The model theory of specification

Wilfrid Hodges
Queen Mary, University of London
April 2006
www.maths.qmul.ac.uk/∼wilfrid/sauro.pdf

2

Sauro Tulipani and me

8 November 1983: S writes to W asking to visit Bedford College London.

Summer 1984: Bedford College closes down.

1984/5: Correspondence between S and W influences 'Building Models by Games'.

Autumn 1985: S visits W at Queen Mary and Westfield College.

February 1990: S speaks to seminar at Queen Mary on 'The first order theory of infinite terms'.

3

Some work I was doing when Sauro visited

A specification is

- a *formal theory*
- which *defines*
- the intended *behaviour* of a system (hardware, software or both).

So it's a species of definition.

4

A paradigm popular in the 1980s:

> The behaviour of a system is a structure A.
>
> The elements of A are the items of data handled by the system.
>
> Functions of A represent the response of the system to data items.
>
> In general the specification is *parametrised*, i.e. the system is a function $S(B, \dots)$ of one or more previously specified systems $B, \dots$.

5

Relations of the structure represent
tests performed on the data by the system.

$$\texttt{append(textline,symbol)} = \begin{cases} \texttt{textline}^\frown\texttt{symbol} & \text{if } lh(\texttt{textline}) < \texttt{LINELENGTH} \\ \texttt{textline}^\frown\texttt{LF}^\frown\texttt{symbol} & \text{otherwise} \end{cases}$$

6

This paradigm looks rather old today:

- One should distinguish the ports through which the system interacts with other systems.
  (Cf. Robin Milner passim.)
- The behaviour should develop through time in response to the behaviours of other systems.
  (Some people advocate games for studying this.)

But it's still widely used.

7

A specification is a theory T that determines a single model (up to isomorphism) in terms of a given structure.

So in some sense T is 'relatively categorical'.

Logicians have proposed using for specifications:

(a) first-order theories that are (say) ω-categorical;

(b) sentences of $L_{\omega_1,\omega}$.

Serious problems with both.

8

Two commercially successful answers:

I. A specification is a finite universal Horn (first-order) theory T. The specified structure $S(B)$ is the unique initial model of the set of atomic consequences of

$$T \cup \text{diagram}^+(B).$$

This paradigm yields *algebraic specification*,
as in ACT TWO or CASL.
There is lots of universal algebra to build on.

9

II. A specification is a Σ_1^+ set-theoretic formula $\phi(x, y)$ such that for each parameter structure B there is a unique structure $S(B)$ satisfying $\phi(B, S(B))$.

This paradigm yields *set-theoretic specification* as in the languages VDM, Z and B.

10

Problem for the theorist:

Here are two formalisations of the same intuitive notion. Do they agree (and in what sense)?

11

Observation 1 (due to Peter Aczel for VDM):

> In both algebraic and set-theoretic specification, a homomorphism $h : B \to B'$ induces a homomorphism $S(h) : S(B) \to S(B')$.

(For set-theoretic specification, this is a consequence of the Σ_1^+ form.)

Thus the 'model' of the specification is in fact a functor on structures, say from finite signature Σ to finite signature Ω.

12

Observation 2: For algebraic specification, we can characterise the functors S that occur, up to natural equivalence, as follows (the 'specification functors').

> There are a finite signature Π expanding Ω and Σ, a functor $F : \Sigma\mathbf{Str} \to \Pi\mathbf{Str}$ and a functor $G : \Pi\mathbf{Str} \to \Omega\mathbf{Str}$ such that
>
> - $S = GF$,
> - F is the left adjoint of a relativised reduct functor,
> - G is a relativised reduct functor.

13

Every functor of this type is (up to natural equivalence) expressible in any of the standard set-theoretic specification formalisms.

The converse fails, because of apparently essential use of infinite sets in some specifications.

But inspection of a range of published Z specifications produced few examples that don't define specification functors.

14

Remarks:

- Somehow computability is completely hidden in the definition of a specification functor. But it's there. A characterisation of computability by Smullyan and Fitting accounts for this.
- Anatoliĭ Mal'tsev studied functors of this kind in an algebraic context under the name 'subreplicas'. Example: tensor product of a pair of abelian groups.

15

Remarks continued:

- The proofs yield effective translations between algebraic specifications and (restricted) set-theoretic specifications.

  The translation from set-theoretic specification to algebraic specification was rediscovered by Hélène Kirchner and Peter Mosses, published 2001, who claim it as a 'novelty'. (More below.)

16

Specification functors are no good as a semantics for set-theoretic specification, for two reasons:

(a) They ignore the infinite sets.

(b) They ignore types (central in these formalisms).

The official semantics of Z uses relation algebras. Martin Henson and Steve Reeves have an alternative approach through 'Z logic', a form of type theory. (It revises Z.)

17

Ray Turner, 'The foundations of specification' (2005):

> '... Hodges has developed an approach to specification based upon hereditarily finite set theory. But he does not really consider the impact of types; the present theory may be considered as an attempt to explicitly put types into hereditarily finite set theory — with all that that entails (e.g. the need to take products as primitive, etc.).'

18

- Turner's languages have a type structure, including type variables and some fixed type constructors, e.g. forming cartesian products and their projections. (Since the types used are Σ_1^+-definable, I suspect the functoriality lifts to this setting, but I haven't checked.)
- The infinite sets used in Z become types. (I believe this restricts the language, though much less than I restricted it.)

19

- The clauses of a specification become in general polymorphic. (I.e. they are ambiguous about the types involved.)
- Chiefly because of polymorphism, the axioms required to justify a specification are more complicated. (I gave a simple set of axioms for the hereditarily finite sets over a given structure.)

20

Turner has greatly improved my treatment of set-theoretic specifications, but not extended the comparison with algebraic specifications.

BUT Kirchner and Mosses (above) do essentially this. Their novelty is not the translation of set theory into universal Horn sentences, but

> 'Types may be polymorphic, and include abstract types as well as the concrete set-theoretical product, power-set, and function types.'

(Actually they overreach. The 'power set' is a countable subset of the power set, as in work of Karl Meinke.)

21

There is much tidying up to do.

Probably I'm not the person to do it.

But if I try, it will be with Sauro in mind.

22

References

- Melvin Fitting, *Computability Theory, Semantics, and Logic Programming*, Oxford University Press, New York 1987.
- Wilfrid Hodges, 'The meaning of specifications I: Initial models,' *Theoretical Computer Science* 152 (1995) 67–89.
- Wilfrid Hodges, 'The meaning of specifications II: Set-theoretic specification', *Semantics of Programming Languages and Model Theory*, ed. Droste and Gurevich, Gordon and Breach, Yverdon 1993, pp. 43–68.

23

- Hélène Kirchner and Peter Mosses, 'Algebraic specifications, higher-order types and set-theoretic models', *Journal of Logic and Computation* 11 (2001) 453–481.
- G. Marongiu and Sauro Tulipani, 'Quantifier elimination for infinite terms', *Archive for Mathematical Logic* 31 (1991) 1–17.
- Raymond Turner, 'The foundations of specification', *Journal of Logic and Computation* 15 (2005) 623–662.