Correctness Criteria for the Animation of Z Specifications via a Logic Programming Language

by

Dr. Margaret M West,

University of Huddersfield,

Queensway, HD1 4DH, UK

email: M.M.West@hud.ac.uk,

WWW home page: `http://scom.hud.ac.uk/scommmw`

ICLP, Porto, Portugal, September 2007

Z is a formal specification notation based on set theory and first order predicate logic. It is used by industry as part of the software (and hardware) development process in Europe, USA and elsewhere. It has recently undergone international ISO standardisation.

The use of mathematics in modelling means that formal reasoning can then be applied to check the consistency of the specification. Further, a version of the specification can be executed or 'animated' in order to demonstrate its functionality and detect flaws.

I have previously used *structure simulation* a technique developed by myself to animate Z using a logic programming language. The method has been applied to both Prolog and to the Gödel logic programming language - Gödel is preferable as it has sets. Breuer and Bowen (1994) developed correctness criteria for the animation of Z - which they call

Abstract Approximation.

The interpretation of Z syntactical objects in both the execution language (in our case the LP) and in Z are compared. For correctness, the interpretation in the LP domain must always *abstract* (underestimate) the interpretation in the Z domain.

It is more common for a *program* to refine a specification rather than the converse. The reason for this more unusual approach: an animation of a Z specification must not contain any more information than the original Z specification, as it may mislead.

Abstract Approximation - in brief

ϵ is a syntactic Z expression.

ρ_{LP}, ρ_Z are environments.

Function γ relates the abstract with the concrete so that

$\rho_Z = \gamma \circ \rho_{LP}$.

The abstract (programming) interpretation of Z syntax is denoted:

$\mathcal{E}_{LP}[\![\ldots]\!]\rho_{LP}$ and the Z interpretation is denoted: $\mathcal{E}_Z[\![\ldots]\!]\rho_Z$:

Criteria for Correctness

$$\gamma(\mathcal{E}_{LP}[\![\epsilon]\!]\rho_{LP}) \sqsubseteq \mathcal{E}_Z[\![\epsilon]\!](\gamma \circ \rho_{LP}).$$

The method has been adapted so that structure simulation has been proved to be a correct animation.

Example - data values and predicates Both domains are extended by the inclusion of a '$\perp$' element *for each type*.

$$a \sqsubseteq b \Leftrightarrow (a = \perp \ or \ a = b).$$

E.g. (1) Integer overflow results in integer variable 'n' remaining undefined in the LP where its interpretation in Z results in a value, say m.

Suppose $\perp^N_{LP}$ is the 'bottom' element for the integer type in the LP and similarly for Z.

$\gamma(\perp^N_{LP}) = \perp^N_Z$ so $\gamma(\perp^N_{LP}) \sqsubseteq m$.

E.g. (2) A program flounders - denoted by $\perp^P_{LP}$ the 'bottom' element for the boolean type. If its Z equivalent results in either 'true' or 'false', then the LP interpretation underestimates the Z.

Further Work

- automate 'structure simulation' so it can be applied to other Case Studies.

- investigate other implementations of 'structure simulation', since the method and proof is generic to LP languages.