Master Thesis Report

Process Enhancement by Incorporating Negative Instances in Model Repair

Dfg Global Dependency Discovery

Kefang Ding

RWTH AACHEN UNIVERSITY

Outlines

- **Problem Introduction**
- **Preliminary**
- **Method Design**
- **Method Implementation**
- **Method Evaluation**
- **Appendix**
  - Plugin Development

Problem Introduction

- **Dfg method can not deal with global dependency.**
  - Can’t discover it
  - Can’t remove it
  - Can’t find structure change

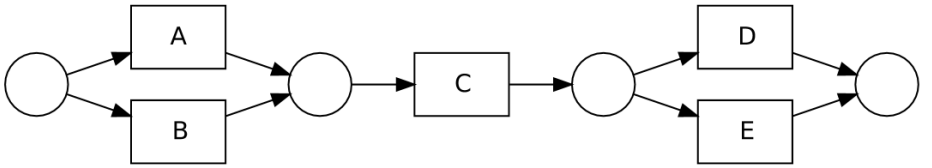
- Given Petrinet discovered by the dfg methods, event log, KPIs, we need to discover the global dependency, remove it or change its structure to the event log.

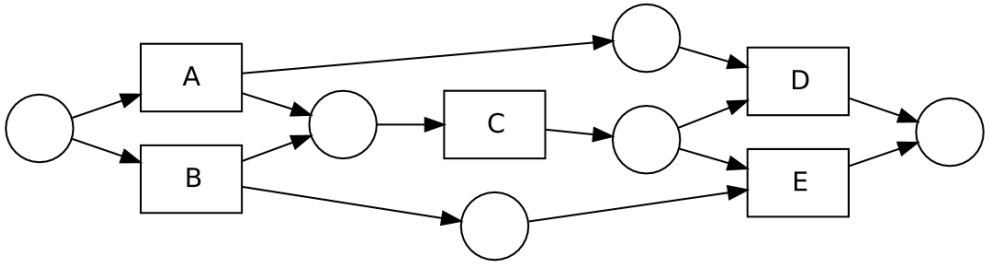
Preliminary

- **Global Dependency Definition:**
  - Relative to local dependency, the decision to execute events depends not only on the events direct before it, but events far away before it.
  - It’s a concept to describe the constraints
- **Non Free Choices Structure Definition:**
  - The choice of events to be executed is constrained by the sequence before events.
  - it’s the structure in Petri net
- **Difference**
  - Global dependency is due to non-free-choice structure.
  - Some non-free-choices don’t have global dependency, but local dependency[can be decided by direct previous events].
- **Characters of Global Dependency**
  - Xor structures exists and are not directly connected
  - Event log has sequences with specific choices from xor structure

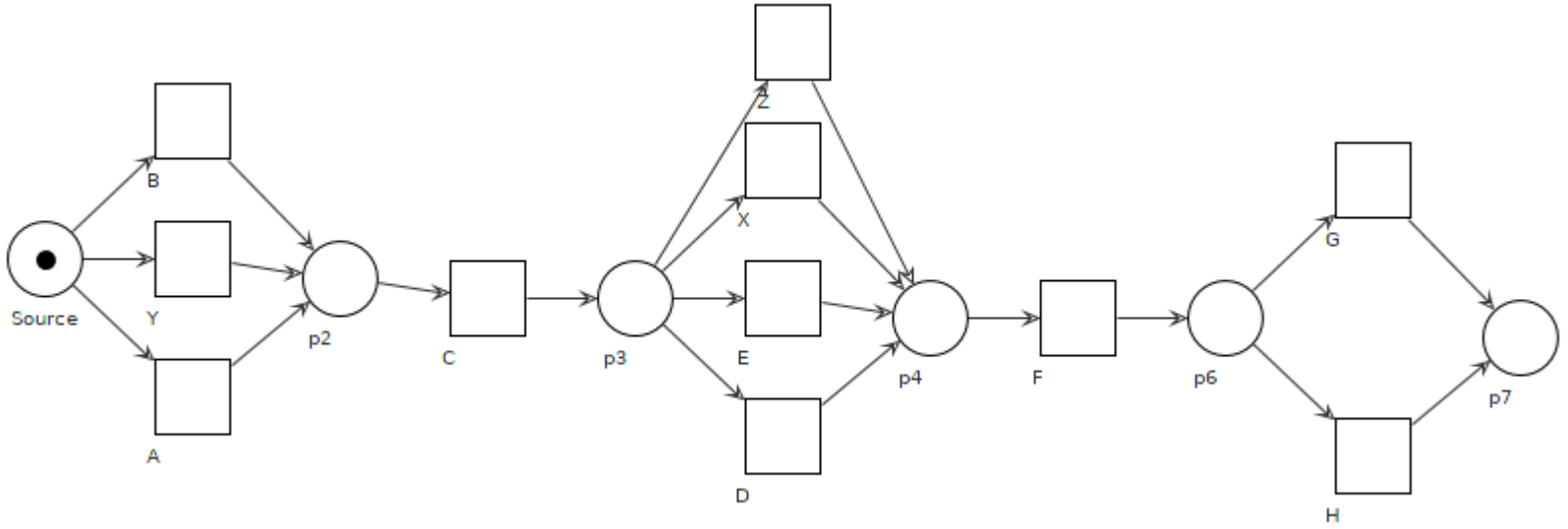
Methods Design

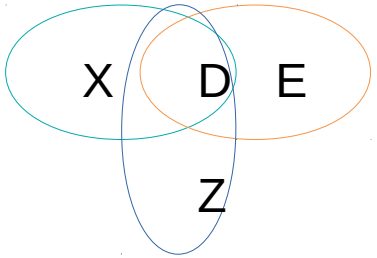
- **Methods Design**
  - Get all possible combinations of xor structure in Petri net, one combination includes:
    - one is xor join to create different precondition
    - one is xor split to decide which events to choose
    - Eg: {A,D}, {A,E}, {B,D},{B,E}
  - Check event log, if they have specific paths through xor structure
    - <ACD, BCE>
      - #{A,D}=true,#{A,E}=false, #{B,D}=false,#{B,E}=true
  - Add places into Petri net to constrain the choices
    - A → o → D , B → o → E

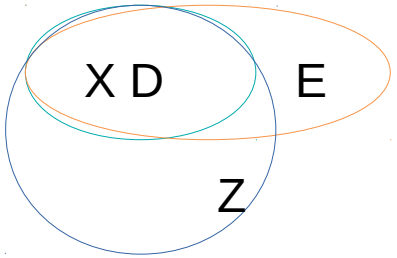


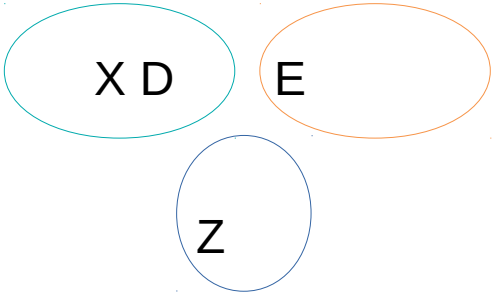


Method Design - {A,B,Y} to {D,E,X,Z}

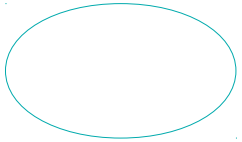




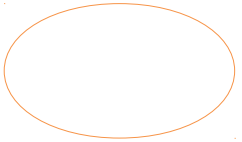




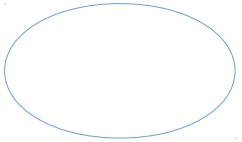
A:



B:



Y:



Method Implementation

- **Places after xor join structure**
  - If any of {A,B,Y} has global dependency, we create post places for each branch, P[A],P[B],P[Y]
- **Places before xor split structure**
  - If any of {X,D,E,Z} has global dependency with xor join before them, create a pre place for each of them
- **Silent transition to connect places**
  - If there is connection of any combination from {A,B,Y} *{X,D,E,Z}, we create a silent transition
  - Connect the silent transition and places

    [postplace] → tau → [preplace]