Network Fusion

Pascal Fradet[1] and **Stephane Hong Tuan Ha**[2]

[1] INRIA Rhone-Alpes

Pascal.Fradet@inria.fr

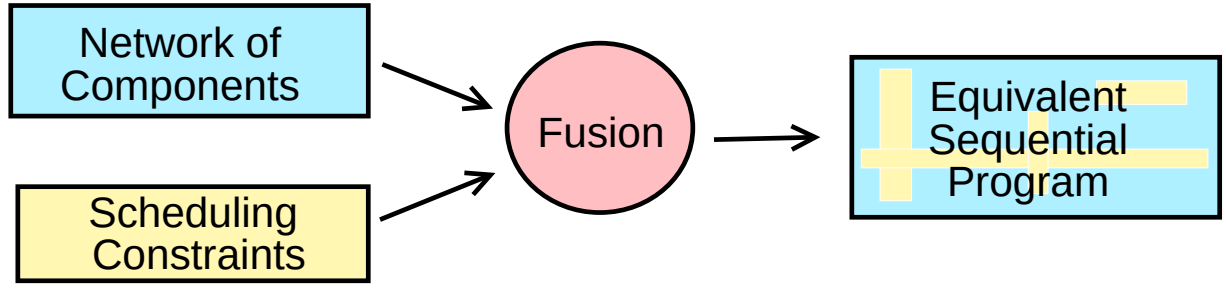
[2] IRISA/INRIA Rennes

shongtua@irisa.fr

Context and Motivation

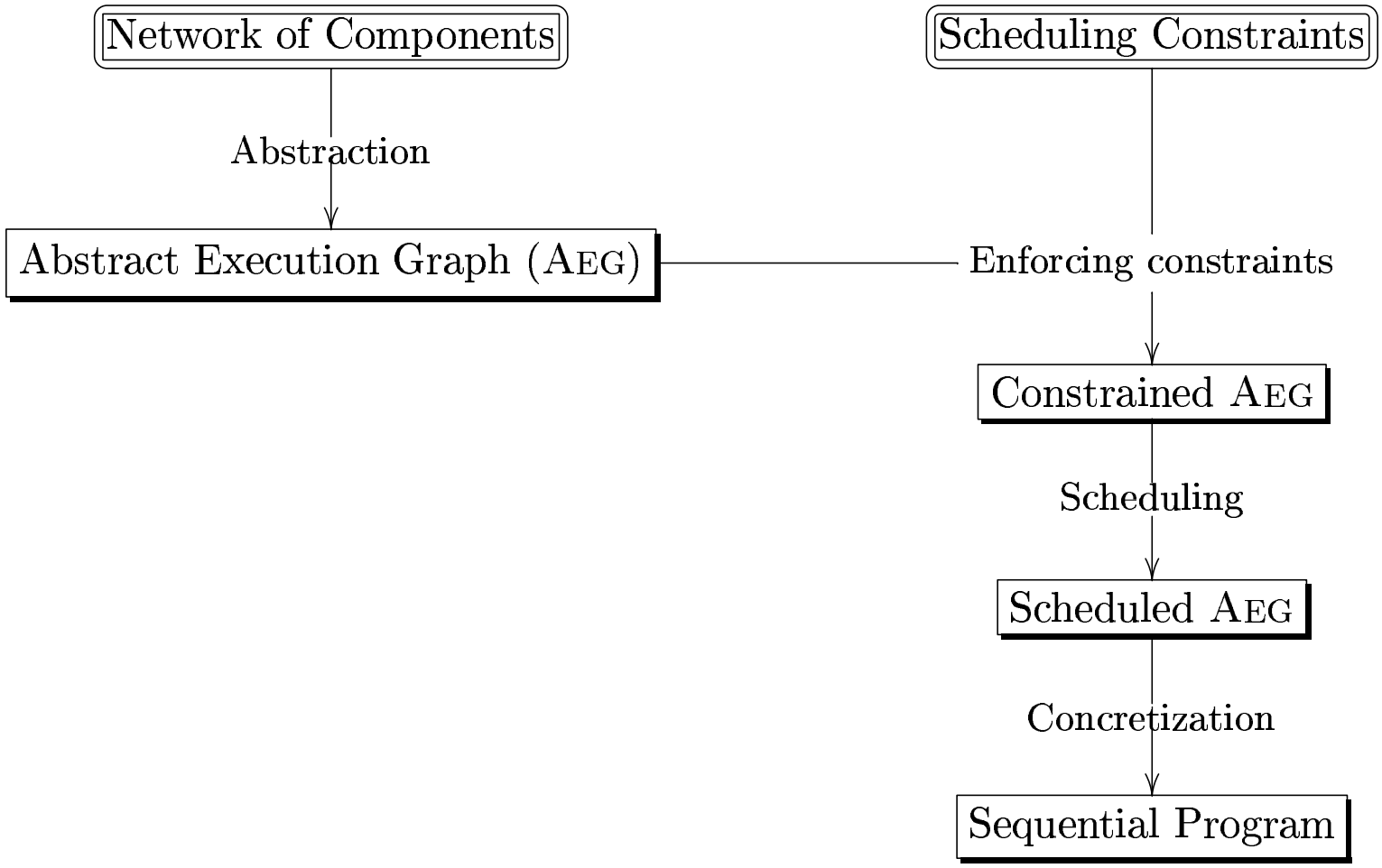
- Component programming
  - Advantages : readability, maintanibility, separate development
  - Disadvantages : efficiency issues
- Filter Fusion
  - Proposed by Proebsting and Watterson at POPL'96
  - Program transformation to fusion filters in pipeline with a particular scheduling strategy
- Our work: extend Filter Fusion to
  - General networks
  - User-defined scheduling strategies

Network Fusion



- Efficient implementation of networks of components
  - Compiles and removes context switches and communication channels
- Generalizes and formalizes filter fusion
  - Based on LTS, synchronized product, and static analyses
- Inspired from AOP
  - Keeps functionnality and scheduling issues separate
  - Can be seen as weaving synchronisation aspects

Overview


Network of Components
Abstraction
Abstract Execution Graph (AEG)
Scheduling Constraints
Enforcing constraints
Constrained AEG
Scheduling
Scheduled AEG
Concretization
Sequential Program

Component Model

- Based on Kahn Process Network (KPN)
  - Simple and formal model
- Main features:
  - Set of asynchronous communicating processes
  - Unbounded FIFO-based communication channels
  - Non-blocking write, blocking read (on an empty channel)
  - At most one producer and one consummer
- Main properties:
  - Deterministic
  - Hierarchical

Component Language

- Definition of components by commands of the form :

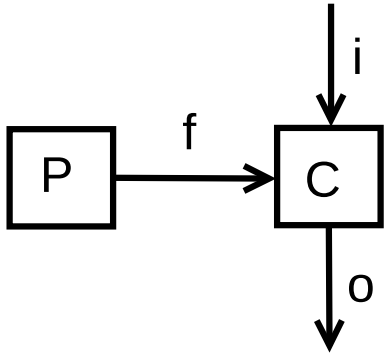
$$\text{Commands} ::= l_1 : G \mid A \rightsquigarrow l_2$$

where l_1, l_2 denote labels, G a guard, and A an action.

- Actions $::=$ $f?x$ read x on channel f
  $\mid$ $f!x$ write x on channel f
  $\mid$ I internal action (e.g. *x:=x+1*)

Example - Network of Components

- Network :



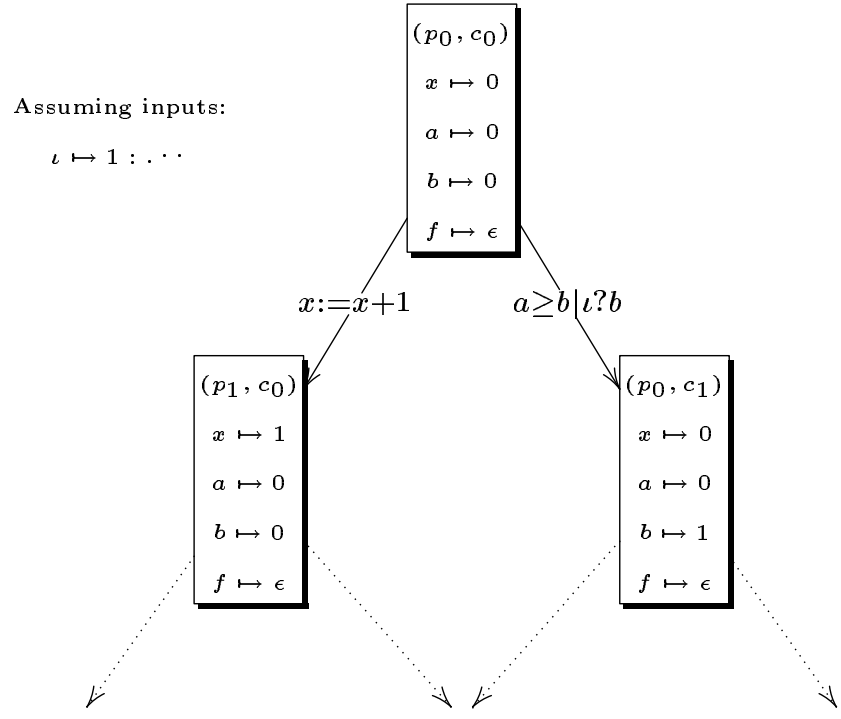
- Components :

$$P = \left\{ \begin{array}{lcll} p_0 & : & x := x + 1 & \rightsquigarrow \; p_1; \\ p_1 & : & f!x & \rightsquigarrow \; p_0 \end{array} \right\}$$

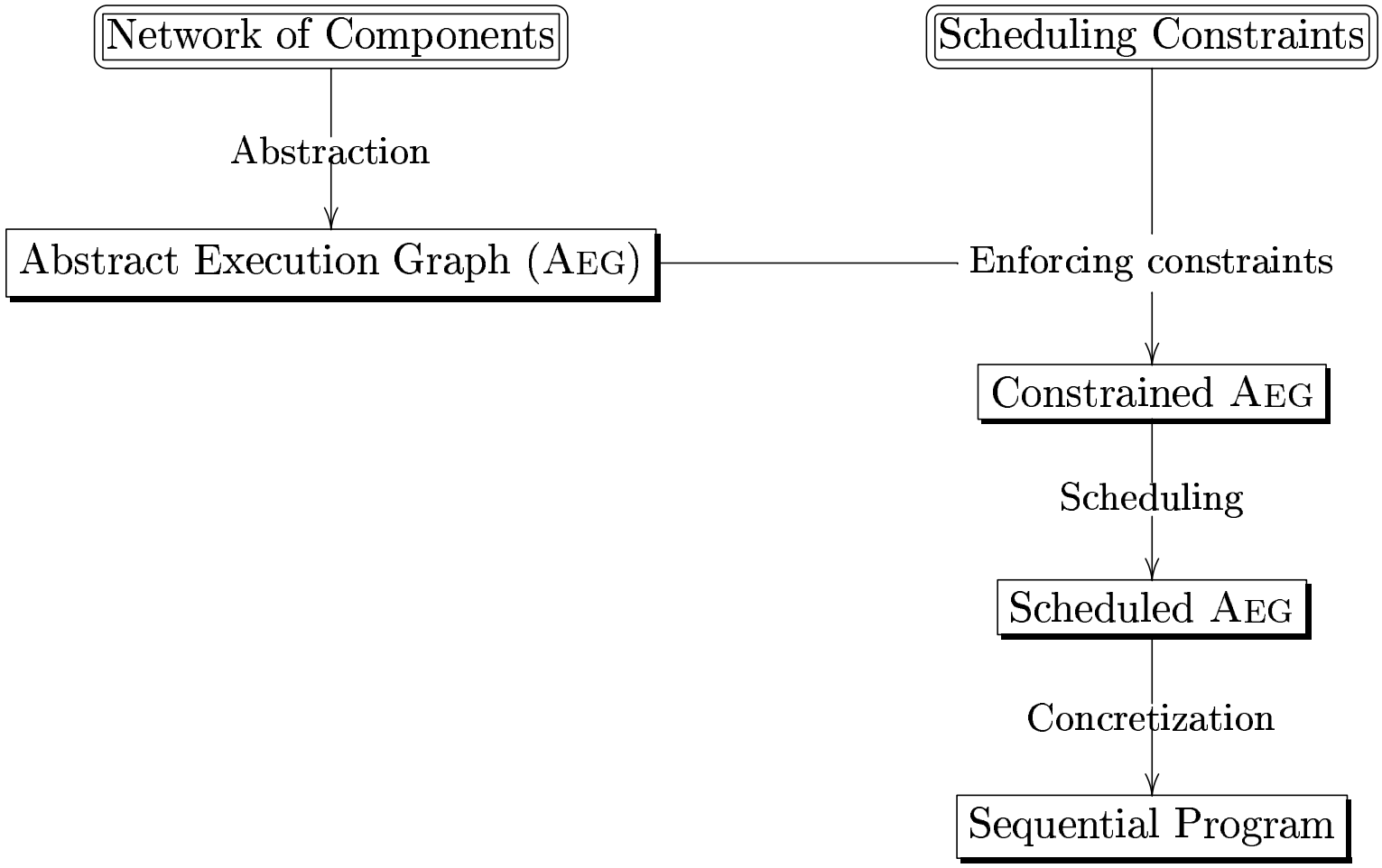
$$C = \left\{ \begin{array}{lcll} c_0 & : & a < b \mid f?a & \rightsquigarrow \; c_1; \\ c_0 & : & a \geq b \mid \iota?b & \rightsquigarrow \; c_1; \\ c_1 & : & o!(a + b) & \rightsquigarrow \; c_0 \end{array} \right\}$$

Semantics of networks

- Operational semantics given by a LTS where
  - States represent program points of process, associations (variable $\rightarrow$ value) and (channel $\rightarrow$ file)
  - Transitions represent one execution step
- Example



Overview

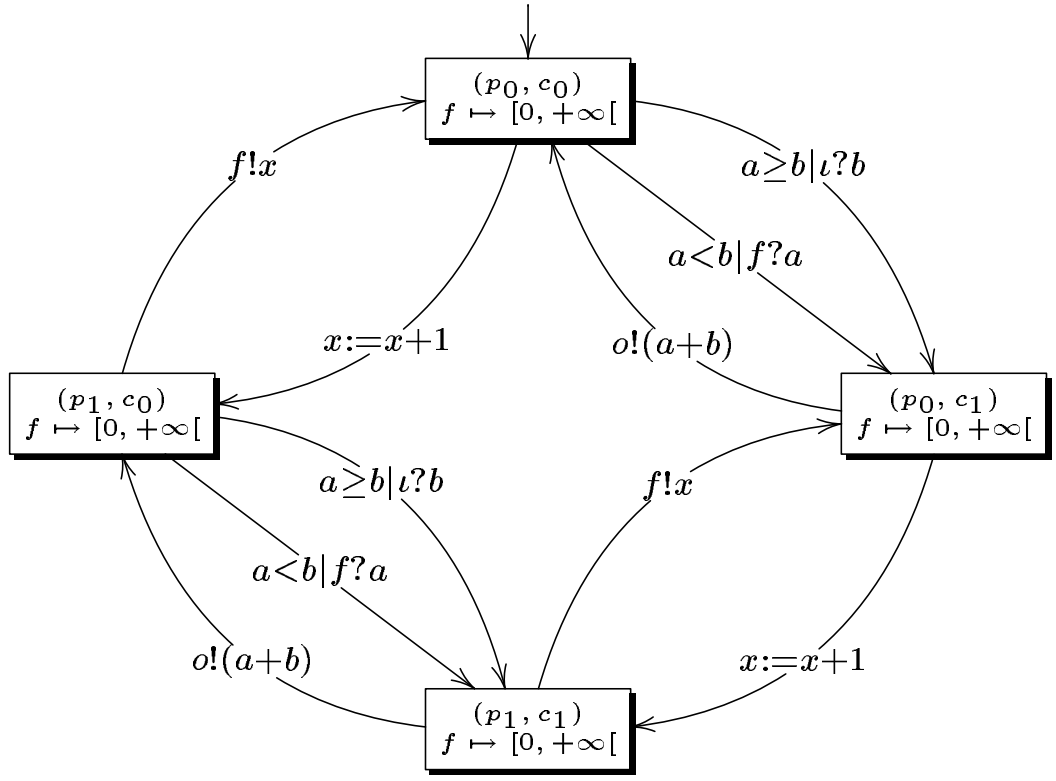

Network of Components
Abstraction
Abstract Execution Graph (AEG)
Scheduling Constraints
Enforcing constraints
Constrained AEG
Scheduling
Scheduled AEG
Concretization
Sequential Program

Abstract Execution Graph (AEG)

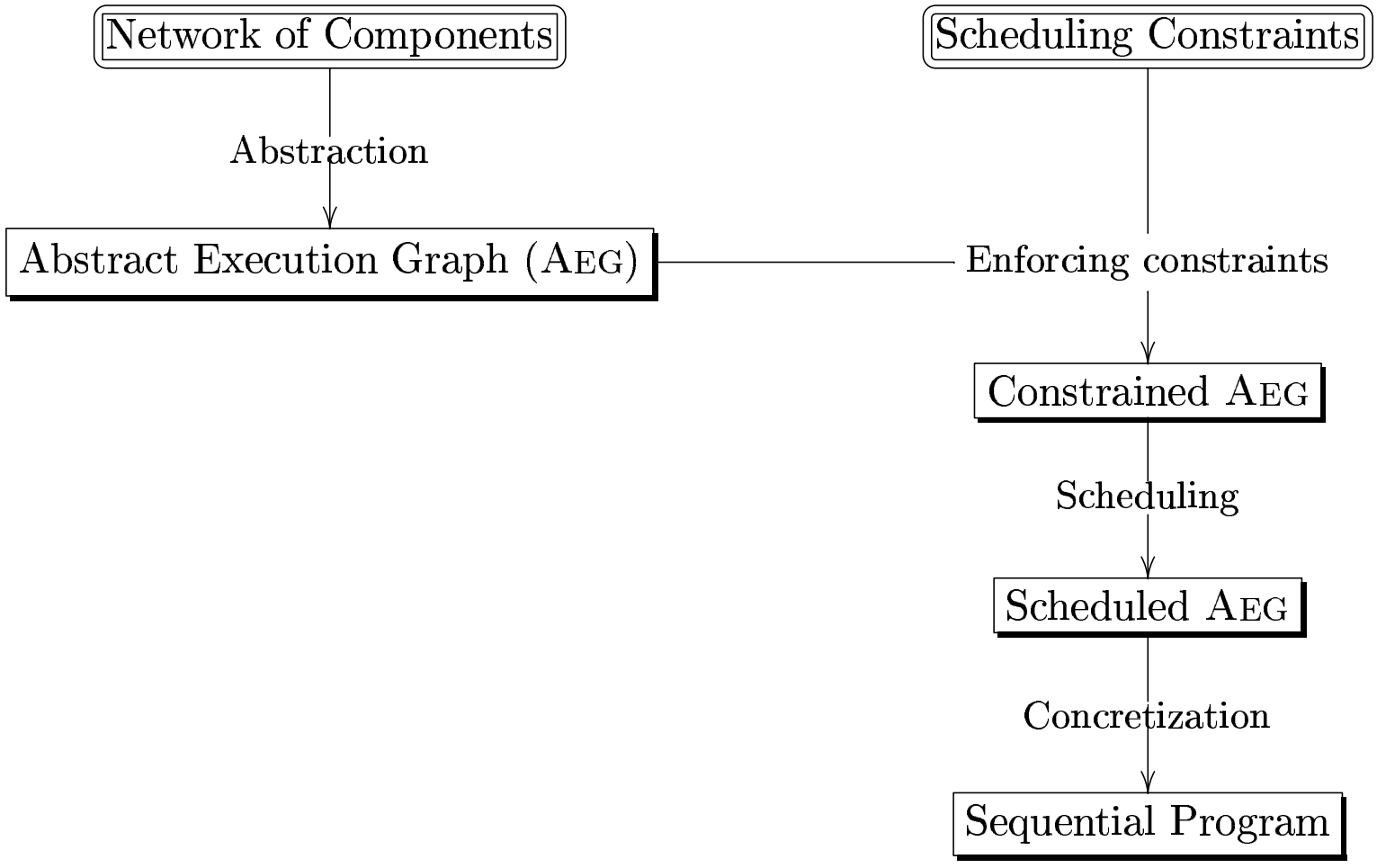
- Finite model upper-approximating all the possible execution of the KPN
  - Based on abstract interpretation techniques
- Must satisfy a few properties (safety, ...)
- Several abstractions possible (size against accuracy, more precise $\Rightarrow$ larger AEG)

Example - AEG

- Abstraction of states into
  - Currents program point of each component
  - Intervals approximating the size of each channel



Overview

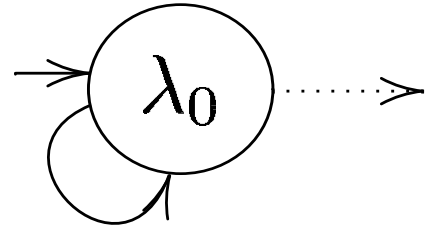

Network of Components
Abstraction
Abstract Execution Graph (AEG)
Scheduling Constraints
Enforcing constraints
Constrained AEG
Scheduling
Scheduled AEG
Concretization
Sequential Program

Scheduling constraints

- Specify the set of allowed executions
- Expressed with respect to
  - IO operations
  - Sizes of files
  - ...
- From no constraint to a complete sequentialization of the execution
- Benefits :
  - impose implementation choices (e.g. bounding a fifo channel)
  - guide and optimize the fusion process

Specifying constraints

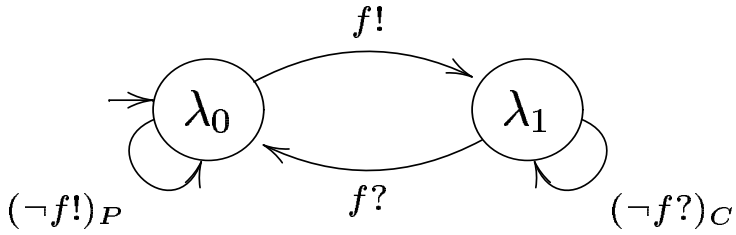
- Expressed by a LTS labeled with guarded actions
- Example :



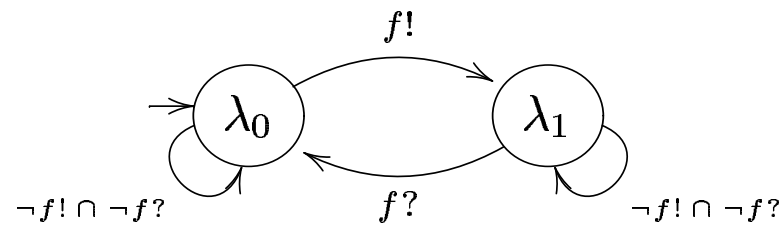
$$\overline{f} < k - 1 \mid f!^{(\neg f!)_P}$$

  - $f!$: any write on channel f
  - $(\neg A)$: all operations except A
  - $(A)_p$: projection of A onto commands of process p
  - $\overline{f} < k|A$: all actions A provided that the size of channel f is less than k

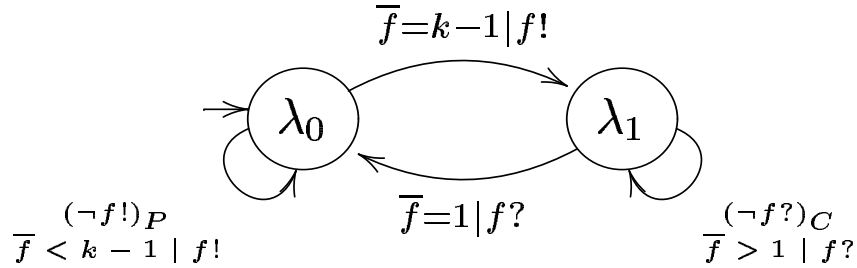
Example - Scheduling Constraints



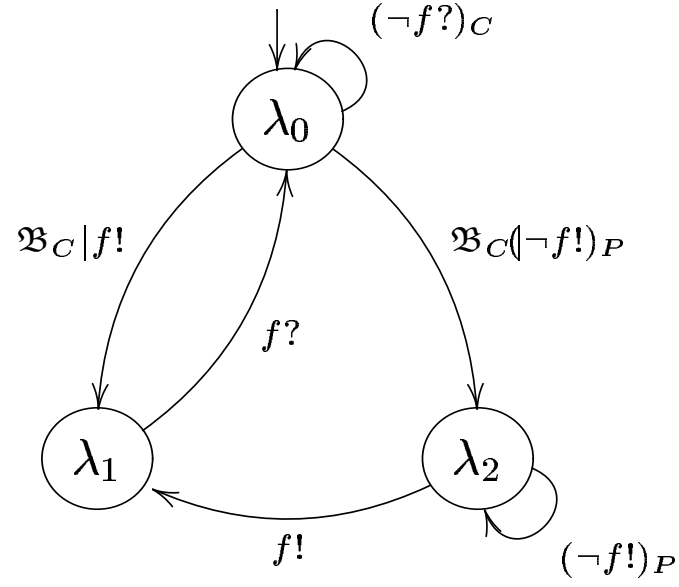
The Filter Fusion Strategy (Λ_{FF})



A Simple Incomplete Strategy (Λ_{IS})

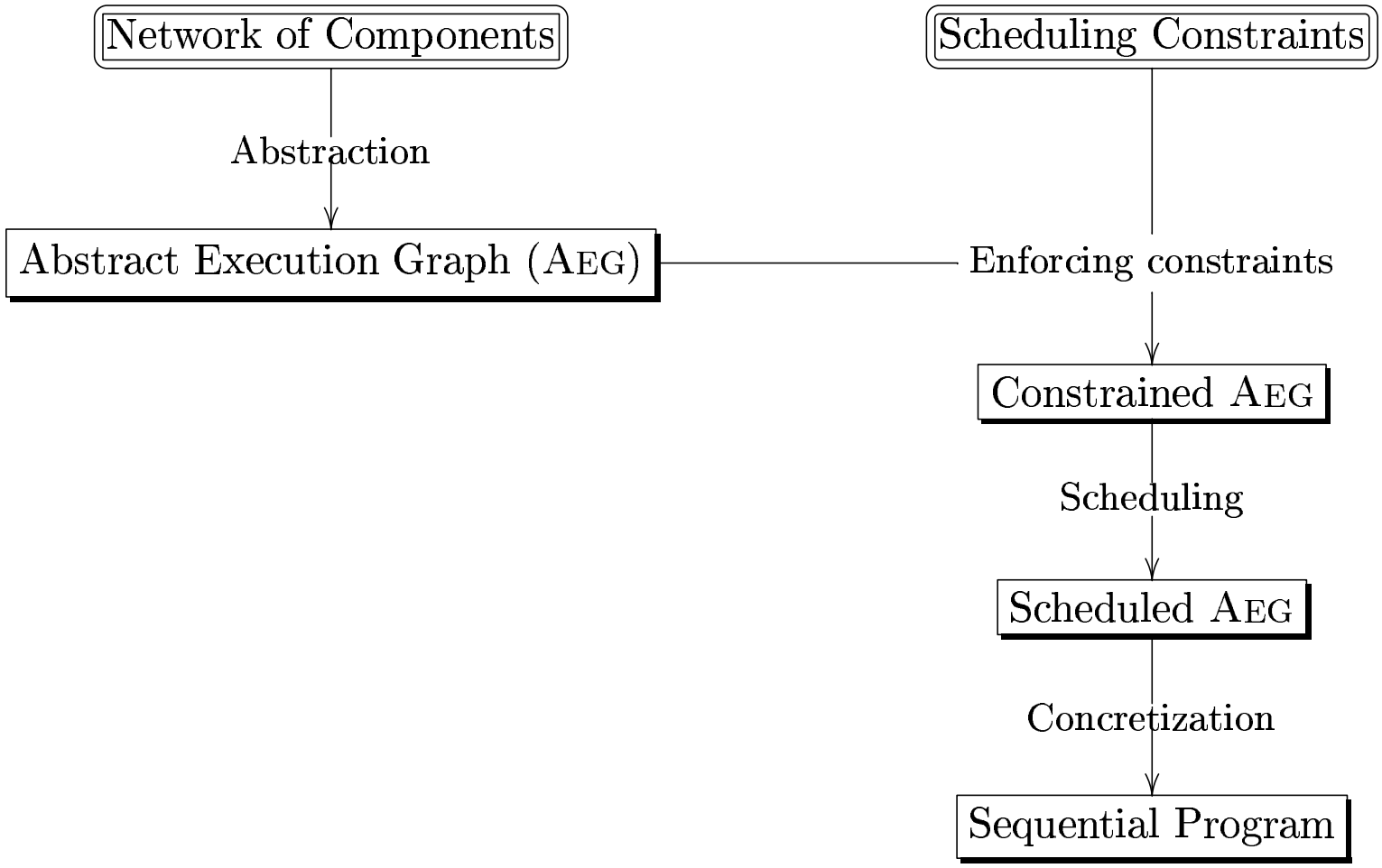


A Filter Fusion Strategy with Bounds (Λ_{FB})



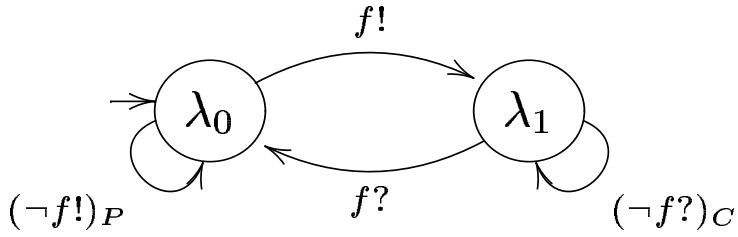
A Demand Driven Strategy (Λ_{DD})

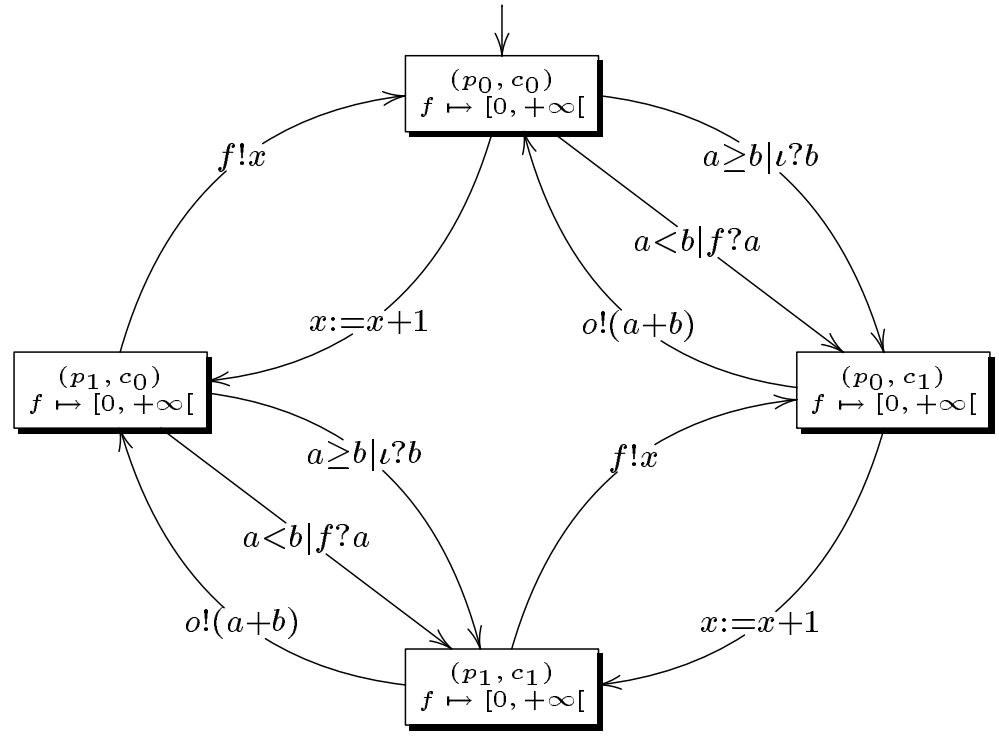
Overview


Network of Components
Abstraction
Abstract Execution Graph (AEG)
Scheduling Constraints
Enforcing constraints
Constrained AEG
Scheduling
Scheduled AEG
Concretization
Sequential Program

Enforcing constraint

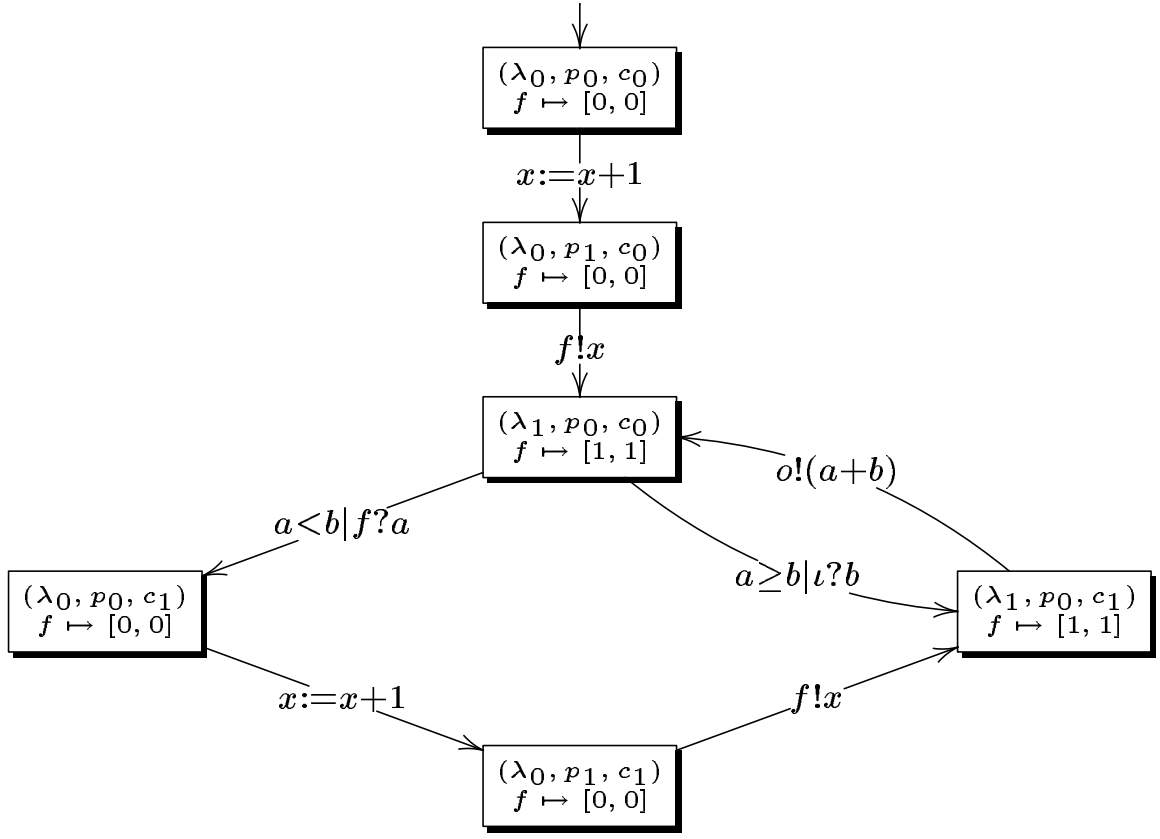
- "Synchronized product" between AEG and constraints
- Example: Aeg $\| \Lambda_{FF}$



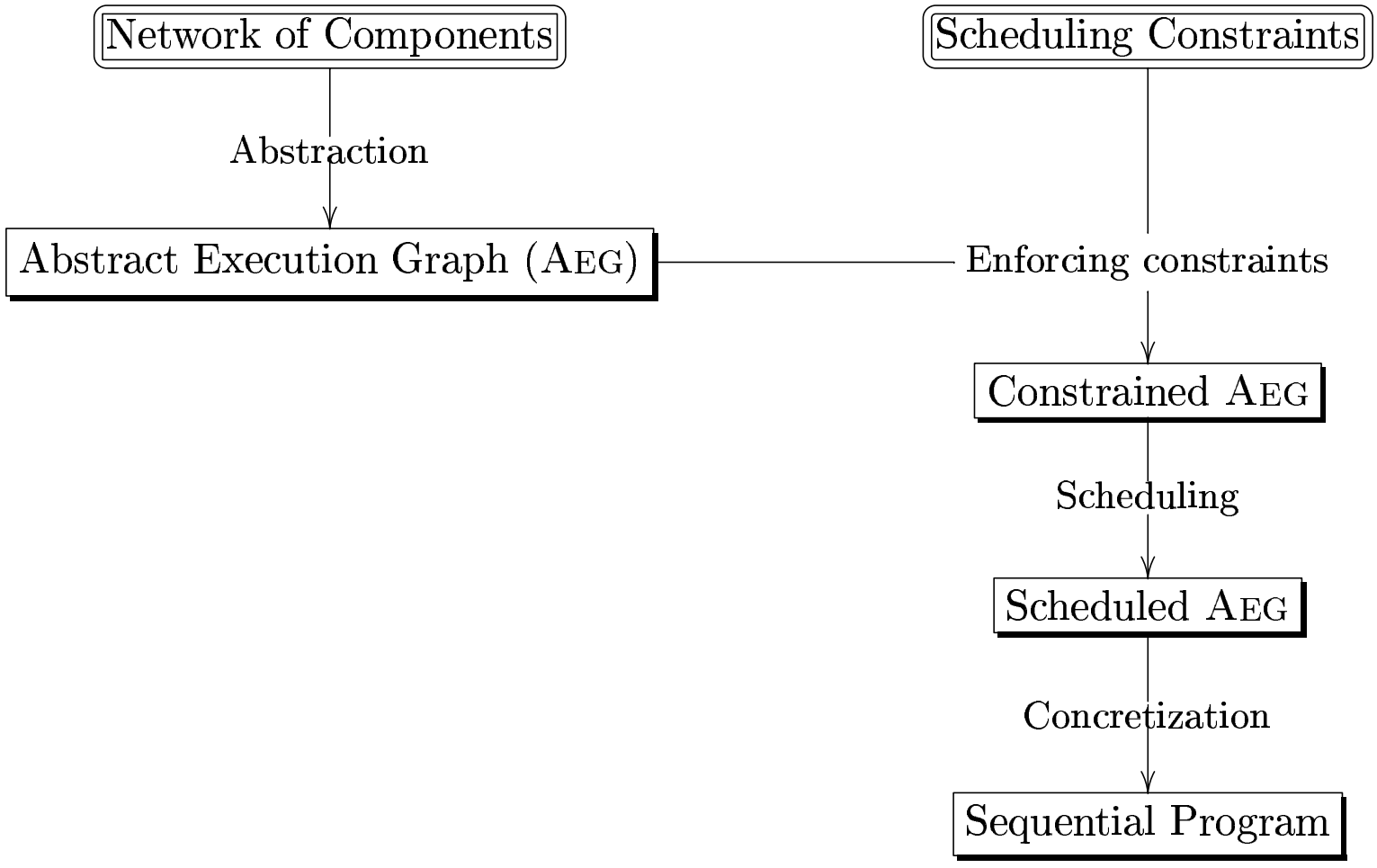


Enforcing constraint

- "Synchronized product" between AEG and constraints
- Example: Aeg $\| \Lambda_{FF}$

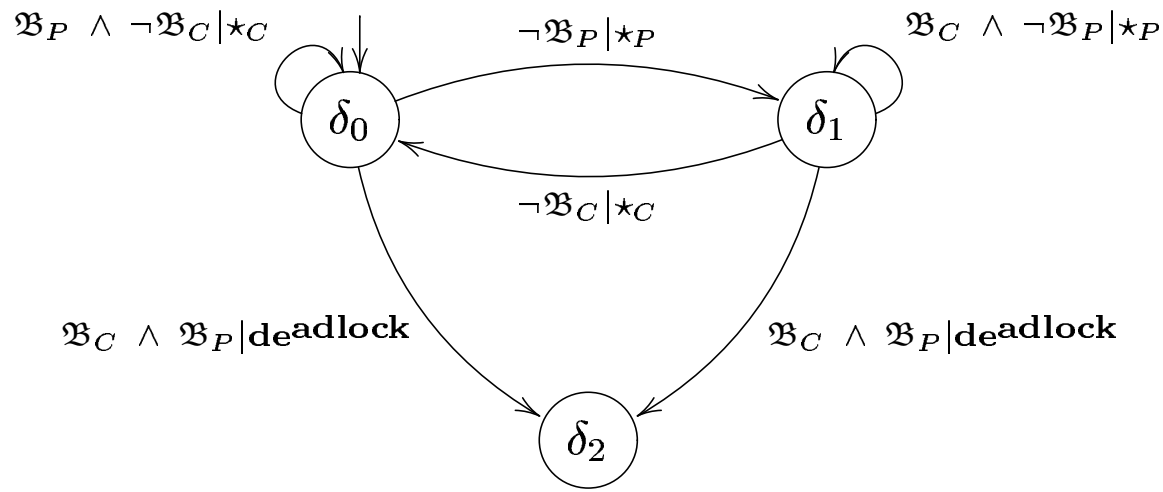


Overview

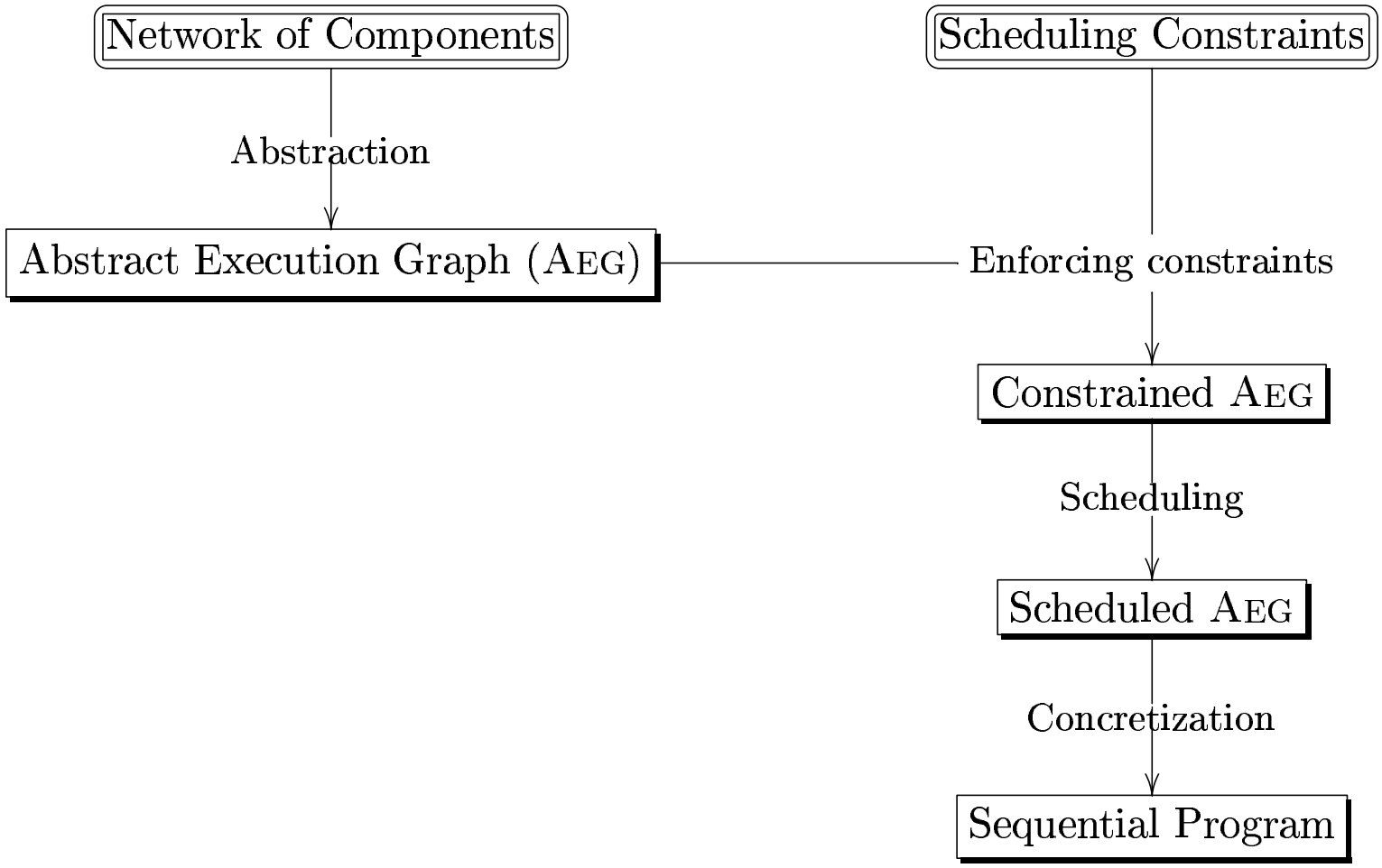

Network of Components
Abstraction
Abstract Execution Graph (AEG)
Scheduling Constraints
Enforcing constraints
Constrained AEG
Scheduling
Scheduled AEG
Concretization
Sequential Program

Completing scheduling

- Removing remaining choices by
  - Synchronized product with a standard sequential scheduler
  - Round-Robin expressed as a LTS:



Overview


Network of Components
Abstraction
Abstract Execution Graph (AEG)
Scheduling Constraints
Enforcing constraints
Constrained AEG
Scheduling
Scheduled AEG
Concretization
Sequential Program

Concretization

$$F = \left\{ \begin{array}{lcll} f_0 & : & x := x+1 & \rightsquigarrow \; f_1; \\ f_1 & : & f!x & \rightsquigarrow \; f_2; \\ f_2 & : & a < b \mid f?a & \rightsquigarrow \; f_3; \\ f_2 & : & a \geq b \mid \iota?b & \rightsquigarrow \; f_4; \\ f_3 & : & x := x+1 & \rightsquigarrow \; f_5; \\ f_4 & : & o!(a+b) & \rightsquigarrow \; f_2; \\ f_5 & : & f!x & \rightsquigarrow \; f_4 \end{array} \right\}$$

- then, replace internal IO instructions by assignments
  - $f!x \mapsto$ tmp-f:=x $f?x \mapsto$ x:=tmp-f
- and optimize code

Related work

- Filter Fusion
- Functionnal programming
  - Listlessness and deforestation
- Embedded/reactive systems
  - Focus on scheduling for FIFO Channel
  - No user-defined scheduling strategy and/or no fusion

Conclusion

- Generic Framework to fusion components
  - Simple, formal, and efficient compilation of components
  - Highly parametric (choice of abstraction, scheduling constraints, and sequential scheduler)
- 3 extensions
  - Linguistic support
  - More precise abstraction
  - Instrumented product
- Future work
  - Complete correctness proof
  - Implement a prototype