Algorithm 2 Constraint Solving over Discrete Domains

```
function SOLVE($C, X = \{x_1, ..., x_n\}, D_0$)    ▷ Constraints, variables, initial domains
    $\sigma \leftarrow \emptyset$    ▷ Assignment function, initially empty
    $D \leftarrow D_0(x_1) \times \cdots \times D_0(x_n)$    ▷ Domains
    $\Delta \leftarrow \emptyset$    ▷ Decision stack
    while true do
        $\sigma, D \leftarrow$ Propagate($C, \sigma, D$)    ▷ Propagate using arc-consistency
        if Conflict $\notin \sigma$ then
            if AllAssigned($\sigma$) then    ▷ Check if all variables are assigned
                return Solution($\sigma$)    ▷ Solved, return the full assignment in solution format
            else
                $\sigma, x \leftarrow$ MakeDecision($C, \sigma, D$)    ▷ Assign some value to an unassigned variable
                $\Delta \leftarrow \Delta$.push($\sigma, x, D$)    ▷ Create backtracking point
            end if
        else
            if $\Delta == \emptyset$ then
                return NoSolution
            else
                $\sigma, D \leftarrow$ Backtrack($\Delta, \sigma$)    ▷ Backtrack to the latest decision
            end if
        end if
    end while
end function
function PROPAGATE($C, \sigma, D$)
    while True do
        if $D(x_i) == \{a\}$ for some unassigned $x_i$ then
            $\sigma = \sigma \cup \{(x_i, a)\}$    ▷ Make assignment if domain becomes singleton
        end if
        if $D(x_i)$ is empty for some $x_i$ then
            return $\sigma \cup \{$Conflict$\}, D$
        end if
        if there exists $i, j$ such that $a_i \in D(x_i)$ is not consistent with any $a_j \in D(x_j)$ in $C$ then
            $D(x_i)$.Remove($a_i$)
        else
            return $\sigma, D$
        end if
    end while
end function
function MAKEDECISION($C, \sigma, D$)
    if $x$ is unassigned in $\sigma$ then
        $a \leftarrow$ some element in $D(x)$
        return $\sigma \cup \{(x, a)\}, x$
    end if
end function
function BACKTRACK($\Delta, \sigma$)
    $\sigma, x, D \leftarrow \Delta.pop()$
    $D(x)$.remove($\sigma(x)$)    ▷ Make sure to remove the previous decision from its domain
    return $\sigma, D$
end function
```