Programming
Weak Synchronization
Models

Christopher S. Meiklejohn
Université catholique de Louvain, Belgium
Instituto Superior Técnico, Portugal

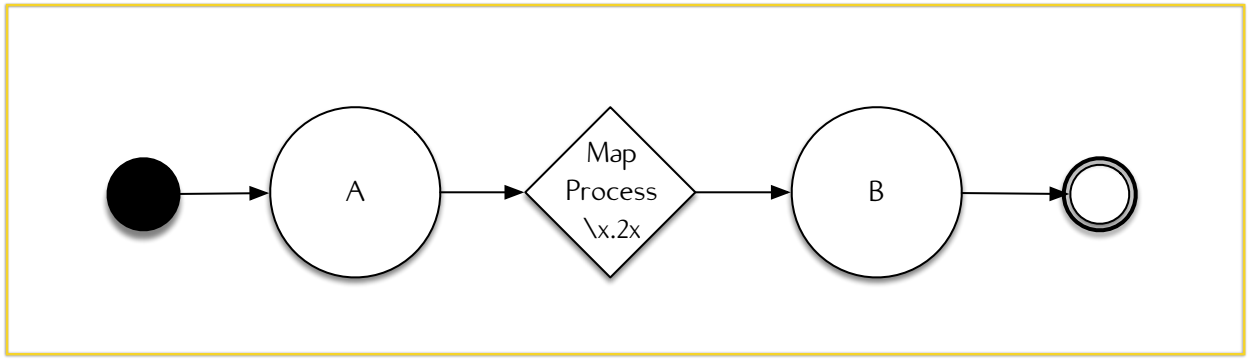



Convergent Programs
Lattice Processing

Lattice Processing

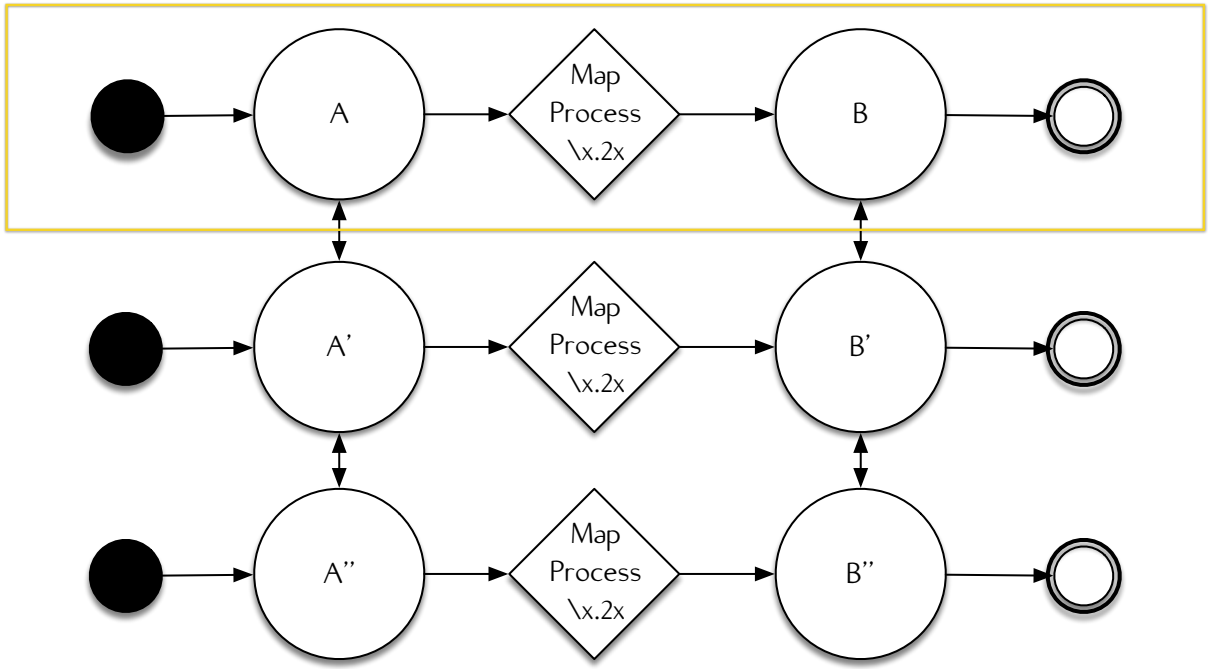
- Asynchronous dataflow with streams
  Combine and transform streams of inputs into streams of outputs
- Convergent data structures
  Data abstraction (inputs/outputs) is the CRDT
- Confluence
  Provides composition that preserves the SEC property

Lattice Processing
Confluence

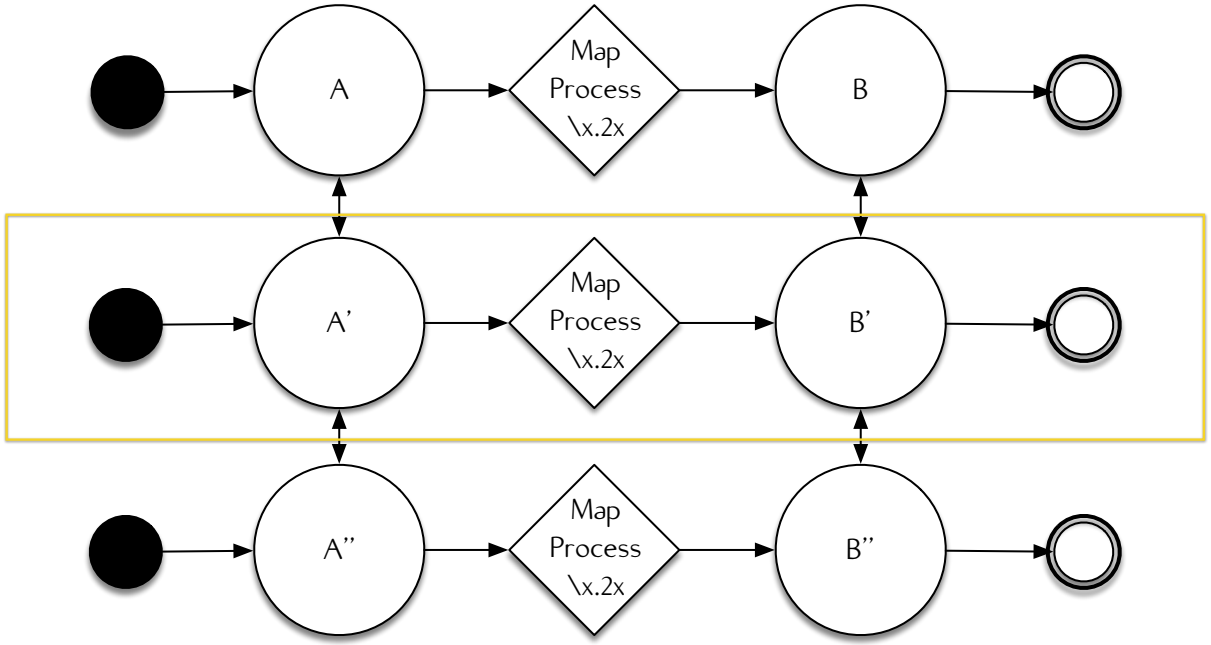


Sequential
specification.

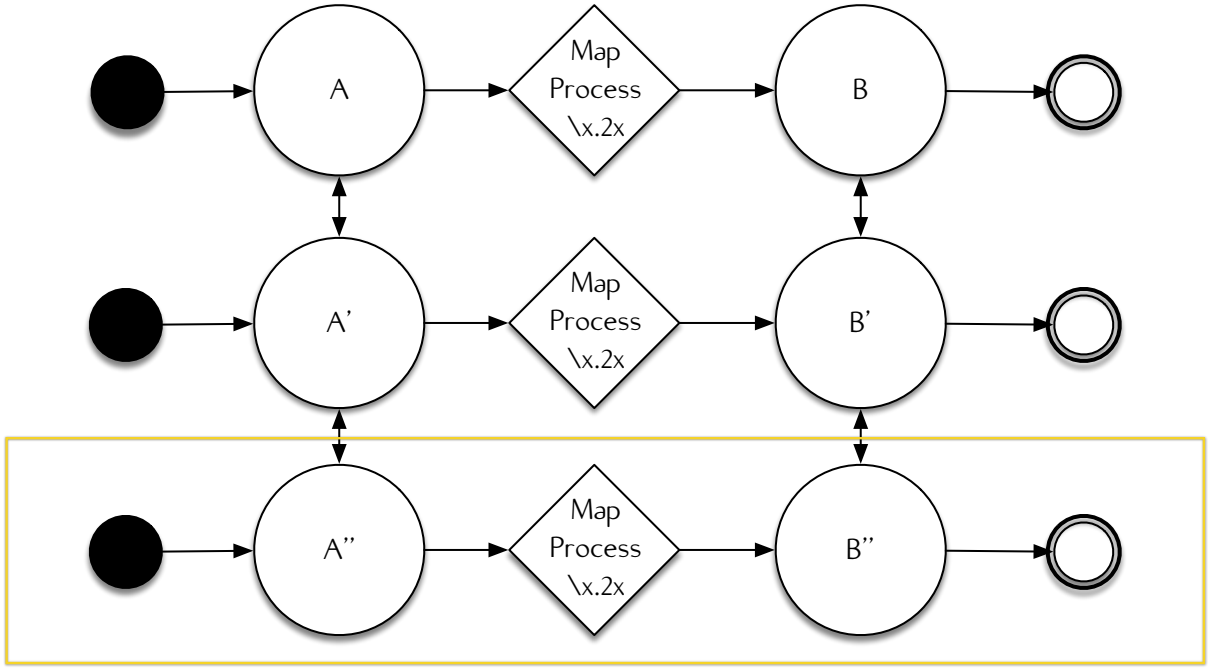
Replication
per node.



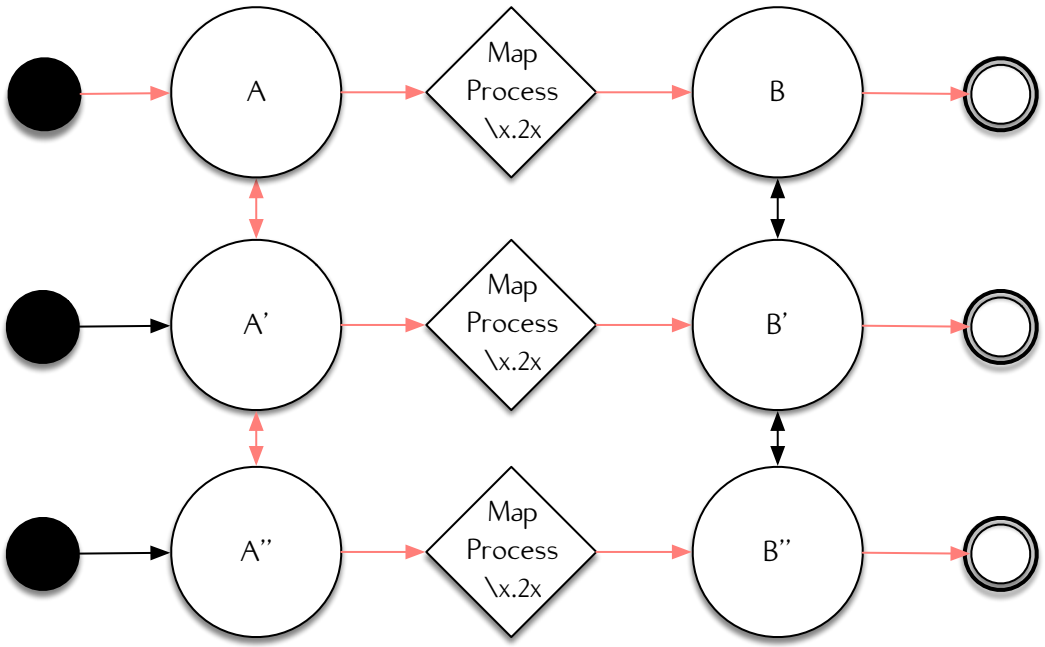
Replication
per node.

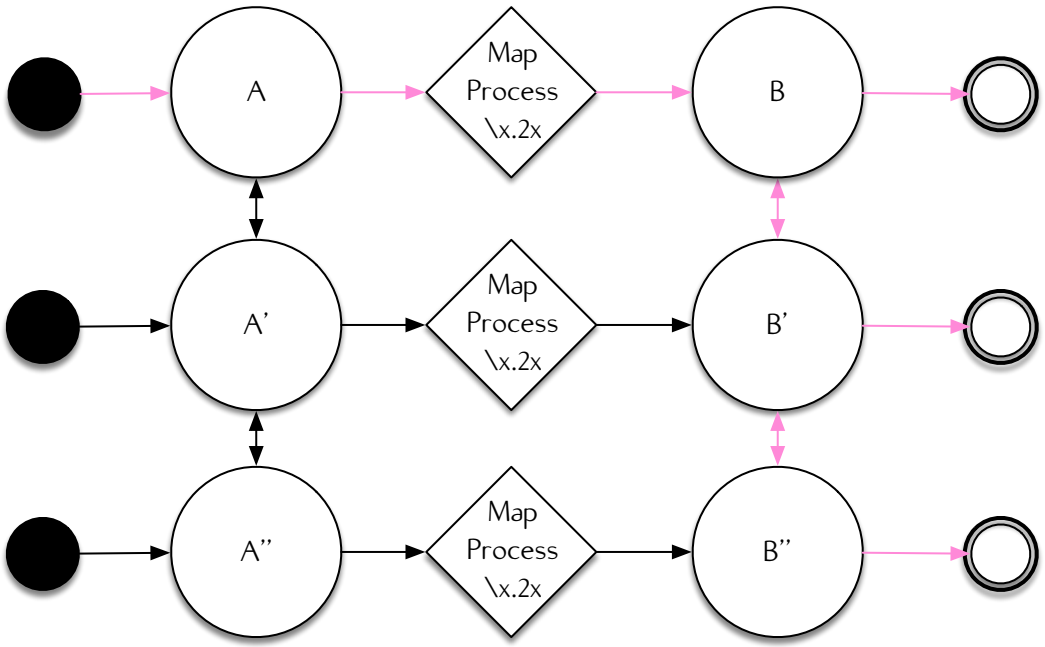


Replication
per node.

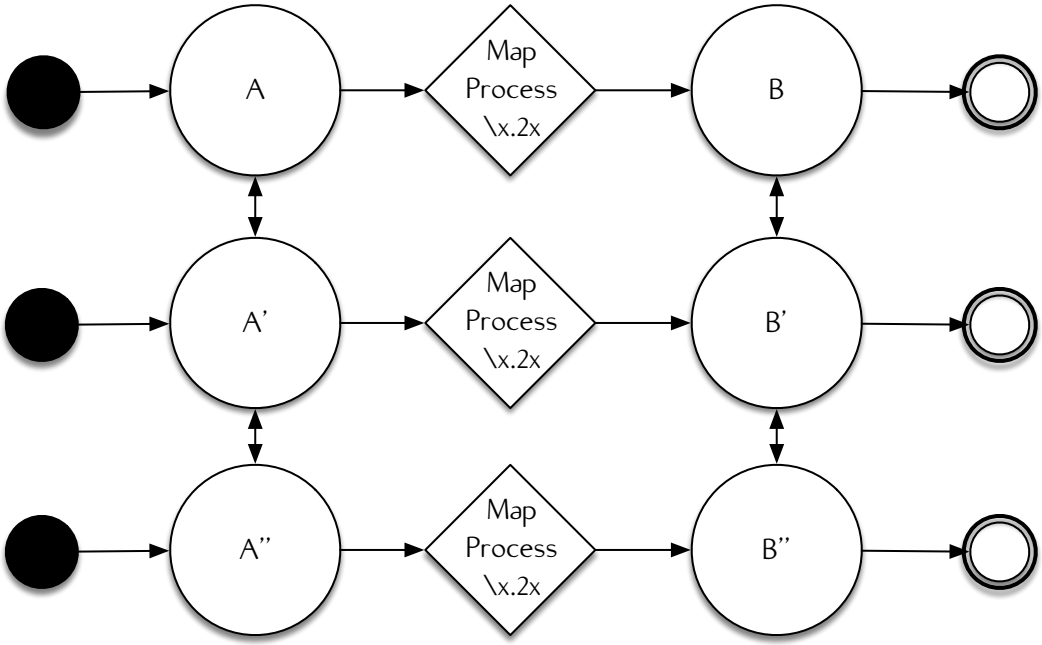


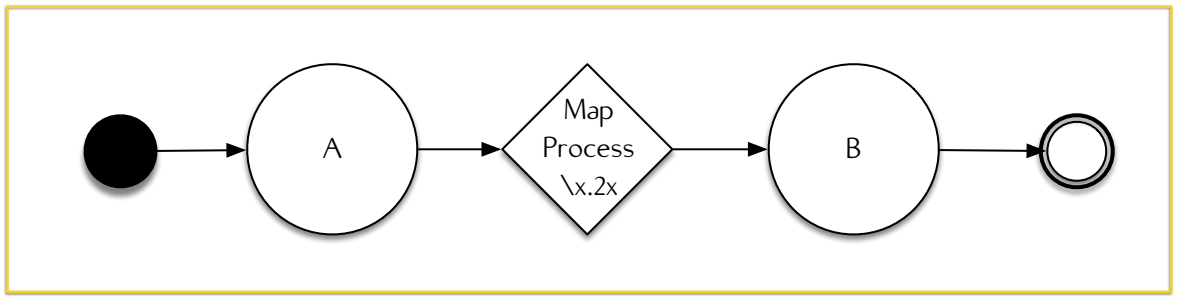
One **possible** schedule….



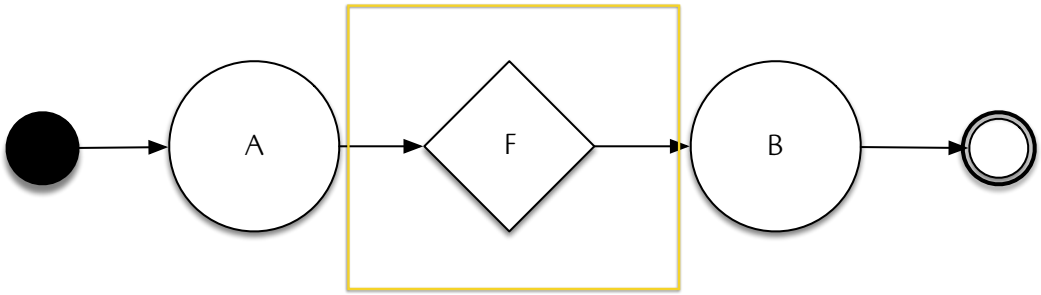


…another **possible** schedule.


A
Map
Process
\x.2x
B
A'
Map
Process
\x.2x
B'
A''
Map
Process
\x.2x
B''



All schedules **equivalent**
to sequential schedule.



Arbitrary
application.

Lattice Processing
Example

```
%% Create initial set.
S1 = declare(set),

%% Add elements to initial set and update.
update(S1, {add, [1,2,3]}),

%% Create second set.
S2 = declare(set),

%% Apply map operation between S1 and S2.
map(S1, fun(X) -> X * 2 end, S2).
```

```
%% Create initial set.
S1 = declare(set),

%% Add elements to initial set and update.
update(S1, {add, [1,2,3]}),

%% Create second set.
S2 = declare(set),

%% Apply map operation between S1 and S2.
map(S1, fun(X) -> X * 2 end, S2).
```

```
%% Create initial set.
S1 = declare(set),

%% Add elements to initial set and update.
update(S1, {add, [1,2,3]}),

%% Create second set.
S2 = declare(set),

%% Apply map operation between S1 and S2.
map(S1, fun(X) -> X * 2 end, S2).
```

```
%% Create initial set.
S1 = declare(set),

%% Add elements to initial set and update.
update(S1, {add, [1,2,3]}),

%% Create second set.
S2 = declare(set),

%% Apply map operation between S1 and S2.
map(S1, fun(X) -> X * 2 end, S2).
```

```
%% Create initial set.
S1 = declare(set),

%% Add elements to initial set and update.
update(S1, {add, [1,2,3]}),

%% Create second set.
S2 = declare(set),

%% Apply map operation between S1 and S2.
map(S1, fun(X) -> X * 2 end, S2).
```