2 Algebraic Preliminaries

Donald Sannella[1] and Andrzej Tarlecki[2]

[1] Laboratory for Foundations of Computer Science, University of Edinburgh, Edinburgh, Scotland
dts@dcs.ed.ac.uk http://www.dcs.ed.ac.uk/~dts/
[2] Institute of Informatics, Warsaw University and Institute of Computer Science, Polish Academy of Sciences, Warsaw, Poland
tarlecki@mimuw.edu.pl http://wwwat.mimuw.edu.pl/~tarlecki/

The purpose of this chapter is to present the basic definitions and results on which the following chapters rely. Most of this material is quite standard and for that reason the presentation will be concise. More detailed presentations with greater emphasis on motivation, exercises, and examples may be found in [EM85, Wir90, LEW96, ST].

The most basic assumption of work on algebraic specification is that a program is modeled as an *algebra*, that is, a set of data together with a number of functions over this set. The branch of mathematics which deals with algebras in a general sense (as opposed to the study of specific classes of algebras, such as groups and rings) is called *universal algebra* or sometimes *general algebra*. This chapter presents the basics of universal algebra, generalized to the *many-sorted* case as required to model programs which manipulate several kinds or *sorts* of data. Some extensions useful for modeling more complex programs are sketched at the end of the chapter.

2.1 Many-sorted sets

When using an algebra to model a program which manipulates several sorts of data, it is natural to partition the underlying set of values in the algebra so that there is one set of values for each sort of data. It is often convenient to manipulate such a family of sets as a unit in such a way that operations on this unit respect the "typing" of data values.

Let S be a set (of sorts). An *S-sorted set* is an S-indexed family of sets $X = \langle X_s \rangle_{s \in S}$, which is *empty* if X_s is empty for all $s \in S$. The empty S-sorted set is written $\emptyset$.

Let $X = \langle X_s \rangle_{s \in S}$ and $Y = \langle Y_s \rangle_{s \in S}$ be S-sorted sets. *Union*, *intersection*, *Cartesian product*, *disjoint union*, *inclusion* (*subset*), and *equality* of X and Y are defined as follows:

$$X \cup Y = \langle X_s \cup Y_s \rangle_{s \in S}$$
$$X \cap Y = \langle X_s \cap Y_s \rangle_{s \in S}$$
$$X \times Y = \langle X_s \times Y_s \rangle_{s \in S}$$

$X \uplus Y = \langle X_s \uplus Y_s \rangle_{s \in S}$ (where $X_s \uplus Y_s$ is the disjoint union of X_s and Y_s)
$X \subseteq Y$ iff $X_s \subseteq Y_s$ for all $s \in S$
$X = Y$ iff $X \subseteq Y$ and $Y \subseteq X$.

An *S-sorted function* $f\colon X \to Y$ is an S-indexed family of functions $f = \langle f_s\colon X_s \to Y_s \rangle_{s \in S}$; X is called the *domain* of f, and Y is called its *codomain*. An S-sorted function $f\colon X \to Y$ is an *identity* (an *inclusion*, *surjective*, *injective*, *bijective*, ...) if for every $s \in S$, the function $f_s\colon X_s \to Y_s$ is an identity (an inclusion, surjective, injective, bijective, ...). The identity S-sorted function on X will be written as $id_X\colon X \to X$.

If $f\colon X \to Y$ and $g\colon Y \to Z$ are S-sorted functions, then their *composition* $f;g : X \to Z$ is the S-sorted function defined by $(f;g)_s(x) = g_s(f_s(x))$ for $s \in S$ and $x \in X_s$.

Let $f\colon X \to Y$ be an S-sorted function and $X' \subseteq X$, $Y' \subseteq Y$ be S-sorted sets. The *image of X' under f* is the S-sorted set $f(X') = \langle \{f_s(x) \mid x \in X'_s\} \rangle_{s \in S} \subseteq Y$. The *coimage of Y' under f* is the S-sorted set $f^{-1}(Y') = \langle \{x \in X_s \mid f_s(x) \in Y'_s\} \rangle_{s \in S} \subseteq X$.

An *S-sorted binary relation on X*, written $R \subseteq X \times X$, is an S-indexed family of binary relations $R = \langle R_s \subseteq X_s \times X_s \rangle_{s \in S}$. For $s \in S$ and $x, y \in X_s$, xR_sy (sometimes written xRy) means $\langle x, y \rangle \in R_s$. R is an *S-sorted equivalence (relation) on X* if it is reflexive (xR_sx), symmetric (xR_sy implies yR_sx), and transitive (xR_sy and yR_sz implies xR_sz). The symbol $\equiv$ is often used for (S-sorted) equivalence relations.

Let $\equiv$ be an S-sorted equivalence on X. If $s \in S$ and $x \in X_s$, then the *equivalence class of x modulo $\equiv$* is the set $[x]_{\equiv_s} = \{y \in X_s \mid x \equiv_s y\}$. The *quotient of X modulo $\equiv$* is the S-sorted set $X/{\equiv} = \langle \{[x]_{\equiv_s} \mid x \in X_s\} \rangle_{s \in S}$.

Let $f\colon X \to Y$ be an S-sorted function. The *kernel of f* is the S-sorted equivalence relation $K(f) = \langle \{\langle x, y \rangle \in X_s \times X_s \mid f_s(x) = f_s(y)\} \rangle_{s \in S} \subseteq X \times X$.

Subscripts selecting components of S-sorted sets (functions, relations, ...) are often omitted where there is no danger of confusion.

2.2 Signatures and algebras

An algebra consists of named sets and named functions on these sets. The set of names associated with an algebra is called its signature. The signature of an algebra defines the *syntax* of the algebra; the algebra itself supplies the *semantics* by assigning interpretations to the names.

A *(many-sorted) signature* is a pair $\Sigma = \langle S, \Omega \rangle$, where S is a set (of sort names) and Ω is an $S^* \times S$-sorted set (of operation names). Here, S^* is the set of finite (including empty) sequences of elements of S. We will sometimes write $sorts(\Sigma)$ for S and $opns(\Sigma)$ for Ω. Σ is a *subsignature* of $\Sigma' = \langle S', \Omega' \rangle$ if $S \subseteq S'$ and $\Omega \subseteq \Omega'$.

Saying that $f\colon s_1 \times \cdots \times s_n \to s$ is in $\Sigma = \langle S, \Omega \rangle$ means that $s_1 \ldots s_n \in S^*$, $s \in S$, and $f \in \Omega_{s_1 \ldots s_n, s}$. Then f is said to have *arity* $s_1 \ldots s_n$ and *result sort* s. If $n = 0$, that is $f\colon \to s$, we use the abbreviation $f\colon s$.

Many-sorted signatures will be referred to as *algebraic* signatures when it is necessary to distinguish them from other kinds of signatures to be introduced later. The above definition of signature permits overloading, since it is possible to have different arities and result sorts for a single operation name.

In the rest of this section, let $\Sigma = \langle S, \Omega \rangle$ be a signature.

A *Σ-algebra* A consists of an S-sorted set $|A|$ of *carrier sets* (or *carriers*); and, for each $f\colon s_1 \times \cdots \times s_n \to s$ in Σ, a function (or *operation*) $(f\colon s_1 \times \cdots \times s_n \to s)_A\colon |A|_{s_1} \times \cdots \times |A|_{s_n} \to |A|_s$. The class of all Σ-algebras will be denoted by $Alg(\Sigma)$.

If $f\colon s_1 \times \cdots \times s_n \to s$ is in Σ for $n = 0$ (i.e., $f\colon s$), then $|A|_{s_1} \times \cdots \times |A|_{s_n}$ is a singleton set containing the empty tuple $\langle\rangle$, and so $(f\colon s)_A$ may be viewed as a constant denoting the value $(f\colon s)_A(\langle\rangle) \in |A|_s$. Notice that $(f\colon s_1 \times \cdots \times s_n \to s)_A$ is a *total* function; see Section 2.10 for several ways of extending the definitions to cope with partial functions. Note also that there is no restriction on the cardinality of $|A|_s$; in particular, $|A|_s$ may be empty (but not if, e.g., $\Omega_{\varepsilon,s} \neq \emptyset$).

We always write f_A in place of $(f\colon s_1 \times \cdots \times s_n \to s)_A$ where there is no possibility of confusion.

Example 2.1. Let $S1 = \{food, car\}$ and let $\Omega 1_{\varepsilon,food} = \{soup\}$, $\Omega 1_{\varepsilon,car} = \{vw\}$, $\Omega 1_{food,food} = \{boil\}$, $\Omega 1_{food\ car,car} = \{f\}$, and $\Omega 1_{w,s} = \emptyset$ for all other $w \in S1^*, s \in S1$. Then $\Sigma 1 = \langle S1, \Omega 1 \rangle$ is a signature which can be presented in tabular form as follows:

$$\begin{aligned} \Sigma 1 = \ & \textbf{sorts}\ food, car \\ & \textbf{opns}\ soup\colon food \\ & \qquad vw\colon car \\ & \qquad boil\colon food \to food \\ & \qquad f\colon food \times car \to car \end{aligned}$$

Let $|A1|_{food} = \{\oplus, \otimes\}$, $|A1|_{car} = \{a, b, c\}$, $soup_{A1} = \oplus \in |A1|_{food}$, $vw_{A1} = b \in |A1|_{car}$ and $boil_{A1}\colon |A1|_{food} \to |A1|_{food} = \{\oplus \mapsto \oplus, \otimes \mapsto \oplus\}$, and let $f_{A1}\colon |A1|_{food} \times |A1|_{car} \to |A1|_{car}$ be defined by the following table:

f_{A1}	a	b	c
$\oplus$	a	c	b
$\otimes$	b	c	c

This defines a $\Sigma 1$-algebra $A1$. Reference will be made to $\Sigma 1$ and $A1$ in examples throughout the rest of this chapter. ∎

Let A and B be Σ-algebras. B is a *subalgebra* of A if $|B| \subseteq |A|$, and if $f_B(b_1, \ldots, b_n) = f_A(b_1, \ldots, b_n)$ for any $f\colon s_1 \times \cdots \times s_n \to s$ in Σ and $b_1 \in |B|_{s_1}, \ldots, b_n \in |B|_{s_n}$. B is a *proper* subalgebra of A if it is a subalgebra of

A and $|B| \neq |A|$. A subalgebra of A is determined by an S-sorted subset $|B|$ of $|A|$ such that for each $f\colon s_1 \times \cdots \times s_n \to s$ in Σ and $b_1 \in |B|_{s_1}, \ldots, b_n \in |B|_{s_n}$, $f_A(b_1, \ldots, b_n) \in |B|_s$.

The intersection of any family of (carriers of) subalgebras of A is a (carrier of a) subalgebra of A. This implies that for any $X \subseteq |A|$, there is a least subalgebra of A that contains X, called the *subalgebra of A generated by X*. A is *reachable* if it has no proper subalgebra (equivalently, if A is generated by $\emptyset$). It follows that every algebra has a unique reachable subalgebra.

Example 2.2. Let $\Sigma 1 = \langle S1, \Omega 1\rangle$ and $A1$ be as in Example 2.1. Define a $\Sigma 1$-algebra $B1$ by $|B1|_{food} = \{\oplus\}$, $|B1|_{car} = \{b, c\}$, $soup_{B1} = \oplus \in |B1|_{food}$, $vw_{B1} = b \in |B1|_{car}$, $boil_{B1}\colon |B1|_{food} \to |B1|_{food} = \{\oplus \mapsto \oplus\}$, and $f_{B1}\colon |B1|_{food} \times |B1|_{car} \to |B1|_{car} = \{\langle\oplus, b\rangle \mapsto c, \langle\oplus, c\rangle \mapsto b\}$. $B1$ is the subalgebra of $A1$ generated by $\emptyset$. That is, $B1$ is the reachable subalgebra of $A1$. ∎

2.3 Homomorphisms and congruences

A homomorphism between algebras is a function between the carrier sets which preserves the operations. Similarly, a congruence relation on an algebra is an equivalence which is preserved by the operations.

Throughout this section, let $\Sigma = \langle S, \Omega\rangle$ be a signature and let A and B be Σ-algebras.

A *Σ-homomorphism* $h\colon A \to B$ is an S-sorted function $h\colon |A| \to |B|$ such that for all $f\colon s_1 \times \cdots \times s_n \to s$ in Σ and $a_1 \in |A|_{s_1}, \ldots, a_n \in |A|_{s_n}$, $h_s(f_A(a_1, \ldots, a_n)) = f_B(h_{s_1}(a_1), \ldots, h_{s_n}(a_n))$.

Example 2.3. Let $\Sigma 1 = \langle S1, \Omega 1\rangle$ and $A1$ be as in Example 2.1. Define a $\Sigma 1$-algebra $B1$ by $|B1|_{food} = |B1|_{car} = \{1, 2, 3\}$, $soup_{B1} = 1 \in |B1|_{food}$, $vw_{B1} = 2 \in |B1|_{car}$ and $boil_{B1}\colon |B1|_{food} \to |B1|_{food} = \{1 \mapsto 1, 2 \mapsto 3, 3 \mapsto 1\}$, where $f_{B1}\colon |B1|_{food} \times |B1|_{car} \to |B1|_{car}$ is defined by the following table:

f_{B1}	1	2	3
1	1	2	3
2	2	1	2
3	2	2	1

Let $h1\colon |A1| \to |B1|$ be the $S1$-sorted function such that $h1_{food} = \{\oplus \mapsto 1, \otimes \mapsto 3\}$ and $h1_{car} = \{a \mapsto 1, b \mapsto 2, c \mapsto 2\}$. It is easy to verify that $h1\colon A1 \to B1$ is a $\Sigma 1$-homomorphism by checking the following:

$$\begin{aligned}
h1_{food}(soup_{A1}) &= soup_{B1}\\
h1_{car}(vw_{A1}) &= vw_{B1}\\
h1_{food}(boil_{A1}(\oplus)) &= boil_{B1}(h1_{food}(\oplus))\\
h1_{food}(boil_{A1}(\otimes)) &= boil_{B1}(h1_{food}(\otimes))\\
h1_{car}(f_{A1}(\oplus, a)) &= f_{B1}(h1_{food}(\oplus), h1_{car}(a))\\
h1_{car}(f_{A1}(\oplus, b)) &= f_{B1}(h1_{food}(\oplus), h1_{car}(b))\\
h1_{car}(f_{A1}(\oplus, c)) &= f_{B1}(h1_{food}(\oplus), h1_{car}(c))\\
h1_{car}(f_{A1}(\otimes, a)) &= f_{B1}(h1_{food}(\otimes), h1_{car}(a))\\
h1_{car}(f_{A1}(\otimes, b)) &= f_{B1}(h1_{food}(\otimes), h1_{car}(b))\\
h1_{car}(f_{A1}(\otimes, c)) &= f_{B1}(h1_{food}(\otimes), h1_{car}(c)).
\end{aligned}$$

■

The identity function on the carrier of a Σ-algebra is a Σ-homomorphism, and composition of Σ-homomorphisms yields another Σ-homomorphism.

Let $h\colon A \to B$ be a Σ-homomorphism, and let A' be a subalgebra of A. Let the *image of A' under h* be the Σ-subalgebra $h(A')$ of B such that $|h(A')| = h(|A'|)$, and $f_{h(A')}(h_{s_1}(a_1), \ldots, h_{s_n}(a_n)) = h_s(f_{A'}(a_1, \ldots, a_n))$ for each $f\colon s_1 \times \cdots \times s_n \to s$ in Σ and $a_1 \in |A'|_{s_1}, \ldots, a_n \in |A'|_{s_n}$. The *coimage* of a subalgebra B' of B under h is a subalgebra $h^{-1}(B')$ of A, defined analogously.

A Σ-homomorphism $h\colon A \to B$ is a *Σ-isomorphism* if it has an inverse, i.e., there is a Σ-homomorphism $h^{-1}\colon B \to A$ such that $h;h^{-1} = id_{|A|}$ and $h^{-1};h = id_{|B|}$. A homomorphism is an isomorphism iff it is bijective.

If there is an isomorphism from A to B, they are called *isomorphic* and we write $h\colon A \cong B$ or just $A \cong B$. Then $\cong$ (as a binary relation on Σ-algebras) is reflexive, symmetric, and transitive, and is therefore an equivalence relation.

Two isomorphic algebras are typically regarded as indistinguishable for all practical purposes: the only way in which they can differ is in the particular choice of data values in the carriers.

Example 2.4. Let $\Sigma 1 = \langle S1, \Omega 1\rangle$ and $A1$ be as in Example 2.1. Define a $\Sigma 1$-algebra $B1$ by $|B1|_{food} = \{\oplus, \otimes\}$, $|B1|_{car} = \{1, 2, 3\}$, $soup_{B1} = \otimes \in |B1|_{food}$, $vw_{B1} = 2 \in |B1|_{car}$, and $boil_{B1}\colon |B1|_{food} \to |B1|_{food} = \{\oplus \mapsto \otimes, \otimes \mapsto \otimes\}$, where $f_{B1}\colon |B1|_{food} \times |B1|_{car} \to |B1|_{car}$ is defined by the following table:

f_{B1}	1	2	3
$\oplus$	2	3	3
$\otimes$	1	3	2

Let $i1\colon |A1| \to |B1|$ be the $S1$-sorted function such that $i1_{food} = \{\oplus \mapsto \otimes, \otimes \mapsto \oplus\}$ and $i1_{car} = \{a \mapsto 1, b \mapsto 2, c \mapsto 3\}$. This defines a $\Sigma 1$-homomorphism $i1\colon A1 \to B1$ which is a $\Sigma 1$-isomorphism, so $A1 \cong B1$. ■

A *Σ-congruence on A* is an (S-sorted) equivalence $\equiv$ on $|A|$ which respects the operations of Σ: for all operations $f\colon s_1 \times \cdots \times s_n \to s$ in Σ and values $a_1, a_1' \in |A|_{s_1}, \ldots, a_n, a_n' \in |A|_{s_n}$, if $a_1 \equiv_{s_1} a_1'$ and $\ldots$ and $a_n \equiv_{s_n} a_n'$ then $f_A(a_1, \ldots, a_n) \equiv_s f_A(a_1', \ldots, a_n')$.

The intersection of any family of Σ-congruences on A is a Σ-congruence on A, which implies that for any S-sorted binary relation R on $|A|$ there exists a least (with respect to $\subseteq$) Σ-congruence on A which includes R.

Let $\equiv$ be a Σ-congruence on A. The *quotient of A modulo $\equiv$* is the Σ-algebra $A/{\equiv}$ such that $|A/{\equiv}| = |A|/{\equiv}$, and $f_{A/\equiv}([a_1]_{\equiv_{s_1}}, \ldots, [a_n]_{\equiv_{s_n}}) = [f_A(a_1, \ldots, a_n)]_{\equiv_s}$ for each $f\colon s_1 \times \cdots \times s_n \to s$ and $a_1 \in |A|_{s_1}, \ldots, a_n \in |A|_{s_n}$.

Example 2.5. Let $\Sigma 1 = \langle S1, \Omega 1\rangle$ and $A1$ be as in Example 2.1, and let $\equiv = \langle \equiv_s \rangle_{s \in S1}$ be the $S1$-sorted congruence on $|A1|$ defined by $\equiv_{food} = \{\langle \oplus, \oplus\rangle, \langle \otimes, \otimes\rangle\}$ and $\equiv_{car} = \{\langle a, a\rangle, \langle b, b\rangle, \langle b, c\rangle, \langle c, b\rangle, \langle c, c\rangle\}$. $A1/{\equiv}$ is the $\Sigma 1$-algebra defined by

$$\begin{array}{rl} |A1/{\equiv}|_{food} = \{\{\oplus\}, \{\otimes\}\}, & |A1/{\equiv}|_{car} = \{\{a\}, \{b, c\}\}, \\ soup_{A1/\equiv} = \{\oplus\} \in |A1/{\equiv}|_{food}, & vw_{A1/\equiv} = \{b, c\} \in |A1/{\equiv}|_{car}, \\ boil_{A1/\equiv}\colon |A1/{\equiv}|_{food} \to |A1/{\equiv}|_{food} = \{\{\oplus\} \mapsto \{\oplus\}, \{\otimes\} \mapsto \{\oplus\}\}, & \end{array}$$

where $f_{A1/\equiv}\colon |A1/{\equiv}|_{food} \times |A1/{\equiv}|_{car} \to |A1/{\equiv}|_{car}$ is defined by the following table:

$f_{A1/\equiv}$	$\{a\}$	$\{b, c\}$
$\{\oplus\}$	$\{a\}$	$\{b, c\}$
$\{\otimes\}$	$\{b, c\}$	$\{b, c\}$

■

The kernel of any Σ-homomorphism $h\colon A \to B$ is a Σ-congruence on A. Moreover, if $\equiv$ is a Σ-congruence on A, and $h_s(a) = [a]_{\equiv_s}$ for $s \in S$, $a \in |A|_s$, then $\langle h_s\colon |A|_s \to (|A|/{\equiv})_s\rangle_{s \in S}$ is a Σ-homomorphism $h\colon A \to A/{\equiv}$. Therefore, a binary relation on $|A|$ is a Σ-congruence on A iff it is the kernel of a Σ-homomorphism from A.

Proposition 2.6. *Let $\equiv$ be a Σ-congruence on A. If $h\colon A \to B$ is a Σ-homomorphism such that $\equiv \subseteq K(h)$, then there is a unique Σ-homomorphism $g\colon A/{\equiv} \to B$ such that $h_s(a) = g_s([a]_{\equiv_s})$ for all $s \in S$ and $a \in |A|_s$.*

The above property characterizes quotient algebras up to isomorphism. It follows in particular that for any Σ-homomorphism $h\colon A \to B$, $A/K(h)$ is isomorphic to $h(A)$.

2.4 Term algebras

Throughout this section, let $\Sigma = \langle S, \Omega\rangle$ be a signature and let X be an S-sorted set (of variables), where $x \in X_s$ for $s \in S$ means that the variable

x is of sort s (written $x\colon s$). Note that "overloading" of variable names is permitted here, since there is no requirement that X_s and $X_{s'}$ be disjoint for $s \neq s' \in S$.

The *Σ-algebra $T_\Sigma(X)$ of terms with variables X* is the Σ-algebra defined as follows:

- $|T_\Sigma(X)|$ is the least S-sorted set such that $x \in |T_\Sigma(X)|_s$ for all $s \in S$ and $x \in X_s$, and $f(t_1, \ldots, t_n) \in |T_\Sigma(X)|_s$ for all $f\colon s_1 \times \cdots \times s_n \to s$ in Σ and $t_1 \in |T_\Sigma(X)|_{s_1}, \ldots, t_n \in |T_\Sigma(X)|_{s_n}$,
- for all $f\colon s_1 \times \cdots \times s_n \to s$ in Σ and $t_1 \in |T_\Sigma(X)|_{s_1}, \ldots, t_n \in |T_\Sigma(X)|_{s_n}$, $f_{T_\Sigma(X)}(t_1, \ldots, t_n) = f(t_1, \ldots, t_n) \in |T_\Sigma(X)|_s$.

Note the distinction between syntactic term formation $f(t_1, \ldots, t_n)$ and the application of the operation named f. If $s \in S$ and $t \in |T_\Sigma(X)|_s$, then t is a *Σ-term of sort s with variables X*; the *free variables of t* is the set $FV(t) \subseteq X$ of variables that actually occur in t.

The *Σ-algebra of ground terms* is the Σ-algebra $T_\Sigma = T_\Sigma(\emptyset)$ of terms without variables. If $s \in S$ and $t \in |T_\Sigma|_s$, then t is a *ground Σ-term*.

Example 2.7. Let $\Sigma 1 = \langle S1, \Omega 1\rangle$ be as in Example 2.1. Then $T_{\Sigma 1}$ is the $\Sigma 1$-algebra defined by

$$|T_{\Sigma 1}|_{food} = \{soup(), boil(soup()), boil(boil(soup())), \ldots\},$$

$$|T_{\Sigma 1}|_{car} = \{vw(), f(soup(), vw()), f(boil(soup()), vw()), f(soup(), f(soup(), vw())), \ldots\}$$

where the operations of $T_{\Sigma 1}$ are the term-formation operations

$$soup_{T_{\Sigma 1}} = soup() \in |T_{\Sigma 1}|_{food}, \qquad vw_{T_{\Sigma 1}} = vw() \in |T_{\Sigma 1}|_{car},$$

$$boil_{T_{\Sigma 1}}\colon |T_{\Sigma 1}|_{food} \to |T_{\Sigma 1}|_{food} = \{soup() \mapsto boil(soup()), boil(soup()) \mapsto boil(boil(soup())), \ldots\},$$

and similarly for $f\colon food \times car \to car$. ∎

It is implicitly assumed above that the result sort of each term is determined unambiguously. If the signature Σ and the set of variables X do not ensure this property, then appropriate sort decorations must be added to terms. We will henceforth assume that variables and constants (0-ary operations) of the same sort are distinct, which allows us to drop the parentheses "()" in terms like $boil(soup())$ above. So, in the example we would have:

$$|T_{\Sigma 1}|_{food} = \{soup, boil(soup), \ldots\},$$

$$|T_{\Sigma 1}|_{car} = \{vw, f(soup, vw), f(boil(soup), vw), \ldots\}$$

In examples we will also use infix notation for binary operations when convenient.

Let A be a Σ-algebra, $v\colon X \to |A|$ be an S-sorted function, $s \in S$, and $t \in |T_\Sigma(X)|_s$ be a Σ-term of sort s. The *value of t in A under the valuation v* is $v^\#(t) \in |A|_s$, defined as follows:

- $v^\#(x) = v(x)$ for all $s \in S$ and $x \in X_s$; and
- $v^\#(f(t_1, \ldots, t_n)) = f_A(v^\#(t_1), \ldots, v^\#(t_n))$ for all $f\colon s_1 \times \cdots \times s_n \to s$ in Σ and $t_1 \in |T_\Sigma(X)|_{s_1}, \ldots, t_n \in |T_\Sigma(X)|_{s_n}$.

Proposition 2.8. *For any Σ-algebra A and S-sorted function $v\colon X \to |A|$, $v^\#\colon T_\Sigma(X) \to A$ is the unique Σ-homomorphism that extends v, i.e., such that $v^\#_s(x) = v_s(x)$ for all $s \in S$, $x \in X_s$.*

It is easy to see that the value of a term $t \in |T_\Sigma(X)|$ depends only on the valuation of variables in $FV(t)$. In particular, the value of a ground term $t \in |T_\Sigma|$ does not depend on the valuation. Hence we write $t_A = \emptyset^\#(t)$, where $\emptyset\colon \emptyset \to |A|$ is the empty function, for the *value of t in A*.

The Σ-algebra A is reachable iff every element in $|A|$ is the value of a ground Σ-term, or equivalently, iff it is isomorphic to a quotient of T_Σ. It follows that there is a one-to-one correspondence between isomorphism classes of reachable Σ-algebras and congruences on T_Σ.

When the algebra A above is a term algebra $T_\Sigma(Y)$ for some S-sorted set Y, valuations are called *substitutions* (of terms in $T_\Sigma(Y)$ for variables), and the value of a term t under a substitution $\theta\colon X \to |T_\Sigma(Y)|$, written $t[\theta]$, is just the result of substituting $\theta(x)$ for all x in t in the usual sense. We write $t[x \mapsto u]$ for the result of replacing x in t by u, regarding $x \mapsto u$ as a shorthand for the obvious substitution which is the identity on all variables in X other than x (here, Y is $X \cup FV(u)$).

2.5 Signature morphisms

A signature morphism defines a mapping from the sort and operation names in one signature to those in another signature, in such a way that the arity and result sort of operations are respected.

More formally, let $\Sigma = \langle S, \Omega\rangle$ and $\Sigma' = \langle S', \Omega'\rangle$ be signatures. A *signature morphism* $\sigma\colon \Sigma \to \Sigma'$ is a pair $\sigma = \langle \sigma_{sorts}, \sigma_{opns}\rangle$ where $\sigma_{sorts}\colon S \to S'$ and $\sigma_{opns} = \langle \sigma_{w,s}\colon \Omega_{w,s} \to \Omega'_{\sigma^*_{sorts}(w),\sigma_{sorts}(s)}\rangle_{w \in S^*, s \in S}$ (where for $w = s_1 \ldots s_n \in S^*$, $\sigma^*_{sorts}(w) = \sigma_{sorts}(s_1) \ldots \sigma_{sorts}(s_n)$). Both σ_{sorts} and σ_{opns} (and its components $\sigma_{w,s}$ for all $w \in S^*, s \in S$) will be denoted by σ.

Signature morphisms as defined above will be referred to as *algebraic* signature morphisms when it is necessary to distinguish them from other kinds of signature morphisms to be introduced later.

Example 2.9. Let $\Sigma = \langle S, \Omega\rangle$ be the signature

sorts *warm*, *cold*, *vehicle*
opns *borscht*: *cold*
heat: *cold* → *warm*
heat: *warm* → *warm*
h: *warm* × *vehicle* → *vehicle*

Let $\Sigma 1 = \langle S1, \Omega 1\rangle$ be the signature in Example 2.1. Define $\sigma_{sorts}\colon S \to S1$ and $\sigma_{opns} = \langle \sigma_{w,s}\colon \Omega_{w,s} \to \Omega 1_{\sigma^*_{sorts}(w),\sigma_{sorts}(s)}\rangle_{w\in S^*,s\in S}$ by $\sigma_{sorts} = \{warm \mapsto food, cold \mapsto food, vehicle \mapsto car\}$, $\sigma_{\varepsilon,cold} = \{borscht \mapsto soup\}$, $\sigma_{cold,warm} = \{heat \mapsto boil\}$, $\sigma_{warm,warm} = \{heat \mapsto boil\}$, $\sigma_{warm\,vehicle,vehicle} = \{h \mapsto f\}$, and $\sigma_{w,s} = \emptyset$ for all other $w \in S^*, s \in S$. Then $\sigma\colon \Sigma \to \Sigma 1$ is a signature morphism. ∎

In the rest of this section, let $\sigma\colon \Sigma \to \Sigma'$ be a signature morphism. This gives rise to a translation of Σ-terms to Σ'-terms, and of Σ'-algebras and homomorphisms to Σ-algebras and homomorphisms, as defined below. Note that the direction of translation of algebras and homomorphisms is "backwards" with respect to the direction of the signature morphism.

Let A' be a Σ'-algebra. The *σ-reduct of A'* is the Σ-algebra $A'|_\sigma$ such that $|A'|_\sigma|_s = |A'|_{\sigma(s)}$ for all $s \in S$, and $f_{A'|_\sigma} = \sigma(f)_{A'}$ for all $f\colon s_1 \times \cdots \times s_n \to s$ in Σ. Similarly, if $h'\colon A' \to B'$ is a Σ'-homomorphism, the *σ-reduct of h'* is the Σ-homomorphism $h'|_\sigma\colon |A'|_\sigma| \to |B'|_\sigma|$ such that $(h'|_\sigma)_s = h'_{\sigma(s)}$ for all $s \in S$.

If Σ is a subsignature of Σ', then we write $A'|_\Sigma$ for $A'|_\sigma$ where $\sigma\colon \Sigma \to \Sigma'$ is the obvious signature inclusion (and similarly for homomorphisms). Then $A'|_\Sigma$ is just A' with some carriers and/or operations removed.

Example 2.10. Let $\sigma\colon \Sigma \to \Sigma 1$ and $A1$ be as in Examples 2.9 and 2.1 respectively. Then $A1|_\sigma$ is the Σ-algebra such that $|A1|_\sigma|_{warm} = |A1|_\sigma|_{cold} = \{\oplus, \otimes\} = |A1|_{food}$, $|A1|_\sigma|_{vehicle} = \{a, b, c\} = |A1|_{car}$, $borscht_{A1|_\sigma} = \oplus = soup_{A1}$, $(heat\colon cold \to warm)_{A1|_\sigma} = \{\oplus \mapsto \oplus, \otimes \mapsto \oplus\} = boil_{A1}$, $(heat\colon warm \to warm)_{A1|_\sigma} = \{\oplus \mapsto \oplus, \otimes \mapsto \oplus\} = boil_{A1}$ and $h_{A1|_\sigma} = \{\langle\oplus, a\rangle \mapsto a, \langle\oplus, b\rangle \mapsto c, \ldots\} = f_{A1}$. ∎

Let X be an S-sorted set of variables such that X_s and $X_{s'}$ are disjoint for $s \neq s' \in S$. Define $X' = \langle\bigcup_{\sigma(s)=s'} X_s\rangle_{s'\in S'}$. The *translation of a Σ-term $t \in |T_\Sigma(X)|$ by σ* is the Σ'-term $\sigma(t) \in |T_{\Sigma'}(X')|$ obtained by replacing each operation name f in t by $\sigma(f)$. (The disjointness assumption on X is for notational convenience only. It may be avoided by taking the disjoint union in the definition of X'.)

Example 2.11. Let $\sigma\colon \Sigma \to \Sigma 1$ be the signature morphism in Example 2.9, where $\Sigma = \langle S, \Omega\rangle$ and $\Sigma 1 = \langle S1, \Omega 1\rangle$. Let X be the S-sorted set of variables $x\colon cold, x'\colon warm, y\colon warm, z\colon vehicle$. The $S1$-sorted set of variables X' is then $x\colon food, x'\colon food, y\colon food, z\colon car$, and

$$\sigma(h(heat(x), h(x', z))) = f(boil(x), f(x', z)),$$

$$\sigma(h(x', h(heat(heat(borscht)), z))) = f(x', f(boil(boil(soup)), z)),$$

and so on. ∎

The following result states that the value of a term is invariant under change of signature.

Proposition 2.12. *Let X be an S-sorted set of variables such that X_s and $X_{s'}$ are disjoint for $s \neq s' \in S$, and $X' = \langle \bigcup_{\sigma(s)=s'} X_s \rangle_{s' \in S'}$. Let A' be a Σ'-algebra and $v' \colon X' \to |A'|$ be a valuation. Define $v \colon X \to |A'|_\sigma|$ by $v_s(x) = v'_{\sigma(s)}(x)$ for $s \in S$ and $x \in X_s$. Then for any Σ-term $t \in |T_\Sigma(X)|$, $v^\#(t) = (v')^\#(\sigma(t))$. In particular, if t is a ground term, then $t_{A'|_\sigma} = \sigma(t)_{A'}$.*

2.6 Equations

In the simple algebraic specifications considered in this chapter, equations are used as axioms to constrain the permitted behaviour of operations.

Throughout this section, let $\Sigma = \langle S, \Omega \rangle$ be a signature.

A *Σ-equation* $\forall X.\ t = t'$ consists of an S-sorted set X (of variables) such that X_s and $X_{s'}$ are disjoint for $s \neq s' \in S$, and two Σ-terms $t, t' \in |T_\Sigma(X)|_s$ for some sort $s \in S$. A Σ-equation $\forall \emptyset.\ t = t'$, sometimes abbreviated $t = t'$, is called a *ground (Σ-) equation*.

A Σ-algebra A *satisfies* (or, *is a model of*) a Σ-equation $\forall X.\ t = t'$, written $A \models_\Sigma \forall X.\ t = t'$, if for every ($S$-sorted) function $v \colon X \to |A|$, $v^\#(t) = v^\#(t')$.

A satisfies (or, is a model of) a set Φ of Σ-equations, written $A \models_\Sigma \Phi$, if $A \models_\Sigma \varphi$ for every equation $\varphi \in \Phi$. A class $\mathcal{A}$ of Σ-algebras satisfies a Σ-equation φ, written $\mathcal{A} \models_\Sigma \varphi$, if $A \models_\Sigma \varphi$ for every $A \in \mathcal{A}$. Finally, a class $\mathcal{A}$ of Σ-algebras satisfies a set Φ of Σ-equations, written $\mathcal{A} \models_\Sigma \Phi$, if $A \models_\Sigma \Phi$ for every $A \in \mathcal{A}$. We sometimes write $\models$ in place of $\models_\Sigma$ where Σ is obvious.

The explicit quantification over X in a Σ-equation $\forall X.\ t = t'$ is essential. For example, if $|A|_s = \emptyset$ but $X_s \neq \emptyset$ for some s in S, then A trivially satisfies any equation $\forall X.\ t = t'$. Thus variables in X may influence satisfaction even if they do not actually occur in t or t'.

Satisfaction of Σ-algebras is preserved under subalgebras and homomorphic images: if $A \models \varphi$ then φ is satisfied by any subalgebra of A and by any homomorphic image of A (and thus by any algebra isomorphic to A).

Let $\sigma \colon \Sigma \to \Sigma'$ be a signature morphism. The translation of Σ-terms to Σ'-terms defined above extends in the obvious way to a translation of Σ-equations to Σ'-equations. We will write $\sigma(\forall X.\ t = t')$ for $\forall X'.\ \sigma(t) = \sigma(t')$, where $X'_{s'} = \bigcup_{\sigma(s)=s'} X_s$ for each $s' \in S'$ as above.

An important result that brings together some of the main definitions above is as follows:

Lemma 2.13 (Satisfaction Lemma [BG80]). *If $\sigma \colon \Sigma \to \Sigma'$ is a signature morphism, φ is a Σ-equation, and A' is a Σ'-algebra, then $A' \models_{\Sigma'} \sigma(\varphi)$ iff $A'|_\sigma \models_\Sigma \varphi$.*

This states that the translations of syntax (terms, equations) and semantics (algebras) induced by signature morphisms are coherent with the definition of satisfaction. The proof follows from Proposition 2.12.

2.7 Presentations and theories

A signature, together with a set of equations over that signature, constitutes a simple form of specification. We refer to these as *flat* (meaning *unstructured*) specifications in order to distinguish them from the *structured* specifications to be introduced in later chapters.

Throughout this section, let Σ be a signature.

A *presentation* (also known as a *flat specification*) is a pair $\langle \Sigma, \Phi \rangle$ where Φ is a set of Σ-equations (called the *axioms* of $\langle \Sigma, \Phi \rangle$). A presentation $\langle \Sigma, \Phi \rangle$ is sometimes referred to as a *Σ-presentation*.

A *model* of a presentation $\langle \Sigma, \Phi \rangle$ is a Σ-algebra A such that $A \models_\Sigma \Phi$. $Mod_\Sigma(\Phi)$ is the class of all models of $\langle \Sigma, \Phi \rangle$. Taking $\langle \Sigma, \Phi \rangle$ to denote the semantic object $Mod_\Sigma(\Phi)$ is sometimes called taking its *loose semantics*.

Example 2.14. Let $Bool = \langle \Sigma Bool, \Phi Bool \rangle$ be the following presentation.

$$
\begin{array}{lll}
Bool = \textbf{sorts} & bool & \\
\textbf{opns} & true : bool & \\
 & false : bool & \\
 & \neg : bool \to bool & \\
 & \wedge : bool \times bool \to bool & \\
\textbf{axioms} & \neg true = false & \\
 & \neg false = true & \\
 & \forall p{:}bool.\ p \wedge true = p & \\
 & \forall p{:}bool.\ p \wedge false = false & \\
 & \forall p{:}bool.\ p \wedge \neg p = false &
\end{array}
$$

Define $\Sigma Bool$-algebras $A1$, $A2$, and $A3$ as follows:

$$
\begin{array}{lll}
|A1|_{bool} = \{\star\} & |A2|_{bool} = \{a, b, c\} & |A3|_{bool} = \{1, 0\} \\
true_{A1} = \star & true_{A2} = a & true_{A3} = 1 \\
false_{A1} = \star & false_{A2} = b & false_{A3} = 0 \\
\neg_{A1} = \{\star \mapsto \star\} & \neg_{A2} = \{a \mapsto b, & \neg_{A3} = \{1 \mapsto 0, \\
 & \quad b \mapsto a, & \quad 0 \mapsto 1\} \\
 & \quad c \mapsto c\} &
\end{array}
$$

$\wedge_{A1}$	$\star$
$\star$	$\star$

$\wedge_{A2}$	a	b	c
a	a	b	b
b	b	b	b
c	c	b	b

$\wedge_{A3}$	1	0
1	1	0
0	0	0

Each of these algebras is a model of *Bool*. (Reference will be made to *Bool* and to $A1$, $A2$, and $A3$ in later sections of this chapter.) ∎

For any class $\mathcal{A}$ of Σ-algebras, $Th_\Sigma(\mathcal{A})$ (the *theory* of $\mathcal{A}$) denotes the set of all Σ-equations satisfied by each Σ-algebra in $\mathcal{A}$:

$$Th_\Sigma(\mathcal{A}) = \{\varphi \mid \varphi \text{ is a } \Sigma\text{-equation and } \mathcal{A} \models_\Sigma \varphi\}.$$

The *closure* of a set Φ of Σ-equations is the set $Cl_\Sigma(\Phi) = Th_\Sigma(Mod_\Sigma(\Phi))$; Φ is *closed* if $\Phi = Cl_\Sigma(\Phi)$.

Proposition 2.15. *For any sets Φ and Ψ of Σ-equations and classes $\mathcal{A}, \mathcal{B}$ of Σ-algebras:*

1a. If $\Phi \subseteq \Psi$ then $Mod_\Sigma(\Psi) \subseteq Mod_\Sigma(\Phi)$.
1b. If $\mathcal{A} \subseteq \mathcal{B}$ then $Th_\Sigma(\mathcal{B}) \subseteq Th_\Sigma(\mathcal{A})$.
2a. $\Phi \subseteq Th_\Sigma(Mod_\Sigma(\Phi))$.
2b. $\mathcal{A} \subseteq Mod_\Sigma(Th_\Sigma(\mathcal{A}))$.
3a. $Mod_\Sigma(\Phi) = Mod_\Sigma(Th_\Sigma(Mod_\Sigma(\Phi)))$.
3b. $Th_\Sigma(\mathcal{A}) = Th_\Sigma(Mod_\Sigma(Th_\Sigma(\mathcal{A})))$.

A Σ-equation φ is a *semantic* (or *model-theoretic*) *consequence* of a set Φ of Σ-equations, written $\Phi \models_\Sigma \varphi$, if $\varphi \in Cl_\Sigma(\Phi)$ (equivalently, if $Mod_\Sigma(\Phi) \models_\Sigma \varphi$). We will write $\Phi \models \varphi$ instead of $\Phi \models_\Sigma \varphi$ where the signature Σ is obvious.

Proposition 2.16. *Semantic consequence is preserved by translation along signature morphisms: for any signature morphism $\sigma\colon \Sigma \to \Sigma'$, set Φ of Σ-equations, and Σ-equation φ,*

if $\Phi \models_\Sigma \varphi$ then $\sigma(\Phi) \models_{\Sigma'} \sigma(\varphi)$.

Proposition 2.17. *Let $\sigma\colon \Sigma \to \Sigma'$ be a signature morphism and let Φ' be a closed set of Σ'-equations. Then $\sigma^{-1}(\Phi')$ is a closed set of Σ-equations.*

A *theory* is a presentation $\langle \Sigma, \Phi \rangle$ such that Φ is closed. A presentation $\langle \Sigma, \Phi \rangle$ (where Φ need not be closed) *presents* the theory $\langle \Sigma, Cl_\Sigma(\Phi) \rangle$. A theory $\langle \Sigma, \Phi \rangle$ is sometimes referred to as a *Σ-theory*. For any theories $\langle \Sigma, \Phi \rangle$ and $\langle \Sigma', \Phi' \rangle$, a *theory morphism* $\sigma\colon \langle \Sigma, \Phi \rangle \to \langle \Sigma', \Phi' \rangle$ is a signature morphism $\sigma\colon \Sigma \to \Sigma'$ such that $\sigma(\varphi) \in \Phi'$ for every $\varphi \in \Phi$.

Example 2.18. Let Σ be the signature

$$\begin{array}{l} \Sigma = \textbf{sorts}\ s, b \\ \quad\ \textbf{opns}\ tt\colon b \\ \qquad\quad ff\colon b \\ \qquad\quad not\colon s \to b \\ \qquad\quad and\colon s \times b \to b \end{array}$$

and recall the presentation $Bool = \langle \Sigma Bool, \Phi Bool \rangle$ in Example 2.14. Define a signature morphism $\sigma\colon \Sigma \to \Sigma Bool$ by $\sigma_{sorts} = \{s \mapsto bool, b \mapsto bool\}$, $\sigma_{\varepsilon,b} = \{tt \mapsto true, ff \mapsto false\}$, $\sigma_{s,b} = \{not \mapsto \neg\}$, and $\sigma_{s\,b,b} = \{and \mapsto \wedge\}$. Let $\Phi = \{\ \forall x{:}s.\ and(x, and(x, not(x))) = ff, \forall x{:}s.\ and(x, ff) = ff\ \}$. Then $Cl_\Sigma(\Phi)$ includes Σ-equations that were not in Φ, such as the equation $\forall x, y{:}s.\ and(y, and(x, and(x, not(x)))) = ff$. The presentations $\langle \Sigma, Cl_\Sigma(\Phi) \rangle$ and $\langle \Sigma Bool, Cl_{\Sigma Bool}(\Phi Bool) \rangle$ are theories – the latter is the theory presented by $Bool$ – and $\sigma\colon \langle \Sigma, Cl_\Sigma(\Phi) \rangle \to \langle \Sigma Bool, Cl_{\Sigma Bool}(\Phi Bool) \rangle$ is a theory morphism. ∎

Proposition 2.19. *Let $\sigma\colon \Sigma \to \Sigma'$ be a signature morphism, Φ be a set of Σ-equations, and Φ' be a set of Σ'-equations. Then the following conditions are equivalent:*

1. *σ is a theory morphism $\sigma\colon \langle \Sigma, Cl_\Sigma(\Phi)\rangle \to \langle \Sigma', Cl_{\Sigma'}(\Phi')\rangle$.*
2. *$\sigma(\Phi) \subseteq Cl_{\Sigma'}(\Phi')$.*
3. *For every $A' \in Mod_{\Sigma'}(\Phi')$, $A'\big|_\sigma \in Mod_\Sigma(\Phi)$.*

2.8 Equational calculus

The set of consequences of a presentation $\langle \Sigma, \Phi\rangle$ has been defined in a model-theoretic way. In this section we present a calculus for deriving consequences of a set of equational axioms in a "syntactic" way. It turns out that these two notions of consequence coincide.

A Σ-equation φ is a *syntactic* (or *proof-theoretic*) *consequence* of Φ, written $\Phi \vdash_\Sigma \varphi$, if φ can be derived from Φ by application of the following inference rules:

Reflexivity:
$$\frac{}{\forall X.\ t = t} \quad t \in |T_\Sigma(X)|$$

Symmetry:
$$\frac{\forall X.\ t = t'}{\forall X.\ t' = t}$$

Transitivity:
$$\frac{\forall X.\ t = t' \qquad \forall X.\ t' = t''}{\forall X.\ t = t''}$$

Congruence:
$$\frac{\forall X.\ t_1 = t_1' \qquad \cdots \qquad \forall X.\ t_n = t_n'}{\forall X.\ f(t_1, \ldots, t_n) = f(t_1', \ldots, t_n')}$$

for $f\colon s_1 \times \cdots \times s_n \to s$ and $t_i, t_i' \in |T_\Sigma(X)|_{s_i}$ for $i \le n$

Instantiation:
$$\frac{\forall X.\ t = t'}{\forall Y.\ t[\theta] = t'[\theta]} \quad \theta\colon X \to |T_\Sigma(Y)|$$

Example 2.20. Recall the presentation $Bool = \langle \Sigma Bool, \Phi Bool\rangle$ from Example 2.14. The following derivation proves $\Phi Bool \vdash_{\Sigma Bool} \forall p{:}bool.\ \neg(p \wedge \neg false) = \neg p$:

$$\dfrac{\dfrac{\dfrac{}{\forall p{:}bool.\ p = p} \qquad \dfrac{\neg false = true}{\forall p{:}bool.\ \neg false = true}}{\dfrac{\forall p{:}bool.\ p \wedge \neg false = p \wedge true}{\forall p{:}bool.\ \neg(p \wedge \neg false) = \neg(p \wedge true)}} \qquad \dfrac{\forall p{:}bool.\ p \wedge true = p}{\forall p{:}bool.\ \neg(p \wedge true) = \neg p}}{\forall p{:}bool.\ \neg(p \wedge \neg false) = \neg p}$$

■

As mentioned above, $\vdash_\Sigma$ is both *sound* (only valid consequences may be derived) and *complete* (all valid consequences may be derived) for $\models_\Sigma$.

Theorem 2.21. *For any set Φ of Σ-equations and any Σ-equation φ, $\Phi \vdash_\Sigma \varphi$ if and only if $\Phi \models_\Sigma \varphi$.*

Simplifying the above calculus by omitting explicit quantifiers in equations yields an unsound system because algebras may have empty carrier sets. In particular, unused variables cannot always be removed from equations. The instantiation rule allows quantified variables to be eliminated when it is sound to do so [GM85].

2.9 Initial models

The class of algebras given by the loose semantics of a Σ-presentation always includes degenerate Σ-algebras with a single value of each sort in Σ, and usually includes unreachable Σ-algebras. Equational axioms are not sufficient to eliminate such obviously undesired models. One standard remedy is to take the so-called *initial semantics* of presentations.

Let A be a model of a presentation $\langle \Sigma, \Phi \rangle$. We say that A *contains junk* if it is not reachable, and that A *contains confusion* if it satisfies a ground Σ-equation that is not in $Cl_\Sigma(\Phi)$.

Example 2.22. Recall the presentation $Bool = \langle \Sigma Bool, \Phi Bool \rangle$ and its models $A1$, $A2$, and $A3$ given in Example 2.14. $A1$ contains confusion ($A1 \models_{\Sigma Bool} \mathit{true} = \mathit{false} \notin Cl_{\Sigma Bool}(\Phi Bool)$) but not junk; $A2$ contains junk (there is no ground $\Sigma Bool$-term t such that $t_{A2} = c \in |A2|_{bool}$) but not confusion; $A3$ contains neither junk nor confusion. There are models of $Bool$ containing both junk and confusion. ∎

A Σ-algebra $A \in Mod_\Sigma(\Phi)$ is an *initial model of* $\langle \Sigma, \Phi \rangle$ if for every $B \in Mod_\Sigma(\Phi)$ there is a unique Σ-homomorphism $h\colon A \to B$.

The initial models of an equational presentation are those that have no junk and no confusion. An initial model may be constructed as a quotient of the algebra T_Σ of ground Σ-terms by the least congruence generated by the axioms:

Theorem 2.23. *$\langle \Sigma, \Phi \rangle$ has an initial model.*

Proof sketch. An initial model of $\langle \Sigma, \Phi \rangle$ is the quotient $T_\Sigma / \equiv_\Phi$, where $\equiv_\Phi$ is the *Σ-congruence generated by Φ*: $t \equiv_\Phi t' \iff \Phi \models_\Sigma \forall \emptyset.\ t = t'$, for all $t, t' \in |T_\Sigma|$. The existence and uniqueness of a Σ-homomorphism from $T_\Sigma / \equiv_\Phi$ to any $B \in Mod_\Sigma(\Phi)$ follows from Proposition 2.6. □

Example 2.24. The model $T_{\Sigma Bool}/{\equiv_{\Phi Bool}}$ of *Bool* (see Example 2.14) is defined as follows:

$$|T_{\Sigma Bool}/{\equiv_{\Phi Bool}}|_{bool} = \{[true]_{\equiv_{\Phi Bool}}, [false]_{\equiv_{\Phi Bool}}\}$$
$$true_{T_{\Sigma Bool}/\equiv_{\Phi Bool}} = [true]_{\equiv_{\Phi Bool}}$$
$$false_{T_{\Sigma Bool}/\equiv_{\Phi Bool}} = [false]_{\equiv_{\Phi Bool}}$$
$$\neg_{T_{\Sigma Bool}/\equiv_{\Phi Bool}} = \{[true]_{\equiv_{\Phi Bool}} \mapsto [false]_{\equiv_{\Phi Bool}}, [false]_{\equiv_{\Phi Bool}} \mapsto [true]_{\equiv_{\Phi Bool}}\}$$

$\wedge_{T_{\Sigma Bool}/\equiv_{\Phi Bool}}$	$[true]_{\equiv_{\Phi Bool}}$	$[false]_{\equiv_{\Phi Bool}}$
$[true]_{\equiv_{\Phi Bool}}$	$[true]_{\equiv_{\Phi Bool}}$	$[false]_{\equiv_{\Phi Bool}}$
$[false]_{\equiv_{\Phi Bool}}$	$[false]_{\equiv_{\Phi Bool}}$	$[false]_{\equiv_{\Phi Bool}}$

where

$$[true]_{\equiv_{\Phi Bool}} = \{true, \neg false, \neg(false \wedge true), \neg(false \wedge \neg false), \dots\},$$
$$[false]_{\equiv_{\Phi Bool}} = \{false, \neg true, \neg(true \wedge true), \neg(true \wedge \neg false), \dots\}.$$

This is an initial model of *Bool* by the proof sketched for Theorem 2.23. $\Sigma Bool$-homomorphisms from $T_{\Sigma Bool}/{\equiv_{\Phi Bool}}$ to $A1$, $A2$, and $A3$ are as follows:

$$h1\colon T_{\Sigma Bool}/{\equiv_{\Phi Bool}} \to A1$$
$$h1_{bool} = \{[true]_{\equiv_{\Phi Bool}} \mapsto \star, [false]_{\equiv_{\Phi Bool}} \mapsto \star\}$$

$$h2\colon T_{\Sigma Bool}/{\equiv_{\Phi Bool}} \to A2$$
$$h2_{bool} = \{[true]_{\equiv_{\Phi Bool}} \mapsto a, [false]_{\equiv_{\Phi Bool}} \mapsto b\}$$

$$h3\colon T_{\Sigma Bool}/{\equiv_{\Phi Bool}} \to A3$$
$$h3_{bool} = \{[true]_{\equiv_{\Phi Bool}} \mapsto 1, [false]_{\equiv_{\Phi Bool}} \mapsto 0\}$$

■

Taking a presentation to denote the (non-empty) class of its initial models is called taking its *initial semantics*. The initiality property identifies a model of $\langle \Sigma, \Phi \rangle$ up to isomorphism: any two initial models are isomorphic, and any model isomorphic to an initial model is itself initial. We therefore refer to *the* initial model of a presentation.

Example 2.25. $A3$ is an initial model of *Bool* (see Example 2.14) since it is isomorphic to $T_{\Sigma Bool}/{\equiv_{\Phi Bool}}$. On the other hand, $A1$ and $A2$ are not isomorphic to $A3$ and hence are not initial models. This can be checked directly as well: for example, $\nexists h\colon A1 \to A2$ and $\nexists h\colon A1 \to A3$. ■

2.10 Variations on a theme

The simple specification framework presented above is the classical one in the field of algebraic specifications. A wide variety of modifications have been made to increase its expressive power and to take account of the various features of software systems which it does not handle adequately. This section is devoted to a sketch of some of these modifications; details may be found in the cited references.

2.10.1 Conditional equations

Equational axioms can be generalized to (positive) conditional equational axioms of the form $\forall X.\ t_1 = t_1' \wedge \ldots \wedge t_n = t_n' \Rightarrow t_0 = t_0'$. A Σ-algebra A *satisfies* such an axiom if for every (S-sorted) function $v\colon X \to |A|$, if $v^\#(t_1) = v^\#(t_1')$ and $\ldots$ and $v^\#(t_n) = v^\#(t_n')$, then $v^\#(t_0) = v^\#(t_0')$. With these changes, most results still apply with appropriate minor modifications. For example, any presentation $\langle \Sigma, \Phi \rangle$, where Φ is a set of conditional Σ-equations, has an initial model which can be constructed in a similar way as in the proof of Theorem 2.23 (see, e.g., [MT92]). There is also a sound and complete proof system for conditional equational consequence [Sel72].

2.10.2 Partial algebras

An obvious way to generalize the standard definition of an algebra is to allow partial functions as interpretations of operation names. Homomorphisms between such *partial algebras* are required to preserve definedness of operations, and (as usual) their results when these are defined. Term evaluation is defined as in ordinary algebras, except that terms need not have defined values. An equation $\forall X.\ t = t'$ is satisfied in a partial algebra A when for all valuations $v\colon X \to |A|$, the values of t and t' under v either are defined and equal, or are both undefined. Additional axioms are required to assert definedness: $\forall X.\ D(t)$ holds in A when the value of t is defined under all valuations of X in A. Every presentation $\langle \Sigma, \Phi \rangle$, where Φ is a set of Σ-equations and definedness formulas, has an initial model which contains no junk, is *minimally defined* (i.e., the value of a ground term t is defined only if $\Phi \models_\Sigma \forall \emptyset.\ D(t)$), and contains no confusion, i.e., the values of two ground terms t, t' are defined and equal only if $\Phi \models_\Sigma \forall \emptyset.\ t = t'$.

This is one possible approach to the specification of partial algebras, following [BW82]. There are various other choices for the basic definitions [Rei87, Bur86].

2.10.3 Error algebras

To model operations that may produce erroneous or exceptional results, we can partition each of the carrier sets of an algebra into an *error* part and an *OK* part. Operations in signatures are classed as *safe* or *unsafe*, where the former are required to yield OK values when applied to OK arguments. Homomorphisms are required to preserve OK-ness. Like operations, variables in equations are classed as safe or unsafe; the former range over OK values only, while the latter range over all values. Again, all presentations have initial models, in which operations propagate errors unless otherwise specified.

The details of this approach may be found in [GDLE84]. Again, there are many other approaches, see for instance [Gog78] or [BBC86].

2.10.4 Order-sorted algebras

In order to model sort inclusion and coercions, signatures may be enriched with an order relation on the set of sorts. An *order-sorted Σ-algebra* A is required to respect the sort ordering in the *order-sorted signature* Σ: if $s \leq s'$ in Σ then we require that $|A|_s \subseteq |A|_{s'}$. Overloading is forced by requiring operations to be applicable to values from subsorts of their argument sorts and to yield results in supersorts of their result sorts. Under certain conditions terms are guaranteed to have least sorts and unambiguous values. Then once more, all presentations have initial models and there is a version of the equational calculus that is sound and complete for order-sorted satisfaction.

For details see [GM85]. Alternative approaches are [Gog84, Poi84, Smo86]; see also [Mos93, GD94, CHKM97].

2.10.5 First-order predicate logic

Signatures may be modified to enable them to include (typed) *predicate names* in addition to operation names, e.g., $\leq\colon nat \times nat$. Atomic formulas are then formed by applying predicates to terms; in *first-order predicate logic with equality*, the predicate $=\colon s \times s$ is implicitly available for any sort s. Formulas are built from atomic formulas using the usual logical connectives and quantifiers. Algebras are modified to include relations on their carriers to interpret predicate names (giving what are sometimes called *relational structures*). Homomorphisms are required to preserve predicates as well as operations. The satisfaction of a *sentence* (a formula without free variables) by an algebra is as usual in first-order logic. Presentations involving predicates and first-order axioms do not always have initial models or even reachable models. Details of first-order predicate logic for use in algebraic specifications may be found in, e.g., [GB92].

2.10.6 Higher-order functions

Higher-order functions (which take functions as parameters and/or return functions as results) can be accommodated by interpreting certain sort names as (subsets of) function spaces. Given a set S of (base) sorts, let $S^{\rightarrow}$ be the closure of S under formation of function types: $S^{\rightarrow}$ is the smallest set such that $S \subseteq S^{\rightarrow}$ and for all $s_1, \ldots, s_n, s \in S^{\rightarrow}$, $s_1 \times \cdots \times s_n \rightarrow s \in S^{\rightarrow}$. Then a *higher-order signature* Σ is a pair $\langle S, \Omega \rangle$ where Ω is an $S^{\rightarrow}$-indexed set of operation names. This determines an ordinary signature $\Sigma^{\rightarrow}$ comprised of the sort names $S^{\rightarrow}$ and the operation names in Ω (as constants of sorts in $S^{\rightarrow}$) together with operation names $apply\colon (s_1 \times \cdots \times s_n \rightarrow s) \times s_1 \times \cdots \times s_n \rightarrow s$ for every $s_1, \ldots, s_n, s \in S^{\rightarrow}$. A *higher-order Σ-algebra* is just an ordinary (total) $\Sigma^{\rightarrow}$-algebra, and analogously for the definitions of higher-order Σ-homomorphism, higher-order Σ-equation, higher-order presentation, etc. A higher-order Σ-algebra A is *extensional* if for all sorts $s_1 \times \cdots \times s_n \rightarrow s \in S^{\rightarrow}$

and values $f, g \in |A|_{s_1 \times \cdots \times s_n \to s}$, $f = g$ whenever $apply_A(f, a_1, \ldots, a_n) = apply_A(g, a_1, \ldots, a_n)$ for all $a_1 \in |A|_{s_1}, \ldots, a_n \in |A|_{s_n}$. In an extensional algebra A, every carrier $|A|_{s_1 \times \cdots \times s_n \to s}$ is isomorphic to a subset of the function space $|A|_{s_1} \times \cdots \times |A|_{s_n} \to |A|_s$. Higher-order equational presentations always have initial extensional reachable models. See [MTW88] for details, and for alternative approaches see, e.g., [Poi86, Mei92].

2.10.7 Polymorphic types

Programming languages such as Standard ML [Pau96] can be used to define *polymorphic types* such as α *list* and *polymorphic values* such as the function *head*:$\forall\alpha.\ \alpha$ *list* $\to \alpha$. To specify such types and functions, signatures are modified to contain *type constructors* in place of sort names. Terms built using these type constructors and *type variables* (such as α above) are the *polymorphic types* of the signature. The set Ω of operation names is then indexed by non-empty sequences of polymorphic types, where $f \in \Omega_{t_1 \ldots t_n, t}$ means $f\colon \forall FV(t_1) \cup \ldots \cup FV(t_n) \cup FV(t).\ t_1 \times \cdots \times t_n \to t$. There are various choices for algebras over such signatures. The most straightforward is to require each algebra A to incorporate a (single-sorted) *algebra of carriers*, $Carr(A)$, having sets which interpret types as values and an operation to interpret each type constructor. Then, for each operation $f \in \Omega_{t_1 \ldots t_n, t}$ and for each instantiation of type variables $i\colon V \to |Carr(A)|$, A has to provide a function $f_{A,i}\colon i^\#(t_1) \times \cdots \times i^\#(t_n) \to i^\#(t)$. Various conditions may be imposed to ensure that the interpretation of polymorphic operations is *parametric*, by requiring $f_{A,i}$ and $f_{A,i'}$ to be appropriately related for different type variable instantiations i, i'. Axioms contain (universal) quantifiers for type variables in addition to quantifiers for ordinary variables, as in System F [GLT89]; alternatively, type-variable quantification may be left implicit, as in Extended ML [KST97].

2.10.8 Non-deterministic functions

Non-deterministic functions may be handled by interpreting operation names in algebras as relations or, equivalently, as set-valued functions. Homomorphisms are required to preserve possible values of functions: for any homomorphism $h\colon A \to B$ and operation $f\colon s_1 \times \cdots \times s_n \to s$, if a is a possible value of $f_A(a_1, \ldots, a_n)$ then $h_s(a)$ is a possible value of $f_B(h_{s_1}(a_1), \ldots, h_{s_n}(a_n))$. Universally quantified inclusions between sets of possible values may be used as axioms: $t \subseteq t'$ means that every possible value of t is a possible value of t'. See [Nip86, Hußß89, BS93, BK98] for details.

2.10.9 Continuous algebras

Following [Sco76], partial functions may be specified as least solutions of recursive equations. To accommodate this, we can use *continuous algebras*, i.e.,

ordinary (total) Σ-algebras with carriers that are complete partially ordered sets (so-called *cpos*) and operation names interpreted as *continuous functions* on these sets. The "bottom" element $\perp$ of the carrier for a sort, if it exists, represents the completely undefined value of that sort. The order on carriers induces an order on (continuous) functions in the usual fashion. A homomorphism between continuous algebras is required to be continuous as a function between cpos. For details see, e.g., [GTWW77]. It is possible to define a language of axioms that allows direct reference to least upper bounds of chains and/or to the order relation itself (see, e.g., [TW86]).

Bibliography

[BBC86] G. Bernot, M. Bidoit, and C. Choppy. Abstract data types with exception handling: an initial approach based on a distinction between exceptions and errors. *Theoretical Computer Science*, 46(1):13–45, 1986.

[BG80] R.M. Burstall and J.A. Goguen. The semantics of CLEAR, a specification language. In D. Björner, editor, *Proc. Copenhagen Winter School on Abstract Software Specification*, volume 86 of *Lecture Notes in Computer Science*, pages 292–332. Springer, 1980.

[BK98] M. Białasik and B. Konikowska. A logic for nondeterministic specifications. In E. Orłowska, editor, *Logic at work. Essays dedicated to the memory of H. Rasiowa*. Kluwer, 1998.

[BS93] R. Berghammer and G. Schmidt. Relational specifications. In *Algebraic Methods in Logic and Computer Science*, volume 28 of *Banach Center Publications*, pages 167–190. Institute of Mathematics, Polish Academy of Sciences, Warsaw, 1993.

[Bur86] P. Burmeister. *A Model Theoretic Oriented Approach to Partial Algebras*. Akademie-Verlag, Berlin, 1986.

[BW82] M. Broy and M. Wirsing. Partial abstract types. *Acta Informatica*, 18(1):47–64, 1982.

[CHKM97] M. Cerioli, A. Hauxthausen, B. Krieg-Brückner, and T. Mossakowski. Permissive subsorted partial logic in CASL. In Michael Johnson, editor, *Algebraic Methodology and Software Technology (AMAST'97)*, volume 1349 of *Lecture Notes in Computer Science*, pages 91–107. Springer, 1997.

[EM85] Hartmut Ehrig and Bernd Mahr. *Fundamentals of Algebraic Specification 1: Equations and Initial Semantics*, volume 6 of *EATCS Monographs on Theoretical Computer Science*. Springer, 1985.

[GB92] J.A. Goguen and R.M. Burstall. Institutions: abstract model theory for specification and programming. *Journal of the Association for Computing Machinery*, 39(1):95–146, 1992.

[GD94] J. Goguen and R. Diaconescu. An Oxford survey of order sorted algebra. *Mathematical Structures in Computer Science*, 4:363–392, 1994.

[GDLE84] M. Gogolla, K. Drosten, U. Lipeck, and H.-D. Ehrich. Algebraic and operational semantics of specifications allowing exceptions and errors. *Theoretical Computer Science*, 34:289–313, 1984.

[GLT89] J.-Y. Girard, Y. Lafont, and P. Taylor. *Proofs and Types*. Cambridge University Press, 1989.

[GM85] J. Goguen and J. Meseguer. Completeness of many-sorted equational logic. *Houston Journal of Mathematics*, 11(3):307–334, 1985.

[Gog78] J.A. Goguen. Abstract errors for abstract data types. In *Proc. IFIP Working Conference on the Formal Description of Programming Concepts*. North-Holland, 1978.

[Gog84] M. Gogolla. Partially ordered sorts in algebraic specifications. In *Proc. 9th Colloquium on Trees in Algebra and Programming*, pages 139–153. Cambridge University Press, 1984.

[GTWW77] J. Goguen, J. Thatcher, E. Wagner, and J. Wright. Initial algebra semantics and continuous algebras. *Journal of the Association for Computing Machinery*, 24(1):68–95, 1977.

[Huß89] H. Hußmann. *Nichtdeterministische algebraische Spezifikation*. PhD thesis, Universität Passau, 1989.

[KST97] S. Kahrs, D. Sannella, and A. Tarlecki. The definition of Extended ML: a gentle introduction. *Theoretical Computer Science*, 173:445–484, 1997.

[LEW96] J. Loeckx, H.-D. Ehrich, and B. Wolf. *Specification of Abstract Data Types*. Wiley, 1996.

[Mei92] K. Meinke. Universal algebra in higher types. *Theoretical Computer Science*, 100(2):385–417, 1992.

[Mos93] Peter D. Mosses. The use of sorts in algebraic specifications. In M. Bidoit and C. Choppy, editors, *Recent Trends in Data Type Specification, Selected Papers from the 8th Workshop on Specification of Abstract Data Types*, volume 655 of *Lecture Notes in Computer Science*, pages 66–91. Springer, 1993.

[MT92] K. Meinke and J. Tucker. Universal algebra. In S. Abramsky, D. Gabbay, and T. Maibaum, editors, *Handbook of Logic in Computer Science, Vol. 1*, pages 189–411. Oxford University Press, 1992.

[MTW88] B. Möller, A. Tarlecki, and M. Wirsing. Algebraic specifications of reachable higher-order algebras. In *Recent Trends in Data Type Specification, Selected Papers from the 5th Workshop on Specification of Abstract Data Types*, volume 332 of *Lecture Notes in Computer Science*, pages 154–169. Springer, 1988.

[Nip86] T. Nipkow. Non-deterministic data types: models and implementations. *Acta Informatica*, 22:629–661, 1986.

[Pau96] L.C. Paulson. *ML for the Working Programmer*. Cambridge University Press, 2nd edition, 1996.

[Poi84] A. Poigné. Another look at parameterization using algebraic specifications with subsorts. In *Proc. 11th Symp. on Mathematical Foundations of Computer Science*, volume 176 of *Lecture Notes in Computer Science*, pages 471–479. Springer, 1984.

[Poi86] A. Poigné. On specifications, theories, and models with higher types. *Information and Control*, 68:1–46, 1986.

[Rei87] H. Reichel. *Initial Computability, Algebraic Specifications, and Partial Algebras*. Oxford University Press, 1987.

[Sco76] D. Scott. Data types as lattices. *SIAM Journal of Computing*, 5:522–587, 1976.

[Sel72] A. Selman. Completeness of calculi for axiomatically defined classes of algebras. *Algebra Universalis*, 2:20–32, 1972.

[Smo86] G. Smolka. Order-sorted Horn logic: semantics and deduction. SEKI report SR-86-17, FB Informatik, Universität Kaiserslautern, 1986.

[ST] D. Sannella and A. Tarlecki. *Foundations of Algebraic Specifications and Formal Program Development*. Cambridge University Press. To appear.

[TW86] A. Tarlecki and M. Wirsing. Continuous abstract data types. *Fundamenta Informaticae*, 9:95–126, 1986.

[Wir90] Martin Wirsing. Algebraic specification. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, Vol. B*, chapter 13, pages 675–788. Elsevier Science Publishers B.V. (North Holland), 1990.