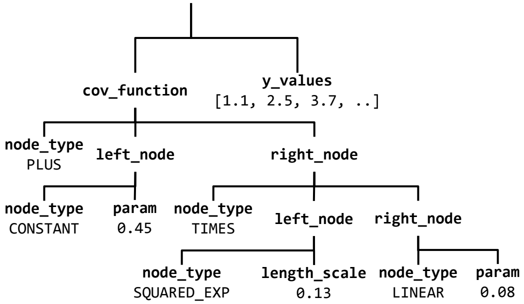
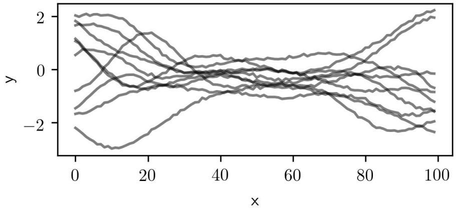
```
@gen function p(x_values::Vector)
   cov_function ~ cov_function_prior()
   cov_matrix = compute_cov_matrix(cov_function, x_values)
   n = length(xs)
   y_values ~ mvnormal(zeros(n), cov_matrix .+ 0.01*I(n))
end

@gen function cov_function_prior()
   node_type ~ categorical(production_rule_probs))
   if node_type == CONSTANT
      param ~ uniform(0, 1)
      return ConstantNode(param)
   elseif node_type == LINEAR
      param ~ uniform(0, 1)
      return LinearNode(param)
   elseif node_type == SQUARED_EXP
      length_scale ~ uniform(0, 1)
      return SquaredExponentialNode(length_scale)
   elseif node_type == PERIODIC
      ..
   elseif node_type == PLUS
      left_node ~ cov_function_prior()
      right_node ~ cov_function_prior()
      return PlusNode(left_node, right_node)
   elseif node_type == TIMES
      left_node ~ cov_function_prior()
      right_node ~ cov_function_prior()
      return TimesNode(left_node, right_node)
   end
end
```

(a) A Gaussian process generative model **p** that uses a PCFG-based prior on a combinatorial space of covariance functions, expressed in a Gen probabilistic modeling language.



(b) An execution trace of the generative model **p** from (a). Each random choice has a unique address in a hierarchical address space that is based on the tree of function calls.

$$k(x, x') = 0.45 + e^{-\frac{(x-x')^2}{0.13}}(x - 0.08)(x' - 0.08)$$



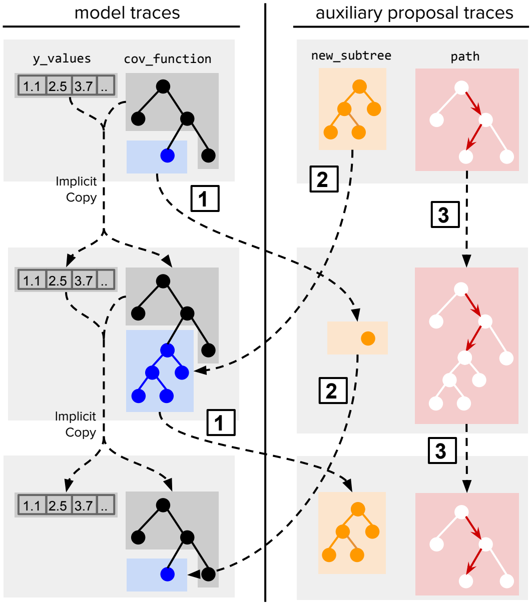
(c) The covariance function represented by the trace in (c), and several samples from the resulting Gaussian process.

```
@gen function q(trace)
   prev_cov_function = trace[:cov_function]
   path ~ walk_tree(prev_cov_function, (:cov_function,))
   new_subtree ~ cov_function_prior()
   return path
end

@gen function walk_tree(node::Node, path)
   if isa(node, LeafNode)
      done ~ bernoulli(1)
      return path
   elseif ({:done} ~ bernoulli(0.5))
      return path
   elseif ({:recurse_left} ~ bernoulli(0.5))
      path = (path..., :left_node)
      return ({:left} ~ walk_tree(node.left, path))
   else
      path = (path..., :right_node)
      return ({:right} ~ walk_tree(node.right, path))
   end
end
```

(d) Auxiliary proposal distribution **q**, which proposes a new subtree in the covariance function, expressed in a Gen probabilistic modeling language.

```
@transform f (model_in, aux_in) to (model_out, aux_out) begin
  path = @read(aux_in[], :discrete)
  subtree_address = foldr(=>, path))                                        1
  @copy(model_in[subtree_address], aux_out[:new_subtree])
  @copy(aux_in[:new_subtree], model_out[subtree_address])
  @copy(aux_in[:path], aux_out[:path])  3                                   2
end
```

(e) An involution **f**, which swaps the previous and newly proposed subtrees.



(f) Schematic showing the involution **f** applied twice. The first application replaces a subtree with one node with a subtree with 5 nodes. The second application reverts the change.