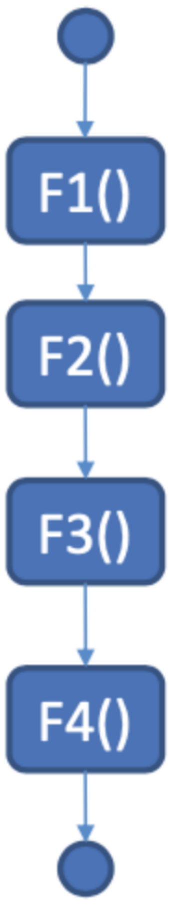
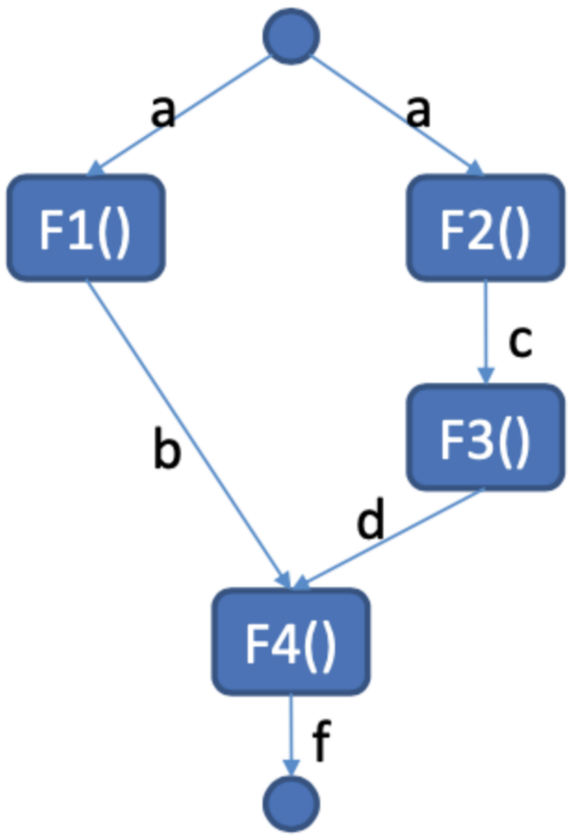
6. Futures + Pipelines

Definition

Concurrency pattern.
Relates to parallelism.
Both are used when different parts of a program depends on a result from a previous part.

Futures





```
var a;
Task<T> futureb = Task.Run(() => F1(a));
```

```
var c = F2(a);
var d = F3 (c);
F4(future.Result, d);
```

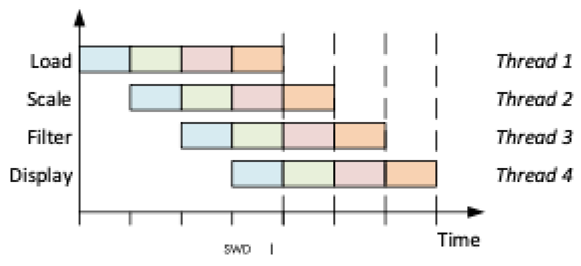
Pipelines

Created using tasks and concurrent queues (`BlockingCollection<T>`)

```
void DoStage (BlockingCollection<T> input,
              BlockingCollection<T> output)
{
    try
    {
        foreach(var item in input.GetConsumingEnumerable())
        {
            var result = ...;
            output.Add(result);
        }
    }
    finally
    {
        output.CompleteAdding();
    }
}
```

`GetConsumingEnumerable` provides an iterator to get values from `BlockingCollection<T>`.
`CompleteAdding()` avoids race conditions, as it signals that the processing has finished.

Pipelined processing



SOLID

S each stage in a pipeline has a responsibility for handling input in a specific way to produce an output
O It should be possible to just add another stage on a pipeline since there are certain pre and post conditions to be met in a pipeline.

Comparison

Parallel aggregation

MapReduce