Josep Carmona

Collaborators:
J. Cortadella (LSI), J. Muñoz (LSI), M. Kishinevsky (Intel), M. Solé (AC)
Process Mining: motivation

- Logs are easy to obtain.
- Traditional knowledge discovery techniques offer limited support to build formal models:
  - Classification, clustering, regression, association-rule learning.
- Obtain a formal explanation of what is going on on a system by observing its logs:
  - Petri nets can express most of the common situations within a process. It has well-defined semantics, and formal techniques can be applied to verify the system.
- Applications: business intelligence, healthcare, EDA, ...
PROCESS MINING OVERVIEW
[source: www.processmining.org]
### Event Log

<table>
<thead>
<tr>
<th>Case</th>
<th>Event</th>
<th>Timestamp</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>reservation</td>
<td>21-02-2009 12:20h</td>
</tr>
<tr>
<td>1</td>
<td>arrival</td>
<td>22-02-2009 21:05h</td>
</tr>
<tr>
<td>2</td>
<td>reservation</td>
<td>23-02-2009 14:00h</td>
</tr>
<tr>
<td>1</td>
<td>payment</td>
<td>23-02-2009 14:50h</td>
</tr>
<tr>
<td>2</td>
<td>cancellation</td>
<td>23-02-2009 16:00h</td>
</tr>
</tbody>
</table>

### Petri Net (PN)

### Transition System (TS)
PRELIMINARIES:
LOGS, TS, PETRI NETS
Transition systems and Petri Nets

![Diagram of transition systems and Petri Nets]
Enabling and Firing

- Transition is enabled if all its input places have tokens (this is AND causality)
- Example: transitions labeled with events a, b, and c are enabled in the initial marking
- An enabled transition can fire
- Firing decrement tokens in input places and increment tokens in output places (firing is atomic)
- In general firing of one transition may prevent firing of another one (not in this example)
- a, b, c can fire in any order without disabling each other. Hence a, b, c are concurrent
Enabling and Firing

a fired
b and c are still enabled
Enabling and Firing

c fired some time later
b is enabled
d is not enabled yet
Enabling and Firing

- b fired
- d is enabled
Enabling and Firing

d fired

e is enabled
Enabling and Firing

e fired
reached the initial marking
Transition systems and Petri Nets

This is the trace (firing sequence) that we examined
Petri Net (PN) Discovery

LOG
abacad badda aabcca ....

TS

PN

a
b

c

a

b

a

b

c

d
From Logs to TSs

LOG
aab
aba

SEQUENCE

PRE

POST

SET
THEORY OF REGIONS
A set of states is a region if every event has an homogeneous relation with the region.

- \( S = \{s_1, s_4, s_5, s_7\} \)
  - \( a \) enters \( S \)
  - \( d \) exits \( S \)
  - \( b \) and \( e \) are inside or outside of \( S \)

- Regions correspond to PN places.
For every event generate a transition labeled with this event.
PN synthesis algorithm

For every (minimal) region generate a place
If region includes the initial state then mark the corresponding place
PN synthesis algorithm

If event exits the region add an arc between the corresponding place and the corresponding transition
PN synthesis algorithm

If event enters the region add and arc between the corresponding transition and the corresponding place.
PN synthesis algorithm

- Polynomial complexity with respect to TS (Badouel’95).
- *Isomorphism* with respect to the initial TS.
- Only particular types of TSs can be synthesized.
- *Label splitting* can be used for synthesis.
The *excitation set* of event $e$ contains all those events where $e$ is enabled.

- $\text{ES}(a) = \{s_0, s_2, s_3, s_6\}$ (region)
- $\text{ES}(d) = \{s_7\}$ (not a region)
Relaxing Synthesis: Excitation closure

- Condition for **bisimulation** (*Cortadella et al. ’98*)
- \( ^o e = \) all regions that \( e \) exits (pre-regions(\( e \)))
- ECTS:
  - For every event \( e \):

\[
ES\ (e) = \bigcup_{r \in \text{o}e} r
\]
Excitation closure

Not EC

EC

ER(e)
Excitation closure

R_1 \quad R_2

ER(e)

R_3

Not EC

R_1 \quad R_2

ER(e)

R_3

EC

R_3 not needed for EC
Extension: General PNs

- k-bounded places, weighted arcs, cleaner and more concise Petri nets

Diagram:
- 2-bounded PN
- Safe PN
General PNs

TS

Safe PN

3-bounded PN

General PNs
General regions

- Idea: every place of a PN corresponds to a \textbf{multiset} of states in TS \cite{Mukund}

\begin{itemize}
  \item Ex.: multiset \( r=\{s_0^2,s_1^1\} \) (or in different notation, \( r(s_0)=2, r(s_1)=1, r(s_2)=0 \)):
\end{itemize}
Gradient in a multiset

Gradient for transition \((s_0, e, s_1)\): \(r(s_1) - r(s_0) = 2 - k\)
Gradient for transition \((s_2, a, s_1)\): \(r(s_1) - r(s_2) = 2 - 0 = 2\)
General regions

- Gradient for transition
  \((s_0, a, s_1): r(s_1) - r(s_0) = -1\)
- Gradient for transition
  \((s_1, a, s_2): r(s_2) - r(s_1) = -1\)
- Gradients for \(r\):
  - \(a\): constant and equal to -1
  - \(b\): constant and equal to -2
  - \(c\): constant and equal to +1

\(r = \{s_0^2, s_1^1\}\) is a region
(all events have constant gradient)
Algorithm for generating regions

\[ P = \text{for all } e \{ ES(e) , SS(e) \} \quad /* \text{seeds of the algorithm} */ \]

while (r = pick new multiset from P)
  if r is not a region
    e = choose event with non constant gradient
    select gradient \( g = (g_{\text{min}} + g_{\text{max}})/2 \)
    \( P = P \cup \text{(expand } r \text{ for gradient}(e) \leq g) \)
    \( P = P \cup \text{(expand } r \text{ for gradient}(e) > g) \)
Keep minimal regions only in P

Theorem: The algorithm generates all k-bounded minimal regions.
PN Mining goals

- **Language containment** with respect to the initial TS.
- There is always a solution, no need for label splitting.
- The mined net should be a good visualization of the log.

LOG

```
aabc
abac
baac
```

TS

Synthesized Petri net

Mined Petri net.

Trace `aaaaabc` accepted!
PN Mining properties

Definition

Let TS = (S, E, A, s_0) a transition system, and k a bound. The PN = (P, T, F, M_0) mined from TS satisfies:

1. L(TS) ⊆ L(PN)
2. T = E, i.e., there is no label splitting
3. Minimal language containment (MLC) property:

∀ PN' = (P', T', F', M_0), T' = E : L(TS) ⊆ L(PN') ⇒ L(PN) ⊆ L(PN')
PN Mining properties

LOG
abc
bac

(a* || b* || c*)
NOT MLC

PN 1

PN 2
(a* || b*)c
NOT MLC

PN 3
(a || b)c
MLC
Theorem
Let PN = (P, T, F, M₀) be the mined net with the set of minimal pre-regions of TS = (S, E, A, sᵢₐ). PN satisfies 1, 2 and 3 from the previous definition.
RESEARCH LINES
Region-based app: limitations

- Large TSs cannot be handled
- The use of efficient data structures only alleviates:
  - Implicit representation (Binary Decision Diagrams)
  - Dynamic Programming
  - Exploration heuristics
Strategies to overcome these limitations:

1) **Decompositional approach**: search for a set of *sequential* views of the log that jointly explain it

2) **Divide-and-Conquer approach**: iteratively project the log into smaller logs that can be handled afterwards

3) **Incremental approach**: derive a set of *bases* that individually generate part of the behavior and can be *joined* into one base
Decompositional Strategy

- Sequential views of the log ⇔ State Machines (SM)

Not an SM: a,b,c concurrent

SM
Decompositional Strategy

Theorem: a **partition** of a TS by regions is a SM
Decompositional Strategy

Theorem: a partition of a TS by regions is a SM
Decompositional Strategy

em, s not covered
Decompositional Strategy
Decompositional Strategy
Decompositional Strategy

General Algorithm:

while (there is still event ev uncovered) do
    find SMi that covers ev;
    i = i + 1;
    update covering table
endwhile

Theorem: \( L(TS) \subseteq L(SMC_1 \parallel \ldots \parallel SMC_n) \)

FOR SOME BENCHMARKS, FROM MINUTES TO SECONDS!
Divide-and-Conquer Strategy

LOG

a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,w,x,y,z

LOG

a,c,e,f,h,j,l,n,p,r,s,u,w,y

LOG

b,d,e,h,i,k,l,m,n,o,q,t,z

LOG

a,c,e,f,h,j,l,y

LOG

n,p,r,s,u,w,y

LOG

b,m,n,o,q,t,z

LOG

b,d,e,h,i,k,l

PN1,PN2

PN3

PN4,PN5

PN6
Divide-and-Conquer Strategy

event relations

\[ a \rightarrow b \]
\[ b \parallel c \]
\[ a \rightarrow d \]

Causal Dependencies Graph (CDG)

log-based ordering, triggering, etc...

edge between x and y if x is related with y

spectral graph partition

LOG

LOG

LOG
Divide-and-Conquer Strategy

General Algorithm:
compute CDG(TS);
(E₁, … , Eₙ) = FindSpectralPartition(CDG(TS), MaxSize);
forall Eᵢ do
  Eᵢ = Eᵢ + AddBorderEvents(CDG(TS), Eᵢ);
  SMDecomposition(TS|Eᵢ)
endwhile

Theorem: \( L(TS) \subseteq L(SMC₁ || … || SMCₙ) \)

LARGEST LOGS CAN ONLY BE HANDLED WITH D&C & C
Tool support: *Genet*

- **Features:**
  - Synthesis/mining of general Petri nets
  - Decompositional/Divide-and-Conquer approaches
  - GenetGUI: graphical user interface (by J. Muñoz)

- [http://www.lsi.upc.edu/~jcarmona/genet.html](http://www.lsi.upc.edu/~jcarmona/genet.html)
  - Binaries for Linux/Sun
  - Related papers
  - Tutorial.
Incremental Mining (with M. Solé)

- Theory based on linear algebra and compatibility relations
- Forthcoming tool: one/two orders of magnitude reduction for the first experiments!
Degree of accuracy of the mined net?

Tool: Plugging within ProM
Hot topics

- Regions and Noise?
  - Approximated theory of regions.
- Unbounded Petri nets (prod-cons-like systems)?
  - How to find buffers connecting subsystems?
- Hierarchical models?
  - Places can be hiding a complex behavior.
- Applications: CAD/EDA, Verification, Web Services, ...
Conclusions

- A good opportunity for bridging the gap between current system design tools and formal methods.
- Several companies, universities, associations involved (e.g., new IEEE Task Force in Process Mining)
- Tools: ProM (OpenSource), Pallas Athena.
Choice and conflict

Shared resource problem: two processes cannot be in the critical section together
States, events, transitions

- Global states of a system
- Transition
- Finite set of event labels
Transition system (TS)

- initial state
- The same event may label many different transitions (marked with the same color here)
- TS are state based models of behavior
Petri Nets (PN)

- Models of concurrent behavior
- Local states and events instead of global states (unlike TS)
- Marking vector defines current state
- Used for modeling protocols, distributed algorithms, asynchronous systems
- Mathematically PN is a vector addition system, but graphical representation is useful in most applications
Structure of PN

- bipartite graph with nodes of two types
- “places” – conditions, resources or local states
- “transitions” – changes in local states
- “events” are labels on transitions (since many transitions may correspond to the same system event)
- arcs – flow relation between places and transitions
Minimal pre-regions

- **Pre-regions**: regions containing the excitation set of some event.

R3 not minimal !!

![Diagram showing regions R1, R2, and R3 with arrows indicating some event ES(a)]