SWEN-220 Math Models

Transitive Closure
Transitive Closure

• Represents reachability

G = { (A -> B), (B -> C)}
Transitive Closure

• Represents reachability
• All paths both direct and indirect

\[ G = \{ (A \rightarrow B), (B \rightarrow C) \} \]

\[^G = \{ (A \rightarrow B), (B \rightarrow C), (A \rightarrow C) \} \]

"C is reachable from A"
Transitive Closure Example

- GrandFatherSon = {(F0,S0), (F0,S1), (F1,S0),(F1, S2), (S1,S3), (S2,S4), (F2,S5)}
Transitive Closure Example

What is $\overline{\text{GrandFatherSon}}$?

\{(F0,S0), (F0,S1), (F1, S0), (F1, S2), (S1,S3),
(S2,S4),(F2,S5), (F0,S3), (F1,S4)\}
Transitive Closure Example

What is $^\wedge$GrandFatherSon?

\{(F0, S0), (F0, S1), (F1, S2), (S1, S3), (S2, S4), (F2, S5),
(F0, S3), (F1, S4)\}
Transitive Closure Example

What is $\text{^\text{GrandFatherSon}}$?

$$\{(F0,S0), (F0,S1), (F1, S2), (S1,S3),(S2,S4),(F2,S5), (F0,S3), (F1,S4)\}$$
Transitive Closure (\(^\))

some sig Course{
    prereqs : set Course
}

run{
    some c : Course | #c.prereqs = 3
} for 5

This solution has courses that have themselves as prerequisites and paths that cycle back through other courses.

Not a Directed Acyclic Graph (DAG)!
Transitive Closure (^)

some sig Course{
  prereqs : set Course
}

fact CannotBeOwnPreq {
  all c : Course | c !in c.^prereqs
}

run{
  some c : Course | #c.prereqs = 3
} for 5

The CannotBeOwnPreq fact prevents cycles from being created in the graph, which is now a DAG.
Transitive Closure (^)

prereqs

\{\text{Course}0\rightarrow\text{Course}2, \text{Course}1\rightarrow\text{Course}0, \text{Course}3\rightarrow\text{Course}0, \text{Course}3\rightarrow\text{Course}1, \text{Course}3\rightarrow\text{Course}2\}\n
^{\text{prereqs}}

\{\text{Course}0\rightarrow\text{Course}2, \text{Course}1\rightarrow\text{Course}0, \text{Course}1\rightarrow\text{Course}2, \text{Course}3\rightarrow\text{Course}0, \text{Course}3\rightarrow\text{Course}1, \text{Course}3\rightarrow\text{Course}2\}\
Note that the results of the transitive closure operation includes the path **Course$1$ -> Course$2$**

"Course$2$ is "reachable" from Course$1" or
or
"Course$2$ is in the prerequisite chain for Course$1"
Transitive Closure (^)

fact CannotBeOwnPreq {
    all c : Course | c ! in (c.(^prereqs))
}

Course$0.^prereqs   //"All courses that are prereqs for Course$0"

Course$0 = { Course$0 }
^prereqs = { Course$0->Course$2, Course$1->Course$0, Course$1->Course$2, Course$3->Course$0, Course$3->Course$1, Course$3->Course$2

Join all elements of Course$0 to all elements of ^prereqs (create triplets)
{
Course$0->Course$0->Course$2, Course$0->Course$1->Course$0, Course$0->Course$1->Course$2, Course$0->Course$3->Course$0, Course$0->Course$3->Course$1, Course$0->Course$3->Course$2
}
Transitive Closure (^)

fact CannotBeOwnPreq {
    all c : Course | c ! In (c.(^prereqs))
}

Course$0 = { Course$0 }
^prereqs = { Course$0->Course$2, Course$1->Course$0,
             Course$1->Course$2, Course$3->Course$0,
             Course$3->Course$1, Course$3->Course$2 }

Select all triplets where the two courses are the same
{
  Course$0->Course$0->Course$2, Course$0->Course$1->Course$0,
  Course$0->Course$1->Course$2, Course$0->Course$3->Course$0,
  Course$0->Course$3->Course$1, Course$0->Course$3->Course$2
}
Transitive Closure (^)

fact CannotBeOwnPreq {
    all c : Course | c ! In (c. (^prereqs))
}

Course$0 = { Course$0 }  
^prereqs = { Course$0->Course$2, Course$1->Course$0,  
            Course$1->Course$2, Course$3->Course$0,  
            Course$3->Course$1, Course$3->Course$2  
             }

Project the result by removing the columns used for selection.
{
   Course$0->Course$0->Course$2, Course$0->Course$1->Course$0,  
   Course$0->Course$1->Course$2, Course$0->Course$3->Course$0,  
   Course$0->Course$3->Course$1, Course$0->Course$3->Course$2  
}
Course$0. ^prereqs = { Course$2 }
Transitive Closure (^)

\(^\text{prereqs}.\text{Course}0 \)/"Courses for which \text{Course}0 is a prereq"\

\(^\text{prereqs} = \{ \text{Course}0->\text{Course}2, \text{Course}1->\text{Course}0, \text{Course}1->\text{Course}2, \text{Course}3->\text{Course}0, \text{Course}3->\text{Course}1, \text{Course}3->\text{Course}2 \}\)

\text{Course}0 = \{ \text{Course}0 \}

\textbf{Join} all elements of \(^\text{prereqs}\) to all elements of \textbf{Course}0 (create triplets)

\{ \text{Course}0->\text{Course}2->\text{Course}0, \text{Course}1->\text{Course}0->\text{Course}0, \text{Course}1->\text{Course}2->\text{Course}0, \text{Course}3->\text{Course}0->\text{Course}0, \text{Course}3->\text{Course}1->\text{Course}0, \text{Course}3->\text{Course}2->\text{Course}0 \}
Transitive Closure (^)

`^prereqs.Course$0 //"Courses for which Course$0 is a prereq"

`^prereqs = { Course$0->Course$2, Course$1->Course$0,
Course$1->Course$2, Course$3->Course$0,
Course$3->Course$1, Course$3->Course$2
}

Course$0 = { Course$0 }

Select all triplets where the two courses are the same{
Course$0->Course$2->Course$0, Course$1->Course$0->Course$0,
Course$1->Course$2->Course$0, Course$3->Course$0->Course$0,
Course$3->Course$1->Course$0, Course$3->Course$2->Course$0
}
Transitive Closure (^)

^{prereqs}.Course$0 //"Courses for which Course$0 is a prereq"
^{prereqs} = { Course$0->Course$2, Course$1->Course$0,
              Course$1->Course$2, Course$3->Course$0,
              Course$3->Course$1, Course$3->Course$2
            }
Course$0 = { Course$0 }

Project the result by removing the columns used for selection.
{
  Course$0->Course$2->Course$0, Course$1->Course$0->Course$0,
  Course$1->Course$2->Course$0, Course$3->Course$0->Course$0,
  Course$3->Course$1->Course$0, Course$3->Course$2->Course$0
}
^{prereqs}.Course$0 = { Course$1, Course$3 }