Unification Algorithm

```
def unify(psi1, psi2):
    if is_variable_or_constant(psi1) or is_variable_or_constant(psi2):
        if psi1 == psi2:
            return {}
        elif is_variable(psi1):
            if occurs_in(psi1, psi2):
                return "FAILURE"
            else:
                return {psi1: psi2}
        elif is_variable(psi2):
            if occurs_in(psi2, psi1):
                return "FAILURE"
            else:
                return {psi2: psi1}
        else:
            return "FAILURE"

    if predicate_symbol(psi1) != predicate_symbol(psi2):
        return "FAILURE"

    if len(psi1['args']) != len(psi2['args']):
        return "FAILURE"

    SUBST = {}

    for i in range(len(psi1['args'])):
        S = unify(psi1['args'][i], psi2['args'][i])
        if S == "FAILURE":
            return "FAILURE"
        if S:
```

```
            psi1 = apply_substitution(S, psi1)
            psi2 = apply_substitution(S, psi2)
            SUBST.update(S)

    return SUBST


def is_variable_or_constant(x):
    return isinstance(x, str) and (x.islower() or x.isalpha())

def is_variable(x):
    return isinstance(x, str) and x.islower()

def occurs_in(var, expr):
    if var == expr:
        return True
    if isinstance(expr, dict):
        return any(occurs_in(var, arg) for arg in expr.get('args', []))
    return False

def predicate_symbol(expr):
    if isinstance(expr, dict) and 'pred' in expr:
        return expr['pred']
    return None

def apply_substitution(subst, expr):
    if isinstance(expr, str):
        return subst.get(expr, expr)
    elif isinstance(expr, dict):
        return {
            'pred': expr['pred'],
```

```
            'args': [apply_substitution(subst, arg) for arg in expr.get('args', [])]
        }
    return expr


psi1 = {'pred': 'P', 'args': ['x', 'y']}
psi2 = {'pred': 'P', 'args': ['a', 'b']}
result = unify(psi1, psi2)
print(result)
```

Output:

```
{'x': 'a', 'y': 'b'}
```