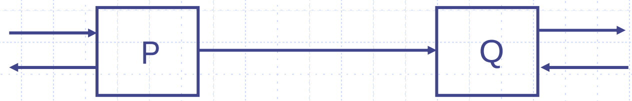
Well-behaved Dataflow Graphs

Arvind
Computer Science & Artificial Intelligence Lab
Massachusetts Institute of Technology

Outline

- Kahnian networks and dataflow
- Streams with holes & Tagged interpretation
- Well behaved graphs

Kahnian Networks



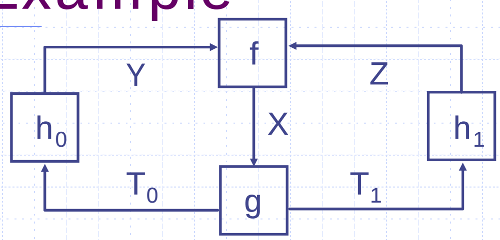
- Computing stations connected by *unbounded, FIFO channels*
- Each station executes a *sequential program*

  wait(ch): blocking read from a channel

  send(x,ch): non blocking

  a station either *blocks* for an input on a specific channel or *computes*

An Example



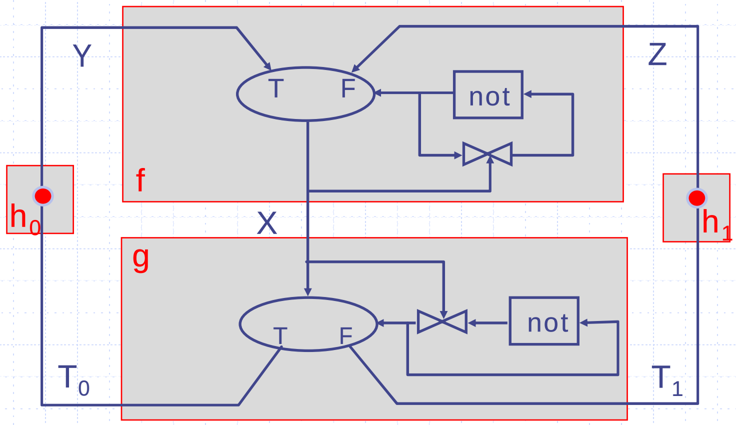
$X = f(Y,Z) \;\; || \;\; T_0 , T_1 = g(X) \;\; || \;\; Y = h_0(T_0) \;\; || \;\; Z = h_1(T_1)$

```
Process  f(U,V; W)
  { b= true;
    While  true do
      { i := if  b then wait(U)
              else wait(V);
        print(i);
        send(i,W);
        b := not b} }
```

```
Process  g(U ; V,W)
  { b= true;
    While  true do
      { i := wait(U);
        if  b then  send(i,V)
        else  send(i,W)
        b := not b } }
```

```
Process  hC (U ; V)
  { send(c,V);
    While  true do
      { i := wait(U);
        send(i,V) } }
```

Kahnian Networks & Dataflow

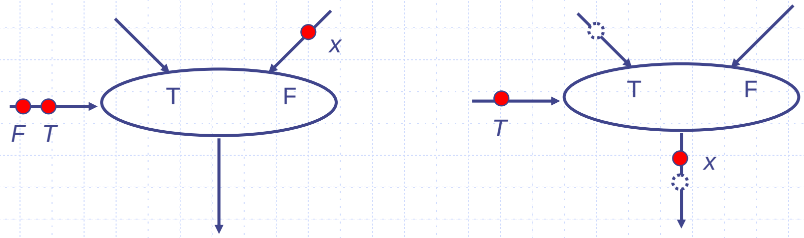


Dataflow Graphs can Express any Kahnian Network
and vice versa

Determinacy

- A computing station in Kahnian network can be viewed as a *monotonic* and *continuous* function from sequences to sequences
- The least fixed point solutions characterize the I/O behavior of such stations
- *Dataflow operators can have any granularity* and can be expressed in any sequential language

Missing Tokens



x cannot be moved to the output because the token corresponding to T is missing.

How to model stream with “holes” ?

Another Interpretation of DFGs

Streams with “holes”

Streams with missing tokens

$\{v_1, v_2, \perp, v_4, \perp, v_6, \ldots\}$

can be modeled by a set of tokens where tokens carry a tag designating their position in the stream

$\{ <1, v_1>, <2, v_2>, <4, v_4>, <6, v_6> \}$

Tagged Semantics of Operators

add(xs,ys) =
$\{<i,x+y> \mid <i,x> \in xs,\ <i,y> \in ys\}$

T-gate(bs,xs) =
$\{<k,x> \mid <i,T> \in bs,\ <i,x> \in xs,$
$\forall j \leq i\ .\ <j,b_j> \in bs\ ,\ k = \text{T-Cnt}(bs,i)\}$

merge(bs,xs,ys) =
$\{<i,x> \mid <i,T> \in bs,\ <k,x> \in xs,$
$\forall j \leq i\ .\ <j,b_j> \in bs,\ k = \text{T-Cnt}(bs,i)\}$
$\cup\ \{<i,y> \mid <i,F> \in bs,\ <k,y> \in ys,$
$\forall j \leq i\ .\ <j,b_j> \in bs,\ k = \text{F-Cnt}(bs,i)\}$

$D_a(xs)$ =
$\{<i+1,x> \mid <i,x> \in xs\} \cup \{<1,a>\}$

Ordering on Streams with Holes

Stream with holes: $\{\ <i,\ v_i>,\ <j,\ v_j>,\ <k,\ v_k>\ \}$

The least element: $\{\ \}$ (aka $\perp$)

The partial order ($\leq$): subset order

It is easy to show that all the operators under the tagged semantics are *monotonic* and *continuous*

$\Rightarrow$ *tagged semantics are also deterministic*

Tagged versus FIFO Interpretation

Theorem:

Suppose the least fixed point of a dataflow program

$X = \{ X_1, \ldots, X_n \}$ in the FIFO interpretation

and

$Y = \{ Y_1, \ldots, Y_n \}$ in the tagged interpretation

then

$X \leq Y$.

Proof: Based on structural induction

1. Show it holds for each operator
2. Show it holds under juxtaposition and iteration

Tagged versus FIFO Interpretation

Theorem:

Suppose the least fixed point of a dataflow program is $X = \{ X_1, \ldots, X_n \}$ in the FIFO interpretation where X_i is the stream associated with a particular arc in the program.

Let $Y = \{ Y_1, \ldots, Y_n \}$ represent the fixed point of the same program in the tagged interpretation, then

$X \leq Y$.

Proof: Based on structural induction

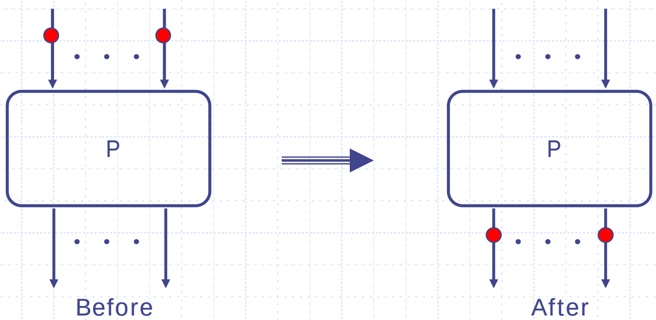
1. Show it holds for each operator
2. Show it holds under juxtaposition and iteration

Tagged interpretation gives more defined answers and has more parallelism

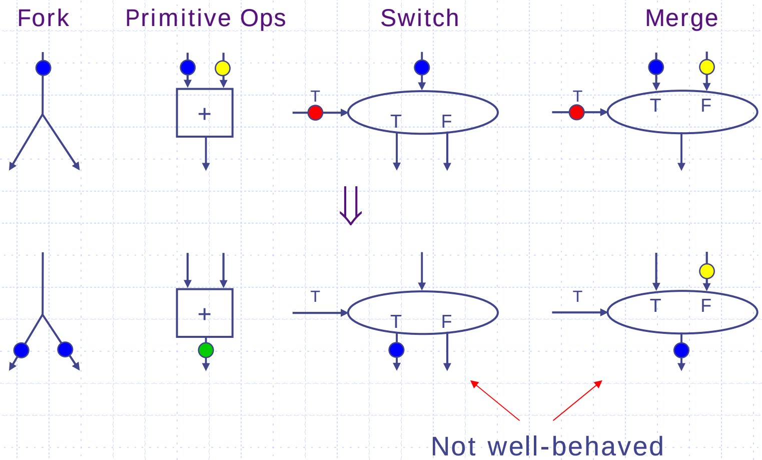
Well Behaved Dataflow Graphs

1. One token on each input arc produces exactly one token on each output arc.

2. The initial distribution of tokens on the arcs is restored.

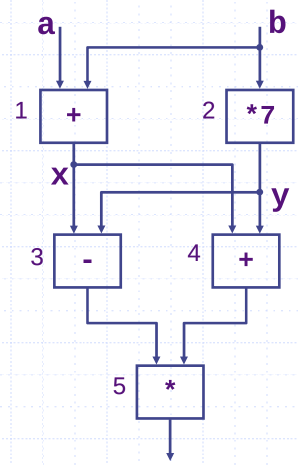


Control Operators are not well-behaved



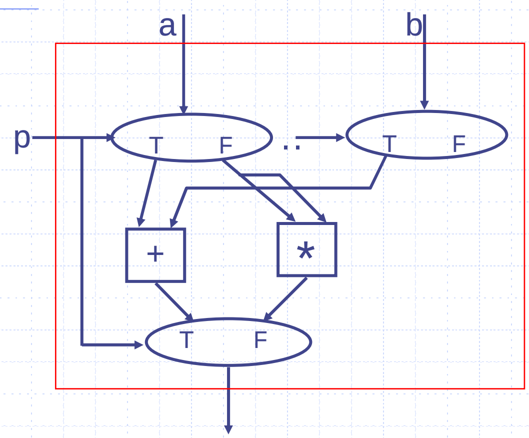
The Block Schema

```
{x = a + b;
 y = b * 7
in
    (x-y) * (x+y)}
```



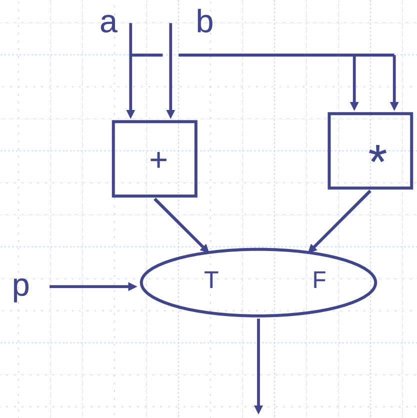
Any acyclic interconnection
of WBGs is a WBG

The Conditional Schema



If p *then* a + b *else* a * a

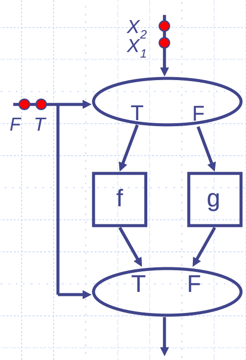
Another Conditional Schema

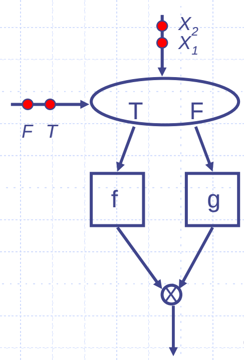


What is wrong with this schema?

If p *then* a + b *else* a * a

Merge Operator is Essential for Determinacy



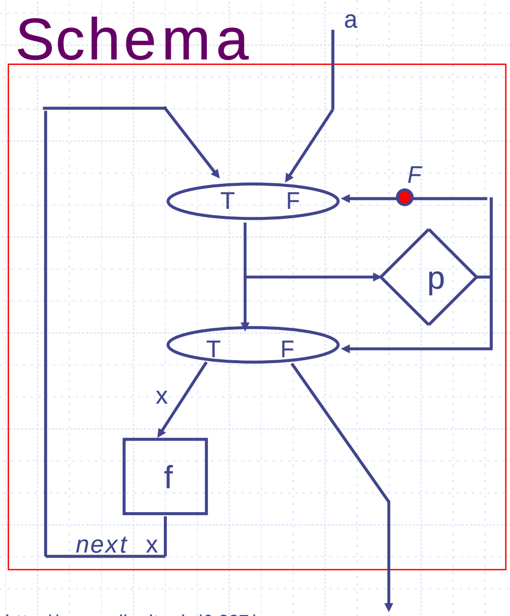


Suppose g(X2) computes much faster than f(X1).

Tokens will come out in the wrong order without the merge operator.

The Loop Schema

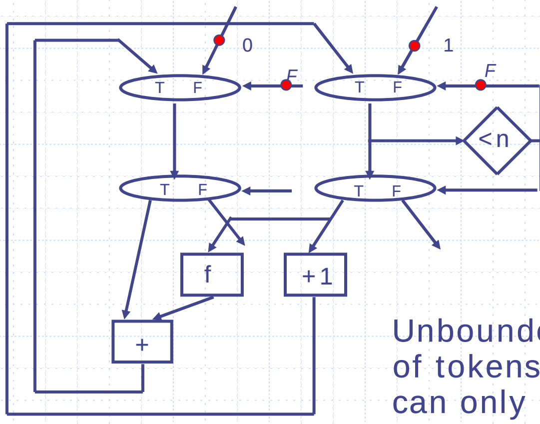
initial x = a
while p(x) *do*
next x = f(x) ;
finally x



Well Behaved Dataflow Graphs (WBGs): Rules to form WBGs

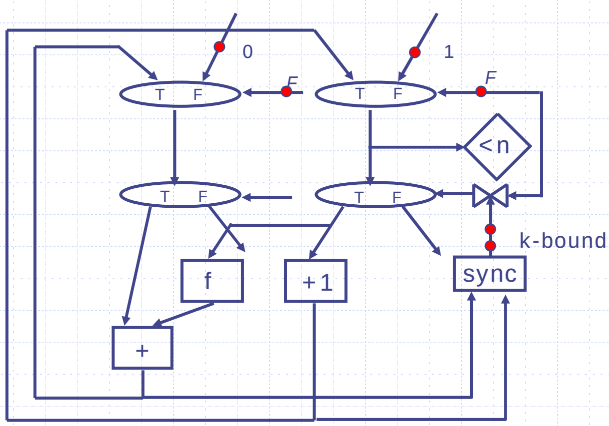
1. *Primitive operators* like + and fork are WBGs (T-gate, F-gate and merge are *not* WBGs).
2. The *block schema*, i.e., an acyclic interconnection of graphs, is a WBG, if all its component graphs are WGBs.
3. The *conditional schema* is a WBG, if the graphs for the True side and False side are WBGs.
4. The *loop schema* is a WBG, if the graphs for the predicate and the body are WBGs.

Unbounded Cyclic Graphs

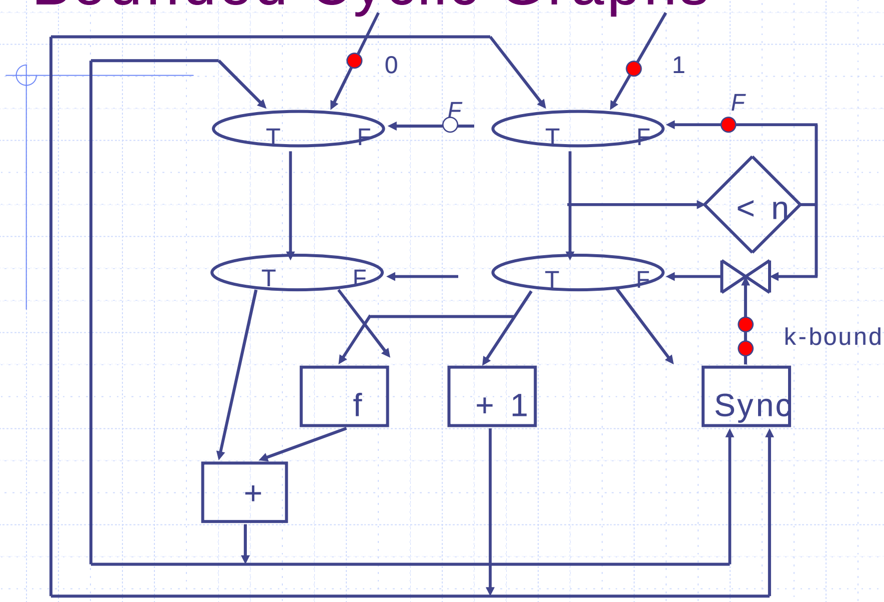


Unbounded number of tokens on an arc can only arise due to cycles.

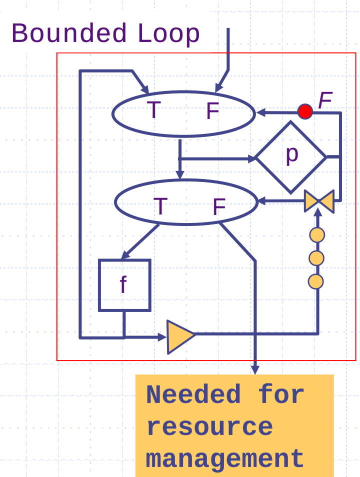
Bounded Cyclic Graphs



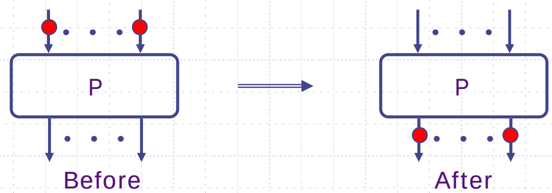
Bounded Cyclic Graphs



Well Behaved Schemas



New Definition of *Well Behavedness*

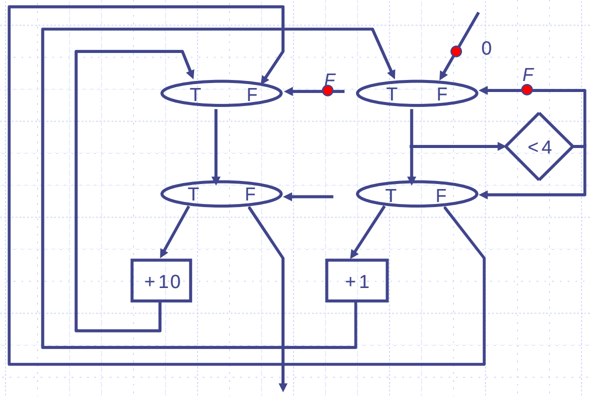


1. One token on each input arc produces exactly one token on each output arc.
2. The initial distribution of tokens on the arcs is restored.
3. No arc can have an unbounded buildup of tokens.

Bounded Cyclic Graphs are *Well Behaved*

- Initial number of tokens at the *gate* input determines the maximum number of tokens on any arc.
- However, loop bounding can alter the "meaning" of a graph, i.e., can cause *deadlock.*
- In general, restricting the number of tokens on an arc causes deadlock.

Can this program deadlock if the number of tokens per arc is restricted?



Static DFGs as a Base Language

- Static DFGs can express all recursively enumerable functions
- Static DFGs are not sufficient as a target for compiling high level languages. Support is lacking for:
  - procedure calls
  - data structures

⇒ *Dynamic Dataflow Graphs*