Axioms for Mutually Recursive Free Type Definitions[*]

R.D. Arthan
Lemma 1 Ltd.
rda@lemma-one.com

1 July 1997

Abstract

This paper proposes a possible axiomatisation for mutually recursive free types for the draft Z standard.

1 Introduction

Ian Toyn has provided an excellent summary of his current thinking on the Z language as a whole in a recent paper [3]. This includes a description of the semantics of free types using the transformation rules (due to Ian and Sam Valentine). The axiomatisation given for the global variables introduced by a free type definition ingeniously avoids many of the dependencies on the toolkit that make the description in Spivey [2] unsuitable for use in the standard.

The present paper describes how the treatment in [3] may be generalised to provide an axiomatisation for mutually recursive free types — a language feature not currently supported in Z but widely considered to be highly desirable (e.g., for modelling the abstract syntax of programming languages). *En passant*, I describe a reworking of the axiomatisation that is slightly more economical of abstract syntax; this may help to address a concern mentioned in the list of known problems in [3].

[*]Author's reference: rda/file/30.tex; Z document reference: D-225; Issued: 17/6/97; Revised: 1/7/97

2 Review of Proposed Axiomatisation

With a few slight adjustments[1] to the notation, Ian Toyn's proposed axiomatisation is as follows:

Given a free type paragraph:

$$T ::= C_1 \mid \ldots \mid C_m \mid D_1 \ll e_1 \gg \mid \ldots \mid D_n \ll e_n \gg$$

where at least one of m and n is non-zero, the transformations derive the declarations:

$$\begin{array}{ll}
[T] & \\
C_j : T & 1 \leq j \leq m \\
D_k : \mathbb{P}(e_k \times T) & 1 \leq k \leq n
\end{array}$$

and the axioms:

$$\begin{array}{lrl}
\forall u : e_k \bullet \exists_1 y : D_k \bullet y.1 = u & 1 \leq k \leq n & \text{(Fnc)} \\
\forall u, v : e_k \mid D_k\, u = D_k\, v \bullet u = v & 1 \leq k \leq n & \text{(Inj)} \\
\forall y_1, y_2 : (1, C_1) \ldots (m, C_m) \mid y_1.2 = y_2.2 \bullet y_1.1 = y_2.2 & \text{(single axiom)} & \text{(Dsj1)} \\
\forall u : e_k \bullet \neg D_k\, u = C_j & 1 \leq j \leq m; 1 \leq k \leq n & \text{(Dsj2)} \\
\forall u : e_k; v : e_l \bullet \neg D_k\, u = D_l\, v & 1 \leq k, l \leq n & \text{(Dsj3)} \\
\forall A : \mathbb{P} T & \text{(single axiom)} & \text{(Ind)} \\
\mid \quad C_1 \in A \land \ldots \land C_m \in A \land & & \\
\quad\; (\forall x : (\mu T == A \bullet e_1) \bullet D_1\, x \in A) \land & & \\
\quad\quad \ldots & & \\
\quad\; (\forall x : (\mu T == A \bullet e_n) \bullet D_n\, x \in A) & & \\
\bullet \quad A = T & &
\end{array}$$

Here, (Fnc) together with the declarations say that each D_k is a function with domain e_k; (Inj) says that each function D_k is an injection; (Dsj1), (Dsj2) and (Dsj3) together say that the sets $\{C_1\}, \ldots, \{C_m\}$, $\mathsf{ran}\, D_1, \ldots, \mathsf{ran}\, D_n$ are pairwise disjoint; and (Ind) gives a principle of proof by induction.

For free types with many nullary constructors (large m) and many non-nullary constructors (large n), the axiomatisation has the disadvantage that (Dsj2) will produce a very large number $(m \times n)$ of individual axioms. This can be ameliorated by replacing (Dsj1), (Dsj2) and (Dsj3) with a portmanteau axiom along the lines of (Dsj1). Following the idea behind (Dsj1), what one would like to write is:

$$\forall y_1, y_2 : (1, C_1) \ldots (m, C_m) \cup \{z : D_1 \bullet (m+1, z.2)\} \cup \ldots \cup \{z : D_n \bullet (m+n, z.2)\} \mid y_1.2 = y_2.2 \bullet y_1.1 = y_2.2$$

[1] For clarity and and to simplify the subscripts later on, I have distinguished various names, in particular, the names of the nullary and non-nullary constructors and numbered the two sorts of constructor in separate sequences.

But, alas, the union operator is not available to us. The following axiom[2] (Dsj) isn't very pretty but does keep the size of the axioms linear in $m+n$:

$$\begin{array}{ll} \forall y_1, y_2 : \{i : \mathbb{N}; t : T \mid & \\ \qquad i = 1 \land t = C_1 \lor \ldots \lor i = m \land t = C_m \lor & \\ \qquad i = m+1 \land t \in \{d : D_1 \bullet d.2\} \lor \ldots \lor i = m+n \land t \in \{d : D_n \bullet d.2\}\} & \\ \mid \qquad y_1.2 = y_2.2 \bullet y_1.1 = y_2.2 & \text{(Dsj)} \end{array}$$

3 Mutual Recursion

The mutually recursive definitions to be considered simply comprise two or more "simultaneous" free type definitions. The concrete syntax used for these is not important here, but for definiteness, I will adopt the use of an '&' to join together the constituent free type definitions. The general case will therefore have the following form:

$$\begin{array}{l} T_1 ::= C_{11} \mid \ldots \mid C_{1m_1} \mid D_{11} \ll e_{11} \gg \mid \ldots D_{1n_1} \ll e_{1n_1} \gg \\ \quad \& \\ \quad \vdots \\ \quad \& \\ T_r ::= C_{r1} \mid \ldots \mid C_{rm_r} \mid D_{r1} \ll e_{r1} \gg \mid \ldots D_{rn_r} \ll e_{rn_r} \gg \end{array}$$

The type rules must arrange for the T_i to be in scope in each of the expressions e_{ij}. The semantics (via the transformation rules) must then require the T_i to provide a minimal solution to the above viewed as a system of simultaneous fixed point equations (in an appropriate sense of finding fixed points modulo a bijection, as discussed in [1]).

But what do we expect the defining properties for the least fixed point of such a system of equations to be? Clearly, we want the constructors, D_{ij} to be injective functions and we want the sets $\{C_{i1}\}, \ldots, \{C_{im_i}\}$, $\mathsf{ran}\, D_{i1}, \ldots, \mathsf{ran}\, D_{in_i}$ to be pairwise disjoint subsets of T_i for each i. So so far, there is no difference from the case of r independent free type definitions and the axioms (Fnc), (Inj), and (Dsj) (or (Dsj1) (Dsj2) and (Dsj3)) express our requirements nicely.

It is the induction axiom that fails to generalise so easily. Consider, for a concrete example, the following:

$$\begin{array}{c} \mathsf{YIN} ::= \mathsf{Yin} \ll \mathsf{YANG} \gg \\ \& \\ \mathsf{YANG} ::= \mathsf{Yang} \ll \mathsf{YIN} \gg \end{array}$$

It is not hard to see that any two sets of the same cardinality will provide a fixed point model for the above definition (with the constructor functions corresponding to bijections between the two sets showing they have the same cardinality). So, for example, a model might take $\mathsf{YIN} = 1\mathrel{..}10$ and $\mathsf{YANG} = 11\mathrel{..}20$ with $\mathsf{Yang}(i) = i + 10$ and $\mathsf{Yin}(j) = j - 10$ ($1 \leq i \leq 10$, $11 \leq j \leq 20$).

The least fixed point solution that we want actually has both YIN and YANG empty.[3] Unfortunately, the appropriate instances of (Ind) do not characterise the desired minimal solution. Indeed, after simplifying the μ-term, the two axioms become:

[2]In this and other axiom schemata, formulae such as $m+1$ and $m+n$ are to be read as the numeric literals obtained by carrying out the addition rather than as object language formulae.

[3]Depending on our views on whether given sets are allowed to be empty and on some other rather more obscure issues, we might prefer to insist on non-empty solutions, in which case we would take two singleton sets for the preferred model.

$$\forall A : \mathbb{P}\mathsf{YIN} \mid (\forall x : \mathsf{YANG} \bullet \mathsf{Yin}(x) \in A) \bullet A = \mathsf{YIN}$$
$$\forall A : \mathbb{P}\mathsf{YANG} \mid (\forall x : \mathsf{YIN} \bullet \mathsf{Yang}(x) \in A) \bullet A = \mathsf{YANG}$$

Unfortunately, all that these axioms tell us is that the constructor functions are surjective; this is insufficient to characterise the solution of interest. Indeed, it is not surprising that the induction principle (Ind) for a single free type definition is too weak in this case, since when we view the two free type definitions separately, they are not recursive.

The required axiom must correspond to a principle of proof by induction of a property of each of the component free type definitions simultanously. Using the style of the proposed axiom for a single free type defintion (Ind), we can capture this in the following axiom (MutInd):

$$\begin{array}{ll}
\forall A_1 : \mathbb{P}T_1; \ldots ; A_r : \mathbb{P}T_r & \text{(MutInd)} \\
\mid \quad C_{11} \in A_1 \wedge \ldots \wedge C_{1m_1} \in A_1 \wedge & \\
\qquad \ldots & \\
\quad C_{r1} \in A_1 \wedge \ldots \wedge C_{rm_r} \in A_r \wedge & \\
\quad (\forall x : (\mu T_1 == A_1; \ldots ; T_r == A_r \bullet e_{11}) \bullet D_{11}\, x \in A_1) \wedge \ldots & \\
\qquad\qquad \ldots \wedge (\forall x : (\mu T_1 == A_1; \ldots ; T_r == A_r \bullet e_{1n_1}) \bullet D_{1n_1}\, x \in A_1) \wedge & \\
\qquad \vdots & \\
\quad (\forall x : (\mu T_1 == A_1; \ldots ; T_r == A_r \bullet e_{r1}) \bullet D_{r1}\, x \in A_r) \wedge \ldots & \\
\qquad\qquad \ldots \wedge (\forall x : (\mu T_1 == A_1; \ldots ; T_r == A_r \bullet e_{rn_r}) \bullet D_{rn_r}\, x \in A_r) & \\
\bullet \quad A_1 = T_1 \wedge \ldots \wedge A_r = T_r &
\end{array}$$

For the YIN-and-YANG example, this gives us the following:

$$\begin{array}{l}
\forall A_1 : \mathbb{P}\mathsf{YIN}; A_2 : \mathbb{P}\mathsf{YANG} \\
\mid \quad (\forall x : (\mu \mathsf{YIN} == A_1; \mathsf{YANG} == A_2 \bullet \mathsf{YANG}) \bullet \mathsf{Yin}\, x \in A_1) \wedge \\
\quad (\forall x : (\mu \mathsf{YIN} == A_1; \mathsf{YANG} == A_2 \bullet \mathsf{YIN}) \bullet \mathsf{Yang}\, x \in A_2) \\
\bullet \quad A_1 = YIN \wedge A_2 = YANG
\end{array}$$

It remains to assess whether the axioms proposed here and in [3] are the right ones to use. An evident economy would be to simplify away the μ-expressions. To check correctness of detail and fitness for purpose one can consider examples and one can work through proofs of specific properties in particular cases and of general meta-theoretic properties. This can perhaps wait until the general approach of [3] and of this paper has been reviewed and the details agreed.

4 Summary

An induction principle (MutInd) has been proposed generalising the axiomatisation for free types in [3] to the mutually recursive case. A minor variant to the treatment in [3] has also been proposed to make the complexity of the axiomatisation linear, rather than quadratic, with respect to the size of the free type definition.

References

[1] R.D. Arthan. On free type definitions in z. In J.E. Nicholls, editor, *Z User Workshop, York 1991*. Springer-Verlag, 1992.

[2] J.M. Spivey. *The Z Notation: A Reference Manual, Second Edition.* Prentice-Hall, 1992.

[3] Ian Toyn. Z notation. Technical report, University of York, 2nd May 1997.