Verified Lustre Normalization with Node Subsampling

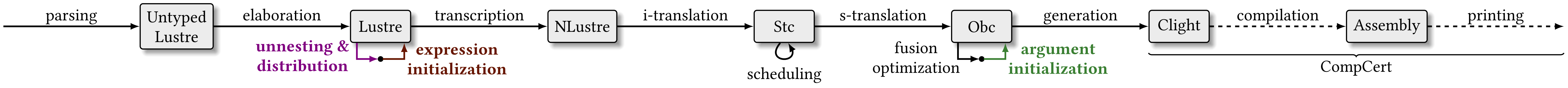
Timothy Bourke, Paul Jeanmaire, Basile Pesin, Marc Pouzet

Inria Parkas team, École Normale Supérieure, PSL University

EMSOFT — Oct. 8-15, 2021

VÉLUS

https://velus.inria.fr


parsing

Untyped Lustre

elaboration

Lustre

transcription

NLustre

i-translation

Stc

s-translation

Obc

generation

Clight

compilation

Assembly

printing

unnesting & distribution

expression initialization

scheduling

fusion optimization

argument initialization

CompCert

Unnesting & distribution

- Place certain operators in their own equations
- Distribute operators over lists
- Fresh name generation with a state monad
- Successive refinements to handle new variables

Expression initialization

- Make initialization of delays explicit
- Simplify later transformations
- Optimize to avoid redundant registers
- Build new streams using an alignment property

```
time = current(0, ck, count_down((res, 1) when ck));
```

```
if (ck) { elab$4 := count_down(i0).step(res, 1) };
time := current(i1).step(0, ck, elab$4)
```

argument initialization

```
if (ck) { elab$4 := count_down(i0).step(res, 1) }
else { elab$4 := 0 };
time := current(i1).step(0, ck, elab$4)
```

```
node count_down(res : bool; n : int)
returns (cpt : int)
let
  cpt = if res then n else (n fby (cpt − 1));
tel
```

unnesting & distribution

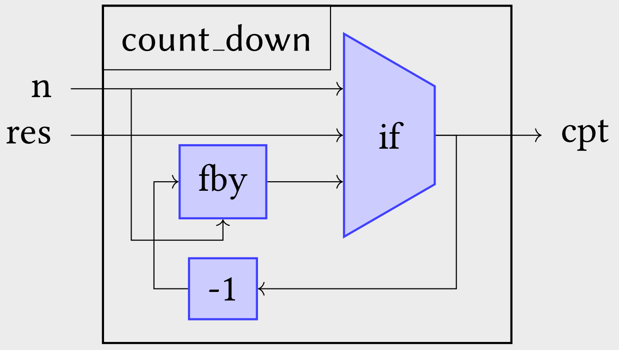
```
node count_down(res : bool; n : int)
returns (cpt : int)
var norm1$1 : int;
let
  norm1$1 = n fby (cpt − 1);
  cpt = if res then n else norm1$1;
tel
```

expression initialization

Argument initialization & node subsampling

- Allows some inputs to be slower than others
- In C99/Clight, all arguments of a function call must be well-defined
- Add default values for slow streams and prove that arguments are then always well-defined

Lustre, a dataflow synchronous language



res	F	T	F	F	F	F	T	F	…
n	3	3	3	3	3	3	3	3	…
cpt	3	3	2	1	0	-1	3	2	…

$$\textsc{variable}\ \frac{H(x) = s}{G, H, bs \vdash x \Downarrow [s]} \qquad \textsc{equation}\ \frac{G, H, bs \vdash \boldsymbol{e} \Downarrow H(\boldsymbol{x})}{G, H, bs \vdash \boldsymbol{x} = \boldsymbol{e}}$$

$$\textsc{node}\ \frac{\begin{array}{c}\texttt{node}(G, f) = n \qquad H(n.\mathbf{in}) = \boldsymbol{xs} \\ H(n.\mathbf{out}) = \boldsymbol{ys} \qquad \forall eq \in n.\mathbf{eqs},\ G, H, (\texttt{base-of}\ \boldsymbol{xs}) \vdash eq\end{array}}{G \vdash f(\boldsymbol{xs}) \Downarrow \boldsymbol{ys}}$$

Formal semantics parameterized by $H : ident \rightarrow stream$

```
node count_down(res : bool; n : int)
returns (cpt : int)
var norm1$1, norm2$2 : int; norm2$1 : bool;
let
  norm2$1 = true fby false;
  norm2$2 = 0 fby (cpt − 1);
  norm1$1 = if norm2$1 then n else norm2$2;
  cpt = if res then n else norm1$1;
tel
```

ENS | PSL | Inria | ANR — Projet financé par l'ANR - Project funded by the ANR