Algebraic Abstract Data Types : Formalism, Models and Properties

Didier Buchs

Université de Genève

7 octobre 2021

Algebraic Abstract Data Types

- Models
- Satisfaction
- Checking syntactic properties
- Equational Theories
- Inductive Theories
- Hierarchical specification and semantic properties

Definition of algebraic specification

Definition (algebraic specification)

A *many sorted algebraic specification* $Spec = \langle S, F, X, AX \rangle$ is a signature extended by a collection of axioms E on variables X.

In what follows, let $\Sigma = \langle S, F \rangle$ be a complete signature.

manque pas de Sort

Notion of models/Implementation

We can consider models i.e. structures that represents the semantics of the specification as potential implementations :

- An implementation is acceptable if it satisfies the requirements.
- The requirements are to respect the interface and the axioms.
- The structures that respect the interface Σ are called $\Sigma - algebra$ noted $Alg(\Sigma)$. (and maybe not the axioms)
- if $Spec = \langle S, F, X, AX \rangle$ and $\Sigma = \langle S, F \rangle$ then $Mod(Spec) \subseteq Alg(\Sigma)$.

Notion of models/Implementation

Definition (Models)

Given a specification $Spec =< \Sigma, X, AX >$, the class of its models is : $Mod(Spec)$ and $\forall ax \in AX, \forall M \in Mod(Spec), M \vdash ax$

modèles satisfont les axiomes

A model is a set of values and operations called $\Sigma - algebra$, we will detail roughly this notion later as well as the notion of satisfaction of the models $\vdash$.

Moreover it is convenient to be able to associate models throug the notion of morphisms (function that respect the sort of the values).

Notion of evaluation

It must be noted that there exist a unique 'morphism' $eval : T_\Sigma \to A$, where $A \in Mod(Spec)$, extended with interpretations $I : X \to A$ to a unique $eval_I$.

We can also define it recursively over the structure of the terms.

Remark :

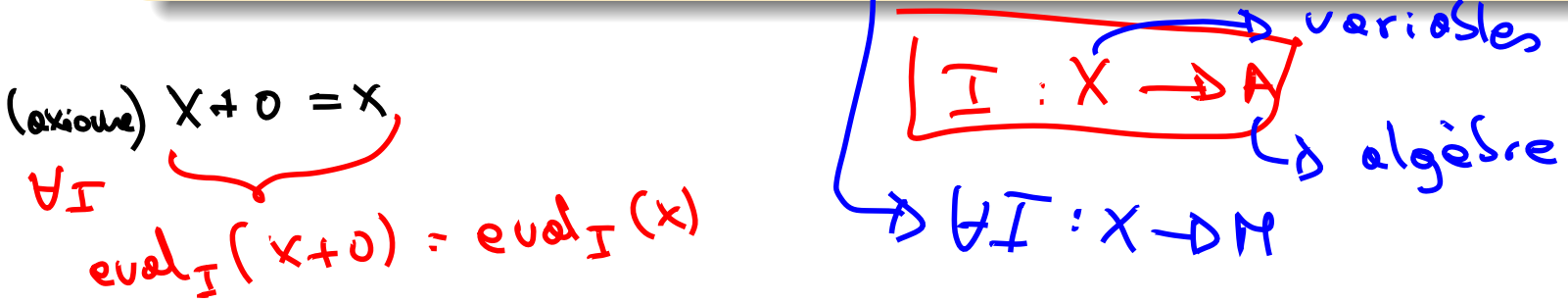
- It is then obviously a morphism (see later)
- The unicity comes from the unique interpretation of the symbols in the algebra and the fact that interpretations are functions.

Notion of satisfaction of models/Implementation

Using evaluation we can 'check' that axioms are valid for any values of the variables (implicit universal quantification of the variables).

Definition (Satisfaction relation)

Given a specification $Spec =< \Sigma, X, AX >$, and $t_1, t_2 \in T_{\Sigma,X}, M \in Mod(Spec), M \vdash t_1 = t_2 \Leftrightarrow \forall I, eval_I(t_1) = eval_I(t_2)$



Definition of models

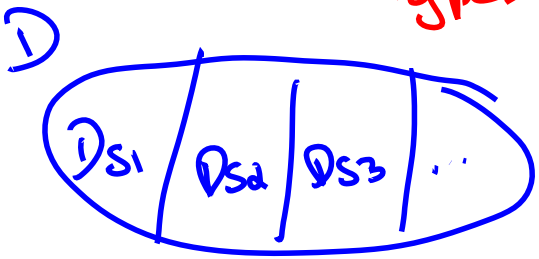
Definition (Σ-Algebra)

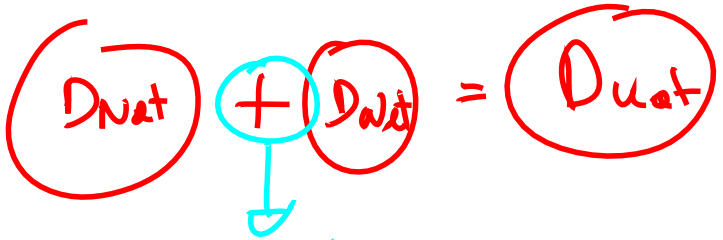
$\Sigma = \langle S, F \rangle$

A Σ-algebra is a couple $A = \langle D, O \rangle$, in which :

- D is a *S-sorted set* of values $(D = D_{s_1} \amalg \ldots \amalg D_{s_n})$
- and O is a set of functions, such that for each function name $f \in O_{w,s}, w = s_1 \ldots s_n$ there is a function $f^A \in O$, defined as $f^A : D_{s_1} \times \ldots \times D_{s_n} \rightarrow D_s$.

types de données





voir saved images for 3 examples of algebres for boolea

Definition of model homomorphisms

Definition (Σ-Morphism)

Given $A = (D_A, O_A), B = (D_B, O_B)$ Σ-algebras. A Σ-Morphism is an application $\mu : A \to B$ st.
for all $f \in O_{w,s}, f \in O^A_{w,s}, f \in O^B_{w,s}, w = s_1 \dots s_n$,
$d_1, ..., d_n : d_1 \in D_{A,s_1}, ...d_n \in D_{A,s_n}$
$\mu(f^A(d_1, ..., d_n)) = f^B(\mu(d_1), ..., \mu(d_n))$.

eval is a Σ-Morphism

What about properties ?

If we consider the specification of naturals :

```
x + 0 = x;
x + succ(y) = succ(x+y);
```

What is :

```
y + succ(succ(0))  = succ(succ(y))?
```

and :

```
succ(succ(0)) + y  = succ(succ(y))?
```

so, what about :

```
x+y = y + x ?
```

Proofs in algebraic specifications :Plan

- Equational Proofs :
  - Equational theories
  - Inductive theories
  - Deduction systems
- Rewriting (next chapter)

Proofs in algebraic specifications

Aim : Proof of specification properties in a systematic way. How to proceed from the axioms :

- Use equational rules (reflexivity,symmetry, transitivity)
- Use functional composition rules
- Use variable substitution rules

$\Rightarrow$ we obtain equational theorems

Example : Proofs in algebraic specifications

To prove : succ(0) + succ(succ(0)) = succ(succ(succ(0)))

axiom : succ(x) + y = succ(x + y) **substitution** rule with s= {x = 0, y = succ(succ(0))}

(1) succ(0) + succ(succ(0)) = succ(0 +succ(succ(0)))

vrai grâce à l'axiome

axiom : 0+ x = x

substitution rule with s= {x = succ(succ(0))}

(2) 0 + succ(succ(0)) = succ(succ(0))

Substitutivity rule with operation succ on (2)

(3) succ(0 + succ(succ(0))) = succ(succ(succ(0)))

Transitivity rule : (1) and (3) ⇒

succ(0) + succ(succ(0)) = succ(succ(succ(0)))

CQFD

Limits of deduction theories

Problem : interesting theorems are more complex : $x + y = y + x$
Which is not deductible in the deduction system.

Inductive definition of equational theories

Given $Spec =< \Sigma, X, AX >$, $\Sigma =< S, OP >$ an algebraic specification
For any $t, t', t_i, t'_i \in (T_{\Sigma,X})_s$ the definition of $Th(Spec)$ is :

① axiome → théorème
② ∀ terme, $t = t$
③ $t = t' \Rightarrow t' = t$
⑤ $f(x) = f(x')$ si $x = x'$

① $Axioms : t = t' \in Ax \Rightarrow t = t' \in Th(Spec)$

② $Reflexivity : \forall t \in (T_{\Sigma,X})_s, t = t \in Th(Spec)$

③ $Symmetry : t = t' \in Th(Spec) \Rightarrow t' = t \in Th(Spec)$

④ $Transitivity : t = t' \in Th(Spec) \wedge t' = t'' \in Th(Spec) \Rightarrow t = t'' \in Th(Spec)$

⑤ $Substitutivity : \forall f \in OP, t_1 = t'_1 \in Th(Spec) \wedge \ldots\ldots \wedge t_n = t'_n \in Th(Spec)$
$\Rightarrow f(t_1, t_2, ..., t_n) = f(t'_1, t'_2, ..., t'_n) \in Th(Spec)$

$Subst. : x \in X_s, u \in (T_{\Sigma,X})_s, t = t' \in Th(Spec) \Rightarrow t[u/x] = t'[u/x] \in Th(Spec)$
substitution

$Cut : Cond_1 \wedge u = u' \wedge Cond_2 \Rightarrow t = t' \in Th(Spec)$
$and\ Cond \Rightarrow u = u' \in Th(Spec)$
$then\ Cond_1 \wedge Cond \wedge Cond_2 \Rightarrow t = t' \in Th(Spec)$

Equational theory validity and completeness

Theorem (validity and completeness)

Given $Spec =< \Sigma, X, AX >$,
$\forall t_1, t_2 \in T_{\Sigma,X}$,
$t_1 = t_2 \in Th(Spec) \Leftrightarrow \forall M \in Mod(Spec), M \vdash t_1 = t_2$

This is the validity and completeness of the deduction

il peut exister des modèles ou d'autres égalités

Theories and properties in implementations

$\forall M \in Mod(Spec), M \vdash t_1 = t_2$ indicates that the equation is valid in all implementations

Some equations are valid in only specific implementation (ex : true = false). This is for instance the case for very simple models such as the final one.

Exercices :

prove : succ(succ(0)) - succ(succ(0)) = 0

prove : succ(succ(succ(0))) - succ(succ(0)) = 0

prove : x - x = 0

Inductive theories

Aim : provide more general theorems deductible from the axioms.
Additional rule :

Definition (Induction rule)

Given G a formula such that x is a free variable,
If $\forall t, G[t/x] \in Th(Spec) \Rightarrow \forall t, G \in Th_{Ind}(Spec)$

Remark : $Th(Spec) \subseteq Th_{Ind}(Spec)$ and the induction rule define the inductive theories.

Inductive definition of inductive theories

Given $Spec =< \Sigma, X, AX >$, $\Sigma =< S, OP >$ an algebraic specification
For any $t, t', t_i, t'_i \in (T_{\Sigma,X})_s$ the definition of $Th(Spec)$ is :

$$Axioms : t = t' \in Ax \Rightarrow t = t' \in Th_{Ind}(Spec)$$
$$Reflexivity : \forall t \in (T_{\Sigma,X})_s, t = t \in Th_{Ind}(Spec)$$
$$Symmetry : t = t' \in Th_{Ind}(Spec) \Rightarrow t' = t \in Th_{Ind}(Spec)$$
$$Transitivity : t = t' \in Th_{Ind}(Spec) \wedge t' = t'' \in Th_{Ind}(Spec) \Rightarrow$$
$$t = t'' \in Th_{Ind}(Spec)$$
$$Substitutivity : \forall f \in OP, t_1 = t'_1 \in Th_{Ind}(Spec) \wedge ..... \wedge t_n = t'_n \in Th_{Ind}(Spec)$$
$$\Rightarrow f(t_1, t_2, ..., t_n) = f(t'_1, t'_2, ..., t'_n) \in Th_{Ind}(Spec)$$
$$Subst. : x \in X_s, u \in (T_{\Sigma,X})_s, t = t' \in Th_{Ind}(Spec) \Rightarrow$$
$$t[u/x] = t'[u/x] \in Th_{Ind}(Spec)$$

...

Inductive definition of inductive theories (cnt'd)

Given $Spec =< \Sigma, X, AX >$, $\Sigma =< S, OP >$ an algebraic specification
For any $t, t', t_i, t'_i \in T_{\Sigma,s}$ the definition of $Th(Spec)$ is :

...

$$Cut : Cond_1 \wedge u = u' \wedge Cond_2 \Rightarrow t = t' \in Th_{Ind}(Spec)$$
$$and\ Cond \Rightarrow u = u' \in Th_{Ind}(Spec)$$
$$then\ Cond_1 \wedge Cond \wedge Cond_2 \Rightarrow t = t' \in Th_{Ind}(Spec)$$
$$Induction : x \in (X_s \cap (Var(t) \cup Var(t'))),$$
$$\bigwedge_{v_i \in (T_{\Sigma,X})_s} (t = t')[v_i/x] \in Th_{Ind}(Spec) \Rightarrow t = t' \in Th_{Ind}(Spec)$$

How to prove such often infinite conjunction of specific proofs ?

By structural induction on the generators :

- it can be done rather informally in a natural deduction style.
- Formally using judgment, from sequent calculus, of the form. $t_1 = t_2 \vdash t_1 = t_2[f(x)]$, where f is a generator.

validity and completeness

Theorem (validity and completeness)

Given $Spec =< \Sigma, X, AX >$,
$\forall t_1, t_2 \in T_{\Sigma,X}$,
$t_1 = t_2 \in Th_{Ind}(Spec) \Leftrightarrow \forall M \in Mod_{Gen}(Spec), M \vdash t_1 = t_2$

This is the validity and completeness of inductive theories for finitely generated models, i.e. models where all values are reachable from a syntactic term (eval is surjective)

Example of induction

Given the boolean specification with operations : true, false et not
We want to deduce : not(not(b)) = b
The predicate defining the property can be written :
P(b) = not(not(b)) = b
Proof :
Base cases : P(true) = not(not(true)) = not(false) = true ;
P(false) = not(not(false)) = not(true) = false ;
Induction Step : not(not(b)) = b implies not(not(not(b))) = not(b)
substitutability rule with 'not' applied to not(not(b)) = b
implies not(not(not(b))) = not(b)
It must be noted that having finitely generated models wrt the generators allow to only use generators in the proofs.

Exercise

Prove the commutativity of addition within naturals with 0, succ, + and the axioms

$x+0 = x$

$x + \text{succ } y = \text{succ } (x + y)$:

$P(x,y) = x + y = y + x$

Exercise ctn'd

Properties in initial and final models

The models can be ordered following an order relation :
$M_1 \leq M_2 \Leftrightarrow Th(M_1) \subseteq Th(M_2)$
Where $Th(M) \Leftrightarrow \{t_1 = t_2 | t_1, t_2 \in T_{\Sigma,X}$ and $M \vdash t_1 = t_2\}$

- The order relation forms a lattice for Horn clauses.
- All equalities between close terms valid in the initial models are valid in the other models (The lower model).
- Initial model $\Rightarrow$ minimal set of equalities between close terms. There is only one initial model for Horn clause theories.
- For each equality valid in the final model there exists a model different from the initial model for which this equality is true.
- final model $\Rightarrow$ maximal set of equalities between close terms, i.e such that $\forall t_1, t_2 \in T_{\Sigma,X}$, $Triv \vdash t_1 = t_2$

Example : $Triv_{Bool} \vdash true = false$

Contradiction and incompleteness

What is happening if contradiction occurs ?

What is happening if incomplete definitions are given ?

In fact these notions are not well defined !

Hierarchies in Algebraic Specifications

- Deal with progressive development of systems
- Flat specification implies no bound on the effect of axioms.
- There is no isolation mechanism in standard logic.
- Need to isolate properties $\Rightarrow$
  no perturbation over hierarchical levels when using specifications.

Hierarchical Specification

- Hierarchical constraints $\Rightarrow$ reflect decomposition of the process of building specifications.
- The modules of the specification should be implemented separately.
- Kind of perturbations :
  - 'junk' : values are added when a module is extended $\Rightarrow$ sufficient completeness.
  - 'confusion' : values are collapsed when extending a module $\Rightarrow$ hierarchical consistency.

Hierarchical models

HMod(Spec) s.t. Spec = ΔSpec + Spec0
The restriction (forget) to sub-modules will preserve their semantics. [1] The semantics of the ground module is chosen as an initial semantics (for ex. booleans with true ≠ false)

Definition (Hierarchical models)

$$HMod(Spec) = \{m \in Mod(Spec) | U(m) \in Mod(Spec0)\}$$

1. The forgetful functor is noted U.

Example :Suffisent Completeness

```
Adt Passuffcomplet;
Interface
  Use Naturals,Booleans;
  Operations
     f : natural-> boolean;
   Body
     Axioms
         f(succ(x)) = false;
      Where x: natural;
End Passuffcomplet;
```

$\Rightarrow$ problem : a new value exists : f(0) of type boolean in the initial model.

Example : hierarchical Consistency

```
Adt Pasconsistant;
Interface
  Use Naturals,Booleans;
  Operations
    f : natural-> boolean;
  Body
     Axioms
       f(succ(x)) = false;
       f(0) = true;
       f(succ(succ(x)) = true;
   Where x: natural;
End Pasconsistant;
```

$\Rightarrow$ problem : we have now that true $=$ false in the initial model.

Other algebraic specification formalisms

Various extensions of the basic presented approach :

- Partial functions $\Rightarrow$ definition predicate
- Sub-sort definitions $\Rightarrow$ inclusion relation between algebras
- Exceptional cases $\Rightarrow$ exceptions as labels on values
- and : State algebras, concurrency, bounds ...

Exercise : Modeling Trees (1)

Exercise : Modeling Trees (2)

Non determinism in implementation

Naive approach to define non-deterministim in implementation :

```
Adt NaiveNondeterminism;

f: natural -> boolean;

Axiom
  f(x) = b;

  Where x:natural; b : boolean;
```

What are the resulting properties of such specification ?
x, b are universally quantified $\Rightarrow$ i.e. $f(0) = true$ and
$f(0) = false \Rightarrow true = false$
Which is not a valid hierarchical model

Correct non determinism in implementation

Consistent non-deterministim in implementation :

```
Adt Nondeterminism;
f: natural -> boolean;
Axiom
  f(x) = b => ((b = true) or (b=false)) = true;

  Where x:natural; b : boolean;
```

What are the resulting properties of such specification ?
x, b are universally quantified $\Rightarrow$ but in the condition it is existentially quantified .
As the functions are deterministic, a correct model has a 'f' function which choose a boolean value.
Exercise :
define the choose function in sets.

Summary

- Deductive and inductive theories have been presented
- The valid properties of the initial model are also valid in all models
- The use of finitelly generated models wrt to constructors simplify inductive proofs.