



UNIVERSITY OF TARTU

# Unraveling Unstructured Process Models

Marlon Dumas

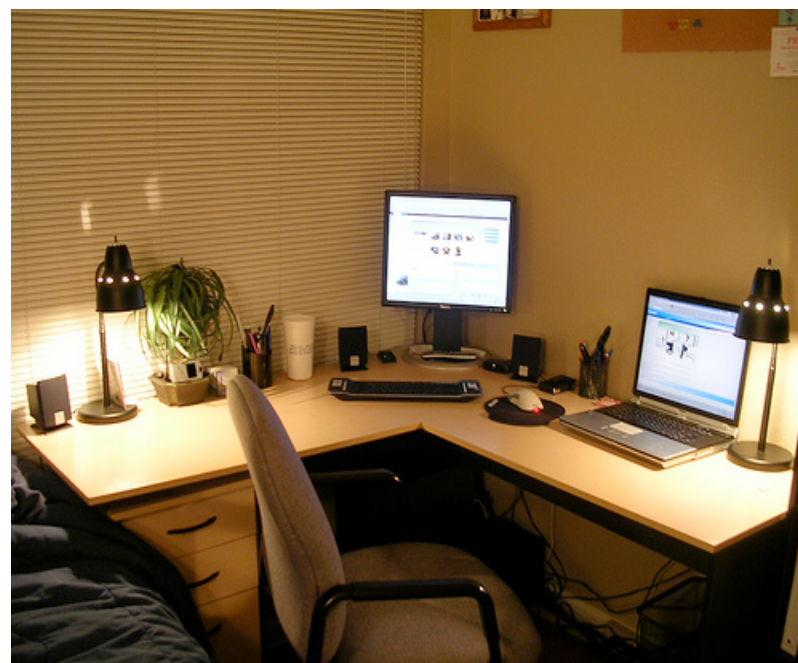
*University of Tartu, Estonia*

Joint work with Artem Polyvyanyy and Luciano García-Bañuelos

Invited Talk, BPMN'2010 Workshop, Potsdam, 14 Oct. 2010

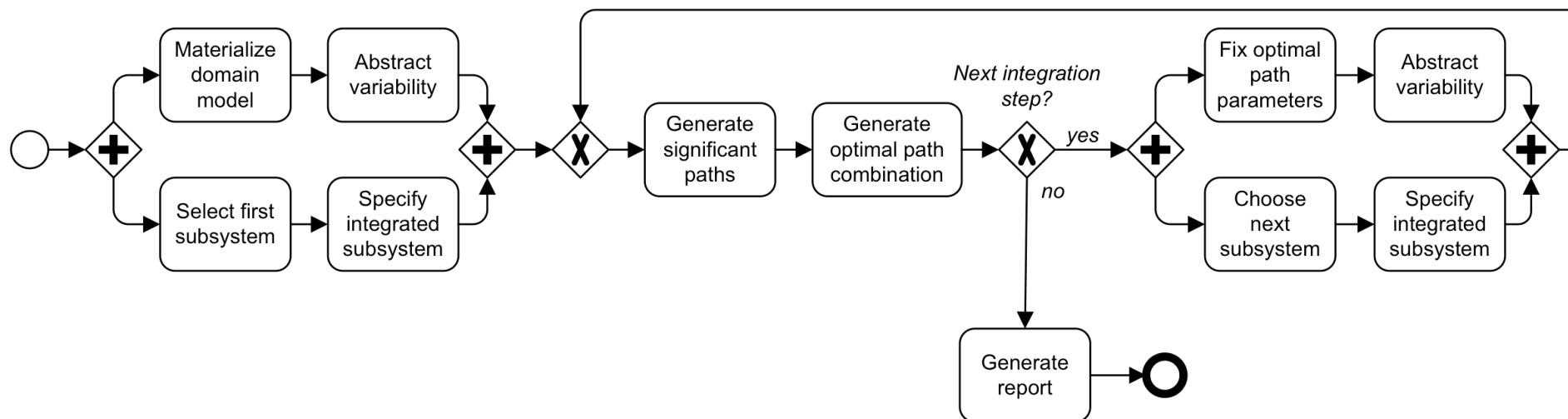
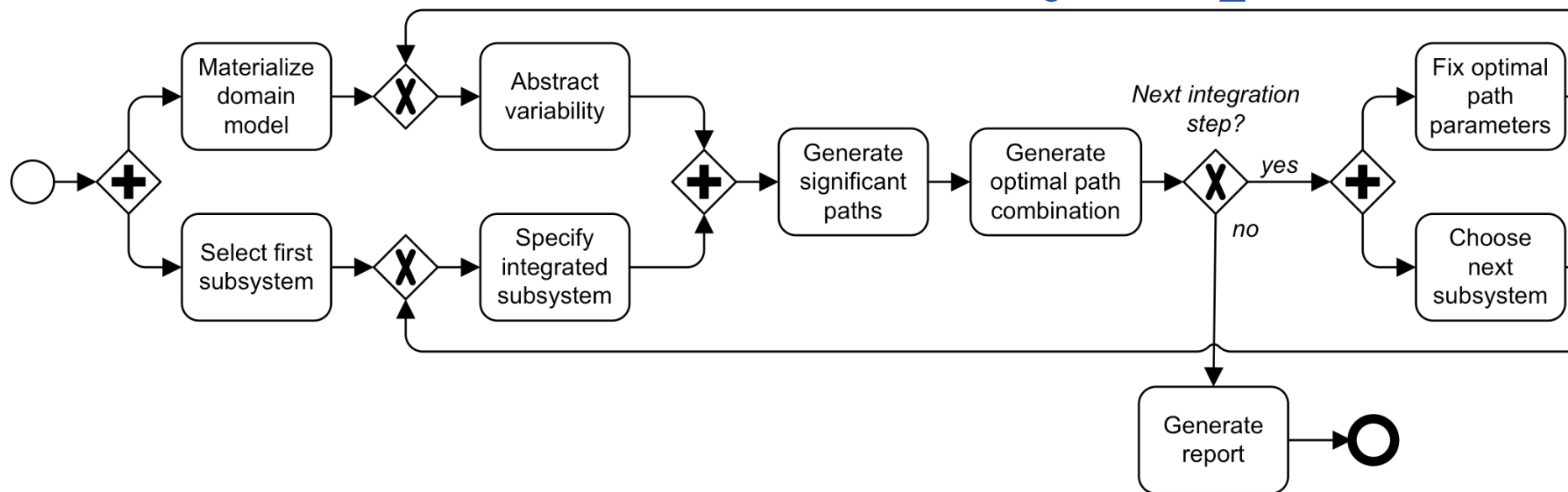


# Poll: Which desk do you prefer?





# Poll: Which model do you prefer?

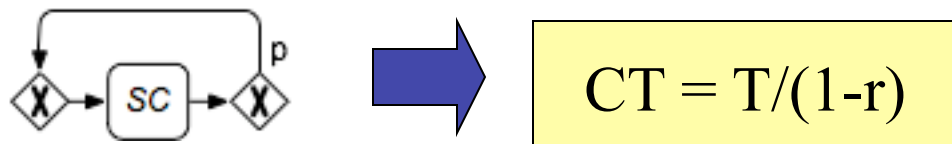
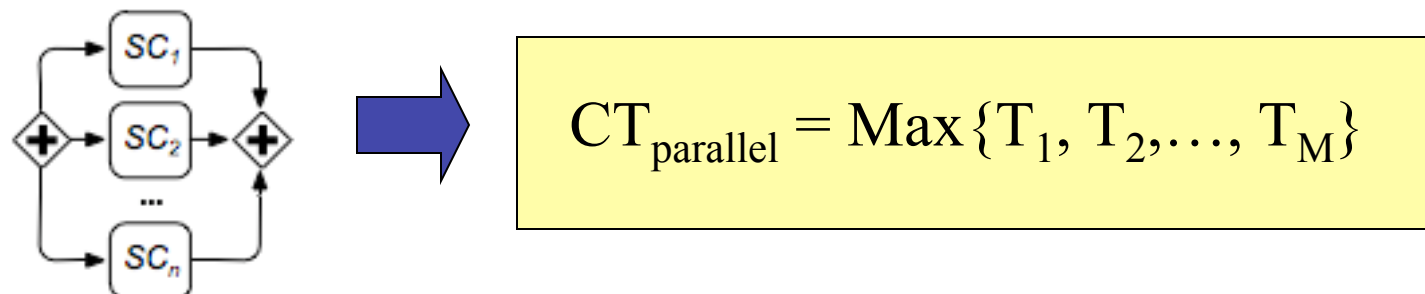
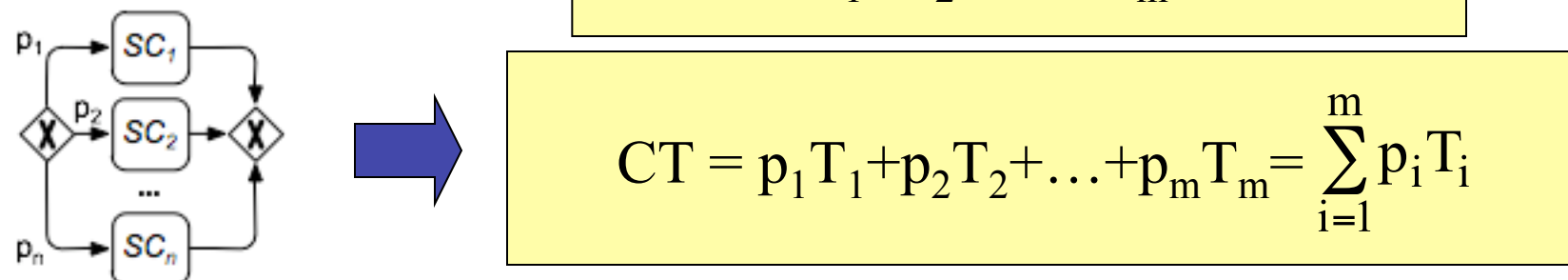
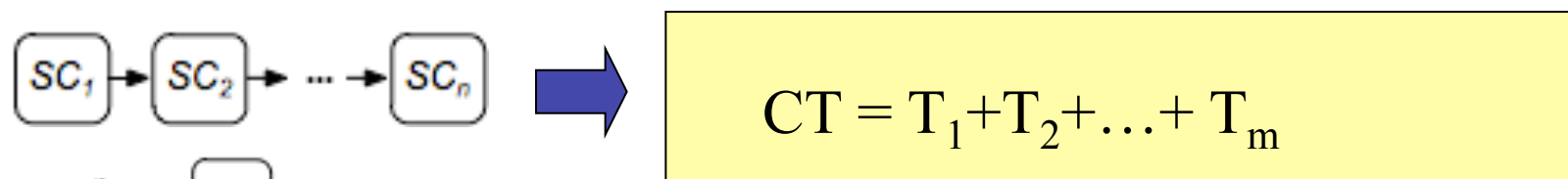




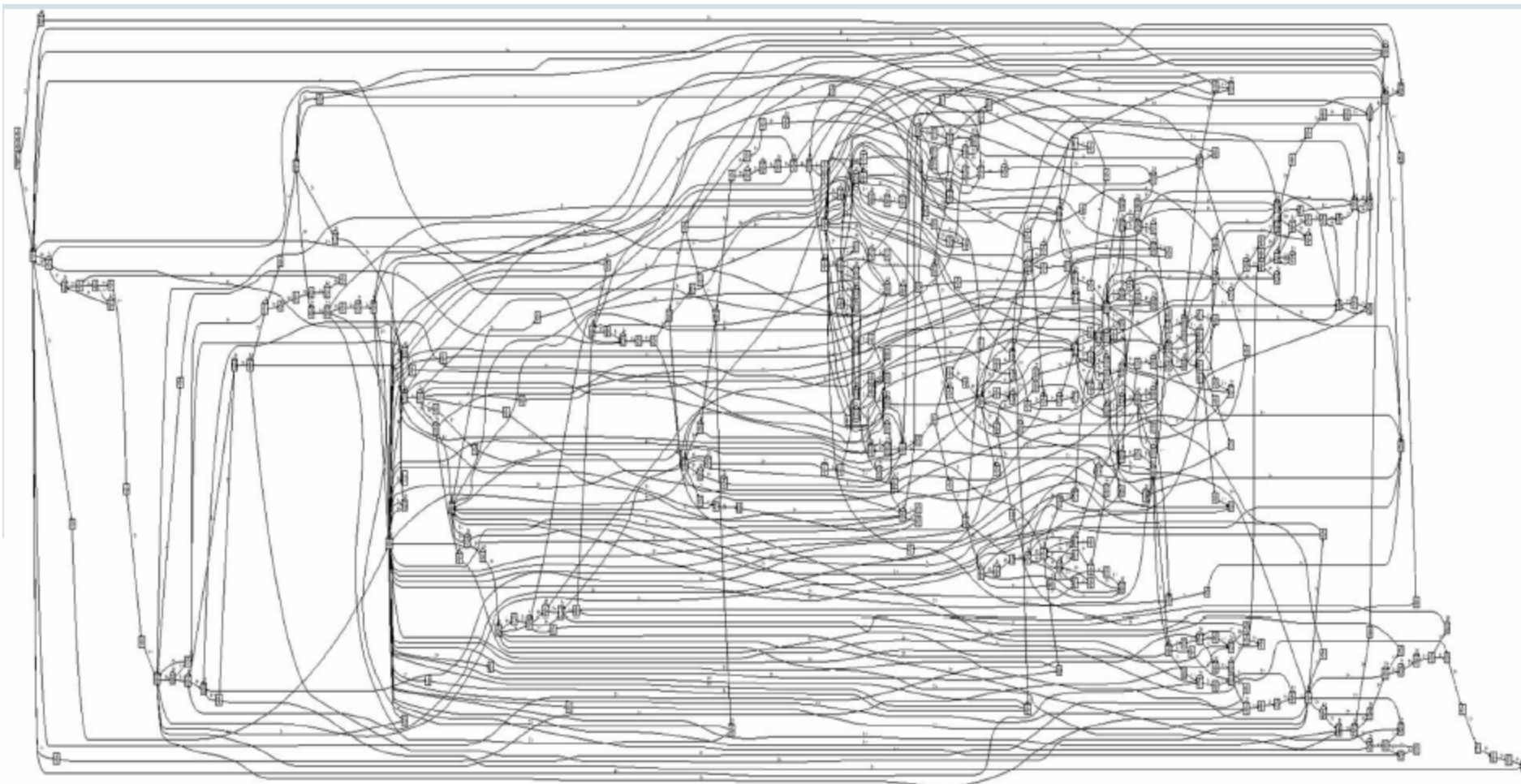
# The Problem

- Premise: Structured is “better”
  - Easier to understand
  - Easier to analyze
  - Easier to automatically layout
  - Easier to abstract (zoom-out)
- We know not all models can be structured...
- Which ones can, which ones can't?

# Analysis of Structured Models



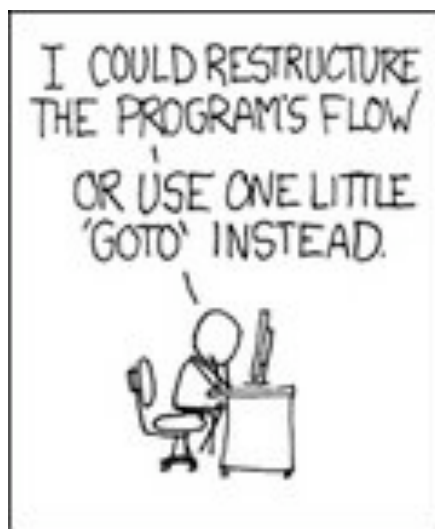
# Automated Layout and Abstraction





# A Bit of History

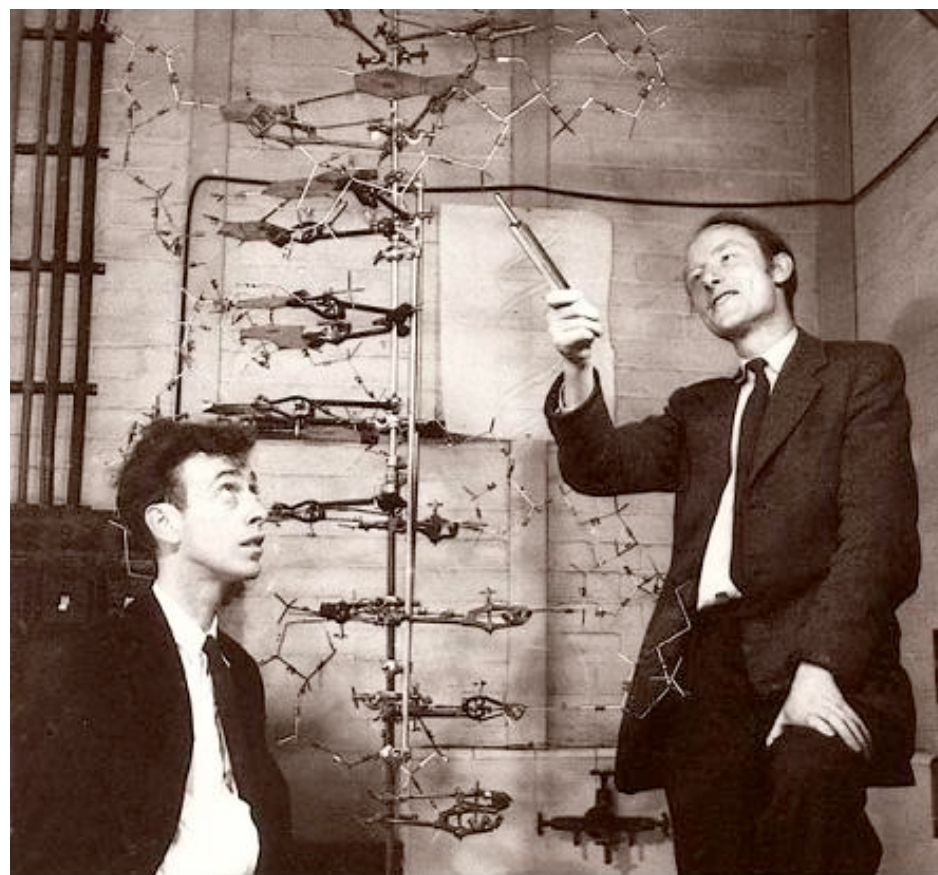
1968



1969



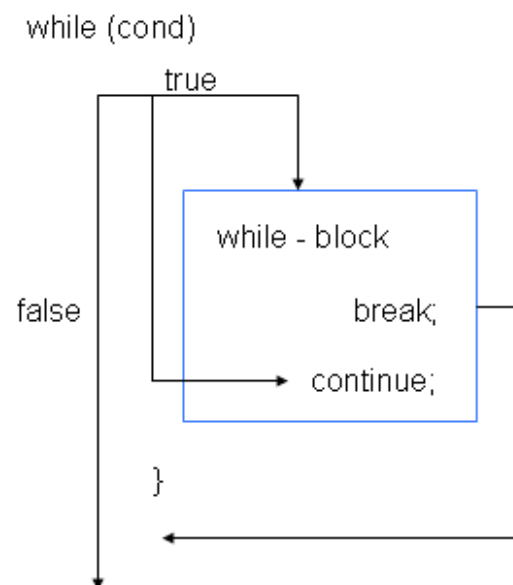
1970s (and 80s)



*"And here you see strong transformational skills..."*

## 20 years (and 200 papers) later...

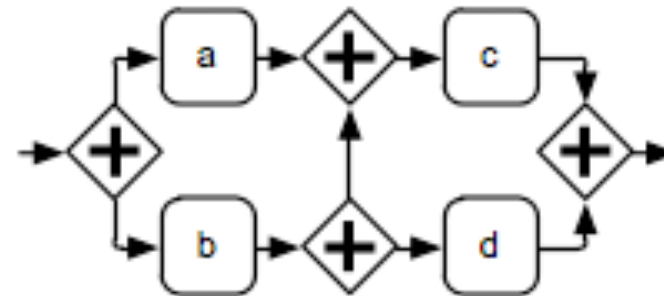
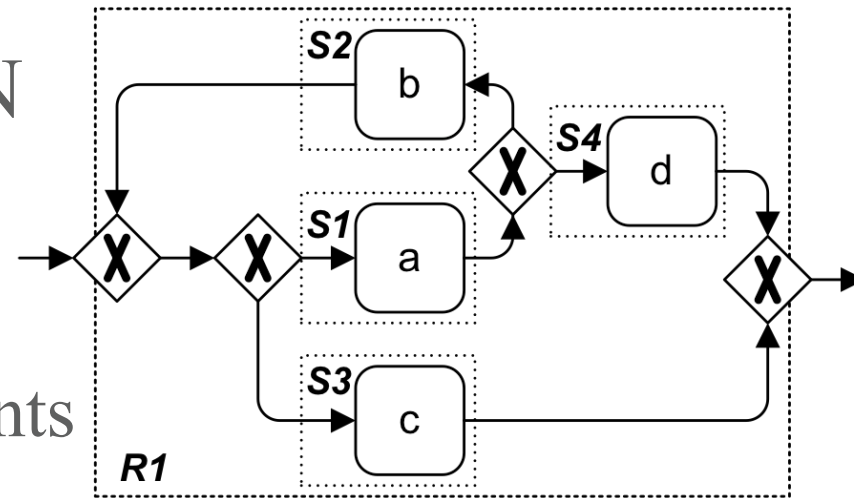
- Everything can be structured if you accept to break and continue



- Everyone forgot to release their implementation (they were ashamed since it was full of GOTOs)

# 40 years (and 400 papers) later

- We can structure any BPMN diagram, except:
  - “Incorrect” ones
  - Cycles with multiple exit points
  - Z-structures
  - Inclusive join gateways, complex gateways and other demons



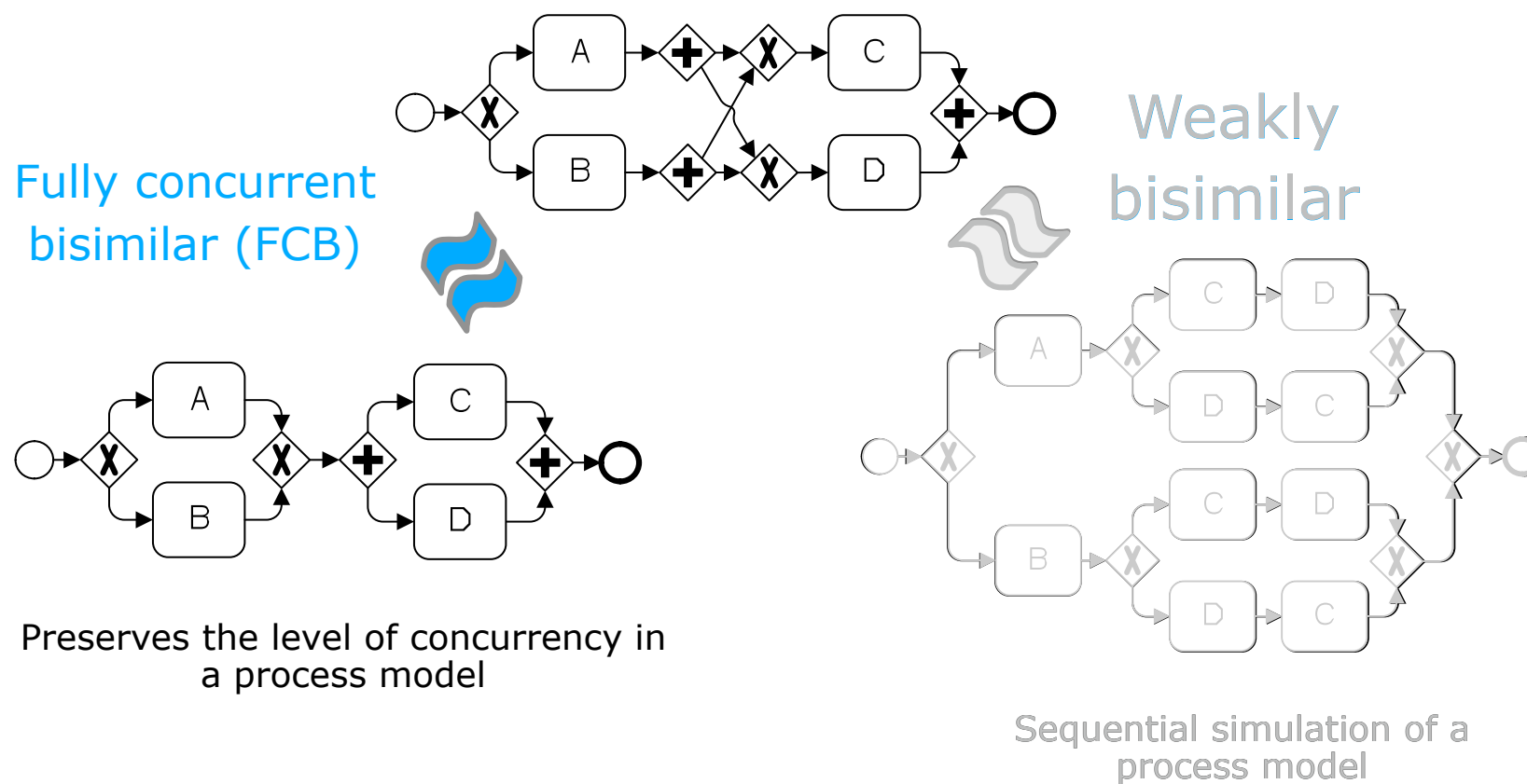
- Try it out: <http://sep.cs.ut.ee/Main/bpstruct>



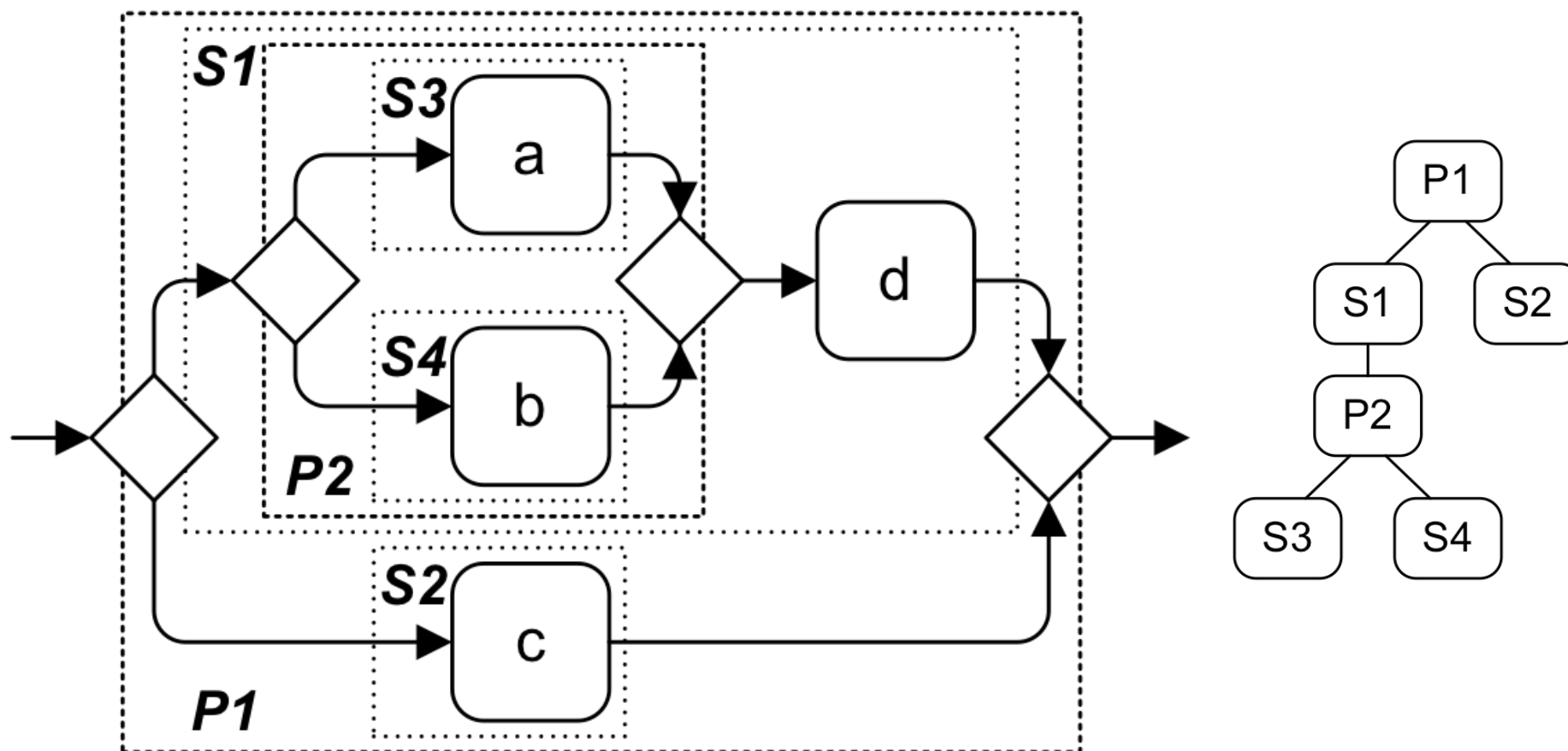
## Corollary (if you care)

- Any (sound) BPMN model can be transformed into an equivalent (readable) BPEL process definition
  - If BPEL had break/continue statements or we use boolean variables to simulate break/continue statements
  - And the BPMN model does not have inclusive join gateways, complex gateways and other demons
  - And some other minor details not worth mentioning

# Behavioral Equivalence

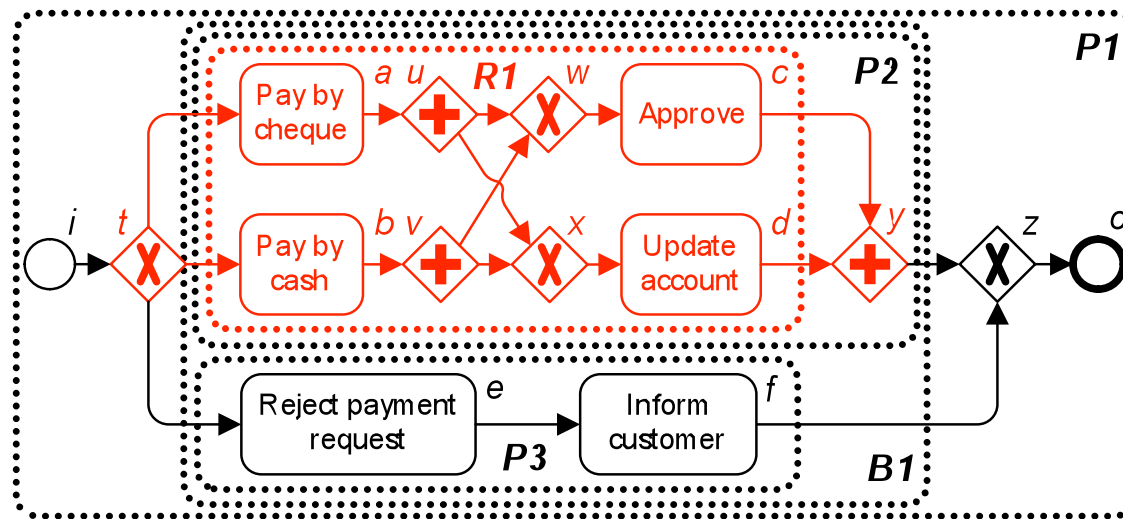
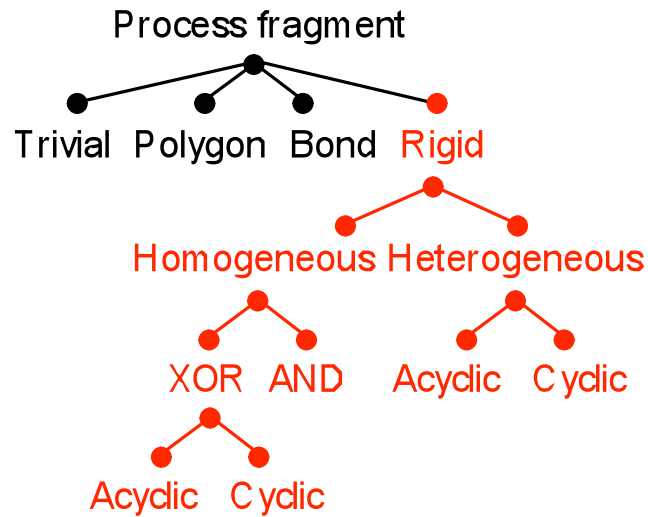


# Starting Point – Process Structure Tree

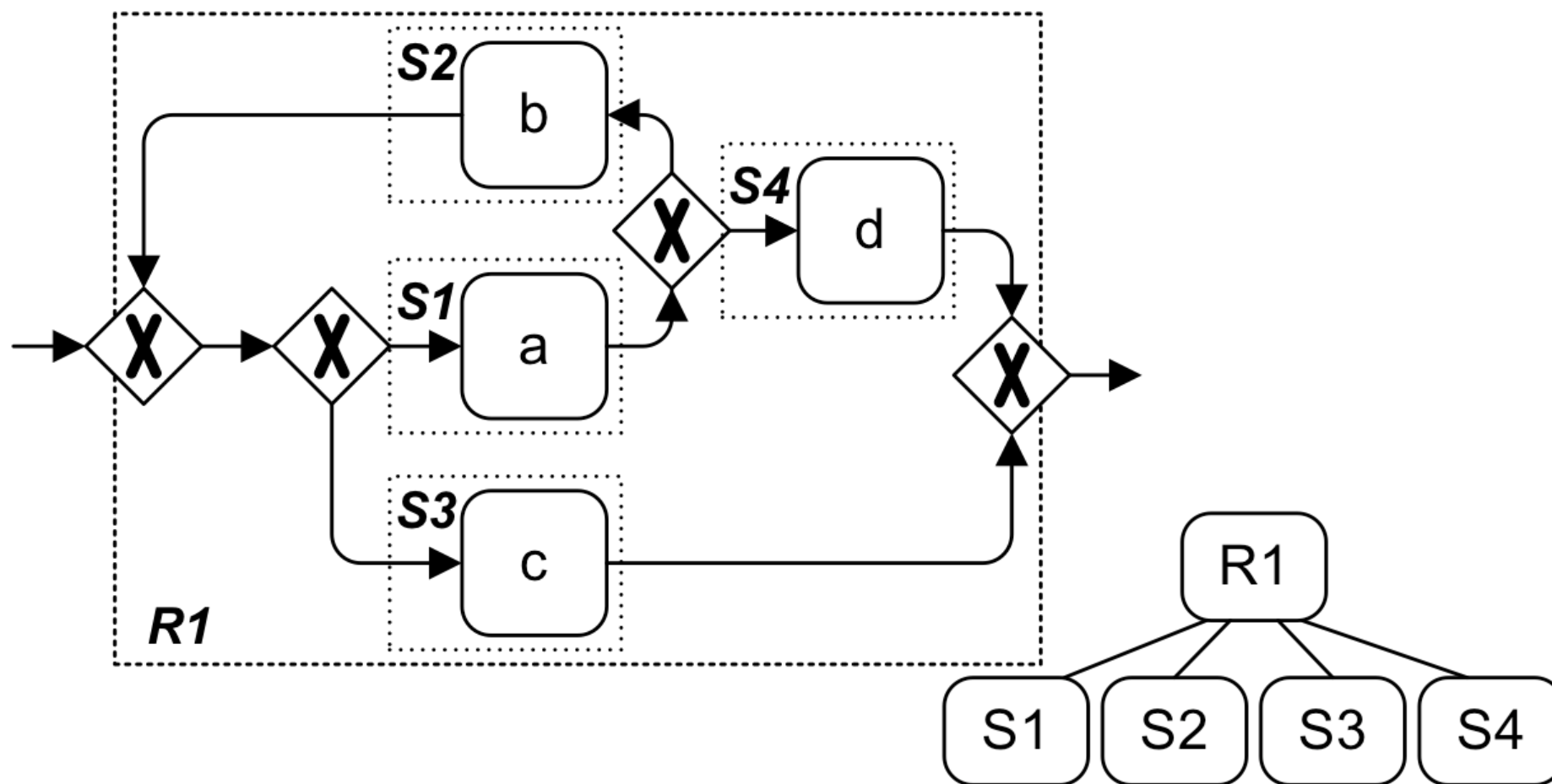


# Taxonomy of Process Fragments

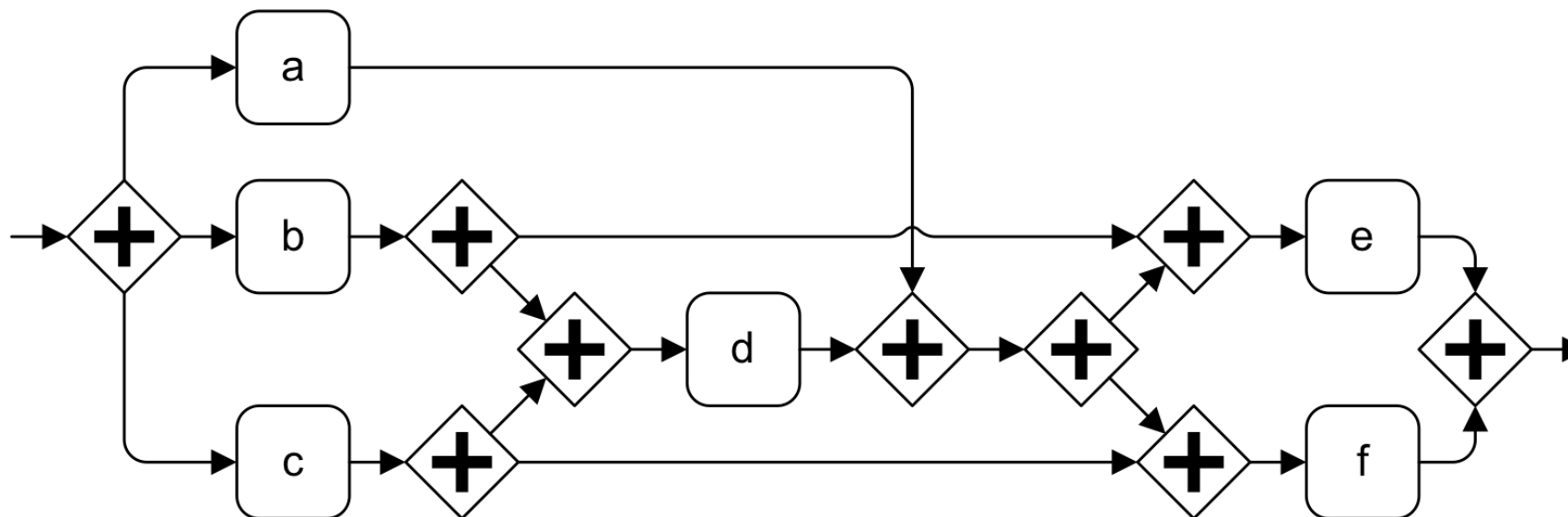
- Trivials, polygons, and bonds are structured fragments
- Rigids are “unstructured”



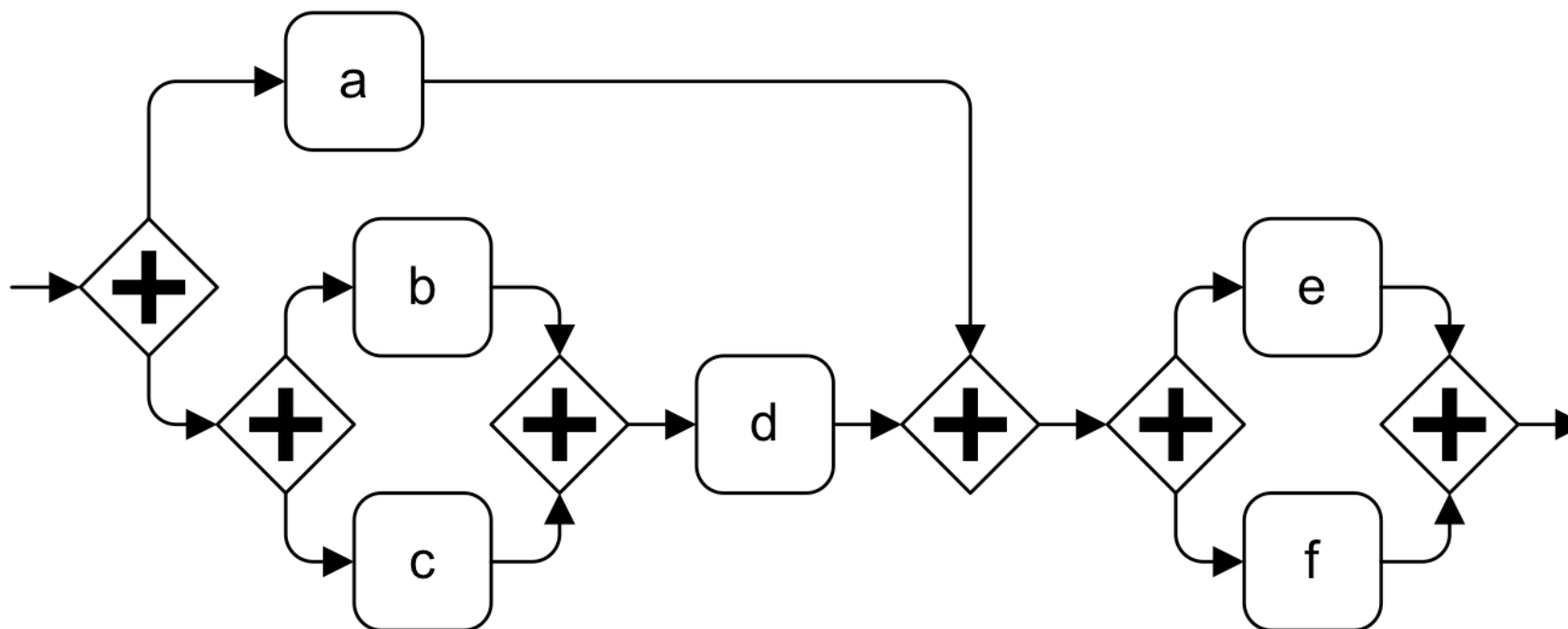
# Homogeneous XOR Rigid



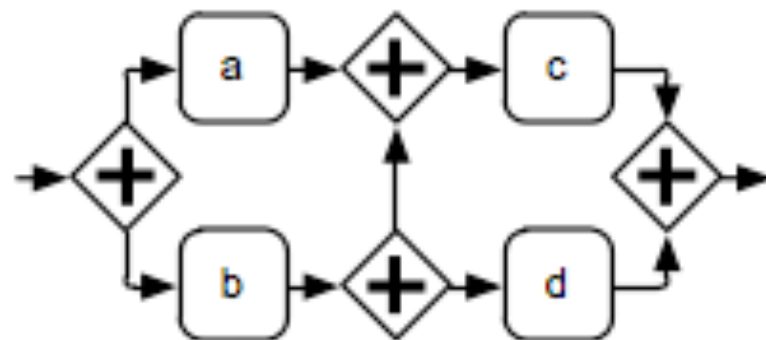
# Homogeneous AND Rigid



## Block-structured version...

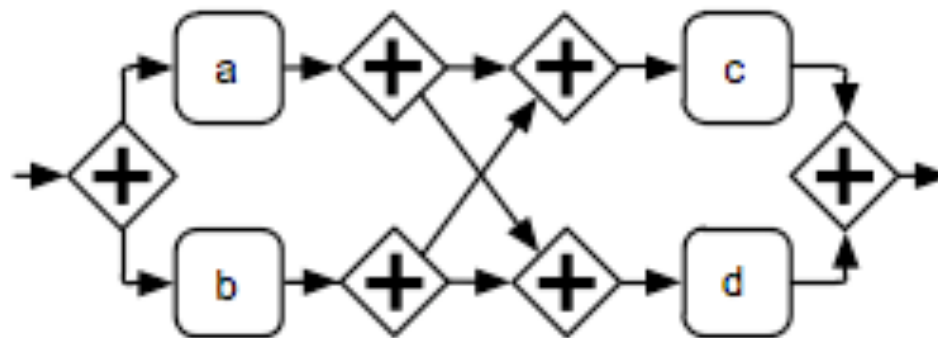


# Homogeneous AND Rigid that cannot be structured



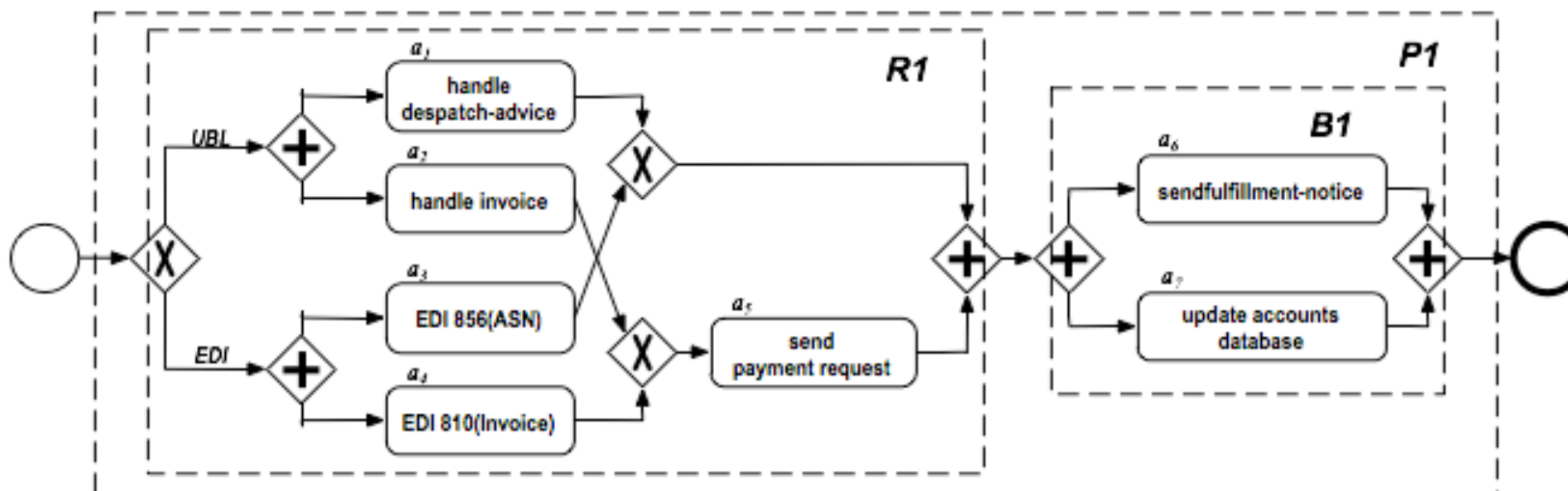
- Causal rules:
  - $\{A, B\} \rightarrow \{C\}$
  - $\{B\} \rightarrow \{D\}$
- Overlap on the left-hand side of the rules

## Compare to this...

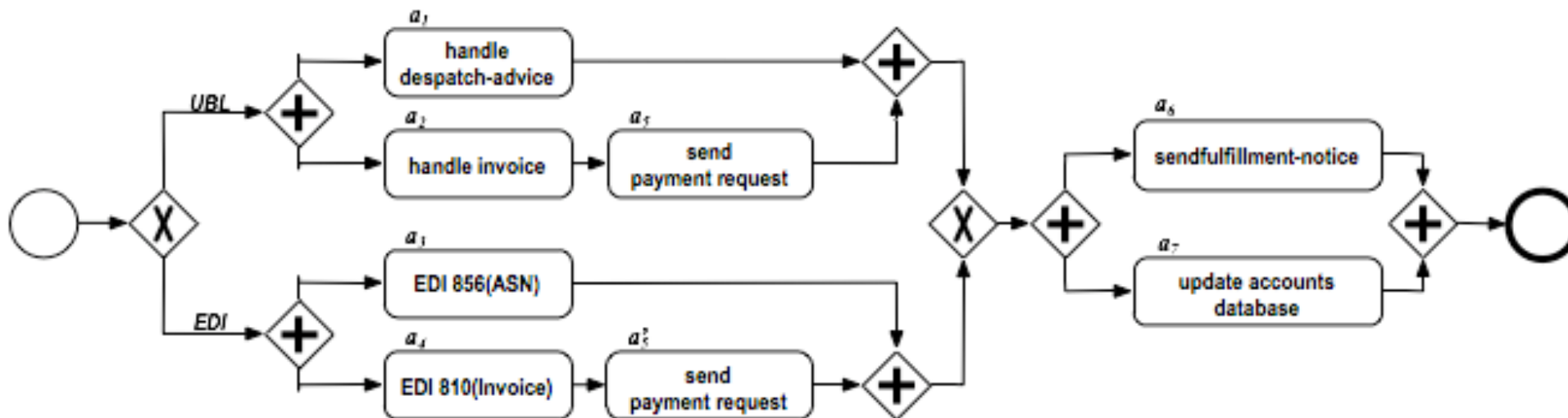


- Causal relations
  - $\{A, B\} \rightarrow \{C, D\}$

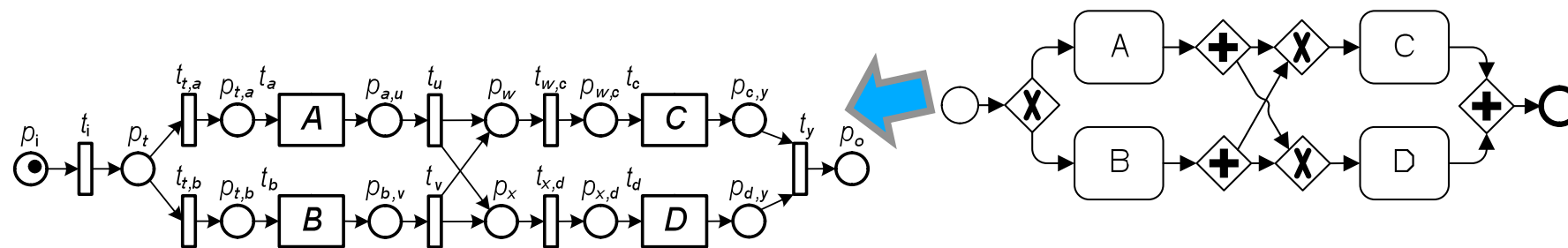
# Heterogeneous Acyclic Rigid



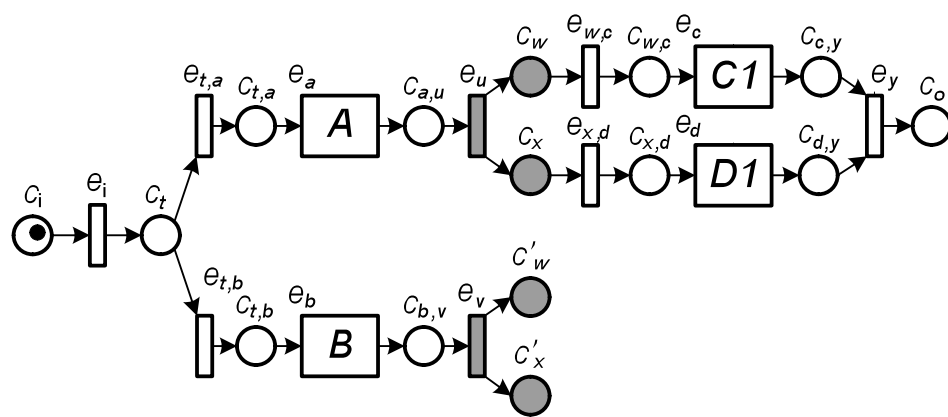
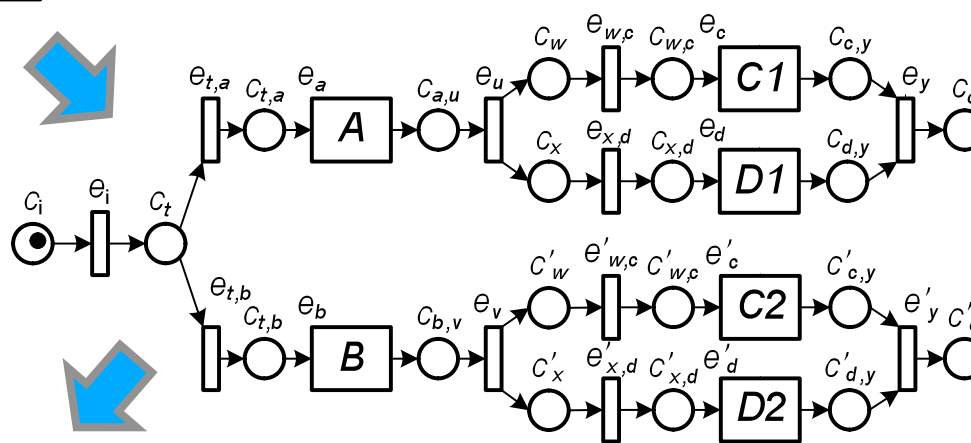
# Equivalent Structured Fragment



# The Key Ingredient: Unfoldings

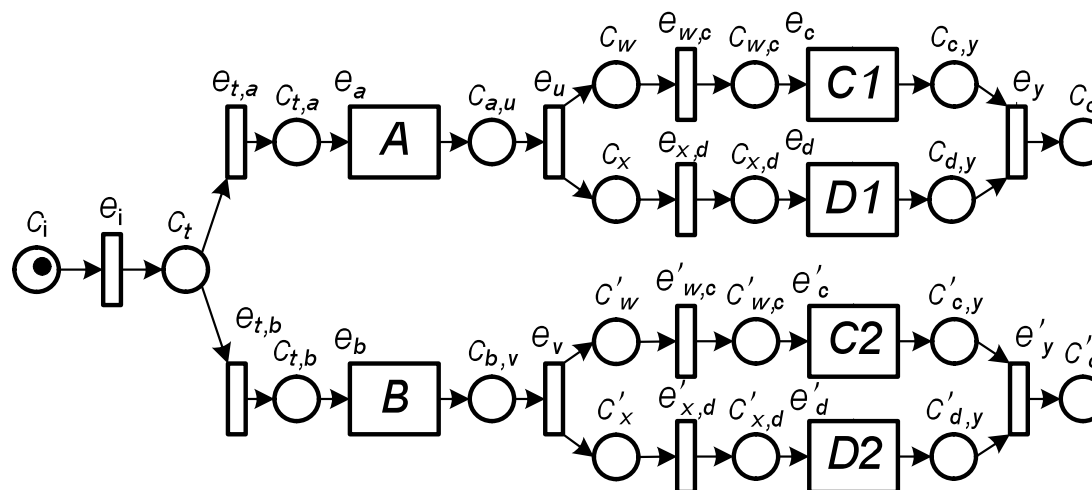


An unfolding is a representation of a net without "merge" points



A **complete prefix unfolding** is a **finite** initial part of the **unfolding** that contains **full information** about the reachable states

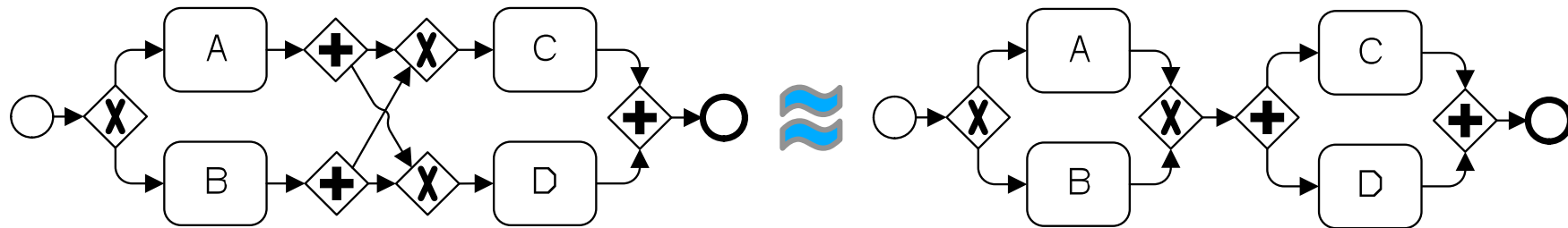
# Ordering Relations

 $A > C1$ 
 $D2 \parallel C2$ 
 $A \# D2$ 
 $B \# A$ 
 $C2 \parallel D2$ 
 $B > D2$ 


- Two transitions of an occurrence net are in one of the following relations:
  - A and B are in *causal* relation ( $A > B$ ), iff there exists a path from A to B
  - A and B are in *conflict* ( $A \# B$ ), iff there are two transitions  $t_1, t_2$  that share an input place and there is a path from  $t_1$  to A and a path from  $t_2$  to B
  - A and B are in *concurrency* ( $A \parallel B$ ) relation iff A and B are neither in causal, nor in conflict relation


# FCB and Ordering Relations

Two process models are FCB-equivalent ...



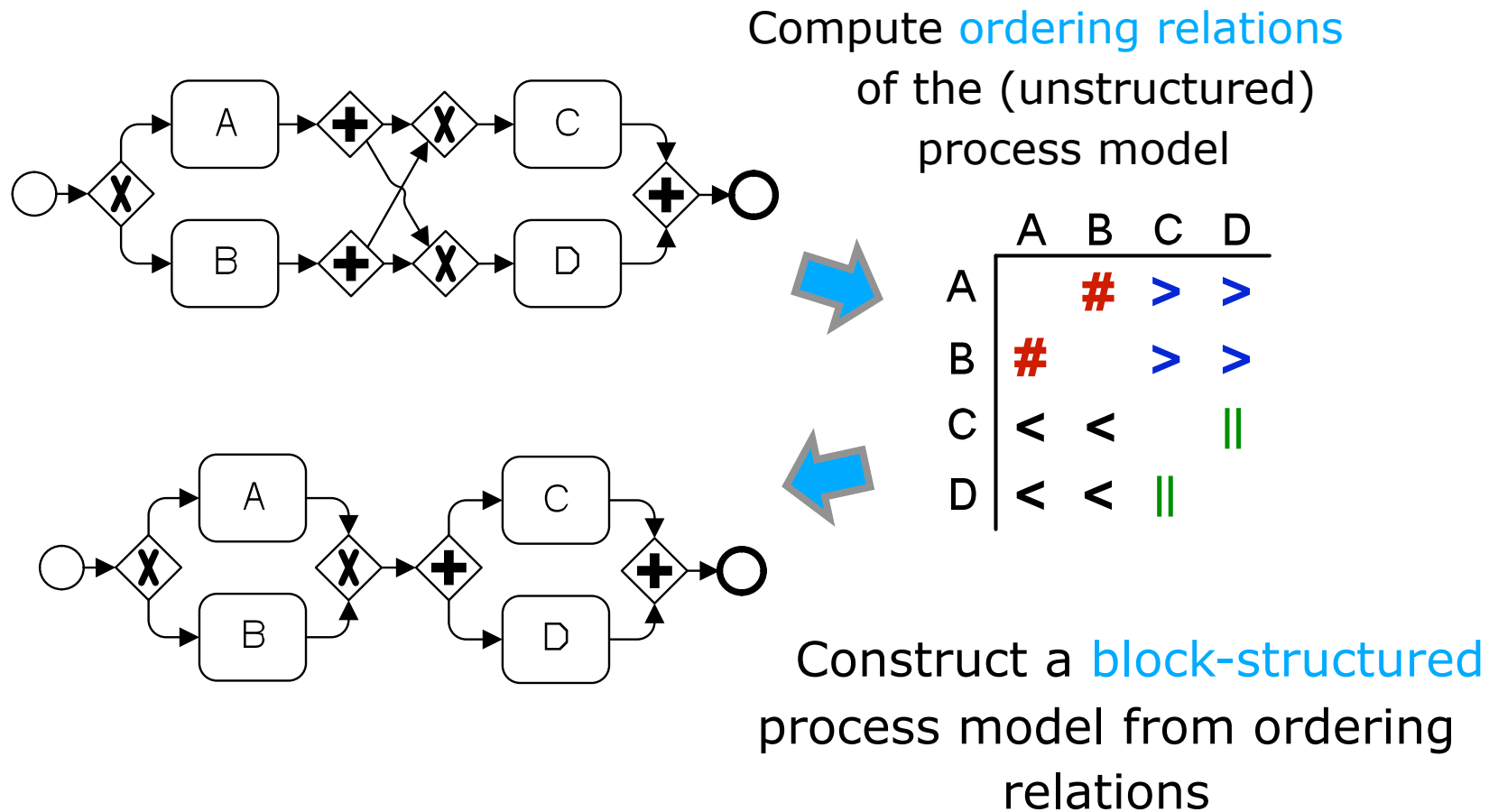
... if and only if, (complete prefix) unfoldings of both models expose same ordering relations

	A	B	C	D
A		#	>	>
B	#		>	>
C	<	<		
D	<	<		



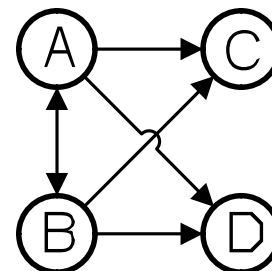
	A	B	C	D
A		#	>	>
B	#		>	>
C	<	<		
D	<	<		

# Structuring Process Models



# Ordering Relations Graph

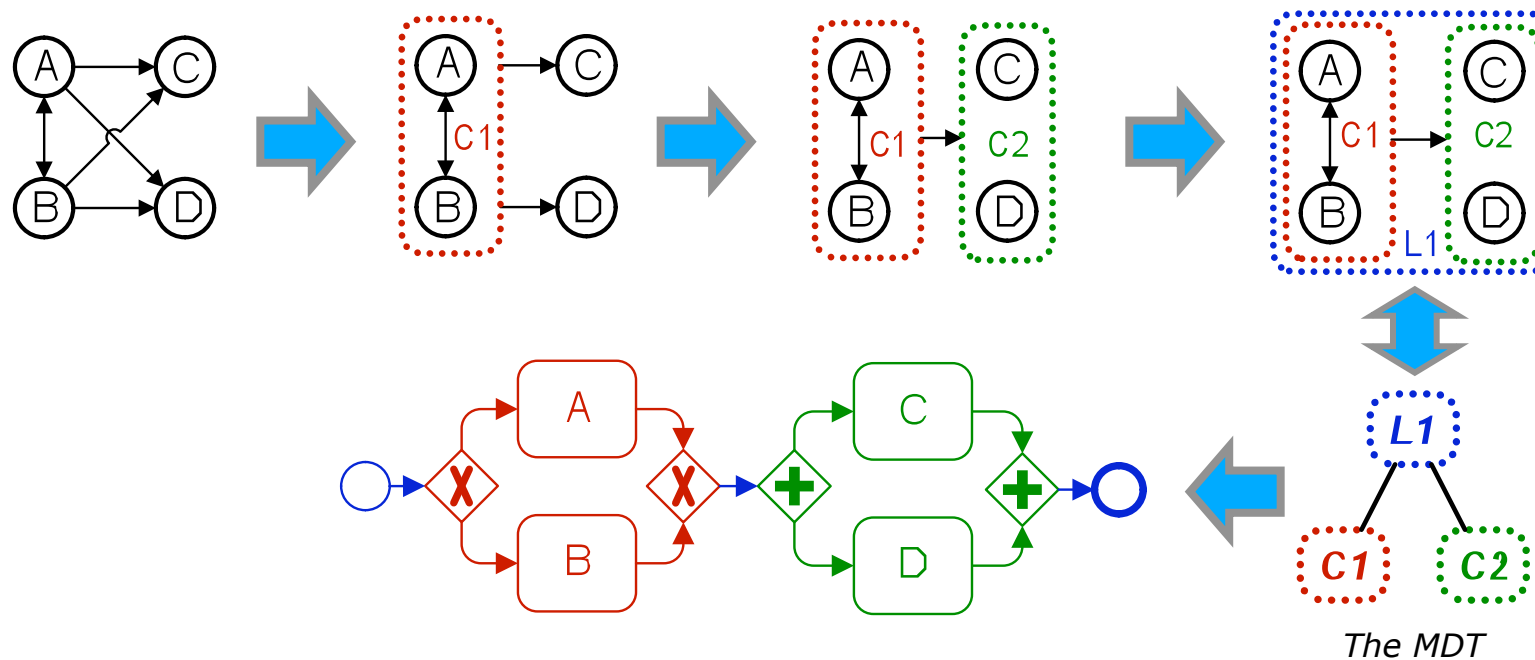
	A	B	C	D
A		#	>	>
B	#		>	>
C	<	<		
D	<	<		



*An ordering relations graph*

# Modular Decomposition Tree (MDT)

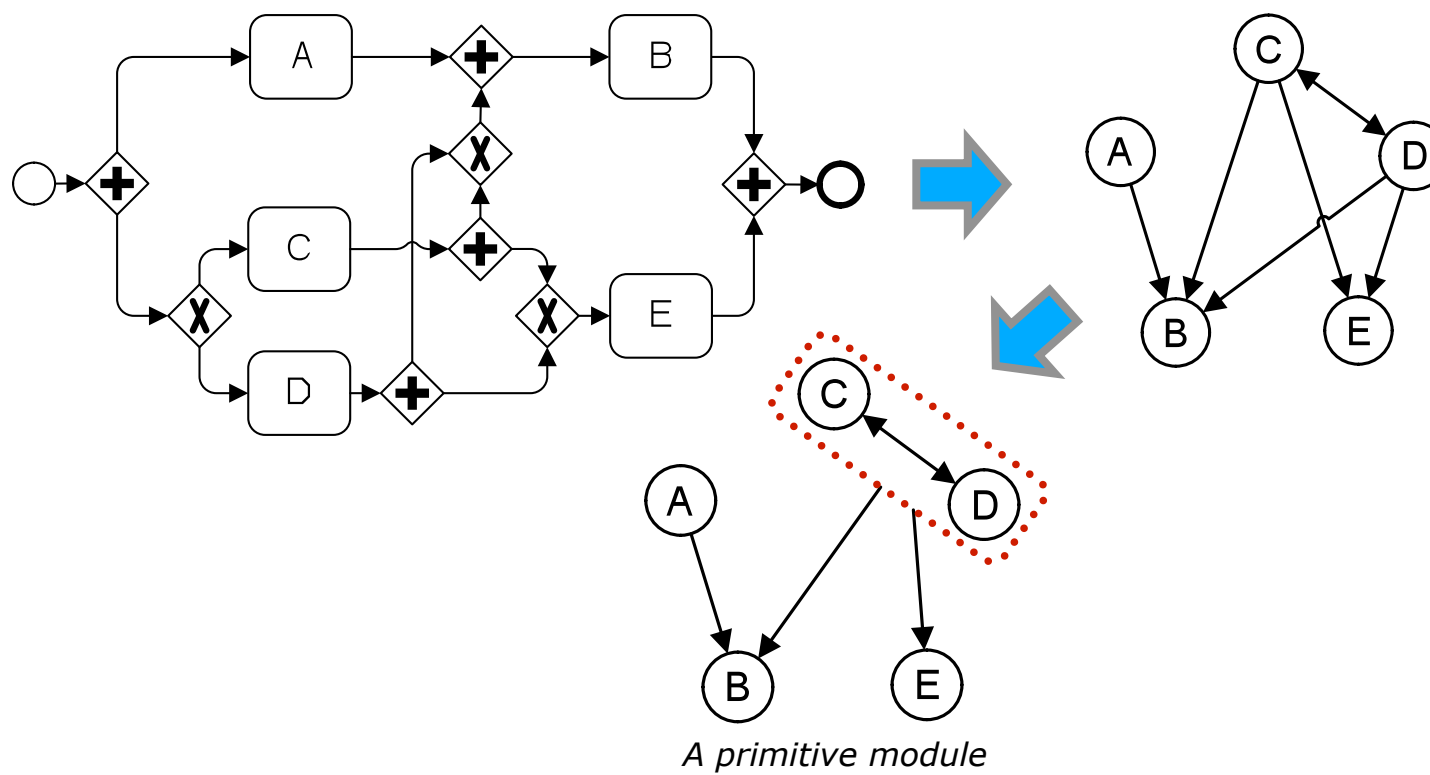
- A module is a set of edges with uniform structure
  - A *linear* (L) module is a total order on a set of nodes of a graph
  - A *complete* (C) module is a complete graph, or a clique
  - A *primitive* (P) module is neither trivial, nor linear, nor complete
- 
- The MDT is *unique* and can be computed in *linear time*



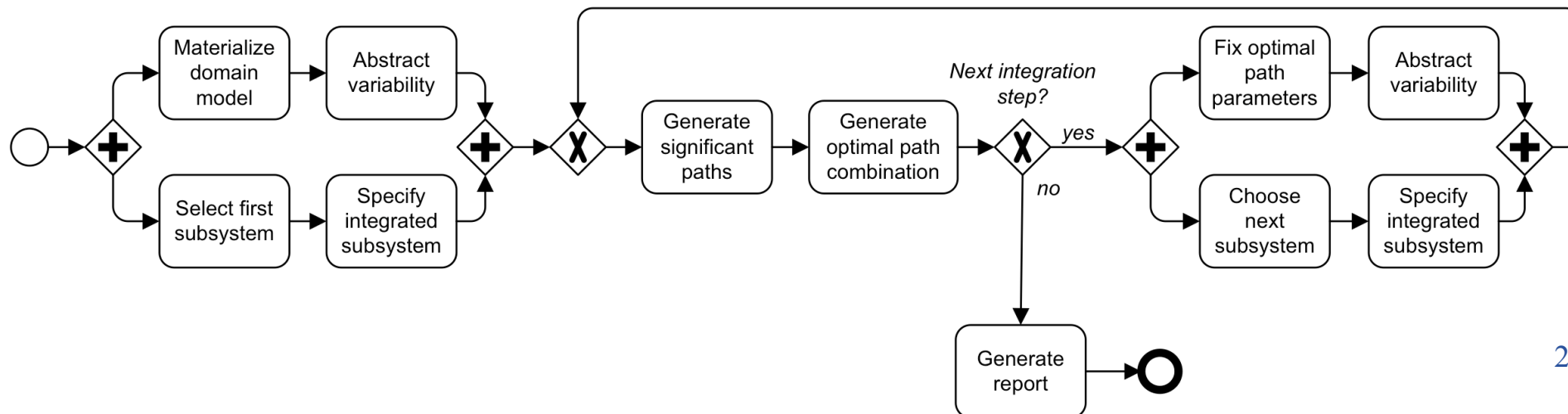
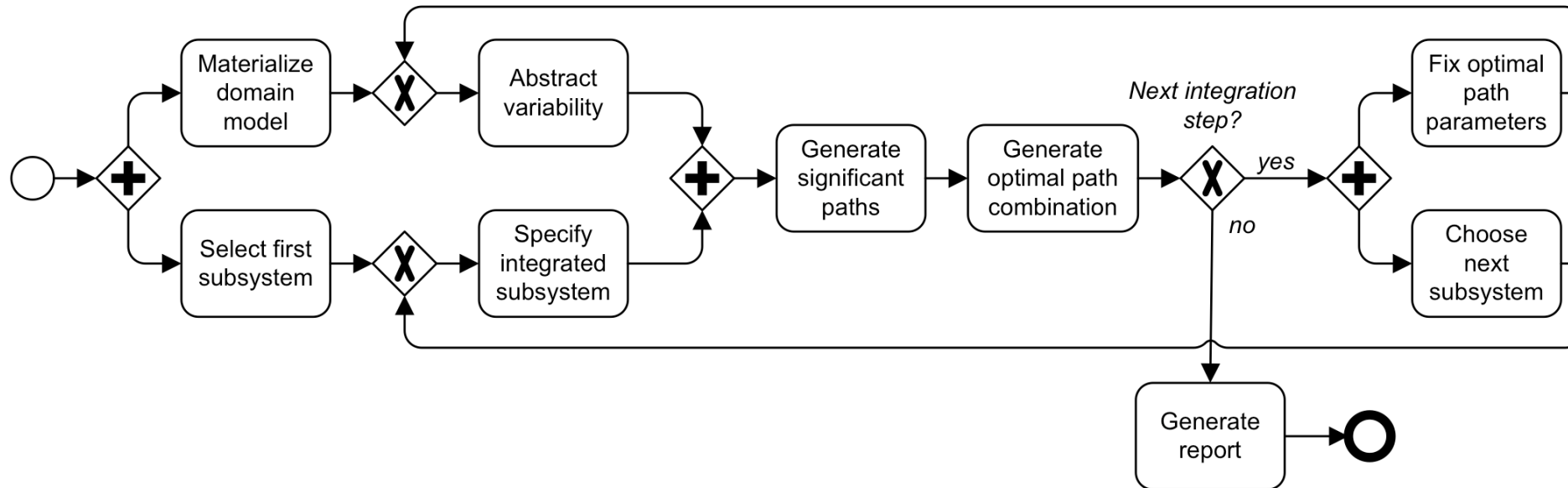
The MDT

# Structuring Acyclic Process Models

Let  $G$  be an ordering relations graph. The MDT of  $G$  has no primitive module, iff there exists a well-structured process model  $W$  such that  $G$  is the ordering relations graph of  $W$ .



# Heterogeneous Cyclic Rigid





## For further details...

Download and try:

- <http://sep.cs.ut.ee/Main/bpstruct>
- <http://code.google.com/p/bpstruct/>

Perspectives:

- Analyze
- Decompose (e.g. for distributed execution)
- Refactor (extract duplicate fragments into subprocesses)
- Visualize