Probabilistic Resource Analysis

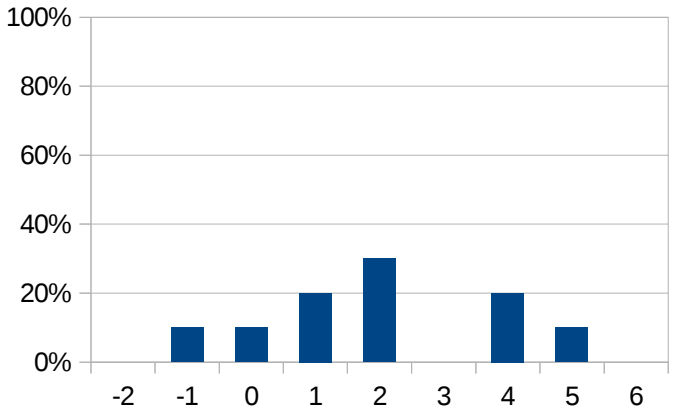
Entra Workshop, Malaga, May 6th 2015

Maja H Kirkeby & Mads Rosendahl

Roskilde University, Denmark

Probabilistic Resource Analysis

Input distribution



Program in XC

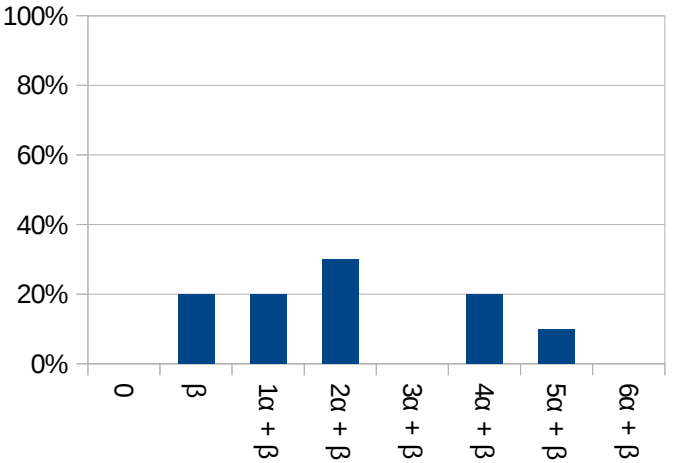
```
int fac(int i){
  if(i<=0) return 1;
  return i*fac(i-1);
}
```

Resource model

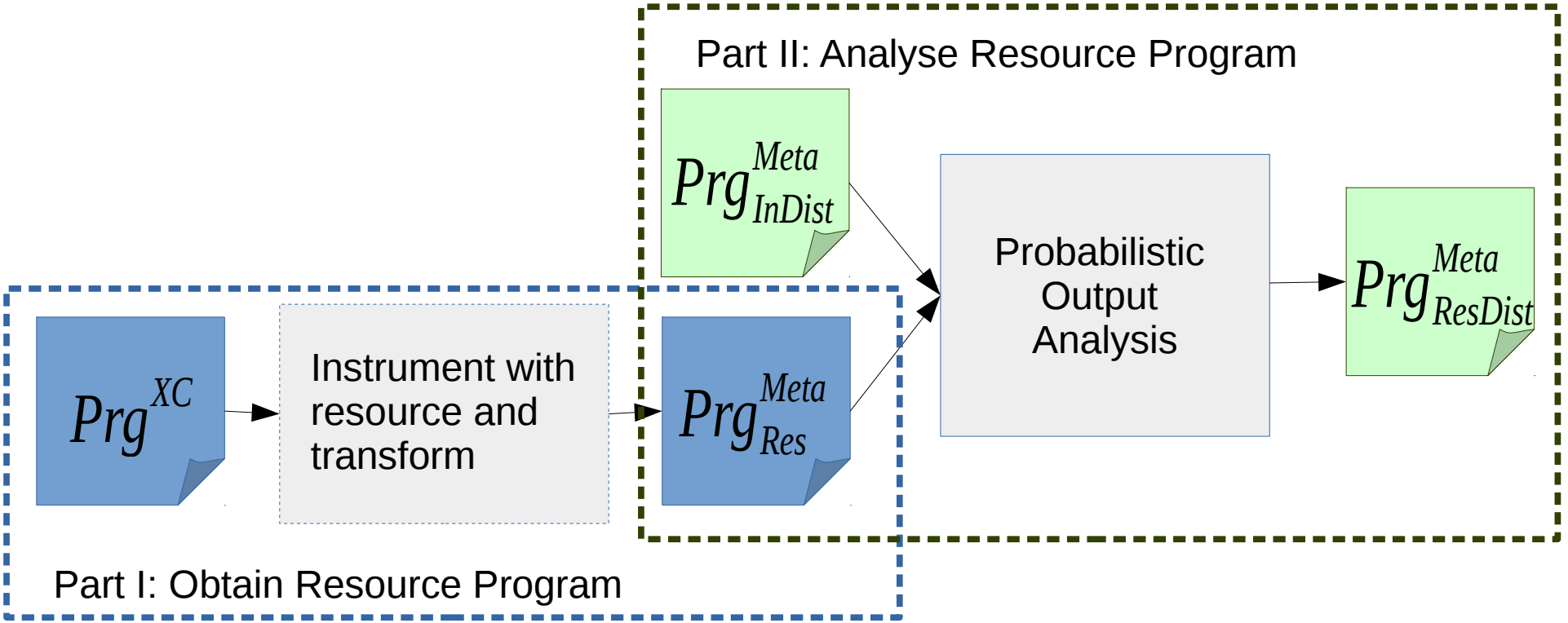
```
if = c1
Return = c2
...
```

Probabilistic Resource Analysis

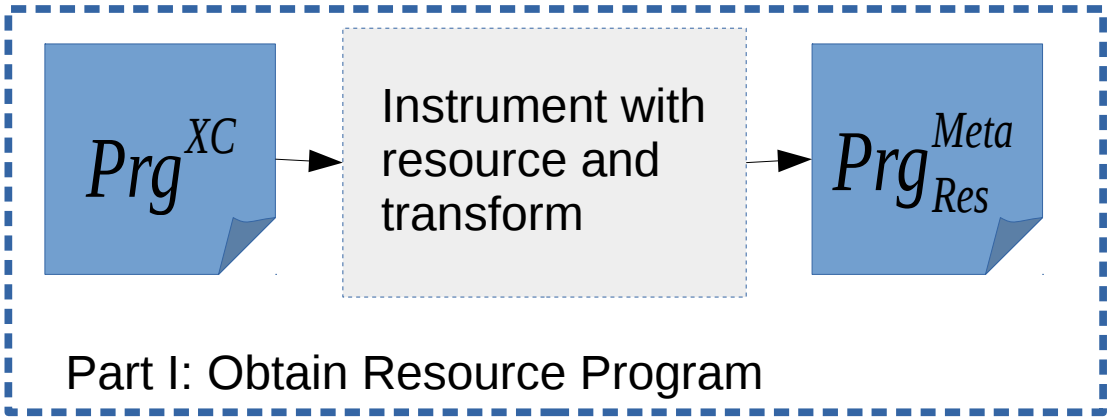
Resource distribution



Probabilistic Resource Analysis



Obtain Resource Program



org. XC

```
int fac(int i){
  if(i<=0) return 1;
  return i*fac(i-1);
}
```

res. Meta

```
facm(i,res) =
 if i≤0 then res+c1
 else facm(i-1,res+c2)
```

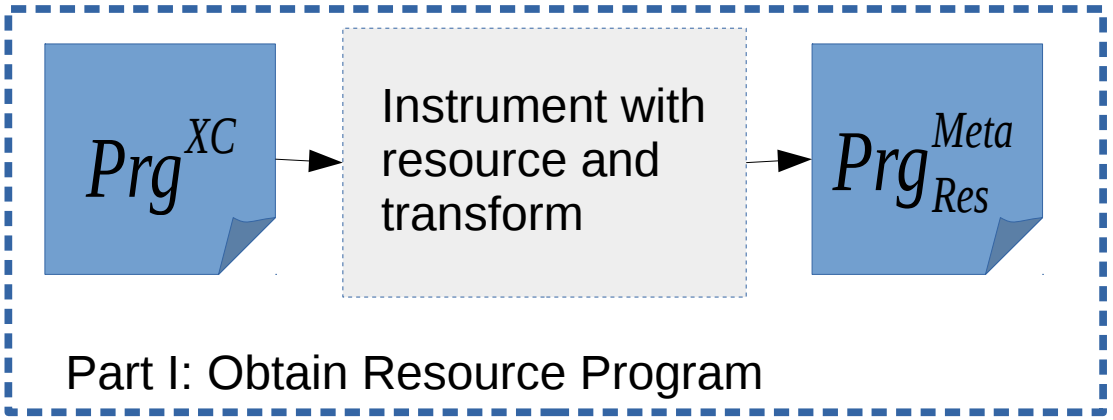
Instr. w. res

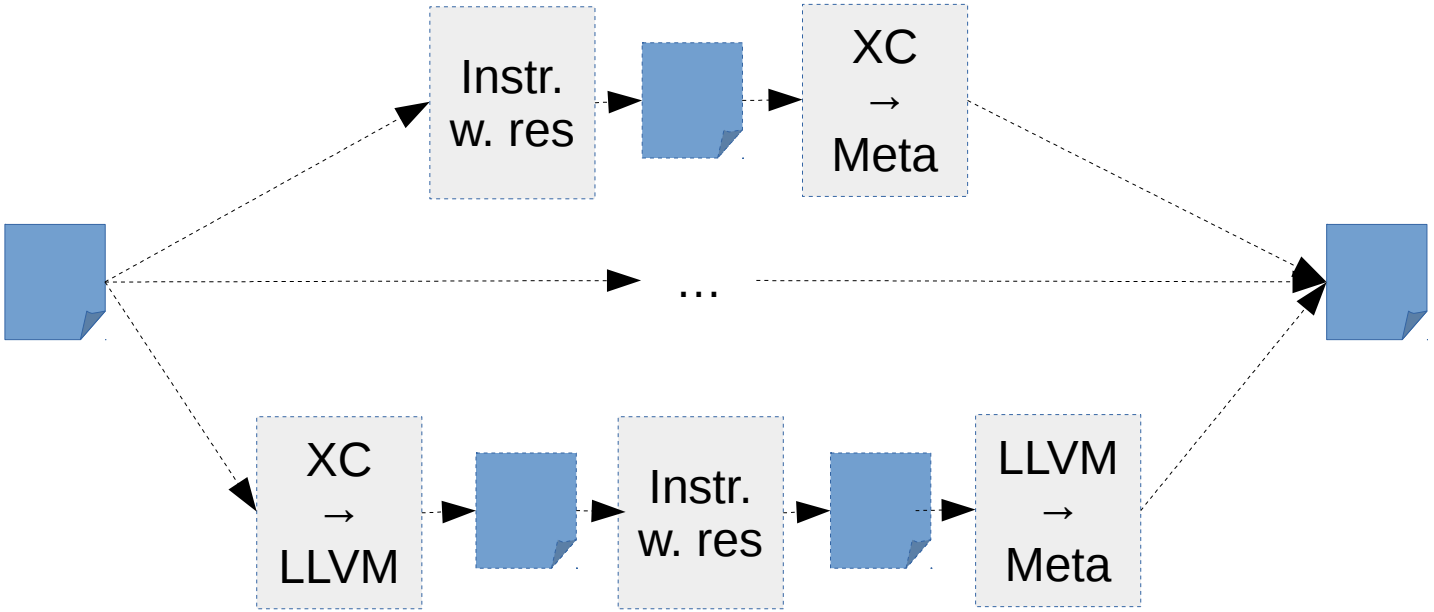
res. XC

```
int facr(int i, int r){
  if(i<=0) return r+c1;
  return facr(i-1,r+c2);
}
```

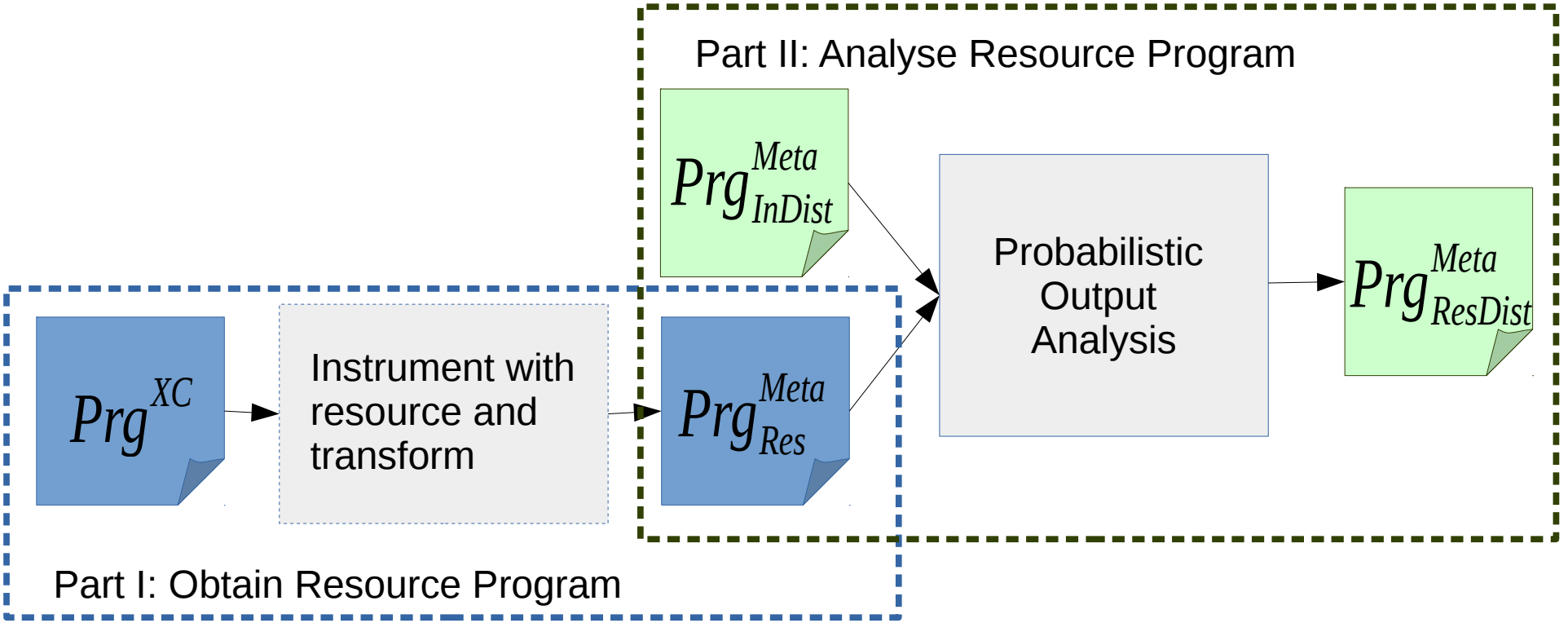
XC → Meta

One problem – More solutions

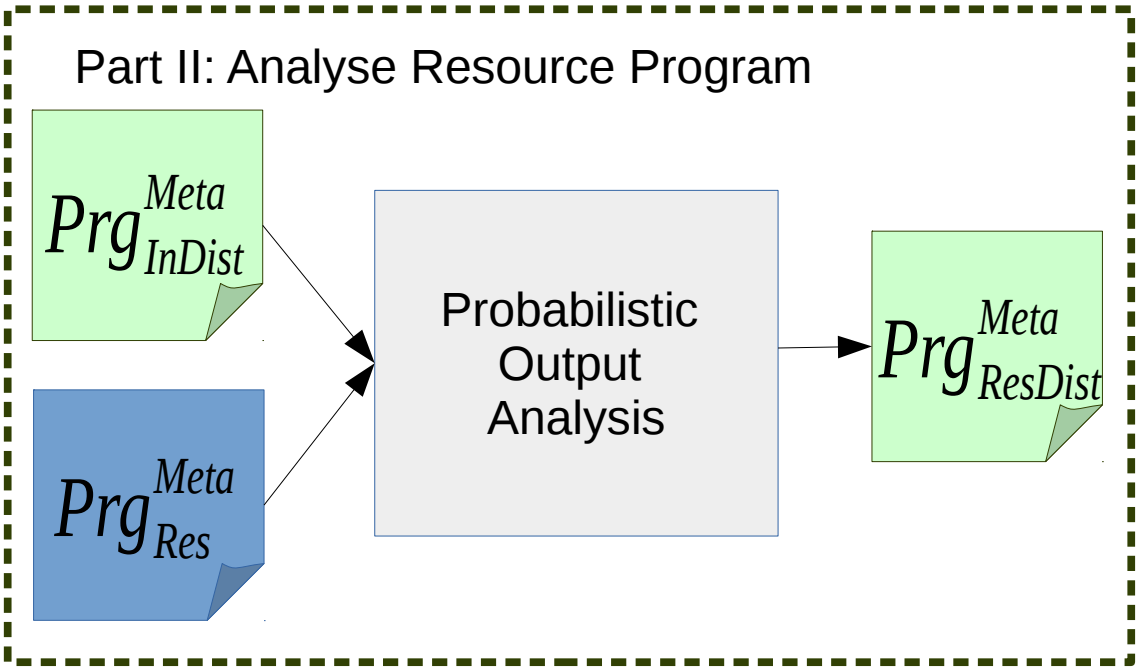




Probabilistic Resource Analysis



Analyse resource program



Meta language:

- Easy to translate simple xc programs into this language
- Simple functional first-order recursive language (f:int→int, P:int→real)
- Probabilities can be described using symbols (e.g. numbers from 1 to n is 1/n)
- Restrictions on function calls (no mutual recursive calls + primitive recursion -no mix of variables in recursive call)

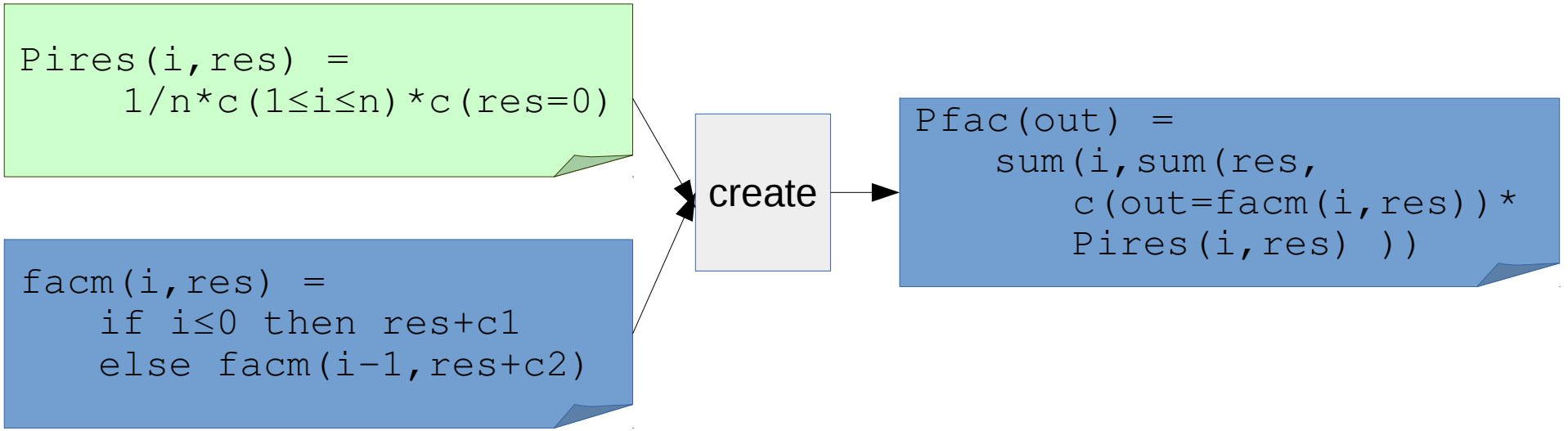
The 3 steps

- **create**: form an expression describing the output distribution using the input probability distribution and the original program

- **unfold calls**: remove all calls to the original program and obtain a pure probability distribution

- **symbolic summation and approximation**: rework the program using approximation into a probability distribution in closed form

Step I: create



$$\text{create}\ \frac{\mathtt{f}(\mathtt{u}_1,\ldots,\mathtt{u}_n) \stackrel{\text{def}}{=} e \qquad \mathtt{P}(\mathtt{v}_1,\ldots,\mathtt{v}_n) \stackrel{\text{def}}{=} e_p}{P_{\mathtt{f}}(\mathtt{z}) \stackrel{\text{def}}{=} \mathtt{sum}(\mathtt{x}_1;\ldots\mathtt{sum}(\mathtt{x}_n;\mathtt{c}(\mathtt{z}=_{\mathtt{i}}\mathtt{f}(\mathtt{x}_1,\ldots,\mathtt{x}_n))\times_{\mathtt{q}}\mathtt{P}(\mathtt{x}_1,\ldots,\mathtt{x}_n)))}$$

Step II: unfold

```
Pfac(out) =
    sum(i,sum(res,
        c(out=facm(i,res))*
        Pires(i,res) ))
```

unfold

```
Pfac(out) =
    sum(i,
     sum(res,
      sum(a,
           c(out=res+a*c2+c1)*
           c(i-a≤0)*
           prod(b,
                c(1≤b)*c(b≤a-1),
                c(i-b>0))
      )*
        1/n*c(1≤i≤n)*c(res=0)
    ))
```

Rules in this step:

- move nested calls
- remove if-expressions
- remove simple calls
- remove recursive calls

Example rules:

$$\text{f-simple}\ \frac{\mathtt{f}(\mathtt{x}_1,\ldots,\mathtt{x}_n) \stackrel{\text{def}}{=} e \qquad e \text{ is non-recursive} \qquad e_1,\ldots,e_n \text{ contain no } \mathtt{f}\text{-calls}}{\mathtt{c}(w =_{\mathtt{i}} \mathtt{f}(e_1,\ldots,e_n)) \rightarrow \mathtt{c}(w =_{\mathtt{i}} e[\mathtt{x}_1/e_1,\ldots,\mathtt{x}_n/e_n])}$$

$$\text{rem-if}\ \frac{}{\mathtt{c}(\mathtt{z} =_{\mathtt{i}} \mathtt{if}\ b\ \mathtt{then}\ e_0\ \mathtt{else}\ e_1) \rightarrow \left(\mathtt{c}(b) \times_{\mathtt{q}} \mathtt{c}(\mathtt{z} =_{\mathtt{i}} e_0) +_{\mathtt{q}} \mathtt{c}(\mathtt{not}(b)) \times_{\mathtt{q}} \mathtt{c}(\mathtt{z} =_{\mathtt{i}} e_1)\right)}$$

Step III: Symbolic summation & approximation

```
Pfac(out) =
    sum(i,
     sum(res,
      sum(a,
           c(out=res+a*c2+c1)*
           c(i-a≤0)*
           prod(b,
                c(1≤b)*c(b≤a-1),
                c(i-b>0))
      )*
       1/n*c(1≤i≤n)*c(res=0)
    ))
```

→ ≈ & →

```
Pfac(out) =
    1/n*min(n,(out-c1)/c2)
        *c(1≤(out-c1)/c2)
```

This step contains around 20 rules and integrates with Mathematica.

Example rules:

$$\text{rem-sum(c)}\ \frac{x \notin \mathcal{V}ar(e_1, e_2)}{\mathtt{sum}(x\,;\mathtt{c}(e_1 \leq_\mathtt{i} x) \times_\mathtt{q} \mathtt{c}(x \leq_\mathtt{i} e_2)) \to \mathtt{i2r}(e_1 -_\mathtt{i} e_2 +_\mathtt{i} 1) \times_\mathtt{q} \mathtt{c}(e_1 \leq_\mathtt{i} e_2)}$$

$$\text{rem-prod-one}\ \frac{x \notin \mathcal{V}ar(e_1, e_2) \qquad x \in \mathcal{V}ar(e_3)}{\mathtt{prod}(x\,;\mathtt{c}(e_1 \leq_\mathtt{i} x) \times_\mathtt{q} \mathtt{c}(x \leq_\mathtt{i} e_2)\,;c(e_3)) \to 1}$$

Example Rules

intro-min
$$\frac{\mathtt{x} \in \mathcal{V}ar(e_i, e_i') \;\; \mathtt{iso}(\mathtt{c}(e_i \leq_\mathtt{i} e_i'), \mathtt{x}) \to \mathtt{iso}(\mathtt{c}(\mathtt{x} \leq_\mathtt{i} e_i''), \mathtt{x}) \;\; \mathtt{x} \notin \mathcal{V}ar(e_i'') \;\; i \in \{1,2\}}{\mathtt{sum}(\mathtt{x}\,; \mathtt{c}(e_1 \leq_\mathtt{i} e_1') \times_\mathtt{q} \mathtt{c}(e_2 \leq_\mathtt{i} e_2') \times_\mathtt{q} e) \to \mathtt{sum}(\mathtt{x}\,; \mathtt{c}(\mathtt{x} \leq_\mathtt{i} \mathtt{min}(e_1'', e_2'')) \times_\mathtt{q} e)}$$

intro-max
$$\frac{\mathtt{x} \in \mathcal{V}ar(e_i, e_i') \;\; \mathtt{iso}(\mathtt{c}(e_i \leq_\mathtt{i} e_i'), \mathtt{x}) \to \mathtt{iso}(\mathtt{c}(e_i'' \leq_\mathtt{i} \mathtt{x}), \mathtt{x}) \;\; \mathtt{x} \notin \mathcal{V}ar(e_i'') \;\; i \in \{1,2\}}{\mathtt{sum}(\mathtt{x}\,; \mathtt{c}(e_1 \leq_\mathtt{i} e_1') \times_\mathtt{q} \mathtt{c}(e_2 \leq_\mathtt{i} e_2') \times_\mathtt{q} e) \to \mathtt{sum}(\mathtt{x}\,; \mathtt{c}(\mathtt{max}(e_1'', e_2'') \leq_\mathtt{i} \mathtt{x}) \times_\mathtt{q} e)}$$

rem-sum(c)
$$\frac{x \notin \mathcal{V}ar(e_1, e_2)}{\mathtt{sum}(x\,; \mathtt{c}(e_1 \leq_\mathtt{i} x) \times_\mathtt{q} \mathtt{c}(x \leq_\mathtt{i} e_2)) \to \mathtt{i2r}(e_1 -_\mathtt{i} e_2 +_\mathtt{i} 1) \times_\mathtt{q} \mathtt{c}(e_1 \leq_\mathtt{i} e_2)}$$

rem-sum(cx+d)
$$\frac{x \notin \mathcal{V}ar(e_1, e_2, e_3, e_4)}{\begin{array}{c}\mathtt{sum}(x\,; \mathtt{c}(e_1 \leq_\mathtt{i} x) \times_\mathtt{q} \mathtt{c}(x \leq_\mathtt{i} e_2) \times_\mathtt{q} \mathtt{i2r}(e_3 \times_\mathtt{i} x +_\mathtt{i} e_4)) \to \\ \mathtt{i2r}(e_4 \times_\mathtt{i} (e_2 -_\mathtt{i} e_1 +_\mathtt{i} 1) +_\mathtt{i} e_3 \times_\mathtt{i} \left(\frac{e_2(e_2+1)}{2} -_\mathtt{i} \frac{e_1(e_1-1)}{2}\right)) \times_\mathtt{q} \mathtt{c}(e_1 \leq_\mathtt{i} e_2)\end{array}}$$

rem-max
$$\frac{\exists e'.e = e'[\mathtt{max}(e_1, e_2)] \qquad x \in \mathcal{V}ar(e_1, e_2)}{\mathtt{sum}(x\,; e) \to \mathtt{sum}(x\,; \mathtt{c}(e_1 \leq_\mathtt{i} e_2) \times_\mathtt{q} e'[e_2] +_\mathtt{q} \mathtt{c}(e_2 \leq_\mathtt{i} (e_1 -_\mathtt{i} 1)) \times_\mathtt{q} e'[e_1])}$$

rem-min
$$\frac{\exists e'.e = e'[\mathtt{min}(e_1, e_2)] \qquad x \in \mathcal{V}ar(e_1, e_2)}{\mathtt{sum}(x\,; e) \to \mathtt{sum}(x\,; \mathtt{c}(e_1 \leq_\mathtt{i} e_2) \times_\mathtt{q} e'[e_1] +_\mathtt{q} \mathtt{c}(e_2 \leq_\mathtt{i} (e_1 -_\mathtt{i} 1)) \times_\mathtt{q} e'[e_2])}$$

Status

- Probability output analysis is terminating and proven correct
- Complexity of rules increase with the complexity input distribution
  - add(x, y, z,v) => removal of summation over x^2
- Approximations are accurate in most cases, and over-approximating in some recursive cases. f.ex.

```
f(x,y)  = if x=0 || x=2 then y else f(x-1,y+1)
```

$P_{xy}(x,y) = c(x=3)*c(y=3)$

=> $P_f(z) = c(z=3+1) + c(z=3+3)$

Future

- How many step III-rules are needed?
- Combine energy-model in ciaopp with POA result for a step-counting instrumented program (current work)
- Can over-approximations be used to get rid of the restrictions on recursive behavior?