Spanning Tree

Root:

```
loop
  for $m := 1$ to $\delta$ do
    write $r_{mi} := r_{im} := \langle 0, 0 \rangle$
  end for
end loop
```

Others:

```
loop
  for $m := 1$ to $\delta$ do
    $lr_{mi}$ := read($r_{mi}$)
    FirstFound := false
    dist := 1 + min$\{lr_{mi}.\text{dis} \mid 1 \leq m \leq \delta\}$

    for $m := 1$ to $\delta$ do
      if FirstFound and $lr_{mi}$.dis = dist - 1 then
        write $r_{im}$ := ⟨1, dist⟩
        FirstFound := true
      else
        write $r_{im}$ := ⟨0, dist⟩
      end if
    end for

  end for
end loop
```

Dijkstras mutual exclusion

P_1:

```
loop
  if $x_1 = x_n$ then
    $x_1 := (x_1 + 1) \bmod (n + 1)$
  end if
end loop
```

P_i $(i \neq 1)$:

```
loop
  if $x_i \neq x_{i\text{-}1}$ then
    $x_i := x_{i\text{-}1}$
  end if
end loop
```

Mutual exclusion for tree structure

Root:

```
loop
   read $lr_{1,i}$ := read($r_{1,i}$)
   if $lr_{\delta,i} = r_{i,1}$ then
      write $r_{i,2} := (lr_{1,i} + 1) \bmod (4n + 5)$
   end if

   for $m := 2$ to $\delta$ do
      $lr_{m,i}$ := read($r_{m,i}$)
      write $r_{i,m+1} := lr_{m,i}$
   end for
end loop
```

Others:

```
loop
   read $lr_{1,i}$ := read($r_{1,i}$)
   if $lr_{1,i} \neq r_{i,2}$ then
      write $r_{i,2} := lr_{1,i}$
   end if

   for $m := 2$ to $\delta$ do
      $lr_{m,i}$ := read($r_{m,i}$)
      write $r_{i,m+1} := lr_{m,i}$
   end for
end loop
```

Maximal matching

```
loop
  if $\mathrm{pointer}_i = \mathrm{null}$ and $(\exists\ P_j \in N(i) \mid \mathrm{pointer}_j = i)$ then
    $\mathrm{pointer}_i := j$
  end if

  if $\mathrm{pointer}_i = \mathrm{null}$ and $(\forall\ P_j \in N(i) \mid \mathrm{pointer}_j \neq i)$ and $(\exists\ P_j \in N(i) \mid \mathrm{pointer}_j = \mathrm{null})$ then
    $\mathrm{pointer}_i := j$
  end if

  if $\mathrm{pointer}_i = j$ and $\mathrm{pointer}_j = k$ and $k \neq i$ then
    $\mathrm{pointer}_i := \mathrm{null}$
  end if
end loop
```

Leader election in general graph

```
loop
  $\langle$candidate, distance$\rangle$ := $\langle$ID(i, 0)$\rangle$
  for all $P_j \in N(i)$ do
    $\langle leader_i[j], dist_i[j] \rangle$ := read($\langle leader_i, dist_i \rangle$ )

    if (($dist_i[j] < N$) and ($leader_i[j] <$ candidate)) or (($leader_i[j] =$ candidate)
    and ($dist_i[j] <$ distance)) then
      $\langle$candidate, distance$\rangle$ := read($\langle leader_i, dist_i \rangle$)
    end if

  end for
  write $\langle leader_i, dist_i \rangle$ := $\langle$candidate, distance$\rangle$
end loop
```

Self-stabilizing counting

Root:

```
loop
  sum := 0
  for all $P_j \in$ children(i) do
    $lr_{j,i}$ := read($r_{j,i}$)
    sum := sum + $lr_{j,i}$.count
  end for
  $count_i$ := sum + 1
end loop
```

Others:

```
loop
  sum := 0
  for all $P_j \in$ children(i) do
    $lr_{j,i}$ := read($r_{j,i}$)
    sum := sum + $lr_{j,i}$.count
  end for
  $count_i$ := sum + 1
  write $r_{i,\ parent}$.count := $count_i$
end loop
```

Self-stabilizing naming

Root:

```
loop
  $ID_i := 1$
  $sum := 0$
  for all $P_j \in children(i)$ do
    $lr_{j,i} := read(r_{j,i})$
    $lr_{j,i} := ID_i + sum + 1$
    $sum := sum\ lr_{j,i}.count$
  end for
end loop
```

Others:

```
loop
  $sum := 0$
  $lr_{parent,i} := read(r_{parent,i})$
  $ID_i := lr_{parent,i}.identifier$
  for all $P_j \in children(i)$ do
    $lr_{j,i} := read(r_{j,i})$
    $lr_{j,i} := ID_i + sum + 1$
    $sum := sum\ lr_{j,i}.count$
  end for
end loop
```

Digital clock synchronization (bounded)

Upon a pulse:

```
for all $P_j \in N(i)$ do
  $send(j, clock_i)$
end for
$max := clock_i$
for all $P_j \in N(i)$ do
  $receive(clock_j)$
  if $max < clock_j$ then
    $max := clock_j$
  end if
end for$clock_i := (max + 1) \bmod ((n+1)d+1)$
```

Self-stabilizing counting in non-rooted tree

```
loop
  for all $P_j \in N(i)$ do
    $lr_{j,i} := read(r_{j,i})$
    $sum_i := 0$

    for all $P_j \in N(i)$ do
      $sum_j := 0$

      for all $P_k \in N(i)$ do
        if $P_j \neq P_k$ then
          $sum_j := sum_j + lr_{ki}.count$
        end if
      end for

    end for
    $count_i[j] := sum_j + 1$
    $sum_i := sum_i + sum_j$
    write $r_{ij}.count := count_i[j]$
  end for
  $count_i := sum_i + 1$
end loop
```

Update algorithm for P_i

```
loop
  Readset := $\emptyset$
  for all $P_j \in N(i)$ do
    $Readset_i := Readset_i \cup read(Processors_j)$
  end for
  $Readset_i := Readset_i \backslash\backslash \langle i, * \rangle$
  $Readset_i := Readset_i ++ \langle *, 1 \rangle$
  $Readset_i := Readset_i \cup \langle i, 0 \rangle$
  for all $P_j \in Processors(Readset_i)$ do
    $Readset_i := Readset_i \backslash\backslash NotMinDis(P_j, Readset_i)$
  end for
  write $Processors_i := ConPrefix(Readset_i)$
end loop
```

Superstabilizing coloring

```
loop
  AColors := $\emptyset$
  GColors := $\emptyset$
  for $m := 1$ to $\delta$ do
    $lr_m := read(r_m)$
    AColors = Acolors $\cup$ $lr_m$.color
    if $ID(m) > i$ then
      GColors = Gcolors $\cup$ $lr_m$.color
    end if
  end for
  if $colors_i = \perp$ or $color_i \in$ GColor then
    $color_i$ = choose(\\ AColor)
  end if
  write $color_i$ := color
end loop
```

Interrupt section

```
if $receiver_{ij}$ and $j > i$ then
  $colors_i = \perp$
  write $colors_i = \perp$
end if
```