```
$ENTRY Go {
  e.X = <Rev ε, e.X>;
}
Rev {
  e.A, s.X e.y = <Rev s.X e.A, e.y>;
  e.A, ε = e.A;
}
```

Так будем получать опасную похожесть

① <Go e.0>
↓
<Rev ε, e.0>
e.0→s.1 e.2 ↘ e.0→ε
<Rev s.1, e.2>

② <Go e0>
↓
<Rev ε, e.0>
↓ ε←e.1
<Rev e.1, e.0> ↘ e.0→ε
e.0→s.2 e.3 ↑ s.2 e.1←e.1, e.3←e.0
e.1
<Rev s.2 e.1, e.3>
получился привычный результат

③ А теперь - ФОКУС!

Попробуем заморозить аккумулятор #e.1

③ - фокус

<Go e0>
↓
<Rev ε, e0>
↓ ε←e.1
<Rev #e.1, e.0> : ⊥
e.0→s.2 e.3 ↙ ↘ e.0→ε
#e.1 : ⊥ ⊗

<Rev s.2 #e.1, e.3> : ⊥
Пока (спекулируя!) не будем ничего тут рассматривать, проигнорируем эту конфигурацию

здесь формат нарушился, обобщаем:
#e.1 ⊓ ⊥ = #e.1

Параметры времени компиляции и времени выполнения в C++:

```
template <int x>
int add(int y){
  return x+y;
}

add<2>(3);
```

параметры с решёткой можно уподобить шаблонным параметрам

(4) - продолжаем фокус.

<Go e.0>

↓

<Rev ε, e.0>

↓ c ← e.1

F1

<Rev #e.1, e.0> : #e.1

e.0 → s.2 e.3

<Rev s.2 #e.1, e.3> : #e.1

конфигурация опасно похожа на родительскую. Что делать будем? Сопоставляем с родительской:

s.2 #e.1 ← #e.1

e.3 ← e.0

<Rev s.2 #e.1, e.3>

↪ s.2 #e.1 : #e.1

формат говорит о том, что функция после расформатирования ничего не возвращает. Константа #e.1 - часть и входного, и выходного формата

не очевидно, как поступать? А это и есть фокус!

Константе #e.1 в родительской конфигурации соответствует выражение s.2 #e.1

Переместим в родительскую - добавим штрих

<Rev #e.1', e.0> : #e.1'

#e.1' $\stackrel{def}{\equiv}$ s.2 #e.1

Из гипотезы формата следует, что первый аргумент как есть пойдет в результат, значит, результатом дочерней конфигурации будет

<Rev #e.1', e.0> : #e.1'

(s.2 #e.1, e.3, s.2 #e.1)

<Rev s.2 #e.1, e.> == s.2 #e.1

Результат s.2 #e.1 $\not\leq$ #e.1, значит, обобщаем форматную гипотезу

s.2 #e.1 $\sqcap$ #e.1 = e.2 #e.1

⑤ – продолжаем или завершаем фокус

<Go e.0>

↓

<Rev e, e.0>

↓ e ← e.1

F1

<Rev #e.1, e.0> : e.2 #e.1

Формат говорит о том, что функция после расформатирования вернёт e.2

Если обозначим рассформатированную конфигурацию, как F1, то её "формат" будет <F1 e.0> : e.2.

e.0 → s.3 e.4

#e.1'

<Rev s.3 #e.1, e.4>

Сопоставляем с родительским форматом:

<Rev s.3 #e.1, e.4> : [<Rev #e.1', e.0'> : e.2 #e.1']

{ s.3 #e.1 ← #e.1'
  e.4 ← e.0'

Результат будет

e.2' s.3 #e.1,

сопоставляем с форматом:

e.2' s.3 #e.1 : e.2 #e.1

Приходим к выводу, что функция возвращает e.2' s.3 ← e.2

~~Рекурсивно~~

Рекурсивно вызываем конфигурацию F2:

<F1 e.4> ← e.5,

получаем формат и тождество; она вернёт e.5 #e.1', где #e.1' ≡ s.3 #e.1,

т.е. e.5 s.3 #e.1.

Сопоставляем с ожидаемым форматом

e.5 s.3 #e.1 : e.2 #e.1,

получается, что как e.2 вернётся e.5 s.3 или <F1 e.4> s.3

e.0 → ε

#e.1 : e.2 #e.1,

e.2 = ε,

т.е. здесь

<F1 ε> вернёт ε

Остаточная программа примет вид

$ENTRY Go {
  e.0 = <F1 e.0> ε;
}

F1 {
  s.3 e.4 = <F1 e.4> s.3;
  ε = ε;
}

конкатенация ε с ε формальна, выходит из формата e.2 #e.1, где ε ← e.1

Отличия в терминах функций форматирования и расформатирования. <Rev #e.1,e.0> : e.2 #e.1 – в обозначениях со слайдов

```
unfmt{
  #e.1, e.2 #e.1 = e.2;
}
```

повёрнуть можно было не ставить

```
fmt{
  e.1, e.2 = e.2 e.1;
}
```

<Go e.0>

↓

<Rev ε, e.0>

↓ ε ← e.1 ←

<Rev #e.1, e.0>

↓ (тождество)

<fmt#e.1, <unfmt #e.1, <Rev #e.1, e.0>>>

↓

let <unfmt #e.1, <Rev #e.1, e.0>> ← e.2
in <fmt #e.1, e.2>

in

<fmt e.1, e.2>

↓

e.2 e.1

let

F1

<unfmt #e.1, <Rev #e.1, e.0>> —e.0→ε→ <unfmt #e.1, #e.1>

↓

ε

e.0 → s.3 e.4

```
<unfmt#e.1,
  <Rev s.3#e.1, e.4>
>
```

↓ (тождество)

```
<unfmt
  #e.1,
  <fmt s.3 #e1,
    <unfmt
      s.3 #e.1,
      <Rev s.3 #e.1, e.4>
    >
  >
>
```

↓ (раскрываем fmt)

```
<unfmt
  #e.1,
  <unfmt
    s.3 #e.1,
    <Rev s.3 #e.1, e.4>
  >
  s.3 #e.1
>
```

↓ (раскрываем внешний unfmt)

<unfmt s.3#e.1, <Rev s.3 #e.1, e.4>> s.3

↓

let <unfmt s.3 # e.1, <Rev s.3 #e1, e.4> ← e.5
in e.5 s.3

↓ let

<unfmt s.3 #e.1, <Rev s.3 #e1, e.4>>

in

e.5 s.3

s.3 #e.1 ← #e.1 = игнорируем при построении остаточной программы

e.4 ← e.0

Остаточная программа та же:

```
$ENTRY Go{
  e.0 = <F1 e.0> ε;
}
F1{
  s.3 e.4 = <F1 e.4> s.3;
  ε = ε;
}
```