THE FOUNDATIONS OF SPECIFICATION II

Raymond Turner
The University of Essex

February 13, 2004

Abstract

In the *Foundations of Specification I*, we developed a *Specification theory* (**CST**) to serve as a vehicle to explore the mathematical foundations of specification. A further aspect of this concerns *type inference*. In this paper we develop and explore a *type inference* system for **CST** and use it to illustrate the foundational issues which arise with such systems.

1 The Grammatical Role of Types.

Type membership is embedded in the actual theory **CST** and, consequently, types play an essentially mathematical role: they carve up the universe of objects into different kinds and our definitions and proofs are subject to these classifications. In this role they serve much the same function as they do in mathematics: they provide conceptual organization to the theory and its application. However, in computer science, types also have a *grammatical* function. This aspect is glossed over, or rather ignored, in **CST**. To see what this function amounts to consider the following simple wff.

$$(a) \qquad x : N \wedge \exists y : N \cdot x \in y$$
$$(b) \qquad x : N \wedge \exists y : Set(N) \cdot x = y$$
$$(c) \qquad x : N \wedge \exists y : N \cdot x \leq y$$

Despite the fact that in the language of **CST** they are all syntactically legitimate, there is a difference between them: the last is *well-typed* whereas the first two are not. Presumably, for them to be so, in the first y should be of type $Set(N)$ and in the second of type N.

To deal with this, specification languages come equipped with some associated *type-checking* tools. Most often such a tool is supplied by a hidden program ([10], [12], [5]). However, in some cases (e.g. [13], [2]) it takes the more explicit form of a *type inference system;* much like those of the Lambda calculus ([1]). However, these systems are rarely properly formulated as logical systems and, as a consequence, even their elementary properties are not documented. In this paper we develop and explore such a system for **CST**. The investigation should serve as a template for the formulation and exploration of such systems.

2 The System T

The objective of the type inference system is to provide rules which determine whether wff, and ultimately *specifications*, are *well-typed*. The system (**T**) has two has two grammatical judgements

$$t : T$$
$$\phi \; prop$$

The first states that the term t has type T and the second that a wff is *well-typed*. We shall use Θ for a judgement of either of these two forms. Judgements are made relative to a declaration c which is here understood as a (possibly empty) set of assumptions

$$x_1 : T_1, ..., x_m : T_m$$

where no individual variable is assigned more than one type. We write $\alpha(c)$ for the set of declared individual variables of a declaration c. We write $c \subseteq c'$ to indicate that c' is a consistent super-set of c' and, where V is a set of variables, we shall write $c \restriction V$ to denote c restricted to the variables in V. We shall say that a context c *covers* an expression e if $FV(e) \subseteq \alpha(c)$. Finally, we shall write $c, x : T$ for the context c updated with the assignment $x : T$ i.e. if $x \notin \alpha(c)$ then $x : T$ is added whereas, if $x \in \alpha(c)$, its type is replaced by T.

The system is determined by the following axioms and rules. We shall suppress the declaration context unless the rule effects it.

There are two *structural rules*: *assumption* and *weakening*.

$$\mathbf{Ax} \quad x : T \vdash x : T \qquad\qquad \mathbf{W} \quad \frac{c \vdash \Theta}{c, x : T \vdash \Theta} \; x \notin \alpha(c) \cup FV(\Theta)$$

For each built-in relation symbol, we associate a sequent that determines the conditions under which it is a proposition. The actual cases for the relations of our theory are given as follows; they reflect the above demands about membership, equality and the numerical ordering relation.

$$\mathbf{A_1} \qquad x : T, y : Set(T) \vdash x \in y \; prop$$
$$\mathbf{A_2} \qquad x : T, y : T \vdash x = y \; prop$$
$$\mathbf{A_3} \qquad x : N, y : N \vdash x < y \; prop$$

For each of the logical connectives there is a rule of formation that lays out the conditions under which complex wff are *well-typed*. Most of these are self evident but notice that the rules for the implication, conjunction and disjunction are identical. Similarly, the rules for universal and existential quantification are identical. This reflects the weaker judgement involved i.e. *being a proposition* versus *truth*.

$$\mathbf{T_1}\quad \Omega\ prop \qquad \mathbf{T_2}\quad \frac{\phi\ prop}{\neg\phi\ prop} \qquad \mathbf{T_3}\quad \frac{\phi\ prop \quad \psi\ \ prop}{\phi \to \psi\ \ prop}$$

$$\mathbf{T_4}\quad \frac{\phi\ \ prop \quad \psi\ \ prop}{\phi \vee \psi\ \ prop} \qquad \mathbf{T_5}\quad \frac{\phi\ \ prop \quad \psi\ \ prop}{\phi \wedge \psi\ \ prop}$$

$$\mathbf{T_6}\quad \frac{c, x : T \vdash \phi\ \ prop}{c \vdash \forall x : T \cdot \phi\ \ prop} \qquad \mathbf{T_7}\quad \frac{c, x : T \vdash \phi\ \ prop}{c \vdash \exists x : T \cdot \phi\ \ prop}$$

$$\mathbf{T_8}\quad \frac{c \vdash \phi[X]\ \ prop}{c \vdash \forall X \cdot \phi\ \ prop} \qquad \mathbf{T_9}\quad \frac{c \vdash \phi[X]\ \ prop}{c \vdash \exists X \cdot \phi\ \ prop}$$

In the rules T_8 and T_9, $X \notin FTV(c)$. The rules for the bound quantifiers are given as follows. Notice that the premise requires that the wff be a proposition in the appropriate context.

$$\mathbf{T}_{10}\ \frac{c, x : N \vdash \phi\ \ prop \qquad c \vdash n : N}{c \vdash \forall x < n \cdot \phi\ \ prop} \qquad \mathbf{T_{11}}\ \frac{c, x : N \vdash \phi\ \ prop \quad c \vdash n : N}{c \vdash \exists x < n \cdot \phi\ \ prop}$$

$$\mathbf{T_{12}}\ \frac{c, x : T \vdash \phi\ \ prop \qquad c \vdash s : Set(T)}{c \vdash \forall x \in s \cdot \phi\ \ prop} \qquad \mathbf{T_{13}}\ \frac{c, x : T \vdash \phi\ \ prop \quad c \vdash s : Set(T)}{c \vdash \exists x \in s \cdot \phi\ \ prop}$$

Almost finally, we provide the rules for the judgement $t : T$. These are quite obvious and should not cause one to pause since they are the closure conditions for our types. Indeed, we give them the same names as the corresponding axioms of **CST**.

$$\begin{array}{ll}
\mathbf{N_1} & 0 : N \\
\mathbf{N_2} & x : N \vdash x^+ : N \\
\mathbf{S_1} & \emptyset : Set(X) \\
\mathbf{S_2} & x : X, y : Set(X) \vdash x \circledast y : Set(X) \\
\mathbf{P_1} & x : X, y : Y \vdash (x, y) : X \otimes Y \\
\mathbf{P_2} & z : X \otimes Y \vdash \pi_1(z) : X \\
\mathbf{P}_3 & z : X \otimes Y \vdash \pi_2(z) : Y
\end{array}$$

Finally, since we have no *logical* quantification rules we require substitution rules.

$$\mathbf{Sub}\ \frac{c, x : T \vdash \Theta \qquad c \vdash t : T}{c \vdash \Theta[t/x]} \qquad \mathbf{SUB}\ \frac{c[X] \vdash \Theta[X]}{c[T/X] \vdash \Theta[T/X]}$$

This completes the rules of the system **T**. It should be clear enough how we use it but we provide a simple example anyway.

Example 1 *Given the axiom A_1 for membership and the conjunction rule (T_5) we have:*

$$x : X, y : X, z : Set(X) \vdash x \in z \wedge y \in z \; prop$$

By the axiom A_2 for equality:

$$x : X, y : X, u : X \vdash u = x \vee u = y$$

By the rule for set quantification

$$x : X, y : X, z : Set(X) \vdash \forall u \in z \cdot u = x \vee u = y \; prop$$

Finally, by the conjunction rule

$$x : X, y : X, z : Set(X) \vdash x \in z \wedge y \in z \wedge \forall u \in z \cdot u = x \vee u = y \; prop$$

2.1 Properties of T

There are some immediate properties of the system which parallel the standard results for any well behaved system of type inference. The following parallel the results for the type systems for the lambda calculus [1].

Proposition 2 *In the system* ***T***

1. If $c \vdash \Theta$ then c covers Θ
2. If $c \vdash \Theta$ then $c \restriction FV(\Theta) \vdash \Theta$
3. If $c \vdash \Theta$ and $c \subseteq c'$ then $c' \vdash \Theta$

Proof. All are by induction on the derivations. We illustrate with some characteristic cases. Consider part (1). The structural rules are automatic: given that c covers the premise it covers the conclusion. Next consider the type quantifier rule.

$$\frac{c, x : T \vdash \phi \; prop}{c \vdash \forall x : T \cdot \phi \; prop}$$

If $c, x : T$ covers the premise then c covers the conclusion. For part (2) we again illustrate with the quantifier rule. By induction,

$$(c, x : T) \restriction FV(\phi) \vdash \phi \; prop.$$

Hence, regardless of whether $x \in FV(\phi)$, we have:

$$c \restriction FV(\forall x : T \cdot \phi), x : T \vdash \phi \; prop$$

Hence, by the rule,

$$c \restriction FV(\forall x : T \cdot \phi) \vdash \forall x : T \cdot \phi \; prop$$

For part (3) we again illustrate with the quantifier rule. Suppose

$$c \vdash \forall x : T \cdot \phi \; prop$$

follows from $c, x : T \vdash \phi \; prop$. By induction, for $c' \supseteq c$, $c', x : T \vdash \phi \; prop$. Hence, $c' \vdash \forall x : T \cdot \phi \; prop$. ■

The next is the standard generation lemma that provides the means of automatically checking that an expression is well-typed by using the system backwards.

Proposition 3 *(Generation)*

1. If $c \vdash t_1 = t_2 \; prop$ then $c \vdash t_1 : T$ for some type T. Moreover, $c \vdash t_1 : T$ iff $c \vdash t_2 : T$
2. If $c \vdash t \in s \; prop$ then $c \vdash t : T$ and $c \vdash s : Set(T)$ for some T
3. If $c \vdash t < s \; prop$ then $c \vdash t : N$ and $c \vdash s : N$
4. If $c \vdash \neg\phi \; prop$ then $c \vdash \phi \; prop$
5. If $c \vdash \phi \circ \psi \; prop$ then $c \vdash \phi \; prop$ and $c \vdash \psi \; prop$ $(\circ = \vee, \wedge, \rightarrow)$
6. If $c \vdash Qx : T \cdot \phi \; prop$ then $c, x : T \vdash \phi \; prop$ $(Q = \exists, \forall)$
7. If $c \vdash Qx < t \cdot \phi \; prop$ then $c \vdash t : N$ *and* $x : N \vdash \phi \; prop$ $(Q = \exists, \forall)$
8. If $c \vdash Qx \in t \cdot \phi \; prop$ then $c \vdash t : Set(T)$ for some T and $x : T \vdash \phi \; prop$ $(Q = \exists, \forall)$
9. If $c \vdash QX \cdot \phi \; prop$ then $c \vdash \phi[X] \; prop$ $(Q = \exists, \forall)$
10. If $c \vdash t^+ : N$ then $c \vdash t : N$
11. If $c \vdash (t_1, t_2) : T_1 \otimes T_2$ then $c \vdash t_1 : T_1$ and $c \vdash t_2 : T_2$
12. If $c \vdash \pi_1(t) : T_1$ then $c \vdash t : T_1 \otimes T_2$ for some T_2
13. If $c \vdash a \circledast b : Set(T)$ then $c \vdash a : T$ and $c \vdash b : Set(T)$ for some T

Proof. By induction on the derivations. Each of the antecedents of the above conditionals can only be result of a rule application whose premises are the consequents of the corresponding conditional or the result of a substitution or the application of a structural rule. The first group of cases are immediate. For the term substitution rules we illustrate with the first and the following instance. Suppose that

$$c \vdash a[t] = b[t] \; prop$$

follows from Sub. The premises yield

$$c, x : T \vdash a[x] = b[x] \; prop \qquad c \vdash t : T$$

Then by induction,

$$c, x : T \vdash a[x] : S$$

for some type S. We now employ the substitution rule itself. The structural rules are easy to verify for all the cases. The following derivation involving the negation instance of the first rule, illustrates the argument. Suppose that

$$c, x : T \vdash \neg\phi \; prop$$

follows by the rule

$$\frac{c \vdash \neg\phi \; prop}{c, x : T \vdash \neg\phi \; prop}$$

Then by induction $c \vdash \phi \; prop$ and so by the structural rule itself $c, x : T \vdash \phi \; prop$. ■

We also have the following admissible rule.

Proposition 4 *(Back Substitution)*

(1) *If* $c \vdash \Theta[t/x]$ *then* $c \vdash t : T$ *and* $c, x : T \vdash \Theta[x]$ *for some type* T
(2) *If* $c \vdash \Theta[t/x]$ *and* $c \vdash t : T$ *then* $c, x : T \vdash \Theta[x]$

Proof. Both parts are by simultaneous induction on the derivations. We consider the first. We illustrate with some significant cases. Suppose that

$$c \vdash a[t/x] \in b[t/x] \; prop$$

Then by the generation lemma, for some A,

$$c \vdash a[t/x] : A \text{ and } c \vdash b[t/x] : Set(A)$$

By induction, for some type T, we have:

$$c, x : T \vdash a[x] : A \text{ and } c, x : T \vdash b[x] : Set(A) \text{ and } c \vdash t : T$$

so we are finished by the membership axiom. Next assume that

$$c \vdash (\phi \wedge \psi)[t/x] \; prop$$

then

$$c \vdash \phi[t/x] \; prop \text{ and } c \vdash \psi[t/x] \; prop$$

Hence, by induction,

$$c, x : T \vdash \phi \; prop \text{ and } c, x : T \vdash \psi \; prop \text{ and } c \vdash t : T$$

and we are finished by the conjunction rule. Now consider part two. We again illustrate with the following case. Suppose that

$$c \vdash a[t/x] \in b[t/x] \; prop \; and \; c \vdash t : T$$

Then by the generation lemma, for some A,

$$c \vdash a[t/x] : A \text{ and } c \vdash b[t/x] : Set(A)$$

By induction, we have:

$$c, x : T \vdash a[x] : A \text{ and } c, x : T \vdash b[x] : Set(A)$$

so we are finished by the membership axiom. ■

2.2 CST and T

The following informs us that any judgement of the form $t : T$ that is provable in the type system, is also provable in the logic. This is the first step in charting the relationship between the two systems.

Proposition 5 *If* $c \vdash_{\mathbf{T}} t : T$ *then* $c \vdash_{\mathbf{CST}} t : T$

Proof. By induction on the derivations in **T**. The structural rules are clear. But then so are the main body of rules with these conclusions: they follow from the corresponding logical rules. The substitution rules also follow directly from the rules of the logic.■

The converse is not provable. This marks a difference between the *grammatical* and *logical* roles of types. Logically, the type membership assertion forms part of the logic of the system and, in particular, in the logical theory type membership is not decidable since the logic is not. However, it is in the type system. These two systems capture the exact difference between these two roles and pinpoints the fact that the computational properties of types emerges from the theory i.e. **T** versus **CST**.

3 Specifications

The principal application of **T** is to type-check specifications. To facilitate this we must introduce a rule for schema specifications that allows wff involving the new relation or function symbol, to be checked.

3.1 Relations

Relation Specifications are treated in a similar fashion to the other atomic wff but now added as a rule with the predicate of the specification supplying the premise. Given a specification

$$R[X_1, ..., X_m] \triangleq [x_1 : T_1, ..., x_n : T_n \mid \phi]$$

we extend the system with the rule

$$\frac{x_1 : T_1, ..., x_n : T_n \vdash \phi[X_1, ..., X_m, x_1, ..., x_n]\ prop}{x_1 : T_1, ..., x_n : T_n \vdash R[X_1, ..., X_m](x_1, ..., x_n)\ prop} \qquad (\mathbf{ST})$$

Call this system **T+ST**. It is then easy to show, by induction on the derivations, that:

Proposition 6 *If* $c \vdash_{\mathbf{T+ST}} \Theta$ *then* $c \vdash \Theta^*$ *where* Θ^* *is obtained from* Θ *by replacing each occurrence of* $R[X_1, ..., X_m](x_1, ..., x_n)$ *by* $\phi[X_1, ..., X_m, x_1, ..., x_n]$

So, in the obvious sense, this addition to the type inference system is conservative. We can also easily establish the following.

Proposition 7 *The following rules are derivable*

1. $x : X, y : X, z : Set(X) \vdash Pair_X(x, y, z)\ prop$
2. $x : Set(X), y : Set(X), z : Set(X) \vdash Union_X(x, y, z)\ prop$
3. $x : Set(Set(X)), y : Set(X) \vdash Genunion_X(x, y)\ prop$
4. $x : Set(X), y : Set(Set(X)) \vdash Pow_X(x, y)\ prop$
5. $$\frac{y : Set(X) \vdash \phi[y]\ prop}{x : Set(X) \vdash \{y \in x \cdot \phi\} : Set(X)}$$

Proof. We illustrate with second. According to the new rule for the specification it is sufficient to show that

$$x : Set(X), y : Set(X), z : Set(X) \vdash \forall u : X \cdot u \in x \leftrightarrow u \in y \vee u \in z\ prop$$

and this is straightforward.■

Much the same is true if we wish to type-check relations which are type independent. Here we add the rule

$$\frac{x_1 : T_1, ..., x_n : T_n \vdash \phi[x_1, ..., x_n]\ prop}{x_1 : T_1, ..., x_n : T_n \vdash R(x_1, ..., x_n)\ prop}$$

The conservative extension result follows the same route as the general case.

3.2 Functions

The situation with the addition of new function symbols is similar to the basic operations of the theory except that we need to ensure they are well-typed. Suppose that we have legitimately introduced a new function symbol via the specification

$$F[X] \triangleq_{Pfun} [x : I[X], y : O[X] \mid \psi[X, x, y]]$$

Then we introduce a new type inference rule

$$\frac{x : I[X], y : O[X] \vdash \psi[X, x, y]\ prop}{x : I[X] \vdash F[X](x) : O[X]\ prop}\ FT$$

Call this system **T+FT**. Once more this addition is conservative. In the following $*$ is the translation from the system with a new function symbol to the system without (see [15])

Proposition 8 *If* $c \vdash_{\mathbf{T+FT}} \Theta$ *then* $c \vdash \Theta^*$ *where* Θ^* *is obtained from* Θ *by replacing every wff by its translation.*

Proof. We illustrate with the total case; the partial one causes no additional complications. We proceed as in the original proof by putting all wff in normal form and then replace each occurrence of $F[X](x) : O[X]$ by $x : I[X] \wedge \exists y : O[X] \cdot \psi[X, x, y]$. We have then only to show that the following is derivable

$$\frac{x : I[X], y : O[X] \vdash \psi[X, x, y]\ prop}{x : I[X] \vdash \exists y : O[X] \cdot \psi[X, x, y]\ prop}$$

which is clear.■

Observe that, given the definition of type membership, by the generation lemma, we have that the following is derivable.

$$If\ c \vdash t : T\ prop\ then\ \ c \vdash t : T$$

Hence, given FT, the following is admissable

$$\frac{x : I[X], y : O[X] \vdash \psi[X, x, y]\ prop}{x : I[X] \vdash F[X](x) : O[X]}$$

It follows that the following are derivable

$$\frac{a : Set(Set(X))}{\cup_X a : Set(X)} \qquad \frac{a : Set(X)}{Set_X(a) : Set(Set(X))}$$

Finally, we may drop the type variables on the function symbol when it has been shown to be type independent. For example, we then have the following derived rules for Generalised union and power set.

$$\frac{a : Set(Set(X))}{\cup a : Set(X)} \qquad \frac{a : Set(X)}{Set(a) : Set(Set(X))}$$

4 A Strongly Typed Theory

Is **CST** a typed theory? Certainly it has types and objects in the theory have them. However, we have observed that not every wff provable in **CST** is well-typed. This follows from our preliminary discussion on the relationship between **T** and **CST**. Hence, **CST** is not a *typed theory* in the traditional sense of Higher order logic (**HOL**) where only *well-typed* wff are provable. In this section we develop a version of **CST** ($\mathbf{CST}_T$) that is typed in this more traditional sense. This will enable us to fully explore the relationship between **CST** and **T**. In particular, we shall show that **CST** is a conservative extension of $\mathbf{CST_T}$ - with respect to well-typed expressions. This will throw some mathematical light on the rather curious and somewhat murky relationship between *type* and *logical inference* that has emerged from the development of specification languages.

4.1 The Theory $\mathbf{CST}_T$

This is a marriage of **CST** and **T**. Consequently, the theory now has three judgements which combine those of **CST** and **T**.

$$t : T$$
$$\phi \; prop$$
$$\phi$$

We use Θ for a judgment of either kind. In particular, and this is important, we now drop the definition

$$t : T \triangleq \exists x : T \cdot x = t$$

Type membership is now taken as a primitive judgment in all the axioms and rules i.e. it is no longer to be interpreted as the above wff.

Such judgements are made relative to a context Γ that now contains wff and type assignments of the form $x : T$. We shall write

$$\Gamma \vdash_{\mathbf{CST_T}} \Theta$$

if the sequent follows from the following rules - where we drop the subscript. We shall write c_Γ for the subset of Γ which consists of its set of type assignments.

We begin with the structural rules which now take the following form. These subsume the structural rules of **CST** and **T**.

$$\mathbf{Ax_1} \quad x : T \vdash x : T \qquad \mathbf{Ax_2} \quad \frac{\Gamma \vdash \phi \; prop}{\Gamma, \phi \vdash \phi}$$

$$\mathbf{W}_1 \quad \frac{\Gamma \vdash \Theta}{\Gamma, x : T \vdash \Theta} \qquad \mathbf{W_2} \quad \frac{\Gamma \vdash \Theta \quad \Gamma \vdash \phi \; prop}{\Gamma, \phi \vdash \Theta}$$

$$where \; x \notin FV(\Gamma) \cup FV(\Theta)$$

The equality axioms/rules remains intact except for the need to include a premise in E_2.

$$\mathbf{E_1} \quad \forall x : X \cdot x = x \qquad \mathbf{E_2} \quad \frac{\Gamma, x : X \vdash \phi[x] \; prop}{\Gamma \vdash \forall x : X \cdot \forall y : X \cdot x = y \rightarrow (\phi[x] \rightarrow \phi[y])}$$

The propositional logical rules are modified to preserve *propositions* from premises to conclusions. However, only the following rules need to be modified.

$$\mathbf{L_2} \quad \frac{\Gamma \vdash \Omega \quad \Gamma \vdash \phi \; prop}{\Gamma \vdash \phi}$$

$$\mathbf{L_8} \quad \frac{\Gamma \vdash \phi \quad \Gamma \vdash \psi \; prop}{\Gamma \vdash \phi \vee \psi} \qquad \mathbf{L_9} \quad \frac{\Gamma \vdash \psi \quad \Gamma \vdash \phi \; prop}{\Gamma \vdash \phi \vee \psi}$$

The axioms and rules for the types remain exactly as before except for the following. We must state the induction axioms as rules to get preservation of *being a proposition* from premises to conclusion.

$$\mathbf{N_5}\ \frac{\phi[0] \wedge \forall x : N \cdot \phi[x] \to \phi[x^+]}{\forall x : N \cdot \phi[x]}$$

$$\mathbf{S}_5\ \frac{\phi[\emptyset] \wedge \forall x : X \cdot \forall y : Set(X) \cdot \phi[y] \to \phi[x \circledast y]}{\forall y : Set(X) \cdot \phi[y]}$$

Almost finally, we modify the bounded quantifier rules to include type inference premises.

$$\mathbf{N_8}\ \frac{x : N \vdash \phi\ Prop}{\forall y : N \cdot \forall x < y \cdot \phi \leftrightarrow \forall x : N \cdot x < y \to \phi}$$

$$\mathbf{N_9}\ \frac{x : N \vdash \phi\ Prop}{\forall y : N \cdot \exists x < y \cdot \phi \leftrightarrow \exists x : N \cdot x < y \wedge \phi}$$

$$\mathbf{S_8}\ \frac{x : X \vdash \phi\ Prop}{\forall y : Set(X) \cdot \forall x \in y \cdot \phi \leftrightarrow \forall x : X \cdot x \in y \to \phi}$$

$$\mathbf{S_9}\ \frac{x : X \vdash \phi\ Prop}{\forall y : Set(X) \cdot \exists x \in y \cdot \phi \leftrightarrow \exists x : X \cdot x \in y \wedge \phi}$$

Alternatively, these could also be stated as pairs of rules without the need to include the type premises. Finally we admit the substitution rules

$$\mathbf{Sub}\ \frac{\Gamma, x : T \vdash \Theta \qquad \Gamma \vdash t : T}{\Gamma \vdash \Theta[t/x]} \qquad \mathbf{SUB}\ \frac{\Gamma[X] \vdash \Theta[X]}{\Gamma[T/X] \vdash \Theta[T/X]}$$

Of course we do not need these for the purely logical cases since they are already derivable form the quantifier rules. This completes the statement of the theory $\mathbf{CST}_T$.

4.2 Strong Typing

It is obviously more cumbersome to use $\mathbf{CST}_T$ than **CST** since a great deal of type inference has to be done in the process of carrying out the proofs. Although this can all be mechanized, it is harder to use by hand. However, $\mathbf{CST}_T$ does have the following very pleasant property that is not shared by **CST**.

Theorem 9 *(Strong Typing)*

(1) *If* $\Gamma \vdash_{\mathbf{CST}_T} \psi$ *then for each* ϕ *in* $\Gamma \cup \{\psi\}, c_\Gamma \vdash_{\mathbf{T}} \phi\ prop$
(2) *If* $\Gamma \vdash_{\mathbf{CST}_T} t : T$ *then* $c_\Gamma \vdash_{\mathbf{T}} t : T$
(3) *If* $\Gamma \vdash_{\mathbf{CST}_T} \phi\ prop$ *then* $c_\Gamma \vdash_{\mathbf{T}} \phi\ prop$

Proof. By induction on the structure of derivations. For part (1) we first consider the inference rules. Most of the rules are easy to check; we provide

some illustrations. Consider first the implication introduction rule. Consider the premise

$$\Gamma, \phi \vdash \psi$$

By part (1) and induction, $c_\Gamma \vdash_{\mathbf{T}} \phi\ prop$ and $c_\Gamma \vdash_{\mathbf{T}} \psi\ prop$. This yields $c_\Gamma \vdash_{\mathbf{T}} \phi \rightarrow \psi\ prop$. Next consider disjunction introduction. The premise is

$$\Gamma \vdash \phi \qquad c_\Gamma \vdash_{\mathbf{T}} \psi\ prop.$$

By induction, $c_\Gamma \vdash_{\mathbf{T}} \phi\ prop$. The result now follows by the grammar rule for disjunction. Next consider existential quantification introduction.

$$\frac{\Gamma \vdash a : T \qquad \Gamma \vdash \phi[a/x]}{\Gamma \vdash \exists x : T \cdot \phi}$$

By induction

$$c_\Gamma \vdash_{\mathbf{T}} a : T \qquad c_\Gamma \vdash_{\mathbf{T}} \phi[a/x]\ prop$$

By the properties of the type system, we may assume that $x \notin \alpha(c_\Gamma)$. Hence, by the back substitution lemma:

$$c_\Gamma, x : T \vdash_{\mathbf{T}} \phi[x]\ prop$$

We are then done by the existential formation rule. The axioms and rules for the types are equally easy to check. In particular the axioms are well-typed by their declaration contexts. The only other concern is with the structural rules which are easy to check. This completes part (1). Parts (2) and (3) are by structural induction and are easy to check.■

Part (1) ensures that only propositions are provable and guarantees that if a propositional term is used in an assumption, the assignment context will guarantee that it is provably a proposition. Notice that, via an obvious induction, the theorem also yields that all derivations are well typed - i.e. all formula in the derivation will be. Part (2) is exactly the property which distinguished the classifier and grammatical roles of types in the theory **CST**.

5 CST,T and $\mathbf{CST}_T$

We shall show that, with respect to well-typed expressions, **CST** is a conservative extension of $\mathbf{CST_T}$. This is achieved by translating **CST** into $\mathbf{CST_T}$ relative to a context c in such a way that each translated expression is, relative to c, *well-typed*.

The types are translated to themselves: we have only to translate the wff and terms. We deal first with the wff. The tricky clauses are those for the

atomic wff and here we translate the *ill-typed* cases as false.

$$
\begin{array}{rcl}
(t \in s)^c & = & \left\{ \begin{array}{l} t^c \in s^c \text{ if } c \vdash_T t^c : T \text{ and } c \vdash_T s^c : Set(T) \\ \Omega \text{ otherwise} \end{array} \right\} \\
(t = s)^c & = & \left\{ \begin{array}{l} t^c = s^c \text{ if } c \vdash_T t^c : T \text{ and } c \vdash_T s^c : T \; for \; some \; T \\ \Omega \text{ otherwise} \end{array} \right\}
\end{array}
$$

$$
(t < s)^c = \left\{ \begin{array}{l} t^c < s^c \text{ if } c \vdash_T t^c : N \text{ and } c \vdash_T s^c : N \\ \Omega \text{ otherwise} \end{array} \right\}
$$

For the main connectives and quantifiers, matters are straightforward.

$$
\begin{array}{rcll}
(\phi \circ \psi)^c & = & \phi^c \circ \psi^c & \circ \; any \; connective \\
(Qx : T \cdot \phi)^c & = & Qx : T \cdot \phi^{c,x:T} & Q = \exists, \forall \\
(QX \cdot \phi)^c & = & QX \cdot \phi^{c,x:T} & Q = \exists, \forall
\end{array}
$$

The translation of the bounded quantifiers follows a similar pattern where we have to take care of the bad cases explicitly.

$$
\begin{array}{rcl}
(\exists x < b \cdot \phi)^c & = & \left\{ \begin{array}{l} \exists x < b \cdot \phi^{c,x:T} \text{ if } c \vdash_T b^c : N \\ \Omega \text{ otherwise} \end{array} \right\} \\
(\forall x < b \cdot \phi)^c & = & \left\{ \begin{array}{l} \forall x < b \cdot \phi^{c,x:T} \text{ if } c \vdash_T b^c : N \\ \Omega \text{ otherwise} \end{array} \right\} \\
(\exists x \in b \cdot \phi)^c & = & \left\{ \begin{array}{l} \exists x \in b \cdot \phi^{c,x:T} \text{ if } c \vdash_T b^c : Set(T) \\ \Omega \text{ otherwise} \end{array} \right\} \\
(\forall x \in b \cdot \phi)^c & = & \left\{ \begin{array}{l} \forall x \in b \cdot \phi^{c,x:T} \text{ if } c \vdash_T b^c : Set(T) \\ \Omega \text{ otherwise} \end{array} \right\}
\end{array}
$$

Finally, for the translation of the terms we have only to be careful about the cases where the types are inappropriate and then assign the result some arbitrary value.

$$
\begin{array}{rcl}
0^c & = & 0 \\
(t^+)^c & = & \left\{ \begin{array}{l} (t^c)^+ \text{ if } c \vdash_T t^c : N \\ 0 \text{ otherwise} \end{array} \right\} \\
(t_1, t_2)^c & = & (t_1^c, t_2^c) \\
\pi_i(t)^c & = & \left\{ \begin{array}{l} \pi_i(t^c) \text{ if } c \vdash_T t^c : A \otimes B \; for \; some \; A, B \\ 0 \text{ otherwise} \end{array} \right\} \\
\Phi^c & = & \Phi \\
(a \circledast b)^c & = & \left\{ \begin{array}{l} a^c \circledast b^c \text{ if } c \vdash_T a^c : T \; and \; c \vdash_T b^c : Set(T) \\ \emptyset \text{ otherwise} \end{array} \right\}
\end{array}
$$

This completes the translation.

Lemma 10 *(Substitution Lemma)*

1. *If* $c \vdash_{\mathbf{T}} t^c : T$ *then* $\phi[t/x]^c = \phi^{c,x:T}[t^c/x]$
2. *If* $c \vdash_{\mathbf{T}} t^c : T$ *then* $s[t/x]^c = s^{c,x:T}[t^c/x]$

Proof. By simultaneous induction on the terms and wff. ■

Our first significant result shows that the translation always yields well-typed expressions.

Proposition 11 *(Typing) Let* c *be any context*

1. If c covers t/ϕ and $c \subseteq e$ then $t^c = t^e$ and $\phi^c = \phi^e$
2. If c covers ϕ then $c \vdash_{\mathbf{T}} \phi^c$ *prop*
3. If c covers t then $c \vdash_{\mathbf{T}} t^c : T$ *for some* T
4. If $c \vdash_{\mathbf{T}} \phi^c$ *prop* then $\phi = \phi^c$

Proof. (1) is by induction on the terms and wff. For parts 2 and 3, we employ a simultaneous induction on the syntax of terms and wff. For example, consider universal quantification. Assume that c covers $\forall x : T \cdot \phi$. We may assume that $FV(\forall x : T \cdot \phi)$ are exactly the variables of c. By induction,

$$c, x : T \vdash_{\mathbf{T}} \phi^{c,x:T} \; prop$$

Hence by the type rule for quantifiers

$$c \vdash_{\mathbf{T}} \forall x : T \cdot \phi^{c,x:T} \; prop$$

Part (4) is also by induction but we only need inspection of the clauses. ■

We can now prove the main theorem which ensures the translation preserves provability in the two systems.

Theorem 12 *(Soundness) If*

$$\Gamma \vdash_{\mathbf{CST}} \phi$$

and c *covers* Γ, ϕ *then*

$$\Gamma^c, c \vdash_{\mathbf{CST_T}} \phi^c$$

where Γ^c *is the translated context.*

Proof. By induction on the derivations in **CST**. Each of the axioms is easy to verify. We illustrate the logical rules with disjunction introduction, implication elimination and the universal quantification introduction rule. For the first assume c covers $\phi \vee \psi$. Then by induction

$$\Gamma^c, c \vdash \phi^c$$

Hence, since $c \vdash \psi^c$ *prop*, we are finished by the $\mathbf{CST}_T$ disjunction rule. For implication elimination,

$$\frac{\Gamma \vdash \phi \qquad \Gamma \vdash \phi \to \psi}{\Gamma \vdash \psi}$$

Let c cover the conclusion and expand it to e to cover ϕ as follows: for each $y \in FV(\Gamma \cup \{\phi\}) - FV(\psi)$ we add $y : N$ to c. Then we have by induction

$$\Gamma^e, e \vdash \phi^e \qquad \Gamma^e, e \vdash \phi^e \to \psi^e$$

Hence by the rules of $\mathbf{CST}_T$,

$$\Gamma^e, e \vdash \psi^e$$

Now by the lemma on the properties of the translation,

$$\Gamma^c, e \vdash \psi^c$$

For simplicity, suppose that y is the only variable in $FV(\Gamma \cup \{\phi\}) - FV(\psi)$. We then have:

$$\Gamma^c, c \vdash \forall y : N \cdot \psi^c$$

Since $y \notin FV(\psi)$, this yields

$$\Gamma^c, c \vdash \psi^c$$

as required. For the universal introduction rule i.e.

$$\frac{\Gamma, x : T \vdash \phi}{\Gamma \vdash \forall x : T \cdot \phi}$$

We may assume c exactly covers the conclusion. By induction, the premise follows. Hence, by the quantifier rule in $\mathbf{CST_T}$:

$$\frac{\Gamma^c, x : T \vdash \phi^{c,x:T}}{\Gamma^c \vdash (\forall x : T \cdot \phi)^c}$$

The introduction rule uses the substitution lemma for the translation.■

Putting this together with part 4 of the previous proposition, we have:

Theorem 13 **CST** *is a conservative extension of* $\mathbf{CST_T}$ *relative to well-typed wff.*

$\mathbf{CST_T}$ is conceptually prior in that it is a *typed logic* in the traditional and natural sense. It thus provides conceptual underpinning for the half-way house instantiated in **CST**. Consequently, we are justified in using our original cavalier and practical syntax together with the post-hoc type inference system. It all comes out in the wash. This is advantageous since we really have enough to do constructing the proofs without having to worry about type-checking.

6 Further Work

CST is a core theory that provides a basic framework for exploring some fundamental issues about specification. However, there are many such issues we have still to explore. Recursive specifications and the impact of *computability* considerations are two such. Set theoretic models of **CST** need to be developed and explored. In addition, we need to study the inclusion of new type constructors such as *recursive* types, *sub-types*, *function space* types and *objects/classes* etc. These will be considered in future publications.

References

[1] Barendregt, H.P.: Lambda Calculus With Types. Handbook of Logic in Computer Science. Oxford Science Publications. Abramsky, S., Gabbay, D.M. and Maibaum, T.S.E. (eds), pp. 118-310, 1992.

[2] Brien, S.M. and Nicholls, J.E. Z Based Standard Version 1.0, Oxford University Computing Laboratory. PRG 107, 1992.

[3] Dawes, J. The VDM-SL Reference Guide. Pitman. 1991.

[4] Diller, A. Z: An Introduction to Formal Methods. John Wiley& Sons, 1990.

[5] Flemming,D. Bruun, H. and Hansen, B.S. On Type Checking in VDM and Related Consistency Issues. VDM '91: Formal Software Development Methods, pp. 45-62, Springer-Verlag, October 1991.

[6] Henson, M. and Reeves, S. Revising Z -I Semantics and Logic. Journal of Formal Aspsects of Computing, vol 11,no 4, pp 359-380, 1999.

[7] Henson, M. and Reeves, S. Revising Z -II Logical Development. Journal of Formal Aspsects of Computing, vol 11,no 4, pp 381-401, 1999.

[8] Jones, C,B. and Middleburg, C.A. A typed Logic of Partial Functions reconstructed classically, Acta Infromatica, 31(5); 399-430, 1994.

[9] Larson, P,G. and Pawlowski, W. The Formal Semantics of ISO VDM-SL. The Journal of Computer Standards and Interfaces. 1995.

[10] Monahan, B. Type Checking for VDM Reference Language, July 1985.

[11] Nicholls, J.E. (ed) : Z-Notation version 1.2, 1995.

[12] Plat ,N., Huijsman, R., Katwijk, C, Van Oosten, G, Pronk ,G and Toetenel, H. Type checking BSI/VDM SL. VDM '90 VDM and Z – Formal Methods in Software Development, pp. 399-425, Springer-Verlag, April 1990.

[13] Spivey, J.M. : Understanding Z. Cambridge University Press, 1988.

[14] Spivey, J.M.: The Z Notation: A Reference Manual. Prentice Hall, 1992.

[15] The Foundations of Specification I, This journal.

[16] Woodcock, J. and Brien, S.M. : W: a logic for Z, in J.E. Nicholls (ed), Z Users Workshop, York, 1991, Proceedings of sixth Annual Z User Meeting. Springer Verlag, 1992.