```python
def unify(x, y, subst=None):
    """
    Unifies two terms `x` and `y` and returns the substitution.
    :param x: First term
    :param y: Second term
    :param subst: Current substitution (default is None)
    :return: A substitution dictionary or failure
    """
    if subst is None:
        subst = {}

    # Case 1: If both terms are identical, no unification is needed
    if x == y:
        return subst

    # Case 2: If either term is a variable, try to unify by substitution
    elif isinstance(x, str) and x.islower():
        return unify_variable(x, y, subst)

    elif isinstance(y, str) and y.islower():
        return unify_variable(y, x, subst)

    # Case 3: If both terms are compound (functions) and have the same functor (name)
    elif isinstance(x, tuple) and isinstance(y, tuple):
        if len(x) != len(y):
            return None  # Arity mismatch, cannot unify
        for x_sub, y_sub in zip(x[1:], y[1:]):  # skip function name (first element)
            subst = unify(x_sub, y_sub, subst)
            if subst is None:
                return None
        return subst

    # Case 4: If we cannot unify, return None (failure)
    else:
        return None


def unify_variable(var, term, subst):
    """
    Helper function to unify a variable with a term.
    :param var: The variable
    :param term: The term
    :param subst: Current substitution
    :return: Updated substitution or failure
    """
    if var in subst:
        return unify(subst[var], term, subst)

    if term == var:
        return subst

    # Check if the term contains the variable (no circular references)
    if isinstance(term, str) and term.islower():
        if var in term:
            return None  # Prevent infinite recursion

    # Otherwise, add the variable substitution
    subst[var] = term
    return subst


# Test the unification function
x = ('f', 'X', 'a')
y = ('f', 'b', 'Y')

result = unify(x, y)
print("Substitution result:", result)
```

```
Substitution result: None
```