# Logic-Based Incremental Process Mining

Stefano Ferilli[1,2(✉)], Domenico Redavid[3], and Floriana Esposito[1,2]

[1] Dipartimento di Informatica, Università di Bari, Bari, Italy
{stefano.ferilli,floriana.esposito}@uniba.it
[2] Centro Interdipartimentale per la Logica e Applicazioni, Università di Bari, Bari, Italy
[3] Artificial Brain S.r.l., Bari, Italy
redavid@abrain.it

**Abstract.** Manually building process models is complex, costly and error-prone. Hence, the interest in process mining. Incremental adaptation of the models, and the ability to express/learn complex conditions on the involved tasks, are also desirable. First-order logic provides a single comprehensive and powerful framework for supporting all of the above. This paper presents a First-Order Logic incremental method for inferring process models. Its efficiency and effectiveness were proved with both controlled experiments and a real-world dataset.

## 1 Introduction

A *process* is a sequence of *events* associated to actions performed by agents. A *workflow* is a formal specification of how a set of tasks can be composed to result in valid processes, including sequential, parallel, conditional, or iterative execution. Each task may have preconditions and postconditions. An *activity* is the actual execution of a task. A *case* is a particular execution of actions compliant to a given workflow. Traces of cases may take the form of lists of events described by 6-tuples $(T, E, W, P, A, O)$ where $T$ is a timestamp, $E$ is the type of the event (begin of process, end of process, begin of activity, end of activity), $W$ is the name of the workflow the process refers to, $P$ is a unique identifier for each process execution, $A$ is the name of the activity, and $O$ is the progressive number of occurrence of that activity in that process. A simple trace for an evening daily routine process case might be:

(201310151930,begin_of_process,evening,1,start,1).
(201310151930,begin_of_activity,evening,1,prepare_dinner,1).
(201310152005,end_of_activity,evening,1,prepare_dinner,1).
(201310152010,begin_of_activity,evening,1,watch_tv,1).
(201310152022,begin_of_activity,evening,1,have_dinner,1).
(201310152113,end_of_activity,evening,1,have_dinner,1).
(201310152240,end_of_activity,evening,1,watch_tv,1).
(201310152245,begin_of_activity,evening,1,use_bathroom,1).
(201310152358,end_of_activity,evening,1,use_bathroom,1).
(201310152358,end_of_process,evening,1,stop,1).


A. Bifet et al. (Eds.): ECML PKDD 2015, Part III, LNAI 9286, pp. 218–221, 2015.
DOI: 10.1007/978-3-s319-23461-8_17

*Process mining* aims at inferring workflow models from examples of cases. As reported in [4], previous works have encountered problems in dealing with concurrency or in considering different occurrences of the same activity. Using statistics about task frequency and mutual ordering yields less and less accurate models as long as the number of parallel tasks and/or nested loops increases. Genetic algorithms require very long times. Previous approaches in *Declarative Process Mining*, concerned with logic-based representations, including an incremental one, need both positive and negative examples, which is not standard in the process mining setting. Some works also tried to handle noise and probabilities and investigated the possibility of mining/inducing simple boolean conditions for the activities in a propositional setting [1,5]. Here we describe WoMan, an incremental process mining system based on First-Order Logic (FOL). FOL provides a great expressiveness potential to describe in a unified framework cases, models and contextual information.

## 2 A FOL-Based Proposal

WoMan [4] works in the declarative multi-relational learning setting. We translate case traces into FOL conjunctions based on two predicates:

`activity(`$S$`,`$T$`)` : at step $S$ task $T$ is executed;
`next(`$S'$`,`$S''$`)` : step $S''$ follows step $S'$.

where the $S$'s represent timestamps denoted by unique identifiers. The previous trace would translate to:

activity(s0,start), next(s0,s1), activity(s1,prepare_dinner), next(s1,s2), activity(s2,watch_tv), next(s1,s3), activity(s3,have_dinner), next(s2,s4), next(s3,s4), activity(s4,use_bathroom), next(s4,s5), activity(s5,stop)

The process model is described as a FOL conjunction based on two predicates:

`task(`$t$`,`$C$`)` : task $t$ occurs in the multiset of cases $C$;
`transition(`$I$`,`$O$`,`$p$`,`$C$`)` : transition $p$, that occurs in the multiset of cases $C$, consists in ending all tasks in $I$ and starting all tasks in $O$.

$C$ can be exploited for computing statistics on the use of tasks and transitions, and thus to handle noise. A fragment of model accounting for the previous case would be:

| | |
|---|---|
| task(prepare_dinner,[1,...]). | transition([start]-[prepare_dinner],1,[1,...]). |
| task(watch_tv,[1,...]). | transition([prepare_dinner]-[watch_tv,have_dinner],2,[1,...]). |
| task(have_dinner,[1,...]). | transition([watch_tv,have_dinner]-[use_bathroom],3,[1,...]). |
| task(use_bathroom,[1,...]). | transition([use_bathroom]-[stop],4,[1,...]). |

This formalism is more expressive than Petri nets, in particular as regards the possibility of specifying invisible or duplicate tasks. It also permits to easily handle complex or tricky cases that require dummy or artificially duplicated

task nodes, that cannot be handled by current approaches in the literature. This representation can be naturally extended by adding relevant contextual information expressed using user-defined, domain-specific predicates.

While learning the workflow structure, the FOL case descriptions, possibly extended with contextual information, can be also exploited as examples for learning task pre-conditions, using a FOL incremental learning system (e.g., InTheLEx [3]). In the previous case, a learned rule might be:

prepare_dinner(X) :- day(X,D), saturday(D), bad_weather(X).

(meaning that if at timestamp $X$ it is bad weather, and it is saturday, then task prepare_dinner is enabled).

Differently from all previous approaches, WoMan is *fully incremental*: it can start with an empty model and learn from one case (while others need a large set of cases to draw significant statistics), and can refine an existing model according to new cases. This is a significant advance to the state-of-the-art, because continuous adaptation of the learned model to the actual practice can be carried out efficiently, effectively and transparently to the users. The learned model can be submitted to experts for analysis purposes, for improving their understanding of the process or for manually tailoring it. It can also be used to generate possible cases, or to supervise future behavior of the users and check whether it is compliant with the learned model, raising suitable warnings otherwise. The user's response to such warnings might be exploited to fix or refine the model.

## 3 Evaluation

A first evaluation of the proposed methodology used 11 artificial workflow models purposely devised to stress the learning methods, each involving different combinations of complexities and potentially tricky features. The experimental setting was as in [6]: 1000 training cases were randomly generated for each model and used to learn the model. This was repeated several times to ensure that the random generation did not affect the outcome. WoMan was able to learn the correct model in all cases within a few seconds and using less than 50 training cases. Even using 1000 examples, the technique in [6] was unable to learn 7 out of 11 test models. [2] learned the correct model only in 2 cases; nearly half of the times the wrong models did not even fulfill the syntactic requirements for Petri Nets; once it could not converge within a 6-hour limit.

WoMan was also tested on a real-world task concerning daily routines of people. Specifically, we used the Aruba dataset from the CASAS repository (`http://ailab.wsu.edu/casas/datasets.html`). It includes 220 cases involving 11 tasks, for a total of 13788 trace events. Learning showed a substantial convergence within the first 10 examples. A YAP Prolog 6.2.2 implementation of WoMan processed the whole dataset in 0.1 sec (including translation of traces into FOL). The learned model's average accuracy, evaluated by 10-fold cross-validation, was 92% (consider that each missed case costs nearly 5% accuracy). [6] and [2] returned formally wrong models according to Petri Net specifications.

Using as contextual information the status of the sensors installed in the Aruba home, task preconditions were learned as well. The Aruba dataset yielded 5976 training examples (27.16 per case on average) for learning task preconditions, but InTheLEx converged to the target theory using very few refinements (12 new clauses + 78 generalizations = 90 overall), taking only 10.97 sec per case on average. It was able to avoid overgeneralization, returning meaningful preconditions for 8 out of 11 tasks.

## 4 Conclusions

This paper presented WoMan, a process mining system based on First-Order Logic representations. It exploits a more expressive representation than previous proposals. Its incremental approach allows to learn from scratch and converge towards correct models using very few examples. It can also handle the context in which the activities take place, thus allowing to learn complex (and human-readable) preconditions for the tasks, using an First-Order Logic incremental learner. It can also handle noise in a straightforward way. Both controlled hard experiments and domain-specific ones, concerning people's daily routines, revealed that the method ensures quick convergence towards the correct model, using much less training examples than would be required by statistical techniques.

**Acknowledgments.** This work was partially funded by the Italian PON 2007-2013 project PON02_00563_3489339 'Puglia@Service'.

## References

1. Agrawal, R., Gunopulos, D., Leymann, F.: Mining process models from workflow logs. In: Schek, H.-J., Saltor, F., Ramos, I., Alonso, G. (eds.) EDBT 1998. LNCS, vol. 1377, pp. 469–483. Springer, Heidelberg (1998)
2. de Medeiros, A.K.A., Weijters, A.J.M.M., van der Aalst, W.M.P.: Genetic process mining: an experimental evaluation. Data Min. Knowl. Discov. **14**(2), 245–304 (2007)
3. Esposito, F., Semeraro, G., Fanizzi, N., Ferilli, S.: Multistrategy theory revision: Induction and abduction in inthelex. Machine Learning Journal **38**(1/2), 133–156 (2000)
4. Ferilli, S.: Woman: Logic-based workflow learning and management. IEEE Transactions on Systems, Man and Cybernetics: Systems **44**(6), 744–756 (2013)
5. Herbst, J., Karagiannis, D.: Integrating machine learning and workflow management to support acquisition and adaptation of workflow models. In: Proceedings of the 9th International Workshop on Database and Expert Systems Applications, pp. 745–752. IEEE (1998)
6. Weijters, A.J.M.M., van der Aalst, W.M.P.: Rediscovering workflow models from event-based data. In: Hoste, V., De Pauw, G., (eds.) Proceedings of the 11th Dutch-Belgian Conference of Machine Learning (Benelearn 2001), pp. 93–100 (2001)