Trace-based probabilistic programming systems allow for the flexible expression of probability measures using addressed randomness and nonstandard execution.

Listing 1. Model program

```plaintext
function model(p1::Float64, p2::Float64)
    z = rand(z, Normal(0.0, 1.0))
    q1 = rand(q1, Normal(p1, exp(z)))
    q2 = rand(q2, Normal(p2, exp(z)))
    x = rand(x, Normal(q1 + q2, 1.0))
    return x
end
```

Inference in these systems is expressed using nonstandard re-execution of the program (with constraints on address and possibly changed arguments). This re-execution can be expensive if the inference primitive has to re-visit redundant computation (choices or computation not affected by changes).

We implement a multi-stage transformation which uses a dataflow-based type system to identify where argument changes flow (before the nonstandard interpretation transformation).

This allows for specialization of inference operations to programs – optimizing performance for iterative algorithms in inference programming.

Listing 8. Original IR

```
l1: (k1, k2, k3)
  l4 = Normal(0.0, 1.0)
  l5 = rand(x, l4)
  l6 = exp(l5)
  l7 = Normal(0.0, l6)
  l8 = rand(x, l7)
  l9 = exp(l8)
  l10 = Normal(l9, 9)
  l11 = rand(x, l10)
  l12 = l8 + l11
  l13 = Normal(l12, 1.0)
  l14 = rand(x, l13)
  return l14
```

Listing 9. Inferred IR

```
l1: (k1 :: var*81#2*, k2 :: NoChange, k3 :: Change)
l4 = Normal(0.0, 1.0) :: NoChange
l5 = (rand(x, l4) :: NoChange
l6 = (exp)(l5) :: NoChange
l7 = (Normal)(l6, l6) :: NoChange
l8 = (rand)(l7, l7) :: NoChange
l9 = (exp)(l8) :: NoChange
l10 = (Normal)(l9, 9) :: Change
l11 = (rand)(l10, l10) :: Change
l12 = (l8 + l11) :: Change
l13 = (Normal)(l12, 1.0) :: Change
l14 = (rand)(x, l13) :: Change
return l14
```

Listing 10. Optimized IR

```
l1: (k1, k2, k3)
l4 = record_cached(LSelf(), i)
l5 = (exp)(l5)
l7 = (Normal)(l6, l6)
l8 = (Self())(rand, l7)
l9 = (exp)(l8)
l10 = (Normal)(l9, 9)
l11 = (Self())(rand, l9)
l12 = (l8 + l11)
l13 = (Normal)(l12, 1.0)
l14 = (Self())(rand, x, l13)
return l14
```

Inference in these systems is expressed using nonstandard randomness and nonstandard execution.

Listing 2. SSA form IR

```
l1 = (k1, k2, k3)
l6 = Normal(0.0, 1.0)
l5 = rand(x, l4)
l6 = exp(l5)
l7 = Normal(0.0, l6)
l8 = rand(x, l7)
l9 = exp(l8)
l10 = Normal(l9, 9)
l11 = rand(x, l10)
l12 = l8 + l11
l13 = Normal(l12, 1.0)
l14 = rand(x, l13)
return l14
```

Listing 3. Trace update operation

```
obs = static([[r1, ] => 5.0])
re, cl, _ = update(obs, cl, A[3.0, NoChange(i), A[6.0, ScalarDiff(1.0)])
```

This is followed by a randomness reachability analysis which determines where addresses with changes propagate in the dataflow. If an IR address depends on a changed randomness address – it can’t be pruned or retrieved from the cached trace and must be re-visited.