Proving sufficient completeness of constructor-based specifications

NAKAMURA Masaki

Toyama Pref. Univ.

25-26 June 2015

Kaken Kiban(S) Workshop
Hotel Arrowle, Kaga, Ishikawa, Japan
Joint work with Daniel Găină, Kazuhiro Ogata and Kokichi Futatsugi (JAIST)
Abstract

- CafeOBJ algebraic specification language supports description of specifications whose models are constructor-based algebras \cite{1,2}.
- **Sufficient completeness** is one of the most important properties of constructor-based specifications, which guarantees the existence of initial models.
- In this study, we give a sufficient condition for **sufficient completeness** of constructor-based specifications based on the theory of term rewriting.

A constructor-based signature \((S, \leq, \Sigma, \Sigma^c)\) (abbr. signature \(\Sigma\)) consists of
- a set \(S\) of sorts,
- a poset \(\leq\) on \(S\),
- a \(S^+\)-sorted set \(\Sigma\) of operators, and
- a set \(\Sigma^c \subseteq \Sigma\) of constructors

We call
- \(S^{cs} = \{s \in S \mid f \in \Sigma^c_{ws}\}\) constrained sorts,
- \(S^{ls} = S \setminus S^{cs}\) loose sorts,
- \(\Sigma^{S^{cs}} = \{f \in \Sigma_{ws} \mid w \in S^*, s \in S^{cs}\}\) constrained operators,
- \(T_{\Sigma^c}(Y)\) constructor terms and
- \(T_{\Sigma^{S^{cs}}}(Y)\) constrained terms, where \(Y\) is a set of loose variables
Examples

mod! N+{  
  [Zero NzNat < Nat]
  op 0 : -> Zero {constr}
  op s_ : Nat -> NzNat {constr}
  op s_ : Nat -> Nat {constr}
  op _+_ : Nat Nat -> Nat
  eq X:Nat + 0 = X .
  eq X:Nat + s Y:Nat = s (X + Y) .
}

Σ⁴ = {0, s_},
S⁴ = {Zero, NzNat, Nat},
S⁴₀ = Ø,
ΣS⁴₀ = {0, s_+_},
0, s 0, s s 0, ... ∈ TΣ⁴(Ø),
0 + s 0, (0 + 0) + 0, ... ∈ TΣS⁴₀(Ø)

mod* LIST{  
  [Elt, List]
  op nil : -> List {constr}
  op (_;_ ) : Elt List -> List {constr}
  op rev : List List -> List
  var E : Elt vars L L' : List
  eq rev(nil, L) = L .
  eq rev(E ; L, L') = rev(L, E ; L') .
}

Σ⁴ = {nil, _;_ },
S⁴₀ = {List},
S⁴₀₀ = {Elt},
ΣS⁴₀₀ = {nil, _;_ rev},
nil, (E; nil), ... ∈ TΣ⁴₀₀({E}),
rev(nil), rev(E; rev(nil)) ... ∈
TΣS⁴₀₀({E})

NAKAMURA Masaki (Toyama Pref. Univ.)  SC of CB spec  25-26 June 2015
Let \((S, \leq, \Sigma, \Sigma^C)\) (abbr. \(\Sigma\)) a signature.

A \((S, \leq, \Sigma)\)-algebra \(M\) interprets all operators as functions on the carrier sets, i.e. \(M_f \in M_{s_0} \times \cdots \times M_{s_n} \to M_s\) for each \(f \in \Sigma_{s_0s_1\ldots s}\).

A (constructor-based) \((S, \leq, \Sigma, \Sigma^C)\)-algebra \(M\) is an \((S, \leq, \Sigma)\)-algebra where the carrier sets for the constrained sorts consist of interpretations of constructor terms.

A (constructor-based) specification \(SP = ((S, \leq, \Sigma, \Sigma^C), E)\) is a pair of a signature and a set of equations on \((S, \leq, \Sigma)\).

A model of \(SP\) is a \((S, \leq, \Sigma, \Sigma^C)\)-algebra satisfying all equations in \(E\).
Sufficient completeness for constructor-based specifications is important since it guarantees the existence of initial models.

**Definition** [1]: A constructor-based specification \(((S, \leq, \Sigma, \Sigma^C), E)\) is sufficient complete if for all \(M \in |\mathbb{Mod}((S, \leq, \Sigma, \Sigma^{Scs}), E)|\) we have \(M \in |\mathbb{Mod}(S, \leq, \Sigma, \Sigma^C)|\)

**Proposition** [1]: If a constructor-based specification is sufficiently complete, it has an initial model.
The following sufficient condition of sufficient completeness is given in [2]:

**Proposition** [2]: A specification $SP = ((S, \leq, \Sigma, \Sigma^C), E)$ is sufficiently complete if for each constrained term $t \in T_{\Sigma^{scs}}(Y)$, there exists a constructor term $u \in T_{\Sigma^c}(Y)$ such that $t =_E u$.

**Example**: $N+$ and $LIST$ are sufficiently complete:

```plaintext
N+> red 0 + (s 0 + s s 0).
(s (s (s 0))):NzNat

%LIST> vars A B C : Elt .
%LIST> red rev(A ; B ; C ; nil, nil) .
(C ; (B ; (A ; nil))):List
```
Proof sketch of the proposition

**Proposition [2]:** A specification \( SP = ((S, \leq, \Sigma, \Sigma^C), E) \) is sufficiently complete if for each constrained term \( t \in T_{\Sigma^{scs}}(Y) \), there exists a constructor term \( u \in T_{\Sigma^C}(Y) \) such that \( t =_E u \).

- Let \( m \in M_s \) for \( M \in \text{Mod}((S, \leq, \Sigma, \Sigma^{S^{cs}}), E) \) and \( s \in S^{cs} \). It suffices to show that there exists a term \( t^c \in T_{\Sigma^C}(Y) \) and an assignment \( f : Y \rightarrow M \), where \( Y \) is a finite set of loose variables, such that \( f^#(t^c) = m \), where \( f^# : T_{\Sigma^{cs}}(Y) \rightarrow M \) is the unique ext. of \( f \) to a \( \Sigma^{S^{cs}} \)-homomorphism.

- From Def., for \( M \in \text{Mod}((S, \leq, \Sigma, \Sigma^{S^{cs}}), E) \), we have \( m = \overline{f}(t) \) for some \( t \in T_{\Sigma^{S^{cs}}}(Y) \) and \( f : Y \rightarrow M \) where \( \overline{f} \) is unique ext. of \( f \) to a \( \Sigma^{S^{cs}} \)-hom. By our hypothesis \( M \models (\forall Y)t = t^c \) for some constructor term \( t^c \in T_{\Sigma^C}(Y) \). It follows that \( m = \overline{f}(t) = \overline{f}(t^c) \). Now let \( f^# : T_{\Sigma^C}(Y) \rightarrow M \) be the unique ext. of \( f \) to a \( \Sigma^C \)-hom. We have \( \overline{f}(t^c) = f^#(t^c) \). Hence \( m = f^#(t^c) \).
Example: a *map* function on lists

**Proposition [2]:** A specification $SP = ((S, \leq, \Sigma, \Sigma^C), E)$ is sufficiently complete if for each constrained term $t \in T_{\Sigma^S^C}(Y)$, there exists a constructor term $u \in T_{\Sigma^C}(Y)$ such that $t =_E u$.

- The above proposition is not complete since we have the following example of a *map* function on lists, where $\Sigma^C = \{ \text{nil}, _; \}$, $S^S^C = \{ \text{List} \}$, $\Sigma^S^S^C = \{ \text{nil}, _; _\text{map} \}$.

- Even if $t$ is a constructor term, $\text{map}(t)$ may be reduced into $t'$ which includes $f \not\in \Sigma^C$.

mod! MAP{
    [ Elt, List ]
    op nil : \rightarrow List \{ constr \} %MAP> vars A B C : Elt .
    op _ ; _ : Elt List \rightarrow List \{ constr \} %MAP> red map(A ; B ; C ; nil) .
    op map : List \rightarrow List
    op f : Elt \rightarrow Elt
    var E : Elt var L : List
    eq map(nil) = nil .
    eq map(E ; L) = f(E) ; map(L) .
}
We weaken the condition in the above proposition as follows:

**Theorem:** A specification $SP = ((S, \leq, \Sigma, \Sigma^C), E)$ is sufficient complete if for each constrained term $t \in T_{\Sigma^C}(Y)$, there exists a term $u \in T_{\Sigma^C \cup \Sigma^{S_{ls}}}(Y)$ such that $t =_E u$, where $\Sigma^{S_{ls}}$ is the set of all operators $f$ of loose sorts, i.e. $f : w \rightarrow s$ and $s \in S^{ls}$

**Example:** In MAP, $S^{ls} = \{Elt\}$ and $f \in \Sigma^{S_{ls}}$

```plaintext
mod! MAP{
  [ Elt, List ]
  op nil : -> List {constr}
  op _ ; _ : Elt List -> List {constr}
  op map : List -> List
  op f : Elt -> Elt
  var E : Elt var L : List
eq map(nil) = nil .
eq map(E ; L) = f(E) ; map(L) .
}
```

%MAP> vars A B C : Elt .
%MAP> red map(A ; B ; C ; nil) .
(f(A) ; (f(B) ; (f(C) ; nil))) : List
Proof sketch of the theorem

**Theorem:** A specification $SP = ((S, \leq, \Sigma, \Sigma^C), E)$ is sufficient complete if for each constrained term $t \in T_{\Sigma^{cs}}(Y)$, there exists a term $u \in T_{\Sigma^C \cup \Sigma^{sls}}(Y)$ such that $t = E u$, where $\Sigma^{sls}$ is the set of all operators $f$ of loose sorts, i.e. $f : w \rightarrow s$ and $s \in S^{ls}$.

- Let $M \in \text{Mod}((S, \leq, \Sigma, \Sigma^{cs}), E)$ and $s \in S^{cs}$. Since we changed $u \in T_{\Sigma^C}(Y)$ to $u \in T_{\Sigma^C \cup \Sigma^{sls}}(Y)$, in the similar way to the proof of the proposition, we have $\forall m \in M_s. \exists t^c \in T_{\Sigma^C \cup \Sigma^{sls}}(Y). \exists f : Y \rightarrow M. f^\#(t^c) = m$. We need to prove the existence of $t^{c'} \in T_{\Sigma^C}(Y')$ and $f' : Y' \rightarrow M$ s. t. $f'^\#(t^{c'}) = m$.

- Make a constructor term $t^{c'}$ by replacing all subterms $u_i$ whose root is $\Sigma^{sls}$ in $t^c$ with fresh variables $y_i$.
  
  e.g. $f(A); f(B); f(C); \text{nil}$ into $y_0; y_1; y_2; \text{nil}$.

  Let $Y'$ be $Y \cup \{y_0, y_1, \ldots\}$. Define $f' : Y' \rightarrow M$ as $f'(y) = y$ ($y \in Y$) and $f'(y_i) = f^\#(u_i)$ ($y_i \in Y' \setminus Y$). Then, we have $f'^\#(t^{c'}) = f^\#(t^c) = m$. 

NAKAMURA Masaki (Toyama Pref. Univ.)

SC of CB spec

25-26 June 2015
Sufficient completeness w.r.t. rewriting

- Term rewriting \((\rightarrow_E)\) is useful for equational reasoning \((\equiv_E)\). We give sufficient completeness w.r.t. rewriting by replacing \(\equiv_E\) with \(\rightarrow^*_E\).

**Definition:** A specification \(SP = ((S, \leq, \Sigma, \Sigma^C), E)\) is sufficient complete w.r.t. rewriting (SCR) if for each constrained term \(t \in T_{\Sigma^{scs}}(Y)\), there exists a term \(u \in T_{\Sigma^C \cup \Sigma^{sls}}(Y)\) such that \(t \rightarrow^*_E u\).

**Corollary:** (1) SCR implies SC (2) For a terminating SCR specification, we can compute such \(u\) for each \(t\).

**Example:** \(MAP\) satisfies SCR (and is terminating). We can reduce any \(t\) of \(List\) into \(u\) constructed from constructors, \(f\) and loose vars.

```mod! MAP{
    [ Elt, List ]
    op nil : -> List {constr}
    op _ ; _ : Elt List -> List {constr}
    op map : List -> List
    op f : Elt -> Elt
    var E : Elt var L : List
    eq map(nil) = nil .
    eq map(E ; L) = f(E) ; map(L) .
}
```

%MAP> vars A B C : Elt .
%MAP> red map(A ; B ; C ; nil) . (f(A) ; (f(B) ; (f(C) ; nil))):List

NAKAMURA Masaki (Toyama Pref. Univ.)
Basic terms for proving SCR

- **Definition:** For $\Sigma' \subseteq \overline{\Sigma} \cap \Sigma$, $f(\overline{t})$ is $\Sigma'$-basic if $f \in \overline{\Sigma} \cup \Sigma'$ and $\overline{t}$ are terms constructed from $\Sigma \cup \Sigma'$ and loose variables $Y$, i.e. $\overline{t} \in T_{\Sigma \cup \Sigma'}(Y)$.

- **Theorem:** If a specification is terminating and all $\Sigma^{S_{ls}}$-basic terms are $E_{SP}$-reducible, it satisfies SCR.

- **Example:** Consider MAP. $\Sigma^C = \{\text{nil}, \_, \_\}$ and $\Sigma^{S_{ls}} = \{f\}$. Then, $\overline{\Sigma^C \cup \Sigma^{S_{ls}}} = \{\text{map}\}$, and $\text{map}(\text{nil})$, $\text{map}(f(X); \text{nil})$ are $\Sigma^{S_{ls}}$-basic. $f(X)$ is not so.

```plaintext
mod! MAP{  
  [ Elt, List ]
  op nil : -> List {constr}
  op _ ; _ : Elt List -> List {constr}
  op map : List -> List
  op f : Elt -> Elt
  var E : Elt var L : List
  eq map(nil) = nil .
  eq map(E ; L) = f(E) ; map(L) .
}
```
CafeOBJ supports a parameterized specification where its parameter can be instantiated by a view:

```
mod* FUN{
  [Elt]
  op f : Elt -> Elt
}
mod* MAP(Z :: FUN){
  [List]
  op nil : -> List {constr}
  op (_;_): Elt List -> List {constr}
  op map : List -> List
  var E : Elt var L : List
  eq map(nil) = nil .
  eq map(E ; L) = f(E) ; map(L) .
}
```

```
mod! N+{[Zero NzNat < Nat]
  op 0 : -> Zero
  op s_ : Nat -> NzNat
  op _+_ : Nat Nat -> Nat
  eq X:Nat + 0 = X .
  eq X:Nat + s Y:Nat = s (X + Y) .
}
```

```
view FN from FUN to N+ {
  sort Elt -> Nat,
  op f(E:Elt) -> (E:Nat + E)
}
```

```
CafeOBJ> red in MAP(FN) : map( 0 ; s 0 ; s s 0 ; nil) .
-- reduce in MAP(Z <= FN) : (map((0 ; ((s 0) ; ((s (s 0)) ; nil))))):List
(0 ; ((s (s 0)) ; ((s (s (s (s 0)))) ; nil))):List
```
In [1], it is shown that all instantiations of sufficiently complete parameterized specifications are also sufficiently complete. It is also true for SCR under some conditions.

Assume all non-constrained operators are declared in a parameter specification of a given parameterized specification ($\Sigma^{S_is} \subseteq \Sigma_P$)

**Definition:** A PS $i : P \rightarrow SP$ is constructor-preserving if there is no $f \in (\Sigma^C_{SP})_{ws} \setminus \Sigma_P$ such that $s \not\in S_P$

**Definition:** A PS $i : P \rightarrow SP$ is left-$P$-free if no $f \in \Sigma_P$ are included in the left-hand sides of all equations in $E'_{SP}$

**Theorem:** Let $i : P \rightarrow SP$ be left-$P$-free and constructor-preserving parameterized specification where $\Sigma^{S_is} \subseteq \Sigma_P$ and $v : P \rightarrow P'$ a view. If both $i : P \rightarrow SP$ and $P'$ are SCR, then so is $SP(v)$
We gave a sufficient condition for sufficient completeness of constructor-based specifications and an application to parameterized specifications.

Future work: operator attributes, e.g. associative, commutative, idempotent, etc.