Algorithm 10.2 Calibration using sum-product message passing in a clique tree

```
    Procedure CTree-SP-Calibrate (
      Φ,    // Set of factors
      T     // Clique tree over Φ
    )
1     Initialize-Cliques
2     while exist i, j such that i is ready to transmit to j
3       δ_{i→j}(S_{i,j}) ← SP-Message(i, j)
4     for each clique i
5       β_i ← ψ_i · ∏_{k∈Nb_i} δ_{k→i}
6     return {β_i}
```

algorithm continues until the leaves of the tree are reached, at which point no more messages need to be sent. This second phase is called the downward pass. The asynchronous algorithm is equivalent to this systematic algorithm, except that the root is simply the first clique that happens to obtain messages from all of its neighbors. In an actual implementation, we might want to *schedule* this process more explicitly. (At the very least, the algorithm would check in line 2 that a message is not computed more than once.)

message scheduling

Example 10.4

Figure 10.3a shows the upward pass of the clique tree algorithm when $\boldsymbol{C}_5$ is the root. Figure 10.5a shows a possible first step in a downward pass, where $\boldsymbol{C}_5$ sends a message to its child $\boldsymbol{C}_3$, based on the message from $\boldsymbol{C}_4$ and its initial potential. As soon as a child of the root receives a message, it has all of the information it needs to send a message to its own children. Figure 10.5b shows $\boldsymbol{C}_3$ sending the downward message to $\boldsymbol{C}_2$. ■

beliefs

At the end of this process, we compute the *beliefs* for all cliques in the tree by multiplying the initial potential with each of the incoming messages. The key is to note that the messages used in the computation of β_i are precisely the same messages that would have been used in a standard upward pass of the algorithm with $\boldsymbol{C}_i$ as the root. Thus, we conclude:

Corollary 10.2

Assume that, for each clique i, β_i is computed as in the algorithm of algorithm 10.2. Then

$$\beta_i(\boldsymbol{C}_i) = \sum_{\mathcal{X} - \boldsymbol{C}_i} \tilde{P}_\Phi(\mathcal{X}).$$

Note that it is important that $\boldsymbol{C}_i$ compute the message to a neighboring clique $\boldsymbol{C}_j$ based on its *initial potential* ψ_i and not its modified potential β_i. The latter already integrates information from j. If the message were computed based on this latter potential, we would be double-counting the factors assigned to $\boldsymbol{C}_j$ (multiplying them twice into the joint).

When this process concludes, each clique contains the marginal (unnormalized) probability over the variables in its scope. As we discussed, we can compute the marginal probability over a particular variable X by selecting a clique whose scope contains X, and eliminating the redundant variables in the clique. A key point is that the result of this process does not depend on the clique we selected. That is, if X appears in two cliques, they must agree on its marginal.