Connection Instances and Arrays

We have

- multi-dimensional arrays for components
- single dimension for features
- Arrays at different levels of the component hierarchy
- Feature arrays only at the leaves of the component hierarchy
  - At higher levels in the hierarchy the features are not declared as arrays
    - They would reflect the dimensionality of lower levels

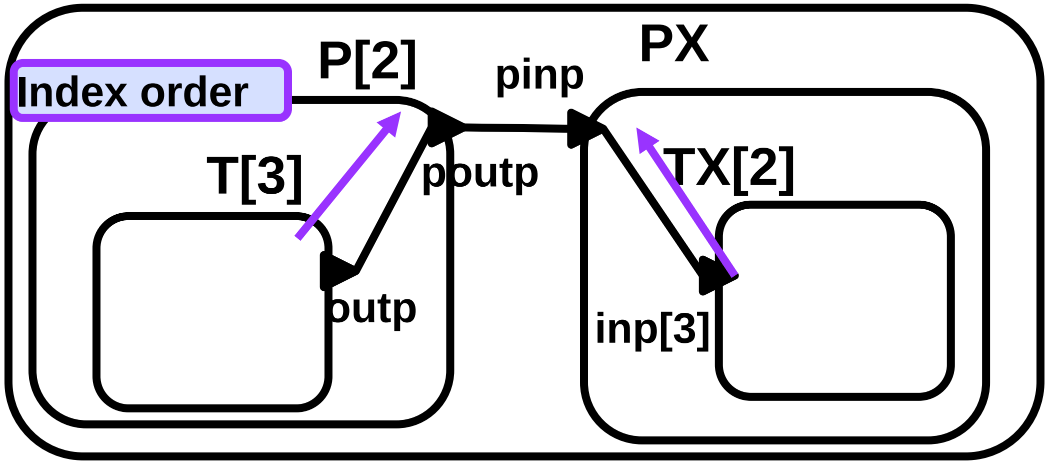
Array declarations at different levels of the hierarchy result in multi-dimensional instance arrays for the leaf components

- We configure connection instances for resulting arrays in instance model

Currently array dimensions not reflected in enclosing interface features

- We do reflect feature aggregation as feature group

Connection Instances and Arrays

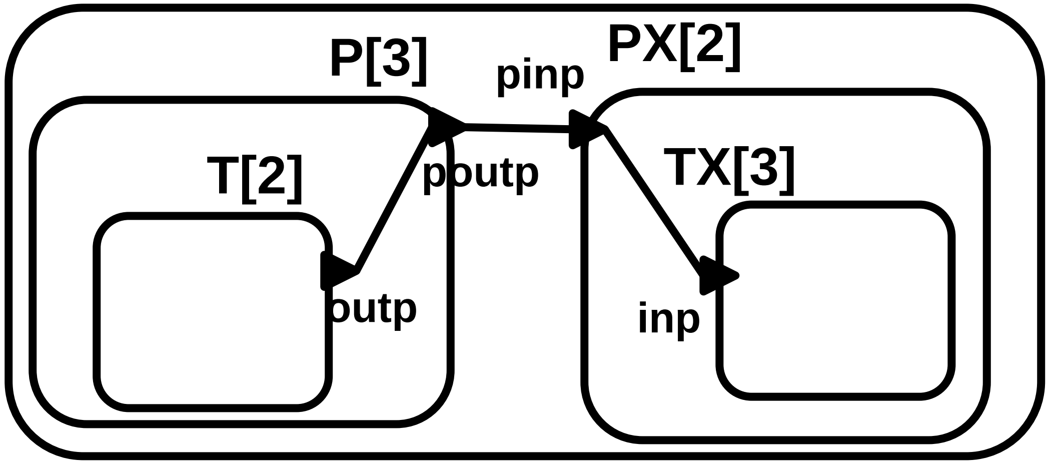


For P.T.outp [3][2] -> PX.TX.inp[3][2]

[1,1] => [2,1]
[3,2] => [1,2]

For P.T [2][3] -> PX.TX[3][2]

[1,1] => [1,1]
[2,1] => [1,2]
[3,1] => [1,3]



Exposing Dimensions in Interface

Approach

- Expose externally visible dimensionality through interface

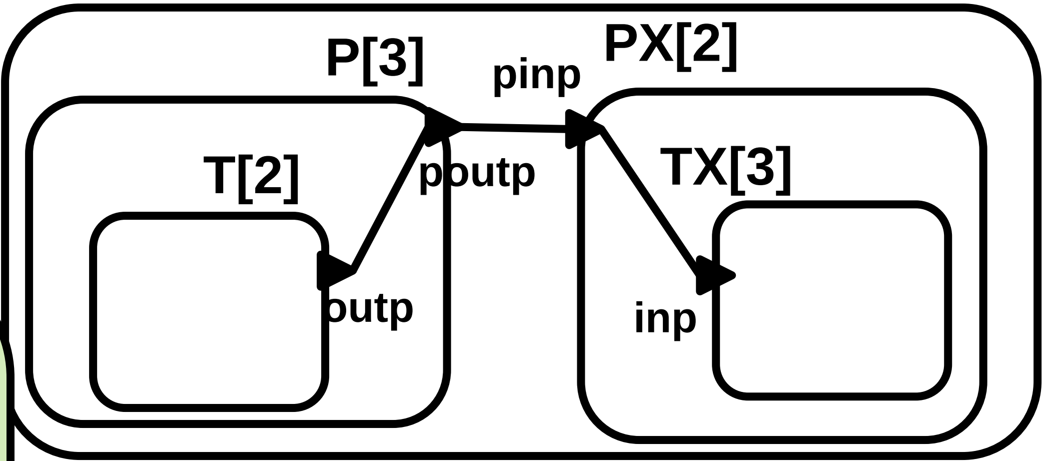
Similar to exposing feature grouping in interface

- Desire to connect elements within nested feature groups at the top level connection

Expose Inner Dimensions as Feature array dimension

```
System p
Features
Poutp: out event port [2]
```

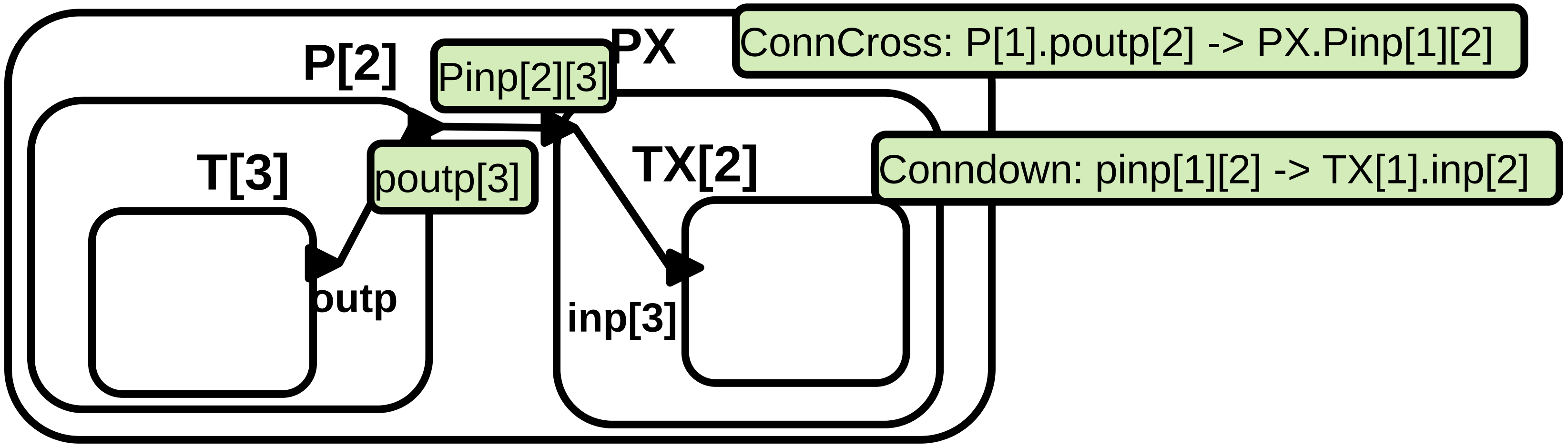
```
System px
Features
Pinp: in event port [3]
End px;

System implementation px.i
Subcomponents
Tx: system tx[3];
Connections
 pinp[1] -> Tx[1].inp;
 pinp[2] -> Tx[3].inp;
 pinp[3] -> Tx[2].inp;
Or
Pinp[] -> Tx[].inp; -- one-to-one
```



```
System implementation top.i
subcomponents
 p: system P[3];
Px: system PX[2];
Connections
C1: p[1].poutp[2] -> px[2].pinp[1];
```

Combination of feature indices (inner dimensions) and subcomponent indices

Connection Instances and Arrays


P[2]
Pinp[2][3]
PX
ConnCross: P[1].poutp[2] -> PX.Pinp[1][2]
T[3]
poutp[3]
TX[2]
Conndown: pinp[1][2] -> TX[1].inp[2]
outp
inp[3]

Reusable Index Mapping

Inline index mappings

- Option 1: Individual connection declarations:

```
Conn1: port sub1.lfea1[1,2] -> sub2.rfea1[2,1];
Conn2: port sub1.lfea1[2,1] -> sub2.rfea2[1,2];
```

- Option 2: mapping inline with interface connection:

```
Conn1: port sub1 -> sub2
       {[1,2] == [2,1], [2,1] == [1,2]};
```

Reusable index mapping for

```
map1: mapping
[1,2] == [2,1], [2,1] == [1,2]
end mapping ;
```

Connections on array subsets

Systems as arrays

```
  Src: system s[10];
  Dst1: system a[3];
  Dst2: system b[7];

  Conn1: Src[1..3].p -> Dst1[1..3].p ;
Conn2: Src[4..10].p -> Dst2[1..7].p ;

  Map1: Src[1..3].p -> extp1[1..3] ;
Map2: Src[4..10].p -> extp2 ;
```

Connection Patterns

Applicable to Cross connections

- Same as in V2: pattern across all dimensions
- One-to-one, all-to-all, next, previous: within a dimension
- Changing dimension order (e.g., first to second & vice versa)

Applicable to up/down connections

- Primary pattern: one-to-one
- Change in dimensionality: X[10].p == outerp[5][2]

Array Dimensions and Configurations

Configuration and dimension values

- Sizes can be configured in
- Dimensionality may be predefined in outer feature dimensions
  - Subcomponent configuration cannot change dimensionality
  - May configure dimension size value to 1