



Hasso  
Plattner  
Institut

IT Systems Engineering | Universität Potsdam

## On Application of Structural Decomposition for Process Model Abstraction

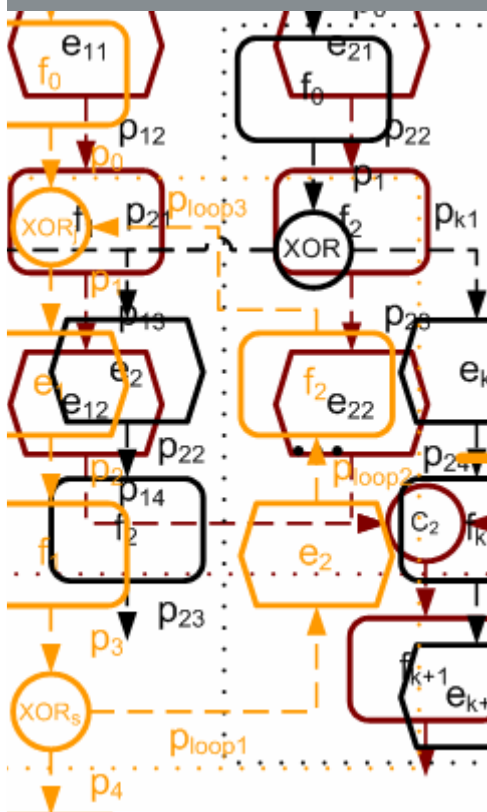
Artem Polyvyanyy

**Sergey Smirnov**

Mathias Weske

**BPSC 2009**

24 March 2009

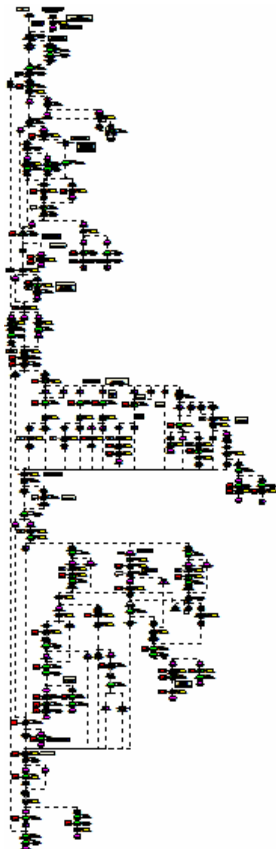


# Motivation

2

Research project with AOK Brandenburg

**Goal:** detailed process models → abstract process models



- $\approx 4\,000$  EPCs
- graph-structured process models
- \* example model:
  - $> 300$  nodes
  - $> 150$  functions
  - graph-structured model

... is generalization of a model, leaving out insignificant process details in order to reduce model complexity and retain information relevant for a particular purpose.

## What

model elements are insignificant?

- *out of scope*
- *possible criteria:*
  - *non-functional properties*
  - *semantics*

## How

to abstract insignificant elements?

- SESE decomposition of a model
- abstraction mechanism

# Process Model

5

$(N, E, type)$  is a business process model, where:

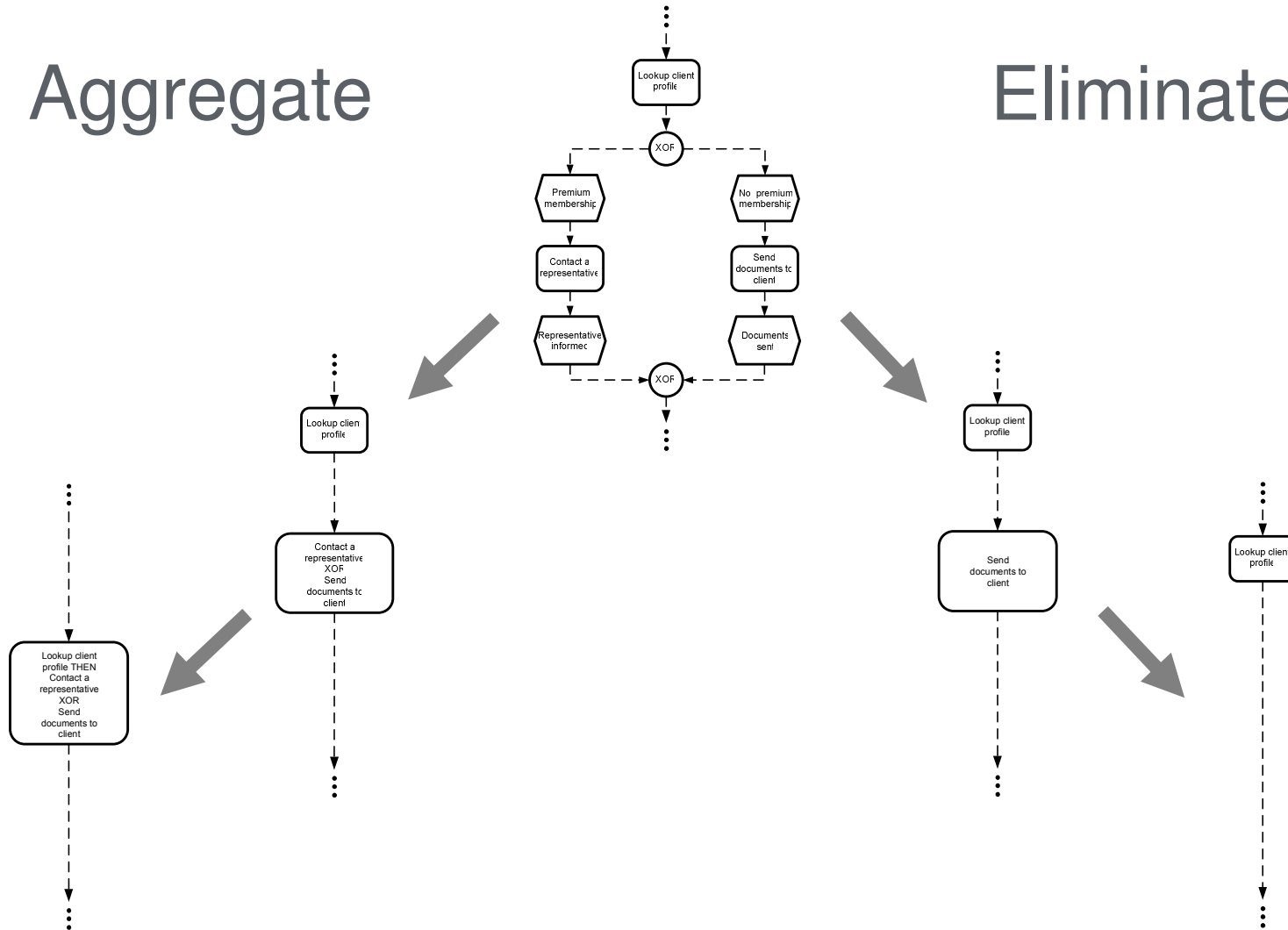
- $N = N_A \cup N_G$  is a set of nodes, where  $N_A \neq \emptyset$  – a set of activities;  $N_G$  – a set of gateways; the sets are disjoint
- $E \subseteq N \times N$  is a set of directed edges between nodes representing control flow
- $(N, E)$  is a connected graph
- every activity has at most 1 incoming & at most 1 outgoing edge
- there is at least 1 activity with no incoming edges (start activity) and at least 1 activity with no outgoing edges (end activity)
- $type : N_G \rightarrow \{and, or, xor\}$  assigns control flow construct to a gateway
- every gateway is either a split or a join; splits have exactly 1 incoming edge and at least 2 outgoing; joins have at least 2 incoming edges and exactly 1 outgoing.

# Aggregation vs. Elimination

6

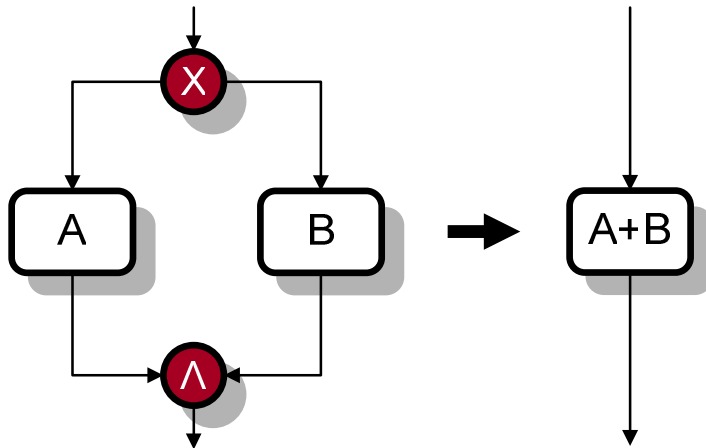
## Aggregate

## Eliminate



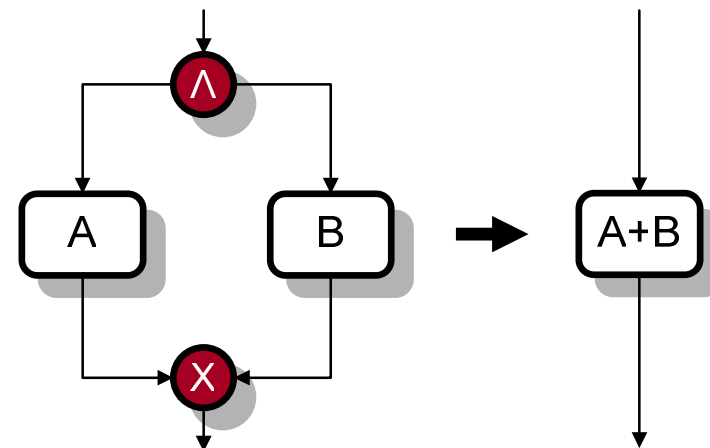
# Assumption: Sound Process Models

7



Hidden  
deadlock

**unsound model**



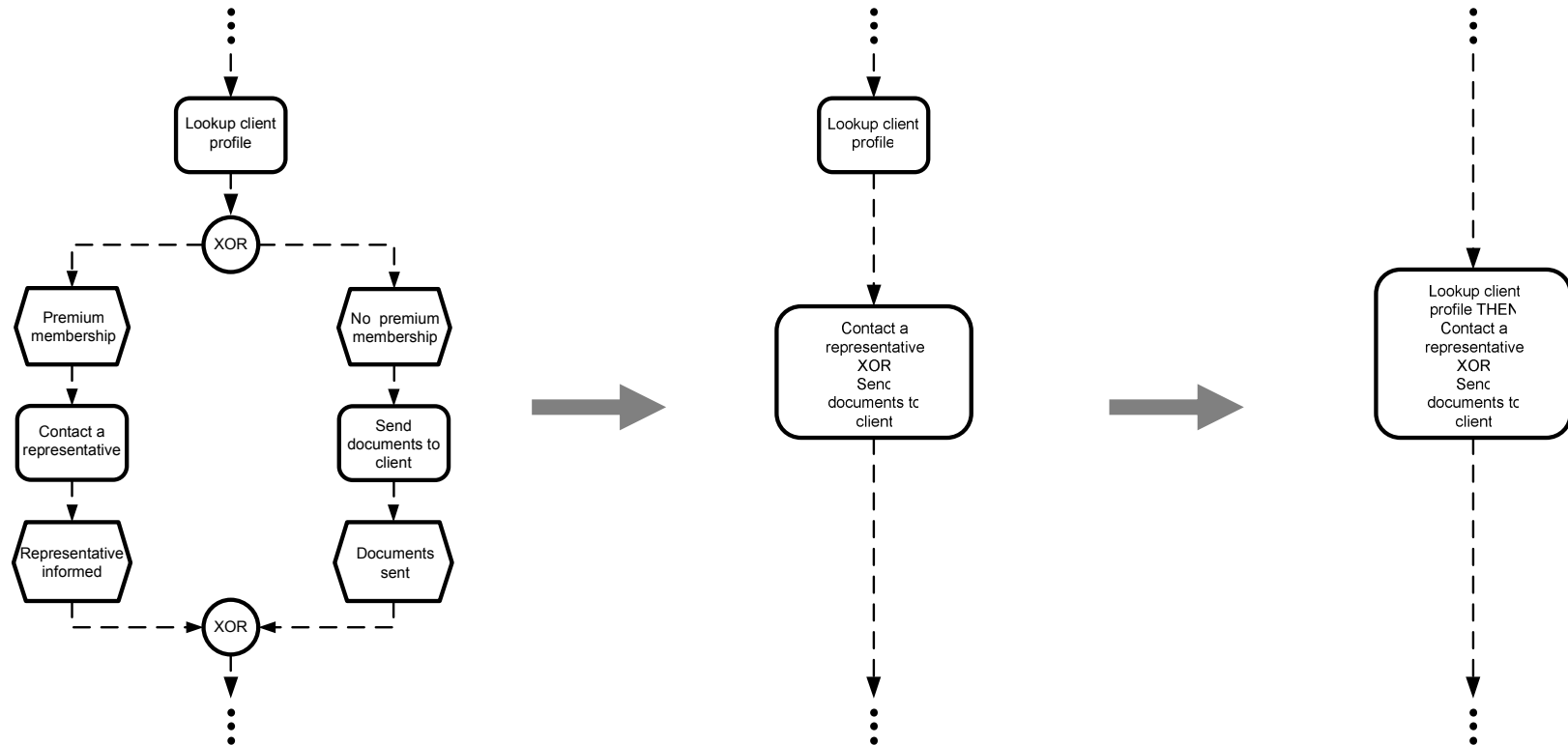
Hidden unsafe  
process fragment

**sound model**

**Assume initial models to be sound**

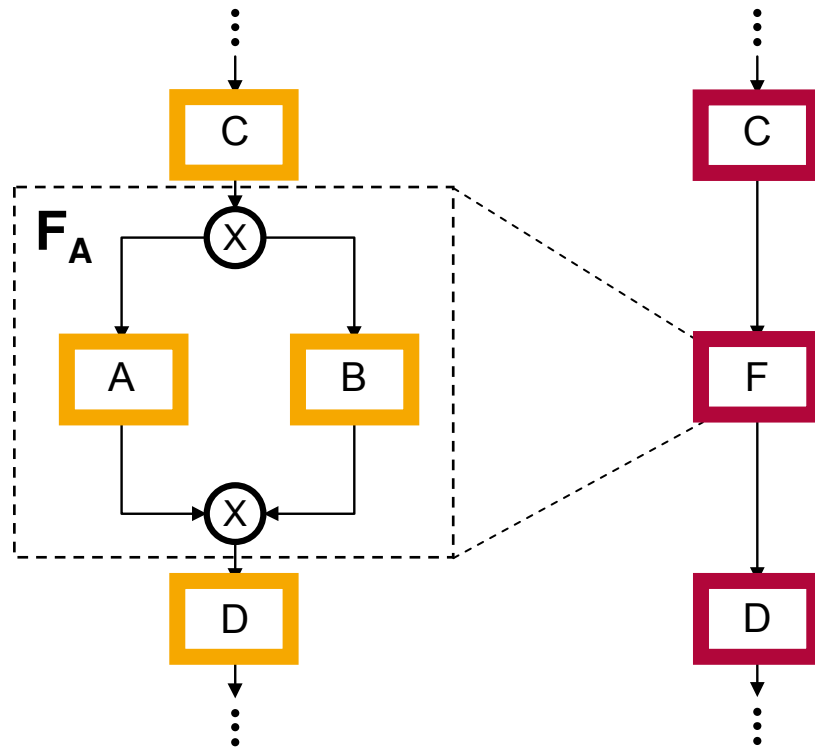
# Stepwise Abstraction

8



# Order Preserving Abstraction

9



**A** and **B** belong to  $F_A$ ,  
ordering constraints are lost

**C** and **D** do not belong to  $F_A$ ,  
ordering constraints are preserved

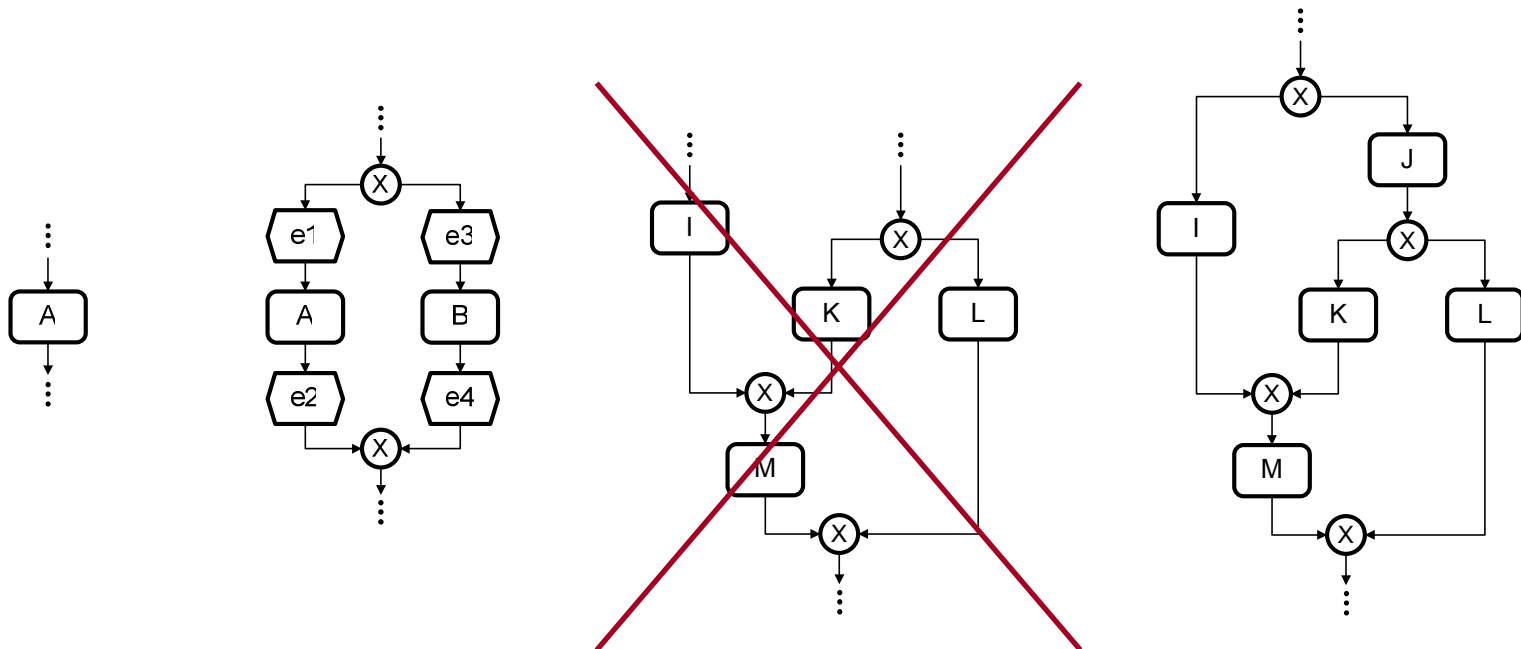
**A** belong to  $F_A$  and **D** does not,  
ordering constraints between  
**F** and **D** as between **A** and **D**

# Single Entry Single Exit Fragment

10

SESE fragment is a fragment which has exactly:

- **1** incoming edge
- **1** outgoing edge

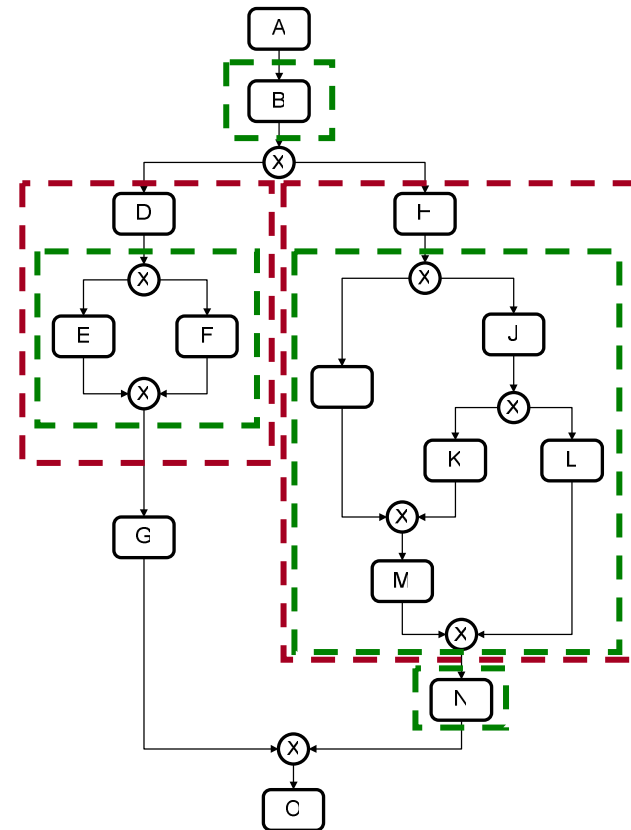


# Canonical SESE Fragment

11



canonical SESE fragments  
non-canonical SESE fragments

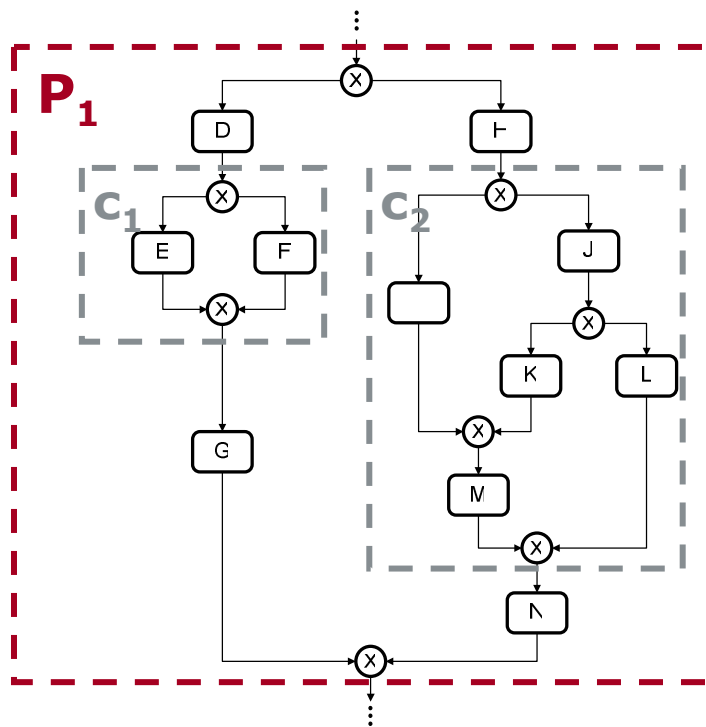


# Relations between SESE Fragments

12

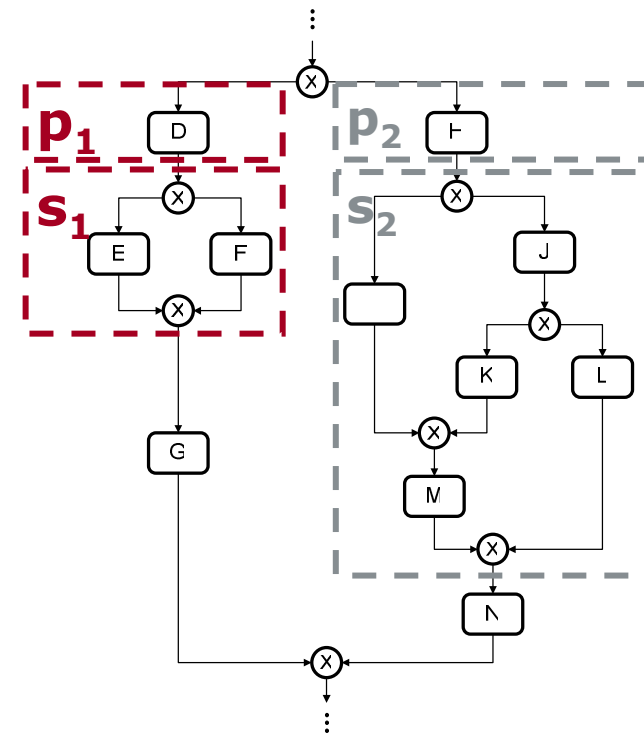
## parent-child

if the node set of SESE fragment  $f_1$  is the subset of node set of SESE fragment  $f_2$ , then  $f_1$  is the child of  $f_2$  and  $f_2$  is the parent of  $f_1$



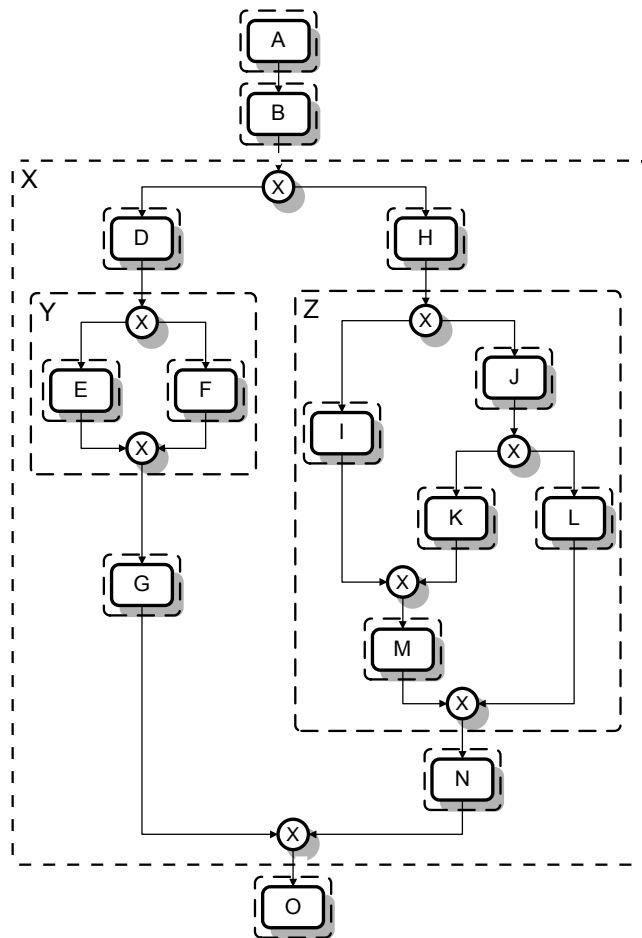
## predecessor-successor

SESE fragment  $f_1$  precedes SESE fragment  $f_2$  (and  $f_2$  succeeds  $f_1$ ) if the outgoing edge of  $f_1$  is the incoming edge of  $f_2$

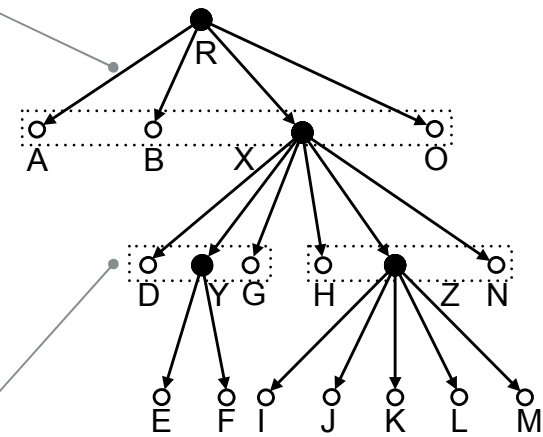


# Process Structure Tree

13



parent-child



predecessor-successor

# Auxiliary Concepts

14

$A$  – an activity to be abstracted

$sese_A$  – canonical SESE fragment containing  $A$  (is a leaf in the PST)

$sese_{min}$  – a minimal canonical SESE fragment containing  $A$  and at least one more activity ( $sese_{min} \neq sese_A$ ); there are 2 options for  $sese_{min}$ :

1. there is canonical sese fragment  $sese_{A'}$ , which is in predecessor-successor relation with  $sese_A$ ; then  $sese_{min}$  is a SESE fragment with the incoming edge of the predecessor and the outgoing edge of the successor
2. if 1 does not hold, than  $sese_{min}$  is a SESE fragment which is the parent of  $sese_A$

# Abstraction Algorithm

15

1. define the set of activities to be abstracted (let it be  $I_A$ );
2. if  $I_A$  has elements, select one activity from the set (let it be  $A$ ); else go to 8;
3. find  $sese_{min}$  for  $A$ ;
4. remove from  $I_A$  all the activities which belong to  $sese_{min}$ ;
5. replace  $sese_{min}$  with aggregating activity with the incoming edge of  $sese_{min}$  and the outgoing edge of  $sese_{min}$ ;
6. if necessary, add the new aggregating activity to  $I_A$ ;
7. go to 2;
8. stop.



# Smoothness Evaluation (I)

17

Experiment with process models:

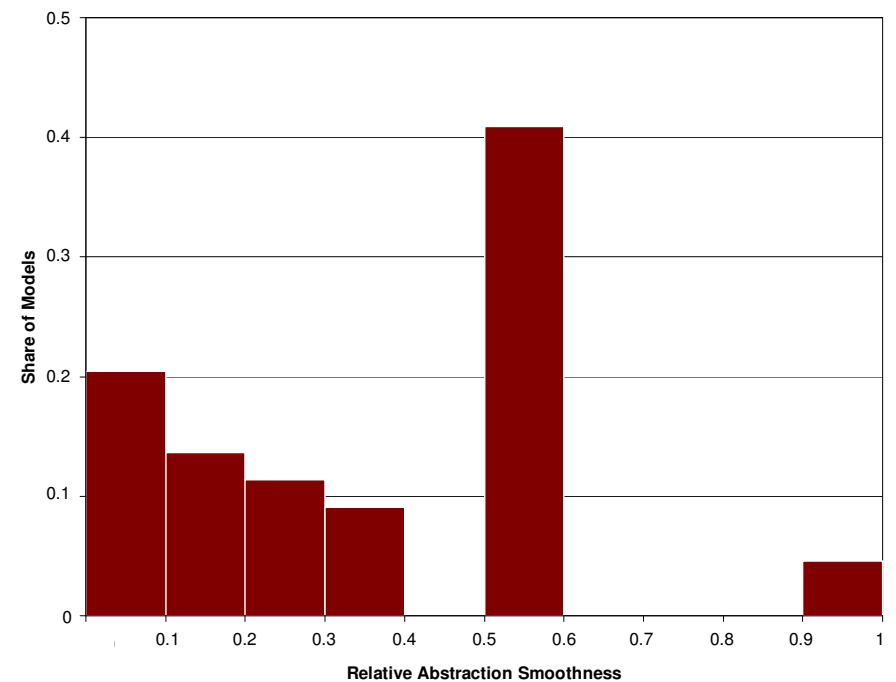
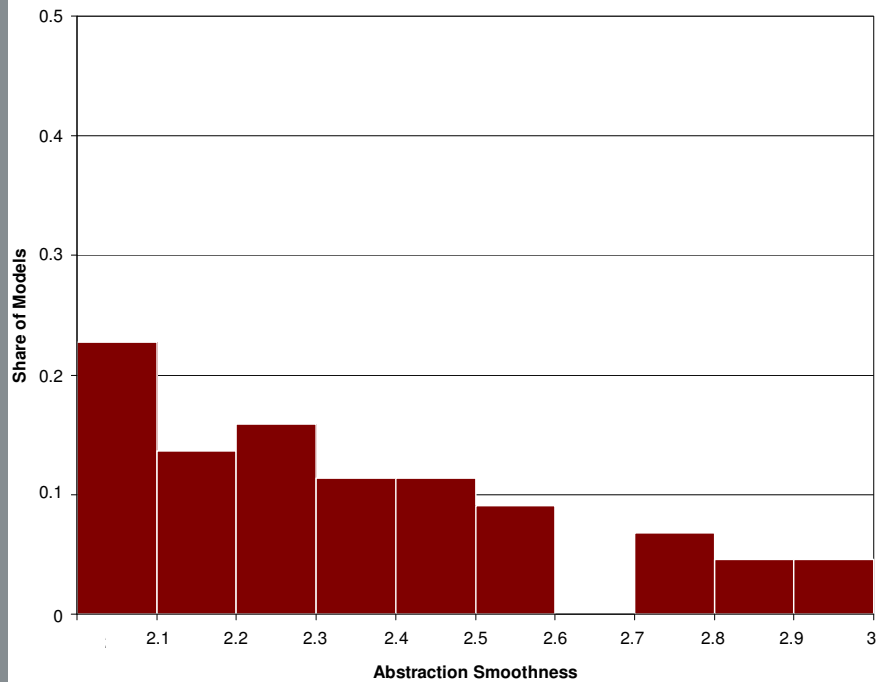
- 50 models
- real world process models
- $50 < |N| < 205$
- graph-structured models

# Smoothness Evaluation (II)

18

## „Optimistic“ algorithm

## „Pessimistic“ algorithm



# Conclusions

19

We proposed the structural abstraction approach based on PST, which is:

- order preserving
- handles graph-structured models

We evaluated the approach regarding smoothness

## What

model elements are insignificant?

- semantics of model elements

## How

to abstract insignificant elements?

- more fine-grained decomposition methods
- prototypical implementation