Nexct

input : • a transition system (S, R)

• H an annotation function over S

output : the set $\mathcal{H}'$ of the annotation functions obtained by applying the component of the unmarked successor formulas

algo :

```
H = ∅ // set of set of annotations functions
For s in S do {
    F = ∅ ; // set of annotation functions obtained by applying the successor formulas of H(s)
    For f in H(s) do {
        if f = Xφ and f is not marked then {
            F = ∅ // set of annotations functions obtained by adding φ to the set of annotations of ONE successor of s
            mark f ;
            For t in R(s) do {
                H' = add(H, t, φ); //add φ to H(t)
                if H' is not patently inconsistent then F = F ∪ {H'};
            }
            if F = ∅ then return ∅ // No completion because f can not be applied
            else S = S ∪ {F} ;
        }
    }
}
```

if $S \neq \emptyset$ then $\mathcal{H} = \mathcal{H} \cup \{S\}$

// $S = \emptyset$ means there is no successor formula in $H(s)$ so there is no completion to add

}

return product($\mathcal{H}$); // compute the product of the annotations functions and remove patently inconsistent ones

---

product

input: $\mathcal{H}$ a set of set of annotations functions alldefined over the same set of states S

output: a set of annotations functions

algo:

$$P = \prod_{i=0}^{size(\mathcal{H})} \mathcal{H}(i)$$

$S = \emptyset$

For h in P {

$H : s \longmapsto \bigcup_{i=0}^{size(\mathcal{H})} h(i)$

if H is not patently inconsistent then

$S = S \cup \{H\}$

}

return S