Algorithm 1: Handling Inconsistent Blockchain Views (G)

Input: Received Blockchain graph G
Output: authoritative blockchain A_v
// Collecte all valid sibling groups to S
1 $S \leftarrow \varnothing$
2 **foreach** $B \in G$ **do**
3 | $s \leftarrow \textsc{Siblings}(B)$
4 | $S \leftarrow S \cup s$
5 **end**
// Initialize weights of each sibling groups
6 **foreach** $s \in S$ **do**
7 | $W(s) \leftarrow 0$
8 **end**
// Calculate weights of all sibling groups
9 **foreach** $B \in G$ **do**
10 | **foreach** $s \in S$ **do**
11 | | **foreach** $v \in \textsc{SubviewsBeginWith}(s)$ **do**
12 | | | **if** $B \in v$ **then**
13 | | | | $W(s) \leftarrow W(s) + \textsc{Miningpower}(B)$
14 | | | | **break**
15 | | | **end**
16 | | **end**
17 | **end**
18 **end**
// Find the authoritative blockchain
19 $s \leftarrow \textsc{Siblings}(B^0)$ **if** $\textsc{ChildSiblingGroups}(s) = \varnothing$ **then**
20 | **return** s
21 **else**
22 | Update $s \leftarrow \underset{s' \in \textsc{ChildSiblingGroups}(s)}{\arg\max} W(s')$
23 | $A_v = A_v \cup s$ Go to line 19
24 **end**

Function: Subviews Begin with

```
   // Function to find all subviews that begin with s
 1 Function SubviewsBeginWith(s):
 2 |  V ← ∅
 3 |  if  ChildSiblingGroups(s) = ∅ then
 4 |  |  v ← ConstructSubview(s)
 5 |  |  V ← V.Append(v)
 6 |  |  return V
 7 |  else
 8 |  |  foreach c ∈ ChildSiblingGroups(s) do
 9 |  |  |  foreach v ∈ SubviewsBeginWith(c) do
10 |  |  |  |  V.Append(v.Append(s))
11 |  |  |  end
12 |  |  end
13 |  end
14 return V
```

Function: Child Sibling Groups

```
   // Function to find child siblings
 1 Function ChildSiblingGroups(s):
 2 |  C ← ∅
 3 |  foreach s′ ∈ S do
 4 |  |  foreach B′ ∈ s′ do
 5 |  |  |  foreach B ∈ s do
 6 |  |  |  |  if B ∈ B′.predecessors then
 7 |  |  |  |  |  C.Append(s′)
 8 |  |  |  |  end
 9 |  |  |  end
10 |  |  end
11 |  end
12 return C
```