Simple Example

\[ F = (A+B) \times (C-D) \]

Nodes - transforms

Edges - flow of tokens (data) from one node to another

- When all edges have at least one token, node may fire
- When node fires, it consumes input tokens, processes transformation, generates output token

Commercial tools (graphical representation)

- Automatically translates to concurrent process model

Pros:
- Easily Parallelizable
- Easy representation using block diagrams

Cons:
- Memory problem (assumes tokens between processes)
A new impose constraints (memory for communication cannot be unbounded)

\[3A - 2B = 0\]
\[9B - 3C = 0\]
\[2C - A = 0\]
\[4B - 3D = 0\]
\[D - 2A = 0\]

Solution

\[a = 2c\]
\[b = 3c\]
\[d = 4c\]

- Determines rules

- Schedule

- If rules determined, any scheduling algorithm that avoids buffer underflow will produce correct schedule

- Many scheduling options
  - reduced code size
  - minimize memory
Example Schedule

```
BBBC CDDDDA

\crick code
```

Looped Code Generation

Single Appearance Schedules

- By looped schedule in which each block appears exactly once.
- Efficient code implementation.
- Often requires more buffers than necessary (other schedules).

```
(3 \textit{vs}) c (4D) (2\textit{A})

\crick code
```

```
\text{for} (i = 0, i < 3, i++) b_i

\text{c}_i = \text{D}_i

\text{\textbullet} \quad \text{A}
```