SWEN-220 Math Models

Fields & Relations

Joins
Simplified Course Model

```plaintext
enum Person {Joe, Tim, Jane, Sally}

abstract sig Course{
    enrolled : set Person
}

one sig SWEN261, SWEN220, SWEN262 extends Course{}

run {}
```
One Solution
One Solution

What is `SWEN220.enrolled`?
One Solution

What is Course.enrolled?
One Solution

What is enrolled.Tim?
One Solution

What is `enrolled.Person`?
Join Operator Definition

Assume $r_1$ and $r_2$ are relations.

- Further assume $r_1$ has $m$ columns ($m$-ary),
- And $r_2$ has $n$ columns ($n$-ary).

Then

- $r_1 \cdot r_2$ only makes sense if the last (rightmost) column of $r_1$ and the first (leftmost) column of $r_2$ are from the same top-level signature.
- $r_1 \cdot r_2$ has $(m + n - 2)$ columns
Join Operator Definition

**SWEN220.enrolled**

- **r1** (unary)
  - `{ SWEN220$0 }`
  - last (rightmost) column = `Course`

- **r2** (binary)
  - `{ SWEN220$0->Joe$0, ... }`
  - first (leftmost) column
The Magical Join Operator (.)

**SWEN220.enrolled**

SWEN220 = \{ SWEN220$0 \}

enrolled = \{ SWEN220$0\rightarrow Joe$0, SWEN220$0\rightarrow Tim$0, SWEN261$0\rightarrow Tim$0, SWEN261$0\rightarrow Jane$0 \}

**Join** all elements of **SWEN220** to all elements of **enrolled** (create triplets)

\{
    SWEN220$0\rightarrow SWEN220$0\rightarrow Joe$0, SWEN220$0\rightarrow SWEN220$0\rightarrow Tim$0, SWEN220$0\rightarrow SWEN261$0\rightarrow Tim$0, SWEN220$0\rightarrow SWEN261$0\rightarrow Jane$0
\}
The Magical Join Operator (.)

SWEN220.enrolled

\[
\text{SWEN220} = \{ \text{SWEN220}\$0 \} \\
\text{enrolled} = \{ \text{SWEN220}\$0 \rightarrow \text{Joe}\$0, \text{SWEN220}\$0 \rightarrow \text{Tim}\$0, \\
\phantom{enrolled} \text{SWEN261}\$0 \rightarrow \text{Tim}\$0, \text{SWEN261}\$0 \rightarrow \text{Jane}\$0 \} \\
\]

Select all triplets where the two courses are the same

\[
\{ \\
\phantom{Select } \text{SWEN220}\$0 \rightarrow \text{SWEN220}\$0 \rightarrow \text{Joe}\$0, \text{SWEN220}\$0 \rightarrow \text{SWEN220}\$0 \rightarrow \text{Tim}\$0, \\
\phantom{Select } \text{SWEN220}\$0 \rightarrow \text{SWEN261}\$0 \rightarrow \text{Tim}\$0, \text{SWEN220}\$0 \rightarrow \text{SWEN261}\$0 \rightarrow \text{Jane}\$0 \\
\}
\]
The Magical Join Operator (.)

\[
SWEN220.enrolled
\]

\[
SWEN220 = \{ \text{SWEN220}\$0 \} \\
enrolled = \{ \text{SWEN220}\$0->\text{Joe}\$0, \text{SWEN220}\$0->\text{Tim}\$0, \text{SWEN261}\$0->\text{Tim}\$0, \text{SWEN261}\$0->\text{Jane}\$0 \}
\]

**Project** the result by removing the columns used for selection.

\[
\{ \\
\text{SWEN220}\$0->\text{SWEN220}\$0->\text{Joe}\$0, \text{SWEN220}\$0->\text{SWEN220}\$0->\text{Tim}\$0, \text{SWEN220}\$0->\text{SWEN261}\$0->\text{Tim}\$0, \text{SWEN220}\$0->\text{SWEN261}\$0->\text{Jane}\$0 \\
\}
\]
The Magical Join Operator (.)

\[ \text{SWEN220.enrolled} = \{ \text{Joe}^0, \text{Tim}^0 \} \]
The Magical Join Operator (.)

Course\_enrolled

Course = \{ SWEN220$0, SWEN261$0, SWEN262$0 \}
enrolled = \{ SWEN220$0->Joe$0, SWEN220$0->Tim$0, SWEN261$0->Tim$0, SWEN261$0->Jane$0 \}

Join all elements of Course to all elements of enrolled (create triplets)

\{
SWEN220$0->SWEN220$0->Joe$0, SWEN220$0->SWEN220$0->Tim$0, SWEN220$0->SWEN261$0->Tim$0, SWEN220$0->SWEN261$0->Jane$0
SWEN261$0->SWEN220$0->Joe$0, SWEN261$0->SWEN220$0->Tim$0, SWEN261$0->SWEN261$0->Tim$0, SWEN261$0->SWEN261$0->Jane$0
SWEN262$0->SWEN220$0->Joe$0, SWEN262$0->SWEN220$0->Tim$0, SWEN262$0->SWEN261$0->Tim$0, SWEN262$0->SWEN261$0->Jane$0
\}
The Magical Join Operator (.)

Course.enrolled

Course = { SWEN220$0, SWEN261$0, SWEN262$0 }
enrolled = { SWEN220$0->Joe$0, SWEN220$0->Tim$0, SWEN261$0->Tim$0, SWEN261$0->Jane$0 }

Select all triplets where the two courses are the same

{ SWEN220$0->SWEN220$0->Joe$0, SWEN220$0->SWEN220$0->Tim$0, SWEN220$0->SWEN261$0->Tim$0, SWEN261$0->SWEN220$0->Joe$0, SWEN261$0->SWEN220$0->Tim$0, SWEN261$0->SWEN261$0->Tim$0, SWEN261$0->SWEN261$0->Jane$0, SWEN262$0->SWEN220$0->Joe$0, SWEN262$0->SWEN220$0->Tim$0, SWEN262$0->SWEN261$0->Tim$0, SWEN262$0->SWEN261$0->Jane$0 }
The Magical Join Operator (.)

Course.enrolled
Course = { SWEN220$0, SWEN261$0, SWEN262$0 }
enrolled = { SWEN220$0->Joe$0, SWEN220$0->Tim$0, SWEN261$0->Tim$0, SWEN261$0->Jane$0 }

Project the result by removing the columns used for selection.

{ SWEN220$0->SWEN220$0->Joe$0, SWEN220$0->SWEN220$0->Tim$0, SWEN220$0->SWEN261$0->Tim$0, SWEN261$0->SWEN220$0->Joe$0, SWEN261$0->SWEN261$0->Tim$0, SWEN261$0->SWEN261$0->Jane$0, SWEN262$0->SWEN220$0->Joe$0, SWEN262$0->SWEN220$0->Tim$0, SWEN262$0->SWEN261$0->Tim$0, SWEN262$0->SWEN261$0->Jane$0 }
The Magical Join Operator (.)

Course.enrolled = {Joe$0, Tim$0, Jane$0}
The Magical Join Operator (.)

\[ \text{enrolled} \cdot \text{Tim} \]

\[
\text{Tim} = \{ \text{Tim}^0 \} \\
\text{enrolled} = \{ \text{SWEN220}^0 \rightarrow \text{Joe}^0, \text{SWEN220}^0 \rightarrow \text{Tim}^0, \text{SWEN261}^0 \rightarrow \text{Tim}^0, \text{SWEN261}^0 \rightarrow \text{Jane}^0 \} \\
\text{Join} \text{ all elements of } \text{enrolled} \text{ to all elements of } \text{Tim} \text{ (create triplets)} \\
\{ \text{SWEN220}^0 \rightarrow \text{Joe}^0 \rightarrow \text{Tim}^0, \text{SWEN220}^0 \rightarrow \text{Tim}^0 \rightarrow \text{Tim}^0, \text{SWEN261}^0 \rightarrow \text{Tim}^0 \rightarrow \text{Tim}^0, \text{SWEN261}^0 \rightarrow \text{Jane}^0 \rightarrow \text{Tim}^0 \} \]
enrolled.Tim

Tim =  
{ Tim$0 } 

enrolled =  
{  SWEN220$0->Joe$0,  SWEN220$0->Tim$0,  
    SWEN261$0->Tim$0,  SWEN261$0->Jane$0 } 

Select all triplets where the two Persons are the same 

{  
    SWEN220$0->Joe$0->Tim$0,  SWEN220$0->Tim$0->Tim$0,  
    SWEN261$0->Tim$0->Tim$0,  SWEN261$0->Jane$0->Tim$0  
}
The Magical Join Operator (.)

enrolled.Tim

Tim = { Tim$0 }
enrolled = { SWEN220$0->Joe$0, SWEN220$0->Tim$0, SWEN261$0->Tim$0, SWEN261$0->Jane$0 }

Project the result by removing the columns used for selection.

{ SWEN220$0->Joe$0->Tim$0, SWEN220$0->Tim$0->Tim$0, SWEN261$0->Tim$0->Tim$0, SWEN261$0->Jane$0->Tim$0 }
enrolled.Tim = \{ \text{SWEN261}\$0, \text{SWEN220}\$0 \}
The Magical Join Operator (.)

enrolled. Person
Players – Teams - Cities

```alloy
some sig Player {
    playsFor : Team
}

some sig City{}

some sig Team {
    location : City
}

run {} for 3
```

Alloy Analyzer 4.2 (build date: 2011-08-16 22:00:00)

Warning: Alloy4 defaults to SAT4J since it is platform-specific. For faster performance, go to Options menu and try the other solver.

If these native solvers fail on your computer, remember that Alloy has a Java Wrapper that provides a Jikes and Z3 back-end.

Executing "Run run$1 for 3"
Solver=sat4j Bitwidth=0 MaxSeq=0 SkolemDepth=1 S 223 vars. 27 primary vars. 330 clauses. 145ms.
Instance found. Predicate is consistent. 48ms.
Navigation
Player3.playsFor.location?
(Player$2 + Player$1 + Player$0).playsFor.location?
Navigation

playsFor.location.City$1?
Navigation

playsFor.location?
Navigation

playsFor.location?

playsFor =  \{ P4\rightarrow T0, P0\rightarrow T3, P3\rightarrow T1, P2\rightarrow T1, P1\rightarrow T2 \}

location =  \{ T0\rightarrow C2, T3\rightarrow C1, T1\rightarrow C1, T2\rightarrow C0 \}
Navigation

playsFor.location?

playsFor = \{ \text{P4} -> \text{T0}, \text{P0} -> \text{T3}, \text{P3} -> \text{T1}, \text{P2} -> \text{T1}, \text{P1} -> \text{T2} \}

location = \{ \text{T0} -> \text{C2}, \text{T3} -> \text{C1}, \text{T1} -> \text{C1}, \text{T2} -> \text{C0} \}

Join to get quadruples

\{ 
\text{P4} -> \text{T0} -> \text{T0} -> \text{C2}, \text{P4} -> \text{T0} -> \text{T3} -> \text{C1}, \text{P4} -> \text{T0} -> \text{T1} -> \text{C1}, \text{P4} -> \text{T0} -> \text{T2} -> \text{C0}, \\
\text{P0} -> \text{T3} -> \text{T0} -> \text{C2}, \text{P0} -> \text{T3} -> \text{T3} -> \text{C1}, \text{P0} -> \text{T3} -> \text{T1} -> \text{C1}, \text{P0} -> \text{T3} -> \text{T2} -> \text{C0}, \\
\text{P3} -> \text{T1} -> \text{T0} -> \text{C2}, \text{P3} -> \text{T1} -> \text{T3} -> \text{C1}, \text{P3} -> \text{T1} -> \text{T1} -> \text{C1}, \text{P3} -> \text{T1} -> \text{T2} -> \text{C0}, \\
\text{P2} -> \text{T1} -> \text{T0} -> \text{C2}, \text{P2} -> \text{T1} -> \text{T3} -> \text{C1}, \text{P2} -> \text{T1} -> \text{T1} -> \text{C1}, \text{P2} -> \text{T1} -> \text{T2} -> \text{C0}, \\
\text{P1} -> \text{T2} -> \text{T0} -> \text{C2}, \text{P1} -> \text{T2} -> \text{T3} -> \text{C1}, \text{P1} -> \text{T2} -> \text{T1} -> \text{C1}, \text{P1} -> \text{T2} -> \text{T2} -> \text{C0} 
\}
Navigation

playsFor.location?

playsFor = \{ P4\rightarrow T0, P0\rightarrow T3, P3\rightarrow T1, P2\rightarrow T1, P1\rightarrow T2 \}  
location = \{ T0\rightarrow C2, T3\rightarrow C1, T1\rightarrow C1, T2\rightarrow C0 \}  
Select quads where the left column from playsFor equals the right column from location.

\{  
P4\rightarrow T0\rightarrow T0\rightarrow C2, \underline{P4\rightarrow T0 \rightarrow T3 \rightarrow C1}, \underline{P4\rightarrow T0 \rightarrow T1 \rightarrow C1}, \underline{P4\rightarrow T0 \rightarrow T2 \rightarrow C0}, 
P0\rightarrow T3\rightarrow T0\rightarrow C2, \underline{P0\rightarrow T3 \rightarrow T3 \rightarrow C1}, \underline{P0\rightarrow T3 \rightarrow T1 \rightarrow C1}, \underline{P0\rightarrow T3 \rightarrow T2 \rightarrow C0}, 
P3\rightarrow T1\rightarrow T0\rightarrow C2, \underline{P3\rightarrow T1 \rightarrow T3 \rightarrow C1}, \underline{P3\rightarrow T1 \rightarrow T1 \rightarrow C1}, \underline{P3\rightarrow T1 \rightarrow T2 \rightarrow C0}, 
P2\rightarrow T1\rightarrow T0\rightarrow C2, \underline{P2\rightarrow T1 \rightarrow T3 \rightarrow C1}, \underline{P2\rightarrow T1 \rightarrow T1 \rightarrow C1}, \underline{P2\rightarrow T1 \rightarrow T2 \rightarrow C0}, 
P1\rightarrow T2\rightarrow T0\rightarrow C2, \underline{P1\rightarrow T2 \rightarrow T3 \rightarrow C1}, \underline{P1\rightarrow T2 \rightarrow T1 \rightarrow C1}, \underline{P1\rightarrow T2 \rightarrow T2 \rightarrow C0}  
\}
playsFor.location?

playsFor =  { P4->T0, P0->T3, P3->T1, P2->T1, P1->T2 }
location =  { T0->C2, T3->C1, T1->C1, T2->C0 }

Project – eliminate the columns used for the match.

{ 
  P4->T0->T0->C2,  P4->T0->T3->C1,  P4->T0->T1->C1,  P4->T0->T2->C0,
  P0->T3->T0->C2,  P0->T2->T3->C1,  P0->T3->T1->C1,  P0->T3->T2->C0,
  P3->T1->T0->C2,  P3->T1->T3->C1,  P3->T1->T1->C1,  P3->T1->T2->C0,
  P2->T1->T0->C2,  P2->T1->T3->C1,  P2->T1->T1->C1,  P2->T1->T2->C0,
  P1->T2->T0->C2,  P1->T2->T3->C1,  P1->T2->T1->C1,  P1->T2->T2->C0
}
playsFor.location =
{P4→C2, P0→C1, P3→C1, P2→C1, P1→C0}
Join Operator Definition

Assume \( r_1 \) and \( r_2 \) are relations.

- Further assume \( r_1 \) has \( m \) columns (\( m \)-ary),
- And \( r_2 \) has \( n \) columns (\( n \)-ary).

Then

- \( r_1 \cdot r_2 \) only makes sense if the last (rightmost) column of \( r_1 \) and the first (leftmost) column of \( r_2 \) are from the same top-level signature.
- \( r_1 \cdot r_2 \) has \((m + n - 2)\) columns