Toolchain for Pip's Gallina code

Towards a Certified Compilation

December 7th, 2018

Source

Pip's Gallina source code must be compiled to get a runnable kernel.

src.v

```
(** The [getPd] function returns the page directory
    of a given partition *)
Definition getPd partition :=
  perform idxPD := getPDidx in
  perform idx := MALInternal.Index.succ idxPD in
  readPhysical partition idx.
```

We need to prove that the properties we proved on the source are still valid for the compiled code.

Extraction

Coq provides a facility to extract computational code into OCaml or Haskell code.

Cons:

- Garbage collector vs Pip's memory management
- OCaml or Haskell runtime
- glue with ASM code

Current situation

$$\begin{tikzcd} \boxed{\texttt{src.v}} \arrow[r, bend right, "\text{Coq (extraction)}"'] & \boxed{\texttt{AST.json}} \arrow[r, bend right, "\text{digger}"'] & \boxed{\texttt{code.c}} \end{tikzcd}$$

code.c

```c
page getPd(page partition) {
    index idxPD = getPDidx();
    index idx = Index_succ(idxPD);
    return readPhysical(partition, idx);
}
```

Current situation (2)

Pros:

- no GC
- no runtime
- standard linking with ASM code

Cons:

- no proof: the properties proved on the `src.v` might not hold for `code.c`
- manual mapping (using `typedef` and `#define`) from elaborate Coq types to C types

Holy Grail

$$\begin{tikzcd}
\texttt{src.v} \arrow[r, bend right, "\text{Coq plugin or ...}"'] & \texttt{Coq-AST.v} \arrow[l, Rightarrow, bend right, "\text{checking consistency}"'] \arrow[r, Rightarrow, bend left, "\text{certified conversion}"] & \texttt{CompCertC-AST.v}
\end{tikzcd}$$