Relative Equational Specification and Semantics

Jo Erskine Hannay

Technical Report ECS–LFCS–97–366
Department of Computer Science
University of Edinburgh

October 8, 1997

Abstract: Standard concepts of initial and final algebra semantics are generalised in a modular hierarchical manner. The resulting *relative* formalism allows a unified view on the relationship between initial and final algebra semantics and gives a dualised notion of consistency. Using this, a modular hierarchical approach to proof by consistency is taken by which only top-level equations need be considered at any level. The formalism also allows non-homogeneous specification schemes and different proof methods at each level.

Contents

1 Introduction

One incentive for *equational (algebraic) specification* of abstract data types is its aptness for a variety of simple and efficient methods of formal semi-mechanised proof. A particularly immediate family of proof methods are those based on term rewriting techniques. Two choices of semantics dealt especially with by rewrite based methods are *initial* and *final/terminal* (algebra) semantics. Roughly, two terms are semantically equal according to initial algebra semantics *only if* their equality can be proven in the equational calculus. Final algebra semantics on the other hand, determine two terms equal *unless* proven otherwise in the calculus. In a purely equational language, to prove an "inequation" is to show distinguishability of the two terms involved w.r.t. a given basic (primitive) semantics, i.e. s cannot be equivalent to t if there is a context c such that $c[s]$ and $c[t]$ are not equivalent according to the primitive/basic semantics. Hence, final semantics is observational and behavioural in that two terms are equivalent if they "behave" equivalently when used in observable (relevant) contexts. Besides being able to easily describe data types which are cumbersome to describe directly (maps, infinite data types, processes [Pad95]), final semantics also provides abstraction up to equivalent observable behaviour as described by behavioural and observational semantics, e.g. [ST87, BHW94].

Proof by consistency is a general method for proving/refuting equations according to initial or final semantics. The idea pioneered by Musser [Mus80], is that when adding a hypothesis to a specification, Knuth-Bendix completion [KB70] may be applied for detecting inconsistencies refuting the hypothesis. The method also goes by the name "induction-less induction", the 'induction' in question being the structural induction of [Bur69, Gut75]. For the initial case, the general method has been elaborated by many authors e.g. [Gog80, HH82, JK89, Küc89, Fri86, Bac88]. In [LPL96] proof by consistency is adapted to higher-order term-rewriting.

A method for proof by consistency in final semantics was described in [Pue84]. Structural *context induction* [Hen91, Hen92] has been developed for proofs according to behavioural and observational semantics, hence one might call proof by consistency in final semantics "induction-less" context induction. Lysne in [Lys92, Lys94] observes that proof by consistency is inherently close in spirit to reasoning according to final semantics, since explicit output is a witness of distinguishability. Lysne describes an approach to proof by consistency according to final semantics which subsumes, fully or partly, several of the approaches designated for initial semantics.

This note elaborates on the relationship between initial and final semantics. Final semantics are defined relatively to an underlying initial semantics in which inequalities are observed. Also, consistency in the initial sense can be seen as relative to a "kernel" semantics. These evident notions of relativity are made explicit by presenting a formalism, *relative semantics*, which provides a common vantage point for initial and final semantics. The formalism expresses consistency more explicitly, and enables a nice dualised understanding of different concepts of consistency. Explicit mention of a "kernel", displays a link with hierarchical specification as in [WPP+83]. The formalism of relative semantics thus suggests a modular hierarchical style of specification linked directly to modular level-wise proof by consistency. Early ideas of a hierarchical approach to consistency and inductive completion for initial semantics appear in [Kir84].

Relative semantics is a generalisation in several respects. Firstly, it generalises standard initial and final semantics into one common framework. Secondly, it generalises initial and final semantics to arbitrary level hierarchies of semantics. Thirdly, whereas other methods yield hierarchical constructs which always can be viewed monolithically, relative semantics also cater for non-homogeneous situations where one cannot reduce a hierarchy to a monolithic heap. Indeed, by catering for non-homogeneous specification the relative construction supports different proof methods at each level.

The formalism of relative semantics presented here first appeared in [Han95]. In [Pad95] a similar formalism is presented in the context of general inductive and co-inductive definitions of relations and functions with corresponding initial (least fixed-point) and final (greatest fixed-point) semantics (e.g. [Gor94]). The way in [Pad95] of dealing with proofs according to final semantics is by explicitly converting co-inductive definitions to inductive ones using finite approximants thereby making inductive proof techniques applicable in the final case. This is somewhat different from the approach described in the present note, where the observation that proof by consistency is inherently co-inductive in nature is used for doing proofs directly according to final semantics. Also, emphasis is on notions of consistency which we find useful for understanding more of the common nature of initial and final semantics, and on modularity of structures and proofs in such a way as to allow different proof methods at each level.

Section 2 reviews basic notions, section 3 reviews definitions of initial and final semantics, sections 4 & 5 introduce relative semantics and notions of consistency, section 6 describes modular level-wise proof by consistency and section 7 illustrates non-homogeneous specification and reasoning.

2 Preliminaries

This section gives some basic notions of algebraic specification and term rewriting that are used directly in this note. They are to be found in greater detail for algebraic specification in [Wir90] or [EM85], and for rewriting in [Klo92] or [DJ90].

We will be dealing with many-sorted specifications whose semantics will be given as classes of total many-sorted algebras with non-empty carriers.

A *signature* $\Sigma = \langle S, F\rangle$ consists of a set of sorts and a set of function symbol profiles of the form $f : s_1 \times \cdots \times s_n \to s$ for $s_1, \ldots, s_n, s \in S$ giving a function symbol along with its arity and sort. A function symbol with arity 0 is called a *constant*. A *sensible* signature has at least one constant of every sort in S. Without further mention all signatures in this note are assumed sensible. A *total heterogeneous* Σ*-algebra* $A = \left\langle \langle A_s\rangle_{s\in S}, \langle f^A\rangle_{f\in F}\right\rangle$ consists of an S-indexed family $\langle A_s\rangle_{s\in S}$ of non-empty carriers and an F-indexed $\langle f^A\rangle_{f\in F}$ family of total functions containing an $f^A \in (A_{s_1}\times\cdots\times A_{s_n} \to A_s)$ for every $f : s_1\times\cdots\times s_n \to s \in F$. All algebras in this note are assumed to be total. A Σ-*term algebra* $T_\Sigma(X)$ for a set of variables X is the free algebra generated from X over the signature Σ, any carrier $T_\Sigma(X)_s$

containing exactly the terms of sort s generated from X and Σ. The *ground*-term Σ-algebra $T_\Sigma(\emptyset)$ is denoted by G_Σ.

In the following, let $\Sigma = \langle S, F\rangle$ be fixed. A *homomorphism* $\phi : A \to B$ is a family $\langle \phi_s \in (A_s \to B_s)\rangle_{s\in S}$ of mappings such that for all $f : s_1 \times \cdots \times s_n \to s \in F$ and all $a_1 \in A_{s_1}, \ldots, a_n \in A_{s_n}$: $\phi_s(f^A(a_1, \ldots, a_n)) = f^B(\phi_{s_1}(a_1), \ldots, \phi_{s_n}(a_n))$.

A *substitution* is a homomorphism $\sigma : T_\Sigma(X) \to T_\Sigma(X)$. A *context* $c[\square]$ is a term with a single *place holder* or *hole* $\square$. We write $c \in T_\Sigma(X)$ for $c[\square] \in T_{\Sigma\cup\{\square\}}(X)$. *Context application* is a context with a term substituted for its place holder, e.g. $c[s]$;

An *interpretation* $\phi^A_{T_\Sigma(X)} : T_\Sigma(X) \to A$ is a homomorphism from a term-algebra to an algebra in which one can think of the terms as being interpreted. A *computation structure* is an algebra A such that $\phi^A_{G_\Sigma}$ is surjective, i.e. all carrier elements are denotable by ground terms.

A *congruence* $\sim$ on an algebra A is a family $\langle\sim_s\rangle_{s\in S}$ such that each $\sim_s$ is an equivalence relation on A_s and if $f : s_1 \times \cdots \times s_n \to s \in F$ then for all $a_1, b_1 \in A_{s_1}, \ldots, a_n, b_n \in A_{s_n}$: $a_1 \sim_{s_1} b_1, \ldots, a_n \sim_{s_n} b_n \;\Rightarrow\; f^A(a_1, \ldots, a_n) \sim_s f^A(b_1, \ldots, b_n)$. The *quotient* $A/\!\sim$ w.r.t. an algebra A and congruence $\sim$ is the algebra with carriers $A_s/\!\sim_s$ for each $s \in S$ and functions defined by $f^{A/\sim}([a_1]_{\sim_{s_1}}, \ldots, [a_n]_{\sim_{s_n}}) = [f(a_1, \ldots, a_n)]_{\sim_s}$, for every $f : s_1 \times \cdots \times s_n \to s \in F$.

An algebra A is *initial* in a class K of Σ-algebras if $A \in K$ and for every B in K there exists exactly one homomorphism from A to B. Conversely A is *final* in K if $A \in K$ and for every B in K there exists exactly one homomorphism from B to A.

A (Σ-)*equation* is an element in $T_\Sigma(X)_s \times T_\Sigma(X)_s$, written $u = v$, for $u, v \in T_\Sigma(X)_s$; $s \in S$, and is *valid in a* Σ-*algebra* A, written $A \models u = v$, iff $\phi^A_{T_\Sigma(X)}(u) = \phi^A_{T_\Sigma(X)}(v)$ for all interpretations $\phi^A_{T_\Sigma(X)}$. A Σ-algebra A is a *model* for a set of Σ-equations E if every equation in E is valid in A. The class of all Σ-algebras which are models of E is denoted by $Mod_\Sigma(E)$. The following is an important fact relating initiality and finality to validity.

Fact 1. *For any $g, g' \in G_\Sigma$*

1. *A Σ-computation structure I is initial in a class K of Σ-algebras iff*
$$I \models g = g' \;\Leftrightarrow\; \text{for all } A \in K : A \models g = g'.$$
2. *A Σ-computation structure F is final in a class C of Σ-computation structures iff*
$$F \not\models g = g' \;\Leftrightarrow\; \text{for all } A \in C : A \not\models g = g'.$$

A *term rewrite system* (TRS) $\langle\Sigma, R\rangle$ consist of a signature and a set R of *rewrite rules* $v \to h$ over $T_\Sigma(X)$. A TRS is often simply denoted by its set of rules R. For any $s, t \in T_\Sigma(X)$, s *rewrites* in R to t in one step, written $s \xrightarrow[R]{} t$, if there is a $v \to h \in R$ and a substitution σ such that $s = c[v\sigma]$ and $t = c[h\sigma]$, for some context $c[\square]$. The task of finding such a σ and $c[\square]$ is called *matching* and is algorithmic. The relation $\xrightarrow[R]{}$ is a *rewrite relation*, meaning it fulfils $s \xrightarrow[R]{} t \;\Rightarrow\; c[s\sigma] \xrightarrow[R]{} c[t\sigma]$ for all contexts $c[\square]$ and substitutions σ. Moreover it is the least such rewrite relation, meaning that $\xrightarrow[R]{}$ is the *rewrite closure* of R.

The transitive closure of $\xrightarrow[R]{}$ is denoted by $\xrightarrow[R]{+}$, the reflexive-transitive closure by $\xrightarrow[R]{*}$ and the symmetric closure by $\xleftrightarrow[R]{}$.

The equational calculus will be dealt with in the equivalent form of symmetric TRSs; i.e. we consider as *provability relation* for a given set of equations E, the relation $\xleftrightarrow[E]{*}$; i.e.

the rewrite, reflexive, symmetric, transitive closure of E. For any terms s and t and set of equations E, we then have $E \vdash s = t \Leftrightarrow s \overset{*}{\underset{E}{\leftrightarrow}} t$. For any set of rules or equations E, the closure $\overset{*}{\underset{E}{\leftrightarrow}}$ is a congruence, and quotients w.r.t. $\overset{*}{\underset{E}{\leftrightarrow}}$ are usually written A/E.

For any relation $\Re$ a *derivation* in $\Re$ or an $\Re$-*derivation* is a possibly infinite sequence $\langle t_0, t_1, \ldots, t_i, \ldots\rangle$ such that for every $0 \leq j$, $t_j \Re t_{j+1}$.

A TRS $\langle \Sigma, R\rangle$ is *terminating* if there is no infinite derivation in $\underset{R}{\rightarrow}$. It is *confluent* if

$$s \overset{*}{\underset{R}{\leftarrow}} w \overset{*}{\underset{R}{\rightarrow}} t \;\Rightarrow\; \exists u \mid s \overset{*}{\underset{R}{\rightarrow}} u \overset{*}{\underset{R}{\leftarrow}} t$$

If both these properties are present the TRS is *complete*. One then has a rewrite driven decision procedure for the relation $\overset{*}{\underset{R}{\leftrightarrow}}$, if R has finitely many rules: Completeness gives that every term $t \in T_\Sigma(X)$ has a *unique normal form* $t!$, i.e. a unique $t!$ such that $t \overset{*}{\underset{R}{\rightarrow}} t!$ and $\nexists u \mid t! \overset{*}{\underset{R}{\rightarrow}} u$, which can be computed in finitely many steps.

Knuth-Bendix *completion* [KB70] was originally designed to generate a complete R with the same equational theory as a given set of equations E. However, in the attempt of generating such a TRS, the procedure derives new theorems in a certain manner which can be useful for other purposes even if it doesn't succeed in its completion task. The method of proof by consistency thus uses the completion process to derive certain inconsistency witnesses generated if an added equation is not an *inductive* consequence of the equations already present. This checks hypotheses according to the initial model, but as mentioned in the introduction the method can also be used according to other semantics, e.g. final algebra semantics.

Completion is usually described by a set of inference rules manipulating a pair $\langle E, R\rangle$ (see any of the two references above) and the relation $\vdash_{KB}$ is defined such that $\langle E, R\rangle \vdash_{KB} \langle E', R'\rangle$ if $\langle E', R'\rangle$ is obtainable from $\langle E, R\rangle$ in one step by applying any of the inference rules. A possibly infinite *completion sequence* $\langle\langle E_0, R_0\rangle, \ldots, \langle E_i, R_i\rangle, \ldots\rangle$ is then such that for every i, $\langle E_i, R_i\rangle \vdash_{KB} \langle E_{i+1}, R_{i+1}\rangle$. Running completion in the original sense means ideally hoping for the following property: $\langle E, \emptyset\rangle \vdash^*_{KB} \langle\emptyset, R\rangle$ and no inference rule is applicable. Then R is complete and has the same equational theory as E. Since the starting point is a set of unordered equations, and since termination is in general undecidable, a completion process is equipped with a user defined term ordering so that equations, original or generated, may be oriented into rewrite rules in such a way that every R_i is terminating.

The driving force in completion is the elimination of non-confluence (this view is due to Bachmair [Bac91]). Any derivation of $s \overset{*}{\underset{R}{\leftrightarrow}} t$ for a non-confluent R might have *peaks* $s_{i-1} \underset{R}{\leftarrow} s_i \underset{R}{\rightarrow} s_{i+1}$ as well as *valleys* $s_{j-1} \underset{R}{\rightarrow} s_j \underset{R}{\leftarrow} s_{j+1}$. A completion process may be viewed as a peak elimination process, ultimately seeking to produce a set of rules such that a *rewrite proof* $s \overset{*}{\underset{R}{\rightarrow}} u \overset{*}{\underset{R}{\leftarrow}} t$ is possible for every s, t s.t. $s \overset{*}{\underset{R}{\leftrightarrow}} t$. We have:

Fact 2. (Bachmair [Bac87]) *Suppose a possibly infinite completion sequence*

$$\langle\langle E_0, R_0\rangle, \ldots, \langle E_i, R_i\rangle, \ldots\rangle$$

is fair *w.r.t. inferences, i.e. no possible inference is overlooked infinitely often and every equation generated is orientable. Then for any s and t:*

$$s \overset{*}{\underset{E_0}{\leftrightarrow}} t \;\Leftrightarrow\; \exists j \mid s!R_j = t!R_j$$

It is this property that we will be using for our version of proof by consistency.

One standard form of proof by consistency is based on *inductive completion*. A term t is said to be *ground reducible* in R if for all ground substitutions $\sigma \in (T_\Sigma(X) \to G_\Sigma)$, there exists a u such that $t\sigma \xrightarrow[R]{+} u$. Ground reducibility is decidable. We have:

Fact 3 (Inductive completion). *Let R be complete and having the same equational theory as E. For any equation $s = t$ consider a completion sequence*

$$\langle \langle E_0, R_0 \rangle, \ldots, \langle E_i, R_i \rangle, \ldots \rangle$$

which is fair *w.r.t. inferences, and where every equation generated is orientable. Then:*

$$G_\Sigma/E \not\models H$$
$$\Updownarrow$$
a rule $\langle l, r \rangle$ is generated during the process s.t.
l is not ground reducible in R.

Constructive function specification suggests viewing a signature Σ as disjointly divided into Σ_c consisting of *constructors* and Σ_d consisting of *defined operators*. Essential syntactic qualities are then captured as follows:

Definition 1. (Guttag [Gut77]) *A set of equations or rules E is* ***sufficiently complete w.r.t.*** *Σ_c if for every $g \in G_\Sigma$ there exists a $g_c \in G_{\Sigma_c}$ such that $g \overset{*}{\underset{E}{\Leftrightarrow}} g_c$.*

Definition 2. (Guttag [Gut77]) *Let E_c be the Σ_c-equations in E. Then, E is* ***consistent w.r.t.*** *Σ_c if $g_c \overset{*}{\underset{E}{\Leftrightarrow}} g'_c \;\Leftrightarrow\; g_c \overset{*}{\underset{E_c}{\Leftrightarrow}} g'_c$ for arbitrary $g_c, g'_c \in G_{\Sigma_c}$.*

Suppose now that the term ordering given for completion is such that every $u \in T_{\Sigma_d \cup \Sigma_c}(X) \setminus T_{\Sigma_c}(X)$ is greater in the ordering than all $v \in T_{\Sigma_c}(X)$, and that R is sufficiently complete w.r.t. Σ_c. Then a refuting witness $l = r$ for inductive completion will be a rule purely over $T_{\Sigma_c}(X)$, and moreover for some ground substitution σ, $l\sigma \overset{*}{\underset{R}{\not\Leftrightarrow}} r\sigma$. Thus, a refuting witness is a rule which collapses R-equivalence classes, i.e. E-equivalence classes in G_Σ/E. Moreover, since R is complete, R is consistent w.r.t. Σ_c: Suppose $g_c \overset{*}{\underset{R}{\Leftrightarrow}} g'_c$. Then by completeness we have $g_c \xrightarrow[R]{*} g_c! \xleftarrow[R]{*} g'_c$, and by the term ordering $g_c \xrightarrow[R_c]{*} g_c! \xleftarrow[R_c]{*} g'_c$. A refuting witness is therefore a witness of inconsistency in the sense of definition 2, resulting from adding the hypothesis $s = t$.

There are essentially two things that can go wrong in any completion based process. One is non-termination of the process. There are however results [Her92] that can be used in some cases to determine whether an infinite sequence will bring anything essentially new, or not, in which case the remaining sequence can be neglected. The more annoying thing that may happen is inability to orient some equation, either because the given term ordering is too weak, or because it is fundamentally impossible to orient the equation without giving a non-terminating set of rules. There are extensions to standard rewriting and completion which try to deal with the latter case. *Ordered* rewriting and completion exploit the possibility that all ground instances of equations may be orientable. *Extended* or *class* rewriting puts all non-orientable equations in a separate set and does rewriting and completion modulo the equational theory of this set. The appropriate versions of facts 2 and 3 hold for these variants.

3 Equational Specification and Semantics

In proof technical considerations as those done for e.g. proof by consistency, the models directly involved are syntax dependent term models of specifications or equations. These are typically, and naturally, initial or final in character. The semantics defined in the following two sections for the specification formalisms presented there all give term models generated directly from the syntax and the equations involved. In section 4, the semantics of relative specifications is generated this way relative to a kernel semantics.

Initial and final semantics represent *monomorphic* interpretation schemes in that model classes for specifications consist of exactly one algebra (up to isomorphism). In a refinement setting [ST97, Mor94] monomorphic semantics are directly relevant at final stages of development, but due to fact 1, these semantics also represent larger model classes and are thus relevant at earlier development stages as well.

This section reviews standard notions of equational specification and initial and final semantics. The next section introduces relative equational specification and initial and final semantics thereof.

Definition 3. *An* ***(equational) specification*** *SP is a pair $SP = \langle \Sigma, E \rangle$ consisting of a signature Σ and a set of Σ-equations E.*

Initial and final semantics are described as follows:

Definition 4. *For $SP = \langle \Sigma, E \rangle$ a specification, the restriction $\overset{*}{\underset{E}{\leftrightarrow}}{}_{G_\Sigma}$ of $\overset{*}{\underset{E}{\leftrightarrow}}$ to G_Σ is the* ***initial semantics*** *specified by SP.*

Definition 5. (e.g. Lysne [Lys92]) *For $SP = \langle \Sigma, E \rangle$ a specification, the congruence*

$$g \sim^{\mathcal{Q}} g' \;\Leftrightarrow\; \neg \exists c \in G_\Sigma ; g_c \in G_{\Sigma_c} \mid (c[g] \overset{*}{\underset{E}{\leftrightarrow}} g_c \overset{*}{\underset{E}{\not\leftrightarrow}} c[g'] \;\text{ or }\; c[g'] \overset{*}{\underset{E}{\leftrightarrow}} g_c \overset{*}{\underset{E}{\not\leftrightarrow}} c[g])$$

is the ***final semantics*** *specified by SP.*

Since for any congruence $\sim$ on G_Σ, $g \sim g' \;\Leftrightarrow\; G_\Sigma/{\sim} \models g = g'$, we here deliberately confuse the semantical model $G_\Sigma/{\sim}$ with the underlying $\sim$.

An algebra's property of being initial or final in some interesting class of algebras may reveal information as to how well the corresponding interpretation scheme fulfils the intentions of specification. The monomorphic intention behind the specification formalisms dealt with here would be that one *specifies all there is*, i.e. all intended semantic equalities are specified (initial intention), or that all semantic *in*equalities are specified (final intention).

Let $\sim$ be any congruence on G_Σ. Consider the class

$$\lfloor \sim \rfloor_\Sigma = \{ A \mid g \sim g' \;\Rightarrow\; A \models g = g' \}$$

of Σ-computation structures. An algebra in $\lfloor \sim \rfloor_\Sigma$ satisfies all the equalities (and perhaps more) induced by $\sim$. For example, the restriction of $Mod_\Sigma(E)$ to computation structures is by completeness (Birkhoff) the class $\lfloor \overset{*}{\underset{E}{\Leftrightarrow}} \rfloor_\Sigma$. On the other hand, the algebras in the class of Σ-computation structures

$$\lceil \sim \rceil_\Sigma = \{A \mid g \not\sim g' \;\Rightarrow\; A \not\models g = g'\}$$

satisfy no equality which is not induced by $\sim$.[1]

The quotient G_Σ/E ($G_\Sigma/\overset{*}{\underset{E}{\Leftrightarrow}}$) is initial in $\lfloor \overset{*}{\underset{E}{\Leftrightarrow}} \rfloor_\Sigma$, meaning that it is the computation structure model of E satisfying fewest equations, i.e. no equation holds unless explicitly entailed in the equational calculus from E. Conversely, $G_\Sigma/\sim^Q$ is final in $\lceil \overset{*}{\underset{E_c}{\Leftrightarrow}} \rceil_{\Sigma_c} \cap \lfloor \overset{*}{\underset{E}{\Leftrightarrow}} \rfloor_\Sigma$, meaning that $G_\Sigma/\sim^Q$ is the computation structure model of E satisfying the greatest amount of equations, but satisfying no more equations over G_{Σ_c} than those explicitly entailed by E_c, i.e. satisfying the "inequations" of the initial semantics on G_{Σ_c}. Further inequalities are derived from these basic inequalities by the congruence principle:

$$c[g] \not\sim c[g'] \;\Rightarrow\; g \not\sim g' \tag{1}$$

(In contrast, the final model of only $\lfloor \overset{*}{\underset{E}{\Leftrightarrow}} \rfloor_\Sigma$ contains one element and is generally not very interesting.) These two extremal modes of semantics (initial and final) are the results of two natural ways of generating term models from syntax and equations.

Specific specifications will be written in a syntax suggested in the following examples. Specification names will coincide with signature names:

Example 1. The following specification just specifies integers generated by 0 and the two operators *successor* and *predecessor*, together with two operations for addition and subtraction:

spec $\mathsf{Int_{op}}$
  sorts int
  constructors 0 : int,   succ, pred : int $\to$ int
  defined oprs +, – : int $\times$ int $\to$ int
  axioms $E_{\mathsf{Int_{op}}}$: succ(pred(x)) = x,
      pred(succ(x)) = x,

      x+0 = x,
      x+succ(y) = succ(x+y),
      x+pred(y) = pred(x+y),

      x – 0 = x,
      x–succ(y) = pred(x–y),
      x–pred(y) = succ(x–y)

The initial semantics $\overset{*}{\underset{E_{\mathsf{Int_{op}}}}{\longleftrightarrow}}_{G_{\mathsf{Int_{op}}}}$ consists of all inductive consequences of $E_{\mathsf{Int_{op}}}$, i.e. all equations whose ground instantiations are logical consequences of $E_{\mathsf{Int_{op}}}$. ◦

[1]or satisfy all inequalities (or more) induced by $\sim$, but strictly speaking this formulation presumes 'inequality' expressible in the formula language.

Example 2. This specification specifies an operator *altsum* that takes as input a sequence of integers and produces the alternating sum of the sequence. Here the alternating sum of e.g. $\langle x_0, x_1, x_2, x_3, x_4\rangle$ is $(x_0 - x_1) + (x_2 - x_3) + x_4$.

$$
\begin{array}{l}
\textbf{spec } \mathsf{AltSumList\text{--}Int_{op}} \\
\quad \textbf{sorts } \mathsf{int}, \mathsf{list} \\
\quad \textbf{constructors } \mathsf{0} : \mathsf{int}, \mathsf{succ}, \mathsf{pred} : \ldots \quad, \varepsilon : \mathsf{list}, \dashv : \mathsf{int} \times \mathsf{list} \to \mathsf{list} \\
\quad \textbf{defined oprs } \mathsf{altsum} : \mathsf{list} \to \mathsf{int} \\
\quad \textbf{axioms } E_{\mathsf{Int_{op}}} \cup E_{\mathsf{AltSumList}} : \begin{array}[t]{l} \mathsf{altsum}(\varepsilon) = 0, \\ \mathsf{altsum}(\mathsf{x} \dashv \varepsilon) = \mathsf{x}, \\ \mathsf{altsum}(\mathsf{x} \dashv (\mathsf{y} \dashv \mathsf{q})) = (\mathsf{x} - \mathsf{y}) + \mathsf{altsum}(\mathsf{q}) \end{array}
\end{array}
$$

The final semantics $\sim^{\mathcal{Q}}$ on $G_{\mathsf{AltSumList-Int_{op_c}}}$ gives by the *observer* **altsum** that *list* constructor terms are semantically equal if and only if they represent integer sequences whose alternating sums are equal. ○

The method of this example of specifying semantics on one type based upon the initial semantics of another is studied and used in [Kam83, Les83, GHJ85, DO91].

We might *bias* specifications by indicating what sort of semantics we want. For example, the initially and finally biased specifications of example 1 would have the headings **spec** $\mathsf{Int_{op}}$ **initial**, and **spec** $\mathsf{AltSumList\text{--}Int_{op}}$ **final** respectively.[2]

4 Relative Specification and Semantics

We now define relative specification and semantics. A relative semantics is defined w.r.t. a given kernel semantics, and is thus hierarchical in nature. Making a kernel explicit in specification and semantics will allow us to discuss consistency in relation to notions of conservative extensions of specifications. We also find that the definition of final semantics is a bit more intuitive when expressed relatively to an explicit kernel.

Definition 6. *An* ***(equational) relative specification*** *is either an equational specification or a pair $RSP = \langle SP, SP^{\kappa}\rangle$ consisting of an equational specification $SP = \langle \Sigma, E\rangle$ and a relative specification $SP^{\kappa} = \langle \Sigma^{\kappa}, E^{\kappa}\rangle$, such that $\Sigma^{\kappa} \subseteq \Sigma$, but $E \cap E^{\kappa} = \emptyset$. The specification SP^{κ} is called the* ***kernel*** *of RSP.*

This is slightly different from the definition of *hierarchical specification* in [WPP+83] and *extension* in [Pad95] where SP always includes SP^{κ}, and in particular the upper level equations of SP always include the equations of SP^{κ}. Our intention is that the two sets of equations are disjoint. This lets us reason modularly, actually disregarding details of the primitive specification. Whereas hierarchical specifications treated in e.g. [WPP+83] can always be seen as non-hierarchical ones, we wish to be able to treat also non-homogeneous situations in which a hierarchy can not be viewed as a monolith (section 7). The formalism of relative specification reflects this.

[2]The notion of 'bias' here is not that discussed in connection to VDM.

A *biased relative specification* is a relative specification with a bias on each component SP and SP^κ.

4.1 Relative Semantics

We now focus on the semantics of biased relative specifications. We start with the final case. As seen in the previous section, final semantics depend upon basic inequalities in a *kernel semantics* in order to give new inequalities. Semantics specified in this manner are therefore inherently hierarchical. Relative semantics makes this explicit:

Definition 7. *Let $RSP = \langle SP, SP^\kappa \rangle$ with $SP = \langle \Sigma, E \rangle$ and biased $SP^\kappa = \langle \Sigma^\kappa, E^\kappa \rangle$ be a relative specification, such that $\sim^\kappa$ on G_{Σ^κ} is the semantics specified by SP^κ. Define the following relation $\sim^\zeta = \langle \sim^\zeta_s \rangle_{s \in S}$ on G_Σ: For any $s \in S$ and all $g, g' \in G_{\Sigma s}$*

$$g \sim^\zeta_s g' \quad \Leftrightarrow \quad \neg \exists c \in G_\Sigma; g_1, g_2 \in G_{\Sigma^\kappa} \mid c[g] \overset{*}{\underset{E}{\Leftrightarrow}} g_1 \not\sim^\kappa_{s'} g_2 \overset{*}{\underset{E}{\Leftrightarrow}} c[g']$$

for some sort $s' \in S$. We call $\sim^\zeta$ the ***final semantics relative to*** *$\sim^\kappa$ specified by RSP. We refer to $\sim^\kappa$ as the* ***kernel*** *of $\sim^\zeta$.*

Example 3. The specification of example 1 can now be rephrased hierarchically:

spec AltSumList **final**
relative to $\mathsf{Int_{op}}$ **initial**
  sorts list
  constructors ε : list, $\dashv$: int $\times$ list $\to$ list
  defined oprs altsum : list $\to$ int
  axioms $E_{\mathsf{AltSumList}}$: $\mathsf{altsum}(\varepsilon) = 0$,
    $\mathsf{altsum}(\mathsf{x} \dashv \varepsilon) = \mathsf{x}$,
    $\mathsf{altsum}(\mathsf{x} \dashv (\mathsf{y} \dashv \mathsf{q})) = (\mathsf{x} - \mathsf{y}) + \mathsf{altsum}(\mathsf{q})$

The final semantics $\sim^\zeta$ on $G_{\mathsf{AltSumList}_c}$ relative to the initial semantics on $G_{\mathsf{Int_{op}}}$ is identical to $\sim^{\mathcal{Q}}$ of example 1. ○

Now, $\sim^{\mathcal{Q}} = \sim^\zeta$ is not true in general, but the identity does hold under certain sound conditions and consistency. We will return to this issue after introducing relative semantics in fuller depth.

The relation $\sim^\zeta$ in definition 7 is not necessarily a congruence. It is always symmetric and satisfies the congruence principle, but reflexivity and transitivity may be not satisfied: Let $\sim^\kappa$ be the identity congruence on $G_{\{a,b\}}$ for constants a, b. Let $E = \{a = b\}$. We then have $a \overset{*}{\underset{E}{\Leftrightarrow}} a \not\sim^\kappa b \overset{*}{\underset{E}{\Leftrightarrow}} a$, so $a \not\sim^\zeta a$. To demonstrate lack of transitivity, let $\sim^\kappa$ on $G_{\{a,b,c,d\}}$ be such that $a \not\sim^\kappa b$, $c \sim^\kappa d$ and $d \sim^\kappa e$. Let $E = \{f(c) = a, f(e) = b\}$. Then we have $f(c) \overset{*}{\underset{E}{\Leftrightarrow}} a \not\sim^\kappa b \overset{*}{\underset{E}{\Leftrightarrow}} f(e)$, giving $c \not\sim^\zeta e$, in spite of $c \sim^\zeta d$ and $d \sim^\zeta e$.

Hence, this relation is strictly speaking not worthy of being called a semantics. However it is reasonable to impose the following restrictions on definition 7:

`KC`: E is ***kernel conservative***: $\overset{*}{\underset{E}{\Leftrightarrow}}_{G_{\Sigma^\kappa}} \subseteq \sim^\kappa$

`SC`: E is sufficiently complete w.r.t. Σ^κ_c.

`KC` asserts that all equalities on Σ^κ-terms explicitly entailed by E are (already) specified by SP^κ. This technically resembles the notion of 'hierarchy conservative' in [WPP+83], the difference being again that here E does not (necessarily) contain the equations of SP^κ. `SC`'s reference to sufficient completeness w.r.t. the constructors of Σ^κ is a slight generalisation of the notion in definition 1, also extended naturally to many-sorted situations. Regarding $\sim^\kappa$ as constructor semantics, focusing on constructive function specification would require that `SC` is fulfilled.

Theorem 1. *Given* `KC` *and* `SC`*, the relation $\sim^\zeta$ in definition 7 is a congruence.*

Now we state a result of finality for $\sim^\zeta$ defined above as we did in the previous section for $\sim^\mathcal{Q}$. The following lemmas are easily proven:

Lemma 2. *$g \not\sim^\kappa g' \;\Rightarrow\; g \not\sim^\zeta g'$: all inequalities in $\sim^\kappa$ are preserved in $\sim^\zeta$.*

Lemma 3. *Assume* `KC` *and* `SC`*. Then $G_\Sigma/\sim^\zeta \in Mod_\Sigma(E)$*

And then we have:

Theorem 4. *Assume* `KC` *and* `SC`*. Then $G_\Sigma/\sim^\zeta$ is final in $\lceil \sim^\kappa \rceil_{\Sigma^\kappa} \cap \lfloor \overset{*}{\underset{E}{\Leftrightarrow}} \rfloor_\Sigma$.*

Proof: By lemma 2 $G_\Sigma/\sim^\zeta$ is in $\lceil \sim^\kappa \rceil_{\Sigma^\kappa}$. By lemma 3 $G_\Sigma/\sim^\zeta$ is in $\lfloor \overset{*}{\underset{E}{\Leftrightarrow}} \rfloor_\Sigma$. Now, suppose $g \not\sim^\zeta g'$ for some $g, g' \in G_\Sigma$, i.e. $c[g] \overset{*}{\underset{E}{\Leftrightarrow}} g_1 \not\sim^\kappa g_2 \overset{*}{\underset{E}{\Leftrightarrow}} c[g']$ for some $c \in G_\Sigma, g_1, g_2 \in G_{\Sigma^\kappa}$. Consider any $A \in \lceil \sim^\kappa \rceil_{\Sigma^\kappa} \cap \lfloor \overset{*}{\underset{E}{\Leftrightarrow}} \rfloor_\Sigma$, and assume $A \models g = g'$. Since $A \in \lfloor \overset{*}{\underset{E}{\Leftrightarrow}} \rfloor_\Sigma$, we have $A \models c[g] = g_1$ and $A \models c[g'] = g_2$. But then $A \models g_1 = g_2$ contradicting $A \in \lceil \sim^\kappa \rceil_\Sigma$. So $A \not\models g = g'$, and the theorem follows by fact 1. $\square$

This result shows that this semantics does indeed match our intention: $G_\Sigma/\sim^\zeta$ is the *least* computational model of E satisfying the inequations in $\sim^\kappa$, meaning that $\sim^\zeta$ is the congruence on G_Σ inducing two terms as semantically equal *unless* their inequality follows with necessity from principle (1) using E and $\sim^\kappa$.

We now generalise initial semantics to relative initial semantics.

Definition 8. *Let $RSP = \langle SP, SP^\kappa \rangle$ with $SP = \langle \Sigma, E \rangle$ and biased $SP^\kappa = \langle \Sigma^\kappa, E^\kappa \rangle$ be a relative specification, such that $\sim^\kappa$ on G_{Σ^κ} is the semantics specified by SP^κ. Consider the rewrite relation $\Re$ on G_Σ such that for all $g, g' \in G_\Sigma$*

$$g \;\Re\; g' \;\Leftrightarrow\; \begin{cases} g \underset{E}{\Leftrightarrow} g' \quad \textit{or} \\ \exists g_1, g_2 \in G_{\Sigma^\kappa}; c \in G_\Sigma \mid g = c[g_1],\; g' = c[g_2],\; g_1 \sim^\kappa g_2 \end{cases}$$

*Define $\sim^\alpha$ on G_Σ by $\sim^\alpha = \Re^*$ We call $\sim^\alpha$ the **initial semantics relative to** $\sim^\kappa$ specified by RSP. We refer to $\sim^\kappa$ as the **kernel** of $\sim^\alpha$.*

The relation $\sim^\alpha$ in definition 8 is clearly a congruence. The case $\Sigma^\kappa = \sim^\kappa = \emptyset$ and the case $\sim^\kappa = \overset{*}{\underset{E}{\Leftrightarrow}}_{G_{\Sigma^\kappa}}$ yield the initial semantics as defined in definition 4.

The way relative initial specification will be utilised in our examples is by giving semantics to constructors by way of a kernel and letting the axioms E give semantics to defined operators.

Example 4. We add a concatenation operator to AltSumList:

> **spec** $\mathsf{AltSumList_{op}}$ **initial**
> **relative to** AltSumList **final**
> **defined oprs** $\vdash\!\dashv$: list $\times$ list $\rightarrow$ list
> **axioms** $E_{\mathsf{AltSumList_{op}}}$: $\varepsilon \vdash\!\dashv \mathsf{r} = \varepsilon$,
> $(\mathsf{x} \dashv \mathsf{q}) \vdash\!\dashv \mathsf{r} = \mathsf{x} \dashv (\mathsf{q} \vdash\!\dashv \mathsf{r})$

○

Example 5. The specification $\mathsf{Int_{op}}$ from example 1 can be rephrased thus:

> **spec** $\mathsf{Int_{op}}$ **initial**
> **relative to** Int **initial**
> **sorts** int
> **defined oprs** $+, -$: int $\times$ int $\rightarrow$ int
> **axioms** x+0 = x,
> x − 0 = x,
> x+succ(y) = succ(x+y),
> x−succ(y) = pred(x−y),
> x+pred(y) = pred(x+y),
> x−pred(y) = succ(x−y)

> **spec** Int **initial**
> **sorts** int
> **constructors** 0 : int, succ, pred : int $\rightarrow$ int
> **axioms** succ(pred(x)) = x,
> pred(succ(x)) = x

○

In both initial and final generalisations the kernel may be final, initial, or even of some other bias or mode (see section 7). This accommodates the use of different strategies for defining equality over constructor terms. We have the initial dual of lemma 2:

Lemma 5. *$g \sim^\kappa g' \;\Rightarrow\; g \sim^\alpha g'$, i.e. all equalities in $\sim^\kappa$ are preserved in $\sim^\alpha$.*

And then it is straight forward to show:

Theorem 6. *The quotient $G_\Sigma/\!\sim^\alpha$ is initial in $\lfloor \overset{*}{\underset{E}{\Leftrightarrow}} \rfloor_\Sigma \cap \lfloor \sim^\kappa \rfloor_\Sigma$.*

So relative initial semantics specifies terms as semantically unequal unless their equality follows with necessity from E in conjunction with $\sim^\kappa$.

4.2 Relative Consistency

The postulate that a mathematical proposition cannot simultaneously be true and false is mirrored in predicate calculus by referring to a set of axioms as inconsistent when the predicate $\mathsf{true} = \mathsf{false}$ is deducible from the axioms. In a generalised view, consistency is related to basic understandings of the semantic domain which are expressed as requirements to the involved axioms and calculus. Guttag's notion of consistency in definition 2 thus presupposes the semantics of constructors specified by the designated set E_c, calling the whole set E inconsistent if E breaches these semantics. This act of viewing consistency always *relatively* to such presuppositions is made explicit by a notion of consistency in relative specification *relative to the kernel specification.*

Definition 9. *A relative specification RSP is* ***initially kernel preserving relative to*** $\sim^\kappa$ *if*

$$\sim^\kappa = \sim^\alpha_{G_{\Sigma^\kappa}}$$

where $\sim^\alpha$ *is the initial semantics specified by RSP relative to* $\sim^\kappa$*.*

The definition of consistency in definition 2 is the special case of definition 9 where $\sim^\kappa$ is a congruence on constructors specified base initially by some $E_c \subseteq E$.

Definition 10. *A relative specification RSP is* ***finally kernel preserving relative to*** $\sim^\kappa$ *if*

$$\sim^\zeta_{G_{\Sigma^\kappa}} = \sim^\kappa$$

where $\sim^\zeta$ *is the final semantics specified by RSP relative to* $\sim^\kappa$*.*

Failure of kernel preservation will be called ***kernel corruption***.

In the final case, several other notions related to consistency are feasible. It is for example natural to identify a signature $\Sigma_o \subseteq \Sigma$ of designated observers meant to specify final semantics by principle (1). Final inconsistency w.r.t. Σ_o would then mean there are defined operators in $\Sigma \setminus \Sigma_o$ whose specification in E gives further inequalities by principle (1) than intended. This is dealt with in [Han97].

A defined operator whose specification gives kernel corruption w.r.t. some presupposed semantics $\sim^\kappa$ on constructors cannot be interpreted as a function in any algebra containing $G_{\Sigma^\kappa}/{\sim^\kappa}$ as a sub-algebra. In this manner the notion of relative consistency generalises the notion of consistency in predicate logic which is equivalent to the nonexistence of any model for any inconsistent set of predicates. In predicate logic, the predefined semantics on $\{\mathsf{true},\mathsf{false}\}$ is $\mathsf{true}\neq\mathsf{false}$. In relative equational semantics, the kernel plays the role of predefined semantics.

However, note that a relative specification does have a model even if the specification is kernel corrupting. In order to investigate notions of consistency it is essential that there are semantics to be discussed in the event of inconsistency as well. This is safe as long as the semantics of an inconsistent specification bears witness of the inconsistency. Relative semantics supports this by its explicit kernel.

5 Modularity under Consistency

A crucial assumption of modular programming is the *context independency* of modules, in the sense that properties of submodules should be unaffected by their use in some larger context. In algebraic specification this is mirrored by the notion of 'conservative extension'. Algebras characterised by complex specifications should have models of the constituent sub-specifications as (intact) sub-algebras. Such algebras are dubbed "hierarchical models" in [WPP+83]. The analogue for relative specification are initial or final models which have the model of the kernel specification $G_{\Sigma^\kappa}/{\sim^\kappa}$ as a sub-algebra. This end is fulfilled by kernel preservation (two analogues, and analogues only, in [WPP+83] being 'hierarchy-conservativeness' and 'hierarchy-faithfulness').

This section investigates kernel preservation and modularity in some depth, resulting in the partial equivalence of generalised initial and final semantics.

5.1 Kernel Preservation

We start by showing the coincidence of initial and final kernel preservation.

Theorem 7. *Assume* `SC`. *Let RSP be a relative specification whose biased kernel has semantics $\sim^\kappa$. We have relatively to $\sim^\kappa$, that RSP is finally kernel preserving if and only if RSP is initially kernel preserving.*

Before proving this we make a few observations. The result is independent of `KC`, but note that the assumption of `KC` excludes a particularly blatant cause of kernel corruption different from the type caused by principle (1) on page 8.

Example 6. Consider the following specification:

> **spec** CorruptInt1 **initial**
> **relative to** Int **initial**
>   **sorts** int
>   **defined oprs** f : int $\to$ int
>   **axioms** f(0) = 0,
>   f(0) = succ(0)

or even:

> **spec** CorruptInt2 **initial**
> **relative to** Int **initial**
>   **sorts** int
>   **axioms** 0 = succ(0)

Both specifications give initial-semantically succ(0) $\sim^\alpha$ 0. The assumption of `KC` rules out such specifications as CorruptInt and CorruptInt2. ○

Note also that the coincidence in theorem 7 cannot be shown simply by establishing $\sim^{\zeta}_{G_{\Sigma^\kappa}} = \sim^{\alpha}_{G_{\Sigma^\kappa}}$, because initial and final corrupted kernel semantics do not coincide. This fact is the counterpart of lemmas 5 and 2. Final kernel corruption means

$$\sim^{\zeta}_{G_{\Sigma^\kappa}} \subset \sim^{\kappa}$$

whereas initial kernel corruption amounts to

$$\sim^{\kappa} \subset \sim^{\alpha}_{G_{\Sigma^\kappa}}$$

The two forms of kernel corruption are duals of each other—initial kernel corruption collapsing the kernel's congruence classes, and final kernel corruption composing new ones.

Example 7. Consider the specification

> **spec** $\mathsf{CorruptInt3}$
> **relative to** Int **initial**
> **sorts** int
> **defined oprs** $\mathsf{f} : \mathsf{int} \to \mathsf{int}$
> **axioms** $\mathsf{f(0) = 0}$,
> $\mathsf{f(succ(x)) = succ(f(x))}$,
> $\mathsf{f(pred(x)) = f(x)}$

Relative to the initial semantics given by Int, the specification $\mathsf{CorruptInt3}$ is both initially and finally kernel corrupting. Initially we have $\mathsf{succ(0)} \sim^{\alpha} \mathsf{0}$ and final-semantically we have $\mathsf{succ(pred(0))} \not\sim^{\zeta} \mathsf{0}$. The relations $\sim^{\alpha}$ and $\sim^{\zeta}$ are not identical. We have $\mathsf{succ(0)} \not\sim^{\zeta} \mathsf{0}$, and also $\mathsf{succ(pred(0))} \sim^{\alpha} \mathsf{0}$. ○

Now for the proof:

Proof of Lemma 7: Let $\sim^{\alpha}$ be the initial semantics relative to $\sim^{\kappa}$.

Assume RSP is not finally kernel preserving relative to $\sim^{\kappa}$. By lemma 2 there must exist $c \in G_{\Sigma}$ and $g_\kappa, g'_\kappa, g_1, g_2 \in G_{\Sigma^\kappa}$ such that

$$c[g_\kappa] \overset{*}{\underset{E}{\Leftrightarrow}} g_1 \not\sim^{\kappa} g_2 \overset{*}{\underset{E}{\Leftrightarrow}} c[g'_\kappa] \quad \text{but} \quad g_\kappa \sim^{\kappa} g'_\kappa$$

Since $g_\kappa \sim^{\kappa} g'_\kappa$ we have $g_\kappa \sim^{\alpha} g'_\kappa$ by lemma 5, giving $c[g_\kappa] \sim^{\alpha} c[g'_\kappa]$. Since $c[g_\kappa] \overset{*}{\underset{E}{\Leftrightarrow}} g_1$ and $g_2 \overset{*}{\underset{E}{\Leftrightarrow}} c[g'_\kappa]$, we then get $g_1 \sim^{\alpha} g_2$. But $g_1 \not\sim^{\kappa} g_2$, so E cannot be initially kernel preserving relative to $\sim^{\kappa}$.

For the converse implication, suppose first that $\mathtt{KC}$ does not hold. Then there are $g_\kappa, g'_\kappa \in G_{\Sigma^\kappa}$ such that

$$g_\kappa \overset{*}{\underset{E}{\Leftrightarrow}} g'_\kappa \quad \text{but} \quad g_\kappa \not\sim^{\kappa} g'_\kappa$$

But then we have $g_\kappa \not\sim^{\kappa} g'_\kappa \overset{*}{\underset{E}{\Leftrightarrow}} g_\kappa$, and E is trivially finally kernel corrupting.

So, assume KC. Suppose RSP is initially kernel corrupting relative to $\sim^\kappa$. By lemma 5 there are $g_\kappa, g'_\kappa \in G_{\Sigma^\kappa}$ such that

$$g_\kappa \sim^\alpha g'_\kappa \quad \text{but} \quad g_\kappa \not\sim^\kappa g'_\kappa$$

We now define

$$g \underset{\kappa}{\leftrightarrow} g' \quad \Leftrightarrow \quad \begin{cases} g \not\overset{*}{\underset{E}{\leftrightarrow}} g' \quad \text{and} \\ \exists c \in G_\Sigma; g_1, g_2 \in G_{\Sigma^\kappa} \mid g = c[g_1],\ g' = c[g_2],\ g_1 \sim^\kappa g_2 \end{cases}$$

Consider an arbitrary $\sim^\alpha$-derivation $\langle g_\kappa, \ldots, g'_\kappa \rangle$ in G_Σ. By KC we must have $g_\kappa \not\overset{*}{\underset{E}{\leftrightarrow}} g'_\kappa$, so there must be at least one $\underset{\kappa}{\leftrightarrow}$-step in the derivation, and since $g_\kappa \not\sim^\kappa g'_\kappa$, there must also be at least one $\underset{E}{\leftrightarrow}$-step in the derivation. We may assume that this latter step is such that $g \underset{E}{\leftrightarrow} g'$ where at least one of g, g' is not in G_{Σ^κ}; otherwise this step would by KC be a $\underset{\kappa}{\leftrightarrow}$-step. There is, then, a term $g \notin G_{\Sigma^\kappa}$ in the derivation. But then there must exist *two* $\underset{E}{\leftrightarrow}$-steps; since a $\underset{\kappa}{\leftrightarrow}$-step cannot link a term in G_{Σ^κ} with a term *not* in G_{Σ^κ} (and $g_\kappa \neq g'_\kappa$, since $\sim^\kappa$ is reflexive). Both these two $\underset{E}{\leftrightarrow}$-steps must be linking terms in G_{Σ^κ} with terms *not* in G_{Σ^κ}. The derivation must therefore be of the form:

$$\begin{array}{l} g_\kappa \overset{*}{\underset{\kappa}{\leftrightarrow}} g_{\kappa 0} \overset{+}{\underset{E}{\leftrightarrow}} c_1[g_1] \overset{*}{\underset{\kappa}{\leftrightarrow}} c_1[g'_1] \overset{*}{\underset{E}{\leftrightarrow}} \cdots \\ \qquad \vdots \\ \cdots \overset{*}{\underset{E}{\leftrightarrow}} c_k[g_k] \overset{+}{\underset{\kappa}{\leftrightarrow}} c_k[g'_k] \overset{*}{\underset{E}{\leftrightarrow}} c_{k+1}[g_{k+1}] \overset{+}{\underset{\kappa}{\leftrightarrow}} c_{k+1}[g'_{k+1}] \overset{*}{\underset{E}{\leftrightarrow}} \cdots \\ \qquad \vdots \\ \cdots \overset{*}{\underset{E}{\leftrightarrow}} c_n[g_n] \overset{*}{\underset{\kappa}{\leftrightarrow}} c_n[g'_n] \overset{+}{\underset{E}{\leftrightarrow}} g_{\kappa n} \overset{*}{\underset{\kappa}{\leftrightarrow}} g'_\kappa \end{array}$$

for $g_{\kappa 0}, g_{\kappa n}, g_i, g'_i \in G_{\Sigma^\kappa}$ and $c_i \in G_\Sigma$; $1 \leq i \leq n$. By SC we have

$$c_i[g'_i] \overset{*}{\underset{E}{\leftrightarrow}} g_{\kappa i} \overset{*}{\underset{E}{\leftrightarrow}} c_{i+1}[g_{i+1}]$$

for some $g_{\kappa i} \in G_{\Sigma^\kappa}$; $1 \leq i \leq n$. Now, we cannot have $g_{\kappa 0} \sim^\kappa \cdots \sim^\kappa g_{\kappa k} \sim^\kappa \cdots \sim^\kappa g_{\kappa n}$, since this would entail $g_\kappa \sim^\kappa g'_\kappa$. Therefore there must exist a $0 \leq l < n$ such that $g_{\kappa l} \not\sim^\kappa g_{\kappa l+1}$. But then we have

$$c_{l+1}[g_{l+1}] \overset{*}{\underset{E}{\leftrightarrow}} g_{\kappa l} \not\sim^\kappa g_{\kappa l+1} \overset{*}{\underset{E}{\leftrightarrow}} c_{l+1}[g'_{l+1}]$$

Since $g_{l+1} \sim^\kappa g'_{l+1}$, RSP is finally kernel corrupting relative to $\sim^\kappa$. □

Based on proposition 7, the relationship between final and initial kernel preservation under KC (and for the case '$n = 1$' in the proof) is illustrated in figure 1.

Final kernel preservation is trivial relative to an identity kernel:

Observation 8. *If a final semantics is a congruence and its kernel $\sim^\kappa$ is an identity relation then lemma 2 gives final kernel preservation relative to $\sim^\kappa$.*

Hence we get the following perhaps obvious corollary from proposition 7 and observation 8.

$$
\begin{tikzcd}
g_1 \arrow[r, leftrightarrow, "*", "E"'] & c[g_\kappa] \arrow[r, leftrightarrow, "*", "\kappa"'] \arrow[d, leftrightarrow, "*"', "E"] & c[g'_\kappa] \arrow[r, leftrightarrow, "*", "E"'] \arrow[d, leftrightarrow, "E"', "*"] & g_2 \\
 & g_1 \arrow[ul, dashed, bend left] \arrow[r, phantom, "\not\sim^{\kappa}"] & g_2 \arrow[ur, dashed, bend right] &
\end{tikzcd}
$$

(a)

$$
\begin{tikzcd}
 & c[g_\kappa] \arrow[r, leftrightarrow, "*", "\kappa"'] \arrow[d, leftrightarrow, "E"', "*"] \arrow[dl, dashed, bend right] & c[g'_\kappa] \arrow[d, leftrightarrow, "*"', "E"] \arrow[dr, dashed, bend left] & \\
c[g_\kappa] \arrow[r, leftrightarrow, "*", "E"'] & g_1 \arrow[r, phantom, "\not\sim^{\kappa}"] & g_2 \arrow[r, leftrightarrow, "*", "E"'] & c[g'_\kappa]
\end{tikzcd}
$$

(b)

Figure 1: Duality. Kernel corruption from two points of view. Final and initial kernel preservation are under KC two manifestations of the same phenomenon. In *(a)* inconsistency declares itself as initial kernel corruption, in that $g_1 \not\sim^{\kappa} g_2$, but $g_1 \sim^{\alpha} g_2$. In *(b)* inconsistency manifests itself in final kernel corruption, in that $g_\kappa \not\sim^{\zeta} g'_\kappa$, but $g_\kappa \sim^{\kappa} g'_\kappa$.

Corollary 9. *Assume* SC*. If the final semantics is a congruence and its kernel $\sim^\kappa$ is the identity relation then we also have initial kernel preservation relative to $\sim^\kappa$.*

That is, a specification cannot corrupt a kernel without the kernel providing a situation where the term-universe is in a many-to-one correspondence with the semantic value space.

Although the matter is not pursued further here, the above results based upon sufficient completeness have interesting formulations as partial results corresponding to assumptions of partial sufficient completeness. Such partial results are interesting e.g. in connection with hidden sorts and symbols and also in connection with partial specifications.

5.2 Separability

Kernel preservation ensures conservativeness in a *hierarchical* sense. Relative semantics arise from two component semantics—the kernel and that given by some equation set E. Under KC and SC, kernel preservation also ensures the preservation of the semantics on *all* terms of kernel sorts, i.e. also the part specified by E. Consider the following relation:

$$(\sim^\kappa \cup \overset{*}{\underset{E}{\Leftrightarrow}}{}_{G_\Sigma})^* \quad \text{[i.e. the reflexive-transitive closure of the union]} \qquad (2)$$

The closure under monotonicity (context application) of this relation is just the relative initial semantics of definition 8. However with monotonicity stripped away, there is no "interaction" between the two components $\sim^\kappa$ and $\overset{*}{\underset{E}{\Leftrightarrow}}_{G_\Sigma}$. Intuitively then, under `SC` and `KC`, $(\sim^\kappa \cup \overset{*}{\underset{E}{\Leftrightarrow}}_{G_\Sigma})^*$ is *separable*, i.e. gives precisely the pure sum of the semantics of its two components. More precisely, it is straight-forward to prove the ***separability*** of $(\sim^\kappa \cup \overset{*}{\underset{E}{\Leftrightarrow}}_{G_\Sigma})^*$:

Lemma 10. *Assume* `SC` *and* `KC`*. For a congruence $\sim^\kappa$ on G_{Σ^κ} and a set of equations E, we have*

1. $(\sim^\kappa \cup \overset{*}{\underset{E}{\Leftrightarrow}}_{G_\Sigma})^*_{G_{\Sigma^\kappa}} = \sim^\kappa$ *(preservation of $\sim^\kappa$)*

2. $\overset{*}{\underset{E}{\Leftrightarrow}}_{G_\Sigma} \subseteq (\sim^\kappa \cup \overset{*}{\underset{E}{\Leftrightarrow}}_{G_\Sigma})^*$ *(preservation of $\overset{*}{\underset{E}{\Leftrightarrow}}$. Completeness)*

3. $g(\sim^\kappa \cup \overset{*}{\underset{E}{\Leftrightarrow}}_{G_\Sigma})^* g_\kappa \;\Rightarrow\; g \overset{*}{\underset{E}{\Leftrightarrow}} g'_\kappa$ *(preservation of $\overset{*}{\underset{E}{\Leftrightarrow}}$. Soundness 1)*
   for any $g \in G_\Sigma$ and $g_\kappa \in G_{\Sigma^\kappa}$, for some $g'_\kappa \in G_{\Sigma^\kappa}$ such that $g_\kappa \sim^\kappa g'_\kappa$.

4. $g(\sim^\kappa \cup \overset{*}{\underset{E}{\Leftrightarrow}}_{G_\Sigma})^* g' \;\Rightarrow\; g \overset{*}{\underset{E}{\Leftrightarrow}} g'$ *(preservation of $\overset{*}{\underset{E}{\Leftrightarrow}}$. Soundness 2)*
   for any $g, g' \in G_\Sigma$ of sorts not in Σ^κ.

Proof: *1*: Suppose $(\sim^\kappa \cup \overset{*}{\underset{E}{\Leftrightarrow}}_{G_\Sigma})^*_{G_{\Sigma^\kappa}} \neq \sim^\kappa$. Since it is evident that $\sim^\kappa \subseteq (\sim^\kappa \cup \overset{*}{\underset{E}{\Leftrightarrow}}_{G_\Sigma})^*_{G_{\Sigma^\kappa}}$, there must exist $g_\kappa, g'_\kappa \in G_{\Sigma^\kappa}$ such that $g_\kappa \not\sim^\kappa g'_\kappa$, in spite of $g_\kappa(\sim^\kappa \cup \overset{*}{\underset{E}{\Leftrightarrow}}_{G_\Sigma})^* g'_\kappa$. But then we must have

$$g_\kappa \sim^\kappa g_1 \overset{*}{\underset{E}{\Leftrightarrow}} g'_1 \sim^\kappa g_2 \overset{*}{\underset{E}{\Leftrightarrow}} g'_2 \sim^\kappa \cdots \sim^\kappa g_i \overset{*}{\underset{E}{\Leftrightarrow}} g'_i \sim^\kappa \cdots \sim^\kappa g_n \overset{*}{\underset{E}{\Leftrightarrow}} g'_n \sim^\kappa g'_\kappa$$

for some $g_i, g'_i \in G_{\Sigma^\kappa}; 1 \leq i \leq n$. By `KC` $g_i \sim^\kappa g'_i$ for every $1 \leq i \leq n$, and then $g_\kappa \sim^\kappa g'_\kappa$, which is a contradiction.

2: Obvious.

3: Suppose $g(\sim^\kappa \cup \overset{*}{\underset{E}{\Leftrightarrow}}_{G_\Sigma})^* g_\kappa$ for some $g \in G_\Sigma, g_\kappa \in G_{\Sigma^\kappa}$. Observe that g must be of the same sort as g_κ. By `SC` we then have $g \overset{*}{\underset{E}{\Leftrightarrow}} g'_\kappa$ for a $g'_\kappa \in G_{\Sigma^\kappa}$. Then

$$g_\kappa(\sim^\kappa \cup \overset{*}{\underset{E}{\Leftrightarrow}}_{G_\Sigma})^* g \overset{*}{\underset{E}{\Leftrightarrow}} g'_\kappa$$

i.e. $g_\kappa(\sim^\kappa \cup \overset{*}{\underset{E}{\Leftrightarrow}}_{G_\Sigma})^* g'_\kappa$. And then *(1)* gives $g_\kappa \sim^\kappa g'_\kappa$.

4: Suppose $g(\sim^\kappa \cup \overset{*}{\underset{E}{\Leftrightarrow}}_{G_\Sigma})^* g'$ for some $g, g' \in G_\Sigma$. Observe that g must be of the same sort as g', and that every component in any $(\sim^\kappa \cup \overset{*}{\underset{E}{\Leftrightarrow}}_{G_\Sigma})^*$-derivation $\langle g, \ldots, g' \rangle$ is of the same sort as g, g'.

Observe also that every $(\sim^x \cup \overset{*}{\underset{E}{\Leftrightarrow}}{}_{G_\Sigma})^*$-derivation step has the form $g_i = g_x \sim^\kappa g'_x = g_{i+1}$, for $g_\kappa, g'_\kappa \in G_{\Sigma^\kappa}$; or the form $g_i \underset{E}{\Leftrightarrow} g_{i+1}$ (for $g_i, g_{i+1} \in G_\Sigma$).

Let S^κ denote the set of sorts of terms occurring in G_{Σ^κ}. Suppose g, g' are not of any type in S^κ. But then we cannot have $g_i \sim^\kappa g_{i+1}$ for any components g_i, g_{i+1} in any $(\sim^\kappa \cup \overset{*}{\underset{E}{\Leftrightarrow}}{}_{G_\Sigma})^*$-derivation $\langle g, \ldots, g' \rangle$. Consequently $g \overset{*}{\underset{E}{\Leftrightarrow}} g'$, and the lemma follows. □

Now we may use $(\sim^\kappa \cup \overset{*}{\underset{E}{\Leftrightarrow}}{}_{G_\Sigma})^*$ to show that initial and final semantics are separable in kernel sorts under kernel preservation; i.e. the components of relative semantics in kernel sorts are untouched. The following two theorems are easily proven:

Theorem 11. *Assume* KC *and* SC*. Let $RSP = \langle SP, SP^\kappa \rangle$ be a relative specification with $SP = \langle \Sigma, E \rangle$ and biased $SP^\kappa = \langle \Sigma^\kappa, E^\kappa \rangle$, such that $\sim^\kappa$ on G_{Σ^κ} is the semantics specified by SP^κ. Then for S^κ the sorts in Σ^κ we have:*

$$(\sim^\kappa \cup \overset{*}{\underset{E}{\Leftrightarrow}}{}_{G_\Sigma})^*_{S^\kappa} = \sim^\alpha_{S^\kappa}$$
$$\Updownarrow$$
$$RSP \text{ is initially kernel preserving relative to } \sim^\kappa$$

Proof: If RSP is initially kernel corrupting relative to $\sim^\kappa$ then $(\sim^\kappa \cup \overset{*}{\underset{E}{\Leftrightarrow}}{}_{G_\Sigma})^*_{S^\kappa} \neq \sim^\alpha_{S^\kappa}$ by proposition 10*(1)*.

So, assume initial kernel preservation. Clearly $(\sim^\kappa \cup \overset{*}{\underset{E}{\Leftrightarrow}}{}_{G_\Sigma})^*_{S^\kappa} \subseteq \sim^\alpha_{S^\kappa}$, so we concentrate on showing $(\sim^\kappa \cup \overset{*}{\underset{E}{\Leftrightarrow}}{}_{G_\Sigma})^*_{S^\kappa} \supseteq \sim^\alpha_{S^\kappa}$. We induce on the length n of an arbitrary $\sim^\alpha_{S^\kappa}$-derivation $\langle g, \ldots, g' \rangle$ in G_Σ for $g, g' \in G_{\Sigma S^\kappa}$:

$n = 1$: Trivially $g(\sim^\kappa \cup \overset{*}{\underset{E}{\Leftrightarrow}}{}_{G_\Sigma})^* g$.

$n = k + 1; k \geq 1$: We then have a derivation $\langle g, \ldots, g_k, g' \rangle$. Since g, g_k must be of a sort in S^κ, the induction hypothesis gives $g(\sim^\kappa \cup \overset{*}{\underset{E}{\Leftrightarrow}}{}_{G_\Sigma})^* g_k$. Suppose $g_k \underset{E}{\Leftrightarrow} g'$. Then $g_k(\sim^\kappa \cup \overset{*}{\underset{E}{\Leftrightarrow}}{}_{G_\Sigma})^* g'$ and the theorem follows. Suppose $g_k = c[g'_k]$ and $c[g''] = g'$ and $g'_k \sim^\kappa g''$, for some $g'_k, g'' \in G_{\Sigma^\kappa}$. Since $c[g'_k]$ and $c[g'']$ are of some sort in S^κ, SC gives $c[g'_k] \overset{*}{\underset{E}{\Leftrightarrow}} g_\kappa$ and $c[g''] \overset{*}{\underset{E}{\Leftrightarrow}} g'_\kappa$ for some $g_\kappa, g'_\kappa \in G_{\Sigma^\kappa}$. Assuming initial kernel preservation or the equivalent notion of final kernel preservation (proposition 7) we have

$$\neg \exists c' \in G_\Sigma; g_1, g_2 \in G_{\Sigma^\kappa} \mid c'[g'_k] \overset{*}{\underset{E}{\Leftrightarrow}} g_1 \not\sim^\kappa g_2 \overset{*}{\underset{E}{\Leftrightarrow}} c'[g'']$$

But then $g_k = c[g'_k] \overset{*}{\underset{E}{\Leftrightarrow}} g_\kappa \sim^\kappa g'_\kappa \overset{*}{\underset{E}{\Leftrightarrow}} c[g''] = g'$, i.e. $g_k(\sim^\kappa \cup \overset{*}{\underset{E}{\Leftrightarrow}}{}_{G_\Sigma})^* g'$. □

Theorem 12. *Assume* KC *and* SC*. Let $RSP = \langle SP, SP^\kappa \rangle$ be a relative specification with $SP = \langle \Sigma, E \rangle$ and biased $SP^\kappa = \langle \Sigma^\kappa, E^\kappa \rangle$, such that $\sim^\kappa$ on G_{Σ^κ} is the semantics specified by SP^κ. Then for S^κ the sorts in Σ^κ we have:*

$$(\sim^\kappa \cup \overset{*}{\underset{E}{\Leftrightarrow}}{}_{G_\Sigma})^*_{S^\kappa} = \sim^\zeta_{S^\kappa}$$
$$\Updownarrow$$
$$RSP \text{ is finally kernel preserving relative to } \sim^\kappa$$

Proof: If RSP is finally kernel corrupting relatively to $\sim^\kappa$ then $(\sim^\kappa \cup \overset{*}{\underset{E}{\Leftrightarrow}}{}_{G_\Sigma})^*_{S^\kappa} \neq \sim^\omega_{S^\kappa}$ by proposition 10*(1)*.

Assume final kernel preservation. Suppose $g(\sim^\kappa \cup \overset{*}{\underset{E}{\Leftrightarrow}}_{G_\Sigma})^* g'$ for arbitrary $g, g' \in G_{\Sigma S^\kappa}$. We show $g \sim^\omega g'$ by induction on the length n of an arbitrary $(\sim^\kappa \cup \overset{*}{\underset{E}{\Leftrightarrow}}_{G_\Sigma})^*_{S^\kappa}$-derivation $\langle g, \ldots, g' \rangle$.

$n = 1$: By KC and SC and fact 1 $\sim^\omega$ is reflexive, so $g \sim^\omega g$.

$n = k + 1; k \geq 1$: We then have a derivation $\langle g, \ldots, g_k, g' \rangle$. Since g, g_k must be of a sort in S^κ, the induction hypothesis gives $g \sim^\omega g_k$. Any $(\sim^\kappa \cup \overset{*}{\underset{E}{\Leftrightarrow}}_{G_\Sigma})^*$-derivation step has the form $g_i = g_\kappa \sim^\kappa g'_\kappa = g_{i+1}$, for $g_\kappa, g'_\kappa \in G_{\Sigma^\kappa}$; or the form $g_i \underset{E}{\Leftrightarrow} g_{i+1}$ (for $g_i, g_{i+1} \in G_{\Sigma S^\kappa}$). Suppose $g_k = g_\kappa \sim^\kappa g'_\kappa = g'$ for $g_\kappa, g'_\kappa \in G_{\Sigma^\kappa}$: Since SP is finally kernel preserving we have $g_k = g_\kappa \sim^\omega_{S^\kappa} g'_\kappa = g'$, and the theorem follows. Suppose $g_k \underset{E}{\Leftrightarrow} g'$: By lemma 3 $g_k \sim^\omega g'$ and since g, g_k are of some sort in S^κ, we get $g_k \sim^\omega g'$.

Now suppose $g \sim^\omega g'$ for arbitrary $g, g' \in G_{\Sigma S^\kappa}$. By SC we have $g \overset{*}{\underset{E}{\Leftrightarrow}} g_\kappa$ and $g' \overset{*}{\underset{E}{\Leftrightarrow}} g'_\kappa$ for $g_\kappa, g'_\kappa \in G_{\Sigma^\kappa}$. Since $g \sim^\omega g'$, we have

$$\neg \exists c \in G_\Sigma; g_1, g_2 \in G_{\Sigma^\kappa} \mid c[g] \overset{*}{\underset{E}{\Leftrightarrow}} g_1 \not\sim^\kappa g_2 \overset{*}{\underset{E}{\Leftrightarrow}} c[g']$$

In particular we have $g \overset{*}{\underset{E}{\Leftrightarrow}} g_\kappa \sim^\kappa g'_\kappa \overset{*}{\underset{E}{\Leftrightarrow}} g'$ thus giving $g(\sim^\kappa \cup \overset{*}{\underset{E}{\Leftrightarrow}}_{G_\Sigma})^* g'$. □

Finally, we get the following essential property as a corollary of theorems 11 and 12:

Theorem 13. *Assume* KC *and* SC. *Under kernel preservation we have that initial and final semantics coincide in kernel sorts:*

$$\sim^\alpha_{S^\kappa} = \sim^\zeta_{S^\kappa}$$

Before ending this section let us return briefly to the non-relative final semantics of definition 5. The analogue of this corollary for a standard non-relative specification $SP = \langle \Sigma, E \rangle$, is that for E convergent and sufficiently complete

$$\sim^{\mathcal{Q}}_{S^\kappa} = \overset{*}{\underset{E}{\Leftrightarrow}}_{S^\kappa}$$

where S^κ denotes designated initial sorts [Lys92]. This holds regardless of inconsistency. This is roughly because in a monolithic definition such as definition 5, at finality the implicit "kernel" is corrupted in the initial direction (see fig. 1) *before* the mechanism of extracting inequalities sets in. In contrast, the relative definition corrupts the kernel *in the process of* extracting inequalities, hence corrupting the kernel in the final direction. In the relative case, inconsistency reveals itself in the exact symmetrical duals of kernel corruption. So on a higher level, proposition 7 and theorem 13 give the equivalence of initial and final semantics on kernel sorts, also for relative semantics.

Under kernel preservation/consistency one can show $\sim^{\mathcal{Q}} = \sim^\zeta$ (where the kernel of $\sim^\zeta$ is the implicit $\overset{*}{\underset{E_c}{\Leftrightarrow}}_{G_{\Sigma_c}}$), by using the separable relation $(\overset{*}{\underset{E_c}{\Leftrightarrow}}_{G_{\Sigma_c}} \cup \overset{*}{\underset{E}{\Leftrightarrow}}_{G_\Sigma})^*$.

5.3 Inconsistency revealed as referential opacity

According to proposition 10*(1)* it may seem that the relation $(\sim^\kappa \cup \overset{*}{\underset{E}{\Leftrightarrow}}_{G_\Sigma})^*$ is in a sense insensitive to inconsistency. However, even though $(\sim^\kappa \cup \overset{*}{\underset{E}{\Leftrightarrow}}_{G_\Sigma})^*$ always preserves its components, inconsistency now reveals itself through other symptoms; namely lack of monotonicity under context application, in other words $(\sim^\kappa \cup \overset{*}{\underset{E}{\Leftrightarrow}}_{G_\Sigma})^*$ becomes referentially opaque.

Theorem 14. *Assume* `KC` *and* `SC`*. Let* $\sim^\zeta$ *be the final semantics specified by a relative biased specification* RSP *relative to* $\sim^\kappa$ *on* G_{Σ^κ}*. We then have for* S^κ *the sorts in* Σ^κ*:*

$$\textit{RSP is finally kernel corrupting relative to } \sim^\kappa$$
$$\Updownarrow$$
$$(\sim^\kappa \cup \overset{*}{\underset{E}{\Leftrightarrow}}_{G_\Sigma})^*_{S^\kappa} \textit{ is not monotone w.r.t. context application}$$

Proof: Suppose RSP is finally kernel corrupting. By observation 2 there must be $c \in G_\Sigma$ and $g_\kappa, g'_\kappa, g_1, g_2 \in G_{\Sigma^\kappa}$ such that

$$c[g_\kappa] \overset{*}{\underset{E}{\Leftrightarrow}} g_1 \not\sim^\kappa g_2 \overset{*}{\underset{E}{\Leftrightarrow}} c[g'_\kappa] \quad \text{but} \quad g_\kappa \sim^\kappa g'_\kappa$$

Since $g_\kappa \sim^\kappa g'_\kappa$ proposition 10*(1)* gives $g_\kappa(\sim^\kappa \cup \overset{*}{\underset{E}{\Leftrightarrow}}_{G_\Sigma})^* g'_\kappa$. Now, suppose

$$c[g_\kappa](\sim^\kappa \cup \overset{*}{\underset{E}{\Leftrightarrow}}_{G_\Sigma})^* c[g'_\kappa]$$

It cannot be the case that $c[g_\kappa] \overset{*}{\underset{E}{\Leftrightarrow}} c[g'_\kappa]$ since this would give $g_1 \overset{*}{\underset{E}{\Leftrightarrow}} g_2$ which contradicts `KC`. Any $(\sim^\kappa \cup \overset{*}{\underset{E}{\Leftrightarrow}}_{G_\Sigma})^*$-derivation $\langle c[g_\kappa], \ldots, c[g'_\kappa]\rangle$ in G_Σ must therefore have the form

$$\begin{array}{l} c[g_\kappa] \overset{*}{\underset{E}{\Leftrightarrow}} g_{\kappa 1} \sim^\kappa g_{\kappa 1}' \overset{*}{\underset{E}{\Leftrightarrow}} g_{\kappa 2} \sim^\kappa g_{\kappa 2}' \overset{*}{\underset{E}{\Leftrightarrow}} \cdots \\ \qquad\qquad\qquad\qquad \vdots \\ \qquad\qquad\qquad \cdots \overset{*}{\underset{E}{\Leftrightarrow}} g_{\kappa i} \sim^\kappa g_{\kappa i}' \overset{*}{\underset{E}{\Leftrightarrow}} \cdots \\ \qquad\qquad\qquad\qquad \vdots \\ \qquad\qquad\qquad\qquad \cdots \overset{*}{\underset{E}{\Leftrightarrow}} g_{\kappa n} \sim^\kappa g_{\kappa n}' \overset{*}{\underset{E}{\Leftrightarrow}} c[g'_\kappa] \end{array} \tag{3}$$

for $n \geq 2$ and $g_{\kappa i}, g_{\kappa i}' \in G_{\Sigma^\kappa}$; $1 \leq i \leq n$. By `KC` $g_{\kappa i}' \sim^\kappa g_{\kappa i+1}$ for $1 \leq i \leq n$. But since $g_1 \overset{*}{\underset{E}{\Leftrightarrow}} c[g_\kappa]$ and $c[g'_\kappa] \overset{*}{\underset{E}{\Leftrightarrow}} g_2$ we get $g_1 \sim^\kappa g_2$ which is a contradiction. Consequently, and since $c[g_\kappa], c[g'_\kappa]$ must be of type S^κ, $c[g_\kappa](\sim^\kappa \not\cup \overset{*}{\underset{E}{\Leftrightarrow}}_{G_\Sigma})^*_{S^\kappa} c[g'_\kappa]$, so $(\sim^\kappa \cup \overset{*}{\underset{E}{\Leftrightarrow}}_{G_\Sigma})^*_{S^\kappa}$ is not monotone w.r.t. context application.

Conversely, suppose that $g(\sim^\kappa \cup \overset{*}{\underset{E}{\Leftrightarrow}}_{G_\Sigma})^* g'$ but $c[g](\sim^\kappa \not\cup \overset{*}{\underset{E}{\Leftrightarrow}}_{G_\Sigma})^* c[g']$ for some $c[g], c[g'] \in G_{\Sigma S^\kappa}$. By `SC` we have $g \overset{*}{\underset{E}{\Leftrightarrow}} g_\kappa$ and $g' \overset{*}{\underset{E}{\Leftrightarrow}} g'_\kappa$ for $g_\kappa, g'_\kappa \in G_{\Sigma^\kappa}$. Hence, $g_\kappa(\sim^\kappa \cup \overset{*}{\underset{E}{\Leftrightarrow}}_{G_\Sigma})^* g'_\kappa$ and by proposition 10*(1)*

$$g_\kappa \sim^\kappa g'_\kappa$$

By `SC` we also have $c[g] \overset{*}{\underset{E}{\Leftrightarrow}} g_1$ and $c[g'] \overset{*}{\underset{E}{\Leftrightarrow}} g_2$ for $g_1, g_2 \in G_{\Sigma^\kappa}$. Now, we cannot have $g_1 \sim^\kappa g_2$, because then $c[g](\sim^\kappa \cup \overset{*}{\underset{E}{\Leftrightarrow}}_{G_\Sigma})^* c[g']$. Accordingly,

$$c[g] \overset{*}{\underset{E}{\Leftrightarrow}} g_1 \not\sim^\kappa g_2 \overset{*}{\underset{E}{\Leftrightarrow}} c[g']$$

and then also

$$c[g_\kappa] \overset{*}{\underset{E}{\Leftrightarrow}} c[g] \overset{*}{\underset{E}{\Leftrightarrow}} g_1 \not\sim^\kappa g_2 \overset{*}{\underset{E}{\Leftrightarrow}} c[g'] \overset{*}{\underset{E}{\Leftrightarrow}} c[g'_\kappa]$$

So, $g_x \not\sim^\omega g'_x$, but $g_\kappa \sim^\kappa g'_\kappa$, thus RSP is finally kernel corrupting. □

And of course we have the dual by proposition 7:

Corollary 15. *Assume* KC *and* SC. *Let* $\sim^\alpha$ *be the initial semantics specified by a biased relative specification RSP relative to* $\sim^\kappa$ *on* G_{Σ^κ}. *We then have for* S^κ *the sorts in* Σ^κ*:*

$$\text{RSP is initially kernel corrupting relative to } \sim^\kappa$$
$$\Updownarrow$$
$$(\sim^\kappa \cup \overset{*}{\underset{E}{\Leftrightarrow}}_{G_\Sigma})^*_{S^\kappa} \text{ is not monotone w.r.t. context application}$$

Example 8. For the specification CorruptInt3 from example 7 we had initial-semantically $\mathsf{f(succ(pred(0)))} \sim^\alpha \mathsf{succ(0)}$ and $\mathsf{f(0)} \sim^\alpha \mathsf{0}$ giving initial kernel corruption by $\mathsf{succ(0)} \sim^\alpha \mathsf{0}$. Consider the associated relation $(\sim^\kappa \cup \overset{*}{\underset{E}{\Leftrightarrow}}_{G_\Sigma})^*$. We have

$$\mathsf{succ(pred(0))}\ (\sim^\kappa \cup \overset{*}{\underset{E}{\Leftrightarrow}}_{G_\Sigma})^*\ \mathsf{0},$$

but

$$\mathsf{f(succ(pred(0)))}\ (\sim^\kappa \not\cup \overset{*}{\underset{E}{\Leftrightarrow}}_{G_\Sigma})^*\ \mathsf{f(0)}.$$

○

6 Proof by Kernel Preservation

The formalism of relative semantics can now be utilised to describe a modular hierarchical approach to proof by consistency, in which at each stage, only the uppermost equations in the hierarchy need be considered for completion. So, given a relative semantics $\sim^r$ we pose the problem of determining whether $G_\Sigma/\sim^r \models u = v$ for any given equation $u = v$ (which is undecidable in general, of course). Although the following result is stated for standard completion and assumes orientability, it also holds for ordered completion assuming only orientability of ground instances. This is evident by examining the proof, since peak elimination is done w.r.t. ground-term derivations. It also holds for extended completion if all unorientable equations are in kernel terms, i.e. over $T_{\Sigma^\kappa}(X)$.

Theorem 16. *Let* $\sim^\zeta$ *be the final semantics specified by a relative biased specification RSP relative to* $\sim^\kappa$ *on* G_{Σ^κ} *under assumption of* KC *and* SC. *Let* H *be a set of equational hypotheses. Suppose a completion sequence* $\langle\langle E \cup H, \emptyset\rangle, \ldots, \langle E_i, R_i\rangle, \ldots\rangle$ *is fair w.r.t. inferences and is s.t. every equation generated is orientable. Assume furthermore that every term in* $T_\Sigma(X) \setminus T_{\Sigma^\kappa}(X)$ *is greater in the completion process's term ordering than all terms in* $T_{\Sigma^\kappa}(X)$. *Then:*

$$G_\Sigma/\sim^\zeta \not\models H$$
$$\Updownarrow$$
$$\text{an equation or rule } \langle l, r\rangle \text{ over } T_{\Sigma^\kappa}(X) \text{ is generated during the process s.t.}$$
$$G_{\Sigma^\kappa}/\sim^\kappa \not\models l = r$$

Proof: Assume $G_\Sigma/{\sim^\zeta} \not\models H$, i.e. there is some $u = v \in H$, ground-instantiation τ and context c such that $c[u\tau] \overset{*}{\underset{E}{\Leftrightarrow}} g_1 \not\sim^\kappa g_2 \overset{*}{\underset{E}{\Leftrightarrow}} c[v\tau]$. But then, $g_1 \overset{*}{\underset{E\cup\{u=v\}}{\longleftrightarrow}} g_2$. Regarding completion processes as derivation transforming processes in the style of Bachmair [Bac91], any derivation $\langle g_1, \ldots, g_2\rangle$ may be transformed into a derivation $\langle g_1, \ldots, g_2\rangle'$ where every component is a term in G_{Σ^κ} (otherwise, by assumption of the term ordering, there would be peak in the derivation). This entails the generation of a set $E_j \cup R_j$ purely over $T_{\Sigma^\kappa}(X)$ such that $g_1 \overset{*}{\underset{E_j\cup R_j}{\longleftrightarrow}} g_2$. But then at least one of $\langle l, r\rangle \in E_j \cup R_j$ must be s.t. $G_{\Sigma^\kappa}/{\sim^\kappa} \not\models l = r$, or else $g_1 \sim^\kappa g_2$, contradicting our assumption.

Conversely, assume an equation or rule $\langle l, r\rangle$ over $T_{\Sigma^\kappa}(X)$ is generated during the process s.t. for some instantiation σ_κ, $l\sigma_\kappa \not\sim^\kappa r\sigma_\kappa$. By KC, $l\sigma_\kappa \overset{*}{\underset{E}{\not\Leftrightarrow}} r\sigma_\kappa$. Then, every $E\cup H$-derivation must be of the form

$$l\sigma_\kappa \overset{*}{\underset{E}{\Leftrightarrow}} c_1[u_1\sigma_1], c_1[v_1\sigma_1] \overset{*}{\underset{E}{\Leftrightarrow}} \cdots \overset{*}{\underset{E}{\Leftrightarrow}} c_n[u_n\sigma_n], c_n[v_n\sigma_n] \overset{*}{\underset{E}{\Leftrightarrow}} r\sigma_\kappa$$

(or any u_i, v_i-symmetrical form), for $\langle u_i, v_i\rangle \in H; 1 \le i \le n$. By SC, for every i there are $g_i, g_i' \in G_{\Sigma^\kappa}$ s.t. $c_i[u_i\sigma_i] \overset{*}{\underset{E}{\Leftrightarrow}} g_i$ and $c_i[v_i\sigma_i] \overset{*}{\underset{E}{\Leftrightarrow}} g_i'$. There must be some $1 \le l \le n$ s.t. $g_l \not\sim^\kappa g_l'$, or else $l\sigma_\kappa \sim^\kappa r\sigma_\kappa$ (since $l\sigma_\kappa \sim g_1$, $g_i \sim g_{i+1}$ and $g_n' \sim r\sigma_\kappa$ by KC). But then $c_l[u_l\sigma_l] \overset{*}{\underset{E}{\Leftrightarrow}} g_l \not\sim^\kappa g_l' \overset{*}{\underset{E}{\Leftrightarrow}} c_l[v_l\sigma_l]$, so $u_l\sigma_l \not\sim^\zeta v_l\sigma_l$, hence $G_\Sigma/{\sim^\zeta} \not\models H$. □

According to theorem 16, equational hypotheses H_0 may be resolved according to a relative hierarchical data-type $G_\Sigma/{\sim^r}$ composed of final modules, by running completion w.r.t. the top-level (n'th-level) equations E_n and extracting a set of new hypotheses H_1 in kernel terms. These new hypotheses are in turn given to completion together with the level $n-1$ equations E_{n-1}, and so on, until some *atomic* module is encountered, which if standard initial can be dealt with by standard methods. Note how reasoning about $G_\Sigma/{\sim^r}$ is done modularly w.r.t. current top-level equations only. This lightens the burden of having to consider for completion a large monolithic set of equations for the whole hierarchical data-type.

What about initial modules? Assuming that all relative initial modules only have kernel-sorts, corollary 13 states that under kernel preservation, initial modules may be treated as final. If kernel preservation is not guaranteed, then by duality, initial kernel corruption will manifest itself in final kernel corruption. Viewing any kernel corruption as simply a sign of *wrongful* specification, initial and final specification may be viewed as equivalent in kernel sorts, and theorem 16 applies.

Theorem 16 says that a *kernel corrupting witness* will reveal itself if and only if the set of hypotheses are not valid in the data-type. Hence, *proof by kernel preservation* is a generalised *relative* version of proof by consistency. By the equivalence in kernel sorts, this is also a generalised relative version of proof by consistency methods designated for standard initial semantics.

Example 9. Consider the following specification:

> **spec** FlatTree **final**
> **relative to** $\mathsf{AltSumList}_{\mathsf{op}}$ **initial**
> **sorts** tree
> **constructors** ϵ : tree, leaf : int $\to$ tree, node : tree $\times$ tree $\to$ tree
> **defined oprs** flatten : tree $\to$ list
> **axioms** E_{FlatTree} : flatten(ϵ) = 0,
> flatten(leaf(x)) = x $\dashv\varepsilon$,
> flatten(node(l,r)) = flatten(l) $\mapsto$ flatten(r)

The observer flatten should give final-semantically that two G_{FlatTree_c}-terms are semantically equal if and only if they represent trees which, when flattened (infix) give semantically equal sequences according to $\mathsf{AltSumList}_{\mathsf{op}}$ **initial**. To check whether

$$h_0\text{: } \mathsf{node(node(leaf(x),leaf(y)),leaf(z))} = \mathsf{node(leaf(x),node(leaf(y),leaf(z)))}$$

is valid in the relative datatype $G_{\mathsf{FlatTree}}/\!\sim^{\zeta}$ specified by FlatTree **final**, the set $\{h_0\} \cup E_{\mathsf{FlatTree}}$ is subjected to completion with a term-ordering in which every term in $T_{\mathsf{FlatTree}}(X) \setminus T_{\mathsf{AltSumList}_{\mathsf{op}}}(X)$ is greater than all terms in $T_{\mathsf{AltSumList}_{\mathsf{op}}}(X)$. This yields a single equation

$$h_1\text{: } ((\mathsf{x}\dashv\varepsilon) \mapsto (\mathsf{y}\dashv\varepsilon)) \mapsto (\mathsf{z}\dashv\varepsilon) = (\mathsf{x}\dashv\varepsilon) \mapsto ((\mathsf{y}\dashv\varepsilon) \mapsto (\mathsf{z}\dashv\varepsilon))$$

over $T_{\mathsf{AltSumList}_{\mathsf{op}}}(X)$. Completion of $\{h_1\} \cup E_{\mathsf{AltSumList}_{\mathsf{op}}}$ does not yield any hypotheses over $T_{\mathsf{AltSumList}}(X)$, so according to theorem 16, h_0 is valid in $G_{\mathsf{FlatTree}}/\!\sim^{\zeta}$.

Completion of

$$h_0'\text{: } \mathsf{node(leaf(x),node(leaf(y),}\epsilon\mathsf{))} = \mathsf{node(}\epsilon,\epsilon\mathsf{)}$$

together with E_{FlatTree} yields the new hypothesis

$$h_1'\text{: } (\mathsf{x} \dashv \varepsilon) \mapsto ((\mathsf{y} \dashv \varepsilon) \mapsto \varepsilon) = \varepsilon \mapsto \varepsilon$$

which completed with $E_{\mathsf{AltSumList}_{\mathsf{op}}}$ yields

$$\mathsf{y} \dashv (\mathsf{x} \dashv (\mathsf{x} \dashv \mathsf{q})) = \mathsf{y} \dashv \mathsf{q}$$

which completed with $E_{\mathsf{AltSumList}}$ produces

$$0 = \mathsf{x}$$

which can be refuted as an inductive consequence of $E_{\mathsf{Int}_{\mathsf{op}}}$ by standard methods. Hence h_1' is not valid in $G_{\mathsf{FlatTree}}/\!\sim^{\zeta}$. ○

Theorem 16 also shows how one might discover kernel corruption. If the procedures refutes a hypothesis which is known to hold in the kernel, then the specification is not kernel preserving. Similar ideas occur in [Kir84].

7 Non-homogeneous hierarchies

In this section two sources of non-homogeneouity to a specification hierarchy are briefly sketched. The key point being that relative specification admits non-homogeneouity and supports different proof methods at each level.
Auxiliary functions may give unintended semantics:

$$
\begin{array}{l}
\textbf{spec}\ \mathsf{AltSumList'}\ \textbf{final} \\
\textbf{relative to}\ \mathsf{Int_{op}}\ \textbf{initial} \\
\quad \textbf{sorts}\ \mathsf{list} \\
\quad \textbf{constructors}\ \varepsilon : \mathsf{list},\ \dashv : \mathsf{int} \times \mathsf{list} \to \mathsf{list} \\
\quad \textbf{defined oprs}\ \mathsf{altsum}, \mathsf{alt1}, \mathsf{alt2} : \mathsf{list} \to \mathsf{int} \\
\quad \textbf{axioms}\ E_{\mathsf{AltSumList'}} : \begin{array}[t]{l}
\mathsf{altsum(q) = alt1(q)}, \\
\mathsf{alt1}(\varepsilon) = 0, \\
\mathsf{alt2}(\varepsilon) = 0, \\
\mathsf{alt1(x \dashv q) = x - alt2(q)}, \\
\mathsf{alt2(x \dashv q) = x + alt1(q)}
\end{array}
\end{array}
$$

The operators $\mathsf{alt1}$ (handling the subtraction part) and $\mathsf{alt2}$ (handling the addidition part) are auxiliary to altsum, and the intended final semantics is that of $\mathsf{AltSumList}$ **final** of example 3. However, this specification is *finally inconsistent* w.r.t. $\Sigma_o = \{\mathsf{altsum}\}$: Letting $g = \mathsf{succ(0)\dashv(succ(0)\dashv(pred(0)}\dashv\varepsilon))$ and $g' = \mathsf{pred(0)}\dashv\varepsilon$, we have

$$\mathsf{alt2}(g) \overset{*}{\underset{E_{\mathsf{AltSumList'}}}{\longleftrightarrow}} \mathsf{succ(succ(succ(0)))} \not\sim^{\kappa} \mathsf{pred(0)} \overset{*}{\underset{E_{\mathsf{AltSumList'}}}{\longleftrightarrow}} \mathsf{alt2}(g')$$

giving $g \not\sim^{\zeta} g'$ (while $\mathsf{altsum}(g) \overset{*}{\underset{E_{\mathsf{AltSumList'}}}{\longleftrightarrow}} \mathsf{pred(0)} \overset{*}{\underset{E_{\mathsf{AltSumList'}}}{\longleftrightarrow}} \mathsf{altsum}(g')$, so the only intended distinguisher altsum does *not* distinguish g and g'). This can be resolved by using structured specifications involving hiding, and the corresponding structured semantics are then consistent, see [Hen97] for an overview. However, when reasoning about such specifications, the hidden information may have to be used, and proofs must then be done according to the structure in the specification and semantics. In [Han97] such structure is introduced into equational logic by restricting the monotonicity axiom in equational logic to allow context application of non-hidden symbols only. (A similar restriction is also applied for proving refinement of non-deterministic data-types in [QG93].) This effectively *protects* auxiliary operators from unauthorised use in the logic as well, i.e. $\mathsf{alt2}$ can only be accessed by the main operator altsum. Using this calculus, syntactic initial and final term models can be generated according to which a modification of proof by consistency checks equations [Han97]. The formulation of these semantics and the modification of proof by consistency benefits greatly from insight gained on notions of consistency (kernel preservation) presented in the current note. Relative semantics cater firstly for the definition of term model semantics according to this modified logic, and secondly by supporting modified versions of proof by consistency at each level.

Specifications like AltSumList' above that are inconsistent in some sense if auxiliary operators are not hidden, occur naturally in refinement proofs, some nice examples are to be found in [ST88]. Relative semantics thus ultimately enables proof by consistency to be used to a greater extent in specification refinement proofs.

Another source of non-homogeneouity is the following: Due to unorientability, no complete pure term-rewrite system exists for the characteristic equations for sets (of natural numbers) $\{\mathsf{add(add(s,x),x)} = \mathsf{add(s,x)},\ \mathsf{add(add(s,x),y)} = \mathsf{add(add(s,y),x)}\}$. However, it is possible to write a convergent specification of a function computing canonical representatives for the classes specified by the characteristic axioms, e.g. in the form of sorted non-repeating lists. So if E_s specifies a canonical representatives function *crep*, this gives rise to *indirect specification* of the form $g \sim^s g' \;\Leftrightarrow\; s(g) \xleftrightarrow[E_s]{*} s(g')$, where s represents *crep*. The issue of convergence is crucial for rewrite related reasoning, and by specifying a *function* inducing a semantics, rather than the semantics itself directly, one may thus indirectly convergently specify semantics for which no direct convergent set of equations exist.

Now, s is not a symbol intended to represent a genuine function in a data-type. In the case of an indirectly specified module occurring in a larger context, one would not want to view the entire entity monolithically. Proof by consistency adapted to indirect semantics would have to be treated as a separate process, making monolithic completion unsuitable. Relative specification and semantics provide the formalism for incorporating this kind of incompatible specification as a module of a larger hierarchical specification.

In certain cases it is possible to reduce indirect semantics to final semantics, and also to initial semantics by introducing the equation $\mathsf{s(x)} = \mathsf{x}$. Proving this benefits greatly from formulation in the relative constructs.

8 Concluding Remarks

The notions of relative specification and semantics were presented in order to (1) make explicit the notion of a kernel in existing notions of initial and final semantics. This provides (2) a uniform view of initial and final semantics and dualised notions of consistency as kernel preservation, and also provides (3) a modular hierarchical generalisation of initial and final semantics admitting a corresponding modular hierarchical method of proof by consistency where at each level the kernel may be disregarded. This extends to the "meta-theory" also—results in this note are stated and proved disregarding details of the kernel. Making the kernel explicit in the formalisms also (4) permits non-homogeneous specification and semantics schemes, admitting diverse proof methods at each level of the hierarchy.

This work on extremal monomorphic semantics is a basic step: It lays a basis on which investigation into relative *loose* semantics may be done. Given a loose kernel semantics consisting of a class of congruences, there are several ways of defining a next relative layer. One option is to extend in a point-wise fashion and consider a *class* of monomorphic relative models as the next layer semantics. Each member of this class will be extremal in some class of their own so one might also consider the union of all these classes to be the relative loose semantics.

Much investigation has been invested into modularity and compositionality properties of rewrite systems [Ohl95]. The approach in this note tends to *de*composition in the sense that only modular parts of an entire specification should be treated at a time. In particular, an analysis of the effects of running completion on parts of an equation set compared to those of completing the entire set, would seem interesting.

Acknowledgements

Thanks to Don Sannella and Olav Lysne for valuable comments, and to the makers of ReDuX, Reinhard Bündgen and Jochen Walter. This research is sponsored by the Norwegian Research Council, grant no. 110904/41

References

[Bac87] L. Bachmair. *Proof methods for equational theories*. PhD thesis, University of Illinois, Urbana-Champaign, 1987.

[Bac88] L. Bachmair. Proof by consistency in equational theories. In *Proceedings 3rd IEEE Symposium on Logic in Computer Science, Edinburgh (UK)*, pages 228–233, 1988.

[Bac91] L. Bachmair. *Canonical equational proofs*. Computer Science Logic, Progress in Theoretical Computer Science. Birkhäuser Verlag AG, 1991.

[BHW94] M. Bidoit, R. Hennicker, and M. Wirsing. Behavioural and abstractor specifications. Res. Rep. 9414, Ludwig-Maximilians-Universität München, August 1994.

[Bur69] R. Burstall. Proving properties of programs by structural induction. *Computer Journal*, 12(1):41–48, 1969.

[DJ90] N. Dershowitz and J.-P. Jouannaud. Rewrite systems. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, chapter 6. Elsevier, 1990.

[DO91] O.-J. Dahl and O. Owe. Formal development with ABEL. Forskningsraport 552, Institutt for informatikk, Universitetet i Oslo, 1991.

[EM85] H. Ehrig and B. Mahr. *Fundamentals of Algebraic Specifications I, Equations and Initial Semantics*, volume 6 of *EATCS Monographs on Theoretical Computer Science*. Springer, 1985.

[Fri86] L. Fribourg. A strong restriction on the inductive completion procedure. In *Proceedings 13th International Colloquium on Automata, Languages and Programming*, volume 226 of *LNCS*, pages 105–115. Springer, 1986.

[GHJ85] J.V. Guttag, J.J. Horning, and J.M.Wing. Larch in five easy pieces. Technical report, Digital Systems Research Center, 1985.

[Gog80] J.A. Goguen. How to prove inductive hypotheses without induction. In *Proceedings 5th International Conference on Automated Deduction*, volume 87 of *LNCS*, pages 356–373. Springer, 1980.

[Gor94] A. Gordon. A tutorial on co-induction and functional programming. In *Proceedings 1994 Glasgow Workshop on Functional Programming*, pages 78–95. Springer, 1994.

[Gut75] J.V. Guttag. *The Specification and Application to Programming of Abstract Data Types*. PhD thesis, Computer Science Department, University of Toronto, 1975.

[Gut77] J.V. Guttag. Abstract datatypes and the development of data structures. *Communications of the ACM*, 20(6):396–404, 1977.

[Han95] J.E. Hannay. Generalised algebraic specification with an application to indirect semantics. *Cand. Scient. Thesis, Inst. for informatikk, Universitetet i Oslo*, May 1995.

[Han97] J.E. Hannay. Proof by consistency for data-types with hidden operators. Available by `http://www.dcs.ed.ac.uk/home/joh/LogicHiding.ps`, January 1997.

[Hen91] R. Hennicker. Context induction: A proof principle for behavioural abstractions and algebraic implementations. *Formal Aspects of Computing*, 3:326–345, 1991.

[Hen92] R. Hennicker. A semi-algorithm for algebraic implementation proofs. *Theoretical Computer Science*, 104:53–87, 1992.

[Hen97] R. Hennicker. Structured specifications with behavioural operators: Semantics, proof methods and applications. Habilitationsschrift, *Institut für Informatik, Ludwig-Maximilians-Universität, München*, 1997.

[Her92] M. Hermann. On the relation between primitive recursion, schematization, and divergence. In *Proceedings of the 3rd International Conference on Algebraic and Logic Programming*, volume 632 of *LNCS*, pages 115–127. Springer, Sept. 1992.

[HH82] G. Huet and J.-M. Hullot. Proofs by induction in equational theories with constructors. *Journal of Computer and System Sciences*, pages 239–266, 1982.

[JK89] J.-P. Jouannaud and E. Kounalis. Automatic proofs by induction in theories without constructors. *Information and Computation*, 82(1):1–33, July 1989.

[Kam83] S. Kamin. Final data types and their specification. *ACM Transactions on Programming Languages and Systems*, 5(1):97–123, January 1983.

[KB70] D.E. Knuth and P.B. Bendix. Simple word problems in universal algebras. In J. Leech, editor, *Computational Problems in Abstract Algebra*, pages 263–297. Pergamon Press, Oxford, 1970.

[Kir84] H. Kirchner. A general inductive completion algorithm and application to abstract data types. In *cade7*, volume 170 of *LNCS*, pages 282–302. Springer, 1984.

[Klo92] J.W. Klop. Term rewriting systems. In S. Abramsky, D.M. Gabbai, and T.S.E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 2, pages 1–116. Oxford University Press, 1992.

[Küc89] W. Küchlin. Inductive completion by ground proof transformation. In H. Aït-Kaci and M. Nivat, editors, *Resolution of Equations in Algebraic Structures*, volume 2 of *Rewriting Techniques*, chapter 7. Academic Press, 1989.

[Les83] P. Lescanne. Behavioural categoricity of abstract data type specifications. *The Computer Journal*, 26(4):289–292, 1983.

[LPL96] H. Linnestad, C. Prehofer, and O. Lysne. Higher-order proof by consistency. In *Proceedings 16th International Conference FSTTCS, India*, volume 1180 of *LNCS*, pages 274–285, 1996.

[Lys92] O. Lysne. Proof by consistency in constructive systems with final algebra semantics. In *Proceedings 3rd International Conference on Algebraic and Logic Programming, Pisa*, volume 632 of *LNCS*, pages 276–290. Springer, 1992.

[Lys94] O. Lysne. Extending Bachmair's method for proof by consistency to the final algebra. *Information Processing Letters*, 51:303–310, 1994.

[Mor94] C. Morgan. *Programming from Specifications, 2nd ed.* Prentice Hall International Series in Computer Science; C.A.R. Hoare, Series Editor. Prentice-Hall, UK, 1994.

[Mus80] D.L. Musser. On proving inductive properties in abstract data types. In *Proceedings 7th Annual ACM Symposium on Principles of Programming Languages*, pages 154–162, January 1980.

[Ohl95] E. Ohlebusch. Modular properties of composable term rewriting systems. *Journal of Symbolic Computation*, 20(1):1–41, 1995.

[Pad95] P. Padawitz. Swinging datatypes. In *Recent Trends in Data Type Specification, Proceedings 11th workshop on Specification of Abstract Data types*, volume 1130 of *LNCS*, pages 409–435, Oslo, September 1995. Springer.

[Pue84] L. Puel. Proof in the final algebra. In B. Courcelle, editor, *Proceedings 9th Colloquium on Trees in Algebra and Programming*, pages 227–242. Cambridge University Press, 1984.

[QG93] X. Qian and A. Goldberg. Referential opacity in nondeterministic data refinement. *LoPLaS*, 2, 1993.

[ST87] D. Sannella and A. Tarlecki. On observational equivalence and algebraic specification. *Journal of Computer and System Sciences*, 34:150–178, 1987.

[ST88] D. Sannella and A. Tarlecki. Toward formal development of programs from algebraic specifications: implementations revisited. *Acta Informatica*, 25:233–281, 1988.

[ST97] D. Sannella and A. Tarlecki. Essential concepts of algebraic specification and program development. *Formal Aspects of Computing*, 1997. To appear. Available by `http://www.dcs.ed.ac.uk/home/dts/alf/concepts.ps`.

[Wir90] M. Wirsing. Algebraic specification. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, chapter 13, pages 675–788. Elsevier, 1990.

[WPP+83] M. Wirsing, P. Pepper, H. Partsch, W. Dosch, and M. Broy. On hierarchies of abstract data types. *Acta Informatica*, 20:1–33, 1983.