9 Relations

9.4 Closure of Relations

The closure of a relation is with respect to a property of the relation:

1. Reflexive closure of R: the reflexive relation S such that $R \subseteq S$, and $|S|$ is the smallest among all such possible relations
   How do we find reflexive closure S?
   - Define the diagonal relation $\Delta = \{(a,a) | \; \forall a \in A\}$. Then $S = R \cup \Delta$
2. Symmetric closure of R: the symmetric relation S such that $R \subseteq S$, and $|S|$ is the smallest among all such possible relations
   How do we find symmetric closure S?
   - Define the inverse relation $R^{-1} = \{(b,a) | \; \forall (a,b) \in R\}$. Then $S = R \cup R^{-1}$
3. Transitive closure of R: the transitive relation S such that $R \subseteq S$, and $|S|$ is the smallest among all such possible relations
   How do we find transitive closure S?
   - Define a path of length n from a to b in a directed graph is a sequence of n edges $(a, a_1), (a_1, a_2), (a_2, a_3), \ldots, (a_{n-1}, b)$ $\exists n \geq 2$. We can denote the path just by the vertices: $a, a_1, a_2, \ldots, a_{n-1}, b$
   - A circuit is a closed path, i.e. $a = b$
   - $(a,b) \in R^n$ iff $\exists$ an $a - b$ path of length n
   - $S = R^*$, where the connectivity relation $R^* = \bigcup_{n=1}^{\infty} R^n$, i.e. it consists of all elements (a,b) such that there is an $a - b$ path.
   - The associated matrix, $M_{R^*} = M_R \vee M_R^{[2]} \vee M_R^{[3]} \vee \ldots \vee M_R^{[n]}$, where
     $A^{[2]}$ is the Boolean product $A \odot A = (a_{i1} \wedge a_{1j}) \vee (a_{i2} \wedge a_{2j}) \vee \cdots \vee (a_{in} \wedge a_{nj})$
     $A^{[3]}$ is the Boolean product $A^{[2]} \odot A$ and so on

ALGORITHM 1 A Procedure for Computing the Transitive Closure.

```
procedure transitive closure (M_R : zero–one n × n matrix)
A := M_R
B := A
for i := 2 to n
   A := A ⊙ M_R
   B := B ∨ A
return B{B is the zero–one matrix for R*}
```

(a) Algorithm 1

ALGORITHM 2 Warshall Algorithm.

```
procedure Warshall (M_R : n × n zero–one matrix)
W := M_R
for k := 1 to n
      for i := 1 to n
            for j := 1 to n
                  w_ij := w_ij ∨ (w_ik ∧ w_kj)
return W{W = [w_ij] is M_R*}
```

(b) Alg 2: Warshall's Algorithm