Algorithm 35 `CircP2`($\mathcal{AC}$, vr(), dr()). Assumes the values of leaf circuit nodes v have been initialized in vr(v) and the circuit alternates between addition and multiplication nodes, with leaves having multiplication parents.

input:

$\mathcal{AC}$:	arithmetic circuit
vr():	array of value registers (one register for each circuit node)
dr():	array of derivative registers (one register for each circuit node)

output: computes the value of circuit output v in vr(v) and computes derivatives of leaf nodes v in dr(v)

main:

```
1:  for each non-leaf node $v$ with children $c$ (visit children before parents) do
2:    if $v$ is an addition node then
3:      $\mathsf{vr}(v) \leftarrow \sum_{c:\,\mathsf{bit}(c)=0} \mathsf{vr}(c)$ {if $\mathsf{bit}(c) = 1$, value of $c$ is 0}
4:    else
5:      if $v$ has a single child $c'$ with $\mathsf{vr}(c') = 0$ then
6:        $\mathsf{bit}(v) \leftarrow 1; \mathsf{vr}(v) \leftarrow \prod_{c \neq c'} \mathsf{vr}(c)$
7:      else
8:        $\mathsf{bit}(v) \leftarrow 0; \mathsf{vr}(v) \leftarrow \prod_{c} \mathsf{vr}(c)$
9:      end if
10:   end if
11: end for
12: $\mathsf{dr}(v) \leftarrow 0$ for all non-root nodes $v$; $\mathsf{dr}(v) \leftarrow 1$ for root node $v$
13: for each non-root node $v$ (visit parents before children) do
14:   for each parent $p$ of node $v$ do
15:     if $p$ is an addition node then
16:       $\mathsf{dr}(v) \leftarrow \mathsf{dr}(v) + \mathsf{dr}(p)$
17:     else
18:       if $\mathsf{vr}(p) \neq 0$ then {$p$ has at most one child with zero value}
19:         if $\mathsf{bit}(p) = 0$ then {$p$ has no zero children}
20:           $\mathsf{dr}(v) \leftarrow \mathsf{dr}(v) + \mathsf{dr}(p)\mathsf{vr}(p)/\mathsf{vr}(v)$
21:         else if $\mathsf{vr}(v) = 0$ then {$v$ is the single zero child}
22:           $\mathsf{dr}(v) \leftarrow \mathsf{dr}(v) + \mathsf{dr}(p)\mathsf{vr}(p)$
23:         end if
24:       end if
25:     end if
26:   end for
27: end for
```

The bottom-up pass in Algorithm 34 clearly takes time linear in the circuit size, where size is defined as the number of circuit edges. However, the top-down pass takes linear time only when each multiplication node has a bounded number of children; otherwise, the time to evaluate the term $\prod_{v' \neq v} \mathsf{vr}(v')$ cannot be bounded by a constant.

This is addressed by Algorithm 35, which is based on observing that the term $\prod_{v' \neq v} \mathsf{vr}(v')$ equals $\mathsf{vr}(p)/\mathsf{vr}(v)$ when $\mathsf{vr}(v) \neq 0$ and, hence, the time to evaluate it can be bounded by a constant if we use division. Even the case $\mathsf{vr}(v) = 0$ can be handled efficiently but that requires an additional bit per multiplication node p:

- $\mathsf{bit}(p) = 1$ when exactly one child of node p has a zero value.