Algorithm 1: Generate an equivalent transition system for a multi-agent planning specification in terms of operational semantics.

Input: $Agents$, KB, C, A, AE, S, P, D, $Constants$, $Dummy\_Agents$
Output: $(S, Act, transitions, I, F, AP, L)$

Each agent in $Agents$ contains five attributes: id, B, $goals$, $sent\_msg$, $received\_msg$.
$I = \{\}$
for *agent in Agents* **do**
    $I.update(\{id : (B, goals)\})$
$S = [I]$
$transitions = []$
$current\_states = [I]$
$Act = \{\}$
$AP = \{\}$
$L = \{\}$
while $current\_states \neq []$ **do**
    $next\_states = []$
    $substate\_dict = \{\}$
    $new\_substate\_dict = \{\}$
    $transition\_dict = \{\}$
    for *state in current_states* **do**
        for *agent in Agents* **do**
            $substate = state[id]$
            $B = substate[0]$
            $goals = substate[1]$
            $atom\_current$ is derived by B, KB, D, $Constants$.
            if $goals \neq []$ **then**
                $G = goals[0]$
                $atom\_goal$ is derived by G, KB, D, $Constants$.
            else
                $atom\_goal = []$
            EC is derived by $atom\_current$, $atom\_goal$, and C.
            EA is derived by $atom\_current$, A, EC, D, and $Constants$.
            if $EA == []$ **then**
                ES is derived by $atom\_current$, EC, S, D, and $Constants$.
            EP is derived by $atom\_current$, $atom\_goal$, $received\_msg$, P, D, and $Constants$.
            EE is derived by $atom\_current$, EA, and AE.
            Update $new\_substate\_dict[name]$ by EP and EE.
            Update $transition\_dict[name]$ by EA and ES.
        Update $next\_states$ by $new\_substate\_dict$, and add new states to S.
        Update $transisitions$ by $transition\_dict$.
    $end\_states = next\_states$
    Remove final state from $next\_states$.
    $current\_states = next\_states$
    Convert $sent\_msg$ to $received\_msg$.

```
F = []
for s ∈ S do
    flag = True
    for id ∈ s do
        substate = S[id]
        if substate[1] ≠ [] then
            flag = False
            break
    if flag then
        F.append(s)
if ∀s ∈ end_states.s ∈ F then
    Add all possible actions to Act
    Add all ground atoms to AP
    Add all transitions to transitions.
    Add all state properties to L
else
    return None
return (S, Act, transitions, I, F, AP, L)
```