(Towards a) Boolean Set Theory in a Three-valued Logic

Jozef Mikušinec

April 2022, unfinished

Summary

TODO this summary got outdated pretty fast. Fix when you have a clearer picture of what this is about. Dear reader, please skip to the introduction if you wish to read this very unfinished document.

This document is a work in progress where we'll try to define the syntax, semantics and a deductive system for Boolean Set Theory, a set theory in a three-valued first-order logic, with the empty set and the universal set, with the universal set closed under union, intersection, powerset, meetset (the set of all intersecting sets) and complement. The construction is carried out in ZFC.

The work in this document is inspired by my magister thesis [2] and works cited therein. Knowledge of the thesis will [/can?] help with understanding ETST, but is not assumed.

The reader is strictly advised not to waste her time on this pile of half-baked ideas and unfinished senten

TODO create an appendix "Reference" that has just the definitions and properties stated, no proofs, no intuition or whys.

Contents

0 Introduction

I'd like to do math just like I do code: in a programming language that computers can verify, and that rests on foundations that I (and, hopefully, other people) find natural and easy to use.

Unfortunately, it seems to me that current theorem proving languages like Coq or Idris suffer on the latter front. Instead, I'd like to have something familiar: types that are understood as collections of values (ie. sets), and (essentially) classical logic.

At the same time, I'd like to deal with the problem of self-referential negations in a way that avoids unnecessary restrictions as much as reasonably possible. The next section explains in detail what I mean.

0.0 Negation and self-reference

Self-reference is a natural part of mathematics. This is very explicit in programming languages, where it is common to define functions and data structures in terms of themselves. However, in math, freely mixing self-references and negations can result in contradictory definitions, like the Russell's set of sets not containing themselves, or even more simply, the set of those elements it does not contain.

Traditionally, this issue is dealt with by syntactically restricting what definitions are considered valid, in a way that guarantees that the valid definitions cannot be contradictory.[0] Under this approach, definitions like the above are simply considered erroneous. A disadvantage of this approach is that it necessarily also disallows some definitions that are intuitively meaningful.[1]

0. In particular, most set theories deal with self-referential negations by restricting which formulas may be used to specify new sets. For example, the most widely used set theory, ZFC, only allows one to create subsets of existing sets when specifying which elements belong to the new set by a logical formula. Positive set theories, like for instance $\mathrm{GPK}^{+}_{\infty}$ [0], allow comprehension for all positive (roughly meaning "negation-free") formulas.
1. Whether a particular definition is well-behaved (ie. classical, as defined later) is Turing-undecidable. Therefore, to disallow all badly behaved definitions, one must necessarily disallow some well-behaved ones as well.

Here we attempt a different approach: instead of restricting which definitions are valid, we give meaning to all of them by interpreting our sets as three-valued collections, where every element either defitely belongs to a set, definitely does not, or its membership in that set is *undetermined*. Correspondingly, our logic is three-valued as well – every formula is either true, false, or undetermined.

Under this approach, "sensible" definitions retain their classical, two-valued semantics. The additional state *undetermined* is used for elements whose membership in a set cannot be reasonably determined because of self-referential negations. Seemingly paradoxical definitions, like the Russell's set, avoid contradictions using this additional state; which may also be interpreted as "maybe true or false", or "unknown".

* * *

By moving to a three-valued logic, we depart to an extent from our initial aspiration of using classical logic. This departure enables placing no syntactic restrictions on the mathematician when it comes to defining her sets, and if she is careful, those sets stay classical – the strange state *undetermined* may be entirely avoided. Deciding whether we succeeded in creating a foundation of mathematics that feels "familiar and natural", we leave to the reader.

0.1 Outline

TODO

0.1.0 Reading order / dependencies

TODO sth. like: This diagram shows the dependencies between chapters [if applicable]. Or: If you are only interested in X, read the zeroth two chapters, then go to chapter Y and Z.

Eg. The chapters on well-founded collections stray into some issues I find interesting on their own, however, if you're here just for the set theory, you'll be ok [with just reading about the least fixed point semantics/with skipping the X and Y]

0.2 Related work

TL;DR: See my magister thesis [2].

TODO For further references and prior work, see [2]. TODO perhaps copy some of that here, perhaps do some more research to find out if there is anything else

out there that is relevant. [Note: Think there is something I should read cite? Let me know.]

0.3 Conventions and notation

This document uses three kinds of colored boxes to highlight certain its parts. Definitions are put in a grey box, like the following one.

The *universal set* is the set that contains all sets.

When introducing new concepts, examples are often useful. If an example appears in a green box, it is there just to ease understanding and may be safely skipped or forgotten. Examples introduced outside the green box may be referred to in latter parts of the document.

TODO an example.

The last kind of box contains related open problems, stylized as conjectures or questions. They are chosen subjectively by the author, and might be informal.

Open question X. Is there a sound and complete deductive system for LPC?

Conjecture Y. There are consistent theories of LPC that contain [TODO arithmetic/Turing machines and] their own truth predicates. TODO will LPC survive the structure overhaul?

This document also includes some warnings:

Warning. This document has not been published or peer reviewed. However, I'm in the process of formalizing all the math in Lean 4 – you can find the source code in the GitHub repository linked in section 0.4.

TODO sth like everything in here is proven in lean 4. Sometimes dues to differences between ZFC and Lean, things are formalized differently to allow using patterns that are more natural in the respective setting. Such differences are noted in this document, (unless trivial or forgotten :D).

Warning. This document is a work in progress.

By convention, our intervals are closed at the lower bound and open at the upper bound unless otherwise stated. Our indexing is proper – it starts at zero.[2]

We'll use $a \between b$ to express that the sets a and b *meet* (have a common element).

2. See "Why numbering should start at zero" by E. W. Dijkstra. Available at `https://www.cs.utexas.edu/users/EWD/transcriptions/EWD08xx/EWD831.html`

0.4 Source and contributing

The source code for this document can be found at:

`https://github.com/mik-jozef/etst`

Feedback and contributions are welcome.

TODO mention the formalization in Lean 4.

1 Well-founded collections

Fix a set D, for example the natural numbers, and some (monotonic) operations on subsets of D, for example addition, substraction (and constants representing zero and one), defined as follows:

$$A + B = \{\, x \mid x = a + b \text{ for some } a, b \in A, B \,\}$$
$$A - B = \{\, x \mid x = a - b \text{ for some } a, b \in A, B \,\}$$

One may then define a system of recursive equations using the above operations together with binary unions and intersections, and arbitrary unions and intersections, for example[3]:

$$\mathit{Even} = \{\, 0 \,\} \cup \mathit{Even} + \{\, 2 \,\}$$
$$\mathit{Div6} = \bigcup_{n \in \mathit{Even}} n + n + n$$

These equations can be viewed as defining the values of the variables to be the equations' least fixed point.[4]

Well-founded collections (WFC) are a generalization of the standard least fixpoint semantics that in addition to the mentioned operations is capable of handling set complements, including those that are contradictory in a classical, two-valued sense.[5]

$$\mathit{Nat} = \sim\{\}$$
$$\mathit{Even} = \mathit{Nat} \cap \sim(\mathit{Even} + \{\, 1 \,\})$$
$$\mathit{Undetermined} = \sim \mathit{Undetermined}$$

In this and the following chapter, we will formally define well-founded collections.

3. $\{2\}$ is a syntactic sugar for $\{1\} + \{1\}$
4. Assuming the equations are viewed as the corresponding (monotonic) function on $P(D)^2$.
5. Well-founded collections are also a generalization of WFRQ, the subject of my magister thesis [2], the difference being that WFRQ uses records (aka. structs or objects) as D, while WFC is generalized to allow any set D with any monotonic operations.

1.0 Three-valued collections

A recurring theme in this document is that of a three-valued collection. Every element either definitely belongs, or does not belong to a two-valued collection. A three-valued collection offers one extra possibility – an element's membership in the collection may be undetermined.

An element that definitely belongs to a collection C is said to be a *definite member* (or just a *member*) of C. An element that definitely does not is said to be a (definite) *nonmember* of C. An element whose membership in C is undetermined is said to *border* C, or be a *bordering element* of C.

(Definite) members and undetermined elements are called the *possible members*. Similarly, nonmembers and undetermined elements are called the *possible nonmembers*.

A three-valued collection is *classical* (or *fully determined*) iff its membership relation is determined for every element, or in other words, iff all its possible members are its definite members.

1.1 Formalizing three-valued collections

Two-valued collections of elements of some set S are just the subsets of S. Three-valued collections can be formalized in two equivalent ways:

Definition 0
A 3-valued collection of elements of S is a pair (L, U), where $L \subseteq U \subseteq S$.

Definition 1
A 3-valued collection of elements of S is a function from S to $\{0, \bot, 1\}$.

These two definitions are equivalent, because every element of S can either

0. be a member of both L and U (be a definite member),
1. be a member of U, but not L (border the collection),
2. be a member of neither U nor L (be a definite nonmember).

The third option – that an element of S is a member of L, but not U – is probihited by the requirement that L is a subset of U. The sets L and U can be thought of as forming a lower and upper bound on the membership of elements of D in the three-valued collection.

The latter definition better illustrates the three-valuedness, but by default, we'll keep using the former, and we'll call the latter the *characteristic function* of a set. To avoid confusion with two-valued sets, we will call the three-valued sets "trisets".

$Set3(D)$ is the set of all trisets of elements of D.

Here are some examples of trisets of natural numbers:

$(\emptyset, \emptyset)$ contains no numbers, and $(\mathbb{N}, \mathbb{N})$ contains all of them. Neither of these trisets borders any element.

$(\emptyset, \mathbb{N})$ definitely contains no numbers, but possibly contains all of them. Similarly, no number is its definite nonmember, but every number is its possible nonmember. It borders every number.

$(\{0, 1\}, \{0, 1, 2\})$ definitely (and possibly) contains numbers 0 and 1, but only possibly contains the number 2. Therefore it borders the number 2.

1.2 Orderings on trisets

We'll define two orderings on three-valued sets: the standard ordering, and the approximation ordering. These will be later used to show that our construction of the semantics of WFC converges to a fixed point.

A triset A is smaller than B in the standard ordering if it contains fewer elements (both definite and possible), and is smaller in the approximation ordering if it agrees with B on the membership of B's determined elements (but may have fewer of them than B).

A triset becomes larger in the standard ordering iff some elements are added to it, while it becomes larger in the approximation ordering if some of its undetermined elements become determined.

In the standard ordering, $(L_0, U_0) \leq (L_1, U_1)$ iff $L_0 \subseteq L_1$ and $U_0 \subseteq U_1$.

In the approximation ordering, $(L_0, U_0) \sqsubseteq (L_1, U_1)$ iff $L_0 \subseteq L_1$ and $U_1 \subseteq U_0$.

Proposition 1.2.0 (Least trisets)**.**
The least triset with respect to the standard (resp. approximation) ordering is the empty triset $(\emptyset, \emptyset)$ (resp. the wholly undetermined triset $(\emptyset, D)$).

Proposition 1.2.1 (Least upper bounds of triset chains)**.**
Let $T \subseteq Set3(D)$ be any chain with respect to the standard (resp. approximation) ordering. Then its least upper bound equals

$$\left(\bigcup_{(l,u)\in T} l, \bigcup_{(l,u)\in T} u\right), \text{ resp. } \left(\bigcup_{(l,u)\in T} l, \bigcap_{(l,u)\in T} u\right)$$

.

Corollary 1.2.2. The orderings on triset chains are chain-complete (that is, every chain has a least upper bound).

1.3 Syntax of WFC

1.3.0 Signatures

A signature specifies a set of operations and their arities. The syntax of well-founded collections is defined relatively to a signature.

A signature is a pair (Op, arity), where
- Op is a set of operations (just an arbitrary set), and
- $\text{arity}\colon\ Op \to \mathbb{N}$ is a function specifying the arities of elements of Op.

1.3.1 Expressions

The syntax of WFC may be denoted using two possible notations – math (eg. $A \cup B$) or code notation (eg. `A | B`) – we'll use both interchangeably.

An expression of WFC is defined recursively as follows:

$E = x$	`E = x`	Variables
$\mid f(...)$	`| f(...)`	Custom operations
$\mid E \cup E$	`| E | E`	Binary union
$\mid E \cap E$	`| E & E`	Binary intersection
$\mid \sim E$	`| ~E`	Complement
$\mid \bigcup_{x \in E} E$	`| Un x: E .. E`	Arbitrary union
$\mid \bigcap_{x \in E} E$	`| Ir x: E .. E`	Arbitrary intersection

$Variable = \mathbb{N}$ is the set of all variables.

Let sig be a signature, Var a set of variables, and n a natural number.

$Expr_{sig,Var,n}$ is the set of all expressions of depth at most n whose free variables belong to Var.

$Expr_{sig,Var}$ is the set of all expressions whose free variables belong to Var.

$Expr_{sig} = Expr_{sig,Variable}$ is the set of all expressions.

1.3.2 Defition lists

A definition is a named expression. Definitions are grouped into definition lists. A definition list allows its definitions to refer to each other.

In math notation, definition lists will be denoted as shown at the start of the chapter (where we called a definition list a "system of recursive equations"). In code notation, definitions will be preceded with the keyword "let":

```
let Even = { 0 } | Even + { 2 };
let Div6 = Un n: Even .. n + n + n;
```

Let *sig* be a signature and *Var* a set of variables.

A *family of expressions* is a function $f\colon\ Var \to Expr_{sig,Var}$.

A *definition list* is a family of expressions that is a union of (possibly infinitely many) finite families of expressions.

$DefList_{sig,Var}$ is the set of all definition lists whose domain is *Var*.

$DefList_{sig} = DefList_{sig,Variable}$ is the set of all definition lists.

We require definition lists to be unions of finite families of expressions to dissalow "definition lists" whose definitions run on forever, like these:

```
let Def0 = Def1;
let Def1 = Def2;
let Def2 = Def3;
let Def3 = ...
```

Such infinitely long definitions are problematic because, if allowed, any (tri)set of natural numbers would suddenly become definable, since we could just enumerate all its elements:

```
let Def0 = Def1 | { 0 };
let Def1 = Def2 | { 2 };
let Def2 = Def3 | { 7 };
let Def3 = ...
```

Note. In Lean, a definition list is simply required to be finite. This does not affect the expressive power of definition lists, but makes them easier to work with (in Lean).

2 Semantics of WFC

2.0 Structures

For our purposes, we will define a structure in a nonstandard way – instead of on elements of the domain, our operations will work on sets of them, and are required to be monotonic.

A structure st is a triple (D, sig, I), where
- D is an arbitrary set, called the domain of st,
- $sig = (Op, \text{arity})$ a signature, and
- $I\colon (op\colon\ Op) \to P(D)^{\text{arity}(op)} \to P(D)$ an interpretation function that to each $op \in Op$ assigns a monotonic function of arity arity(op) on sets of elements of D.

2.1 Valuations

A valuation is an assignment of trisets to variables. Valuations let us interpret expressions as representing a concrete triset.

A valuation is a function $v\colon\ Var \to Set3(D)$ for some $Var \subseteq Variable$ and D.

$Valuation_{Var,D}$ is the set of all valuations from Var to $Set3(D)$

Where the domain can be assumed from context, $v_e = x \mapsto (\emptyset, \emptyset)$ is the empty valuation, and $v_u = x \mapsto (\emptyset, D)$ is the undetermined valuation.

The standard and approximation orderings can be extended to valuations as well.

For either ordering, valuations are ordered pointwise.

That is, a valuation $v_0\colon\ Valuation_{Var,D}$ is less than $v_1\colon\ Valuation_{Var,D}$ iff for every $x \in Var$, $v_0(x)$ is less than $v_1(x)$.

Proposition 2.1.0 (Least valuations)**.**
The least element of the standard ordering is $x \mapsto (\emptyset, \emptyset)$, and the least element of the approximation ordering is $x \mapsto (\emptyset, D)$.

Proposition 2.1.1 (Least upper bounds of valuation chains)**.**
Let T be any chain of valuations with respect to the standard (resp. approximation) ordering. Then its least upper bound exists, and its value at x equals the least upper bound of the values of valuations in T at x.

Corollary 2.1.2. The orderings on valuations are chain-complete.

2.2 An informal overview

Our goal in this chapter is to, given a particular structure and definition list dl, associate a three-valued set to each definition of dl. More precisely, we need to find a valuation that assigns to every definition of dl a triset that "agrees" with the expression that forms that definition.

Our obstacle is that definitions are typically circular. The solution is to build the valuation incrementally, in stages. Here we give an informal overview of the formal machinery of this chapter. It is in principle based on the one used in [1].

2.2.0 The standard solution

When dealing with circular definitions, the usual solution would be the following:

Zeroth, we would define an interpretation function that takes a valuation and an expression, and returns the value of that expression when the value of its free variables is determined by the valuation.

Starting with the empty valuation v_e, we would then iteratively build new valuations using the interpretation of the definitions of dl under the valuation from the previous step[6]:

```
let Nat = { 0 } | Nat + { 1 };
```

$$\{\} \to \{0\} \to \{0, 1\} \to \{0, 1, 2\} \to ... \to \mathbb{N}$$

Formally, we would define an operator O_{dl} on valuations that takes the nth such valuation to the $n + 1$th one, and define the semantics of dl as O_{dl}'s least fixed point – the least valuation v such that $v = O_{dl}(v)$.

$$v = O_{dl}(v) = O_{dl}(O_{dl}(v)) = ...O_{dl}(O_{dl}(O_{dl}(v_e)))...$$

6. Perhaps transfinitely many times.

2.2.1 The issue

The least fixed point may not always exist for an arbitrary operator. One way to ensure its existence is by exhibiting a certain[7] ordering with respect to which the operator is monotonic. Intuitively, one may view such an ordering as preventing cycling between distinct valuations.[8]

The standard approach works for definitions not containing complements or arbitrary intersections, because all the other operations are monotonic with respect to the standard ordering (where a triset is larger iff it contains more elements).

However, it has a serious flaw: complement and the arbitrary intersection are not monotonic. The problem is not just with the choice of the ordering – definitions using them can fail to have a (least) fixpoint.

```
let Bad = ~Bad;
```

$$\{\} \to \mathbb{N} \to \{\} \to \mathbb{N} \to ... \rightsquigarrow !$$

2.2.2 Our solution

To fix this flaw, we'll introduce another valuation, called "background" (we'll call the original valuation "context"). The interpretation gets both valuations as parameters, and is modified to use the background valuation when interpreting complements and (domains of) arbitrary intersections (and their subexpressions), and uses the context for everything else.

Starting with the undetermined valuation as the background and the empty valuation as context, our iterative construction now proceeds in two dimensions:

$$\begin{array}{c} b_0, c_{0,0} \to b_0, c_{0,1} \to b_0, c_{0,2} \to ... \to b_0, b_1 \\ \downarrow \\ b_1, c_{1,0} \to b_1, c_{1,1} \to b_1, c_{1,2} \to ... \to b_1, b_2 \\ \vdots \\ b_n, c_{n,0} \to b_n, c_{n,1} \to b_n, c_{n,2} \to ... \to b_n, b_n \end{array}$$

In the inner (horizontal) sequences, context is updated as before, while the background stays constant. Because the background is constant, the interpretation of complements is constant as well, so the horizontal sequences are monotonic with respect to the standard ordering. When a fixed context is reached, it is used as the next background in the next step of the outer (vertical) sequence.

7. Not any ordering will do. We need the supremums of directed subsets to exist.
8. Infinite ascending chains are not a problem, since in that case, we just take their supremum and continue the process. This is why we need transfinite stages, and the supremums to exist.

Just like the standard iterative construction, the above two-dimensional one can also be expressed in terms of an operator. (More precisely, one outer operator, and one family of inner operators, one for each possible background.) To prove the existence of the least fixed point of the outer operator, we'll show it is monotonic with respect to the approximation ordering (where a triset becomes larger iff some of its undetermined elements become determined). The fixed point will serve as the definition of the semantics of WFC.

* * *

Besides defining the operators themselves, we'll also define the results of the iterative construction of the fixpoint at ordinal stages. That is, for an operator $f\colon T \to T$, we'll also define $f_{n\colon Ord}\colon T$ in terms of its previous stages, where f_n equals the result of applying the operator f n times.

Such a construction is also formally valid on its own (by the transfinite recursion theorem [4]), and is more practical, too. We'll, however, define the operators as well, to show that the iterative construction agrees with the least fixed point of a naturally defined operator. Equivalence between these two approaches is proven.

2.2.3 An algorithmic view

If one is willing to entertain the idea of an algorithm that terminates after a (potentially) transfinite number of steps, she can view a single application of the operator O_{dl} as corresponding to a single iteration of the loop below on the left, and finding O's least fixpoint as executing that loop. Similarly, the (outer and inner) operators can be viewed as the nested loops on the right.

```
v = the empty valuation;

while (v has changed) {
  v = (x) => I(v)(dl(x));
}
```

```
b = the undetermined valuation;
c = the empty valuation;

while (b has changed) {
  while (c has changed) {
    c = (x) => I(b, c)(dl(x));
  }

  b = c;
  c = the empty valuation;
}
```

2.3 Interpretation of expressions

Interpretation is a function that takes two valuations and an expression, and returns a three-valued set represented by that expression. We'll define it recursively

on the structure of the expression. The valuations b and c stand for "background" and "context".

Let $st = (D, sig, I_s)$ be a structure, and Var a set of variables.

$I_{st}\colon (b, c\colon \mathit{Valuation}_{Var,D}) \to \mathit{Expr}_{sig,Var} \to \mathit{Set3}(D)$

The interpretation of variables is given by the context valuation, and the interpretation of custom operations by the structures' interpretation function I_s, which is applied separately to the definitive and possible members.

$$I_{st}(b,c)(\texttt{var}) = c(\texttt{var})$$
$$I_{st}(b,c)(\texttt{f([e0], ..., [eN])}) = (I_s(\texttt{f})(I_{st}(b,c)(\texttt{e0})[0], ..., I_{st}(b,c)(\texttt{eN})[0]),\\ I_s(\texttt{f})(I_{st}(b,c)(\texttt{e0})[1], ..., I_{st}(b,c)(\texttt{eN})[1]))$$

The definite (resp. possible) members of triset union are the union of the definite (resp. possible) members of the arguments. Same for intersection.

$$I_{st}(b,c)(\texttt{[e0] | [e1]}) = (I_{st}(b,c)(\texttt{e0})[0] \cup I_{st}(b,c)(\texttt{e1})[0],\\ I_{st}(b,c)(\texttt{e0})[1] \cup I_{st}(b,c)(\texttt{e1})[1])$$

$$I_{st}(b,c)(\texttt{[e0] \& [e1]}) = (I_{st}(b,c)(\texttt{e0})[0] \cap I_{st}(b,c)(\texttt{e1})[0],\\ I_{st}(b,c)(\texttt{e0})[1] \cap I_{st}(b,c)(\texttt{e1})[1])$$

The triset complement swaps definite members for definite nonmembers, and vice versa. Undetermined elements stay undetermined. Notice that the complement is defined only in terms of the background valuation.

$$I_{st}(b,c)(\texttt{\~{}[e]}) = (D \setminus I_{st}(b,b)(\texttt{e})[1],\\ D \setminus I_{st}(b,b)(\texttt{e})[0])$$

To define the interpretation of arbitrary unions and intersections, we zeroth need a supporting definition:

The function *updateValuation* accepts a valuation, a variable x and an element e of D, and returns a valuation that under the variable x contains the triset containing just e, and leaves other variables unchanged.

$\mathit{updateValuation}\colon \mathit{Valuation}_{Var,D} \times (x\colon \mathit{Variable}) \times D \to \mathit{Valuation}_{Var \cup \{x\},D}$

$$\mathit{updateValuation}(v, x, r)) \stackrel{\text{def}}{=} v[x \mapsto r] = x_n \mapsto \begin{cases} (\{r\}, \{r\}) & x_n = x \\ f(x_n) & \text{else} \end{cases}$$

Note that in the case of arbitrary intersection, the domain is defined only in terms of the background, like in the case of complement.

$$I_{st}(b,c)(\texttt{Un x: [e0] .. [e1]}) =$$
$$\Big(\bigcup_{r \in I_{st}(b,c)(\texttt{e0})[0]} I_{st}(b[x{\mapsto}r], c[x{\mapsto}r])(\texttt{e1})[0],$$
$$\bigcup_{r \in I_{st}(b,c)(\texttt{e0})[1]} I_{st}(b[x{\mapsto}r], c[x{\mapsto}r])(\texttt{e1})[1] \Big)$$

$$I_{st}(b,c)(\texttt{Ir x: [e0] .. [e1]}) =$$
$$\Big(\bigcap_{r \in I_{st}(b,b)(\texttt{e0})[1]} I_{st}(b[x{\mapsto}r], c[x{\mapsto}r])(\texttt{e1})[0],$$
$$\bigcap_{r \in I_{st}(b,b)(\texttt{e0})[0]} I_{st}(b[x{\mapsto}r], c[x{\mapsto}r])(\texttt{e1})[1] \Big)$$

When the background is not provided, it is assumed to equal the context, that is, $I_{st}(c)$ is defined as $I_{st}(c, c)$.

Proposition 2.3.0 (Interpretation is a triset)**.**
The interpretation of any expression e under any valuations b, c is a triset (L, U), that is, $L \subseteq U$.

Lemma 2.3.1 (Standard monotonicity of interpretation)**.**
For any valuation b: $\mathit{Valuation}_{Var,D}$ and expression $e \in \mathit{Expr}_{sig,Var}$, the function $c \mapsto I_{st}(b, c)(e)$ is monotonic with respect to the standard ordering.

(Informally, if the context gets more elements, the interpretation gets more elements.)

Proof. By induction over the structure of the expression, and an extensive number of cases. We'll explicitly prove only the inductive step for the quantifiers, the rest is trivial. (The complement is particularly easy, since it is constant with respect to c.)

Let c_0 and c_1 be valuations such that $c_0 \leq c_1$. Then for any x and r,

$$c_0[x{\mapsto}r] \leq c_1[x{\mapsto}r]\,.$$

Let $e = \bigcup_{x \in e_0} e_1 \in \mathit{Expr}_{sig,Var}$. Then $e_0 \in \mathit{Expr}_{sig,Var}$ and $e_1 \in \mathit{Expr}_{sig,Var \cup \{x\}}$.

By the induction hypothesis, we have that

$$I_{st}(b', c'_0)(e_0) \leq I_{st}(b', c'_1)(e_0)$$
$$I_{st}(b', c'_0)(e_1) \leq I_{st}(b', c'_1)(e_1)$$

for any c'_0, c'_1 and b' such that $c'_0 \leq c'_1$, and therefore also for $b' = b[x \mapsto r]$, $c'_0 = c_0[x \mapsto r]$ and $c'_1 = c_1[x \mapsto r]$, for any $r \in D$.

To recap:

$$s_0 = I_{st}(b, c_0)(e_0) \leq s_1 = I_{st}(b, c_1)(e_0)$$
$$t_0 = I_{st}(b[x \mapsto r], c_0[x \mapsto r])(e_1) \leq t_1 = I_{st}(b[x \mapsto r], c_1[x \mapsto r])(e_1)\,.$$

By the definition of the standard ordering, $s_0[j] \subseteq s_1[j]$ and $t_0[j] \subseteq t_1[j]$. By the definition of I,

$$I_{st}(b, c_i)(e)[j] = \bigcup_{r \in s_i[j]} t_i[j]\,.$$

Finally,

$$\bigcup_{r \in s_0[j]} t_0[j] \subseteq \bigcup_{r \in s_1[j]} t_1[j]\,.$$

therefore

$$I_{st}(b, c_0)(e) \leq I_{st}(b, c_1)(e)\,.$$

The case of the universal quantifier is a little simpler: the index set of the intersections in its definition is constant with respect to c, and the "body" of the intersections is proven monotonic as above, so the whole expression is monotonic. □

Lemma 2.3.2 (Approximation monotonicity of interpretation)**.**
Let $e \in \mathit{Expr}_{sig, Var}$ be an expression, and $b_0, b_1, c_0, c_1 \colon \mathit{Valuation}_{Var,D}$ valuations such that $b_0 \sqsubseteq b_1$ and $c_0 \sqsubseteq c_1$. Then $I(b_0, c_0)(e) \sqsubseteq I(b_1, c_1)(e)$.

(Informally, if the either background or context become more determined, the interpretation becomes more determined.)

Proof. Similar in principle to the proof of standard monotonicity. □

We can extend the interpretation of expressions to entire definition lists, pointwise.

$I_{st} \colon (b, c \colon \mathit{Valuation}_{Var,D}) \to \mathit{DefList}_{sig, Var} \to \mathit{Valuation}_{Var,D}$
$I_{st}(b, c)(dl) = x \mapsto I_{st}(b, c)(dl(x))$

Proposition 2.3.3. This extended interpretation is monotonic with respect to the standard and approximation orderings in the same way as the interpretation of expressions. (See lemmas 2.3.1 and 2.3.2.)

A valuation v is *stable* with respect to a definition list dl, or a *model* of dl, iff $I(v)(dl) = v$.

2.3.0 A theorem about the interpretation

This theorem illustrates a nice property of the interpretation, although it isn't used later.

A (proper) classical completion of a triset s is a classical triset (strictly) greater than s in the approximation ordering.

Given a nonempty set of trisets S, the *mix* of S is the triset such that:
- a value belongs to mix(S) iff it belongs to every element of S,
- a value doesn't belong to mix(S) iff it belongs to no element of S
- a value borders mix(S) otherwise.

Proposition 2.3.4. The mix of S is the greatest triset less than any element of S in the approximation ordering.

Theorem 2.3.5 (Interpretation and mixing)**.**
The interpretation of any expression e under any valuation v is TODO less wrt the approximation ordering than the mix of the interpretations of e under the classical completions of v. In symbols:

$$I_{st}(v)(e) = \mathrm{mix}_{v' \in V}(I_{st}(v')(e))\,,$$

where V is the set of classical completions of v.

TODO: THIS IS FALSE!!! And Trivially so! Take just "p or not p". What was I smoking when I wrote this?!

Proof. TODO proof. □

2.3.1 The conditional operator

In practice, a conditional operator often makes for convenient definitions. Luckily for us, an operator `C ? A : B` that would take a triset C and return the triset A when C is nonempty, else would return the triset B, is expressible using the other operations as follows:

```
(Ex _: C .. B) | (Ex _: ~C .. B).
```

2.4 The operators and the iterative construction

Now we'll define a family of operators C_b that, given a background b, accept a context valuation and return the next context, and the operator B that accepts a background valuation and returns the next background,

Let dl: $\mathit{DefList}_{sig,Var}$ a definition list, and b: $\mathit{Valuation}_{Var,D}$ a valuation.

$C_{st,dl,b}$: $\mathit{Valuation}_{Var,D} \to \mathit{Valuation}_{Var,D}$
$C_{st,dl,b}(c) = I_{st}(b,c)(dl)$

$B_{st,dl}$: $\mathit{Valuation}_{Var,D} \to \mathit{Valuation}_{Var,D}$
$B_{st,dl}(b) = \mathrm{lfp}(C_{st,dl,b})$

...together with the corresponding iterative constructions.

TODO sth like: note that the limit stages are least upper bounds of the previous stages in the respective orderings

Let n be any ordinal, and l be any limit ordinal.

$C_{st,dl,b,n}$: $\mathit{Valuation}_{Var,D}$

$$
\begin{aligned}
C_{st,dl,b,0} &= v_e \\
C_{st,dl,b,n+1} &= I_{st}(b, C_{st,dl,b,n})(dl) \\
C_{st,dl,b,l} &= x \mapsto (\bigcup_{i\in l} C_{st,dl,b,i}(x)[0],\ \bigcup_{i\in l} C_{st,dl,b,i}(x)[1])
\end{aligned}
$$

$B_{st,dl,n}$: $\mathit{Valuation}_{Var,D}$

$$
\begin{aligned}
B_{st,dl,0} &= v_u \\
B_{st,dl,n+1} &= C_{st,dl,b,|2^D|}, \text{ where } b = B_{st,dl,n} \\
B_{st,dl,l} &= x \mapsto (\bigcup_{i\in l} B_{st,dl,i}(x)[0],\ \bigcap_{i\in l} B_{st,dl,i}(x)[1])
\end{aligned}
$$

. Recall that v_e and v_u are the empty and the undetermined valuation, respectively.

The rest of the section is filled with proofs of the equivalence of these, which the reader so inclined may freely skip.

Lemma 2.4.0 (Monotonicity of C_b)**.**
The operator $C_{st,dl,b}$ is monotonic with respect to the standard ordering,

Proof. A direct consequence of the lemma 2.3.3. □

Lemma 2.4.1 (Well-ordered chains of valuations are small)**.**
Let *Var* be a *finite* set of variables, and $V \subseteq \mathit{Valuation}_{\mathit{Var},D}$ a well-ordered chain of valuations with respect to either ordering. Then the cardinality of V equals that of D up to a finite difference.

Proof. A proof is in the appendix A. □

Lemma 2.4.2 (Correctness of the construction of least fixed points)**.**
Let T be a set with a chain-complete partial order $\leq$ such that all well-founded chains of T have a cardinality at most k, and $O\colon T \to T$ a monotonic operator on T.

Let $O_{n\colon \mathit{Ord}}\colon T$ be an element of T defined like this:

$$\begin{aligned} O_0 &= \text{ the least element of } T \\ O_{n+1} &= O(O_n) \\ O_l &= \text{ the least upper bound of } O_{n<l} \end{aligned}$$

Then $O_{|2^k|}$ is the least fixed point of O.

Proof. A proof of this theorem is in the appendix A. It is a variation the Bourbaki–Witt theorem and other similar theorems. □

Lemma 2.4.3 (Correctness of construction of C_b's least fixed point)**.**
Let

- $st = (D, \mathit{sig}, I)$ be a structure,
- *Var* a set of variables,
- b a $\mathit{Valuation}_{\mathit{Var},D}$, and
- dl a $\mathit{DefList}_{\mathit{sig},\mathit{Var}}$.

Then the least fixed point of the operator $C_{st,dl,b}$ equals $C_{st,dl,b,|2^{D\times\mathbb{N}}|}$.

Proof. We'll start with the case that *Var* is finite.

$T = \mathit{Valuation}_{\mathit{Var},D}$ is a set with a chain-complete partial order $\leq$ (the standard ordering, lemma 2.1.2) such that all well-founded chains of T have a cardinality at most $D \times \mathbb{N}$ (lemma 2.4.1), and $C_{st,dl,b}$ is a monotonic operator on T.

The iterative construction $C_{st,dl,b,n}$ agrees with the one prescribed by lemma 2.4.2 (lemmas 1.2.0, 2.1.1). By that lemma,

$$\mathrm{lfp}(C_{st,dl,b}) = C_{st,dl,b,|2^{D\times\mathbb{N}}|}\,.$$

It remains to prove the same when *Var* is infinite.

By definition, any infinite definition list dl is a union of finite definition lists fdl. The iterative construction of dl is the "union"[9]of the iterative constructions of

elements of *fdl*. Therefore, the least ordinal at which *dl*'s fixpoint appears in *dl*'s iterative construction equals the supremum of the least such ordinals of elements of *fdl*. All such ordinals are at most $|2^{D+\mathbb{N}}|$.

□

Now we can show that the operator B is monotonic with respect to the approximation ordering, and that its corresponding iterative construction produces B's least fixed point.

Lemma 2.4.4 (Monotonicity of B)**.**
The operator $B_{st,dl}$ is monotonic with respect to the approximation ordering.

Proof. Let $b_0, b_1 :$ *Valuation* $_{Var,D}$ such that $b_0 \sqsubseteq b_1$. We'll show that for any n,

$$C_{st,dl,b_0,n} \sqsubseteq C_{st,dl,b_1,n}\,,$$

from which it follows that

$$\mathrm{lfp}(C_{st,dl,b_0}) \sqsubseteq \mathrm{lfp}(C_{st,dl,b_1})\,.$$

Suppose $C_{st,dl,b_0,n} \sqsubseteq C_{st,dl,b_1,n}$. Then $C_{st,dl,b_0,n+1} \sqsubseteq C_{st,dl,b_1,n+1}$ follows from the approximation monotonicity of interpretation (lemma 2.3.2).

Suppose that for any n less than some limit ordinal l,

$$C_{st,dl,b_0,n} \sqsubseteq C_{st,dl,b_1,n}\,.$$

Then, by the definition of the approximation ordering,

$$\begin{aligned} C_{st,dl,b_0,n}[0] &\subseteq C_{st,dl,b_1,n}[0]\,,\\ C_{st,dl,b_0,n}[1] &\supseteq C_{st,dl,b_1,n}[1]\,. \end{aligned}$$

So

$$\begin{aligned} \bigcup_{n\in l} C_{st,dl,b_0,n}[0] &\subseteq \bigcup_{n\in l} C_{st,dl,b_1,n}[0]\,,\\ \bigcup_{n\in l} C_{st,dl,b_0,n}[1] &\supseteq \bigcup_{n\in l} C_{st,dl,b_1,n}[1]\,. \end{aligned}$$

9. Technically, it is the union of unions of the valuations of the corresponding iterative constructions at the respective ordinal stages. This is because if new definitions are added to a definition list, they do not interfere with the semantics of the already existing ones.

Therefore

$$(\bigcup_{n\in l} C_{st,dl,b_0,n}[0], \bigcup_{n\in l} C_{st,dl,b_0,n}[1]) \sqsubseteq (\bigcup_{n\in l} C_{st,dl,b_1,n}[0], \bigcup_{n\in l} C_{st,dl,b_1,n}[1])$$
$$C_{st,dl,b_0,l} \sqsubseteq C_{st,dl,b_1,l}$$

□

Lemma 2.4.5 (Correctness of construction of B's least fixed point)**.**
Let

- $st = (D, sig, I)$ be a structure,
- Var a set of variables, and
- dl a $DefList_{sig,Var}$.

Then the least fixed point of the operator $B_{st,dl}$ equals $B_{st,dl,|2^{D\times\mathbb{N}}|}$.

Proof. Analogous to that of lemma 2.4.3, with the approximation ordering. □

2.5 The well-founded model

Now that we defined the operators (and the iterative constructions), we can finally define the semantics of WFC.

Let $st = (D, sig, I)$ be a structure, and dl a $DefList_{sig,Var}$.
$M_{st,dl} = \mathrm{lfp}(B_{st,dl}) = B_{st,dl,|2^{D\times\mathbb{N}}|}$ is the *well-founded model* of dl.

The well-founded model $M_{st,dl}$ is a valuation such that $M_{st,dl}(x)$ is the triset represented by the definition x of dl.

Lemma 2.5.0 (The well-founded model is a model)**.**
Let $st = (D, sig, I)$ be a structure. Then for any definition list $dl\colon DefList_{sig,Var}$, $M_{st,dl}$ is a model of dl.

Proof.

$$\begin{aligned} M_{st,dl} &= \mathrm{lfp}(B_{st,dl}) \\ &= \mathrm{lfp}(C_{st,dl,M_{st,dl}}) \\ &= C_{st,dl,M_{st,dl}}(M_{st,dl}) \\ &= I_{st}(M_{st,dl})(dl) \end{aligned}$$

□

Lemma 2.5.1 (Existence of models)**.**
Every definition list has one or more models.

Proof. Every definition list has at least the well-founded model as a model. A definition list can have multiple models, take the one consisting of a single definition `type T = T`, every its valuation is stable. □

Given a structure st, a triset s is definable iff there exists a definition list dl such that $M_{st,dl}(x) = s$ for some x.

2.6 Conjecture: simpler semantics

This iterative construction only uses one valuation, which it updates using a standard interpretation function (ie. eg. complements are really complements), and whose stages at limit ordinals are defined using limit mix – an operation analogical to lim sup.

It seems to work for trivial examples, however, I'm not sure how I would go about proving equivalence with the semantics above. (Not that I've given it much thought.)

TODO define limit of sequences of valuations, note that it corresponds with set limit and limit of functions when trisets are taken as characteristic functions. A limit of a sequence of valuations contains the value v if from some index all valuations have v as member, does not contain the value if from some index all valuations don't, else the limit borders v.

MBS stands for Mix-Based-Semantics.

Let st be a structure. The gist of the idea is this:

$$\begin{aligned} MBS_{st,dl,0} &= v_e \\ MBS_{st,dl,n+1} &= I_{st}(MBS_{st,dl}) \\ MBS_{st,dl,l} &= \lim_{n \to l} \operatorname{mix} MBS_{st,dl,n} \end{aligned}$$

However, that is too coarse because of definitions like `let T = ~~T`, which are (and should be) undetermined. What we need is to define the steps not per definitions, but per every expression of the definition.

TODO Open question: is there a natural operator whose iterative construction is MBS?

3 Tracking undeterminedness

Although it is possible in general that the membership of some elements in a triset is undetermined, it is desirable that the trisets we work with in practice are classical. In this chapter, we investigate conditions under which this is the case.

TODO this chapter. There's a lot of stuff to copy from in my thesis.

4 Example WFCs

In this chapter, we provide a few example structures that define interesting WFCs. From now on, we will also use WFC as a countable noun that refers to a particular structure, or the set of trisets definable in it.

4.0 Logic programming

TODO $D = \{\top\}$, signature is empty.

4.1 Natural numbers

Let $\mathit{natSig} = (\{0, \mathrm{succ}, +, *\}, \{(0,0), (\mathrm{succ}, 1), (+, 2), (*, 2)\})$ be a signature.

$\mathit{NatC} = (\mathbb{N}, \mathit{natSig}, I)$, is the structure of well-founded collections of natural numbers, where:

$$I(0)() = \{\, 0 \,\}$$
$$I(\mathrm{succ})(N) = \{\, n + 1 \mid n \in N \,\}$$
$$I(+)(A, B) = \{\, a + b \mid a, b \in A, B \,\}$$
$$I(*)(A, B) = \{\, a * b \mid a, b \in A, B \,\}$$

Conjecture 0. *NatC* is the least expressive WFC of all the example WFCs of this chapter.

Open question 1. If so, why? What makes so inexpressive? What characterizes the *NatC*-definable trisets? What characterizes the classical ones?

Open question 2. Can exponentiation be defined in *NatC*?

4.2 Pairs

This WFC consists of the zero-tuple, which for convenience is considered an (improper) pair, and pairs formed of other pairs. Zeroth we define the domain.

The zero tuple () equals the empty set {}. Pairs are defined in stages in the following way:

$$\begin{aligned} Pair_0 &= \{\,()\,\} \\ Pair_{n+1} &= Pair_n \cup \{\,(a,b) \mid a,b \in Pair_n\,\} \\ Pair &= \bigcup_{n\in\mathbb{N}} Pair_n \end{aligned}$$

The structure contains just one operation – cartesian product.

Let $pairSig = (\{(), \times\}, \{\,((),0),(\times,2)\,\})$ be a signature.

$PairC = (Pair, pairSig, I)$ is the structure of well-founded collections of pairs, where:

$$I(())() = \{\,()\,\}$$
$$I(\times)(A,B) = \{\,(a,b) \mid a,b \in A,B\,\}$$

The projections can be defined as syntactic sugar:

```
zth(x) = x[0] = Un a, b: Pair .. (a, b) & x ? a : Empty
fst(x) = x[1] = Un a, b: Pair .. (a, b) & x ? b : Empty
```

Open question: under a natural coding of between pairs and naturals where in the arithmetical hierarchy are the $PairC$-definable sets of pairs? And infinite time Turing machines?

4.3 S-Pairs

Concatenation of tuples will be denoted using three dots: for example if t is a tuple and v an arbitrary value, then $(v, ...t)$ is the tuple consisting of v followed by the elements of t.

We could also allow pairs that potentially contain themselves, or pairs that contain such pairs, like for instance the pair $p = (p, p)$, or $((), p)$. We will call such pairs self-containing, or S-pairs, and we will denote them $<p, p>$, $<<>, p>$.

Because the sets of ZFC are not allowed to contain themselves, we will need to get a little creative to define such pairs. Think of how a pair p might be thought of as a container that under the index i contains the value $p[i]$. But $p[i]$ might itself be a pair, so it makes sense to think about $p[i][j]$, $p[i][j][k]$ and so on.

We will call sequences of indices like (i, j, k) *index paths*, and we can think of pairs as functions from index paths to what these pairs contain under said paths, or a special set *null* if the zero-tuple is encountered along the way.

We will represent "what these pairs contain under said paths" by the set of available indices at those paths – that is, if f is a pair such that $f[i]$ is a proper pair, then $f((i)) = \{0, 1\} = 2$, else if $f[i]$ is a zero-tuple, then $f((i)) = \{\} = 0$, else if f itself is a zero-tuple, then $f((i)) = null$.

A (potentially) self-containing pair is a function $f\colon 2^* \to \{0, 2, null\}$ such that
- $f(()) \neq null$, and
- $f(s) \neq null \wedge i \in f(s)$ iff $f((...s, i)) \neq null$.

SPair is the set of all such pairs.[10]

What distinguishes self-containing from the non-self-containing pairs is that every non-self-containing pair p has a finite depth. In other words, for every index path *path* longer than some constant, $p(path) = null$, while this does not hold for self-containing pairs.

The signature of *SPairC* is identical to that of *PairC*.

$SPairC = (SPair, pairSig, I)$ is the structure of well-founded collections of s-pairs, where:

$$I(<>)() = \{<>\}$$
$$I(\times)(A, B) = \{<a, b> \mid a, b \in A, B\}$$
$$ab = (i, ...s) \mapsto \begin{cases} a(s) & i = 0 \\ b(s) & i = 1 \end{cases}$$

One can define the projections in the same way as for pairs.

Pairs are a subset of mpairs under a reasonable coding, and one may easily define the set of all non-self-containing pairs in $SPairC$[a], so all trisets of pairs definable in *PairC* are definable in *SPairC* as well.

Open question 3. Is the reverse true in some sense? Can one simulate (definable) mpairs and collections of them in *PairC*?

a. `let p = | pp`

4.4 Quantified boolean grammars

Boolean grammars [3] are an extension of context-free grammars that include the connectives conjunction and negation. Quantified boolean grammars are a WFC that extends boolean grammars with arbitrary unions and intersections.

Let $Alph$ be an arbitrary set. $Word_{Alph} = Alph^*$ is the set of finite tuples of elements of $Alph$.

$wordSig_{Alph} = (Word_{Alph}, \{\,(), .\,\}, I)$ is the structure of quantified boolean grammars (well-founded collections of words), where

$$I(())() = \{\,()\,\}$$
$$I(.)(A, B) = \{\,(...a, ...b) \mid a, b \in A, B\,\}$$

Open question 4. How does this one compare to the other ones in terms of expressive power?

4.5 Open problems

Perhaps (some of) these problems have the same answer for any (reasonable) WFC, but I thought it safe to ask relatively to the concrete structures defined above (at least at first).

Open question 5. Is every definable triset definable in a definition list with only one model?

Open question 6. Is there a definition list that is not classical, but is desirable in the sense that one might prefer it to one of its classical completions, for whatever purpose?

Conjecture 7. There is a definable type that has no definable classical completion.

Then we have some questions that are specifically about our example WFCs:

Open question 8. How do our example WFCs compare in terms of expressive power? Where are their definable sets in the arithmetical hierarchy? How can these questions be made precise? (I suppose there are multiple ways of answering

the last question.)

5 WFCs and computation/Definable sets

TODO - HERE BE DRAGONS

TODO can the definable sets of PairC be computed by an ITTM? Can PairC define the ITTM-computable sets or their floors?

TODO define a triset s as computable by an (IT)TM t iff

0. t accepts a member of s, rejects a nonmember and doesn't halt on bordering elements.
1. alternatively t is allowed to do anything on the bordering elements. (TODO weakly computable?)

How do these two notions compare? Is the latter a useful concept?

A triset is semi-decidable by an (IT)TM t iff

0. t accepts a member of s, doesn't accept a nonmember and doesn't halt on a bordering element.
1. alternatively t is allowed to do anything on the bordering elements.

Is this the right notion of semidecidability?

6 The candidates for sets/Sets as zeroth-class citizens

In this chapter, we describe the structures that we see as potential candidates for the sets of our set theory.

TODO looking for models of an unknown theory is strange.

6.0 The deep/rough syntactic model

TODO if the set of down-classical/floored sets (make all undetermined elements not belong) of the rough model is larger than the set of classical sets, then the set of down-classical sets should also be considered a potential model of ETST

TODO take into account the regularized version of the deep model (take all the classical sets that a set is approximating, define membership to be true if true for all combinations, false if false for all combinations, undetermined otherwise). Is it isomorphic to the shallow model?

TODO also try using (arbitrary) union and its complement. Union of A - a set of sets S such that there exists E containing S in A Dual union of A - a set of sets S such that for all E containing S, E in A

D(A) = U(A)

TODO naturalness check: do the models correspond to ITTM-3-computable sets?

A 3-function is (strictly) computable iff for some model of computation, there exists a machine that accepts on all members and rejects all nonmenbers (and else doesn't halt).

Given a reasonable An ITTM-computable set is an ITTM-computable function from I functions computable by an ITTM

does the floor (or, equivalently, ceil) of the rough model correspond to computable sets of ITTMs with oracle for ordinary ITTMs?

6.1 The shallow syntactic model

A shallow model is built of the classical sets of the deep model.

6.2 The positive syntactic model

6.3 The "sets" of WFRQ

We hope this one is equivalent to the deep syntactic model, if not, we need to consider it (and its shallow version) as another potential semantics for ETST.

6.4 Equivalence classes of T = $\leq \omega$-tuples of T

Basically, the "sets" of SPairC

6.5 Some countable set of MRecords

I've played with the idea of defining MRecords as equivalence classes of leq-omega-tuples where swapping indices is equivalent, and then somehow choosing just those that are definable.

One can easily define the operations on sets of MRecords (cardinality concerns: just relax powerset appropriately), but then selecting the "definable" ones seems to be a problem.

7 A general three-valued logic

TODO make a logic where variables hold values, not collections of them, then make the Logic of Partial Collections a special case of it?

TODO describe the problems of making the logic too general - eg. if one adds one predicate per original predicate that is true iff the original predicate is undetermined, and else false, then undetermined ceases to act like "unknown". So should such predicates be disallowed (by some monotonicity condition? on what partial order?), should such structures be allowed, but not (finitely?) axiomatizable, am I wrong about that the "unknown" interpretation ceases to apply, or should I get used to that it no longer does in first-order three-valued logic?

8 A three-valued logic/Logic of partial collections

In this chapter, we describe a logic similar to first-order logic, tentatively named "Logic of Partial Collections" (LPC), where the "partial" should be understood as "partially specified".

The logic differs from first-order logic in that variables hold collections of elements instead of elements themselves. However, like in first-order logic, quantifiers in LPC range over elements, or rather one-element collections, instead of over arbitrary collections as in second-order logic.

The collections are three-valued in the sense that every element either belongs to a collection, doesn't belong to it, or its membership in that collection is undetermined. Collections with undetermined elements are meant to potentially represent any two-valued collection that would result from the three-valued one if the membership of all undetermined elements became settled. The undetermined part of a three-valued collection represents our lack of information of which collection precisely we are talking about.

Likewise, the logical connectives of LPC satisfy the following condition: if the replacement of the undetermined part of an input i with classical logical values always results in the same output, then that output is the output corresponding to the input i, else the output is undetermined. This implies "undetermined" acts as "unknown", and getting to know more of the input may only result in potentially getting to know the previously unknown output, not a change in the output if it was already known.

As a consequence, the logic is *not* functionally complete. For instance, a unary logical connective mapping true and false to false and undetermined to true is inexpressible, because such a connective would break the above condition. LPC is, however, functionally complete in its classical part.

I followed [**firstOrderLogic**] to make sure my formalization of LPC is reasonably close to first-order logic.

8.0 Syntax

The syntax of our logic depends on the chosen signature that determines which functions and predicates can be used in its terms and formulas.

A signature is a triple $(\mathit{func}, \mathit{pred}, \text{arity})$, where
0. *func* and *pred* are disjoint sets containing the function and predicate symbols, respectively, and
1. arity: $\mathit{func} \cup \mathit{pred} \to \mathbb{N}$ determines the arity of the symbols.

8.0.0 Terms

$$E = x \mid \mathcal{E} \mid \mathcal{U} \mid f(...) \mid E \cup E \mid E \cap E \mid {\sim}E \mid \bigcup_{x \in E} E \mid \bigcap_{x \in E} E \mid F \ ? \ E : E$$

Terms represent collections of elements.

Let *Var* be the set of all variables, we assume there are countably many.

Given a signature $(\mathit{func}, \mathit{pred}, \text{arity})$, terms are recursively defined as follows:
0. Variables are terms.
1. $\mathcal{E}$ and $\mathcal{U}$, representing the empty and universal collection, respectively, are terms.
2. If $t_0, ..., t_n$ are terms, $f \in \mathit{func}$ and $\text{arity}(f) = n$, then $f(t_0, ..., t_n)$ is a term.
3. If t_0 and t_1 are terms, then the binary union $t_0 \cup t_1$ is a term.
4. If t_0 and t_1 are terms, then the binary intersection $t_0 \cap t_1$ is a term.
5. If t_0 is a term, then the complement $\sim t_0$ is a term.
6. If t_0 and t_1 are terms, then the quantified union $\bigcup_{x \in t_0} t_1$ is a term.
7. If t_0 and t_1 are terms, then the quantified intersection $\bigcap_{x \in t_0} t_1$ is a term.
8. If f_0 is a formula (see below), and t_0 and t_1 are terms, then the conditional $f_0 \ ? \ t_0 : t_1$ is a term.

$_V$, where V is a set of variables, is the set of all terms whose free variables are in V.[a]

$=_{Var}$ is the set of all terms.

a. A formal definition of $_V$ in ZFC is given in appendix **??**.

8.0.1 Definitions

TODO replace "name" by variable everywhere?

Named terms are called definitions. The following definitions are implicitly part of all definition lists (see below) where they are mentioned (unless being defined otherwise).

```
let Empty = Empty;
let Universal = ~Empty;
let Undetermined = ~Undetermined;
```

8.0.2 Definition lists

TODO statements should be monotonic under the approximation ordering

Definitions are grouped into definition lists. A definition list allows its definition to refer to each other.

A family of terms is a function $f : V \to_V$ for some $V \subseteq Var$.

A family of terms is a *definition list* iff it is the union of (possibly infinitely many) finite families of terms.

$_V$ is the set of definition lists whose domain is V.
$=_V ar$ is the set of all definition lists.

The purpose of the restriction that defines definition lists is to disallow those definition lists that would contain infinite dependency chains between definitions, like: `let T0 = T1; let T1 = T2; let T = T3; ...`

8.0.3 Formulas

$$F = \quad tt \mid ff \mid E = E \mid P(...) \mid F \vee F \mid F \wedge F \mid \neg F \mid \exists x \in T : F \mid \forall x \in T : F$$

Formulas represents mathematical statements, and have a truth value. Given a signature $(func, pred, \text{arity})$, formulas are recursively defined as follows:

0. ff and tt, representing true and false, respectively, are formulas.
1. If t_0 and t_1 are terms, then $t_0 = t_1$ is a formula.
2. If $t_0, ..., t_n$ are terms, $P \in pred$ and $\text{arity}(P) = n$, then $P(t_0, ..., t_n)$ is a formula.
3. If f_0 and f_1 are formulas, then the disjunction $f_0 \vee f_1$ is a formula.
4. If f_0 and f_1 are formulas, then the conjunction $f_0 \wedge f_1$ is a formula.
5. If f_0 is a formula, then the negation $\neg f_0$ is a formula.

6. If t_0 is a term and f_0 is a formula, then $\exists x \in t_0 \colon f_0$ is a formula.
7. If t_0 is a term and f_0 is a formula, then $\forall x \in t_0 \colon f_0$ is a formula.

We may use $F \implies F$ as syntax sugar for $\neg F \vee F$.

8.1 Semantics

In the whole section, we will assume a given signature $s = (\mathit{func}, \mathit{pred}, \text{arity})$.

8.1.0 Relations

An n-ary relation on D is a three-valued collection of n-tuples of two-valued collections of elements of D.

TODO perhaps a relation should be a three-valued collection of two-valued relations?

Let D be an arbitrary set, which we'll call the domain, and n a finite ordinal.

An n-tuple of elements of some set S is a function from n to S.

An n-ary relation on D is a pair (L, U), where L and U are sets of n-tuples of subsets of D and $L \subseteq U$.

Functions are a special case of relations.

A relation (L, U) is classical, or two-valued, iff $L = U$.

A functional n-ary relation is a classical n-ary relation such that $0 < n$ and for all tuples $t_a = (t_{a,0}, ..., t_{a,n})$ and $t_b = (t_{b,0}, ..., t_{b,n})$ of that relation, $(t_{a,1}, ..., t_{a,n}) = (t_{b,1}, ..., t_{b,n})$ implies $t_a = t_b$.

Open question 9. Could I loosen the definition of functional relation so that functions with several undetermined return values are allowed?

Open question 10. Should all functions be required to be monotonic?

TODO delete the rest of the subsection?: We defined a relation such that it naturally contains, doesn't contain, or borders tuples of two-valued collections. We will now extend the membership relation of relations to tuples of three-valued collections, using the notion of an extended relation – a three-valued collection of tuples of three-valued collections.

A three-valued collection (L, U) approximates a two-valued collection C iff $L \subseteq C \subseteq U$.

An n-ary expanded relation on D is a pair (L, U), where L and U are sets n-tuples of pairs (l, u), $L \subseteq U$, and $l \subseteq u \subseteq D$.

A relation r induces an extended relation $r_e = (L, U)$ iff:
0.

lemma: the membership status of a tuple t of three-valued collections in a relation equals the mix of the membership statuses of all the tuples of two-valued collections that t approximates. TODO or perhaps just define the interpretation of formulas using mix, and delete "extended relations"?

8.1.1 Structures

A structure is an object relative to which one judges the truth of formulas, and which determines how to interpret terms and definitions.

A structure is a pair (D, v) where
0. D is called the domain of the structure, and
1. v is called valuation, and is a function from symbols to relations such that:
   (a) for every $f \in \mathit{func}$, $v(f)$ is an $(\text{arity}(f) + 1)$-ary functional relation,
   (b) for every $P \in \mathit{pred}$, $v(P)$ is an $(\text{arity}(P))$-ary relation.

TODO structures are descriptively equivalent of they satisfy the same theories

8.1.2 Interpretation

The interpretation of terms and formulas is defined recursively on their structure.

TODO

TODO the mix is the greatest lower bound with respect to the approximation ordering.

TODO two kinds of semantics - arbitrary collections vs definable collections

perhaps call them: freewheeling semantics (does not bother to define what a collection is) vs grounded (?, or perhaps "restricted") semantics (the collections are the definable collections)

TODO if functions are only arbitrarily defined on classical collections, and the definition on nonclassical collections N is extended by sth like "for all classical collections that N could represent, return the union of that", should the union be $\cup_{\ldots} X$ or $\cup_{\ldots} \{X\}$?

TODO possible restriction - only allow relations/functions that are defined pointwise, ie that are definable on the elements, not collections of them.

$= \{0, \perp, 1\}$ is the set of truth values.

8.2 Deductive system

9 Formulas as syntactic sugar

TODO can formulas be algorithmically converted to definitions that are inhabited iff those formulas are true?

10 Definitions as syntactic sugar

Can all definitions be converted to formulas which uniquely characterize the triset represented by that definition?

11 Boolean set theory

In this chapter, we define the syntax of ETST, and informally describe its meaning. The syntax may be denoted using two possible notations – math (eg. $A \cup B$) or code notation (eg. `A | B`). We'll use both interchangeably in this document.

conjecture: for each theory/structure that has predicates that are undetermined on some unit element, there is another equivalent structure where all predicates are classical on all unit arguments (an element E is a unit element if for all elements A, B such that A—B=E, E = A or E = B)

11.0 Terms and formulas of ETST

Expressions are recursively defined as follows:

$$E = x \mid \mathcal{E} \mid \mathcal{U} \mid P(E) \mid M(E) \mid E \cup E \mid E \cap E \mid {\sim}E \mid \bigcup_{x \in E} E \mid \bigcap_{x \in E} E \mid F \mathbin{?} E : E$$

$$F = \ tt \mid ff \mid E \in E \mid \ E = E \mid F \vee F \mid F \wedge F \mid \neg F \mid \exists x \in E \colon F \mid \forall x \in E \colon F$$

We'll use the letter e with an optional subscript (eg. e_0) in math notation and square brackets (eg. `[e0]`) in code notation as metavariables that hold expressions. In detail, the expressions are:

Names
: Refer to other sets. Our names begin with a letter, which is followed by letters and numbers.[11]

The empty set
: Represents the set containing no elements.

The universal set
: Represents the set containing all sets.

11. Using regex notation, our names are those strings that match [a-zA-Z][a-zA-Z0-9]*.

Power set

$P(e)$ or `P([e])` represents the set of all subsets of the set represented by the expression e.

Meet set

$M(e)$ or `M([e])` represents the set of all sets having a common element with the set represented by the expression e.

Binary union

$e_0 \cup e_1$ or `[e0] | [e1]` represents the set of those sets that are elements of either e_0 or e_1.

Binary intersection

$e_0 \cap e_1$ or `[e0] & [e1]` represents the set of those sets that are elements of both e_0 and e_1.

Complement

$\sim e$ or `~[e]` represents the set of those sets that are *not* elements of e.

Quantified union

$\bigcup_{x \in e_0} e_1$ `Un x: [e0] => [e1]` `Un x: [e0] { [e1] }`

Represents the union of all instances of its body e_1 where the name x is replaced by the members of e_0.

Quantified intersection

$\bigcap_{x \in e_0} e_1$ `Ir x: [e0] => [e1]` `Ir x: [e0] { [e1] }`

Represents the intersection of all instances of its body e_1 where the name x is replaced by the members of e_0.

Conditional operator

f_0 ? e_0 : e_1 or `[f0] ? [e0] : [e1]` represents a set that equals e_0 if the formula (defined below) f_0 is true, and equals e_1 if f_1 is false.

This operator is included for convenience, but as we'll see, it can be considered just syntax sugar. To prove it, though, we'll define it as an ordinary expression.

[While theories usually are specified by axioms, we are going to .. from the back/ go the opposite direction: we'll start with our desired models, and then see if we can find reasonable axioms whose models/theories are close enough.]

There is a question of whether the undeterminedness should be shallow or not, in the sense of whether $\{\{A?\}\}$ and $\{\{\}?, \{A\}?\}$ should be the same set or not. Answering yes leads to shallow semantics (TODO which I'm not sure can be defined), answering no leads to deep semantics (which are surely possible, but I believe less desirable, because in such semantics, quantifying over all sets necessarily quantifies over the non-classical ones, so there's no way to avoid it).

The models presented below are called syntactic models because they use the syntax of ETST to define its semantics[12]. This makes them countable; however, this does not imply there aren't (more) natural models of higher cardinalities.

Because the syntactic models are interpret a set S as a collection of formulas that define sets that are elements of S.

11.1 Deep syntactic model

11.1.0 Valuations

Valuations let us interpret expressions by giving meaning to names.

$= P() \times P())$

A valuation is a function $v\colon\ \to$.

11.1.1 Interpretation

$I_{b\colon \to}\colon (c\colon\ \to) \to \to$

$I_b(c)(x) = c(x)$

$I_b(c)(\mathcal{E}) = (\emptyset, \emptyset)$
$I_b(c)(\mathcal{U}) = (,)$

$$I_b(c)(P(e)) = \bigcup_{x\in} \begin{cases} x & b(x) \subseteq c(e) \\ \emptyset & \text{else.} \end{cases}$$

$$I_b(c)(M(e)) = \bigcup_{x\in} \begin{cases} x & c(x) \circledcirc c(e) \\ \emptyset & \text{else.} \end{cases}$$

$I_b(c)(e_0 \cup e_1) = I_v(e_0) \cup I_v(e_1)$
$I_v(e_0 \cap e_1) = I_v(e_0) \cap I_v(e_1)$
$I_v(\sim x) = \setminus\ v(x)$

12. In other words, they can be considered "syntax in disguise".

$$I_v(\bigcup_{x\in\mathcal{U}} e) = \bigcup_{x\in} I_{v[x\mapsto v(x)]}(e)$$

$$I_v(\bigcap_{x\in\mathcal{U}} e) = \bigcap_{x\in} I_{v[x\mapsto v(x)]}(e)$$

11.1.2 Positive expressions

We'll define an expression e as positive iff

0. it does not contain the conditional operator,
1. for all its subexpressions `Un/Ir x: [e0] => [e1]`, $e_0 = \mathcal{U}$, and
2. for all its subexpressions $\sim e$, e is a bound name.

PExpr is the set of all positive expressions.

11.1.3 Two-interpretation

A two-interpretation is an extension of two-valuations to all positive expressions.

$I_{v:\to P()}\colon PExpr \to P()$

$I_v(x) = v(x)$

$I_v(\mathcal{E}) = \emptyset$

$I_v(\mathcal{U}) =$

$$I_v(P(e)) = \bigcup_{x\in} \begin{cases} x & v(x) \subseteq v(e) \\ \emptyset & \text{else.} \end{cases}$$

$$I_v(M(e)) = \bigcup_{x\in} \begin{cases} x & v(x) \cap v(e) \text{ is non-empty} \\ \emptyset & \text{else.} \end{cases}$$

$I_v(e_0 \cup e_1) = I_v(e_0) \cup I_v(e_1)$

$I_v(e_0 \cap e_1) = I_v(e_0) \cap I_v(e_1)$

$I_v(\sim x) = \setminus\ v(x)$

$$I_v(\bigcup_{x\in\mathcal{U}} e) = \bigcup_{x\in} I_{v[x\mapsto v(x)]}(e)$$

$$I_v(\bigcap_{x\in\mathcal{U}} e) = \bigcap_{x\in} I_{v[x\mapsto v(x)]}(e)$$

TODO shit, powerset is not monotonic wrt the standard ordering. We need a better ordering. Or another approach.

12 Comparing deep and shallow ETST

Open question 11. Are the classical deep sets the same as the classical shallow sets?

Open question 12. Is there a set of sets classical in either semantics, in either semantics?

Open question 13. What about mutual expressibility?

13 Deductive system

TODO move this into other chapters

13.0 Rules of inference

13.1 Axioms of ETST

Some candidates:

$\forall x\colon x \notin \mathcal{E}$
$\forall x\colon x \in \mathcal{U}$
$\forall x, y\colon (x \in P(y) \iff \forall z\colon z \in x \implies z \in y)$
$\forall x, y\colon (x \in M(y) \iff \exists z\colon z \in x \land z \in y)$
$\forall x, y, z\colon (z \in x \cup y \iff z \in x \lor z \in y$
$\forall x, y, z\colon (z \in x \cap y \iff z \in x \land z \in y$

14 Algorithms, type checking, etc?

TODO does this belong here?

When you prove something once, it should be proven everywhere. Yes, this leads to non-locality of type checking, because, where a proof in one part of the code affects the correctness of code everywhere else.

A proof in [my] programming language should ideally be readable even to someone that has never seen the language before.[13]

13. Assuming the particular proof in ETST and ZFC would look essentially the same.

15 The productive fixed point

TODO This chapter may be safely skipped.
TODO: Define equations as pairs of expressions that consist of variables, custom operations, and perhaps also union, intersection and complement.

Define a theory as a signature with a set of equations, define a structure in terms of a theory, not a signature, and add to the definition of a structure the requirement that the equations are satisfied. Perhaps define the interpretation of op as a partial function of type $D^{\text{arity}(op)} \to D$, and define

TODO define partial algebras as (D, op, I) (or similarly), define a signature as containing a set of equational laws (pairs of (certain) expressions)

Helpful source: `https://books.google.com/books/about/Introduction_to_the_Theory_of_Abstract_A.html?id=8RunBAAAQBAJ&printsec=frontcover&source=kp_read_button&hl=en&redir_esc=y#v=onepage&q&f=false`

or Universal Algebra by G. Gratzer

then extend the interpretation such that it also accepts approximative values and is required to return values that are approximated to an arbitrary degree.

Bibliography

[0] Olivier Esser. "On the Consistency of a Positive Theory". In: *Mathematical Logic Quarterly* 45.1 (1999), pp. 105–116. DOI: https://doi.org/10.1002/malq.19990450110. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/malq.19990450110. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/malq.19990450110.

[1] Vassilis Kountouriotis, Christos Nomikos, and Panos Rondogiannis. "Well-founded semantics for Boolean grammars". In: *Information and Computation* 207.9 (2009), pp. 945–967. ISSN: 0890-5401. DOI: https://doi.org/10.1016/j.ic.2009.05.002. URL: https://www.sciencedirect.com/science/article/pii/S0890540109001473.

[2] Jozef Mikušinec. "Well-founded semantics for dependent, recursive record types with type complement". MA thesis. Brno, Czechia: Masaryk University, June 2022.

[3] Alexander Okhotin. "Conjunctive and Boolean grammars: The true general case of the context-free grammars". In: *Computer Science Review* 9 (2013), pp. 27–59. ISSN: 1574-0137. DOI: https://doi.org/10.1016/j.cosrev.2013.06.001. URL: https://www.sciencedirect.com/science/article/pii/S157401371300018X.

[4] Benjamin G. Rin. "Transfinite recursion and computation in the iterative conception of set". In: *Synthese* 192 (2014), pp. 2437–2462. URL: https://link.springer.com/article/10.1007/s11229-014-0560-9.

A Proof of correctness of iterative constructions

We'll start with a lemma limiting the cardinality of well-ordered chains of finite-domain valuations.

The lemma is not strictly necessary if one is willing to accept $|2^{2^D}|$ as the ordinal up to which we need to build our iterative constructions, instead of just $|2^D|$. The larger limit follows by a simple application of the pigeonhole principle.[14]

Lemma 2.4.1 (Well-ordered chains of valuations are small)**.**
Let *Var* be a *finite* set of variables, and $V \subseteq \mathit{Valuation}_{\mathit{Var},D}$ a well-ordered chain of valuations with respect to either ordering. Then the cardinality of V equals that of D up to a finite difference.

Proof. For the purposes of this proof, we'll use the definition 1 of trisets, that is, a triset is a function from D to $\{0, \bot, 1\}$. Note the standard ordering is the pointwise order for $0 < \bot < 1$, and the approximation ordering is the pointwise order for $\bot < 0, 1$.

Let V' be V without its greatest element, if it exists. For any $v \in V'$, v has a successor in V – the least element greater than v.

Let $f \colon V' \to \mathit{Var} \times D \times \{0, \bot, 1\}$ be a function that to v assigns a triple (x, d, t) such that the truth value t of membership of d in $v(x)$ increases at v's successor. (More formally, that to v_0 assigns a triple $(x, d, v_0(x)(d))$ where v_1 is v_0's successor and $v_0(x)(d) < v_1(x)(d)$.)

The function f is an injection. (In a chain, no element can "increase" its membership twice to the same value.) Imagine $f(v_0) = (x, d, a)$ and $f(v_1) = (x, d, b)$, with $v_0 < v_1$. Then $a < v_1(x)(d) = b$, so $a \neq b$.)

We have an injection from V' to $\mathit{Var} \times D \times \{0, \bot, 1\}$. The latter has the cardinality of D (up to a finite difference), so V' and V must as well. □

14. I feel limiting the cardinality of the iterative construction has certain benefits, although I am unaware of any.

Lemma 2.4.2 (Correctness of the construction of least fixed points)**.**
Let T be a set with a chain-complete partial order $\leq$ such that all well-founded chains of T have a cardinality at most k, and $O\colon T \to T$ a monotonic operator on T.

Let $O_{n\colon Ord}\colon T$ be an element of T defined like this:

$$\begin{aligned} O_0 &= \text{ the least element of } T \\ O_{n+1} &= O(O_n) \\ O_l &= \text{ the least upper bound of } O_{n<l} \end{aligned}$$

Then $O_{|2^k|}$ is the least fixed point of O.

Proof. We'll do the proof in two steps. Zeroth, we'll prove that for any ordinal n, O_n is (non-strictly) less than any fixed point of O, using transfinite induction. First, we'll prove that O_{2^k} is a fixed point of O. Together, these two facts imply that O_{2^k} is the least fixed point of O.

Subproof 0. O_0 is less than any element of T, including any fixpoint t of O. (Any chain-complete ordering has a least element.)

Suppose O_n is less than t. Then by the monotonicity of O,

$$O_{n+1} = O(O_n) \leq O(t) = t\,.$$

Suppose that for every n less than some limit ordinal l, O_n is less than a fixpoint t. t is thus an upper bound of $O_{n<l}$. But O_l is the least uppper bound, so it must be less than t. ■

Subproof 1. Elements O_n for some n form a chain (by the monotonicity of O). The chain is well-founded, so its cardinality is at most k.

There are 2^k many ordinals less than 2^k, so by the pidgeonhole principle, there must be distinct n and m such that $O_n = O_m$; let n be the lesser one. Every element between O_n and O_m equals O_n, including $O_{n+1} = O(O_n)$. O_n is thus a fixpoint of O, and for all $k \geq n$, O_k equals O_n. In particular, O_{2^k} does, so it is a fixpoint of O. ■

□

B TODO notes

TODO another candidate for a model of ETST: WFC of natural numbers with addition and multiplication.

TODO I should revisit the idea of defining the semantics using dual definitions: let A = B — A ; let A' = B' & A';

let A = A ; let A' = A';

Note if this works, then it's unnecessary, because just using positive formulas defines all the collections definable with complement as well.

TODO how to define functions? through least fixpoints wrt partial descriptions, optimal fixed points, greatest fixed points?

should this contain only finite sets of natural numbers? I guess so let SetNat = — Ex n: Nat .. SetNat — n

similarly, should this be the empty set? I guess not. let F = Any -¿ F;

Here the problem is self-reference. Greatest-fixpoint semantics when it comes to self-reference are ok, general greatest-fixpoint semantics are not. A set should be isomorphic to a function Any -¿ 2. Getting better approximations depth-wise would populate F, but SetNat has constant depth. And breadth-wise, there should be no greatest-fixpoint-like semantics.

Also, 'let R = P(R)' should include the set s = s.

let Nat = — Ex n: Nat .. n — n

x in Ex x: X .. B ¡=¿ x in B[x in X]

// The set S of all sets whose elements are all in S. // Should this set include the set $A = A$? let S = Ex s .. All e: s .. Ex c: e & S .. s;

let A = A ;

I found an answer to my question why record sets are best defined using least fixed points, but function sets with greatest fixed points. The reason is that records are well-founded, while functions are not (A = (null, A)) is a function, accepting null and returning itself. Least fixed points do not work well for non-well-founded data structures.

TODO can I mix least and greatest fixpoint definitions? Is it stricty more expressive as either?

could a set be formalized as 'f: N* -¿ P(N) — null' such that f([]) != null, f(s)

!= null & e in f(s) ¡=¿ f([...s, e])!

B.0 The rest

Let *Variable* = $\mathbb{N}$ be the set of all names.

TODO: The set containing exactly the set a (ie, the set $\{a\}$) can be defined like this:

```
Ex seSuA: P(P(a)) =>
  Ex _: (All suA: seSuA => All _: a & ~suA => None)
    & (Ex _: seSuA => Universal) => seSuA.
```

TODO this: https://arxiv.org/pdf/2112.04436.pdf and references therein seem to be unrelated, they use three-valued logic to study partial functions, and I believe my use case is different. I have only looked at it very briefly, so I might be wrong.

TODO does my set theory contain all the sets of ZF(C)? what extra axioms are necessary for that, if possible at all? More generally, what is the relationship between the sets of ETST and ZFC

TODO does aczel's anti-foundation "axiom" hold in ETST? Is ETST anti-well-founded?

TODO is ETST effectively axiomatizable?

TODO Soft question: are there things that are only provable using three-valued objects? Are there things that can be proven fully classically, but where the three-valued proof is considerably shorter/ more elegant / more desirable in any sense?

TODO Are WFCs with non-well founded values just conservative extensions of WFCs without them? (Can I simulate the former using the latter?)

TODO this might be relevant, read it: https://www.cambridge.org/core/journals/journal-of-symbolic-logic/article/abs/self-reference-with-negative-types/5621B22269886CAB0048F97C5DB

TODO because of well-known cardinality issues, we're gonna have to cheat a little.

TODO in general, is it possible to have a set of all non-approximated values? TODO in general, is it possible to have a set of all well-founded values? TODO in practical programming, would one get spurious (unintended) non-well-founded values in their sets either often enough that it would be irritationg, or could it happen that one could not easily tell just by looking at a definition whether it induces non-well-founded values, or that once could not tell how to easily or elegantly get rid of them? In general, would the non-well-founded values be a pain in the ass, at least sometimes?

TODO some questions, like "Does complement increase the expressive power of WFC" have an easy answer because of degenerate WFCs, like ones where some values are "hidden", ie. no custom operations return them under any circumstances.

Is there a reasonably general condition that disallows such degenerate cases? General both in the sense that it's unrestrictive, as well as that it is useful for more than one purpose.

TODO call hereditarily countable bisets hecoses?

TODO a condition that avoids pathological WFC signatures? "There are only finitely many operations, and for every value of D, the triset containing exactly that value is definable."

TODO another candidate: self-containing pairs, where sets are infinite sequences (val0, (val1, (...))) representing { val0, val1, ... } another candidate: infinite sequences, as above

TODO extensionality should hold, also the truth of partially undetermined formulas should be determined by the mix of their classical completions

TODO on one hand, these sets have different classical completions. A = ?, A ? : , , A, ,A B = A? : , A on the other hand, they only possibly contain the classical sets and A, so I guess that an argument can be made that by extensionality, they should be the same set, right? Because they have the same characteristic function. . But if they are equal, it would mean that X & ! and Y & ! would also have the same characteristic function – in effect, you could never remove . it seems two incompatible views are emerging: 0. trisets as sets of their classical completions, or 1. trisets as characteristic functions . it seems that the view 1 is the one compatible with WFC. I mean, WFC is based on three-valued characteristic functions

TODO conjecture: this is a useful sanity condition: "every triset that appears in the iterative construction is definable" . another conjecture: for any insane (pick a better name) WFC system, there exists a corresponding one that satisfies the condition

TODO add an open question about the supremum of ordinals clockable by particular natural WFCs (look at the open question 0 in your magister thesis). . In general, salvage as many conjectures and open questions from your MAT as reasonable.

TODO Are there certain (natural) type definitions that cannot be proven to be wholly determined, but whose 2-valuedness as an assumption leads to interesting consequences? TODO is the expressive power of WFCs equal to that of ITTMs?

TODO: the issue I have encountered when trying to automatically define the approximative values given the rest of the signature is this: . My idea was to treat sets of D as the approximative values for its elements. For instance, $* = P(D)$, $(*, 3) = (x, 3) \mid x \in D$, etc. However, given a triset (containing such approximative values), I don't know how to define which elements of D are actually approximated (and I'm afraid this approach is just fundamentally flawed). . The reasonable step forward is to say that a triset s approximates a value d iff it contains arbitrarily

good approximations of d. However, take this triset:

$$\{*, (*,*), (*,(*,*)), (*,(*,(*,*))), (*,(*,(*,(*,*))))\}\,.$$

. The issue is that the approximations do get arbitrarily better, but only in some directions. What we need is for them to get arbitrarily better in all directions in order for them to actually approximate a value. However, it seems to me that in order to define what it means to "get better in all directions", one needs a notion of "depth" of an approximative value, and I'm afraid that this notion of depth is not definable – the information is simply not in the set, for how would you distinguish between $*$ and $()|(*,*)$, when in the world of pairs, they are the same set? . Does it even make sense to talk about "productive" definitions without an additional structure on D?

TODO mention that distinct sets of classical trisets can have the same characteristic function (and therefore are represented by the same triset) . "Maybe there is an approach to a three-valued set theory that is based on classical completions, however such an approach seems to be at odds with how WFCs work. WFCs"

TODO replace all $x[0]$ and $x[1]$ with $x\rfloor$ and $\lceil x\rceil$, respectively.

TODO add chapter about two-valued collections, without complement, but with value approximation

TODO is there a nice (not sure what that meas, but I guess it includes "non-syntactic") characterization of positive/classical/definable (tri)sets? Or at least a nice generalization. Something like what Scott-continuity is for computability?

Is every floor of a definable type definable?

TODO this might be of relevance to you: A Completeness Proof for A Regular Predicate Logic with Undefined Truth Value - Antti Valmari https://arxiv.org/pdf/2112.04436.pdf . perhaps also: A simple logic for reasoning about incomplete knowledge - Mohua Banerjee, Didier Dubois https://www.sciencedirect.com/science/article/pii/S0888613X13002478?ref $RR - 2rr = 720bf7b3ec9bb36c$

TODO given a definition in PC/WFC in productive/greatest fixpoint semantics, can it be mechanically transformed into an equivalent one in least fixpoint semantics? . The answer is yes for Least and greatest fixed points, and do prove it somewhere, and do mention it in the chapter 2.

TODO so we have quite a lot of semantics: least, productive, greatest, and uncommitted fixed point semantics, for PC and for WFC. How do they relate in terms of expressive power? . For PC: every greatest point can be defined as the least fixed point of the negated definition, and vice versa. . a definition list has only one model iff its uncommitted fixed point is classical

TODO Open question: for any positive-definable set, is the complement positive-definable? TODO Open question: for any WFC-definable triset, is the complement WFC-definable?

TODO (when) is there a positive "triset of all positive trisets"*? TODO (when) is there a "triset of all positive trisets"*? TODO (when) is there a "triset of all trisets"*? *in the appropriate sense.

TODO given a (ascending/descending) chain of trisets/valuations, their limit equals their least upper bound/greatest lower bound

TODO is there a classical definable triset that is not positive-definable?

TODO is every (positive-)definable triset (positive-)definable in a definition list with a single model?

TODO axioms to prove: for every (classical?) definable triset S, there exists a value V such that [V contains/expanding V gives] exactly the elements of S.

TODO our model [X] is supposed to be a *candidate* model for ETST, not (necessarily) *the*, or the *standard* model of ETST. . It would appear more orderly to me to start with some proposed set of axioms of ETST, and then try to construct models satisfying those. However, (not only not having any particular axioms in mind, but also lacking a (three-valued) logic in which those axioms could be stated), we have chosen the opposite approach, and we've started with constructing a candidate model for which we can attempt to find suitable axioms (in a suitable logic). This does not mean other models couldn't be just as, or even more suitable for this purpose. . [other possible models: classical floor (including of other models, including repeatedly), s-pairs, positive vs all definable trisets, the "monotonic" models (thing an infinite-time-computable analogue of Scott-continuity)]

TODO rename Boolean Set Theory? The problem is, it is not a boolean algebra. So is the fact that it has union, intersection and "complement" (which is not complement according to the definition of boolean algebra) enough to justify calling it "boolean"? . TODO perhaps unfortunately, our set theory is "boolean" only in that the operations set union, intersection and complement are defined on any sets. These operations, however, do not form a boolean algebra, because for the complement, `a | ~a = 1` and `a & ~a = 0` do not hold. TODO should I pick a better name?

Two sanity conditions: 0. every definable collection is a member of the universal set (not true if our sets are the halting hamkins machines, bc an unhalting machine is a bordering element) 1. indeterminacy "distributes". eg. $A? = ?, A ?$

Is there a classical triset of pairs that can only be defined in a non-classical definition list?

https://www.irif.fr/ doumane/these.pdf completeness of park induction z esik https://logic.las.tu-berlin.de/Members/Kreutzer/Publications/lics02.pdf https://sci-hub.se/https://doi.org/10.1016/S0049-237X(08)71120-0 http://ndl.ethernet.edu.et/bitstream/1234

possible models of sets: hyperarithmetical/hyperelementary/inductive&coinductive (nope) globally stratifiably definable (nope – definitions that are not wouldn't have

a set to represent) locally stratifiably definable (nope?) classical and definable floor of definable . none of this works – ‘Ex s: Any: s nin s‘ must be classical, because ‘s‘ is classical in both context and background (being a bound variable), and thus causes paradoxes.