ECAI 2016
G.A. Kaminka et al. (Eds.)

doi:10.3233/978-1-61499-672-9-1734

Abducing Workflow Traces: A General Framework to Manage Incompleteness in Business Processes

Federico Chesani[1] and **Riccardo De Masellis**[2] and **Chiara Di Francescomarino**[2] and **Chiara Ghidini**[2] and **Paola Mello**[1] and **Marco Montali**[3] and **Sergio Tessaris**[3]
[1] **University of Bologna, Viale Risorgimento 2, 40136 – Bologna, Italy**
[2]**FBK-IRST, Via Sommarive 18, 38050 Trento, Italy**
[3]**Free University of Bozen–Bolzano, piazza Università, 1, 39100 Bozen-Bolzano, Italy**

Abstract. The capability to store data about Business Process executions in so-called Event Logs has brought to the identification of a range of key reasoning services (consistency, compliance, runtime monitoring, prediction) for the analysis of process executions and process models. Tools for the provision of these services typically focus on one form of reasoning alone. Moreover, they are often very rigid in dealing with forms of incomplete information about the process execution. While this enables the development of ad hoc solutions, it also poses an obstacle for the adoption of reasoning-based solutions. In this paper we exploit the power of abduction to provide a flexible, and yet computationally effective framework able to reinterpret key reasoning services in terms of incompleteness and observability in a uniform and effective way.

1 Introduction

The proliferation of IT systems able to store process executions in so-called event logs has originated, in the Business Process (BP) community, a quest towards tools for discovering, checking the conformance and enhancing process models based on actual behaviours. Focusing on conformance, that is, to assess how a *prescriptive* (or "de jure") process model relates to the execution traces, this general notion can be declined in specific "use cases", such as *model consistency*, *trace compliance*, *runtime monitoring* and *prediction/recommendation*.

A number of different tools offer ad hoc solutions that cover only few of the "use cases" and do not easily adapt to different workflow languages. This poses a problem, given the current trend of enriching BP languages with new constructs complementing the control flow knowledge. A second rigidity is in dealing with *incomplete information* about the process execution: the presence of not monitorable activities or errors in the logging procedure makes handling *incomplete event data* one of the main challenges of the BP community

In this work we exploit the paradigm of Abductive Constrained Logic Programming (ACLP, [4]), and the SCIFF abductive framework [1] to provide a general purpose environment able to support *conformance in its different "use cases"* in the presence of *incomplete event data*. Indeed, abduction fits in a natural manner: facts are observed in the execution traces, and need to be explained/diagnosed with respect to what is envisaged by the process model.

2 Process Models, Reasoning, and Incompleteness

We focus on structured process models in the spirit of [5], enriched with temporal constraints. We illustrate our investigation with an example of temporal workflow taken from [6], and reproduced in the left hand side of Figure 1. This workflow contains: 14 activities ($\mathsf{A1}, \ldots \mathsf{A14}$), 2 pairs of exclusive gateways ($\langle \mathsf{X1}, \mathsf{X4} \rangle, \langle \mathsf{X2}, \mathsf{X3} \rangle$), and 1 pair of parallel gateways ($\langle \mathsf{P1}, \mathsf{P2} \rangle$). In addition, activities are labelled with expressions of the form $[d_{min}, d_{max}]$ (*duration range* of the activity), while dashed arrows between activities expresses *inter-task constraints* involving the start or end of the activities. A sample trace that logs the execution of the aforementioned process is:

$$\begin{gathered}\{(\mathsf{A1}, [2, 7]), (\mathsf{A2}, [10, 15]), (\mathsf{A3}, [16, 46]), (\mathsf{A4}, [50, 200]), \\ (\mathsf{A13}, [300, 317]), (\mathsf{A14}, [320, 330])\}\end{gathered} \tag{1}$$

To incorporate incompleteness into the picture, the process models we consider are also equipped with *observability information*. Activities can be: (i) *observable*, if they are always explicitly logged, even if some information (e.g., the activity name or the starting time) may be missing; (ii) *non-observable*, if they are never logged, even if executed; (iii) *partially observable*, if they may or may not have been logged. We introduce the notion of *structured process with observability and time* (SPOT) as the tuple $\langle \mathcal{A}, obs, P, dur, tcon \rangle$, where $\mathcal{A}$ is a finite set of *activity (names)*; $obs : \mathcal{A} \rightarrow \{\mathtt{o}, \mathtt{n}, \mathtt{p}\}$ provides observability about each activity; P is the *top process block*, inductively defined as either a $\mathtt{a} \in \mathcal{A}$, or a combination of $\{\mathtt{seq}, \mathtt{xor}, \mathtt{and}, \mathtt{or}\}$ blocks; dur and $tcon$ captures the activity duration and the inter-task constraints.

A trace is a set of triples $\langle \mathtt{a}, s, e \rangle$ denoting the happened event $\mathtt{a}$, and the start/end timestamps s, e. Missing information units will be denoted with the special symbol "_". A trace is *partially specified* if some of its event triples contain the symbol _, *fully specified* otherwise. Moreover, we distinguish between *total* and *partial traces*, where in the latter additional event triples may be implicitly present even if not explicitly listed in the trace.

The notion of **Strong Compliance** applies to total traces, and requires a trace to be compliant to the blocks structure, and also the duration and inter-task temporal constraints. It reflects the usual notion of compliance for business processes, where it is assumed that the trace represents a complete end-to-end execution, such as in (1).

Other resoning services supported in our framework are the **Model Consistency**, that checks if a SPOT enables acceptable executions from start to end.; and, more related to incompleteness, the **Conditional Compliance**, that handles the case where the trace under analysis is indeed partial and/or partially specified: if such incompleteness hinders the possibility of replaying it on the process model, strong compliance might be regained by assuming that the trace included additional information on the missing or partially specified

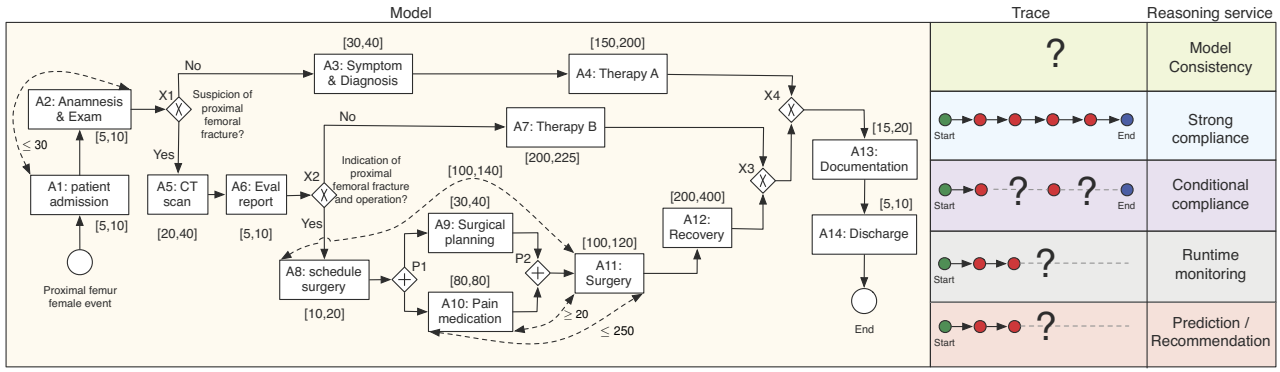

Figure 1. A process for femoral fracture treatment, reasoning services and incomplete execution traces.

events. The trace:

$$\{(\mathsf{A1},[2,7]),(\mathsf{A2},\_)(\mathsf{A4},[50,200]),(\mathsf{A13},[300,317]),(\mathsf{A14},[320,330])\} \quad (2)$$

is conditionally compliant, i.e. it might be considered strong compliant under the hypotheses that A2 was executed (satisfying the temporal constraints) and A3 (missing in the trace) was executed as well. Note that the set of assumptions needed to reconstruct strong compliance is not necessarily unique. This because alternative strongly compliant real process executions might have led to the recorded partial trace.

Finally, when dealing with ongoing executions, **Runtime Monitoring** aims to detect early violations of compliance / ensure the existence of a positive outcome, while **Prediction/Recommendation** provides (if possible) a completion that satisfies the model.

3 Abduction and Incomplete Processes

Abductive Logic Program (ALP) [4] is a non-monotonic reasoning process where hypotheses are made to explain observed facts. Given a set Γ of logical assertions known to hold, and a formula ϕ (corresponding to observed facts), abduction looks for a further set Δ of hypothesis, taken from a given set of abducible $\mathcal{A}$, which complements Γ in such a way that ϕ can be inferred (in symbols $\Gamma \cup \Delta \models \phi$). The set Δ is called *abductive explanation* (of ϕ).

In this paper we leverage on ACLP and on the SCIFF abductive logic programming framework [1], efficiently implemented using the CHR framework [2]. Beside the general notion of abducible, the SCIFF framework has been enriched with the notions of *happened event*, *expectation*, and *compliance* of an observed execution with a set of expectations. In our context, happened events account for events that have been logged in the trace, while abducibles are used to make hypothesis on events that are not recorded in the examined trace.

Our solution consists on translating a SPOT model into an ACLP program encoded using the SCIFF notation: observable activities are always expected to be observed, while partially obsefvable activities can be abduced (hypothesized) if not presente in the trace. Then, the SCIFF proof procedure is queried for possible abductive answers Δ_i that explain the observations (the given trace), under the conditions imposed by the process model. Temporal constraints are mapped by means of CLP predicates, that are fully supported within the SCIFF framework, thus allowing also the required temporal reasoning.

The different reasoning services are all supported within the SCIFF framework: *Model consistency* is directly supported by assuming that all the activities are partially observable, and the trace to be verified is empty. *Runtime monitoring* is alike, with the difference that the trace captures the events already happened. *Strong compliance* corresponds to the original reasoning task of the SCIFF, and it is still supported in our context: indeed, it corresponds to looking for at least an empty abductive answer Δ_i. *Conditional compliance* is supported as well, by looking at non-empty abductive answers. Similarly, *prediction/recommendation* is supported by simply applying out approach at run-time, and by interpreting the Δ_i as suggestions. A prototype implementation of the framework is currently available for download at http://ai.unibo.it/AlpBPM.

4 Conclusions

We have presented an abductive framework to support business process compliance, in its different forms, by attacking the different forms of incompleteness that may be present in an execution trace. Empirical evaluation using the model in Figure 1 shows that the different reasoning services can be computated with times spanning from few millisec. (when looking for a solution) up to several minutes (when looking for all the solutions). Such difference is directly related to the degree of incompleteness in the model and in the trace.

Concerning future development, the SCIFF framework is based on first-order logic, thus paving the way towards the incorporation of data [3] and the management of more sophisticated forms of incompleteness. A further reasoning service on temporal workflows is the one of controllability. Moreover, an extension of our work to deal with dynamic controllability, by integrating constraint propagation and filtering, would be an interesting and feasible future work.

Acknowledgements. This research has been partially carried out within the Euregio IPN12 KAOS, funded by the "European Region Tyrol-South Tyrol-Trentino" (EGTC) under the first call for basic research projects.

REFERENCES

[1] M. Alberti, F. Chesani, M. Gavanelli, E. Lamma, P. Mello, and P. Torroni, 'Verifiable agent interaction in abductive logic programming: The SCIFF framework', *ACM Trans. Comput. Log.*, **9**(4), (2008).
[2] M. Alberti, M. Gavanelli, and E. Lamma, 'The chr-based implementation of the SCIFF abductive system', *Fundamenta Informaticae*, **124**(4), 365–381, (2013).
[3] R. De Masellis, F. M. Maggi, and M. Montali, 'Monitoring data-aware business constraints with finite state automata', in *Proc. of ICSSP*. ACM Press, (2014).
[4] A. C. Kakas, R. A. Kowalski, and F. Toni, 'Abductive logic programming', *J. Log. Comput.*, **2**(6), (1992).
[5] Bartek Kiepuszewski, Arthur Harry Maria ter Hofstede, and Christoph J. Bussler, 'On structured workflow modelling', in *Seminal Contributions to Information Systems Engineering*, Springer, (2013).
[6] A. Kumar, S. R. Sabbella, and R. R. Barton, *Business Process Management: 13th Intl. Conf., BPM 2015, Innsbruck, Austria, August 31 – September 3, 2015, Procs.*, chapter Managing Controlled Violation of Temporal Process Constraints, 280–296, Springer International Publishing, Cham, 2015.