Recall:
Execution Policy for a Dataflow Actor

Suppose $s \in S^n$ is a concatenation of firing rules,

$$s = u_1, u_2, u_3, \ldots$$

Then the output of the actor is the concatenation of the results of a sequence of applications of the firing function:

$$F_0(s) = \bot_n$$

$$F_1(s) = (\phi(F_0))(s) = f(u_1)$$

$$F_2(s) = (\phi(F_1))(s) = f(u_1) \cdot f(u_2)$$

$$\ldots$$

The problem we address now is scheduling: how to choose which actor to fire when there are choices.
Dataflow variants constrain firing rules and trade off expressiveness and analyzability

- Computation graphs [Karp & Miller - 1966]
- Process networks [Kahn - 1974]
- Static dataflow [Dennis - 1974]
- Dynamic dataflow [Arvind, 1981]
- K-bounded loops [Culler, 1986]
- Synchronous dataflow [Lee & Messerschmitt, 1986]
- Structured dataflow [Kodosky, 1986]
- PGM: Processing Graph Method [Kaplan, 1987]
- Synchronous languages [Lustre, Signal, 1980’s]
- Well-behaved dataflow [Gao, 1992]
- Boolean dataflow [Buck and Lee, 1993]
- Multidimensional SDF [Lee, 1993]
- Cyclo-static dataflow [Lauwereins, 1994]
- Integer dataflow [Buck, 1994]
- Bounded dynamic dataflow [Lee and Parks, 1995]
- Parameterized dataflow [Bhattacharya and Bhattacharyya 2001]
- Structured dataflow (again) [Thies et al. 2002]
- …

Recall: Synchronous Dataflow (SDF)

\[
\Gamma = \begin{bmatrix}
1 & -1 & 0 \\
0 & 2 & -1 \\
2 & 0 & -1
\end{bmatrix}
\]

production/consumption matrix

\[
q = \begin{bmatrix}
q_1 \\
q_2 \\
q_3
\end{bmatrix}
\]

firing vector

\[
\Gamma q = \vec{0} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}
\]

balance equations
Consistent Models

Let $a$ be the number of actors in a connected model. The model is consistent if $\Gamma$ has rank $a - 1$.

If the rank is $a$, then the balance equations have only a trivial solution (zero firings).

When $\Gamma$ has rank $a - 1$, then the balance equations always have a non-trivial solution.

Recall: Boolean Dataflow uses Symbolic Production/Consumption Rates

Imperative equivalent:

```java
while (true) {
    x = f1();
    b = f7();
    if (b) {
        y = f3(x);
    } else {
        y = f4(x);
    }
    f6(y);
}
```

Production and consumption rates are given symbolically in terms of the values of the Boolean control signals consumed at the control port.
Interpretations of Symbolic Rates

- General interpretation: \( p \) is a symbolic placeholder for an unknown.
- Probabilistic interpretation: \( p \) is the probability that a Boolean control input is \textit{true}.
- Proportion interpretation: \( p \) is the proportion of \textit{true} values at the control input in one complete cycle.

\textbf{NOTE:} We do not need numeric values for \( p \). We always manipulate it symbolically.

Symbolic Balance Equations

The two connections above imply the following balance equations:

\[
q_2 \ p = q_3 \\
q_2 \ (1 - p) = q_4
\]
The balance equations have a solution \( q(\vec{p}) \) if and only if \( \Gamma(\vec{p}) \) has rank 6. This occurs if and only if \( p_7 = p_8 \), which happens to be true by construction because signals 7 and 8 come from the same source. The solution is given at the right.

\[
\begin{bmatrix}
1 & 1 \\
1 & 1 \\
1 & 1 \\
1 & 1 \\
1 & 1 \\
1 & 1
\end{bmatrix}
\]
Strong and Weak Consistency

A strongly consistent dataflow model is one where the balance equations have a solution that is provably valid without concern for the values of the symbolic variables.

- The if-then-else dataflow model is strongly consistent.

A weakly consistent dataflow model is one where the balance equations cannot be proved to have a solution without constraints on the symbolic variables that cannot be proved.

- Note that whether a model is strongly or weakly consistent depends on how much you know about the model.

Weakly Consistent Model

This production/consumption matrix has full rank unless $p = 1$.

Unless we know $f_4$, this cannot be verified at compile time.
Another Example of a Weakly Consistent Model

This one requires that actor 7 produce half true and half false (that $p = 0.5$) to be consistent. This fact is derived automatically from solving the balance equations.

Use Boolean Relations

Symbolic variables across logical operators can be related as shown.
Routing of Boolean Tokens

Symbolic variables across switch and select can be related as shown.

\[ p_3 = pr(b_2 \mid b_1) \]
\[ p_4 = pr(b_2 \mid \overline{b_1}) \]

Recall If-Then-Else Pattern

The if-then-else model is strongly consistent and we can give a quasi-static schedule for it:

\[ (1, 7, 2, b?3, \overline{b} \overline{b}4, 5, 6) \]

Solution to the symbolic balance equations:

\[ \mathbf{q}(\mathbf{\hat{p}}) = \begin{bmatrix} 1 \\ 1 \\ p_7 \\ 1 - p_7 \\ 1 \\ 1 \\ 1 \end{bmatrix} \]

\[ \mathbf{\hat{p}} = \begin{bmatrix} p_7 \\ p_8 \end{bmatrix} \]
Quasi-Static Schedules & Traces

A quasi-static schedule is a finite list of guarded firings where:

- The number of tokens on each arc after executing the schedule is the same as before, regardless of the outcome of the Booleans.
- If any arc has a Boolean token prior to the execution of the schedule, then it will have a Boolean token with the same value after execution of the schedule.
- Firing rules are satisfied at every point in the schedule.

A trace is a particular execution sequence.

Solution to the symbolic balance equations:

\[ q(\bar{p}) = \begin{bmatrix} 1 & 1 & p_7 & 1-p_7 & 1 & 1 \end{bmatrix}^T \]

Quasi-static schedule: (1, 7, 2, b?3, !b?4, 5, 6)
Possible trace: (1, 7, 2, 3, 5, 6)
Another possible trace: (1, 7, 2, 4, 5, 6)
Proportion Vectors

- Let $S$ be a trace. E.g. $(1, 7, 2, 3, 5, 6)$
- Let $q_S$ be a repetitions vector for $S$. E.g.
  
  $$q_S = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 1 \end{bmatrix}^T$$
- Let $t_{i,S}$ be the number of TRUEs consumed from Boolean stream $b_i$ in $S$. E.g. $t_{7,S} = 1, t_{8,S} = 1$.
- Let $n_{i,S}$ be the number of tokens consumed from Boolean stream $b_i$ in $S$. E.g. $n_{7,S} = 1, n_{8,S} = 1$.
- Let

  $$\vec{p}_S = \begin{bmatrix} t_{7,S} / n_{7,S} \\ t_{8,S} / n_{8,S} \end{bmatrix}$$

- We want a quasi-static schedule s.t. for every trace $S$ we have $\Gamma(\vec{p}_S)q_S = \vec{0}$.

Proportion Interpretation

Recall the balance equations depend on $\vec{p}$, a vector with one symbolic variable for each Boolean stream that affects consumption production rates:

$$\Gamma(\vec{p})q(\vec{p}) = \vec{0}$$

Under a proportion interpretation, for a trace $S$, $\vec{p}_S$ represents the proportion of TRUEs in $S$. We seek a schedule that always yields traces that satisfy

$$\Gamma(\vec{p}_S)q_S = \vec{0}$$
Proportion Interpretation for If-Then-Else

Quasi-static schedule: (1, 7, 2, b?3, !b?4, 5, 6)
Possible trace: $S = (1, 7, 2, 3, 5, 6)$
$$\bar{p} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}^T$$  
$$q_S = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}^T$$

Another possible trace: (1, 7, 2, 4, 5, 6)
$$\bar{p} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}^T$$  
$$q_S = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}^T$$
Both satisfy the balance equations.

Limitations of Consistency

Consistency is necessary but not sufficient for a dataflow graph to have a bounded-memory schedule. Consider:

[Gao et al. ’92]. This model is strongly consistent. But there is no bounded schedule (e.g., suppose $b_7 = (F, T, T, ...)$.)
Limitations of Consistency

Even out-of-order execution (as supported by tagged-token scheduling [Arvind et al.] doesn’t solve the problem:

Gao’s Example has no Quasi-Static Schedule

Solution to the symbolic balance equations is

\[ q(\bar{p}) = \begin{bmatrix} 2 & 2 & p_7 & 1 - p_7 & 2 & 2 \end{bmatrix} \]

A trace \( S \) with \( N \) firings (\( N \) even) of actor 1 must have

\[ q_S = \begin{bmatrix} N & N & t_{7,S} / 2 & (N - t_{7,S}) / 2 & N & N & N \end{bmatrix} \]

But this cannot be unless \( t_{7,S} \) is even. There is no assurance of this.
Another Example

The model is strongly consistent.

Solution to symbolic equations:

\[ q(\bar{p}) = \begin{bmatrix} 2 & 2 & 2p & 1 - p & 2 \end{bmatrix} \]

A trace \( S \) with \( N \) firings (\( N \) even) of actor 1 must have:

\[ q_S = \begin{bmatrix} N & N & t & (N - t) / 2 & N \end{bmatrix} \]

where \( t \) is the number of TRUEs consumed. There is no finite \( N \) where this is assured of being an integer vector.

Clustered Quasi-Static Schedules

Consider the clustered schedule:

\[
\begin{align*}
\text{n} &= 0; \\
\text{do} \{ \\
\text{fire 1;} \\
\text{fire 5;} \\
\text{fire 2;} \\
\text{if (b) \{ \\
\text{fire 3;} \\
\text{\} else \{ \\
\text{\text{n} += 1;} \\
\text{\}} \\
\text{\}} \text{while (n < 2);} \\
\text{fire 4;} \\
\text{\}
\end{align*}
\]

This schedule either fails to terminate or yields an integer vector of the form:

\[ q_S = \begin{bmatrix} N & N & t & (N - t) / 2 & N \end{bmatrix} \]
Delays Can Also Cause Trouble

This model is weakly consistent, where the balance equations have a non-trivial solution only if $p_7 = p_8$, in which case the solution is:

$$q(\bar{p}) = \begin{bmatrix} 1 & 1 & p_7 & 1 - p_7 & 1 & 1 \end{bmatrix}^T$$

Relating Symbolic Variables Across Delays

For the sample delay:

What is the relationship between $p_1$ and $p_2$?

Since consistency is about behavior in the limit, under the probabilistic of the interpretation for the symbolic variables, it is reasonable to assume $p_1 = p_2$.

Is this reasonable under the proportion interpretation?
Delays Cause Trouble with the Proportion Interpretation

Solution to the symbolic balance equations is

\[ q(\vec{p}) = \begin{bmatrix} 1 & 1 & p_7 & 1 - p_7 & 1 & 1 & 1 \end{bmatrix} \]

A trace \( S \) with \( N \) firings of actor 1 must have

\[ q_S = \begin{bmatrix} N & N & t_{7,S} & (N - t_{7,S}) & N & N & N \end{bmatrix} \]

But for no value of \( N \) is there any assurance of being able to fire actor 5 \( N \) times. This schedule won’t work.

Do-While Relies on a Delay

Imperative equivalent:

```java
while (true) {
    x = f1();
    b = false;
    while(!b) {
        (x, b) = f3(x);
    }
    f5(x);
}
```

Is this model strongly consistent? Weakly consistent? Inconsistent?
Checking Consistency of Do-While

This model is consistent if and only if \( p_5 = p_6 \), which is true under the probabilistic interpretation, but not under the proportion interpretation.

\[
\Gamma(\tilde{p}) = \begin{bmatrix}
1 & -p_5 & 0 & 0 & 0 \\
0 & 1 & -1 & 0 & 0 \\
0 & 0 & 1 & -1 & 0 \\
0 & 0 & 0 & p_6 & -1 \\
0 & -1 & 1 & 0 & 0 \\
0 & 0 & 1 & -1 & 0 \\
0 & -(1-p_3) & 0 & 1-p_6 & 0
\end{bmatrix}
\]

Let \( p = p_5 = p_6 \), then the solution to the balance equations is:

\[
q(\tilde{p}) = \begin{bmatrix}
1 & 1/p & 1/p & 1/p & 1
\end{bmatrix}^T
\]
This schedule yields traces $S$ for which $p_5 = p_6 = 1/N$ and

$$q_S = \begin{bmatrix} 1 & N & N & N & 1 \end{bmatrix}^T$$

compare:

$$q(\bar{p}) = \begin{bmatrix} 1 & 1/p & 1/p & 1/p & 1 \end{bmatrix}^T$$

Extensions

- State enumeration scheduling approach: Seek a finite set of finite guarded schedules that leave the model in a finite set of states (buffer states), and for which there is a schedule starting from each state.

- Integer dataflow (IDF [Buck ’94]): Allow symbolic variables to have integer values, not just Boolean values. Extension is straightforward in concept, but reasoning about consistency becomes harder.
Taking Stock

- BDF and IDF generalize the idea of balance equations and introduce *quasi-static scheduling*.
- BDF and IDF are Turing complete, so existence of quasi-static schedules is undecidable.
- Can often construct quasi-static schedules anyway.
- Tricks like clustered schedules make the set of manageable models larger.
- Are Switch and Select like unrestricted GOTO?

Extensions of SDF that Improve Expressiveness

- **Structured Dataflow** [Kodosky 86, Thies et al. 02]
  - Boolean dataflow [Buck and Lee, 93]
  - Cyclostatic Dataflow [Lauwereins 94]
  - Multidimensional SDF [Lee & Murthy 96]
  - Heterochronous Dataflow [Girault, Lee, and Lee, 97]
  - Parameterized Dataflow [Bhattacharya et al. 00]
  - Teleport Messages [Thies et al. 05]

Many of these remain decidable
LabVIEW uses homogeneous SDF augmented with syntactically constrained forms of feedback and rate changes:
- While loops
- Conditionals
- Sequences
LabVIEW models are decidable.

vs. Dynamic Dataflow, which uses token routing for control flow

Imperative equivalent:

```plaintext
while (true) {
    x = f1();
    b = f7();
    if (b) {
        y = f3(x);
    } else {
        y = f4(x);
    }
    f6(y);
}
```

The if-then-else model is not SDF. But we can clearly give a bounded quasi-static schedule for it:

(1, 7, 2, b?3, !b?4, 5, 6)
vs. Dynamic Dataflow, which uses token routing for control flow

This model uses conditional routing of tokens to iterate a function a data-dependent number of times.

Imperative equivalent:

```java
while (true) {
    x = f1();
    b = false;
    while (!b) {
        (x, b) = f3(x);
    }
    f5(x);
}
```

Syntax: Graphical or Textual?

The graphical vs. textual debate obscures a more important question:

*Are actors and streams a programming language technology or a software component technology?*
SR models are intrinsically bounded, but have related consistency issue: Clock consistency.

Two Interpretations of Clocks

A clock is a property of a model, and signals may be absent at ticks of the clock (Esterel)

A clock is a property of a signal, and components impose constraints on clock relationships (Lustre, Signal)

The choice between these has profound consequences
The Ptolemy II SR Director realizes Esterel-style clocks with hierarchical clock domains.

In this example, the CountDown composite issues a "ready" signal to the EnabledComposite, which then issues a number. The CountDown composite counts down from that number to 0, then issues another ready.

The EnabledComposite has a clock that ticks only when the enable input is present and true. It issues the sequence 1, 5, 3, 2, followed by absent henceforth.

The three displays show (left to right):
- Requested numbers to count down from.
- The count down for these numbers.
- The enable signal for the EnabledComposite actor.
The Clock is a Property of the Model

Hierarchical clock domains bear some resemblance to structured dataflow

Opaque hierarchy can do:

- Conditioning an internal tick on an external signal
  - Like a conditional
  - If the internal component is an instance of the external, then this amounts to recursion
- Multiple internal ticks per external tick
  - Like a do-while
- Iterated internal ticks over a data structure (use IterateOverArray higher-order actor)
  - Like a for
A Consequence: Pre and NonStrictDelay have different behaviors!

Alternative Semantics:
The Clock is a Property of the Signal

In Lustre and Signal, a clock is a property of a signal, and Pre and NonStrictDelay behave identically by constraining the input clock to be the same as the output clock. They only fire when the clock of the input signal ticks.

This leads to a clock consistency problem, which is in general undecidable.

Inconsistent clocks

In Lustre-style clock systems, the AddSubtract actor imposes the constraint that all its input signals have the same clock. The above model becomes inconsistent and will not execute.
Clock Calculus

- Let $T$ be a totally ordered set of tags.
- Let $s: T \rightarrow V \cup \{ \epsilon \}$ be a signal of type $V$, where $\epsilon$ means “absent.”
- Let $c: T \rightarrow \{-1, 0, 1\}$ be a clock associated with $s$ where
  
  $s(t) = \epsilon \Rightarrow c(t) = 0$
  $s(t) = true \Rightarrow c(t) = 1$
  $s(t) = false \Rightarrow c(t) = -1$

  If $V$ is not boolean, then when $s(t)$ is present, $c(t)$ has value or 1 or $-1$ (we will make no distinction).

Operations on Clocks

Arithmetic on clocks is in GF-3 (a Galois field with 3 elements), as follows:

$$
\begin{align*}
0 + x &= x \\
1 + 1 &= -1 \\
-1 + -1 &= 1 \\
-1 + 1 &= 0
\end{align*}
$$
Clock Relations: Simple Synchrony

Most actors require that the clocks on all signals be the same. For example:

\[ \forall t \in T, \quad c_1^2(t) = c_2^2(t) = c_3^2(t) \]

This means that either all are present, or all are absent.

Clock Relations: When Operator

Assuming that \( s_1 \) is a boolean-valued signal (which it must be), the clocks on signals interacting through the when operator are related as follows:

\[ \forall t \in T, \quad c_3(t) = c_1(t)(-c_2(t) - c_2^2(t)) \]

This means:
- If \( s_1 \) is absent, then \( s_3 \) is absent.
- If \( s_2 \) is false, then \( s_3 \) is absent.
- If \( s_2 \) is true, then \( s_3 \) is the same as \( s_1 \).
Consistency Checking

Consider the following model:

\[ \forall t \in T, \quad c_1^2(t) = c_4^2(t) = c_3^2(t) \]

\[ \forall t \in T, \quad c_3(t) = c_1(t)(-c_2(t) - c_2^2(t)) \]

These two together imply that:

\[ \forall t \in T, \quad c_1^2(t)(1 + c_2^2(t)) = -c_2(t)c_1^2(t) \]

where we have used the fact that:

\[ (-c_2(t) - c_2^2(t))^2 = (-c_2(t) - c_2^2(t)) \]

Interpretation of Consistency Result

Consistency check implies that:

\[ \forall t \in T, \quad c_1^2(t)(1 + c_2^2(t)) = -c_2(t)c_1^2(t) \]

This means:

\[ s_1 \text{ is absent if and only if } s_2 \text{ is absent or false.} \]
Logic Operators Affect Clocks

The output of the When actor has a clock that depends on the Boolean control signal. Clocks of Boolean-valued signals reflect the signal value as follows:

\[
\forall t \in T, \quad c_2(t) = -c_1(t)
\]

\[
\begin{align*}
&c_3(t) = (c_1(t)c_2(t))^2 \left((-c_1(t) + 1)(c_2(t) + 1) - 1\right) \\
&c_3(t) = (c_1(t)c_2(t))^2 \left((c_1(t) - 1)(c_2(t) - 1) + 1\right)
\end{align*}
\]

Token Routing Also Affects Clocks

Switch and Select affect the clocks as follows:

\[
\forall t \in T, \quad c_4(t) = c_3(t)(c_2(t)(1 - c_3(t)) - c_1(t)(1 + c_3(t)))
\]

\[
\begin{align*}
&- (c_3(t) + 1)c_3(t) = c_1^2(t) \\
&- (c_3(t) - 1)c_3(t) = c_2^2(t)
\end{align*}
\]

\[
\begin{align*}
&c_3(t) = -c_2(t)(c_2(t) + 1)c_1(t) \\
c_4(t) = c_2(t)(1 - c_2(t))c_1(t) \\
c_2^2(t) = c_1^2(t)
\end{align*}
\]
Example 1 Using Switch and Select

What can you infer about the clock of $s_6$?

$$c_6(t) = 0$$

Example 2 Using Switch and Select

What can you infer about the clocks?

$$c_1(t) = 0 \quad \text{and} \quad c_3(t) = 0 \quad \text{or} \quad 1 + c_3(t) = 0$$

This means that $s_1$ is absent and $s_3$ is either absent or false.
What About Delays?

Clock relations across the delays become dependent on the tags. E.g., if $T$ is the natural numbers, then we get a nonlinear dynamical system:

$$c_1^2(t) = c_2^2(t) \quad \text{and}$$
$$c(0) = \text{initial state}$$
$$c(t + 1) = (1 - c_1^2(t))c(t) + c_1(t)$$
$$c_2(t) = c_1^2(t)c(t)$$

This makes clock analysis very difficult, in general.

Default Operator

Default: The output equals the left input, if it is present, and the bottom input otherwise:

$$\forall t \in T, \quad c_3(t) = c_1(t) + c_2(t)(1 - c_1^2(t))$$

This means the clock of $s_3$ is equal to the clock of $s_1$, if it is present, and to the clock of $s_2$ otherwise.
Default Operator in SIGNAL is Nondeterministic

In SIGNAL semantics, the following model has many behaviors:

The two generated sequences have independent clocks (defined over incomparable values of \( t \in T \)), and the output sequence is any interleaving that preserves the ordering.

Guarded Count in SIGNAL

Instead of generating a “ready” signal, in SIGNAL, the count hitting zero can be synchronized with the input being present.
Conclusion and Open Issues

- When clocks are a property of the model, the result is structured synchronous models, where differences between clocks are explicit and no consistency checks are necessary (and signals may be *absent* at ticks of the clock).

- When clocks are a property of a signal, the result is similar to Boolean Dataflow (BDF). It is arguable that clock operators like “when,” “default,” “switch,” and “select” become analogous to unstructured gotos. Clock consistency checking becomes undecidable.

- When further extended as in SIGNAL to partially ordered clock ticks, models easily become nondeterministic.