Projet Minilucy

Paul Patault & Émilien Lemaire

December 14, 2022

ENS Paris-Saclay

Sommaire

idée générale

when

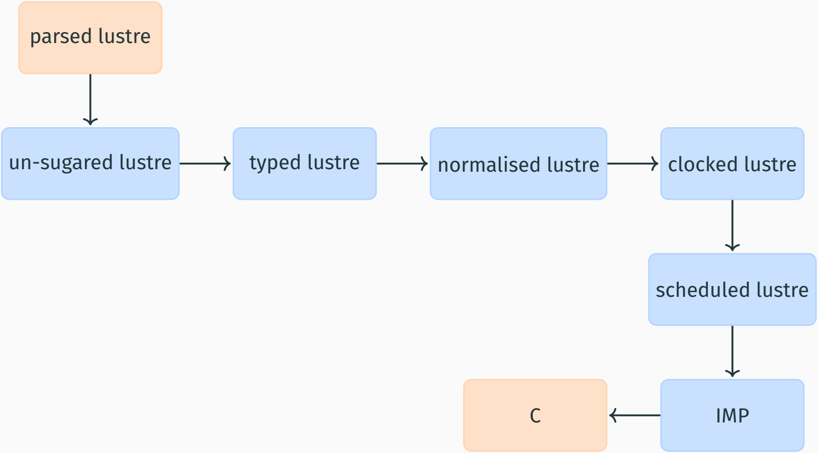
automate

merge

mémoire

reset

Schéma de compilation



Features réalisées

- `when`
- `merge`
- `reset`
- `automates` (en surface uniquement)

Sommaire

idée générale

when

automate

merge

mémoire

reset

Vérification des horloges

```
node clock_error (c: bool) returns (o: int);
  var x: int;
let
  x = 0 when True(c);
  o = x;
tel
```

```
Clocking error: The expected clock is Base, got Base on True(c)
```

Sommaire

idée générale

when

automate

merge

mémoire

reset

Traduction de l’automate

```
node syracuse (i: int) returns (o: int);
let
  automaton
  | Even ->
      o = i -> pre o / 2;
      until (o mod 2 = 1) continue Odd
  | Odd ->
      o = i -> pre o * 3 + 1;
      until (o mod 2 = 0) continue Even
  end
tel
```

Traduction de l’automate

```
type t = Even | Odd

node syracuse (i: int) returns (o: int);
    var state: t; cond__4: bool; cond__3: bool;
let
  cond__4 = o mod 2 = 0;
  cond__3 = o mod 2 = 1;
  state = Even ->
           pre (merge state
                 (Even -> if cond__3 then Odd else Even)
                 (Odd -> if cond__4 then Even else Odd));
  o = merge state
        (Even -> i -> pre o / 2)
        (Odd  -> i -> pre o * 3 + 1);
tel
```

Automate (une slide)

```
node syracuse (i: int)
  returns (o: int);
let
  automaton
  | Even ->
      o = i -> pre o / 2;
      until (o mod 2 = 1)
      continue Odd
  | Odd ->
      o = i -> pre o * 3 + 1;
      until (o mod 2 = 0)
      continue Even
  end
tel
```

```
type t = Even | Odd

node syracuse(i: int)
  returns (o: int);
  var state: t;
      cond__4, cond__3: bool;
let
  cond__4 = o mod 2 = 0;
  cond__3 = o mod 2 = 1;
  state = Even ->
           pre (merge state
                 (Even ->
                   if cond__3 then Odd else Even)
                 (Odd ->
                   if cond__4 then Even else Odd));
  o = merge state
        (Even -> i -> pre o / 2)
        (Odd  -> i -> pre o * 3 + 1);
tel
```

Sommaire

idée générale

when

automate

merge

mémoire

reset

Compilation des « merge »

```
type t = A | B

node main0 () returns (o: t);
let
  o = A't fby
        (merge o
          (A't -> B't when A(o))
          (B't -> A't when B(o)));
tel
```

Compilation des « merge »

```
enum t { A, B };
// ...
enum t main0(...) {
    // ...
    switch (o) {
        case A: {
            res = B;
            break;
        }
        case B: {
            res = A;
            break;
        }
    };
    // ...
    return res;
}
```

Compilation des « merge »

```
type t = A | B

node main0 () returns (o: t);
let
  o = A't fby
        (merge o
          (A't -> B't when A(o))
          (B't -> A't when B(o)));
tel
```

```
enum t { A, B };
// ...
enum t main0(...) {
    // ...
    switch (o) {
        case A: {
            res = B;
            break;
        }
        case B: {
            res = A;
            break;
        }
    };
    // ...
    return res;
}
```

Sommaire

- idée générale
- when
- automate
- merge
- mémoire
- reset

Gestion de la mémoire

on créé une mémoire pour chaque nœud si :

- présence d'un `fby` dans le nœud
- appel d'un nœud qui a une mémoire
- présence d'un automate

Gestion de la mémoire

```
node f () returns (o:int);
let
  o = 1;
tel

node main0 () returns (o:int);
let
  o = f();
tel
```

```
node f () returns (o:int);
let
  o = 1 fby 2;
tel

node main0 () returns (o:int);
let
  o = f();
tel
```

Gestion de la mémoire

```c
struct f_mem {
  int o;
};
struct main0_mem {
  struct f_mem f_next;
};
//...
void f_init (struct f_mem* mem) {
  mem->o = 1;
}
//...
void main0_init (struct main0_mem* mem) {
  f_init(&(mem->f_next));
}
//...
```

Gestion de la mémoire

```
node f () returns (o:int);
let
  o = 1 fby 2;
tel

node main0 () returns (o:int);
let
  o = f();
tel
```

```
struct f_mem {
  int o;
};
struct main0_mem {
  struct f_mem f_next;
};
//...
void f_init (struct f_mem* mem) {
  mem->o = 1;
}
//...
void main0_init (struct main0_mem* mem) {
  f_init(&(mem->f_next));
}
//...
```

Sommaire

idée générale

when

automate

merge

mémoire

reset

Compilation des « reset »

```
node incr () returns (cpt: int);
let
  cpt = 0 fby cpt + 1;
tel

node main0 (i: bool) returns (o: int);
let
  o = reset incr() every i;
tel
```

→ ajouter un appel à `incr_init()` quand `i` est vrai

Démo !