Esterel

Tools

Thew following tools are installed locally in /home/esterel/bin/

IDE

- [Esterel Studio](#)

Compiler

- [Inria Esterel Compiler](#) (esterel)
- [Columbia Esterel Compiler](#) (cec)
- [KEP](#) Kiel Esterel Processor

Managing Traces

Traces of Esterel programs are usually given as [ESI](#) or [ESO](#) files. The following tools to work with esi and eso files are installed in /home/esterel/bin

- eso2esi: removes all outputs from an eso file
- esoDiff: compares two eso files and gives the first trace they differ in, if they do
- esoPrint: pretty printer for eso Files

Additional tools

- v5tov7 is a script based on the CEC to convert Esterel V5 programs into Esterel V7

Semantic differences between v5 / v7 (incomplete)

Modules

In the Esterel v5, the semantic of the instantiation of modules is simply a textual copy and paste, with renaming of signals. In Esterel v7, the behavior is more subtle.

- emitting inputs across modules does not work in v7: define that input as output with the same name

Example:

```
    main module main_mod:
    input i, env_i; output o; run sub;
    loop await i do
        emit o
    end await end loop;
end module
module sub:
    input i,env_i; output i; change input i to output and it works fine sustain {
        i <= env_i
    }
end module
```

Similar, reading outputs does not work:

```
    main module M:
    input I; output O:int init 0; signal S: int init 0 in
        every I do
            run Count[S/C]; emit O(?S);
        end every
    end signal
end module
module Count:
    output C : int; emit C(sat<32>(pre(?C)+1));
end module
```

The global initialization does not reach C, hence it is not initialized when I occurs for the first time. Hence C is a local signal, when it is read, but it is a global signal when it is written. So you have to declare C as inputoutput;

New statements

- Esterel v7 allows the use of expressions as conditions for strong aborts, but this is somehow tricky. The expression is evaluated before the abort body is executed. When a variable is changed inside the body, making the abort condition true, no abort takes place, not even a weak one.

```
main module T:
    input I; output O : int;
    var v : int in
        v:=0; abort
            every I do v:=sat<32>(v+1); end
        when v=3; emit O(v);
    end var
end module
```

The O will be present in the instant after the third I occurred.

- It is now possible to emit signals in the next instant. Unfortunately, this will be silently omitted, if the module terminates in the current instant: main module M:

```
main module T:
    signal S in run sub[S/O]; await immediate S; halt; control never reaches this point. end signal
end module
module sub:
    output O : reg; emit next O; pause;
end module
```

If possible, you can simply add a pause at the end of the module, but of course this can change the overall behavior of the model.