Tolerant (Parallel) Programming with F-Nets and Software Cabling

David C. DiNucci
MRJ Technology Solutions
Parallel Tools Team
NASA Ames Research Center
www.nas.nasa.gov/Tools/

MRJ Technology Solutions

What is an Algorithm?

Derived from Knuth et al:

Algorithm: A means of expressing a computation
Computation: A sequence of operations

Structured algorithms have proximity property:

Algorithm

```
while a
   b;c;d;
   if e then
      f;g;h
   else
      i
   endif
endwhile
```

Computation

```
a,b,c,d,e,f,g,h,a,b,c,d,e,

i,a,b,c,d,i,...
```

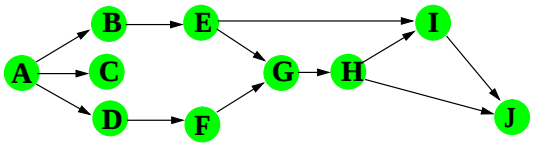

What is a Parallel Algorithm

Obviously,

Parallel Algorithm: A means of expressing a parallel computation

and, derived from Milner et al:

Parallel Computation: A partial ordering of operations

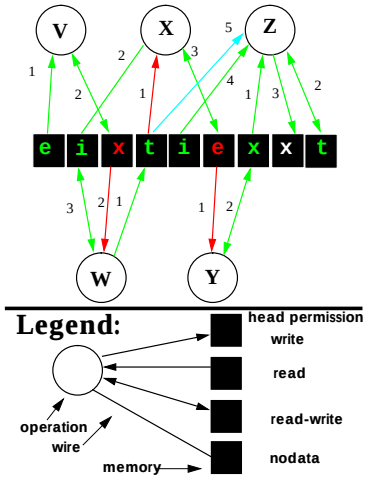


A precedes B
B precedes H
B and D?
B and F?

A seemingly reasonable approach:

Fold up the partial ordering somehow, and allow it to unfold during execution.

F-Nets: Model to Represent Par. Algs

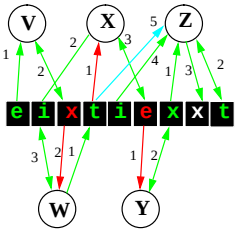


Firing Function for W

Index 2	Index 3	Contents 1	Contents 2	Contents 3
e	e	e	x	e
e	x	e	x	x
e	i	e	x	i
e	t	e	x	t
x	e	x	x	x
x	x	x	x	i
x	i	x	x	t
x	t	x	x	e
i	e	i	x	i
i	x	i	x	t
i	i	i	x	e
i	t	i	x	x
t	e	t	x	t
t	x	t	x	e
t	i	t	x	x
t	t	t	x	i

F-Nets: Operational Semantics

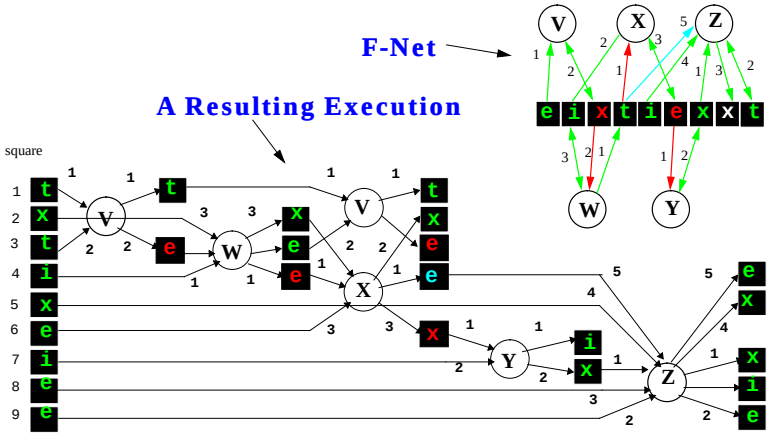
Definition: A **ready** operation is one for which each of the wires is the same color as the memory attached to it.



To Execute an F-Net:

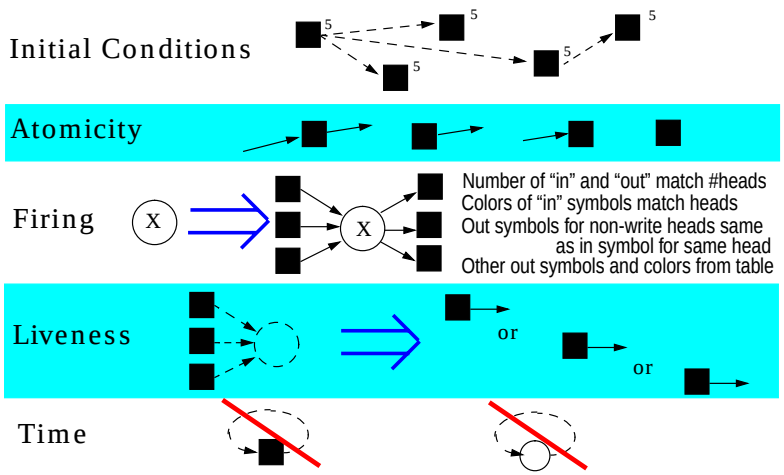
Color all memories green
While a ready operation exists
| Look at symbols at its readable wires
| Find entry corresponding to those symbols in table
| Replace symbol in all writable memories with symbol from table
| Color all memories to match color in table
endWhile

F-Nets: Execution Graph



Can be considered as trace or p.o. of ops or function

F-Nets: Axiomatic Semantics



SC: Software Cabling

SC is a very-high-level graphical parallel "coordination" programming language.

The following description uses a hardware analogy.

Software Cabling: Modules

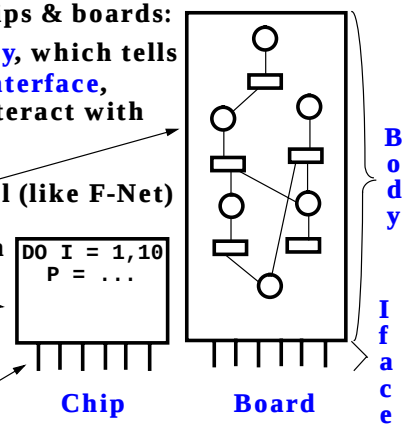
Program consists of chips & boards:

Both consist of a **body**, which tells what to do, and an **interface**, which allows it to interact with its environment

Board body is graphical (like F-Net)

Chip body is written in your favorite language (e.g. Fortran)

Interface consists of **pins** which carry **data** and **signals**.



SC: Chip Body

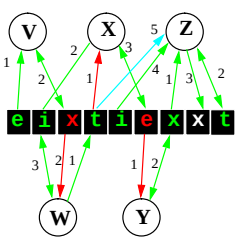
```
function w(in a, inout b, out c)

integer i, ix[20], bx[20], x

do i = 1, 50
  x = ix[i] - bx[i] * a + b
enddo

b = b + x;

post done to b

c = x + 10;

if (c .lt. 20) then
  post again to c
else
  post ready to c
endif
return
```

Chip body is a subroutine (or function) in your favorite language. Pins become arguments.

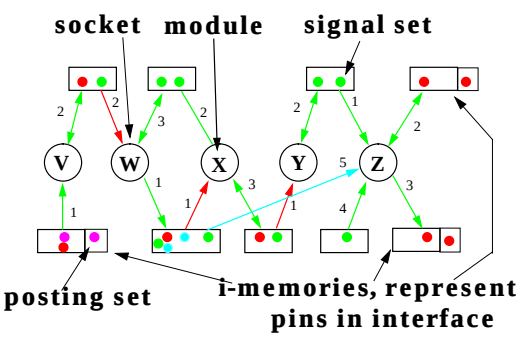
These are the only special statements in the code, and they do not block or otherwise change the local behavior of the code*

*except for optionally making the pin argument inaccessible

SC: Board Body



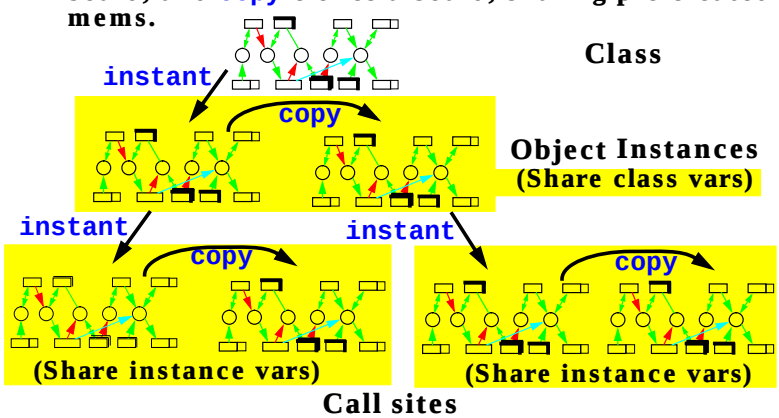
F-Net



SC Board Body

SC: Objects & Classes

Two operations: `instant` pre-creates some mems on a board, and `copy` clones a board, sharing pre-created mems.

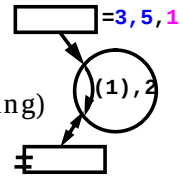


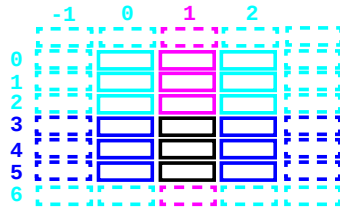
SC: Array Operations

Memory rectangle with "n" hash marks represents n-dimensional array

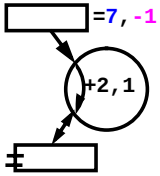
Operations:

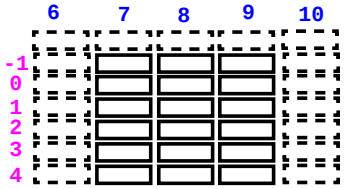
Selection
(aka subscripting)





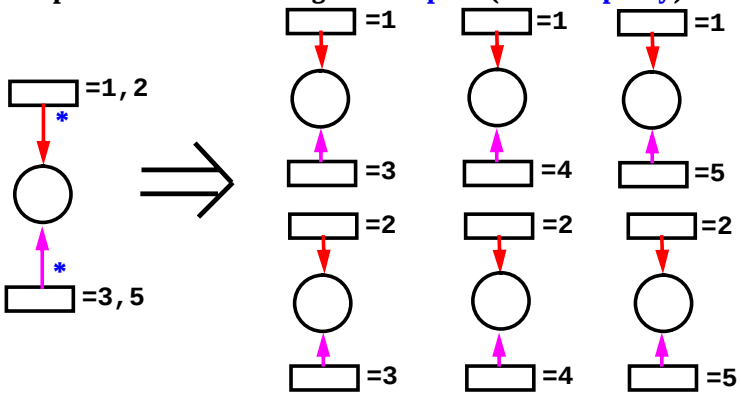
Translation



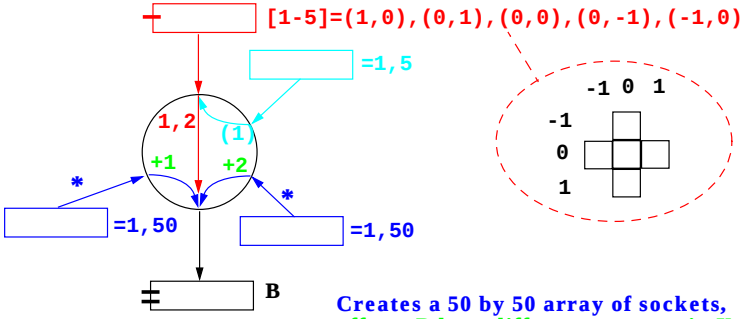


SC: Socket Replication (Dup)

For data parallelism, sockets (and therefore modules) are replicated in SC using the DupAll (also DupAny)



SC: Array Example w/the works



Creates a 50 by 50 array of sockets, offsets B by a different amount in X and Y for each, and accesses a cross-shaped stencil from B centered at 0,0

F-Nets & SC: Summary

- **Not shown: Templates (skeletons)**
- **Formal, functional semantics allow program proving.**
- **Natural bond between theory and practice.**
- **User declares req'd comm semantics, non-determinism.**
- **Latency-hiding on message-passing architectures, no extra copy required on shared-memory architectures, parallelism can be compiled away for seq. execution.**
- **Atomic semantics enable use of seq. tools/reasoning to build/debug code, and implemention of fault tolerance.**
- **Supports object-based programming style (including templates), even when programming in non-OO langs.**
- **Non-deterministic portions detectable syntactically, so only those parts need instrumentation for replay.**