

Key Particle methods

```
class Particle {
public:
    virtual operator int ();           casts to standard types
    virtual operator float ();        convert particles to data
    virtual operator double ();       and also change types.
    ... etc...

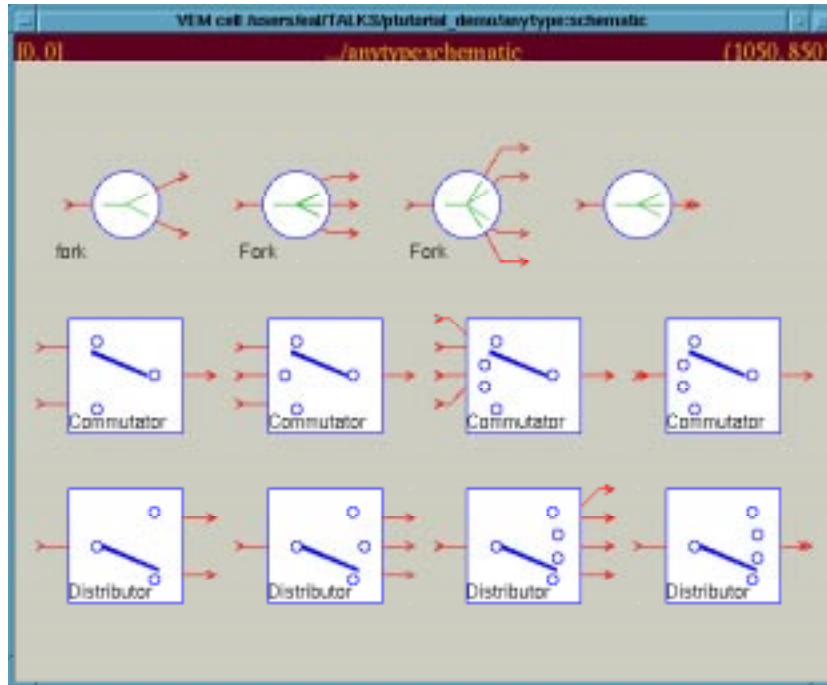
    virtual StringList print ();      generate ASCII
    virtual Particle& initialize();   reset to "zero"

    // Load the Particle with data
    virtual void operator << (int);
    virtual void operator << (double);
    ... etc...

    // Copy a Particle
    virtual Particle* clone();

    // compare two particles
    virtual int operator == (const Particle&);
```

Anytype



Fork:
replicate a particle

Commutator:
merge particle streams

Distributor:
split particle streams

Red stems indicate stars that operate on any particle. The code inside refers only to type "Particle," so these stars are polymorphic.

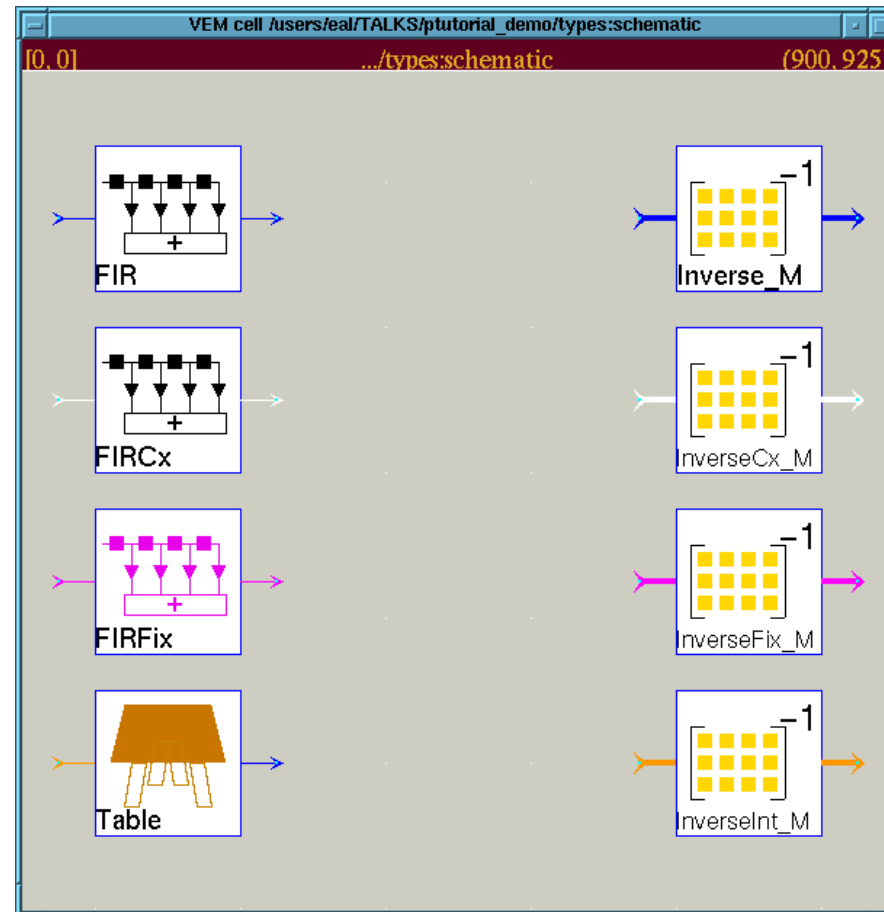
Color Hints at Data Types

blue = float (double)

white = complex

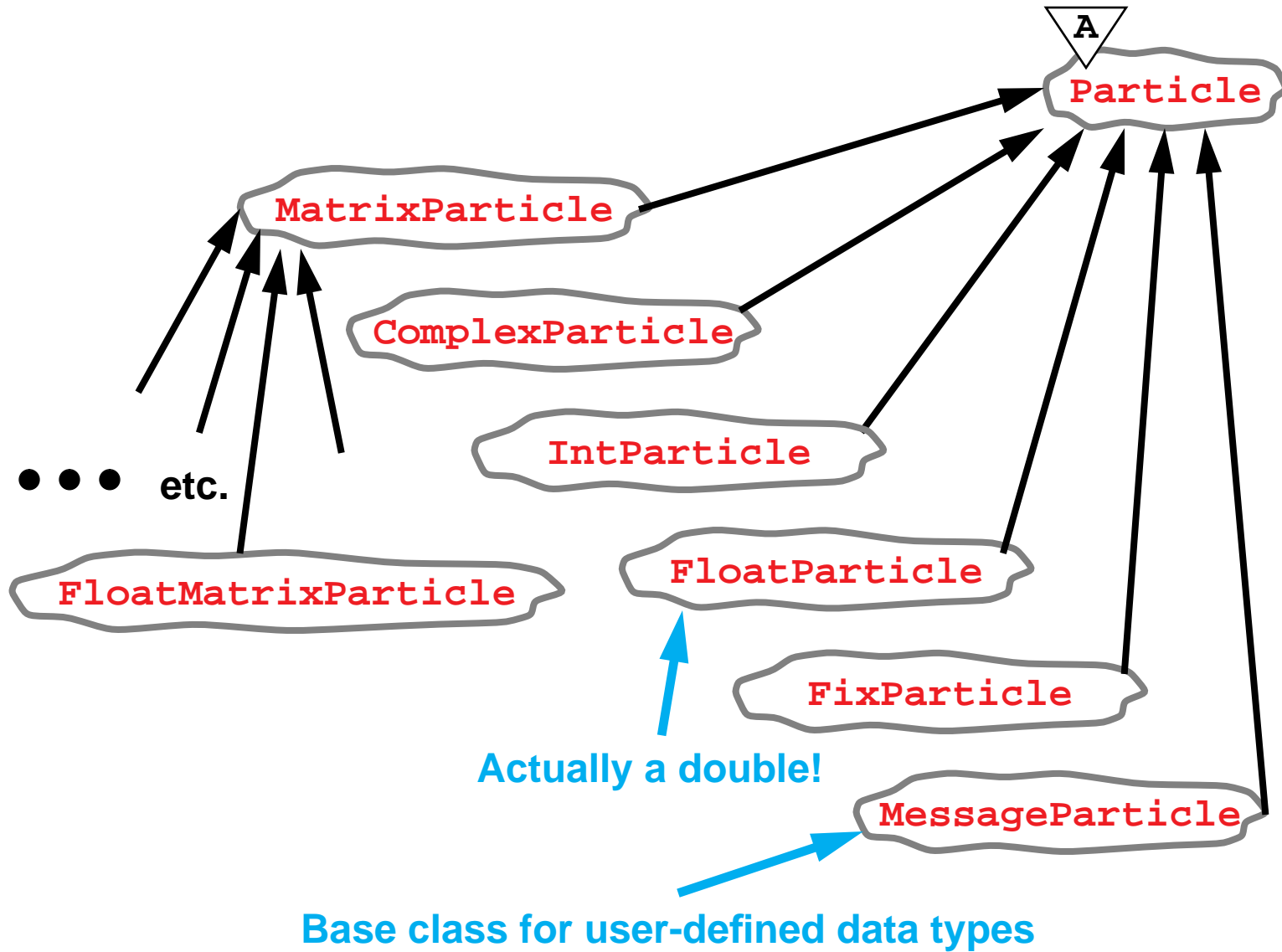
magenta = fixed point

orange = integer

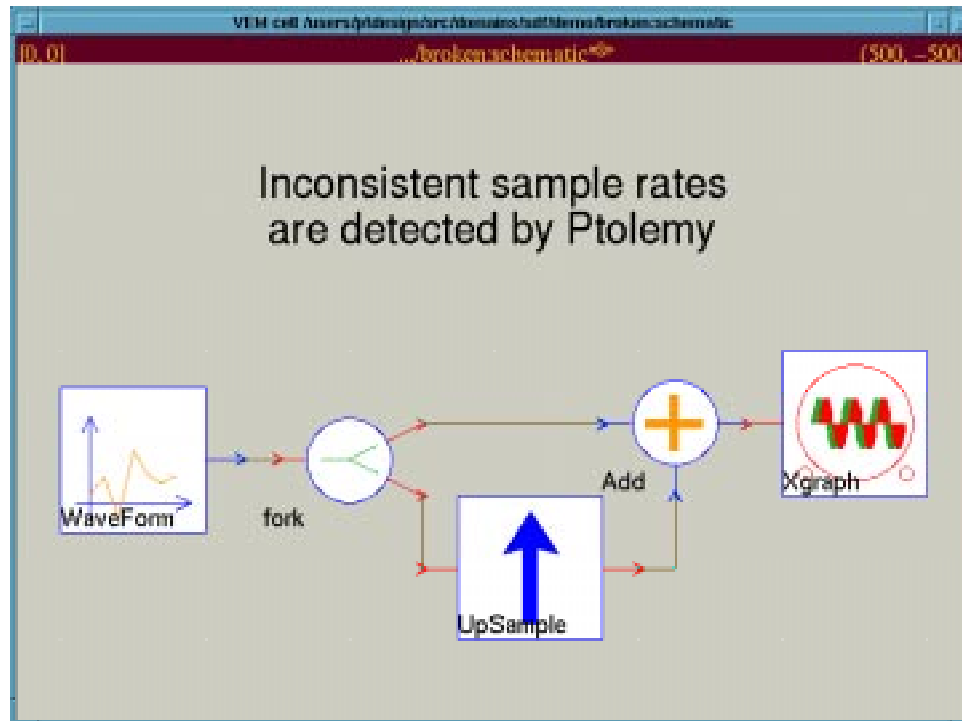


thicker stems indicate matrices

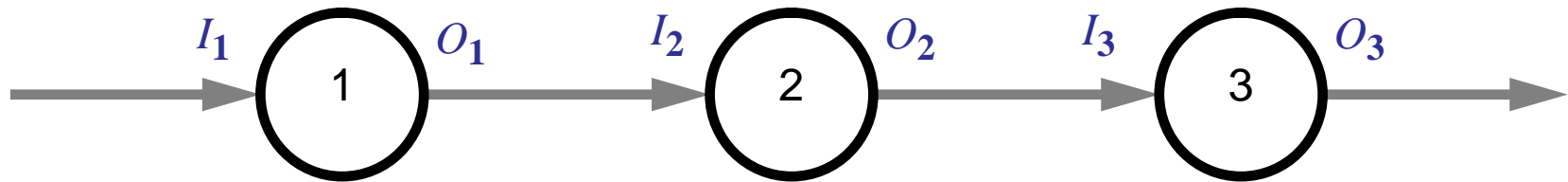
Data Types in Ptolemy



Example of an Inconsistent System



Consistency



Balance equations:

$$r_1 O_1 = r_2 I_2$$

$$r_2 O_2 = r_3 I_3$$

Solve for the smallest integers r_i .
















Then schedule according to data dependencies until repetitions r_i have been met for all actors.

The balance equations have no solution if the graph is *inconsistent*.

Some SDF Stars

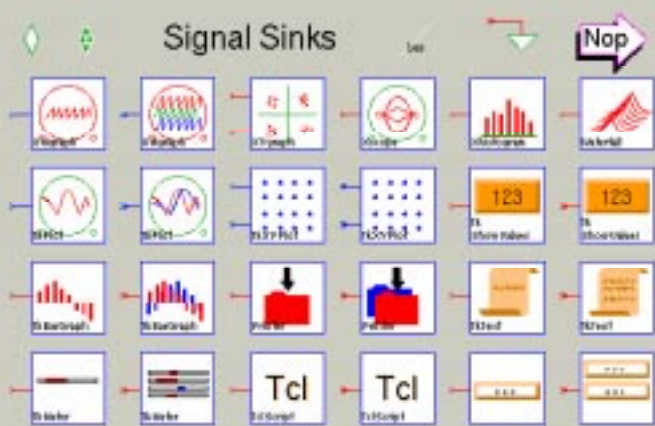
VEM cell \$PTOLEMY/src/domains/sdf/cons/main.pal:scher
[0, 0] .../main.pal:schematic (400, -2050)

SDF Stars

-  Signal Sources
-  Signal Sinks
-  Arithmetic
-  Nonlinear Functions
-  Logic
-  Control
-  Conversion
-  Signal Processing
-  Image Processing
-  Communications
-  Matrix Functions
-  Matlab Functions
-  Higher Order
-  User Contributions 

VEM cell /users/ptdesign/src/domains/sdf/cons/sinks.pal:schematic
[0, 0] .../sinks.pal:schematic (650, 1000)

Signal Sinks

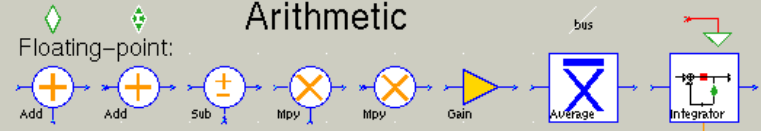


To customize the number of inputs of multi-input stars, use the Nop stars, accessible through the icon on the upper right.


VEM cell /users/ptdesign/src/domains/sdf/cons/arithmatic.pal:schematic
[0, 0] .../arithmatic.pal:schematic (825, 925)

Arithmetic


Floating-point:




Complex:



Fixed-point:















Integer:



Some SDF Demos

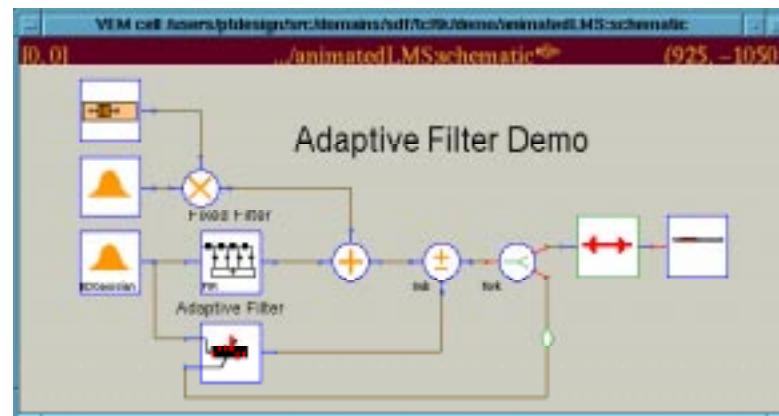
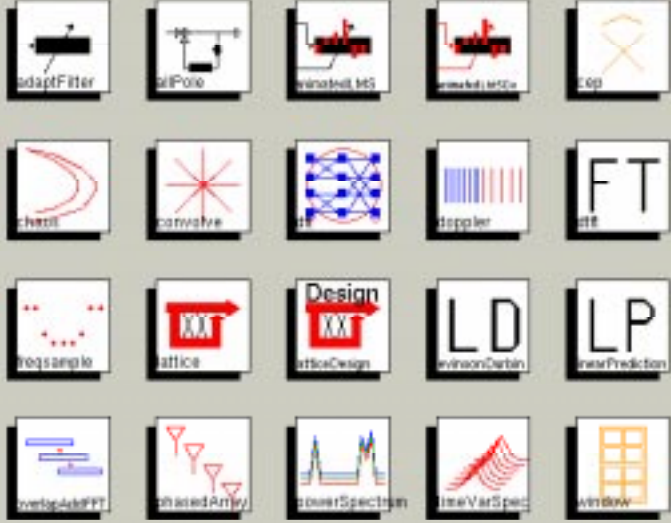
VEM cell /users/ptdesign/src/domains/sdf/demo/init
[0, 0] .../init.pal:schematic (350, -1850)

SDF Demos
Synchronous dataflow (SDF) is used to model signal processing systems with deterministic control flow.

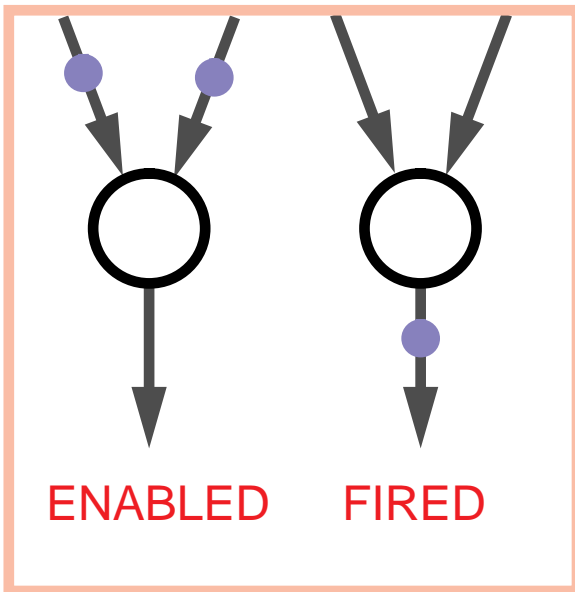
-  Basic
-  Multirate
-  Communications
-  Signal Processing
-  Sound
-  Image Processing
-  Fixed-Point Demos
-  Tcl/Tk Graphics Demos
-  Matrix Demos
-  Matlab Demos
-  Higher-Order Functions
-  Scripted Runs

VEM cell /users/ptdesign/src/domains/sdf/demos/bsp.pal:schematic
[0, 0] .../bsp.pal:schematic (825, 800)

Signal Processing Systems



Synchronous Dataflow



Properties

- Flow of control is predictable at compile time
- Schedule can be constructed once and repeatedly executed
- Suitable for synchronous multirate signal processing

