Inducing Declarative Logic-Based Models from Labeled Traces

Evelina Lamma[1] Paola Mello[2] Marco Montali[2]
Fabrizio Riguzzi[1] Sergio Storari[1]

[1]ENDIF – Università di Ferrara, Italy

[2]DEIS – Università di Bologna, Italy

Alessandria, 17/01/2008

Process Management

- Declarative process models: high-level process behavior without expressing how it is procedurally executed
- Languages:
  - Graphical: DecSerFlow, ConDec, CigDec [van der Aalst, Pesic 06]
  - Logical: $\mathcal{S}$CIFF

Process Mining

- Automatically inferring a process model from a log
- Log: set of traces of executions of the process
- Trace: sequence of activities

Mining Declarative Models

- Proposals:
  - ILP07 [Lamma et al. 07]: $\mathcal{S}$CIFF models
  - BPM07 [Lamma et al. 07]: DecSerFlow models
- BPM07 Approach:
  - learn a model in $\mathcal{S}$CIFF
  - translate the model into DecSerFlow

Process Modeling with $\mathcal{S}$CIFF

- Trace=interpretation (set of ground atoms)

$$\{ \quad \mathbf{H}(order(bob, fiat, car), 5), \\ \mathbf{H}(pay(bob, fiat), 7), \\ \mathbf{H}(deliver(fiat, car), 9) \quad \}$$

- Model=$\mathcal{S}$CIFF theory, i.e. set of $\mathcal{S}$CIFF constraints

Restrictions on $\mathcal{S}$CIFF

- For computational reasons, a restricted syntax is used

$$Body \rightarrow DisjE_1 \vee \ldots \vee DisjE_n \vee DisjEN_1 \vee \ldots \vee DisjEN_m$$

- $Body = b_1 \wedge \ldots \wedge b_l$
- $DisjE_i = \mathbf{E}(ev, T) \wedge d_1 \wedge \ldots \wedge d_k$
- $DisjEN_i = \mathbf{EN}(ev, T) \wedge d_1 \wedge \ldots \wedge d_k$

Test of Compliance

- Test of compliance of a trace t with an IC C by means of Prolog
- Let:
  - $DisjE'_i = \mathbf{H}(ev, T) \wedge d_1 \wedge \ldots \wedge d_k$
  - $DisjEN'_i = \mathbf{H}(ev, T) \wedge d_1 \wedge \ldots \wedge d_k$
- Run the query:

$$? - Body, not(DisjE'_1), \ldots not(DisjE'_n), DisjEN'_1, \ldots, DisjEN'_m.$$

  in a database containing the atoms of t as facts
- If the query fails, the trace is compliant
- If the query succeeds, the trace is not compliant

IC Example

$$\mathbf{H}(a(bob), T) \wedge T < 10$$
$$\rightarrow \mathbf{E}(b(alice), T1) \wedge T < T1 \vee$$
$$\mathbf{EN}(c(mary), T1) \wedge T < T1 \wedge T1 < T + 10$$

Compliance Example

$$?-\mathbf{H}(a(bob),T),T<10,$$
$$not(\mathbf{H}(b(alice),T1),T<T1),$$
$$\mathbf{H}(c(mary),T1),T<T1,T1<T+10.$$

- $t_1 = \{\mathbf{H}(a(bob),5)\}$
- $t_2 = \{\mathbf{H}(a(bob),5),\mathbf{H}(c(mary),11)\}$
- $t_3 = \{\mathbf{H}(a(bob),5),\mathbf{H}(b(alice),7),\mathbf{H}(c(mary),11)\}$
- t_1, t_3 compliant, t_2 not compliant.

Clausal Logic

- $\mathcal{S}$CIFF ICs are a generalization of logical clauses:

$$b_1 \wedge \cdots \wedge b_n \rightarrow h_1 \vee \cdots \vee h_m$$

- Mining a clausal theory from interpretations is studied in the learning from interpretation setting of Inductive Logic Programming

Process Mining with $\mathcal{S}$CIFF

Given:

- a space of possible $\mathcal{S}$CIFF theories $\mathcal{H}$ (language bias);
- a set P of positive traces;
- a set N of negative traces;

Find a $\mathcal{S}$CIFF $H \in \mathcal{H}$ such that;

- for all $p \in P$, p is compliant with H;
- for all $n \in N$, n is not compliant with H.

Negative Traces

- Negative traces represent failed or unwanted process executions
- Fundamental to limit the number of constraints found:
  - only discriminative constraints are returned
- Can be generalized to more than two classes

Mining $\mathcal{S}$CIFF theories

- Adapting techniques from learning from interpretations
- Truth of a clause in an interpretation replaced by compliance of a trace to an IC
- Adaptation of θ-subsumption as a generality order for ICs
- Generalization operator:
  - add a literal to the body
  - add a disjunct to the head composed of:
    - a single **EN** atom
    - an **E** atom followed by all the constraints allowed by the language bias for the disjunct
  - add a constraint to a **EN** disjunct
  - remove a constraint from a **E** disjunct

Mining $\mathcal{S}$CIFF theories

- Algorithm inspired to ICL [De Raedt, Van Laer 95]
- Two nested loops: covering and generalization loops
- Covering loop:
  - at each step we add a new IC C
  - negative traces not compliant with C are removed from N
- Generalization loop:
  - heuristic search in the space of ICs ordered by the generality relation

DecMiner

- Learns a $\mathcal{S}$CIFF theory and translates it into DecSerFlow
- Uses the language bias to ensure that the learned ICs can be translated into DecSerFiow
- Language bias: set of templates, one or two for each DecSerFlow constraint
- Template: specifies the form of a single IC
  - literals to add to the body
  - disjunctions to add to the head
  - literals to add to a **EN** disjunct
  - literals to remove from a **E** disjunct

DecMiner

- We implemented a DecMiner plugin for ProM
- The log in MXML is translated into Prolog
- The user selects the (binary) DecSerFlow constraints she is interested in
- The language bias is automatically generated
- DecMiner is run
- The learned ICs are shown to the user, translated into DecSerFlow and saved to an XML file
- The XML file is shown using Declare

Experiments

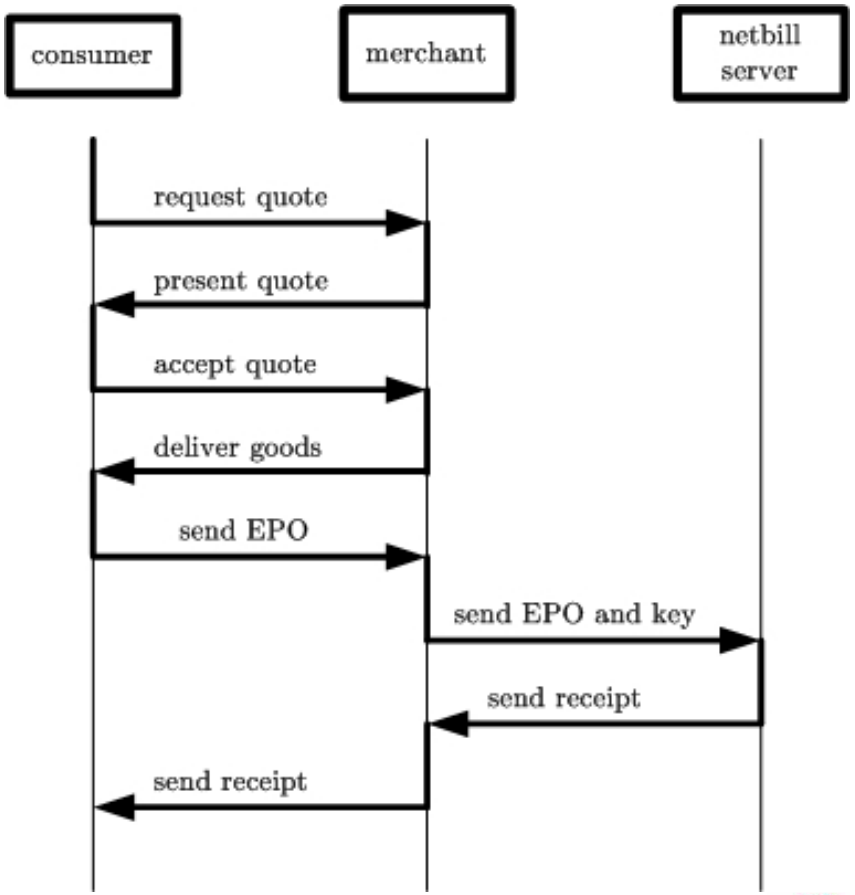
- Aim: test the accuracy
- Datasets:
  - Cervical cancer screening
  - NetBill protocol [Cox et al. 1995]
- Algorithms:
  - declarative: DecMiner
  - non-declarative: α-algorithm [van der Aalst et al. 04] (Petri nets), Multi-Phase Miner [van Dongen, van der Aalst, 04] (instance graphs)

Cervical cancer screening

- Process used by the public health organization of Emilia Romagna
- Screening of the female population in oder to early detect cancer in the uterus cervix
- It is usually composed by five phases: planning, invitation, pap-test, colposcopy, biopsy
- 55 positive traces and 102 negative traces
- Negative traces: incorrect process executions

NetBill

- Security and transaction protocol for selling and delivering low-priced information goods across the Internet
- Datasets generated randomly from a correct model of the process



NetBill Trace Generation

- Negotiation phase: we add to the end of the trace a *request* or *present* message. The length of the negotiation phase is selected randomly between 2 and 5
- Transaction phase 1: either an *accept* or a *refuse* message is added to the trace and the transaction phase is entered with probability 4/5, otherwise the trace is closed
- Transaction phase 2: The messages *deliver*, *epo*, *epo_and_key*, *receipt* and *receipt_client* are added to the trace. With probability 1/4 a message from the whole trace is then removed

Experimental Setup

- Screening: five fold cross validation
- NetBill:
  - 5 datasets for training and 5 for testing
  - 2000 positive and 2000 negative traces in each dataset
- Accuracy: fraction of correctly classified traces over the total number of traces

Results

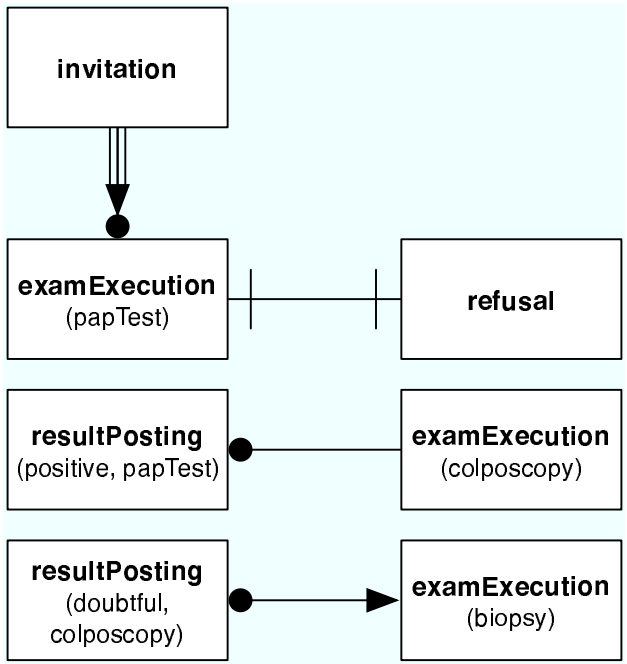
- Accuracy:

Experiment	DecMiner	α algorithm	MP Miner
Screening	97.44%	96.15%	94.89%
NetBill	96.11%	66.81%	60.52%

- Average time:
  - DecMiner: 2 minutes (Screening) and 6.5 hours for NetBill
  - α-algorithm, MP Miner: under one minute for both datasets
- Conclusion: accuracy comparable or superior to non-declarative approaches

Screening

- On the whole dataset, the following model was induced:



Conclusions and Future Work

- Conclusions
  - Declarative process mining
  - DecMiner: mining $\mathcal{S}$CIFF + translation
  - Good accuracy
  - ProM plugin
- Future Work
  - Mining of non binary DecSerFlow constraints
  - Investigating more thoroughly the application to medical guidelines (CigDec)
  - More than two classes