Translation of SML programs into Coq programs

The goal of this document is to define formally the translation of SML programs, represented by their abstract syntax tree (AST) into Coq programs, represented by Gallina's AST. Before defining the translation, we specify the two ASTs in question.

G: TODO: better introduction, mention the tools we used and motivation.

1 HaMLet

HaMLet [?] is an SML implementation whose goal is to be an accurate implementation of the language definition [?], and a platform for experimentation. There are two main reasons why we chose it for this work. First of all, it is faithful to SML's definition, and all bugs and "grey areas" and well documented. Secondly, it is easy to access various steps of the compilation process. Particularly, the abstract syntax tree from the elaboration phase was readily available.

In order to have a translation as precise as possible, we needed a few modifications in HaMLet. All changes were related to derived forms.Derived forms are SML expressions that can be replaced by equivalent forms during parsing, resulting in SML parse trees that are formed by only a subset of the constructs in syntax trees. For example, boolean operators can be replaced by the equivalent `if ... then ... else`:

G: check if this is still the case at the end.

$$\begin{array}{rcl} e_1 \ \texttt{orelse} \ e_2 & \Rightarrow & \texttt{if} \ e_1 \ \texttt{then true else} \ e_2 \\ e_1 \ \texttt{andalso} \ e_2 & \Rightarrow & \texttt{if} \ e_1 \ \texttt{then} \ e_2 \ \texttt{else false} \end{array}$$

HaMLet translates the derived forms as specified in SML's definition. Translating this subset of SML's syntax to Coq is correct – semantically speaking – however, the resulting Coq code will look very different: an `orelse` expression will be translated to an `if`... `then`... `else` expression. Therefore, we decided to keep the original grammatical form and translate that to Gallina.

G: explain the changes in more detail (see googledoc)

2 SML's abstract syntax tree

3 Gallina's abstract syntax tree