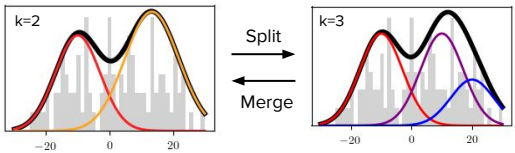


(a) A pair of split-merge transitions in a Gaussian mixture model. The split transition splits the orange component into purple and blue components.

```
@gen function p(n::Int)
    k ~ poisson_plus_one(1)
    means = [({(:mu, j)} ~ normal(0, 10)) for j in 1:k]
    vars = [({(:var, j)} ~ inv_gamma(1, 10)) for j in 1:k]
    weights ~ dirichlet([2.0 for j in 1:k])
    for i in 1:n
        {(:x, i)} ~ mixture_of_normals(weights, means, vars)
    end
end
```

(b) An infinite Gaussian mixture model **p** expressed in a Gen probabilistic modeling language

```
@gen function q(trace)
    k = trace[:k] # current number of clusters
    split = (k == 1) ? true : ({:split} ~ bernoulli(0.5))
    if split
        cluster_to_split ~ uniform_discrete(1, k)
        u1 ~ beta(2, 2); u2 ~ beta(2, 2); u3 ~ beta(1, 1)
    else
        cluster_to_merge ~ uniform_discrete(1, k-1)
    end
end
```

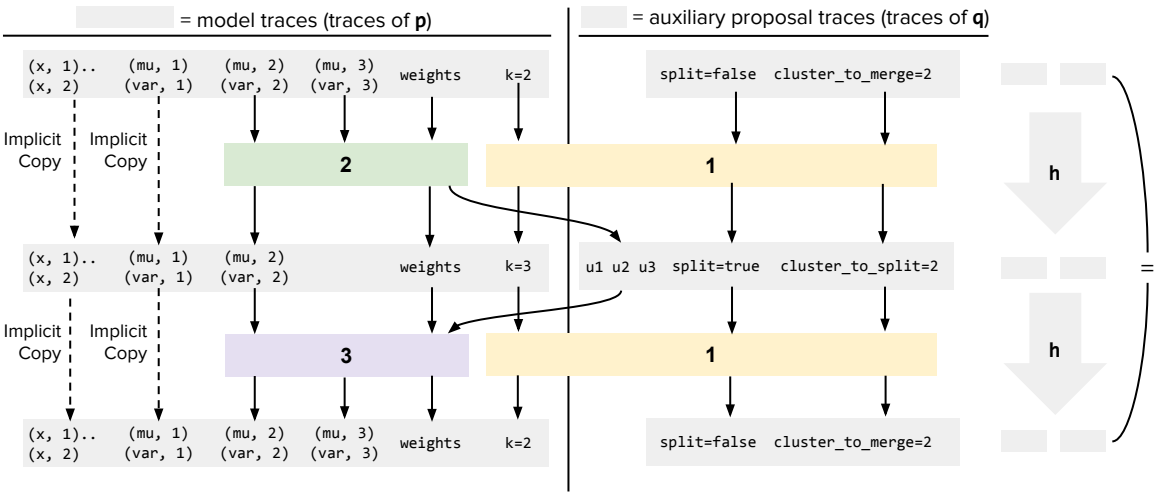
(c) Auxiliary proposal distribution **q** for the split-merge reversible jump MCMC move, expressed in a Gen probabilistic modeling language

```
(observations, n) = get_observations()
trace, = Gen.generate(p, (n,), observations)
for iter in 1:100
    trace = Gen.involutive_mcmc(trace, q, (), h)
end
```

(e) Julia code that applies the split-merge reversible jump MCMC move defined by **q** and **h** 100 times to a trace of the model **p**, using Gen's involution MCMC operator.

```
@trace_bijection h from (p_in, q_in) to (p_out, q_out) begin
    k = @read(p_in, :k, discrete)
    split = (k == 1) ? true : @read(q_in, :split, discrete)                    1
    if split
        cluster_to_split = @read(q_in, :cluster_to_split, discrete)
        @write(p_out, :k, k+1, discrete)
        @copy(q_in, :cluster_to_split, q_out, :cluster_to_merge)
        @write(q_out, :split, false, discrete)
    else
        cluster_to_merge = @read(q_in, :cluster_to_merge, discrete)
        @write(p_out, :k, k-1, discrete)
        @copy(q_in, :cluster_to_merge, q_out, :cluster_to_split)
        if (k > 2) @write(q_in, :split, true, discrete) end
    end
    if split
        u1 = @read(q_in, :u1, continuous)                                      2
        u2 = @read(q_in, :u2, continuous)
        u3 = @read(q_in, :u3, continuous)
        weights = @read(p_in, :weights, continuous)
        mu = @read(p_in, (:mu, cluster_to_split), continuous)
        var = @read((p_in, :var, cluster_to_split), continuous)
        new_weights = split_weights(weights, cluster_to_split, u1, k)
        w1 = new_weights[cluster_to_split]; w2 = new_weights[k+1]
        (mu1, mu2, var1, var2) = split_params(mu, var, u2, u3, w1, w2)
        @write(p_out, :weights, new_weights, continuous)
        @write(p_out, (:mu, cluster_to_split), mu1, continuous)
        @write(p_out, (:mu, k+1), mu2, continuous)
        @write(p_out, (:var, cluster_to_split), var1, continuous)
        @write(p_out, (:var, k+1), var2, continuous)
    else
        mu1 = @read(p_in, (:mu, cluster_to_merge), continuous)                 3
        mu2 = @read(p_in, (:mu, k), continuous)
        var1 = @read(p_in, (:var, cluster_to_merge), continuous)
        var2 = @read(p_in, (:var, k), continuous)
        weights = @read(p_in, :weights, continuous)
        w1 = weights[cluster_to_merge]; w2 = weights[k]
        (new_weights, u1) = merge_weights(weights, cluster_to_merge, k)
        w = new_weights[cluster_to_merge]
        (mu, var, u2, u3) = merge_params(mu1, mu2, var1, var2, w1, w2, w)
        @write(p_out, :weights, new_weights, continuous)
        @write(p_out, (:mu, cluster_to_merge), mu, continuous)
        @write(p_out, (:var, cluster_to_merge), var, continuous)
        @write(p_out, (:u1, u1), continuous)
        @write(p_out, (:u2, u2), continuous)
        @write(p_out, (:u3, u3), continuous)
    end
end
```

(d) Involution **h** for the split-merge reversible jump MCMC move, expressed in a Gen differentiable programming language.



(f) Schematic of the involution **h** being applied twice in succession to a model trace and auxiliary proposal trace. The first application merges cluster 2 with the last cluster (cluster 3); the second application splits the resulting cluster, and returns the original pair of traces.