A cute and direct axiomatization of TST with finitely many templates

M. Randall Holmes

April 28, 2024

This axiomatization parallels the structure of first order logic exactly. It can be read as a reduction of the axiom scheme of comprehension of TST(U) to finitely many axiom templates (up to type assignment) or as a reduction of the axiom scheme of stratified comprehension to finitely many axioms. Probably one should assume weak extensionality: nonempty sets with the same elements are equal.

axiom of pairing: for any x, y, $\{z : z = z \vee z = y\}$ exists, and we call it $\{x, y\}$ We write $\{x\}$ for $\{x, x\}$. We also write $\iota(x)$ for $\{x\}$, and $\iota^1(x)$ for $\iota(x)$ and $\iota^{n+1}(x)$ for $\iota(\iota^n(x))$.

definition of the ordered pair: We write (x, y) for the ordered pair $\{\{x\}, \{x, y\}\}$.

domain of a relation: $\{x : (\exists y : (x, y) \in f)\}$, written $\texttt{dom}(f)$.

the intuitive idea: The underlying idea is that a formula $\phi(x_1, \ldots, x_n)$ is to be implemented by the set of functions $\{f : \phi(f(v_1), \ldots, f(v_n))\}$ where the objects v_n representing the variables are simply arbitrarily chosen objects of the theory.

the collection of all functions: $F = \{f : (\forall z \in f : (\exists xy : z = (x, y)) \wedge (\forall xyz : (x, y) \in f \wedge (x, z) \in f) \rightarrow y = z)\}$

merger of two functions: $f + g = \{(x, y) : f \in F \wedge g \in F \wedge ((x, y) \in f \wedge (x, y) \in g) \vee ((x, y) \in f \wedge x \not\in \texttt{dom}(g)) \vee ((x, y) \in g \wedge x \not\in \texttt{dom}(f))\}$

The notion of union proper to assignments of values to variables.

the collection of all predicates: $\Pi = \{P : (\forall fg \in P : f \in F \wedge g \in F \wedge \mathtt{dom}(f) = \mathtt{dom}(g))\}$

A predicate is simply a collection of functions all with the same domain (assignments of values to the same set of variables). It is intended that they have finite domain but there is no need to say so in the axiomatization.

definition (domain of a predicate): If P is a predicate, $\mathtt{Dom}(P)$ is defined as $\{x : (\exists f \in P : x \in \mathtt{dom}(f)\}$. We do not need an axiom to provide this: it is always either the domain of a function or the empty set, and both are already provided (you can merge functions badly to get the empty set).

complement of a predicate: $P^- = \{f : P \in \Pi \wedge f \in F \wedge \mathtt{dom}(f) = \mathtt{Dom}(P) \wedge f \notin P\}$

The domain of elements of the complement is the same as the domain of elements of the original predicate, and this domain is empty when the original predicate is empty.

union of predicates: $P \oplus Q = \{f : P \in \Pi \wedge Q \in \Pi \wedge f \in F \wedge (\forall x : x \in \mathtt{dom}(f) \leftrightarrow (x \in \mathtt{Dom}(P) \vee x \in \mathtt{Dom}(Q)) \wedge (\exists gh : (g \in P \vee g \in Q) \wedge g + h = f)\}$

The union of two predicates consists of the elements of each of the predicates minimally padded to have the same domain.

existential quantifier: $D_x(P) = \{f : f \in F \wedge P \in \Pi \wedge (\forall y : y \in \mathtt{dom}(f) \leftrightarrow y \in \mathtt{Dom}(P) \wedge y \neq x)) \wedge (\exists gh : g \in P \wedge \mathtt{dom}(h) = \{x\} \wedge f + h = g + h)\}$

This implements quantification over a "variable" x.

singleton image of a function: $f^\iota = \{(\{x\}, \{y\}) : (x, y) \in f\}$

Notice that the types of the "variables" have to change when this operation is applied. If one were actually to work with this much, I would suggest the convention that non-singletons be used in the first instance as "variables" so that one can always tell the intended type of the object actually being considered.

singleton image of a predicate: $P^{\iota+} = \{f^\iota : P \in \Pi \wedge f \in P\}$.

This allows any predicate defined on objects of lower type to be cast to iterated singletons in any higher type (by repeating this operation as required).

equality: $[=]_{x,y} = \{\{(x,z),(y,z)\} : x \neq y\}$

atomic inclusion: $[\in]_{x,y} = \{\{(x,\{z\}),(y,w)\} : x \neq y \wedge z \in w\}$

set from predicate: $\Sigma(P) = \{u : P \in \Pi \wedge (\exists x : \{(x,u)\} \in P)\}$

set union: $\iota^{-1+}(x) = \{y : \{y\} \in x\}$

The explicit assertions that domains of functions, the set of functions, and the set of predicates are sets are probably not needed: the uses of these are really as definitions of predicates. But it is more economical to say it this way, and these statements are consequences of the other axioms.

That this axiomatization works is extremely direct. Sets of assignments of values to variables are defined to represent predicates. Operations on predicates of complement and union are supported, so all boolean operations are supported. Domains are expanded as needed when unions are taken. Domains are contracted appropriately when quantifiers are applied. Equality and atomic inclusion are provided: atomic inclusion codes membership. All objects are cast into the same higher type using singleton image as necessary. Parameters can be supplied: $\{\{(x,a)\}\}$ codes the condition $x = a$, then intersection of predicates can be applied and the variable x can be quantified away. When down to one variable, the Σ operation allows the construction of a set, which may be at too high a type, but ι^{-1+} allows lowering of types to the intended type.

It is mildly weird that we use arbitrary objects in type theory to code variables. There is not an issue about having enough of them: even if type 0 is finite, a sufficiently high type will have as many variables as needed for any given set definition, and we have type lowering tools. It has the serious advantage that we never have to reorder variables: there is no role of any kind of converse operation in this axiom set.

It is better than Hailperin in that all lists of values to be assigned to variables live at the same type displacement from the values. The statement of the axioms are long, which might be construed as a disadvantage. Proving formally with enough metamathematics that it works should be straightforward, though.