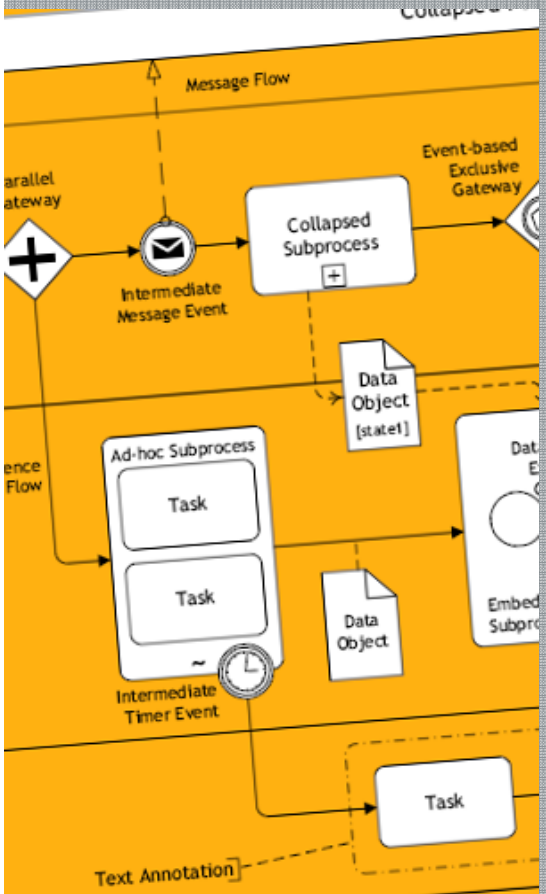


HPI Hasso Plattner Institut

IT Systems Engineering | Universität Potsdam

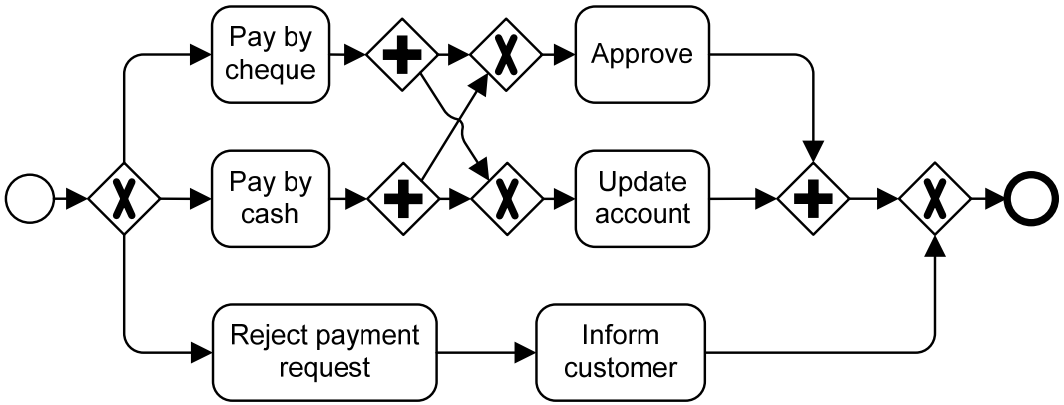
Structuring Acyclic Process Models

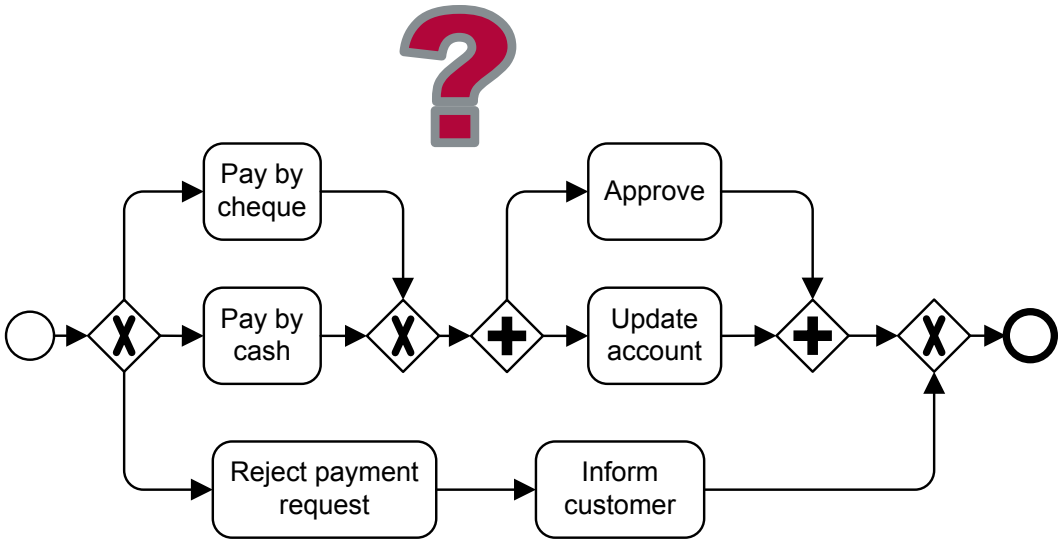
Artem Polyvyanyy
Luciano García-Bañuelos
Marlon Dumas

Business Process Management
September 2010, Hoboken, NJ

Motivation:
Graph- or Block-structured Modeling?

A process model is *block-structured* if splits and joins are always paired into Single-Entry-Single-Exit (SESE) fragments; otherwise *graph-structured*



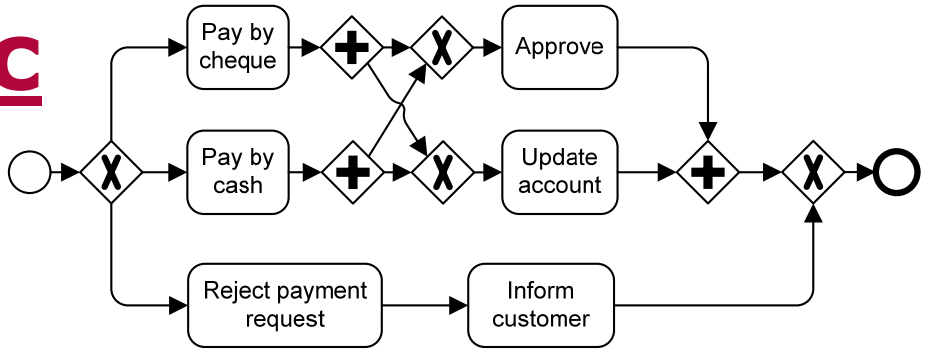


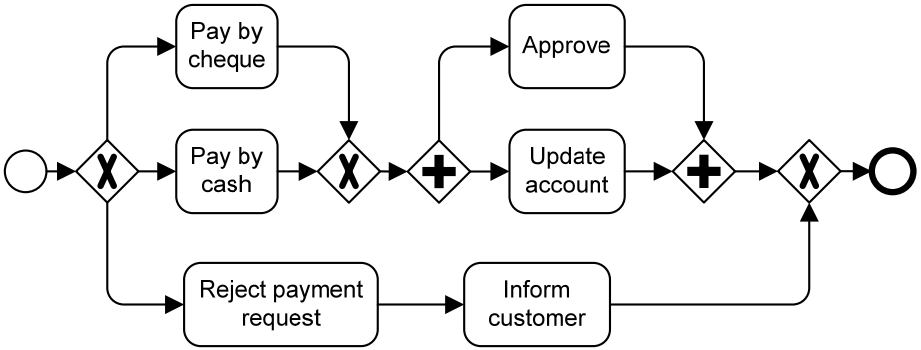
Premise: block-structured is "better"

- Easier to understand
- Less error-prone
- Easier to automatically layout
- Easier to analyze (aggregate QoS, time constraint checking, etc.)
- Easier to abstract (zoom in/out)
- Can be expressed by block-structured languages, e.g., BPEL
- Easier execution optimization (resource allocation)

Research Problem:
Structuring Process Models

Given an **arbitrary** **acyclic** (graph-structured) process model …





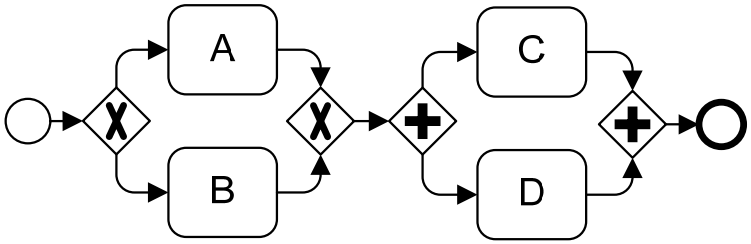
Does a behaviorally equivalent **block-structured** process model **exist?**

And if such a model exists, **how** can it be **constructed?**

Behavioral Equivalence: *Bisimulation*

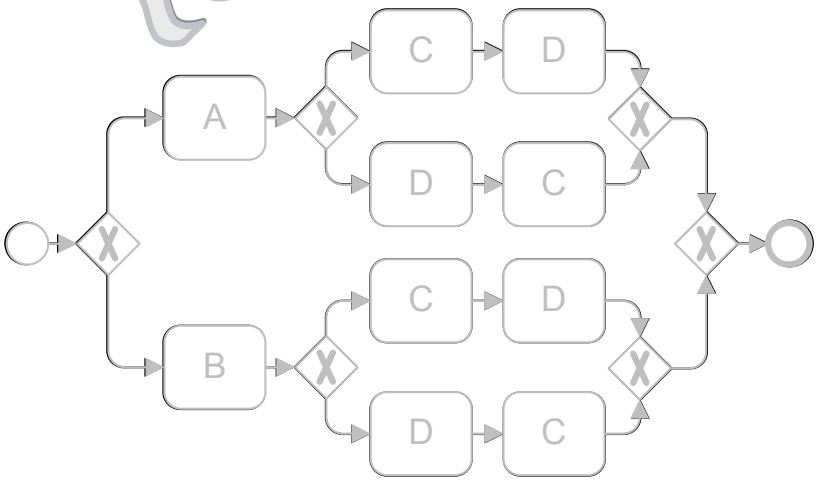
- There exist many notions of behavioral equivalence [van Glabbeek 1990]

Fully concurrent bisimilar (FCB)



Preserves the level of concurrency in a process model

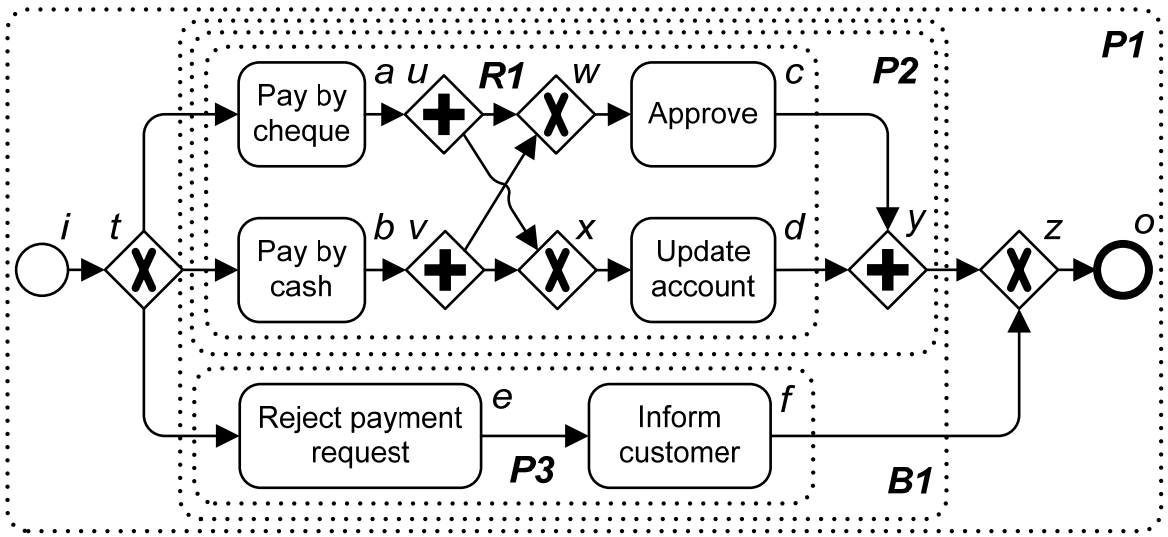
Weakly bisimilar

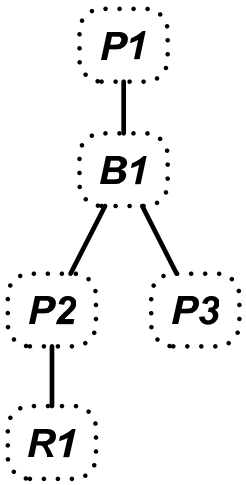


Sequential simulation of a process model

The Refined Process Structure Tree (RPST)

- The RPST is a technique for parsing process models into a hierarchy of *canonical* Single-Entry-Single-Exit (SESE) fragments
  [Vanhatalo, Völzer, and Koehler 2008], [Polyvyanyy, Vanhatalo, and Völzer 2010]
- The RPST is *unique*, *modular*, and can be computed in *linear time*

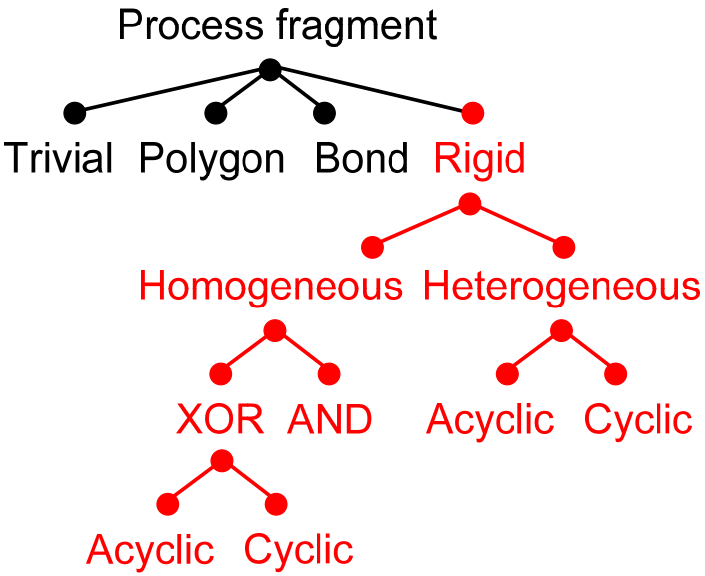


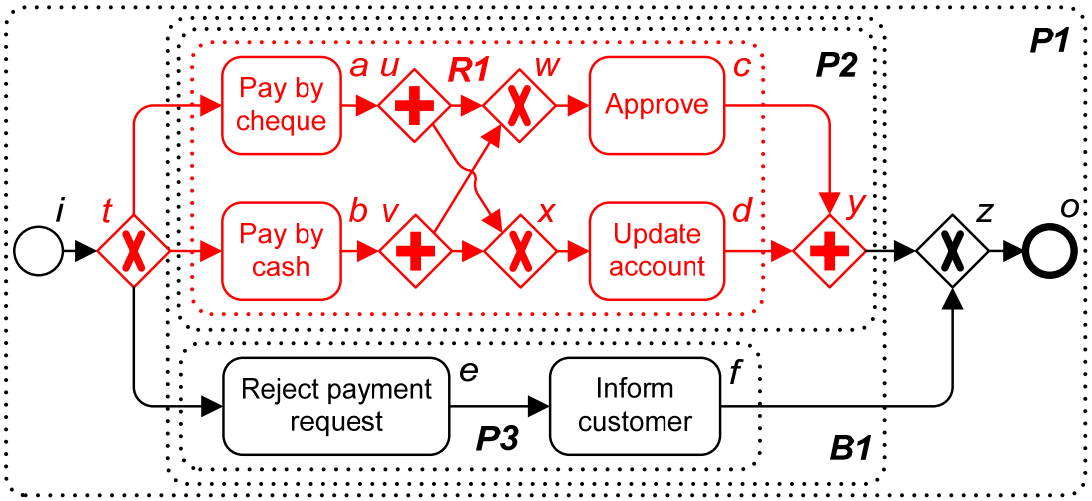


- A *trivial* (T) fragment consists of a single edge, e.g., (*i*,*t*)
- A *polygon* (P) fragment consists of a sequence of fragments, e.g., ((*i*,*t*), B1, (*z*,*o*))
- A *bond* (B) fragment consists of a set of fragments that share two nodes, e.g., {P2, P3}
- A *rigid* (R) fragment is neither trivial, nor polygon, nor bond, e.g., fragment R1

Taxonomy of Process Fragments

- Trivials, polygons, and bonds are structured process fragments
- Rigid fragments explicitly define what makes process models unstructured

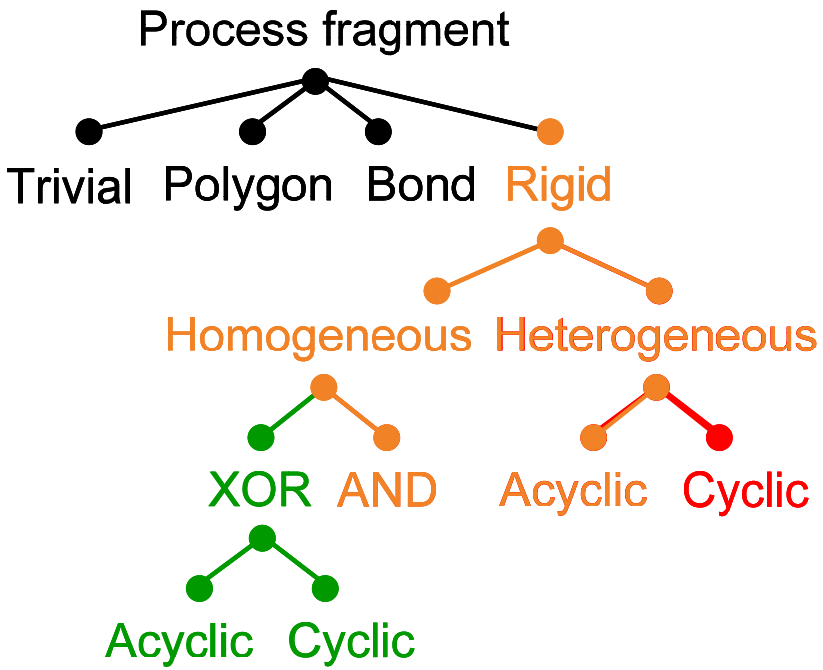




An acyclic heterogeneous rigid fragment

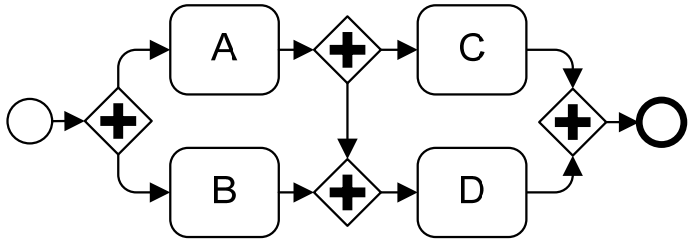
- A *homogeneous* rigid contains either only XOR or only AND gateways
- A *heterogeneous* rigid contains a mixture of AND/XOR gateways
- Rigid fragments are classified as *cyclic* or *acyclic*
- Cyclic homogeneous AND rigids are not considered as they specify *livelocks*

State of the Art: *Structuring Process Models*

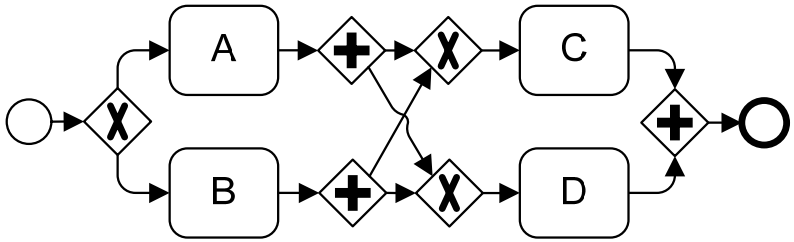


Techniques for structuring flowcharts (**homogeneous XOR rigids**) date back to the late '70s [Oulsnam 1982]

There exist **arbitrary** process models that do not have (FCB-)**equivalent** process models [Kiepuszewski et al. 2000]



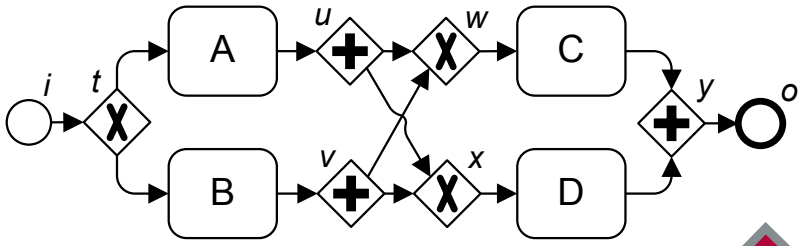
A Z-structure



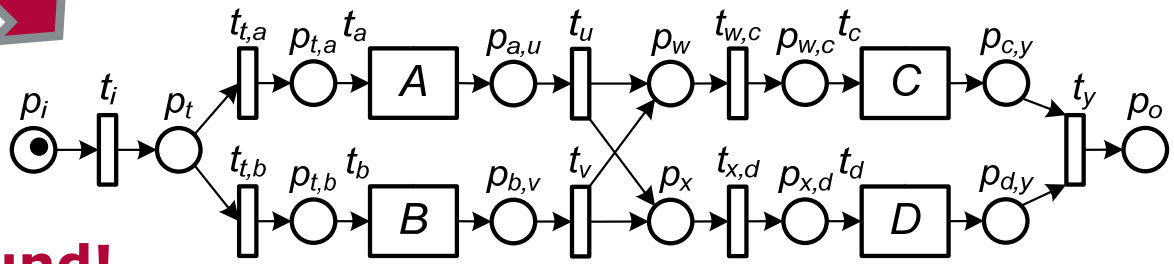
An overlapping structure

Overlapping structures (a special type of **acyclic heterogeneous rigids**) have an (FCB-)**equivalent** process models [Liu and Kumar 2005]

Semantics

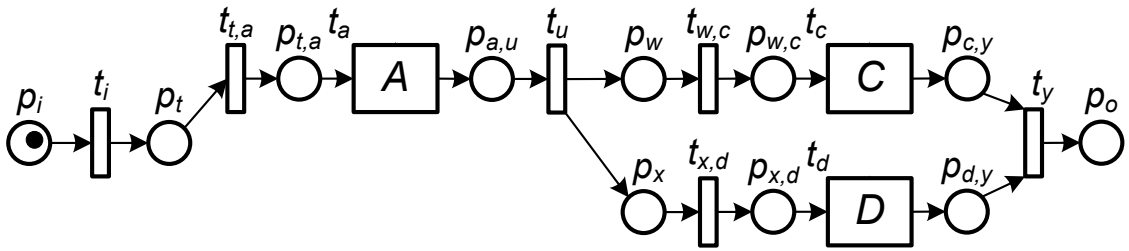


Execution semantics of process models is defined by means of a mapping to labeled (free-choice) **workflow** systems

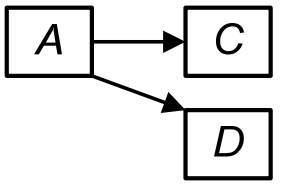


Require process models to be correct, i.e., corresponding workflow systems must be **sound!**

- The behavior of a concurrent system can be described by a partial order (for Petri nets by [Nielsen, Plotkin, and Winskel 1980])

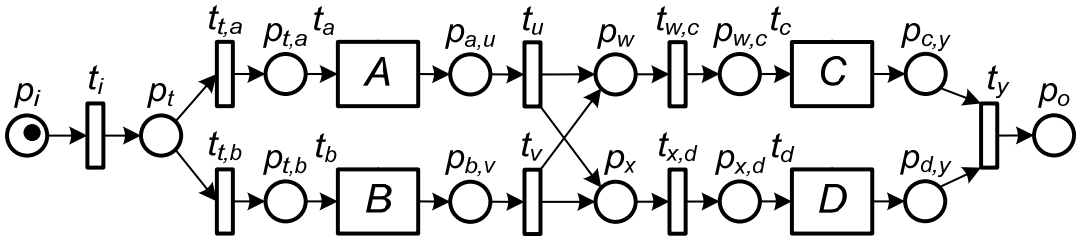


A concurrent run of a system



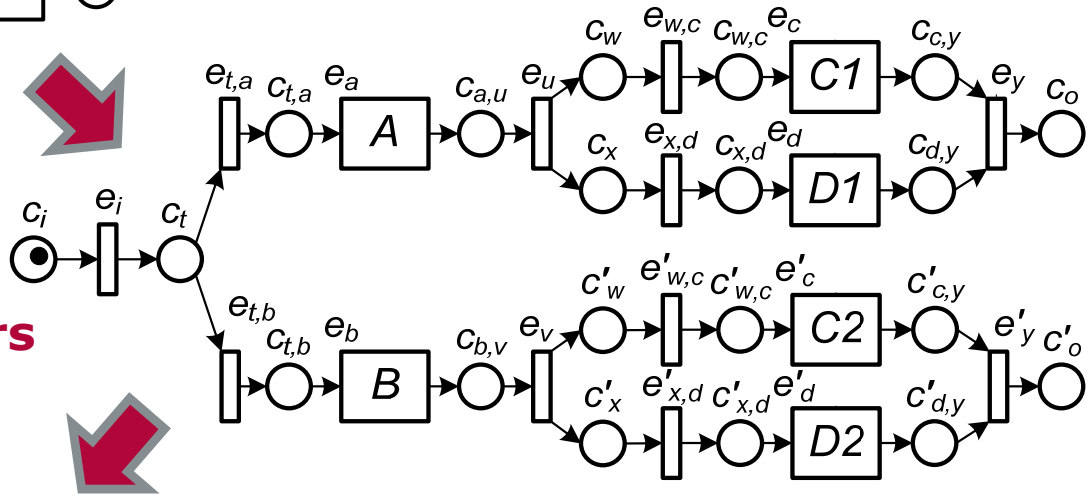
Abstraction of a run

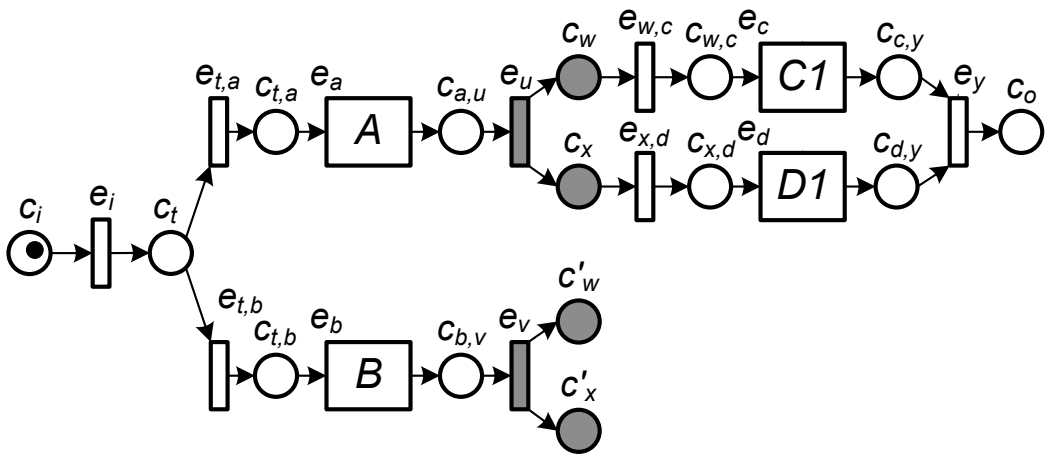
Complete Prefix Unfolding



The technique of **unfolding** is based on a partial order semantics of Petri nets

An unfolding is a "compact" representation of **all runs** of a system together with **all points where a choice occurs** between qualitatively different behaviors of a system



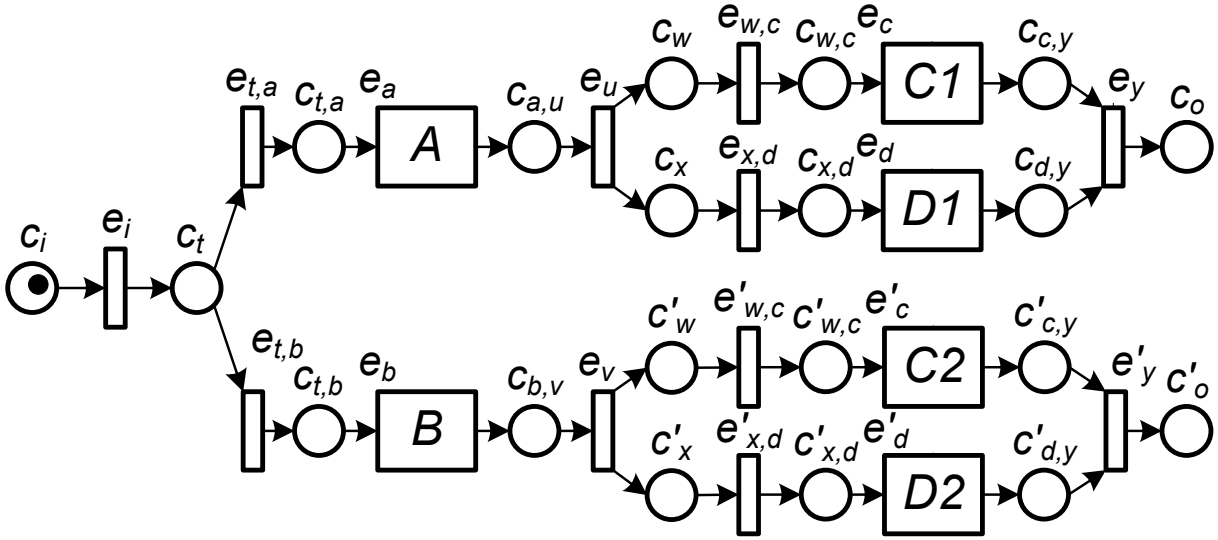


A **complete prefix unfolding** is a finite initial part of the unfolding that contains **full information** about the reachable states of a system [McMillan 1995]

Ordering Relations

The structure of an unfolding is given by a net with a "**simple**" structure – an **occurrence net**

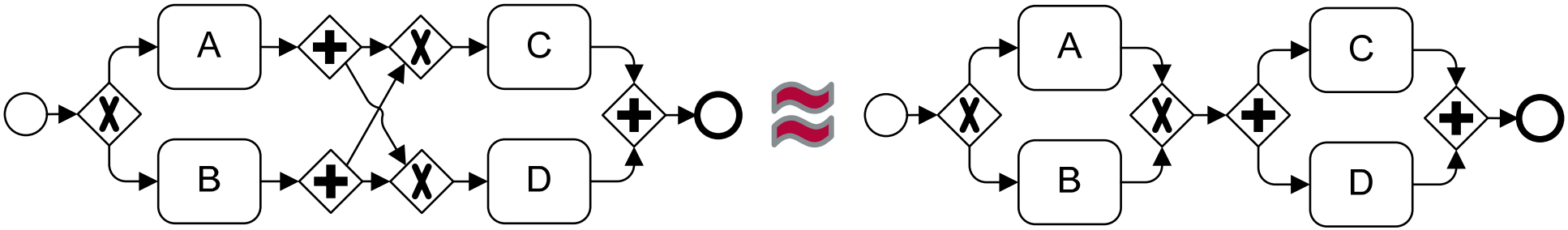
A>C1 **D2||C2** **A#D2** **B#A** **C2||D2** **B>D2**



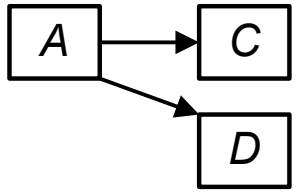
- Two transitions of an occurrence net are in one of the following relations:
  - **A** and **B** are in ***causal*** relation (A>B), if and only if there exists a path from **A** to **B**
  - **A** and **B** are in ***conflict*** (A#B), if and only if there exist two distinct transitions **t1**, **t2** that share an input place and there exist a path from **t1** to **A** and a path from **t2** to **B**
  - **A** and **B** are in ***concurrency*** (A||B) relation, if and only if **A** and **B** are neither in causal, nor in conflict relation

Behavioral Equivalence: *Fully Concurrent Bisimulation*

Two process models are FCB-equivalent …



… if and only if, for each abstraction of a run in one system there exists an isomorphic abstraction of a run in another system, and vice versa



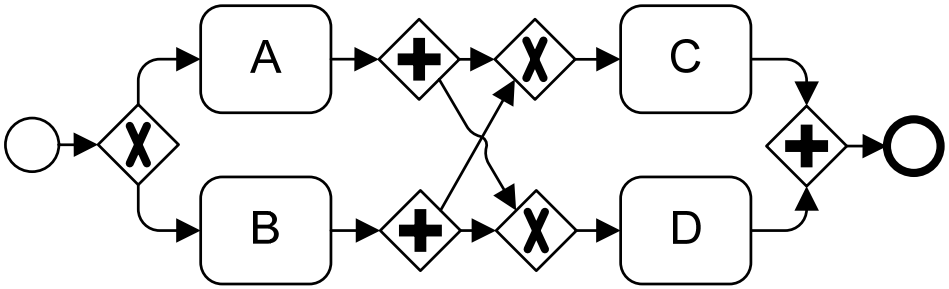
… if and only if, (complete prefix) unfoldings of both models expose same ordering relations

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

≈

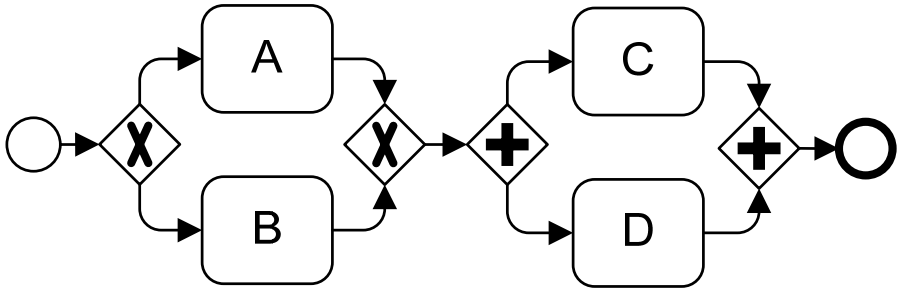
	A	B	C	D
A		#	>	>
B	#		>	>
C	<	<		‖
D	<	<	‖	

Structuring Process Models: *Core Idea*



Compute **ordering relations**
of a graph-structured
process model

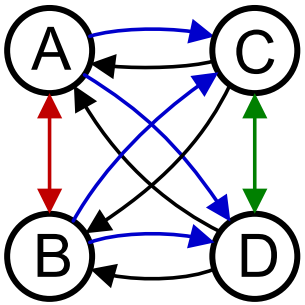
	A	B	C	D
A		#	>	>
B	#		>	>
C	<	<		‖
D	<	<	‖	



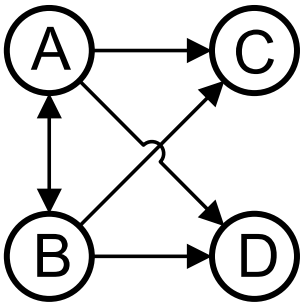
Construct a **block-structured**
process model from ordering relations

Structuring Process Models: *Ordering Relations Graph*

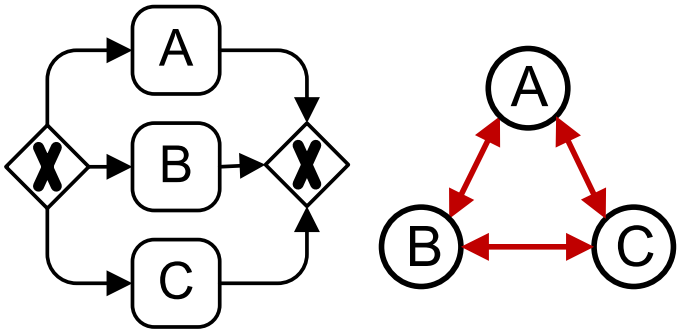
	A	B	C	D
A		#	>	>
B	#		>	>
C	<	<		‖
D	<	<	‖	



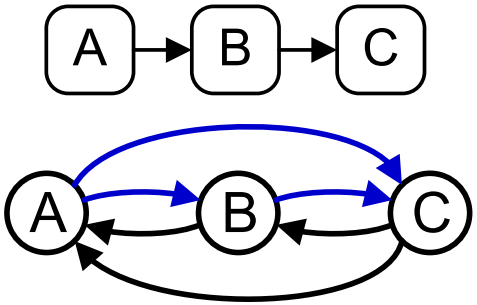
A 2-structure



An ordering relations graph



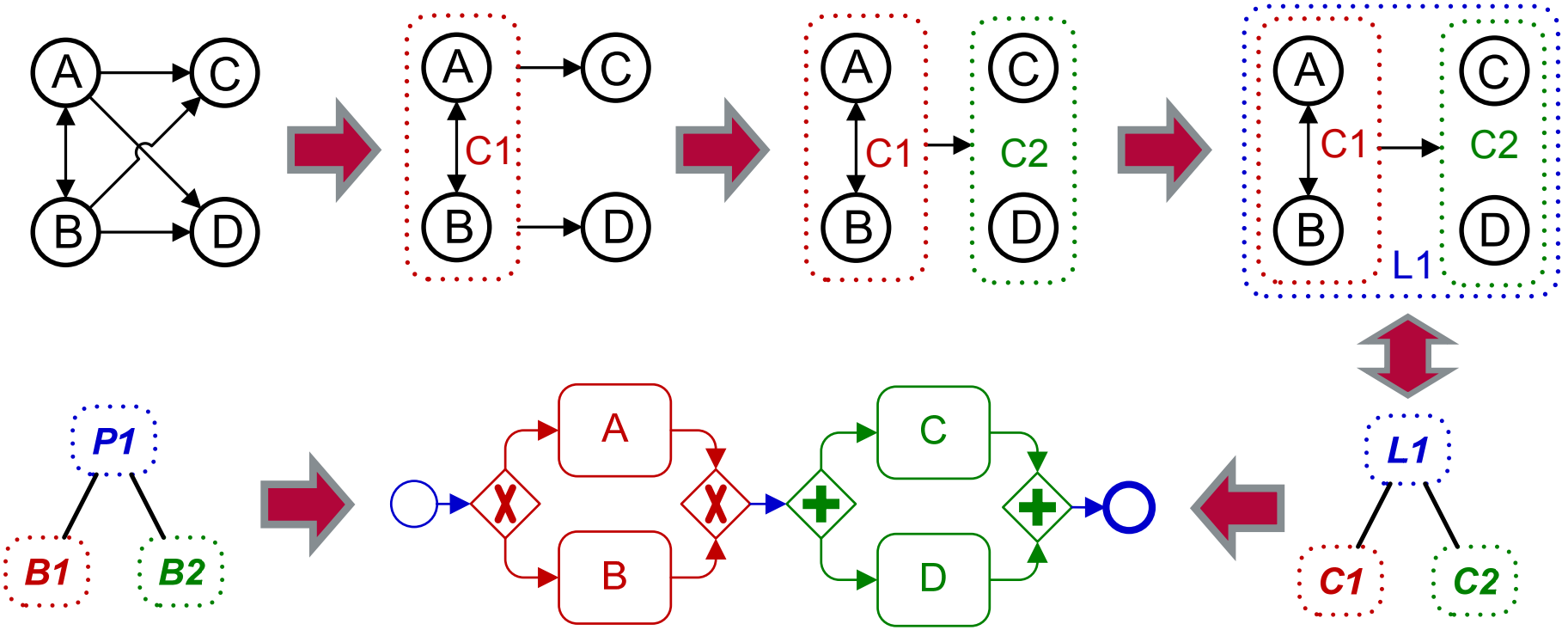
A complete graph of relations of the same type can be represented as a block



A total order of causal relations can be represented as a sequence

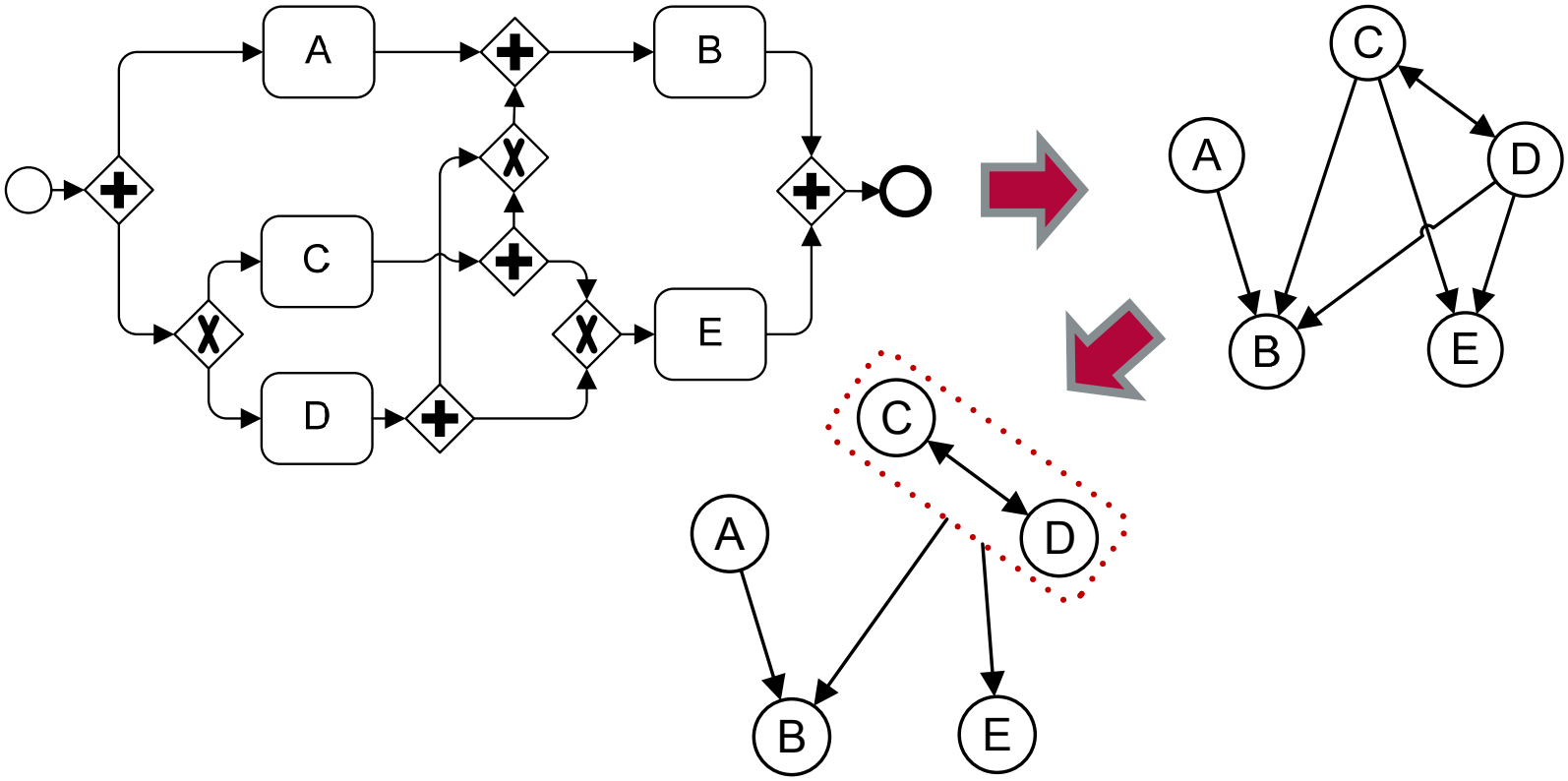
Structuring Process Models: *The Modular Decomposition Tree (MDT)*

- A module is a set of nodes with uniform relations to all neighbors:
  - A *trivial* (T) module is a singleton node of a graph
  - 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*



Structuring Process Models

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



A primitive module

Conclusion

- We proposed a technique for structuring acyclic process models
  - Under an FCB-equivalence notion
  - Block-structured models do not introduce variables to encode control-flow constraints
  - Task duplication depends on the "quality" of the complete prefix unfolding
  - Homogeneous AND rigids can be structured in linear time
  - In general, the complexity of structuring algorithm is determined by the exponential complexity of the unfolding
- Future work
  - Structuring cyclic process models
  - Introduction of variables to encode control-flow constraints
  - Partial structuring of process models
- An early prototype is available at: http://code.google.com/p/bpstruct/