MULTIPLE PARAMETERS AND THEIR INSTANTIATION

Ionuț Țuțu

Școala Normală Superioară București

Sinaia, 2011

Multiple Parameters and their Instantiation

R. Diaconescu, I. Țuțu:

On the Algebra of the Structured Specifications

Introduction

- parameterization arises naturally
  - in many cases datatypes are generic
  - the datatype of lists is generic regarding the elements
- *parameterization*
  - mechanism that improves the reusability of specifications
  - the genericity of a specification can be presented explicitly by declaring parameters
  - parameterized specifications can be instantiated by providing fitting argument specifications
- the goal of this work: discuss how to define and instantiate parameterized specifications, with examples from many sorted algebra (**MSA**)

Towards a generic specification of lists

Lists of natural numbers

$$\begin{array}{llllll}
\textbf{free} & \textbf{sorts} & \mathsf{Nat} & & & \\
 & & \mathsf{List} & & & \\
 & \textbf{ops} & 0 & : & & \rightarrow \mathsf{Nat} \\
 & & \mathsf{s}\_ & : & \mathsf{Nat} & \rightarrow \mathsf{Nat} \\
 & & [] & : & & \rightarrow \mathsf{List} \\
 & & \_\_ & : & \mathsf{Nat}\,\mathsf{List} & \rightarrow \mathsf{List} \\
 & & \mathsf{length} & : & \mathsf{List} & \rightarrow \mathsf{Nat} \\
 & \textbf{axioms} & & & & \mathsf{length}([]) = 0 \\
 & & (\forall N : \mathsf{Nat}, L : \mathsf{List}) & & & \mathsf{length}(N\,L) = \mathsf{s}\,\mathsf{length}(L)
\end{array}$$

Towards a generic specification of lists

Lists of natural numbers

in CafeOBJ

```
mod! PNAT {
  [ Nat ]
  op 0  :     -> Nat
  op s_ : Nat -> Nat
}

mod! LIST-NAT {
  pr(PNAT)
  [ List ]
  op [] :          -> List
  op __ : Nat List -> List

  op length : List -> Nat
  var N : Nat var L : List
  eq length([]) = 0 .
  eq length(N L) = s length(L) .
}
```

in CASL

```
library LIST_NAT

spec PNAT =
     free type Nat ::= 0 | s__(Nat)
end

spec LIST_NAT =
     PNAT
then free
     {type  List ::= [] | __ __(Nat; List)
     op     length : List → Nat
     ∀ N : Nat; L : List
     • length([]) = 0
     • length(N L) = s length(L)
     }
end
```

Towards a generic specification of lists

Lists of arbitrary elements

in CafeOBJ

```
mod! PNAT {
  [ Nat ]
  op 0  :       -> Nat
  op s_ : Nat -> Nat
}

mod* ELT {
  [ Elt ]
}

mod! LIST-ELT {
  pr(ELT + PNAT)
  [ List ]
  op [] :            -> List
  op __ : Elt List -> List

  op length : List -> Nat
  var E : Elt var L : List
  eq length([]) = 0 .
  eq length(E L) = s length(L) .
}
```

in CASL

```
library LIST_ELT

spec PNAT =
     free type Nat ::= 0 | s__(Nat)
end

spec ELT =
     sort  Elt
end

spec LIST_NAT =
     ELT
and  PNAT
then free
     {type  List ::= [] | ____(Elt; List)
      op    length : List → Nat
      ∀ E : Elt; L : List
      • length([]) = 0
      • length(E L) = s length(L)
     }
end
```

Towards a generic specification of lists

Parameterized lists

in CafeOBJ

```
mod* ELT {
  [ Elt ]
}

mod! LIST(E :: ELT) {
  pr(PNAT)
  [ List ]
  op [] :              -> List
  op __ : Elt List -> List

  op length : List -> Nat
  var E : Elt var L : List
  eq length([]) = 0 .
  eq length(E L) = s length(L) .
}

make LIST-NAT(
  LIST(view to PNAT {
         sort Elt -> Nat
       }))
```

in CASL

```
spec ELT =
     sort  Elt
end

spec LIST[ELT] given PNAT =
     free
     {type  List[Elt] ::= []  |
                        __ __(Elt; List[Elt])
     op     length : List[Elt] → Nat
     ∀ E : Elt; L : List[Elt]
     • length([]) = 0
     • length(E L) = s length(L)
     }
end

spec LIST_NAT =
     LIST[PNAT fit Elt ↦ Nat]
end
```

Foundations of parameterized specifications

From specifications to signatures

$$Sig : \text{Specifications} \rightarrow \text{Signatures}$$

- parameterized specifications and their instantiation depend heavily on the properties of both signatures and *Sig*

Sig(*PNAT*)

sorts Nat
ops 0 : → Nat
s_ : Nat → Nat

Sig(*ELT*)

sorts Elt

Sig(*LIST_NAT*)

sorts Nat
List
ops 0 : → Nat
s_ : Nat → Nat
[] : → List
__ : Nat List → List
length : List → Nat

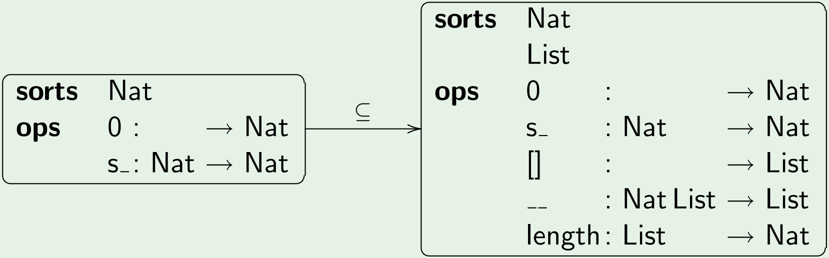
Foundations of parameterized specifications

Inclusions of **MSA** signatures

Definition (Inclusion of signatures)

An inclusion of **MSA** signatures is a morphism of signatures with all the components set theoretic inclusions.

Example



sorts Nat
ops 0 : → Nat
s_ : Nat → Nat

⊆

sorts Nat
List
ops 0 : → Nat
s_ : Nat → Nat
[] : → List
__ : Nat List → List
length : List → Nat

Foundations of parameterized specifications

Inclusions of **MSA** signatures - properties

Proposition

Every signature morphism $\varphi\colon (S,F) \to (S',F')$ *can be factored uniquely as*

$$\begin{tikzcd} (S,F) \arrow[r, "e_\varphi"] & \varphi(S,F) \arrow[r, "i_\varphi"] & (S',F') \end{tikzcd}$$

with i_φ *an inclusion and* e_φ *an abstract surjection*

(for any symbol s' *from* $\mathrm{cod}(e_\varphi)$ *there exists* s *in* $\mathrm{dom}(e_\varphi)$ *such that* $e_\varphi(s) = s'$*).*

Example

$$\begin{tikzcd} \boxed{\textbf{sorts}\quad \mathrm{Elt}} \arrow[r, "\mathrm{Elt} \mapsto \mathrm{Nat}"] \arrow[rr, bend right, "\mathrm{Elt} \mapsto \mathrm{Nat}"'] & \boxed{\textbf{sorts}\quad \mathrm{Nat}} \arrow[r, "\subseteq"] & \boxed{\begin{array}{lll} \textbf{sorts} & \mathrm{Nat} & \\ \textbf{ops} & 0 : & \to \mathrm{Nat} \\ & \mathrm{s\_} : \mathrm{Nat} & \to \mathrm{Nat} \end{array}} \end{tikzcd}$$

Foundations of parameterized specifications

Inclusions of **MSA** signatures - operations

Definition (Union of **MSA** signatures)

$$\begin{tikzcd}(S_1, F_1) \arrow[r, "\subseteq"] & (S_1, F_1) \cup (S_2, F_2) & (S_2, F_2) \arrow[l, "\subseteq"']\end{tikzcd}$$

$(S_1, F_1) \cup (S_2, F_2) = (S, F)$, where $S = S_1 \cup S_2$ and

$$F_{w \to s} = \bigcup_{\substack{i \in \{1,2\} \\ w \in S_i^*,\, s \in S_i}} (F_i)_{w \to s}.$$

Definition (Intersection of **MSA** signatures)

$$\begin{tikzcd}(S_1, F_1) & (S_1, F_1) \cap (S_2, F_2) \arrow[l, "\supseteq"'] \arrow[r, "\supseteq"] & (S_2, F_2)\end{tikzcd}$$

$(S_1, F_1) \cap (S_2, F_2) = (S, F)$, where $S = S_1 \cap S_2$ and

$$F_{w \to s} = (F_1)_{w \to s} \cap (F_2)_{w \to s}.$$

Parameterized specifications

On the semantics of specifications

For each specification SP we consider

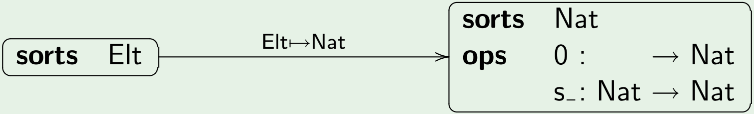
$$\begin{aligned} \mathit{Sig}(\mathrm{SP}) &: \text{ the signature of } \mathrm{SP}\text{, and} \\ \mathit{Mod}(\mathrm{SP}) &: \text{ the class of models (algebras) of } \mathrm{SP}. \end{aligned}$$

Definition (Morphism of specifications)

A *morphism of specifications* $v\colon \mathrm{SP} \to \mathrm{SP}'$ is a morphism of signatures $v\colon \mathit{Sig}(\mathrm{SP}) \to \mathit{Sig}(\mathrm{SP}')$ such that for any algebra $M' \in \mathit{Mod}(\mathrm{SP}')$ we have $M' \upharpoonright_{\mathit{Sig}(\mathrm{SP})} \in \mathit{Mod}(\mathrm{SP})$.

Example

$$\begin{tikzcd} \mathit{ELT} \arrow[r, "\textbf{view from } \mathit{ELT} \textbf{ to } \mathit{PNAT}\{\ \mathrm{Elt}\mapsto\mathrm{Nat}\ \}"] & \mathit{PNAT} \end{tikzcd}$$



sorts Elt

Elt↦Nat

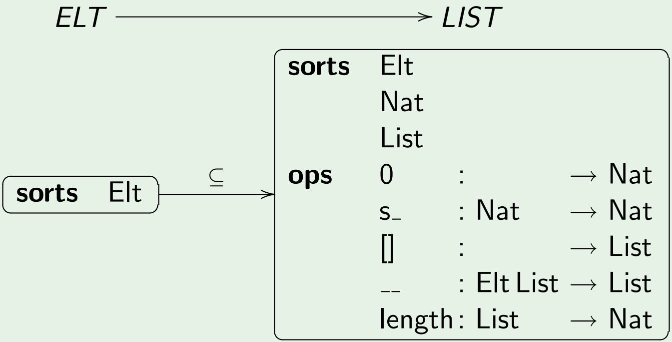
sorts Nat
ops 0 : → Nat
s_ : Nat → Nat

Parameterized specifications

Definition (Parameterized specification)

A *parameterized specification*, denoted $\mathrm{SP}(P)$, consists in a specification morphism $P \to \mathrm{SP}$ such that its underlying signature morphism is an inclusion $Sig[P] \subseteq Sig[\mathrm{SP}]$.

Example



$$\begin{tikzcd} ELT \arrow[r] & LIST \end{tikzcd}$$

sorts Elt

$\subseteq$

sorts	Elt			
	Nat			
	List			
ops	0	:		→ Nat
	s_	:	Nat	→ Nat
	[]	:		→ List
	__	:	Elt List	→ List
	length	:	List	→ Nat

Parameterized specifications

in CafeOBJ

$\mathrm{SP}(P)$ corresponds to $\mathrm{SP}(p :: P_0)$

- $p \colon P_0 \to P$ is an isomorphism
- $Sig(P_0)$ and $Sig(P)$ are disjoint
- we need to make sure there is no sharing between the parameter and other parts of the specification

Example (E :: ELT)

$$\begin{tikzcd}
ELT \arrow[r, "\mathrm{Elt} \mapsto \mathrm{Elt.E}"] & ELT \star E \\
\boxed{\textbf{sorts}\quad \mathrm{Elt}} \arrow[r, "\mathrm{Elt} \mapsto \mathrm{Elt.E}"] & \boxed{\textbf{sorts}\quad \mathrm{Elt.E}}
\end{tikzcd}$$

Parameterized specifications

Instantiation of parameters

Definition (Instantiation of parameters)

- consider a parameterized specification $\mathrm{SP}(P)$ and
- a specification morphism $v\colon P \to P'$ such that $Sig(P)$ and $Sig(P')$ are disjoint

The instantiation of the parameterized specification $\mathrm{SP}(P)$ *by* v is

$$\mathrm{SP}(P \Leftarrow v) = \mathrm{SP} \star v' + P' \star (Sig(P') \subseteq \Sigma')$$

given by the *pushout* of signatures depicted below.

$$\begin{tikzcd}
Sig(P) \cup (Sig(SP) \cap Sig(P')) \arrow[r, "\subseteq"] \arrow[d, "v+\mathrm{id}"'] \arrow[dr, phantom, "PO"] & Sig(SP) \arrow[d, "v'"] \\
Sig(P') \arrow[r, "\subseteq"'] & \Sigma'
\end{tikzcd}$$

Parameterized specifications

Instantiation of parameterized lists

Example ($LIST(ELT \star E \Leftarrow \textbf{view from } ELT \star E \textbf{ to } PNAT\{\ \mathrm{Elt.E} \mapsto \mathrm{Nat}\ \}))$

$$\begin{tikzcd}
\begin{array}{llll} \textbf{sorts} & \mathrm{Elt.E} & & \\ & \mathrm{Nat} & & \\ \textbf{ops} & 0 : & & \to \mathrm{Nat} \\ & s\_ : & \mathrm{Nat} & \to \mathrm{Nat} \end{array} \arrow[r, "\subseteq"] \arrow[d, "\begin{array}{c} \mathrm{Elt.E} \mapsto \mathrm{Nat} \\ \mathrm{Nat} \mapsto \mathrm{Nat} \\ 0 \mapsto 0 \\ s\_ \mapsto s\_ \end{array}"'] \arrow[dr, phantom, "PO"] & \begin{array}{lllll} \textbf{sorts} & \mathrm{Elt.E} & & & \\ & \mathrm{Nat} & & & \\ & \mathrm{List} & & & \\ \textbf{ops} & 0 & : & & \to \mathrm{Nat} \\ & s\_ & : & \mathrm{Nat} & \to \mathrm{Nat} \\ & [] & : & & \to \mathrm{List} \\ & \_\_ & : & \mathrm{Elt.E}\ \mathrm{List} & \to \mathrm{List} \\ & \mathrm{length} & : & \mathrm{List} & \to \mathrm{Nat} \end{array} \arrow[d, "\begin{array}{c} \mathrm{Elt.E} \mapsto \mathrm{Nat} \\ \mathrm{Nat} \mapsto \mathrm{Nat} \\ \mathrm{List} \mapsto \mathrm{List} \\ \dots \end{array}"] \\
\begin{array}{llll} \textbf{sorts} & \mathrm{Nat} & & \\ \textbf{ops} & 0 : & & \to \mathrm{Nat} \\ & s\_ : & \mathrm{Nat} & \to \mathrm{Nat} \end{array} \arrow[r, "\subseteq"] & \begin{array}{lllll} \textbf{sorts} & \mathrm{Nat} & & & \\ & \mathrm{List} & & & \\ \textbf{ops} & 0 & : & & \to \mathrm{Nat} \\ & s\_ & : & \mathrm{Nat} & \to \mathrm{Nat} \\ & [] & : & & \to \mathrm{List} \\ & \_\_ & : & \mathrm{Nat}\ \mathrm{List} & \to \mathrm{List} \\ & \mathrm{length} & : & \mathrm{List} & \to \mathrm{Nat} \end{array}
\end{tikzcd}$$

Parameterized specifications

Multiple parameters

- a parameterized specification may have several parameters

in CafeOBJ

```
mod* ELT {
  [ Elt ]
}

mod! PAIR(E1 :: ELT, E2 :: ELT) {
  [ Pair ]
  op pair : Elt.E1 Elt.E2 -> Pair

  op fst : Pair -> Elt.E1
  op snd : Pair -> Elt.E2
  var F : Elt.E1 var S : Elt.E2
  eq fst(pair(F, S)) = F .
  eq snd(pair(F, S)) = S .
}
```

in CASL

```
library PAIR

spec PAIR[sort Elt1][sort Elt2] =
     free type
          Pair[Elt1,Elt2] ::=
               pair(fst : Elt1; snd : Elt2)
end
```

Parameterized specifications

Multiple parameters

Definition (Multiple parameterized specification)

A *multiple parameterized specification* is a specification $\mathrm{SP}(P_1, \ldots, P_n)$ with more than one parameter such that the signatures of any two distinct parameters are disjoint.

Proposition

For any multiple parameterized specification $\mathrm{SP}(P_1, \ldots, P_n)$*, the specification* $\mathrm{SP}(P_1 + \cdots + P_n)$ *is a single parameterized specification.*

Parameterized specifications

Multiple parameters

Lemma

For any set of specifications $\{P_1, \ldots, P_n\}$ *whose signatures are pairwise disjoint, and any morphisms* $\{v_i \colon P_i \to P'_i \mid 1 \leq i \leq n\}$ *such that* $Sig(P_i)$ *and* $Sig(P'_j)$ *are disjoint, for* $1 \leq i, j \leq n$

- *the signatures of* $P_1 + \cdots + P_n$ *and* $P'_1 + \cdots + P'_n$ *are disjoint,*
- *there exists an unique morphism*

  $$v_1 + \cdots + v_n \colon P_1 + \cdots + P_n \to P'_1 + \cdots + P'_n$$

  making the diagram below commutative.

$$\begin{tikzcd}
P_i \arrow[r, "\subseteq"] \arrow[d, "v_i"'] & P_1 + \cdots + P_n \arrow[d, "v_1 + \cdots + v_n"] \\
P'_i \arrow[r, "\subseteq"'] & P'_1 + \cdots + P'_n
\end{tikzcd}$$

Parameterized specifications

Simultaneous instantiation of parameters

Definition (Simultaneous instantiation of parameters)

- consider a specification $\mathrm{SP}(P_1, \ldots, P_n)$ and
- a set of morphisms $\{v_i \colon P_i \to P'_i \mid 1 \leq i \leq n\}$ such that $Sig(P_i)$ and $Sig(P'_j)$ are disjoint, for $1 \leq i, j \leq n$

The *simultaneous instantiation* of

$$\mathrm{SP}(P_1, \ldots, P_n) \text{ by } \{v_i \colon P_i \to P'_i \mid 1 \leq i \leq n\}$$

is defined as the single parameter instantiation of

$$\mathrm{SP}(P_1 + \cdots + P_n) \text{ by } v_1 + \cdots + v_n.$$

Parameterized specifications

Simultaneous instantiation of parameters

Example ($PAIR\_PNAT\_BOOL$)

$$\begin{array}{l} PAIR(ELT\star E1 \Leftarrow \ \textbf{view from } ELT\star E1 \textbf{ to } PNAT\{\ \mathrm{Elt.E1} \mapsto Nat\ \}, \\ \qquad\quad ELT\star E2 \Leftarrow \ \textbf{view from } ELT\star E2 \textbf{ to } BOOL\{\ \mathrm{Elt.E2} \mapsto Bool\ \}) \end{array}$$

$$\begin{array}{lllll} \text{where } PNAT = \textbf{free} & \textbf{sorts} & \mathrm{Nat} & & \\ & \textbf{ops} & 0: & \to & \mathrm{Nat} \\ & & \mathrm{s\_}: \mathrm{Nat} & \to & \mathrm{Nat} \\ \text{and } BOOL = \textbf{free} & \textbf{sorts} & \mathrm{Bool} & & \\ & \textbf{ops} & \mathrm{true}: & \to & \mathrm{Bool} \\ & & \mathrm{false}: & \to & \mathrm{Bool} \end{array}$$

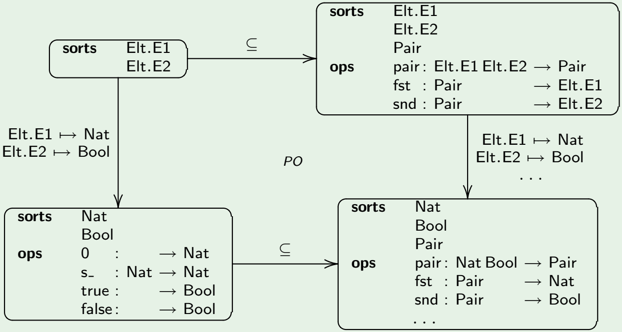
$$\begin{array}{ll} PAIR(ELT\star E1 + ELT\star E2 \Leftarrow & \textbf{view from } ELT\star E1 + ELT\star E2 \\ & \textbf{to } PNAT + BOOL \\ & \{\ \mathrm{Elt.E1} \mapsto \mathrm{Nat},\ \mathrm{Elt.E2} \mapsto \mathrm{Bool}\ \}) \end{array}$$

Parameterized specifications

Simultaneous instantiation of parameters

Example (*PAIR_PNAT_BOOL*)

$$
\begin{aligned}
PAIR(ELT \star E1 + ELT \star E2 \Leftarrow\ & \textbf{view from } ELT \star E1 + ELT \star E2\\
& \textbf{to } PNAT + BOOL\\
& \{\ \mathsf{Elt.E1} \mapsto \mathsf{Nat},\ \ \mathsf{Elt.E2} \mapsto \mathsf{Bool}\ \})
\end{aligned}
$$



sorts Elt.E1
Elt.E2

⊆

sorts Elt.E1
Elt.E2
Pair
ops pair : Elt.E1 Elt.E2 → Pair
fst : Pair → Elt.E1
snd : Pair → Elt.E2

Elt.E1 ↦ Nat
Elt.E2 ↦ Bool

PO

Elt.E1 ↦ Nat
Elt.E2 ↦ Bool
. . .

sorts Nat
Bool
ops 0 : → Nat
s_ : Nat → Nat
true : → Bool
false : → Bool

⊆

sorts Nat
Bool
Pair
ops pair : Nat Bool → Pair
fst : Pair → Nat
snd : Pair → Bool
. . .

Parameterized specifications

Sequential instantiation of parameters

- there are other ways to obtain the result $PAIR\_PNAT\_BOOL$
- instantiate the parameters $ELT \star E1$ and $ELT \star E2$ one by one:
  - instantiate $ELT \star E1$ by **view from** $ELT \star E1$ **to** $PNAT\{$ Elt.E1 $\mapsto$ Nat $\}$ to obtain a parameterized specification $PAIR(ELT \star E1 \Leftarrow \ldots)(ELT \star E2)$
  - continue with the instantiation of $ELT \star E2$ by **view from** $ELT \star E2$ **to** $BOOL\{$ Elt.E2 $\mapsto$ Bool $\}$ to obtain the final result

$$PAIR(ELT \star E1 \Leftarrow \textbf{ view from } ELT \star E1 \textbf{ to } PNAT\{ \text{ Elt.E1} \mapsto Nat \ \})$$
$$(ELT \star E2 \Leftarrow \textbf{ view from } ELT \star E2 \textbf{ to } BOOL\{ \text{ Elt.E2} \mapsto Bool \ \})$$

On the sequential instantiation of parameters

Instantiation of parameters - equivalent definition

- consider the single instantiation $\mathrm{SP}(P \Leftarrow v)$

$$\begin{tikzcd}
Sig(P) \cup (Sig(SP) \cap Sig(P')) \arrow[r, "\subseteq"] \arrow[d, "\subseteq"'] & Sig(SP) \arrow[d, "\subseteq"] \\
Sig(P) \cup Sig(P') \arrow[r, "\subseteq"] \arrow[d, "v+1_{Sig(P')}"'] \arrow[dr, phantom, "PO"] & Sig(SP) \cup Sig(P') \arrow[d, "\nu'"] \\
Sig(P') \arrow[r, "\subseteq"'] & \Sigma'
\end{tikzcd}$$

Proposition

The outer square is a pushout square if and only if the lower square is a pushout square.

On the sequential instantiation of parameters

Free extensions along inclusions

Definition (Preservation of objects)

A signature morphism $\varphi\colon \Sigma \to \Sigma_1$ *preserves* an object Σ_0 when $(\Sigma \cap \Sigma_0 \subseteq \Sigma); \varphi$ is an inclusion.

Definition (Free extension)

Let $\varphi\colon \Sigma \to \Sigma_1$ be a signature morphism and $\Sigma \subseteq \Sigma'$. A *free extension* of φ along $\Sigma \subseteq \Sigma'$ is a signature morphism $\varphi'\colon \Sigma' \to \Sigma'_1$ such that the square below is a pushout square and every signature preserved by φ is also preserved by φ'.

$$\begin{tikzcd}
\Sigma \arrow[r, "\subseteq"] \arrow[d, "\varphi"'] \arrow[dr, phantom, "PO"] & \Sigma' \arrow[d, "\varphi'"] \\
\Sigma_1 \arrow[r, "\subseteq"'] & \Sigma'_1
\end{tikzcd}$$

On the sequential instantiation of parameters

Free extensions along inclusions

Example (Free extensions of functions)

A function $f\colon A \to A_1$ admits free extensions along $A \subseteq A'$ if and only if A_1 and $A' \setminus A$ are disjoint. The free extension $f'\colon A' \to A'_1$ is defined by $A'_1 = A_1 \cup (A' \setminus A)$ and

$$f'(a) = \begin{cases} f(a) & a \in A, \\ a & a \notin A. \end{cases}$$

Proposition (Free extensions of **MSA** signature endo-morphisms)

Every **MSA** *signature morphism* $\varphi\colon (S, F) \to (S, F)$ *has free extensions* φ' *along any inclusion of signatures* $(S, F) \subseteq (S', F')$. *Moreover, for any fixed signature* (S_0, F_0), *we can choose the free extension* $\varphi'\colon (S', F') \to (S', F'_1)$ *such that*

$$(S_0, F_0) \cap (S', F'_1) \subseteq (S_0, F_0) \cap (S', F').$$

On the sequential instantiation of parameters

Free extensions - properties

- consider the next presentation of a free extension of an idempotent morphism φ, i.e. $\varphi;\varphi = \varphi$

$$\begin{tikzcd}
\Sigma \arrow[r, "\subseteq"] \arrow[d, bend right=60, "\varphi"'] \arrow[dr, phantom, "FE"] & \Sigma' \arrow[d, bend left=60, "\varphi'"] \\
\Sigma \arrow[r, "\subseteq"'] & \Sigma'_1
\end{tikzcd}$$

On the sequential instantiation of parameters

Free extensions - properties

- consider the next presentation of a free extension of an idempotent morphism φ, i.e. $\varphi;\varphi = \varphi$
- factor φ and φ'

$$\begin{tikzcd}
\Sigma \arrow[r, "\subseteq"] \arrow[d, "e_\varphi"'] \arrow[dd, bend right=60, "\varphi"'] & \Sigma' \arrow[d, "e_{\varphi'}"] \arrow[dd, bend left=60, "\varphi'"] \\
\varphi(\Sigma) \arrow[d, "i_\varphi"'] \arrow[r, phantom, "FE"] & \varphi'(\Sigma') \arrow[d, "i_{\varphi'}"] \\
\Sigma \arrow[r, "\subseteq"'] & \Sigma'_1
\end{tikzcd}$$

On the sequential instantiation of parameters

Free extensions - properties

- consider the next presentation of a free extension of an idempotent morphism φ, i.e. $\varphi ; \varphi = \varphi$
- factor φ and φ'
- then $\varphi(\Sigma) \subseteq \varphi'(\Sigma')$

$$\begin{tikzcd}
\Sigma \arrow[r, "\subseteq"] \arrow[d, "e_{\varphi}"'] \arrow[dd, bend right=60, "\varphi"'] & \Sigma' \arrow[d, "e_{\varphi'}"] \arrow[dd, bend left=60, "\varphi'"] \\
\varphi(\Sigma) \arrow[r, "\subseteq" description] \arrow[d, "i_{\varphi}"'] & \varphi'(\Sigma') \arrow[d, "i_{\varphi'}"] \\
\Sigma \arrow[r, "\subseteq"'] & \Sigma'_1
\end{tikzcd}$$

On the sequential instantiation of parameters

Free extensions - properties

- consider the next presentation of a free extension of an idempotent morphism φ, i.e. $\varphi;\varphi = \varphi$
- factor φ and φ'
- then $\varphi(\Sigma) \subseteq \varphi'(\Sigma')$

$$\begin{tikzcd}
\Sigma \arrow[r, "\subseteq"] \arrow[d, "e_\varphi"'] \arrow[dd, bend right=60, "\varphi"'] \arrow[dr, phantom, "FE"] & \Sigma' \arrow[d, "e_{\varphi'}"] \arrow[dd, bend left=60, "\varphi'"] \\
\varphi(\Sigma) \arrow[r, "\subseteq"] \arrow[d, "i_\varphi"'] \arrow[dr, phantom, "PO"] & \varphi'(\Sigma') \arrow[d, "i_{\varphi'}"] \\
\Sigma \arrow[r, "\subseteq"'] & \Sigma'_1
\end{tikzcd}$$

Proposition

If the outer square describes a free extension then both inner squares are pushout squares, with the upper one also describing a free extension.

On the sequential instantiation of parameters

Instantiation of parameters via free extensions

- consider the instantiation $\mathrm{SP}(P \Leftarrow v)$

$$\begin{tikzcd}
Sig(P) \cup Sig(P') \arrow[r, "\subseteq"] \arrow[d, "v+1_{Sig(P')}"'] \arrow[dr, phantom, "PO"] & Sig(SP) \cup Sig(P') \arrow[d, "\nu'"] \\
Sig(P') \arrow[r, "\subseteq"'] & \Sigma'
\end{tikzcd}$$

On the sequential instantiation of parameters

Instantiation of parameters via free extensions

- consider the instantiation $\mathrm{SP}(P \Leftarrow v)$
- extend the instantiation diagram with inclusions like below

$$\begin{tikzcd}
\mathit{Sig}(P) \cup \mathit{Sig}(P') \arrow[r, "\subseteq"] \arrow[d, swap, "v + 1_{\mathit{Sig}(P')}"] \arrow[dr, phantom, "PO"] & \mathit{Sig}(SP) \cup \mathit{Sig}(P') \arrow[d, "\nu'"] \\
\mathit{Sig}(P') \arrow[r, swap, "\subseteq"] \arrow[d, swap, "i"] \arrow[d, "\subseteq"] & \Sigma' \arrow[d, swap, "\subseteq"] \arrow[d, "i'"] \\
\mathit{Sig}(P) \cup \mathit{Sig}(P') \arrow[r, swap, "\subseteq"] & \Sigma'_1
\end{tikzcd}$$

On the sequential instantiation of parameters

Instantiation of parameters via free extensions

- consider the instantiation $\mathrm{SP}(P \Leftarrow v)$
- extend the instantiation diagram with inclusions like below
- *restrict the instantiation such that* $\nu'; i'$ *is a free extension of the idempotent morphism* $(v + 1_{Sig(P')}); i$

$$\begin{tikzcd}
Sig(P) \cup Sig(P') \arrow[r, "\subseteq"] \arrow[d, "v+1_{Sig(P')}"'] \arrow[dd, bend right=60, "(v+1_{Sig(P')});i"'] \arrow[dr, phantom, "PO"] & Sig(SP) \cup Sig(P') \arrow[d, "\nu'"] \arrow[dd, bend left=60, "\nu';i'"] \\
Sig(P') \arrow[r, "\subseteq"'] \arrow[d, "i"', "\subseteq"] & \Sigma' \arrow[d, "\subseteq"', "i'"] \\
Sig(P) \cup Sig(P') \arrow[r, "\subseteq"'] & \Sigma'_1
\end{tikzcd}$$

On the sequential instantiation of parameters

Instantiation of parameters via free extensions

Proposition

Let

- $\mathrm{SP}(P_1, \ldots, P_n)$ *be a multiple parameterized specification and*
- $v_1 : P_1 \to P_1'$ *a morphism such that* $Sig(P_1')$ *and* $Sig(P_j)$ *are disjoint, for* $1 \leq j \leq n$.

If the instantiation is restricted by free extensions then

$$\mathrm{SP}(P_1 \Leftarrow v_1)(P_2, \ldots, P_n)$$

is a parameterized specification.

On the sequential instantiation of parameters

Instantiation of parameters via free extensions

Example (*PAIR_PNAT_BOOL*)

$$
\begin{aligned}
&PAIR(ELT \star E1 \Leftarrow \ \textbf{view from } ELT \star E1 \textbf{ to } PNAT\{\ \mathrm{Elt.E1} \mapsto Nat\ \})\\
&\qquad (ELT \star E2 \Leftarrow \ \textbf{view from } ELT \star E2 \textbf{ to } BOOL\{\ \mathrm{Elt.E2} \mapsto Bool\ \})
\end{aligned}
$$

$$\begin{tikzcd}
ELT \star E1 + PNAT \arrow[r, "\subseteq"] \arrow[d, "v_1 + 1_{PNAT}"'] \arrow[dr, phantom, "PO"] & PAIR + PNAT \arrow[d, "\nu'_1"] & \\
PNAT \arrow[r, "\subseteq"'] & PAIR(ELT \star E1 \Leftarrow \ldots) \arrow[d, dashed, "\subseteq"'] & ELT \star E2 \arrow[l, dashed, "\supseteq"'] \arrow[d, dashed, "\subseteq"] \\
& PAIR(ELT \star E1 \Leftarrow \ldots) + BOOL \arrow[d, "\nu'_2"'] \arrow[dr, phantom, "PO"] & ELT \star E2 + BOOL \arrow[l, "\supseteq"'] \arrow[d, "v_2 + 1_{BOOL}"] \\
& PAIR(ELT \star E1 \Leftarrow \ldots)(ELT \star E2 \Leftarrow \ldots) & BOOL \arrow[l, "\supseteq"']
\end{tikzcd}$$

Simultaneous vs sequential instantiation of parameters

- Are there any differences between the following results of the instantiation of *PAIR*?

$$\begin{aligned}&PAIR(ELT\star E1 \Leftarrow \ \textbf{view from } ELT\star E1 \textbf{ to } PNAT\{\ \text{Elt.E1} \mapsto Nat\ \},\\ &\qquad ELT\star E2 \Leftarrow \ \textbf{view from } ELT\star E2 \textbf{ to } BOOL\{\ \text{Elt.E2} \mapsto Bool\ \})\end{aligned}$$

(simultaneous instantiation)

$$\begin{aligned}&PAIR(ELT\star E1 \Leftarrow \ \textbf{view from } ELT\star E1 \textbf{ to } PNAT\{\ \text{Elt.E1} \mapsto Nat\ \})\\ &\qquad (ELT\star E2 \Leftarrow \ \textbf{view from } ELT\star E2 \textbf{ to } BOOL\{\ \text{Elt.E2} \mapsto Bool\ \})\end{aligned}$$

(sequential instantiation)

$$\begin{aligned}&PAIR(ELT\star E2 \Leftarrow \ \textbf{view from } ELT\star E2 \textbf{ to } BOOL\{\ \text{Elt.E2} \mapsto Bool\ \})\\ &\qquad (ELT\star E1 \Leftarrow \ \textbf{view from } ELT\star E1 \textbf{ to } PNAT\{\ \text{Elt.E1} \mapsto Nat\ \})\end{aligned}$$

(sequential instantiation)

Simultaneous vs sequential instantiation of parameters

Theorem

Let $\mathrm{SP}(P_1, P_2)$ *be a multiple parameterized specification. If*

- *the inclusion system of the signatures is epic and distributive,*
- *there exists an initial signature* 0 *that is also initial with respect to the signature inclusions, and*
- *each idempotent signature morphism has free extensions along any signature inclusion,*

then the results of the simultaneous and the sequential instantiation of multiple parameters are isomorphic such that the diagram below is commutative, provided that $\mathrm{SP}(P_1 \Leftarrow v_1)$ *can be chosen such that* $Sig[\mathrm{SP}_2] \cap Sig[\mathrm{SP}(P_1 \Leftarrow v_1)] \subseteq Sig[\mathrm{SP} \cup \mathrm{SP}_1]$.

$$\begin{tikzcd}
\mathrm{SP}(P_1 \cup P_2 \Leftarrow v_1 + v_2) \arrow[rr, "\cong"] & & \mathrm{SP}(P_1 \Leftarrow v_1)(P_2 \Leftarrow v_2) \\
& \mathrm{SP}_1 \cup \mathrm{SP}_2 \arrow[ul, "\subseteq"] \arrow[ur, "\subseteq"'] &
\end{tikzcd}$$

Thank you!

Parameterized specifications

Instantiation of parameters

Definition (Pushout square (of signatures))

A commutative square of signature morphisms $[\varphi_1, \varphi_2, \varphi_1', \varphi_2']$ as below is a *pushout square* when for any other commutative square $[\varphi_1, \varphi_2, \theta_1, \theta_2]$ with the same upper and left edges there exists an unique morphism θ' making the diagram commutative.

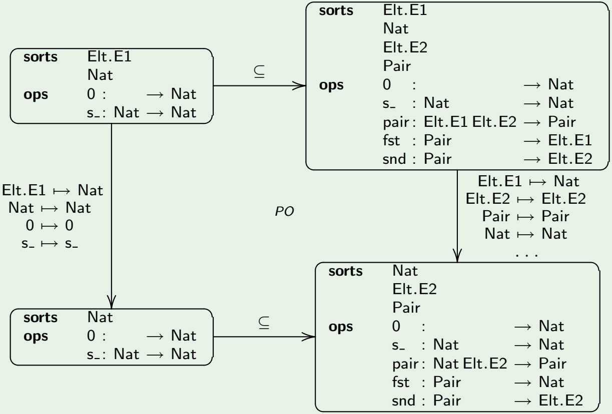
$$\begin{tikzcd}
\Sigma \arrow[r, "\varphi_1"] \arrow[d, "\varphi_2"'] & \Sigma_1 \arrow[d, "\varphi_1'"] \arrow[ddr, dashed, bend left, "\theta_1"] & \\
\Sigma_2 \arrow[r, "\varphi_2'"'] \arrow[drr, dashed, bend right, "\theta_2"'] & \Sigma_2 \arrow[dr, dashed, "\theta'"] & \\
 & & \Sigma'
\end{tikzcd}$$

◂ Back

On the sequential instantiation of parameters

Instantiation of parameters via free extensions

Example (*PAIR_PNAT_BOOL*)



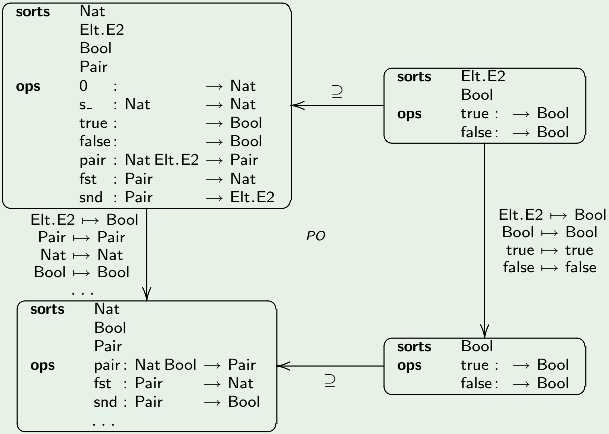
$$\begin{tikzcd}
\boxed{\begin{array}{lllll} \textbf{sorts} & \mathrm{Elt.E1} & & & \\ & \mathrm{Nat} & & & \\ \textbf{ops} & 0: & & \longrightarrow & \mathrm{Nat} \\ & s\_: & \mathrm{Nat} & \longrightarrow & \mathrm{Nat} \end{array}} \arrow[r, "\subseteq"] \arrow[d, "\begin{array}{rcl} \mathrm{Elt.E1} & \mapsto & \mathrm{Nat} \\ \mathrm{Nat} & \mapsto & \mathrm{Nat} \\ 0 & \mapsto & 0 \\ s\_ & \mapsto & s\_ \end{array}"'] \arrow[dr, phantom, "PO"] & \boxed{\begin{array}{lllll} \textbf{sorts} & \mathrm{Elt.E1} & & & \\ & \mathrm{Nat} & & & \\ & \mathrm{Elt.E2} & & & \\ & \mathrm{Pair} & & & \\ \textbf{ops} & 0 & : & \longrightarrow & \mathrm{Nat} \\ & s\_ & : \mathrm{Nat} & \longrightarrow & \mathrm{Nat} \\ & \mathrm{pair} & : \mathrm{Elt.E1}\ \mathrm{Elt.E2} & \longrightarrow & \mathrm{Pair} \\ & \mathrm{fst} & : \mathrm{Pair} & \longrightarrow & \mathrm{Elt.E1} \\ & \mathrm{snd} & : \mathrm{Pair} & \longrightarrow & \mathrm{Elt.E2} \end{array}} \arrow[d, "\begin{array}{rcl} \mathrm{Elt.E1} & \mapsto & \mathrm{Nat} \\ \mathrm{Elt.E2} & \mapsto & \mathrm{Elt.E2} \\ \mathrm{Pair} & \mapsto & \mathrm{Pair} \\ \mathrm{Nat} & \mapsto & \mathrm{Nat} \\ & \ldots & \end{array}"] \\
\boxed{\begin{array}{lllll} \textbf{sorts} & \mathrm{Nat} & & & \\ \textbf{ops} & 0: & & \longrightarrow & \mathrm{Nat} \\ & s\_: & \mathrm{Nat} & \longrightarrow & \mathrm{Nat} \end{array}} \arrow[r, "\subseteq"] & \boxed{\begin{array}{lllll} \textbf{sorts} & \mathrm{Nat} & & & \\ & \mathrm{Elt.E2} & & & \\ & \mathrm{Pair} & & & \\ \textbf{ops} & 0 & : & \longrightarrow & \mathrm{Nat} \\ & s\_ & : \mathrm{Nat} & \longrightarrow & \mathrm{Nat} \\ & \mathrm{pair} & : \mathrm{Nat}\ \mathrm{Elt.E2} & \longrightarrow & \mathrm{Pair} \\ & \mathrm{fst} & : \mathrm{Pair} & \longrightarrow & \mathrm{Nat} \\ & \mathrm{snd} & : \mathrm{Pair} & \longrightarrow & \mathrm{Elt.E2} \end{array}}
\end{tikzcd}$$

◂ Back

On the sequential instantiation of parameters

Instantiation of parameters via free extensions

Example (*PAIR_PNAT_BOOL*)



sorts Nat
Elt.E2
Bool
Pair
ops 0 : → Nat
s_ : Nat → Nat
true : → Bool
false : → Bool
pair : Nat Elt.E2 → Pair
fst : Pair → Nat
snd : Pair → Elt.E2

$\supseteq$

sorts Elt.E2
Bool
ops true : → Bool
false : → Bool

Elt.E2 ↦ Bool
Pair ↦ Pair
Nat ↦ Nat
Bool ↦ Bool
. . .

PO

Elt.E2 ↦ Bool
Bool ↦ Bool
true ↦ true
false ↦ false

sorts Nat
Bool
Pair
ops pair : Nat Bool → Pair
fst : Pair → Nat
snd : Pair → Bool
. . .

$\supseteq$

sorts Bool
ops true : → Bool
false : → Bool

◂ Back