TPTP Tea Party: Pog2dk

Claude Stolze

Goals

- B is a certified programming language based on a set-inspired type theory
- Encode the B theory in Dedukti
  - Translate B proof obligations to Why3
  - Use Why3 to call automated provers (Zenon Modulo)
  - Get back a Dedukti proof
- My exemple file (00001.pog) has 22305 LOCs and 466 goals in 12 proof obligation blocks
- I can e.g. prove the latest goal using Alt-Ergo:

```
$ ./main.native -p Alt-Ergo -a 11 2 -i 00001.pog -o output.ae
$ alt-ergo output.ae
File "output.ae", line 4961, characters 16-136: Valid (0.2251) (318 steps) (goal goal_0)
```

For now, I cannot make Zenon Modulo work

B proof obligations

- Proofs are encoded in an XML format (.POG)
- It contains
  - definitions of types (structures or abstract types)
  - free variables (parameters)
  - definition blocks containing definitions of sets and axioms
  - proof obligation blocks containing goals, with a list of hypotheses

Proof obligations and their encodings in Why3

```
...<Define name="seext" hash="6300490539737942914">
    <Exp_Comparison op=":">
        <Id value="s804" typref="1" />
        <Id value="s745" typref="0" />
    </Exp_Comparison>
    <Exp_Comparison op=":">
        <Id value="s805" typref="1" />
        <Id value="s743" typref="0" />
    </Exp_Comparison>
    <Exp_Comparison op=":">
        <Id value="s806" typref="1" />
        <Id value="s741" typref="0" />
    </Exp_Comparison>
    <Exp_Comparison op=":">
        <Id value="s806" typref="1" />
        <Nary_Exp op="{" typref="0">
            <Integer_Literal value="1" typref="1" />
            <Integer_Literal value="2" typref="1" />
        </Nary_Exp>
    </Exp_Comparison> ...
</Define>
<Proof_Obligation goalHash="4467172256600396633">
    <Definition name="B definitions" />
    ...
    <Definition name="ass" />...
    <Simple_Goal>
        <Goal>
            <Exp_Comparison op="&gt;=i">
                <Integer_Literal value="0" typref="1" />
                <Integer_Literal value="0" typref="1" />
            </Exp_Comparison>
        </Goal>
    </Simple_Goal>
</Proof_Obligation>
```

```
...
constant anon_set_0_14 : int -> bool
axiom anon_set_0_14_pred =
  forall x:int. mem x anon_set_0_14 <-> x = 1 \/ x = 2
predicate seext =
  mem v_s804_1 v_s745_0 /\
  mem v_s805_1 v_s743_0 /\
  mem v_s806_1 v_s741_0 /\
  mem v_s806_1 anon_set_0_14 /\ ...
...
goal goal_0 :
  ass ->
  inv6 ->
  abs1 ->
  seext ->
  imext -> imprp -> imlprp -> aprp ->
  mchcst -> ctx -> b_def -> 0 >= 0
```