# On Injections

Grigore Roșu

August 15, 2018

## 1 Introduction

## 2 Parametric Productions

Recall parametric productions (written using frontend syntax):

$$\texttt{syntax}\{P_1, \ldots, P_k\}\ N ::= T_1 N_1 \ldots T_n N_n T_{n+1}$$

The non-terminals $N, N_1, \ldots, N_n$ can make use of parameters $P_1, \ldots, P_k$. We also discussed about allowing the parameters to be optionally specified among the terminals $T_1, \ldots, T_n, T_{n+1}$ and nonterminals $N_1, \ldots, N_n$, e.g. $T_1\{P_1\}N_1 \ldots$, but that is irrelevant for this discussion. The implicit semantics of parametric productions is that of infinitely many instances, one for each concrete parameter instance. In KORE, to each production like above we associate a parametric symbol:

$$\sigma\{P_1, \ldots, P_k\}(N_1, \ldots, N_n) : N$$

## 3 Injections: The Problem

Now let's consider the special case of injections, which correspond to productions like above where $n = 1$ and $T_1 = T_2 = \epsilon$, that is,

$$\texttt{syntax}\ \{P_1, \ldots, P_k\}N ::= N_1$$

As a concrete example, assume the following parametric definitions:

$$\texttt{sort}\ \texttt{Map}\{K, V\}$$
$$\texttt{syntax}\{V\}\ \texttt{MapInt}\{V\} ::= \texttt{Map}\{\texttt{Int}, V\}$$

Dorel: A minor observation. Even if we use curly brackets for the list of parameters, their order in the list is important for instantiation, Otherwise, we should write something like $\texttt{Map}\{K \Rightarrow \texttt{Int}, V \Rightarrow V\}$. If the parameters order matters, then their names in the "sort" definition/production does not matter.

Grigore: Agreed. The name of parameters in sort declarations is irrelevant, but the order is. So we can just as well write
`sort Map{_, _}`
Let me know if you think the names should matter. I regard them same as function arguments, where the order matters.

Dorel: I was thinking that the second syntax production can be simply written as

$$\texttt{syntax}\{V\}\ \texttt{MapInt} ::= \texttt{Map}\{\texttt{Int}, V\}$$

i.e., a production

$$\texttt{syntax}\{P_1, \ldots, P_k\}\ N ::= \ldots$$

is equivalent to

$$\texttt{syntax}\{P_1, \ldots, P_k\}\ N\{P_1, \ldots, P_k\} ::= \ldots$$

Grigore: agreed and fixed.

Dorel: Ihm, I guess I have to take it back and I claim now that the two productions have different semantics. For instance, if we want to define a sort of all Maps, then we may write a production like

$$\texttt{syntax}\{K, V\}\ \texttt{MapUniv} ::= \texttt{Map}\{K, V\}$$

which is clearly different from

$$\texttt{syntax}\{K, V\}\ \texttt{MapUniv}\{K, V\} ::= \texttt{Map}\{K, V\}$$

and from

$$\texttt{syntax}\{K, V\}\ \texttt{MapUniv}\{K\} ::= \texttt{Map}\{K, V\}$$

and from

$$\texttt{syntax}\{K, V\}\ \texttt{MapUniv}\{V\} ::= \texttt{Map}\{K, V\}$$

In the first case we have

$$\theta(\texttt{Map}\{K, V\}) = \texttt{Map}\{\theta(K), \theta(V)\} \leq \theta(\texttt{MapUniv}) = \texttt{MapUniv}$$

in the second case,

$$\theta(\texttt{Map}\{K, V\}) = \texttt{Map}\{\theta(K), \theta(V)\} \leq \theta(\texttt{MapUniv\{K,V\}}) = \texttt{MapUniv}\{\theta(K), \theta(V)\}$$

in the third case,

$$\theta(\texttt{Map}\{K, V\}) = \texttt{Map}\{\theta(K), \theta(V)\} \leq \theta(\texttt{MapUniv\{K\}}) = \texttt{MapUniv}\{\theta(K)\}$$

and in the fourth case,

$$\theta(\texttt{Map}\{K, V\}) = \texttt{Map}\{\theta(K), \theta(V)\} \leq \theta(\texttt{MapUniv\{V\}}) = \texttt{MapUniv}\{\theta(V)\}$$

for each instance $\theta$ of the two parameters $K$ and $V$.
By abuse of notation, let

$$\texttt{inj}_{N_1,N}\{P_1,\ldots,P_k\}(N_1):N$$

be the parametric symbol corresponding to the generic injection production above. Note that, for now, `inj` is *not* parametric in the two sorts, each $\texttt{inj}_{N_1,N}$ is a separately-defined ordinary symbol. For our example,

$$\texttt{inj}_{\texttt{Map}\{\texttt{Int},V\},\texttt{MapInt}\{V\}}\{V\}(\texttt{Map}\{\texttt{Int},V\}):\texttt{MapInt}\{V\}$$

Brandon: if each way of subscripting `inj` is a distinct non-parametric symbol, how do we have parameter $V$

Dorel: I think that the (ordinary) injections should be defined only for the ground instances. I guess that Grigore wants to say that the name of injection is unique to each subsort production. Since the production is parametric in $V$, the name of the injection is parametric in $V$ as well. If $\theta$ and $\theta'$ are two parameter instantiations, then we have two (ordinary/concrete) injections: $\texttt{inj}_{\texttt{Map}\{\texttt{Int},V\},\texttt{MapInt}\{V\}}\{\theta(V)\}$ and $\texttt{inj}_{\texttt{Map}\{\texttt{Int},V\},\texttt{MapInt}\{V\}}\{\theta'(V)\}$ (note that the name of the operation is unchanged). However, the name of the parameter is not important in the name of the operation.

Injection symbols are different from ordinary symbols, in that they inherit an expected semantics of "injections". For example, take an instance $\theta$ of the parameters $P_1,\ldots,P_k$. Then any element of sort $\theta(N_1)$ is expected to also be found among the elements of sort $\theta(N)$, possibly renamed.
Dorel: It is not clear for me how the renaming can be involved here. I think that $P_1,\ldots,P_k$ are the only parameters allowed in the production definition.

Grigore: I was talking about elements in models. I do not want to enforce subsets for subsorts, like in OSA.
Moreover, such injections are expected to be consistent.

**Example 3.1.** For example, assume another production

$$\texttt{syntax}\{Q_1,\ldots,Q_\ell\}\ M ::= M_1$$

with corresponding injection

$$\texttt{inj}_{M_1,M}\{Q_1,\ldots,Q_\ell\}(M_1):M$$

such that there is some instance $\rho$ of the parameters $Q_1,\ldots,Q_\ell$ with $\rho(M_1)=\theta(N_1)$ and $\rho(M)=\theta(N)$.
Dorel: We should formally define what exactly means an equality $\rho(M)=\theta(N)$.

Grigore: I thought it is obvious. M and N are terms of sort Sort, so it is the usual extension of substitutions from variables to terms.

Dorel: Meanwhile I included the formal definitions at the end of the Section 4.

Dorel: Here is an example showing how I understand it:
We suppose that the above example is written as follows:

$$\begin{array}{l}\texttt{sort Map}\{K, V\}\\ \texttt{syntax}\{V'\}\ \texttt{MapInt}\{V'\} ::= \texttt{Map}\{\texttt{Int}, V'\}\end{array}$$

(this could be a particular example when the parameter $V$ is renamed as $V'$).
If $\theta(K) = \texttt{Int}$, $\theta(V) = \texttt{String}$, and $\rho(V') = \texttt{String}$ then $\theta(\texttt{Map}\{K, V\}) = \rho(\texttt{Map}\{\texttt{Int}, V'\})$.
Supposing that the grammar include only these productions, can we deduce that $\theta(\texttt{Map}\{K, V\}) = \rho(\texttt{MapInt}\{V'\})$? This could makes sense since $\texttt{MapInt}\{V'\}$ is the smallest set including $\texttt{Map}\{\texttt{Int}, V'\}$.

Brandon: I think this simply means applying the substitution gives identical instantiations of the same parameterized sort.

Dorel: You are right. I mentioned the "semantic" equality $\theta(\texttt{Map}\{K, V\}) = \rho(\texttt{MapInt}\{V'\})$ just as a possible extension. I remember that we had in the past a question regarding how to define a synonym of a sort.

Grigore: yes, this is true, but it is true even if you have more productions. Recall that the semantics of parameters is "all instances". So $\theta$ and $\rho$ are two such instances that make the two symbols equal. Which is what the equality below enforces semantically.
Then we certainly want the following to hold:

$$\texttt{inj}_{N_1,N}\{\theta(\bar{P})\}(\varphi : \theta(N_1)) = \texttt{inj}_{M_1,M}\{\rho(\bar{Q})\}(\varphi : \rho(M_1))$$

Grigore: What I meant was that identical instances of two different parametric injection symbols should be identical semantically, too. That is, it is the same injection, not two different ones. I regard it more like "injections are consistent wrt parameter instances".

**Example 3.2.** As another example, assume productions

$$\begin{array}{c}\texttt{syntax}\{A_1, \ldots, A_a\}\ X ::= X_1\\ \texttt{syntax}\{B_1, \ldots, B_b\}\ Y_1 ::= Y_2\\ \texttt{syntax}\{C_1, \ldots, C_c\}\ Z ::= Z_2\end{array}$$

such that there are some parameter instances $\alpha$, $\beta$, and $\gamma$ with $\alpha(X_1) = \beta(Y_1)$, $\alpha(X) = \gamma(Z)$, and $\beta(Y_2) = \gamma(Z_2)$. Then we certainly want

$$\texttt{inj}_{X_1,X}\{\alpha(\bar{A})\}(\texttt{inj}_{Y_2,Y_1}\{\beta(\bar{B})\}(\varphi : \beta(Y_2))) = \texttt{inj}_{Z_2,Z}\{\gamma(\bar{C})\}(\varphi : \gamma(Z_2))$$

Dorel: This should hold even if $\alpha$, $\beta$, and $\gamma$ are non-ground substitutions (= parameter instances), case in which the above is an axiom schema.

Grigore: Your intuition is probably right. But the way I looked at it was more like "injections compose consistently".

**Example 3.3.** We also want parametric sorts to lift subsorts. That is, if

$$\texttt{inj}_{N_1,N}\{P_1,\ldots,P_k\}(N_1) : N$$

is an injection parametric symbol corresponding to production

$$\texttt{syntax}\{P_1,\ldots,P_k\}\ N ::= N_1,$$

then we also want to have parametric symbols

$$\texttt{inj}_{sort\{N_1,\ldots\},sort\{N,\ldots\}}\{P_1,\ldots,P_k,\ldots\}(sort\{N_1,\ldots\}) : sort\{N,\ldots\}$$

for any other parametric sort $sort\{\_,\ldots\}$.
Dorel: Here is an instance of the above rule. Suppose that $sort\{\_,\ldots\}$ is $\texttt{List}\{S\}$, then we have

$$\texttt{inj}_{\texttt{List}\{\texttt{Map}\{\texttt{Int}\{\},V\}\},\texttt{List}\{\texttt{MapInt}\{V\}\}}\{V\}(\texttt{List}\{\texttt{MapInt}\{V\}\}) : \texttt{List}\{\texttt{Map}\{\texttt{Int}\{\},V\}\}$$

**Example 3.4.** The situation is actually a lot more complex! The injection symbols need to be consistent not only among themselves, but also w.r.t. other symbols that end up being overloaded due to parametricity. Consider, for example:

$$\begin{aligned}&\texttt{syntax Exp} ::= \texttt{Int}\\&\texttt{syntax}\{S\}\ \texttt{List}\{S\} ::= \texttt{cons}(S,\texttt{List}\{S\})\end{aligned}$$

or in KORE:

$$\begin{aligned}&\texttt{sort List}\{S\}\\&\texttt{symbol inj}_{\texttt{Int},\texttt{Exp}}(\texttt{Int}) : \texttt{Exp}\\&\texttt{symbol cons}\{S\}(S,\texttt{List}\{S\}) : \texttt{List}\{S\}\end{aligned}$$

Then we need to add the following axiom

$$\begin{aligned}&\texttt{inj}_{\texttt{List}\{\texttt{Int}\},\texttt{List}\{\texttt{Exp}\}}(\texttt{cons}\{\texttt{Int}\}(\varphi : \texttt{Int},\psi : \texttt{List}\{\texttt{Int}\}))\\&\qquad = \texttt{cons}\{\texttt{Exp}\}(\texttt{inj}_{\texttt{Int},\texttt{Exp}}(\varphi),\texttt{inj}_{\texttt{List}\{\texttt{Int}\},\texttt{List}\{\texttt{Exp}\}}(\psi))\end{aligned}$$

**Example 3.5.** But what if the result of the parametric symbol does not depend on the parameter, that is, say

$$\texttt{symbol length}\{S\}(\texttt{List}\{S\}) : \texttt{Int}$$

Then we need to add an axiom as follows

$$\texttt{length}\{\texttt{Int}\}(\varphi : \texttt{List}\{\texttt{Int}\}) = \texttt{length}\{\texttt{Exp}\}(\texttt{inj}_{\texttt{List}\{\texttt{Int}\},\texttt{List}\{\texttt{Exp}\}}(\varphi))$$

Dorel: This example is a particular case of the previous one if we consider

$$\texttt{inj}_{S,S}(\varphi) = \varphi$$

Grigore: Yes. And we can use this to justify adding such an equation. In fact, that's the role of all these examples here, to raise awareness of things that we need to axiomatize.

**Example 3.6.** On the other hand, if the result of a symbol depends on a parameter that is not constrained by its arguments, then we cannot add any injection axioms. For example, consider

$$\texttt{symbol cast}\{P_1, P_2\}(P_1) : P_2$$

While it makes sense to add axioms like

$$\texttt{inj}_{\texttt{Int},\texttt{Exp}}(\texttt{cast}\{\texttt{Int},\texttt{Int}\}(\varphi : \texttt{Int})) = \texttt{cast}\{\texttt{Int},\texttt{Exp}\}(\varphi)$$

you cannot instantiate $P_2$ with everything, e.g., you cannot add

$$\texttt{inj}_{\texttt{Int},\texttt{String}}(\texttt{cast}\{\texttt{Int},\texttt{Int}\}(\varphi : \texttt{Int}) = \texttt{cast}\{\texttt{Int},\texttt{String}\}(\varphi)$$

because `Int` is not a subset of `String`.
Dorel: The rule we can learn from this example is: "we have to consider only those injections defined on sort instances that are in a *subsort partial-order relation*".

Grigore: well, I am not sure we can achieve that staying at the core level, because you have infinitely many subsortings ... as I said in Section 4(1) below.

## 4 Injections: The Proposal

So things are really tricky. What I propose is to add the following axioms for now, and *in parallel* to have somebody really interested in this problem study it in depth. Note that knowledge of order-sorted algebra will be a big plus here!

(1) Define/assume one parametric symbol

$$\texttt{symbol inj}\{P_1, P_2\}(P_1) : P_2$$

This is what we do now, too. Note that this is *different* from having one `inj` symbol for each subsorting. In particular, for the subsorting discussed in the previous section,

$$\texttt{syntax}\{V\}\ \texttt{MapInt}\{V\} ::= \texttt{Map}\{\texttt{Int}\{\}, V\}$$

instead of the injection label we had there, namely

$$\texttt{symbol inj}_{\texttt{Map}\{\texttt{Int}\{\},V\},\texttt{MapInt}\{V\}}\{V\}(\texttt{Map}\{\texttt{Int}\{\}, V\}) : \texttt{MapInt}\{V\}$$

we refer to this subsorting using the generic injection

$$\texttt{inj}\{\texttt{Map}\{\texttt{Int}\{\}, V\}, \texttt{MapInt}\{V\}\}.$$

I do not know how to state that a subsorting has been declared, or if it is needed, so I postpone this aspect for now.

Dorel: Here is a proposal, based on my remarks included at the end of the paper.
Each subsort parametric production $\pi$

$$\texttt{syntax}\{P_1, \ldots, P_k\}\ N ::= N_1,$$

defines a unique injection parametric symbol

$$\texttt{inj}_\pi\{P_1, \ldots, P_k\}(N_1) : N$$

We assume that the name $\pi$ uniquely identifies the susbsort production. The instances of the above parametric symbol are of the form

$$\texttt{inj}_\pi\{\theta(P_1), \ldots, \theta(P_k)\}(\theta(N_1)) : \theta(N)$$

where $\theta$ is a parameter instantiation, i.e. $\theta(P_i)$ is a concrete sort and $\theta(N_1)$, $\theta(N)$ are the extensions of $\theta$ to sort-terms. The relationship between this parametric symbol associated to each production and the generic one

$$\texttt{symbol inj}\{P'_1, P'_2\}(P'_1) : P'_2$$

can be expressed by the existence of an instantiation $\gamma_{\pi,\theta}$ of $\{P'_1, P'_2\}$, for each $\theta$ over $\bar{P}$, such that

$$\gamma_{\pi,\theta}(P'_1) = \theta(N_1),\ \gamma_{\pi,\theta}(P'_2) = \theta(N)$$

i.e.

$$\texttt{inj}_\pi\{\theta(P_1), \ldots, \theta(P_k)\}(\theta(N_1)) : \theta(N) = \texttt{inj}\{\gamma_{\pi,\theta}(P'_1), \gamma_{\pi,\theta}(P'_2)\}(\gamma_{\pi,\theta}(P'_1)) : \gamma_{\pi,\theta}(P'_2)$$

Note that $\gamma_{\pi,\theta}$ is uniquely determined by the production $\pi$ (which supplies $N_1$ and $N$) and the parameter instantiation $\theta$. Moreover, we have

$$\gamma_{\pi,\theta}(P_1') \leq \gamma_{\pi,\theta}(P_2')$$

in the terms of ordered sorts.

We discuss the above examples in this new setting.

**Example 3.1.** Let $\pi'$ the production

$$\texttt{syntax}\{Q_1, \ldots, Q_\ell\}\ M ::= M_1$$

Then the equalities

$$\rho(M_1) = \theta(N_1) \text{ and } \rho(M) = \theta(N)$$

become

$$\gamma_{\pi',\rho}(P_1) = \gamma_{\pi,\theta}(P_1) \text{ and } \gamma_{\pi',\rho}(P_2) = \gamma_{\pi,\theta}(P_2),$$

i.e., $\gamma_{\pi',\rho} = \gamma_{\pi,\theta}$ as parameter instantiations over $\{P_1', P_2'\}$. Then the equality

$$\texttt{inj}_\pi\{\theta(\bar{P})\}(\varphi : \theta(N_1)) = \texttt{inj}_{\pi'}\{\rho(\bar{Q})\}(\varphi : \rho(M_1))$$

becomes a trivial (syntactic) one

$$\texttt{inj}\{\gamma_{\pi,\theta}(P_1'), \gamma_{\pi,\theta}(P_2')\}(\varphi : \gamma_{\pi,\theta}(P_1')) = \texttt{inj}\{\gamma_{\pi',\rho}(P_1'), \gamma_{\pi',\rho}(P_2')\}(\varphi : \gamma_{\pi',\rho}(P_1'))$$

**Example 3.2.** Let $\pi_x$, $\pi_Y$, and $\pi_Z$, respectively, the following productions:

$$\begin{gathered}\texttt{syntax}\{A_1, \ldots, A_a\}\ X ::= X_1 \\ \texttt{syntax}\{B_1, \ldots, B_b\}\ Y_1 ::= Y_2 \\ \texttt{syntax}\{C_1, \ldots, C_c\}\ Z ::= Z_2\end{gathered}$$

In order to avoid confusions, we denote with $\gamma'$ the instantiation over $\bar{C}$. Then the equalities

$$\alpha(X_1) = \beta(Y_1),\ \alpha(X) = \gamma'(Z), \text{ and } \beta(Y_2) = \gamma'(Z_2)$$

become

$$\begin{aligned}\gamma_{\pi_X,\alpha}(P_1') &= \gamma_{\pi_Y,\beta}(P_2') \\ \gamma_{\pi_X,\alpha}(P_2') &= \gamma_{\pi_Z,\gamma'}(P_2') \\ \gamma_{\pi_Y,\beta}(P_1') &= \gamma_{\pi_Z,\gamma'}(P_1')\end{aligned}$$

which together with

$$\gamma_{\pi_X,\alpha}(P_1') \leq \gamma_{\pi_X,\alpha}(P_2') \quad \gamma_{\pi_Y,\beta}(P_1') \leq \gamma_{\pi_Y,\beta}(P_2') \quad \gamma_{\pi_Z,\gamma'}(P_1') \leq \gamma_{\pi_Z,\gamma'}(P_2')$$

implies

$$\begin{aligned}
&\gamma_{\pi_Y,\beta}(P'_1) \leq \gamma_{\pi_Y,\beta}(P'_2) = \gamma_{\pi_X,\alpha}(P'_1) \leq \gamma_{\pi_X,\alpha}(P'_2) = \gamma_{\pi_Z,\gamma'}(P'_2)\\
&\gamma_{\pi_Y,\beta}(P'_1) = \gamma_{\pi_Z,\gamma'}(P'_1) \qquad\qquad\qquad\quad \leq \gamma_{\pi_Z,\gamma'}(P'_2) = \gamma_{\pi_X,\alpha}(P'_2)
\end{aligned}$$

Let us consider three new (fresh) variables $P_1, P_2, P_3$ and a substitution $\theta$ such that

$$\begin{aligned}
\theta(P_2) &= \gamma_{\pi_X,\alpha}(P'_1) = \gamma_{\pi_Y,\beta}(P'_2)\\
\theta(P_3) &= \gamma_{\pi_X,\alpha}(P'_2) = \gamma_{\pi_Z,\gamma'}(P'_2)\\
\theta(P_1) &= \gamma_{\pi_Y,\beta}(P'_1) = \gamma_{\pi_Z,\gamma'}(P'_1)
\end{aligned}$$

then the equality

$$\mathtt{inj}_{X_1,X}\{\alpha(\bar{A})\}(\mathtt{inj}_{Y_2,Y_1}\{\beta(\bar{B})\}(\varphi : \beta(Y_2))) = \mathtt{inj}_{Z_2,Z}\{\gamma'(\bar{C})\}(\varphi : \gamma'(Z_2))$$

becomes

$$\begin{aligned}
&\mathtt{inj}\{\gamma_{\pi_X,\alpha}(P'_1), \gamma_{\pi_X,\alpha}(P'_2)\}(\mathtt{inj}\{\gamma_{\pi_Y,\beta}(P'_1), \gamma_{\pi_Y,\beta}(P'_2)\}(\varphi : \gamma_{\pi_Y,\beta}(P'_1))) =\\
&\qquad\qquad\qquad\qquad \mathtt{inj}\{\gamma_{\pi_Z,\gamma'}(P'_1), \gamma_{\pi_Z,\gamma'}(P'_2)\}(\varphi : \gamma_{\pi_Z,\gamma'}(P'_1))
\end{aligned}$$

which is equivalent to

$$\mathtt{inj}\{\theta(P_2), \theta(P_3)\}(\mathtt{inj}\{\theta(P_1), \theta(P_2)\}(\varphi : \theta(P_1))) = \mathtt{inj}\{\theta(P_1), \theta(P_3)\}(\varphi : \theta(P_1))$$

and which is an instance of the parametric axiom

$$\mathtt{inj}\{P_2, P_3\}(\mathtt{inj}\{P_1, P_2\}(\varphi : P_1)) = \mathtt{inj}\{P_1, P_3\}(\varphi : P_1)$$

**Example 3.3.** It is equivalent to say that for each production in the input front-end grammar $G$

$$\mathtt{syntax}\{P_1, \ldots, P_k\}\ N ::= N_1,$$

and each parametric sort $\mathit{sort}\{\_, \ldots\}$ defined by $G$, there is a production

$$\mathtt{syntax}\{P_1, \ldots, P_k, \ldots\}\ \mathit{sort}\{N_\_, \ldots\} ::= \mathit{sort}\{N_{1\_}, \ldots\},$$

in $G$ as well. If the former production has the name $\pi$, then let $\mathit{sort}\{\pi_\_, \ldots\}$ denote the later one. Then we apply the usual mechanism for (parametric) productions.

**Example 3.4.** This refers to well-definedness of the overloaded operations.

Let $\pi$ the production

$$\mathtt{syntax}\{S\}\ \mathtt{List}\{S\} ::= \mathtt{cons}(S, \mathtt{List}\{S\})$$

This defines an overloaded operation

$$\sigma_\pi \in \Sigma_{\theta(S)\,\mathtt{List}\{\theta(S)\},\mathtt{List}\{\theta(S)\}}$$

for each sort instance $\theta(S)$. Note that the existence of an injection

$$\mathtt{inj}_\eta(\theta(S)) : \theta'(S)$$

implies the existence of an injection

$$\mathtt{inj}_{\mathtt{List}\{\eta\}}(\mathtt{List}\{\theta(S)\}) : \mathtt{List}\{\theta'(S)\}$$

by the rule given by the previous example. The well-definedness of $\sigma_\pi$ says that the restriction of

$$\sigma_\pi \in \Sigma_{\theta'(S)\,\mathtt{List}\{\theta'(S)\},\mathtt{List}\{\theta'(S)\}}$$

to $(\theta(S), \mathtt{List}\{\theta(S)\})$ must be equal to (or agree with)

$$\sigma_\pi \in \Sigma_{\theta(S)\,\mathtt{List}\{\theta(S)\},\mathtt{List}\{\theta(S)\}}.$$

But the restriction of $\sigma_\pi(\_,\_) \in \Sigma_{\theta'(S)\,\mathtt{List}\{\theta'(S)\},\mathtt{List}\{\theta'(S)\}}$ to $(\theta(S), \mathtt{List}\{\theta(S)\}$ is

$$\sigma_\pi(\mathtt{inj}_\eta(\_ : \theta(S)) : \theta'(S), \mathtt{inj}_{\mathtt{List}\{\eta\}}(\_ : \mathtt{List}\{\theta(S)\})) : \mathtt{List}\{\theta'(S)\}$$

The result of $\sigma_\pi(\_,\_) \in \Sigma_{\theta(S)\,\mathtt{List}\{\theta(S)\},\mathtt{List}\{\theta(S)\}}$ seen as an element of $\mathtt{List}\{\theta'(S)\}$ is

$$\mathtt{inj}_{\mathtt{List}\{\eta\}}(\sigma_\pi(\_ : \theta(S), \_ : \mathtt{List}\{\theta(S)\}) : \mathtt{List}\{\theta(S)\}) : \mathtt{List}\{\theta'(S)\}$$

Now the well-definedness of $\sigma_\pi$ can be expressed by the axiom

$$\begin{aligned}\sigma_\pi(\mathtt{inj}_\eta(\varphi_1 : \theta(S)) : \theta'(S), \mathtt{inj}_{\mathtt{List}\{\eta\}}(\varphi_2 : \mathtt{List}\{\theta(S)\}) : \mathtt{List}\{\theta'(S)\}) : \mathtt{List}\{\theta'(S)\} = \\ \mathtt{inj}_{\mathtt{List}\{\eta\}}(\sigma_\pi(\varphi_1 : \theta(S), \varphi_2 : \mathtt{List}\{\theta(S)\}) : \mathtt{List}\{\theta(S)\}) : \mathtt{List}\{\theta'(S)\}\end{aligned}$$

which is equivalent to

$$\begin{aligned}&\sigma_\pi(\mathtt{inj}\{\gamma_{\eta,\theta}(P'_1), \gamma_{\eta,\theta}(P'_2)\}(\varphi_1 : \gamma_{\eta,\theta}(P'_1)), \\ &\quad \mathtt{inj}\{\gamma_{\mathtt{List}\{\eta\},\theta}(\mathtt{List}\{P'_1\}), \gamma_{\mathtt{List}\{\eta\},\theta}(\mathtt{List}\{P'_2\})\}(\varphi_2 : \gamma_{\mathtt{List}\{\eta\},\theta}(\mathtt{List}\{P'_1\}))) : \mathtt{List}\{\theta'(S)\} = \\ &\quad \mathtt{inj}\{\gamma_{\mathtt{List}\{\eta\},\theta}(\mathtt{List}\{P'_1\}), \gamma_{\mathtt{List}\{\eta\},\theta}(\mathtt{List}\{P'_2\})\}(\sigma_\pi(\varphi_1 : \theta(S), \varphi_2 : \mathtt{List}\{\theta(S)\}) : \mathtt{List}\{\theta(S)\})\end{aligned}$$

Note that $\gamma_{\mathtt{List}\{\eta\},\theta}(P'_1) = \mathtt{List}\{\theta(S)\}$ by the definition of $\gamma_{\_,\_}$. If we consider the parametric symbol $\sigma_\pi\{S\}$ such that $\sigma_\pi \in \Sigma_{\theta(S)\,\mathtt{List}\{\theta(S)\}}$ can be represented by the symbol instance $\sigma_\pi\{\theta(S)\}$, then the well-definedness of $\sigma_\pi$ becomes equivalent to

$$\begin{aligned}&\sigma_\pi\{\gamma_{\eta,\theta'}(P'_2)\}(\mathtt{inj}\{\gamma_{\eta,\theta}(P'_1), \gamma_{\eta,\theta}(P'_2)\}(\varphi_1 : \gamma_{\eta,\theta}(P'_1)), \\ &\qquad \mathtt{inj}\{\gamma_{\mathtt{List}\{\eta\},\theta}(\mathtt{List}\{P'_1\}), \gamma_{\mathtt{List}\{\eta\},\theta}(\mathtt{List}\{P'_2\})\}(\varphi_2 : \gamma_{\mathtt{List}\{\eta\},\theta}(\mathtt{List}\{P'_1\}))) = \\ &\qquad\quad \mathtt{inj}\{\gamma_{\mathtt{List}\{\eta\},\theta}(\mathtt{List}\{P'_1\}), \gamma_{\mathtt{List}\{\eta\},\theta}(\mathtt{List}\{P'_2\})\}(\sigma_\pi\{\gamma_{\eta,\theta}(P'_1)\}(\varphi_1, \varphi_2))\end{aligned}$$

which is an instance of

$$\sigma_\pi\{P_2'\}(\texttt{inj}\{P_1', P_2'\}(\varphi_1 : P_1'),\\ \texttt{inj}\{\texttt{List}\{P_1'\}, \texttt{List}\{P_2'\}\}(\varphi_2 : \texttt{List}\{P_1'\})) =$$

$$\texttt{inj}\{\texttt{List}\{P_1'\}, \texttt{List}\{P_2'\}\}(\sigma_\pi\{P_1'\}(\varphi_1 : P_1', \varphi_2 : \texttt{List}\{P_1'\}))$$

(2) Axiomatize that `inj` is functional, because otherwise we will not be able to prove that $1 + x$ (i.e., $1 + \texttt{inj}\{\texttt{Id}, \texttt{Int}\}(x)$), etc., are "terms":

$$\texttt{axiom}\{P_1, P_2\}\ \forall x : P_1.\exists y : P_2.\texttt{inj}\{P_1, P_2\}(x) = y$$

Dorel: What if we want that an identifier not to be a `Bool` and an `Int` in the same time?

$$(\forall X)\neg(\texttt{inj}\{\texttt{Id}, \texttt{Bool}\}(X) \wedge \texttt{inj}\{\texttt{Id}, \texttt{Int}\}(X))$$

I do not think that we want to axiomatize that `inj` is an *injective* function, because we may want to inject sets of larger cardinalities (e.g.identifiers) into sets of smaller cardinalities (e.g., `Bool`: ($x$ or $y$) and $x = x$, etc.).

Dorel: I do not understand the above example. If $x$ and $y$ are two different identifiers, then their injections denote different `Bool` names.

Grigore: Think of models. Sort Id has infinitely many elements. Sort Bool has 2 elements. Then injId,Bool cannot be an injective function.

Dorel: Ihm, I do not think that the injections can be used to interpret expressions ...

(3) Axiomatize that `inj` is reflexive:

$$\texttt{axiom}\{S\}\ \texttt{inj}\{S, S\}(\varphi : S) = \varphi$$

(4) Axiomatize that injections compose (or are transitive):

$$\texttt{axiom}\{P_1, P_2, P_3\}\ \texttt{inj}\{P_2, P_3\}(\texttt{inj}\{P_1, P_2\}(\varphi : P_1)) = \texttt{inj}\{P_1, P_3\}(\varphi)$$

(5) And here is an aggressive axiom, but which I dare to claim is OK: Unrestricted propagation through parametric symbols. For each symbol

$\sigma\{P_1,\ldots,P_k\}(S_1,\ldots,S_n) : S$ where $P_1,\ldots,P_k$ are parameters and $S_1,\ldots,S_n,S$ are sorts potentially parametric in $P_1,\ldots,P_k$, add axiom

$$\begin{aligned}&\texttt{axiom}\{P_1,\ldots,P_k,P'_1,\ldots,P'_k\}\\ &\qquad\qquad \texttt{inj}\{S,S'\}(\sigma\{P_1,\ldots,P_k\}(\varphi_1 : S_1,\ldots,\varphi_n : S_n))\\ &\qquad\qquad\quad = \sigma\{P'_1,\ldots,P'_k\}(\texttt{inj}\{S_1,S'_1\}(\varphi_1),\ldots,\texttt{inj}\{S_n,S'_n\}(\varphi_n))\end{aligned}$$

OK, now I can hear you guys yelling at me, "what the heck is this? It is too aggressive!". In particular, that it admits nonsensical properties to be proven, like the one for the cast symbol in Example 3.6 above. And I would agree, but I do believe that we should either disallow symbols like in Example 3.6 above, or otherwise give a serious warning. Note that `inj` itself is such a symbol ;-). Dorel: Here is a counter-example, when the above axioms cannot be applied:

```
syntax Exp ::= Int
syntax Int ::= Int ":Int" [symbol("ucast")]
syntax Exp ::= Exp ":Exp" [symbol("ucast")]
```

Assume that `ucast` is parametric, i.e. it is represented by the instances `ucast{Int}` and `ucast{Exp}`. Then the above axiom becomes

$$\texttt{inj}\{\texttt{Int},\texttt{Exp}\}(\texttt{ucast}\{\texttt{Int}\}(\varphi : Int)) = \texttt{ucast}\{\texttt{Exp}\}(\texttt{inj}\{\texttt{Int},\texttt{Exp}\}(\varphi : Int))$$

But for `3:Exp` only the pattern `ucast{Exp}(inj{Int,Exp}(3))` correctly describes the user's intention to parse `3` as an `Exp`.

What makes me believe that axiom (5) above is OK is that it corresponds to the main requirement of order-sorted algebra, namely *regularity*. Let me elaborate.

## 4.1 Order-Sorted Regularity (or pre-regularity?)

In Order-Sorted Algebra(OSA), you have a partial order $(S,\leq)$ on sorts $S$. Symbols are allowed to be overloaded, but they must obey the *regularity* property in order for things to make sense mathematically:

**Definition 4.1.** $(S,\Sigma')$ is regular iff for any $\sigma : s_1 \times \cdots \times s_n \to s$ and $\sigma : s'_1 \times \cdots \times s'_n \to s'$ such that $s_1 \leq s'_1,\ldots,s_n \leq s'_n$ we have $s \leq s'$.

In our setting, since we disallow overloaded symbols except for parametric ones, and since we build our subset relation constructively through parametric sorts starting with a base subsort relation, I dare to claim two things:

(a) That the axiom in (5) above corresponds to regularity in OSA, which is therefore a reasonable thing to have;

(b) Under a mild restriction, that in each

$$\texttt{symbol}\ \sigma\{P_1, \ldots, P_n\}(S_1, \ldots, S_n) : S$$

the sorts $S_1, \ldots, S_n$ already refer to *all* parameters $P_1, \ldots, P_k$, we can show that the resulting order-sorted signature, whose only overloaded symbols are the parametric ones, is *regular*. The "resulting" partial order on sorts is the least relation $\leq$ closed under the following.

- `syntax` $S' ::= S$ implies $S \leq S'$
  Dorel: $S$ and $S'$ basic sorts?

  Brandon: Good point, what happens if these are instances of parametric sorts?

  Grigore: Good question ... I think they can be parametric, too. Do you have a counterexample? We'd need to prove it anyway.
- reflexive and transitive
- If $s_1 \leq s_1', \ldots, s_k \leq s_k'$ and $sort\{P_1, \ldots, P_k\}$ is a parametric sort, then $sort\{s_1, \ldots, s_k\} \leq sort\{s_1', \ldots, s_k'\}$.

Dorel: The following definitions are from [GM92]:

**Definition 4.2.** An *order-sorted signature* is a triple $(S, \leq, \Sigma)$ such that $(S, \Sigma)$ is a many-sorted signature, $(S, \leq)$ is a poset, and the operations satisfy the following *monotonicity condition*:

$$\sigma \in \Sigma_{w,s} \cap \Sigma_{w',s'} \text{ and } w \leq w' \text{ imply } s \leq s'.$$

Grigore: Do my conditions above imply this monotonicity condition?

Dorel: Unfortunately no. I included counterexamples at the end of Section 4.

Dorel:

**Remark 4.1.** If $w = s_1 \ldots s_m$ and $w' = s_1' \ldots s_n'$ then $w \leq w'$ iff $m = n$ and $s_i \leq s_i'$ for $i = 1, \ldots, n$.

**Definition 4.3.** An order-sorted signature $(S, \leq, \Sigma)$ is *regular* iff

- given $\sigma \in \Sigma_{w',s'}$ and $w_0 \leq w'$ there is a least rank $\langle w, s\rangle \in S^* \times S$ such that $w_0 \leq w$ and $\sigma \in \Sigma_{w,s}$.

An order-sorted signature $(S, \leq, \Sigma)$ is *pre-regular* iff

- given $\sigma \in \Sigma_{w',s'}$ and $w_0 \leq w'$ there is a least sort $s \in S$ such that $\sigma \in \Sigma_{w,s}$ for some $w$ with $w_0 \leq w$.

Regularity implies pre-regularity (Fact 2.4 in [GM92]). I guess that Maude checks only the pre-regularity, which ensures the existence of the least sort for any term.

**Fact 4.1.** *[GM92] Let $(S, \leq, \Sigma)$ be an order-sorted signature such that $(S, \leq)$ is* coNoetherian*, i.e., there is no strictly decreasing infinite chain $s_1 > s_2 > s_3 > \cdots$. Then $(S, \leq, \Sigma)$ is regular iff whenever $\sigma \in \Sigma_{w',s'} \cap \Sigma_{w'',s''}$ and $w_0 \leq w', w''$ then there is some $w \leq w', w''$ such that $\sigma \in \Sigma_{w,s}$ and $w_0 \leq w$.*

**Definition 4.4.** Let $(S, \leq, \Sigma)$ be an order-sorted signature. An $(S, \leq, \Sigma)$-*algebra* is a many-sorted $(S, \Sigma)$-algebra $M$ satisfying:

1. $s \leq s'$ implies $M_s \subseteq M_{s'}$;
2. $\sigma \in \Sigma_{w',s'} \cap \Sigma_{w'',s''}$ and $w' \leq w''$ implies $M_\sigma : M_{w'} \to M_{s'}$ equals $M_\sigma : M_{w''} \to M_{s''}$ on $M_{w'}$,

where if $w = s_1 \ldots s_n$ then $M_w = M_{s_1} \times \cdots \times M_{s_n}$.

I think that the "aggressive equation" on page 11 formalizes in fact the second condition in the above definition.

Dorel: Now we sketch out how a front-end grammar $G$ inductively defines an order-sorted signature $(S^{\text{osa}}, \leq, \Sigma^{\text{osa}})$. Consider first an example.

**Example 4.1.** Let $G_0$ denote the following front-end grammar:

```
sort Int
syntax Bool ::= true
syntax Bool ::= false
syntax Exp ::= Int
syntax Exp ::= Bool
syntax Exp ::= Exp "+" Exp [symbol("plus")]
syntax Int ::= Int "+" Int [symbol("plus")]
syntax Exp ::= Id "(" Exp ")"[symbol("funapp")]
syntax{V} List{V} ::= empty
syntax{V} List{V} ::= V
syntax{V} List{V} ::= cons(V,List{V})
syntax{K,V} Map{K,V} ::= "empty"
syntax{K,V} Map{K,V} ::= insert(K,V,Map{K,V})
```

**Example 4.2.** The OSA signature $(S^{\text{osa}}, \leq, \Sigma^{\text{osa}})$ defined by the grammar given in Example 4.1 is:

1. the set of sorts $S^{\mathrm{osa}}$:

   $\mathtt{Int}, \mathtt{Bool}, \mathtt{Exp}$
   $\mathtt{List\{Int\}}, \mathtt{List\{Bool\}}, \mathtt{List\{Exp\}},$
   $\mathtt{Map\{Int, Int\}}, \mathtt{Map\{Int, Bool\}}, \mathtt{Map\{Int, Exp\}},$
   $\mathtt{Map\{Bool, Bool\}}, \mathtt{Map\{Bool, Int\}}, \mathtt{Map\{Bool, Exp\}},$
   $\ldots$
   $\mathtt{List\{List\{Int\}\}}, \mathtt{List\{List\{Bool\}\}}, \mathtt{List\{List\{Exp\}\}},$
   $\mathtt{List\{Map\{Int, Int\}\}}, \mathtt{List\{Map\{Int, Bool\}\}}, \mathtt{List\{Map\{Int, Exp\}\}},$
   $\ldots$
   $\mathtt{Map\{List\{Int\}, List\{Int\}\}}, \mathtt{Map\{List\{Int\}, List\{Bool\}\}}, \mathtt{Map\{List\{Int\}, List\{Exp\}\}},$
   $\ldots$
   $\mathtt{Map\{Map\{Int, Int\}, Map\{Int, Int\}\}}, \mathtt{Map\{Map\{Int, Int\}, Map\{Int, Bool\}\}},$
   $\ldots$

2. the subsort relation $\leq$:

   $\mathtt{Int} \leq \mathtt{Exp}, \mathtt{Bool} \leq \mathtt{Exp},$
   $\mathtt{Int} \leq \mathtt{List\{Int\}}, \mathtt{List\{Int\}} \leq \mathtt{List\{Exp\}}, \mathtt{Bool} \leq \mathtt{List\{Bool\}}, \mathtt{List\{Bool\}} \leq \mathtt{List\{Exp\}},$
   $\mathtt{Exp} \leq \mathtt{List\{Exp\}},$
   $\mathtt{Int} \leq \mathtt{List\{Exp\}}, \mathtt{Bool} \leq \mathtt{List\{Exp\}},$
   $\mathtt{Map\{Int, Int\}} \leq \mathtt{Map\{Exp, Int\}}, \mathtt{Map\{Int, Int\}} \leq \mathtt{Map\{Int, Exp\}}, \mathtt{Map\{Int, Int\}} \leq \mathtt{Map\{Exp, Exp\}},$
   $\ldots$
   $\mathtt{List\{List\{Int\}\}} \leq \mathtt{List\{List\{Exp\}\}}, \mathtt{List\{List\{Bool\}\}} \leq \mathtt{List\{List\{Exp\}\}},$
   $\ldots$
   $\mathtt{Map\{List\{Int\}, List\{Int\}\}} \leq \mathtt{Map\{List\{Exp\}, List\{Int\}\}},$
   $\mathtt{Map\{List\{Int\}, List\{Int\}\}} \leq \mathtt{Map\{List\{Int\}, List\{Exp\}\}},$
   $\ldots$

3. the set of function symbols $\Sigma^{\mathrm{osa}}$:

   (a)

   $$\begin{aligned}
   &\Sigma^{\mathrm{osa}}_{w,\mathtt{Int}} = \emptyset && w \neq \mathtt{Int}\,\mathtt{Int},\\
   &\Sigma^{\mathrm{osa}}_{\mathtt{Int}\,\mathtt{Int},\mathtt{Int}} = \{\mathtt{plus}\},\\
   &\Sigma^{\mathrm{osa}}_{\varepsilon,\mathtt{Bool}} = \{\mathtt{false}, \mathtt{true}\},\\
   &\Sigma^{\mathrm{osa}}_{\mathtt{Exp}\,\mathtt{Exp},\mathtt{Exp}} = \{\mathtt{plus}\},\\
   &\Sigma^{\mathrm{osa}}_{\mathtt{Id}\,\mathtt{Exp},\mathtt{Exp}} = \{\mathtt{funapp}\}
   \end{aligned}$$

   The symbol `plus` is overloaded, but its definition satisfy both the monotonicity and the regularity conditions.

(b) The case of the production

$$\mathtt{syntax}\{V\}\ \mathtt{List}\{V\} ::= \text{"}\mathtt{empty}\text{"}$$

is problematic. If we consider a single overloaded `empty` symbol,

$$\Sigma^{\mathrm{osa}}_{\varepsilon,\mathtt{List\{Int\}}} = \Sigma^{\mathrm{osa}}_{\varepsilon,\mathtt{List\{Bool\}}} = \Sigma^{\mathrm{osa}}_{\varepsilon,\mathtt{List\{Exp\}}} = \cdots = \{\mathtt{empty}\}$$

then we loose the monotonicity. If we consider a distinguished symbol $\mathtt{empty}\{\theta(V)\}$ for each instance $\theta(V)$,

$$\begin{aligned}
&\Sigma^{\mathrm{osa}}_{\varepsilon,\mathtt{List\{Int\}}} = \{\mathtt{empty\{Int\}}\}\\
&\Sigma^{\mathrm{osa}}_{\varepsilon,\mathtt{List\{Bool\}}} = \{\mathtt{empty\{Bool\}}\}\\
&\Sigma^{\mathrm{osa}}_{\varepsilon,\mathtt{List\{Exp\}}} = \{\mathtt{empty\{Exp\}}\}\\
&\ldots
\end{aligned}$$

we have, e.g., three unrelated terms of sort `List{Exp}`: `empty{Int}`, `empty{Bool}`, `empty{Exp}`. In ML these term patterns are related with the corresponding instances of the "aggressive" axiom on page 11 (adapted for constants):

$$\begin{aligned}
&\mathtt{inj\{List\{Int\},List\{Exp\}\}(empty\{Int\})} = \mathtt{empty\{Exp\}()}\\
&\mathtt{inj\{List\{Bool\},List\{Exp\}\}(empty\{Bool\})} = \mathtt{empty\{Exp\}()}
\end{aligned}$$

Another solution to get a (monotonic) order-sorted signature is to forbid productions as above and use a new sort for the unique terminal:

$$\begin{aligned}
&\mathtt{syntax\ EmptyList} ::= \mathtt{empty}\\
&\mathtt{syntax}\{V\}\ \mathtt{List}\{V\} ::= \mathtt{EmptyList}
\end{aligned}$$

We get $\mathtt{EmptyList} \le \mathtt{List\{Int\}}$, $\mathtt{EmptyList} \le \mathtt{List\{Bool\}}$, $\mathtt{EmptyList} \le \mathtt{List\{Exp\}}$, $\mathtt{EmptyList} \le \mathtt{List\{List\{Int\}\}}$, ... .

(c) The symbol `cons` may be overloaded:

$$\Sigma^{\mathrm{osa}}_{\mathtt{Int\,List\{Int\},List\{Int\}}} = \Sigma^{\mathrm{osa}}_{\mathtt{Bool\,List\{Bool\},List\{Bool\}}} = \Sigma^{\mathrm{osa}}_{\mathtt{Exp\,List\{Exp\},List\{Exp\}}} = \cdots = \{\mathtt{cons}\}$$

and it satisfies the monotonicity and regularity conditions.

(d) The case of operations on $\mathtt{Map}\{K,V\}$ is similar to those of lists.

Here is a proposal for a formal definition for the front-end grammar.

**Definition 4.5.** 1. A *simple sort name* is just an identifier.

2. A *simple parameter name* is just an identifier.

3. A *parametric sort name* is an expression of the form $N\{P_1,\ldots,P_k\}$, where $N$ is a simple sort name and $P_i$ is a simple parameter name, $i = 1,\ldots,k$.

**Definition 4.6.** Let $P$ denote the set of parameter names $\bar{P} = \{P_1, \ldots, P_k\}$. A *$P$-sort-term* is inductively defined as follows:

- a simple sort name is a $\bar{P}$-sort-term;
- a parameter $P_i$ is a $\bar{P}$-sort-term;
- if $N\{Q_1, \ldots, Q_\ell\}$ is a parametric sort name and $N_1, \ldots, N_\ell$ are $\bar{P}$-sort-terms, then $N\{N_1, \ldots, N_\ell\}$ is a $\bar{P}$-sort-term.

Let $\mathbb{ST}[\bar{P}]$ denote the set of $\bar{P}$-sort-terms. A *ground sort-term* is a $\emptyset$-sort-name.

**Definition 4.7.** A *(parametric) production* is a production of the form

$$\texttt{syntax}\{P_1, \ldots, P_k\}\ N ::= T_1 N_1 \ldots T_n N_n T_{n+1}\ [\texttt{symbol}(\sigma)]$$

where:

1. $N$ is a simple sort name (identifier) or a parametric name $N'\{P_{i_1}, \ldots, P_{i_j}\}$, where $N'$ is a simple sort name and $\{P_{i_1}, \ldots, P_{i_j}\} \subseteq \{P_1, \ldots, P_k\}$;
2. $N_i$ is "" or a $\{P_1, \ldots, P_k\}$-sort-name, $i = 1, \ldots, n$;
3. $T_i$ is "" or a terminal, $i = 1, \ldots, n$;
4. $\sigma$ is the name of the associated symbol. The argument $[\texttt{symbol}(\sigma)]$ is optional, when it is missing the name of the associated symbol is predefined (we define later what exactly means that).

If $k = 0$, then we have a *non-parametric production* and it is written as

$$\texttt{syntax}\ N ::= T_1 N_1 \ldots T_n N_n T_{n+1}\ [\texttt{symbol}(\sigma)]$$

where $N$ is a simple sort name and $N_i$ is a ground sort-term, $i = 1, \ldots, n$.
If $n = 0$ then we have only a *sort declaration* and it is written as

$$\texttt{sort}\ N$$

**Definition 4.8.** Let $\bar{P}$ and $\bar{Q}$ two sets of parameters. A *sort substitution* is a function $\theta : \bar{P} \to \mathbb{ST}[\bar{Q}]$. The function $\Sigma$ is extended to $\mathbb{ST}[\bar{P}]$ as usual:

- if $N$ is a simple sort name, then $\theta(N) = N$;
- $\theta(N\{N_1, \ldots, N_\ell\}) = N\{\theta(N_1), \ldots, \theta(N_\ell)\}$.

A *ground sort substitution (parameter valuation)* is a substitution $\rho : \bar{P} \to \mathbb{ST}[\emptyset]$.

Whenever there is no confusion, we use substitution instead of sort substitution.

**Definition 4.9.** Let $G$ be a front-end grammar. The order-sorted signature $(S^{\text{osa}}, \leq, \Sigma^{\text{osa}})$ associated to $G$ is inductively defined as follows:

1. each non-parametric sort declaration

$$\texttt{sort}\ N$$

define a sort $N$ in $S^{\mathrm{osa}}$. Recal that $N$ is a simple sort name in this case.

2. each parametric sort declaration

$$\texttt{sort}\ N\{P_1, \ldots, P_k\}$$

together with each parameter valuation $\theta : \{P_1, \ldots, P_k\} \to S^{\mathrm{osa}}$ define a sort $\theta(N) = N\{\theta(P_1), \ldots, \theta(P_k)\}$ in $S^{\mathrm{osa}}$;

3. each non-parametric production $\pi$

$$\texttt{syntax}\ N ::= T_1 N_1 \ldots T_n N_n T_{n+1}\ [\texttt{symbol}(\sigma)]$$

where $N$ is a simple sort name, $N_i$ is a ground sort-term for $i = 1, \ldots, n$, and $n > 1$ or at least a $T_i$ is diferent from "", defines

   (a) a sort $N$ in $S^{\mathrm{osa}}$, provided that it is not already defined by another production, and

   (b) a symbol $\sigma$ in $\Sigma^{\mathrm{osa}}_{N_1 \ldots N_n, N}$.

4. each parametric production $\pi$

$$\texttt{syntax}\{P_1, \ldots, P_k\}\ N ::= T_1\ [\texttt{symbol}(\sigma)]$$

where $T_1$ is diferent from "", and each parameter valuation $\theta : \{P_1, \ldots, P_k\} \to S^{\mathrm{osa}}$ define

   (a) a sort $\theta(N) = N\{\theta(P_1), \ldots, \theta(P_k)\}$ in $S^{\mathrm{osa}}$, provided that it is not already defined by another production, and

   (b) a symbol $\sigma\{\theta(\bar{P})\}$ in $\Sigma^{\mathrm{osa}}_{\varepsilon, \theta(N)}$ (a distinguished symbol name for each $\theta$);

**Remark 4.2.** 1. I guess that we may have a distinguished name $\sigma\{\theta(\pi)\}$ only for those $\theta$ with $\theta(N)$ $\leq$-minimal.
2. The productions

$$\texttt{syntax}\{P_1, \ldots, P_k\}\ N ::= T_1 \ldots T_{n+1}$$

(only terminals in the rhs) are processed in a similar way.

5. each parametric production $\pi$

$$\texttt{syntax}\{P_1, \ldots, P_k\}\ N ::= T_1 N_1 \ldots T_n N_n T_{n+1}\ [\texttt{symbol}(\sigma)]$$

where at least a $N_i$ is different from "" and ($n > 1$ or at least a $T_i$ is diferent from ""), and each parameter valuation $\theta : \{P_1, \ldots, P_k\} \to S^{\mathrm{osa}}$ define

   (a) a sort $\theta(N) = N\{\theta(P_1), \ldots, \theta(P_k)\}$ in $S^{\text{osa}}$, provided that it is not already defined by another production, and

   (b) a symbol $\sigma$ in $\Sigma^{\text{osa}}_{\theta(N_1)\ldots\theta(N_n),\theta(N)}$;

6. each non-parametric production

$$\texttt{syntax}\ N ::= N_1$$

   where $N$ is a simple sort name and $N_1$ a ground sort-term, defines

   (a) a sort $N$ in $S^{\text{osa}}$, provided that it is not already defined by another production, and

   (b) a relation $N_1 \leq N$;

7. each parametric production

$$\texttt{syntax}\{P_1, \ldots, P_k\}\ N ::= N_1$$

   and each parameter valuation $\theta : \{P_1, \ldots, P_k\} \to S^{\text{osa}}$ define

   (a) a sort $\theta(N) = N\{\theta(P_1), \ldots, \theta(P_k)\}$ in $S^{\text{osa}}$, provided that it is not already defined by another production, and

   (b) a relation $\theta(N_1) \leq \theta(N)$;

8. each parametric production

$$\texttt{syntax}\{P_1, \ldots, P_k\}\ N ::= \ldots$$

   and each pair of parameter valuations $\theta, \theta' : \{P_1, \ldots, P_k\} \to S^{\text{osa}}$ such thar $\theta(P_i) \leq \theta'(P_i)$, for $i = 1, \ldots, n$, define

   (a) a relation $\theta(N) \leq \theta'(N)$.

9. $\leq$ is reflexive and transitive.

**Assumption 4.1.** Assume that $(S^{\text{osa}}, \leq)$ is a poset. This means that the input front-end grammar $G$ does not include cyclic subsort declarations.

**Lemma 4.1.** *Let $N$ be a $\bar{P}$-sort-term, where $\bar{P} = \{P_1, \ldots, P_k\}$ denotes the set of parameters occuring in $N$. If $\theta, \theta' : \bar{P} \to S^{\text{osa}}$ such that $\theta(N) \leq \theta'(N)$, then $\theta(P_i) \leq \theta(P'_i)$ for $i = 1, \ldots, n$.*

*Proof.* By structural induction on $N$. □

**Corollary 4.1.** *Let $N$ be a $\bar{P}$-sorted-term, where $\bar{P} = \{P_1, \ldots, P_k\}$ denotes the set of parameters occurring in $N$. If $\theta, \theta' : \bar{P} \to S^{\text{osa}}$ such that $\theta(N) = \theta'(N)$, then $\theta(P_i) = \theta(P'_i)$ for $i = 1, \ldots, n$.*

**Lemma 4.2.** *Let $N$ be $\bar{P}$-sorted-term and let $N'$ be $\bar{Q}$-sorted-term, where $\bar{P}$ denotes the set of parameters occurring in $N$ and $\bar{Q}$ denotes the set of parameters occurring in $N'$. If $\theta : \bar{P} \to S^{\mathrm{osa}}$ and $\theta' : \bar{Q} \to S^{\mathrm{osa}}$ such that $\theta(N) = \theta'(N')$, then there is a parameter rename $\rho : \bar{Q} \to \bar{P}$, possible non-injective, such that $\theta' = \theta \circ \rho$.*

*Proof.* By structural induction on $N$. TBC (To Be Checked). □

**Proposition 4.1.** *There are grammars $G$ such that the order-sorted signature $(S^{osa}, \leq, \Sigma^{osa})$ associated to $G$ does not satisfy the monotonicity condition.*

*Proof.* Such an example is given by the `cast` operator. Let $G$ be the following grammar:

$$\begin{array}{l}
\texttt{sort Int} \\
\texttt{sort Bool} \\
\texttt{syntax Exp} ::= \texttt{Int} \\
\texttt{syntax Exp} ::= \texttt{Bool} \\
\texttt{syntax}\{P_1, P_2\}\ P_2 ::= \texttt{cast}(P_1)
\end{array}$$

We have $\texttt{cast} \in \Sigma^{\mathrm{osa}}_{\texttt{Int},\texttt{Bool}} \cap \Sigma^{\mathrm{osa}}_{\texttt{Exp},\texttt{Int}}$, $\texttt{Int} \leq \texttt{Exp}$ and $\texttt{Bool} \not\leq \texttt{Int}$. □

**Question 4.1.** Is the *monotonicity* a property really needed for a ML theory with subsorting? We use parameters and injections to "encode" an order-sorted signature as a many-sorted signature. The `cast` operator is a parametric one and its instances clearly disambiguate ambiguous order-sorted terms. For instance, $\texttt{cast}\{\texttt{Int},\texttt{Bool}\}(\varphi : \texttt{Int})$ and $\texttt{cast}\{\texttt{Exp},\texttt{Bool}\}(\texttt{inj}\{\texttt{Int},\texttt{Exp}\}(\varphi : \texttt{Int}))$ are two disambiguations of the ambiguous order-sorted term $\texttt{cast}(\varphi : Int)$.

**Proposition 4.2.** *There are grammars $G$ such that the order-sorted signature $(S^{osa}, \leq, \Sigma^{osa})$ associated to $G$ is not regular.*

*Proof.* Obviously, the definition of `cast` is such an example. We have $\texttt{cast} \in \Sigma^{\mathrm{osa}}_{\texttt{Exp},\texttt{Exp}}$, $\texttt{Int} \leq \texttt{Exp}$, and the set $\{\langle s, s'\rangle \mid \texttt{Int} \leq s, \texttt{cast} \in \Sigma^{\mathrm{osa}}_{s,s'}\}$ has no a least rank; $\langle \texttt{Int}, \texttt{Int}\rangle$ and $\langle \texttt{Int}, \texttt{Bool}\rangle$ are two distinct minimal elements.

Another example is the following one:

$$\begin{array}{l}
\texttt{sort}\ N_1 \\
\texttt{sort}\ N_2 \\
\texttt{syntax}\ N_3 ::= N_1 \\
\texttt{syntax}\ N_4 ::= N_2 \\
\texttt{syntax}\ N_5 ::= \texttt{f}(N_1, N_4)\ [\texttt{symbol}(\texttt{"f"})] \\
\texttt{syntax}\ N_5 ::= \texttt{f}(N_3, N_2)\ [\texttt{symbol}(\texttt{'f"})]
\end{array}$$

We have $N_1\,N_2 \leq N_1\,N_4$, $N_1\,N_2 \leq N_3\,N_2$, and the set $\{\langle s_1 s_2, s\rangle \mid N_1\,N_2 \leq s_1 s_2, \texttt{f} \in \Sigma^{\mathrm{osa}}_{s_1 s_2, s}\}$ has no a least rank, because $N_1\,N_4$ and $N_3\,N_2$ are non-comparable □

**Question 4.2.** The same as that of monotonicity: Is the *regularity* a property needed? In ML we may express the ambiguity of the order-sorted term $\mathtt{f}(\varphi_1 : N_1, \varphi_2 : N_2)$ by $\mathtt{f}(\varphi_1 : N_1, \mathtt{inj}\{N_2, N_4\}(\varphi_2 : N_2)) \vee \mathtt{f}(\varphi_1 : \mathtt{inj}\{N_1, N_3\}(N_1), \varphi_2 : N_2)$.

**Definition 4.10.** Let $par(N)$ denote the set of parameters occurring in the sort-term $N$ (the set of parameters the sort $N$ depends on). A parametric production

$$\mathtt{syntax}\{P_1, \ldots, P_k\}\ N ::= T_1 N_1 \ldots T_n N_n T_{n+1}\ [\mathtt{symbol}(\sigma)]$$

is *well-typed* if $\bigcup_{i=1}^{n} par(N_i) = \{P_1, \ldots, P_k\}$.

**Lemma 4.3.** *Let $G$ be a front-end grammar such that all the overloaded symbols are given by well-typed parametric productions. The order-sorted signature $(S^{osa}, \leq, \Sigma^{osa})$ associated to $G$ satisfies the monotonicity condition.*

*Proof.* Let $\sigma \in \Sigma^{\mathrm{osa}}_{w',s'} \cap \Sigma^{\mathrm{osa}}_{w'',s''}$ with $w' \leq w''$. Since the only overloaded symbols are those corresponding to instances of a syntax production, it follows that there are a parametric production $\pi$

$$\mathtt{syntax}\{P_1, \ldots, P_k\} N ::= T_1 N_1 \ldots T_n N_n T_{n+1}$$

and parameter valuations $\theta', \theta'' : \{P_1, \ldots, P_k\} \to S^{\mathrm{osa}}$ such that $\sigma = \sigma_\pi$, $w' = \theta'(N_1) \ldots \theta'(N_n)$, $s' = \theta'(N)$, $w'' = \theta''(N_1) \ldots \theta''(N_n)$, and $s'' = \theta''(N)$. Note that $w' \neq \varepsilon \neq w''$ by the definition of $\Sigma^{\mathrm{osa}}$. From $\theta'(N_i) \leq \theta''(N_i)$, $i = 1, \ldots, n$, it follows $\theta'(P_i) \leq \theta''(P_i)$, $i = 1, \ldots, k$ by Lemma 4.1. Hence $s' = \theta'(N) \leq \theta''(N) = s''$, which finishes the proof. □

## 5 Next Steps

**1** Define an ML-theory $(S^{\mathrm{ml}}, \Sigma^{\mathrm{ml}}, F^{\mathrm{ml}})$. As Grigire sugested, this can done in two ways:

a) using non-parametric symbols

b) using parametric symbols

**2** Find the relationships between $(S^{\mathrm{osa}}, \leq, \Sigma^{\mathrm{osa}})$ and $(S^{\mathrm{ml}}, \Sigma^{\mathrm{ml}}, F^{\mathrm{ml}})$ (both versions).

**3** Identify the properties that define a "good front-end grammar".

**4** Develop algorithms for checking these properties.

## References

[GM92] Joseph A. Goguen and Jos Meseguer. 1992. *Order-sorted algebra I: equational deduction for multiple inheritance, overloading, exceptions and partial operations.* Theor. Comput. Sci. 105, 2 (November 1992), 217-273.