```
input : caller : Proc, s : Inst
output:
switch TypeOf(s) do
    case COPY [p = q]
        if inFlow_s[q] = ∅ then
            outFlow_s[p] ← inFlow_s[q] ∪ {s};
        end
    endsw
    case LOAD [p = *q]
        foreach a ∈ pt_[s](q) do
            if inFlow_s[a] = ∅ then
                outFlow_s[p] ← inFlow_s[p] ∪ {s}
            end
        end
    endsw
    case SOURCE [call func(a_0, a_1, ..., a_n)]
        foreach k ∈ {0, 1, ..., n} do
            if taint(k) then
                outFlow_s[a_k] ← inFlow_s[a_k] ∪ {s}
            end
        end
    endsw
    case CALL [call func(a_0, a_1, ..., a_n)]
        if caller = func then
            foreach a_k, k ∈ {0, 1, ..., n} do
                f_k ← formal(func, a_k)
                if a_k ∈ P then
                    foreach b ∈ pt_[s](a_k) do
                        inFlow_s[f_k] ← inFlow_s[f_k] ∪ inFlow_s[b]
                    end
                end
                else if a_k ∈ A then
                    t_k ← topLevel(a_k)
                    foreach b ∈ pt_[s](t_k) do
                        inFlow_s[f_k] ← inFlow_s[f_k] ∪ inFlow_s[b]
                    end
                end
            end
            Flow(caller, func)
        end
    endsw
    case ADDROF [p = &a]
    case STORE [*p = q]
    case SINK [call func]
    endsw
endsw
```

Algorithm 2: Flow