Initial Specifications

Wolfgang Schreiner
Wolfgang.Schreiner@risc.uni-linz.ac.at

Research Institute for Symbolic Computation (RISC)
Johannes Kepler University, Linz, Austria
http://www.risc.uni-linz.ac.at

Initial Specifications

We fix as our logic the equational logic EL.

- Initial specification $sp = (\Sigma, \Phi)$.
  - Signature Σ, set of formulas $\Phi \subseteq EL(\Sigma)$.
- Semantics $\mathcal{M}(sp) = \{A \in Alg(\Sigma) \mid A \simeq T(\Sigma, \Phi)\}$.
  - The class of algebras isomorphic to the quotient term algebra over Φ.
  - Carriers are (isomorphic to) classes of terms that have the same value in all models of Φ.

An initial specification specifies as the abstract datatype the class of all algebras isomorphic to the quotient term algebra over its formula set.

Concrete Syntax

initial spec
  sorts *sort* ...
  opns *operation* ...
  vars *variable*: *sort* ...
  eqns *equation* ...
endspec

- Signature $\Sigma = (\{sort, \ldots\}, \{operation, \ldots\})$.
- Set of formulas $\Phi = \{(\forall variable : sort, \ldots \ . \ equation), \ldots\}$.

We will only use the concrete syntax to define specifications.

Example

initial spec
  sorts *nat*
  opns
    $0 :\rightarrow nat$
    $Succ : nat \rightarrow nat$
    $\_ + \_ : nat \times nat \rightarrow nat$
  vars $m, n : nat$
  eqns
    $n + 0 = n$
    $n + Succ(m) = Succ(n + m)$
endspec

- $T(\Sigma, \Phi)(nat) = \{[0], [Succ(0)], [Succ(Succ(0))], \ldots\}$
  - $[0] = [0 + 0] = [0 + (0 + 0)] = \ldots$
  - $[Succ(0)] = [Succ(0) + 0] = [0 + Succ(0)] = [Succ(0 + 0)] = \ldots$
  - ...

Initiality

Take signature Σ, set of formulas $\Phi \in EL(\Phi)$.

- Theorem: $T(\Sigma, \Phi) \in Mod_\Sigma(\Phi)$.
  - Quotient term algebra of Φ is itself a model of Φ.
  - Not true for formula set Φ from every logic L.
- Corollary: $T(\Sigma, \Phi)$ is initial in $Mod_\Sigma(\Phi)$.
  - Consequence of the final theorem of the previous section.
- Corollary: Every algebra of $\mathcal{M}(sp)$ is initial in $Mod_\Sigma(\Phi)$.
  - The specified algebras distinguish most among all models of Φ.

The specified abstract datatype is the most "distinguishing" one.

Datatype Properties

Take initial specification $sp = (\Sigma, \Phi)$.

- $\mathcal{M}(sp)$ is not empty.
  - It contains $T(\Sigma, \Phi)$.
  - An initial specification is *consistent*.
- $\mathcal{M}(sp)$ is monomorphic.
  - By definition, all algebras in $\mathcal{M}(sp)$ are isomorphic.
  - An initial specification describes a *single datatype*.
- $\mathcal{M}(sp)$ only contains generated algebras.
  - Every carrier is isomorphic to (the congruence class of) a ground term.
  - The specified datatype *does not contain junk*.

An initial specification is much more specific than a loose specification.

Logical Properties

Take initial specification (Σ, Φ), ground equation $t = u$ and equation $\forall X.v = w$ in $EL(\Phi)$.

- $T(\Sigma, \Phi) \models t = u$ iff $\Phi \models t = u$.
  - $\Phi \models t = u$ iff $A \models t = u$ for every Σ-algebra A that is a model of Φ.
  - The equation holds in specified datatype, iff the equation is a logical consequence of the specification equations.
- $T(\Sigma, \Phi) \models \forall X.v = w$ iff $\Phi \models_{Ind} \forall X.v = w$.
  - $\Phi \models_{Ind} t = u$ iff $A \models t = u$ for every model A of Φ that is generated.
  - The equation holds in the specified datatype, iff the equation is a logical consequence of the specification equations; for proving this, we may apply the principle of structural induction.

In the specified abstract datatype, no other equalities hold than those that are consequences of Φ.

Example

initial spec
  sorts nat, set
  opns
    $0 :\to nat$
    $Succ : nat \to nat$
    $\emptyset :\to set$
    $Insert : set \times nat \to set$
  vars $m, n : nat, s : set$
  eqns
    $Insert(Insert(s, n), n) = Insert(s, n)$
    $Insert(Insert(s, n), m) = Insert(Insert(s, m), n)$
endspec

Strictly adequate specification of "classical" set algebra (compare with a corresponding loose specification with constructors).

Example

loose spec
  sorts freely generated *nat*
  opns
    constr $0 :\to nat$
    constr $Succ : nat \to nat$
    $Pred : nat \to nat$
  vars $n : nat$
  axioms $Pred(Succ(n)) = n$
endspec

initial spec
  sorts *nat*
  opns
    $0 :\to nat$
    $Succ : nat \to nat$
    $Pred : nat \to nat$
  vars $n : nat$
  eqns $Pred(Succ(n)) = n$
endspec

- Loose specification is polymorphic:
  - Algebra A may define any value $A(Pred(0)) \in A(nat)$.
  - Same is true for the "classical" algebra of natural numbers.
- Initial specification is monomorphic:
  - $T(\Sigma, \Phi)(nat)$ has $[Pred(0)], [Pred(Pred(0))], [Succ(Pred(0))], \ldots$
  - Certainly not isomorphic to the "classical" algebra of natural numbers.

Initial specifications may yield carrier sets that are larger than expected.

Possible Solution Attempts

- Add equation $Pred(0) = 0$.
  - Ambiguity is resolved by fixing the value of $Pred(0)$.
  - Problem: specification is less abstract than possible.
- Add equation $Succ(Pred(n)) = n$.
  - Unsatisfactory: carriers $[Pred^i(0)], i \geq 1$ are not removed.
- Add constant $Error :\to nat$.
  - Additional equations:

$$Pred(0) = Error$$
$$Succ(Error) = Error$$
$$Pred(Error) = Error$$

  - Problem: additional carrier $[Error]$ has to be considered.
- Combine last two solutions:
  - $[Succ^{i+1}(Pred(0))] = [Succ^i(0)]$.
  - $[Succ^{i+1}(Pred(0))] = [Succ^{i+1}(Error)] = [Error]$.
  - Effect: $T(\Sigma, \Phi)(nat) = \{[Error]\}$.

Loose vs Initial Specifications

- Adding an additional formula to a specification.
  - Loose specification: some models are removed.
    - (Not strictly) adequate specification becomes "more adequate".
  - Initial specification: some carriers are identified.
    - Carrier set becomes smaller.
    - Adequate specification becomes inadequate.
- Adding an incompatible formula to a specification.
  - Loose specification: $\mathcal{M}(sp) = \emptyset$.
    - Specification becomes inconsistent; all models are "killed".
  - Initial spec: $\mathcal{M}(sp)$ consists of algebras with singleton carrier sets.
    - Specification becomes too constrained; some carrier sets "collapse".

In an initial specification, adding an "incompatible" equation lets a carrier set collapse to a singleton.

Expressive Power of Initial Specifications

- Theorem: Any generated algebra may be specified by an initial specification consisting of ground equations only.
  - Catch: number of equations may be infinite and even not recursively enumerable (e.g. Peano arithmetic).
- Theorem: There exist algebras that can be specified by an initial specification with a finite number of equations but cannot be specified with a finite number of ground equations.
  - Universally quantified variables really add expressive power.
- Theorem: There exist generated algebras that cannot be specified by an initial specification consisting of a finite number of equations.
  - Universally quantified variables do not suffice.

Seems to impose fundamental limitations on initial specifications.

Example

- $\Sigma = (\{nat\}, \{0 :\rightarrow nat, Succ : nat \rightarrow nat, Square : nat \rightarrow nat\})$.
  - Classical Σ-algebra A:
    $A(nat) = \mathbb{N}$, $A(0) = 0$, $A(Succ)(n) = n + 1$,
    $A(Square)(n) = n^2$.
  - Attempt to an adequate specification of A:
    $Square(0) = 0$
    $Square(Succ(n)) = \ldots ?$
- $\Sigma' = (\ldots, \{\ldots, + : nat \times nat \rightarrow nat\})$.
  - Classical Σ'-algebra A:
    $\ldots, A(+)(n, m) = n + m$
  - Adequate specification of A:
    $\ldots$
    $n + 0 = n$
    $n + Succ(m) = Succ(n + m)$
    $Square(Succ(n)) = Succ(Square(n) + (n + n))$

Additional operations may be needed for an adequate initial specification.

Expressive Power of Initial Specifications

Take finite signature Σ.

- Theorem: For every generated Σ-algebra A with computable functions, there exists a signature $\Sigma' \supseteq \Sigma$ and an initial specification (Σ', Φ) with a finite set of formulas $\Phi \subseteq EL(\Sigma')$ such that
  $$A \simeq T(\Sigma', \Phi)|\Sigma$$
  - $\mathcal{C}|\Sigma$ is the Σ-reduct of class $\mathcal{C}$.
    - From every algebra of C, all carrier sets and functions are removed that correspond to sorts and operations not mentioned in Σ.

Any computable function can be defined by recursive equations, thus any generated algebra with computable functions can be adequately specified by an initial specification (after extending the signature appropriately).

Other Specification Logics

Not in every logic there exists an initial model of a set of formulas.

- Specification (Σ, Φ):
  - $\Sigma = (\{S\}, \{a, b, c, d :\rightarrow S\})$.
  - $\Phi = \{(a \neq b \wedge c = d) \vee (a = b \wedge c \neq d)\}$.
- Σ-algebra $B \in Mod_\Sigma(\Phi)$:
  $B(a) \neq B(b), B(c) = B(d)$.
- Σ-algebra $C \in Mod_\Sigma(\Phi)$:
  $C(a) = C(b), C(c) \neq C(d)$.
- Assume that some Σ-algebra A is initial in $Mod_\Sigma(\Phi)$:
  - Since A is initial, an equation that does not hold for some algebra in $Mod_\Sigma(\Phi)$, does also not hold for A.
  - Since $B(a) \neq B(b)$, also $A(a) \neq A(b)$.
  - Since $C(c) \neq C(d)$, also $A(c) \neq A(d)$.
  - Since $A(a) \neq A(b)$ and $A(c) \neq A(d)$, $A \notin Mod_\Sigma(\Phi)$.

Not every logic is suitable for initial specifications.

Other Specification Logics

Take signature Σ.

- Theorem: If $\Phi \subseteq CEL(\Sigma)$, $T(\Sigma, \Phi) \in Mod_\Sigma(\Phi)$.

initial spec
  sorts $bool, nat, list$
    $True :\rightarrow bool$
    $False :\rightarrow bool$
    $0 :\rightarrow nat$
    $Succ : nat \rightarrow nat$
    $[\,] :\rightarrow list$
    $Add : nat \times list \rightarrow list$
    $\_.\_ : list \times list \rightarrow list$
    $Isprefix : list \times list \rightarrow bool$
  vars $l, m : list, e, e' : nat$
  cond eqns
    $[\,].l = l$
    $Add(e, l).m = Add(e, l.m)$
    $Isprefix(Add(e, l), [\,]) = False$
    $Isprefix(Add(e, l), Add(e, m)) = Isprefix(l, m)$
    $Isprefix(Add(e, l), Add(e', m)) = True \Rightarrow e = e'$
endspec

Initial specifications may also use conditional equations.

Properties and Proofs

Additional proof techniques for ADTs defined by initial specifications.

- Take initial specification $sp = (\Sigma, \Omega)$.
  - Goal: $\mathcal{M}(sp) \models \forall X.v = w$.
  - Prove: $\Omega \models v\sigma = w\sigma$ for each ground substitution $\sigma : X \to T_\Sigma$.
    - Proof by induction on the structure of the substitution.
- Example: $sp = (\Sigma, \Phi)$, $\Sigma = (\{nat\}, \{0, Succ, +\})$

  $\Phi = \{(1)\ n + 0 = n,\ (2)\ n + Succ(m) = Succ(n + m),$
  $(3)\ (n + m) + p = n + (m + p)\}$.
  - Goal: $\mathcal{M}(sp) \models 0 + n = n$.
  - Prove: $\Phi \models 0 + t = t$, for every ground term $t \in T_\Sigma$.
    - Case $t = 0$: $0 + 0 =_{(1)} 0$.
    - Case $t = Succ(t')$: $0 + Succ(t') =_{(2)} Succ(0 + t') =_{(Ind)} Succ(t')$.
    - Case $t = (t' + t'')$: $0 + (t' + t'') =_{(3)} (0 + t') + t'' =_{(Ind)} t' + t''$.

Proof by induction on the structure of substitution terms.

Properties and Proofs

Take initial specification (Σ, Φ) and Σ-algebra A.

- Goal: (Σ, Φ) adequately specifies A.
  - $A \simeq T(\Sigma, \Phi)$.

1. Prove A is a model of Φ.

   Since $T(\Sigma, \Phi)$ is initial in $Mod_\Sigma(\Phi)$, we have an evaluation homomorphism $h : T(\Sigma, \Phi) \to A$, i.e. $h([t]) := A(t)$.
2. Prove A is generated.

   Thus for every carrier a there exists a term t with $A(t) = a$ and consequently $h([t]) = a$, i.e. h is surjective.
3. Prove h is injective.

   Prove, for arbitrary $t, u \in T_\Sigma$,
   $h([t]) = h([u]) \Rightarrow [t] = [u]$, i.e.
   $A(t) = A(u) \Rightarrow [t] = [u]$, i.e.
   $A(t) = A(u) \Rightarrow T(\Sigma, \Phi)(t) = T(\Sigma, \Phi)(u)$, i.e.
   $A(t) = A(u) \Rightarrow \Phi \models t = u$.

Example

- $sp = (\Sigma, \Phi)$, $\Sigma = (\{nat\}, \{0, Succ, +\})$

  $\Phi = \{(1)\ n + 0 = n,\ (2)\ n + Succ(m) = Succ(n + m),$
  $(3)\ (n + m) + p = n + (m + p)\}$.
- Goal: sp adequately specifies classical Σ-algebra A.

1. Prove $A \models \Phi$.
2. Prove A is generated.
3. Take terms $t, u \in T_\Sigma$ with $A(t) = A(u)$ and prove $\Phi \models t = u$.

   3.1 Lemma: $A(v) = n$ implies $\Phi \models v = Succ^n(0)$, for $n \in \mathbb{N}, v \in T_\Sigma$.

   Proof by induction on the structure of v

   3.2 Take $n = A(t) = A(u)$. Then, by the lemma, $\Phi \models t = Succ^n(0)$ and $\Phi \models u = Succ^n(0)$. Thus $\Phi \models t = u$.

A canonical term representation for carriers simplifies proofs.

Characteristic Term Algebras

Take signature $\Sigma = (S, \Omega)$ and initial specification $sp = (\Sigma, \Phi)$.

- A Σ-algebra C with $C(s) \in T_{\Sigma,s}$ for every sort $s \in S$ is called a characteristic term algebra for sp, if
  - $C \models \Phi$
  - $\Phi \models C(t) = t$ for each ground term $t \in T_\Sigma$.
    - C maps every ground term to another term with the same value.
- Theorem: If C is charact. term alg. for (Σ, Φ), then $C \simeq T(\Sigma, \Phi)$.
  - C is a term algebra isomorphic to the quotient term algebra.
  - C may serve in proofs as a replacement of the quotient term algebra.

By using characteristic term algebras, proofs can be simplified.

Example

- $sp = (\Sigma, \Phi)$, $\Sigma = (\{nat\}, \{0, Succ, +\})$
  $\Phi = \{n + 0 = n, n + Succ(m) = Succ(n + m)\}$
- Σ-algebra C:
  $C(nat) = \{Succ^n(0) \mid n \in \mathbb{N}\}$
  $C(0) = 0$
  $C(Succ)(t) = Succ(t)$, for all $t \in C(nat)$
  $C(+)(t, u) = Succ^{p+q}(0)$ where $p, q \in \mathbb{N}$ such that
  $t = Succ^p(0)$, $u = Succ^q(0)$, for all $t, u \in C(nat)$.

C maps *nat*-terms to terms involving 0 and *Succ* only.

Example (Contd)

C is a characteristic term algebra for (Σ, Φ):

1. $C \models t + 0 = t$: $C(t + 0) = C(+)(C(t), C(0)) = C(+)(C(t), 0) = C(t)$.
   $C \models t + Succ(u) = Succ(t + u)$: $C(t + Succ(u)) =$
   $C(+)(C(t), C(Succ)(C(u)))) = C(+)(Succ^p(0), Succ(Succ^q(0))) =$
   $Succ^{p+(1+q)}(0) = Succ(Succ^{p+q}(0)) = Succ(C(+)(Succ^p(0), Succ^q(0))) =$
   $C(Succ)(C(+)(C(t), C(u))) = C(Succ(t + u))$.
2. $\Phi \models C(t) = t$: proof by induction on t.
   $t = 0$: $C(0) = 0$, thus $\Phi \models C(0) = 0$.
   $t = Succ(u)$:
   $C(Succ(u)) = C(Succ)(C(u)) = Succ(C(u))$. By induction hypothesis, $\Phi \models C(u) = u$, thus $\Phi \models C(Succ(u)) = Succ(u)$.
   $t = u + v$:
   $C(u + v) = C(+)(C(u), C(v)) = C(+)(Succ^p(0), Succ^q(0)) =$
   $Succ^{p+q}(0)$. By the lemma below, $\Phi \models Succ^{p+q}(0) = C(u) + C(v)$.
   By induction hypothesis, $\Phi \models C(u) = u$ and $\Phi \models C(v) = v$, thus
   $\Phi \models Succ^{p+q}(0) = u + v$ and finally $\Phi \models C(u + v) = u + v$.
   Lemma: $\phi \models Succ^{p+q}(0) = Succ^p(0) + Succ^q(0)$, for all $p, q \in \mathbb{N}$.
   Proof by induction on q.

Example (Contd'2)

Once it has been proved that C is a characteristic term algebra of *sp*, proving properties of *sp* becomes simple.

- Prove that *sp* adequately specifies classical A with $A(nat) = \mathbb{N}$.
  - Prove that $A \simeq C$.
- Find isomorphism $h : C \to A$.
  $h_{nat}(Succ^n(0)) := n$.
  - Clearly h is bijective.
  - h is a homomorphism:
    $h(C(Succ)(Succ^n(0))) = h(Succ^{n+1}(0)) = n + 1 =$
    $A(Succ)(n) = A(Succ)(h(Succ^n(0)))$.
    $h(C(+)(Succ^n(0), Succ^m(0))) = h(Succ^{n+m}(0)) = n + m =$
    $A(+)(n, m) = A(+)(h(Succ^n(0)), h(Succ^m(0)))$.

Proof can make use of the structure of characteristic terms.

Example (Contd'3)

- Prove that $\mathcal{M}(sp) \models 0 + t = t$.
- Prove that $C \models 0 + t = t$.
  $C(0 + t) = C(+)(C(0), C(t)) = C(+)(0, Succ^n(0)) = Succ^n(0) =$
  $C(t)$.

The work invested in proving that C is characteristic is well spent, since all other proofs become much simpler.

Summary

- Initial specifications have some nice properties:
  - The specified ADT is not empty, it is monomorphic, and it only consists of generated algebras.
    - The specification already describes a concrete implementation design.
  - The quotient term algebra is the canonical representative of the ADT.
    - Carriers are classes of terms whose values are considered identical.
  - Only those equalities hold that are explicitly specified.
- However, there are also potential pitfalls:
  - Adding "incompatible" equations lets carrier sets collapse.
    - Still necessary to investigate the adequacy of a specification.
  - "Undefined" terms represent additional carriers.
    - Carrier sets then contain more elements than intended.
  - Rather than undefined terms, one may prefer error values.
    - However, then one has to deal with these values in all computations.
  - Only generated algebras can be specified.
    - E.g. no initial specifications of the reals.

The necessity to be concrete had advantages and disadvantages.