point that '$\{x : x \notin x\} = \{x : x \notin x\}$' is a good thing to consider when discussing eager *vs* lazy evaluation. If you evaluate it top-down it appears to be true. If you evaluate it bottom up like the King of France it's obviously false.

one way. This calls for thought. However, the problem that i wish to push in your direction is a different one ...

As noted above, the rule of $\in$-elimination by itself does not form a terminating confluent rewrite system, beco's of $\{x : x \in x\} \in \{x : x \in x\}$,. Jamie Gabbay `https://researchportal.hw.ac.uk/en/persons/jamie-gabbay` claims that $\in$-elimination by itself *does* give a confluent terminating rewrite system if one restricts it to stratifiable formulæ. I haven't seen a proof and i won't be convinced until i do. It sounds plausible... the kind of cuteness one wishes to be true, so one doesn't look too closely. I suspect that ϕ being stratifiable is neither neccessary nor sufficient. After all, $\{x : x \in^2 x\} \in \{x : x \in^2 x\}$ admits the following terminating sequence of rewrites and biconditionals:

$$\begin{array}{ll} & \{x : x \in^2 x\} \in \{x : x \in^2 x\} \\ \Longrightarrow & \{x : x \in^2 x\} \in^2 \{x : x \in^2 x\} \\ \longleftrightarrow & (\exists y)(\{x : x \in^2 x\} \in y \wedge y \in \{x : x \in^2 x\}) \\ \Longrightarrow & (\exists y)(\{x : x \in^2 x\} \in y \wedge (\exists z)(y \in z \wedge z \in y)) \end{array}$$

But that exploits rewriting using biconditionals.

We'd better have a proof of this allegation of Gabbay's; either that or a counterexample. It shouldn't be hard.

" A message from jamie about confluence of the King-of-France rule, 20/x/20:

> It's easy to inject TST into what has been called the "purely negative" fragment of the untyped lambda-calculus, as follows:
>
> Map levels i to types α of the simply-typed lambda-calculus over a single base type o (thought of as a type of booleans):
>
> $[[0]] = o$
> $[[i+1]] = [[i]] \to o$
>
> Thus for instance $[[2]] = (i \to o) \to o$. This is very easily seen to be iterated powersets/powertypes over the base type o.
>
> Now identify TST constants with corresponding constants in the simply-typed lambda-calculus, which for simplicity we give the same name:
>
> $[[\top]] = \top : o$;
> $[[\vee]] = \vee : o \to o \to o$;
> $[[\neg]] = \neg : o \to o$;
> $[[\exists]] = \exists : (\alpha \to o) \to o$.
>
> Comprehension maps to lambda. If a is a variable of level i / of type $[[i]]$ then:
>
> $$[[a : \phi]] \; = \; \lambda a.[[\phi]] : [[i]] \to o$$
>
> Then the comprehension rewrite rule

$$t \in a : \phi - > \phi[a ::= t]$$

corresponds precisely to beta-reduction,

$$(\lambda a.s)t - > s[a ::= t]$$

and—here is the critical observation—confluence of the comprehension rewrite rule corresponds precisely to confluence of beta-reduction.

By this view, the 34 pages you refer to are a restatement of lambda-calculus confluence for this special case. Does that answer your question?"

Jamie

I haven't got a proof but i've made a start; perhaps you can do something with this stub.

The key notion is that of the **depth** of a formula.

A formula containing no set abstracts has depth 0;

a formula containing a subformula that is a set abstract $\{x : \phi\}$ where ϕ is of depth at least n is of depth at least $n + 1$.

However we want depth$(y \in \{x : \phi\})$ to be precisely depth(ϕ), as tho' to anticipate the $\in$-elimination.

Obviously the thought is that performing a reduction on a formula should output a formula of lower[2] depth. Let us see!

Evidently $D(\{x : \psi\} \in \{y : \phi\})$—the depth of '$\{x : \psi\} \in \{y : \phi\}$'—is $\max(D(\psi), D(\phi)) + 1$. But $D([\{x : \psi\}/y]\phi)$ might be as much as $D(\psi) + D(\phi)$, beco's there could be occurrences of 'y' buried under a deep rotting pile of curly brackets inside ϕ. So depth is not enough!

More to be done here

It would be nice to find a natural maximal class Γ such that $\in$-elimination for formulæ in Γ gives a confluent terminating rewrite system. At all events it would seem (and this is a conjecture of Jamie's) that Γ can be taken to contain all formulæ that are stratifiable-mod-n. I have a highly conjectural excluded-substructure candidate: the set of all formulæ that do not have any subformulæ of the form $x \in x$. What role does the difference between stratifiable and weakly stratifiable play in all this?

but of course if you have extensionality this simply becomes the $\exists\forall$ formula

$$(\exists y)((\forall x)(x \in y \longleftrightarrow \phi) \wedge \Psi(y))$$

and it's not equivalent to the $\forall\exists$ formula.

$$(\forall w)((\forall x)(x \in y \longleftrightarrow \phi) \rightarrow \Psi(w))$$

[2] 'smaller' might be a better word!

But of course if you have an axiom that $(\exists y)(\forall x)(x \in x \longleftrightarrow \phi)$ then the King of France *is* equivalent to the $\forall$ formula! So you can have the rewrite rule

$$\Psi(\{x : \phi\}) \implies (\forall w)((\forall x)(x \in y \longleftrightarrow \phi) \to \Psi(w))$$

If we have an axiom scheme of set existence for formulæ in Γ, then we can introduce terms for set abstracts for formulæ in Γ. There will be a translation back from the extended language into the original one. We can execute this translation by means of a rewrite rule

Rewrite $\{x : \phi\} \in y$ to
$(\exists w \in y)(\forall x)(\phi \longleftrightarrow x \in w) \wedge (\forall w)[(\forall x)(x \in w \longleftrightarrow \phi) \to w \in y]$.

(King of France stuff) and this works for all Γ. The interesting thing is that, for some Γ, we can do the back-translation by means simply of an $\in$-elimination rule. Well, we can't entirely, beco's of things like $\{x : \phi\} \in y$ where 'y' is a variable. But the rewrite corresponding to the $\in$-elimination rule:

$t \in \{y : \phi\}$ rewrites to $[t/y]\phi$

(Here i am writing 't' to remind myself that the thing to the left of the '$\in$' can be any old term of the language and doesn't have to be a variable) is terminating and confluent if Γ is the set of stratified formulæ. (This is a result claimed by Jamie)

So just what syntactic discipline *do* we need? No subformulæ $x \in x$?

Anyway, we get a termination result, but the formulæ that lack redexes might nevertheless still have set abstracts inside them, as above. So we want to add rules that enable us to get rid of all set abstracts. It's pretty clear what we will want those rules to be but it's not clear what syntactic disciplines we need to make the rewrite system terminating nor whether the system will be confluent.

Clearly we want two rules:

Rewrite	$\{x : \phi\} \in s$	to	$(\exists y \in s)(\forall x)(x \in y \longleftrightarrow \phi)$
Rewrite	$\{x : \phi\} = \{x : \psi\}$	to	$(\forall x)(\phi \longleftrightarrow \psi)$

The h-u-g-e problem now is that $\{x : \phi\} \in \{x : \psi\}$ matches the (input to the) first of the two new rules as well as (the input to) the old rule. This creates extra work for people trying to prove confluence.

Things to worry about.

• Existence of a normal abstract-free form
$\leq$ every reduction sequence terminates
$\leq$ all normal forms are equivalent.

• Then we have to think about weakly stratified set abstracts. And formulæ that are stratifiable-mod-n. Do we still get normalisation if we allow set abstracts that are stratifiable-mod-n?

- Is there a case for considering acyclic formulæ here?
- What about rewrite rules for equations containing set abstracts?

Yes! As we have seen, $D \in D$ reduces to $\neg(\exists x)(D \in x \wedge x \not\in^2 x)$ which is in normal form! (Is this anything to do with the fact that the nonexistence of D is a theorem of (constructive) first-order logic? That fact surely fits in somewhere.)

This last formula has a set abstract inside it. If we add a new rule that rewrites $\{x : \phi\} \in s$ to $(\exists y \in s)(\forall x)(x \in y \longleftrightarrow \phi)$ then $D \in D$ has a normal form (wrt this set of rules) that has no set abstracts in it!

H I A T U S

Jamie doesn't say anything about equations, but presumably we rewrite
$\{x : \phi\} = \{x : \psi\}$ to $(\forall x)(\phi \longleftrightarrow \psi)$
and we rewrite

$$\{x : \phi\} \in y$$

to

$$(\exists w \in y)(\forall x)(\phi \longleftrightarrow x \in w) \wedge (\forall w)[(\forall x)(x \in w \longleftrightarrow \phi) \rightarrow w \in y].$$

(King of France stuff).

The problem here is that
$\{x : \phi\} \in \{x : \psi\}$ is a special case of $\{x : \phi\} \in y$, and so can be reduced in two ways. This will make it very hard to prove confluence!

5 Acyclic Formulæ

A note on the expressivity of acyclic formulae

Nathan Bowler

April 25, 2014

Abstract

We show that every stratified formula ϕ in the language of set theory is equivalent to an acyclic formula ψ, that is, one for which the multigraph whose nodes are variables in ψ and whose edges are occurrences of atomic formulas in ψ is acyclic.

1 Introduction

NF is an axiomatisation of set theory introduced by Quine in which Russel's paradox is avoided by only requiring comprehension for formulae which satisfy a syntactic restriction called *stratifiability*. A *stratification* of a formula ϕ is a function d sending variables appearing in ϕ to integers such that for any atomic subformula $x = y$ of ϕ we have $d(x) = d(y)$ and for any atomic subformula $x \in y$ of ϕ we have $d(x) = d(y)-1$. A formula is *stratified* if there is some stratification of that formula. Thus, for example, the formula $x \notin x$ is not stratified. The axioms of NF are extensionality and comprehension for stratified formulae. NF was recently shown to be consistent (on the assumption that ZF is) by Holmes, three quarters of a century after it was first introduced TODOCITE.

In TODOCITE, Al-Johar, Holmes and Bowler investigated a more stringent syntactic restriction called acyclicity. A formula ϕ in the language of set theory is *acyclic* if the multigraph whose nodes are variables in ϕ and whose edges are occurrences of atomic formulas in ϕ is acyclic. In particular, no atomic subformula mentions the same variable twice or appears more than once. This notion is apparently implementation dependent: for example, suppose we wish to express $x \in y \leftrightarrow x \in z$. This is apparently acyclic, but if our primitive logical operations do not include $\leftrightarrow$, we may try to express it as $(x \in y \rightarrow x \in z) \wedge (x \in z \rightarrow x \in y)$, which is not acyclic. For the purposes of this note, we will take $\neg$ and $\wedge$ as primitive logical operations. We will also freely make use of $\vee$ and $\rightarrow$, which is not a problem because the usual ways of expressing these in terms of $\neg$ and $\wedge$ do not involve copying atomic formulae and so do not affect acyclicity.

What was shown in TODOCITE was that comprehension for acyclic formulae implies comprehension for stratified formulae, so that NF could alternatively be axiomatised with extensionality and comprehension for acyclic formulae. However, this leaves open the question of how much can be expressed

From Nathan

"I agree that this is dependent on the precise definition of subformula in play, so maybe it would be good to make that explicit. For example, Thomas' example doesn't fit either of the definitions given here `https://planetmath.org/Subformula`. On the first definition, $x \in x$ would still be weakly stratifiable, since it is a subformula of $(\exists y)x \in y$. On the second, it wouldn't be a subformula of any stratifiable formula. On either definition it seems that there are formulæ which are weakly stratifiable (even stratifiable) but not weakly acyclic, such as the formula $(\exists x)(\exists y)(\exists z)(\exists t)x \in y \wedge x \in z \wedge y \in t \wedge z \in t$."

Nathan Bowler's unpublished theorem is that every stratifiable formula of

Set Theory is equivalent to an acyclic formula. I still don't properly understand the proof, but at least i now know how it's supposed to work. The idea is to express your stratifiable formula as a boolean combination of [acyclic] formulæ each with only the one variable ('f') free. Specifically, every atomic subformula appearing in our input formula gives us one of these acyclic-formulæ-with-only-'f'-free. f is a function sending concrete Zermelo naturals[3] to iterated singletons of stuff.

I have some email correspondence with Nathan and Zuhair about this which i am gradually turning into LaTeXto show you if it tickles your fancy.

5.0.1 A contribution from Randall on April Fools' Day 2018

I define a selection of interesting acyclic predicates.

• "x has one element": there is an a such that for all y, $y \in x$ iff $y = a$
That is acyclic.

• "x has one or two elements": there are a and b such that, for all y, $y \in x$ iff $(y = a \vee y = b)$

• "x has two elements" $= x$ has one or two elements and $\neg(x$ has one element$)$

• "x is a (K) pair": (x has one element and all elements of x have one element) or (x has two elements and some element of x has one element and some element of x has two elements and the set of elements of elements of x has two elements)

• "the set of elements of elements of x has two elements" expands to

"there is z such that for all w, w is in z iff there is u such that w is in u and u is in x, and z has two elements"

• "x is the first projection of p": p is a pair and every element of p contains x

• "y is the second projection of p": p is a pair and the set of elements of p which contain y has one element.

• "the set of elements of p which contain y has one element" expands to
"There is z such that for all w, w is in z iff w is in p and y is in w, and z has one element."

So, the K pair can be managed.

I append Nathan Bowler's proof as an appendix, with his permission.
Zuhair writes: (LaTeXed and edited by tf)
The key gadget is Nathan's function f.
Each variable has two numerical indices.

One is the **identity index**; the other is the **stratification type**. The first is a natural number and the second is a negative integer (or otherwise use natural types but with reverse stratification where the type of 'y' in the subformula '$y \in x$' is one step HIGHER than the type of 'x'). We have countably

[3] A Zermelo Natural is an iterated singleton of $\emptyset$.

many variables and we fix an enumeration e of them; the identity index of a(n occurrence of a) variable in a formula is simply e of the variable of which our occurrence is an occurrence.

Nathan's function f sends iterated singletons of the empty set to iterated singletons of objects substituting the relevant variable. In other words we have the following:

$\langle a, b\rangle \in f$ if and only if there exists a variable v in the stratified formula such that :

$a = \iota^i(\emptyset)$ and $b = \iota^d(v)$, where i is the identity index of v and $-d$ is the stratification type of v.

Consider for example the formula '$x \in y \wedge z \in y$'. Suppose the identity index sends 'x to 1', 'y' to 2 and 'z' to 3. The stratification index sends 'x' to -2 , 'y' to -1 and 'z' to -2, so our f function will be:

$$f = \{\langle\{\emptyset\}, \{\{x\}\}\rangle \langle\{\{\emptyset\}\}, \{y\}\rangle \langle\{\{\{\emptyset\}\}\}, \{\{z\}\}\rangle\}$$

Now he uses this function f to code the variables in the formula in terms of f and iterated singletons of $\emptyset$. Once that's done then the repetition of occurrences of iterated singletons of $\emptyset$ is always acyclic because you can rename all bound variables in those at each occurrence.

We illustrate this process by giving the f function of the formula:

$$x \in y \wedge z \in y \wedge w \in x \wedge w \in z$$

Suppose the identity indices of 'x', 'y', 'z' and 'w' are 1, 2, 3 and 4 respectively; the stratification indices of 'x', 'y', 'z' and 'w' are evidently -2, -1, -2 and -3 respectively. So the f function would be

$$f = \{\langle\{\emptyset\}, \{\{x\}\}\rangle \langle\{\{\emptyset\}\}, \{y\}\rangle \langle\{\{\{\emptyset\}\}\}, \{\{z\}\}\rangle \langle\{\{\{\{\emptyset\}\}\}\}, \{\{\{w\}\}\}\rangle\}$$

Now all atomic subformulæ of the formula are membership atomic formulas, and those need to be interpreted in terms of f images of iterated singletons of $\emptyset$. But before we go to that I'll give first the bottom line of the argument for membership, but I'll leave the present example for a simpler one:

Take any variables 'r' and 's'.

Now examine the following formula:

$$\exists m \forall k[k = \iota^{i+1}(r) \vee k = \iota^i(s) \rightarrow m \in^{i+1} k] \qquad \text{(FORMU 1)}$$

FORMULA 1 is evidently EQUIVALENT TO "$r \in s$" [given the weak assumptions of Nathan]. So this is the bottom line of the membership atomic formulas. However, to achieve acyclicity we need to code that in terms of iterated singletons of $\emptyset$ instead of using variable symbols r, s in the formula! and here where the f function will trip in!

Now suppose that the identity index of 'r' is 1 and of 's' is 2, and the stratification types of them are -2 of 'r' and -1 of 's', then our function f would be

$$f = \{\langle\{\emptyset\}, \{\{r\}\}\rangle, \langle\{\{\emptyset\}\}, \{s\}\rangle\}$$

This f is an expression in the *object language*, not the metalanguage to which the identity index and the stratification index belong.

If we now rephrase FORMULA 1 in terms of f, we get:

$$\exists m \forall k[k = f(\{\emptyset\}) \vee k = f(\{\{\emptyset\}\}) \rightarrow m \in^2 k] \qquad \text{(FORMULA 2)}$$

Notice that FORMULA 1 is equivalent to "$r \in s$". Not *logically* equivalent, but equivalent modulo some very very weak set theory.

Notice, too, that FORMULA 2 has only one free variable, namely 'f'. Not two! This is key: if you can rewrite a formula in such a way that all its atomic subformulæ have been replaced by (possibly complex) *monadic* formulæ then the result will be acyclic. (So why can't we do this for *all* formulæ, not just stratified ones? Perhaps that will become clear later—be patient!)

All the other symbols are either constants or are bound, and ALL of those can be renamed altogether. So suppose you have the formula $r \in s$ occurring twice which is ACYCLIC but yet STRATIFIED (as you demanded), then still this iteration won't cause cyclicity because we rename ALL bound variables (and constants) in the above FORMULA 2 versions of them, so the only recurrent variable would be f and this will not be connected to the same variables in both occurrences (because all other variables in these formulas are renamed at each occurrence!) This is the general tactic that Nathan adopted to break acyclicity.

Now let's return to our formula:

$$x \in y \wedge z \in y \wedge w \in x \wedge w \in z.$$

Naturally we replace each of the four atomic formulæ with its FORMULA2 version, but notice that at each occurrence we rename all bound variables and all constants. This of course forestalls cycles. The point is that every variable (i.e. 'x' or 'y' or 'z' or 'w') even though it recurs in the formula yet it is represented by different variables at each occurrence. only f is recurrent in its representation but still that won't cause any harm because, at each occurrence of f, f is connected to a different set of variables, so no cyclicity will be raised!

Similarly but more obviously we can write the atomic formula $r = s$ in this way; this gives us a formula

$$f(t_r) = f(t_s)$$

where t_r, t_s are the terms $\iota^\rho(\emptyset)$, $\iota^\sigma(\emptyset)$ respectively where ρ and σ are the identity indices of 'r' and 's' respectively. Now $f(t_r)$ is the term $\iota^d(r)$, while $f(t_s)$ is the term $\iota^g(s)$, where d and g are the stratification indices of 'r' and 's' respectively. Since '$r = s$' is a subformula of a stratified formula then the two variables would have the same stratification index and the translation is a self-identity.

One minor issue: I think we need to add the stipulation that if the stratified formula ϕ has a total of n (of the metalanguage) many distinct variables, then the size of f must be exactly n; fortunately there is an acyclic formula that expresses that:

$$(\exists m_1, ..., m_n)(\forall p \in f)(p = m_1 \vee \cdots \vee p = m_n)$$

Now I think this can replace $Fn(f)$ big formula of Nathan, (in the final expression of the equalizing acyclic formula (the last of formulas in Nathan's proof)), so it both shortens the expression and also prevents having two distinct elements in the domain of f that are singleton^i iterations of empty objects (for the same i), The problem is that when we don't have extensionality we can have many of those in the domain of f, and it is better to prevent that!

I think that Nathan's formulation will break down if there is no extensionality over empty objects, the way how he wrote it makes it possible to have distinct same-degree iterated singletons of empty objects, and this can be disastrous. For example, let's take the atomic formula $x = y$. let the discriminative indexing be 1 for 'x' and 2 for 'y', now let the stratification types be -1 for each of 'x', 'y'. Now the way how Nathan wrote his formulation makes the following function f possible

$$f = \{\langle\{\emptyset\}, \{x\}\rangle, \langle\{\{\emptyset\}\}, \{y\}\rangle, \langle\{\emptyset*\}, \{z\}\rangle\}$$

where $\emptyset$ and $\emptyset*$ are distinct empty objects.

Now even if we have $x = y$, now if $z \neq y$, then we'll conclude that $f(\{\emptyset\}) \neq f(\{\{\emptyset\}\})$ the reason is because the formulation doesn't discriminate between $\emptyset$ and $\emptyset*$, so there will not exist an object k that is the second projection of all elements of f that has their first projections being either a set of an empty object or a set of a set of an empty object. so his method will break.

This will be resolved if we add the size criterion over f, as I've mentioned in the prior message above, this size criterion can simply replace $Fn(f)$ in his last formula, actually, the size criterion formula is much shorter than the formula $Fn(f)$ of Nathan's.

[Nathan on 9/ii/18]

Hi all,

sorry for taking so long to get involved with this conversation. I'm flattered that you're looking over this document again, and I think Zuhair's explanation does well at unpacking a lot of things that were hidden by my notation. I'm looking forward to seeing how Thomas explains and clarifies things, too. By the way, Zuhair, I have no objection to you keeping this on your website.

There are a couple of small issues that I'd like to comment on. The first is what happens if we don't assume extensionality. Although I was certainly assuming extensionality in my head when I came up with this argument, and extensionality-based intuitions guided my thoughts, it turns out that extensionality is completely unnecessary. In fact, all that is needed in the object theory

is a tiny bit of comprehension: the existence of an empty set and the existence, for any sets x and y, of a set whose elements are precisely x and y (I hope it is obvious that the relevant comprehension axioms are acyclic). Zuhair's counterexample falls foul of the following clause of my formula F_n:

$$(\exists y_1)(\forall q_1)[q_1 \in f \wedge \pi_1(q_1) = \iota(\emptyset)] \rightarrow \pi_2(q_1) = y_1$$

this is because each of the sets 0 and 0^* satisfies the formula '$x = \emptyset$', so there are 2 possibilities for q_1, namely $< \{0\}, \{x\} >$ and $< \{0*\}, \{z\} >$, and so there is no way to choose a suitable y_1.

On the other hand, I think that the various clauses of my F_n are needed to make the whole thing work, since we do want f to be a function. So I'm skeptical of the idea that we can replace F_n by a formula stating that f has at most n elements.

Best wishes,

Nathan

Thanks for the approval.

Just to mention one point. ALL the clauses of your formula F_n are indeed met when you replace $F_n(f)$ by the size formula I've suggested, in the FINAL formula. The reason is because the "identification string" of yours will enforce them by brute force.

On Fri, Feb 9, 2018 at 1:29 PM, Zuhair Abdul Ghafoor Al-Johar <zaljohar@yahoo.com> wrote:

My idea was that in the FINAL formula, you have the identification string

$$f(\iota(\emptyset)) = \iota^{-d_1}(x_1) \wedge \ldots \wedge f(\iota^n(\emptyset)) = \iota^{-d_n}(x_n)$$

Now this string will enforce having n-many elements of f, now size string that I proposed will block having more than n-many elements of f, so those two strings coupled together will enforce that f has EXACTLY n-many elements and only those that we've stipulated in the identification string, and I think that's all of what we want, this will shun any uncontrolled pair from occurrence in f, and delimit f to what we want exactly. The information present in the identification string poses no threats at all to the proof.

Of course my argument doesn't need any axiom of Extensioanlity.

Zuhair

Dear Zuhair,

I understand now. Yes, I think that does give a more compact way to say that f is a function of the desired kind.

Best wishes,

Nathan

Dear Zuhair,

yes, I agree, boolean union is also used. The trick of first reducing to prenex normal form isn't really necessary, I think, but it greatly simplifies the argument.

If we want to avoid it then we have to introduce a new variable for a coding function for every quantifier and so the argument gets a bit more convoluted. Essentially if there are already variables $x_1 \dots x_n$ in play, as coded by a function f, and we want to add a clause with an existentially quantified further variable x_{n+1} then we would do so with the formula $(\exists f')F_{n+1}(f')$ and $f \subseteq f'$ and [stuff for the rest of the clause]. Universal quantifiers can be dealt with similarly. But writing the argument formally would be messy.

My idea was that in the FINAL formula, you have the identification string $f(\iota(\emptyset)) = \iota^{-d_1}(x1) \wedge \ldots \wedge f(\iota^n(\emptyset)) = \iota^{-d_n}(xn)$

Now this string will enforce having n-many elements of f, now size string that I proposed will block having more than n-many elements of f, so those two strings coupled together will enforce that f has EXACTLY n-many elements and only those that we've stipulated in the identification string, and I think that's all of what we want, this will shun any uncontrolled pair from occurrence in f, and delimit f to what we want exactly. The information present in the identification string poses no threats at all to the proof. Of course my argument doesn't need any axiom of Extensionality.

For the moment, I should say that I'd remain skeptical of this multiple coding functions approach, the problem is that quantifiers can be so nested that things are not in successive order always, a variable x can occur free in $\phi(x, y, z, ..)$ which is a quantified formula, however x can at the same time occur external to ϕ and be linked with other variables not occurring in ϕ, now we should use say the coding function f to cover the matrix of ϕ which includes covering relations between x and variables quantified by ϕ, but the problem is that IF we need to use f' to code occurrences of x outside ϕ, and cover its relations to variables outside of ϕ, then the problem is if we say $f \subseteq f'$, then we establish a link between them and since both are linked to x, then this would be cyclic. However, I think it can be resolved, since I think what you wrote tells us that for each variable we'll be using ONE function to code it throughout the formula and so f' won't be connected to x, it will only be connected to variables not occurring in ϕ, all those occurring in ϕ shall be coded using the f function, including occurrences of them outside of ϕ, this will make us use mixed formulations like $f(\iota^j(\emptyset)) = f'(\iota^i(\emptyset))$. This way as you wrote f' need only to be connected to the NEW variables that f don't cover. This might work, I need to think of it..., one thing that I don't really grasp is why should we be using multiple coding functions? why not one coding function for the whole formula, I mean if a variable is coded by one function despite its occurrence within the quantifications cope of other variables or outside them, then we might as well use ONE coding function since it appears that quantifiers are irrelevant to this assignment?

I should admit that this multi-coding is confusing me. Suppose I have formula π being a subformula of ϕ, and π has two free variables in it x, y that are also occurring outside of π in ϕ, now since π has quantifiers then we need our f function occurring "after" these quantifiers, (I get that now), but also we need an $f*$ function being after the quantifiers of ϕ (outside of π), lets say that these

coding functions are coding disjoint sets of variables and that $f \subseteq f*$ is not mentioned. There will be also a problem lets say that x and y are connected to variables quantified by π, then through these variables, in the interpreting formula, f will be connected with $f*$ twice raising cyclicity! Now if one says that all variables coded by f are also coded by $f*$, and since those are different symbols, then we don't have that connectivity across f and $f*$, so there will be no cyclicity raising in the coding formulas. However, still we'll have a problem since we have more than one variable shared by π and ϕ, then f would be connected to $f*$ twice via x and y, this occurs through their identification strings! so this is cyclic also!

We need a way to make $f(\iota^i(\emptyset)) = f * (\iota^j(\emptyset))$ not causing connectivity between f and $f*$? and also the same for the $\in_d$ relation?

References

[1] `https://planetmath.org/Subformula`.

[2] Zuhair Al-Johar, M. Randall Holmes and Nathan Bowler: "The Axiom Scheme of Acyclic Comprehension" Notre Dame J. Formal Logic **55**, Number 1, (2014), pp 1–155. `http://projecteuclid.org/euclid.ndjfl/1390246432`

lambda calculus?

Stratification generalises to typing in λ-calculus. Is there a good notion of acyclicity in *lambda*-calculus? The S combinator is not acyclic.