Advances in Quantitative Analysis of Free-Choice Workflow Petri Nets[*]

Javier Esparza

Technische Universität München, Munich, Germany
esparza@in.tum.de

Abstract

We survey recent results on the development of efficient algorithms for the quantitative analysis of business processes modeled as workflow Petri nets. The algorithms can be applied to any workflow net, but have polynomial runtime in the free-choice case.

1998 ACM Subject Classification D.2.4 Software/Program Verification, D.2.2 Design Tools and Techniques, F.1.1 Models of Computation

Keywords and phrases Free-choice Petri Nets, concurrency theory, quantitative verification

Digital Object Identifier 10.4230/LIPIcs.TIME.2017.2

Category Invited Talk

1 Introduction

Workflow Petri nets are a successful formalism for the representation and formal analysis of business processes. They are also much used as a formal back-end for different notations like BPMN (Business Process Modeling Notation), EPC (Event-driven Process Chain), or UML Activity Diagrams [1, 2, 5]. In this note we assume that the reader is familiar with basic Petri net terms: place, transition, token, marking, and the firing rule, that is, the rule that determines whether a transition is enabled at a marking, and how the marking changes when the transition fires.

In a nutshell, a workflow Petri net is just a Petri net with a distinguished initial place and a distinguished final place. These places induce distinguished initial and final markings, which contain a token in the initial/final place and no tokens elsewhere. In a well-designed workflow, every marking reachable from the initial marking enables some firing sequence leading to the final marking, a property known as *soundness* [1]. In particular, sound workflow nets are both deadlock-free and livelock-free.

Workflow Petri nets can be analyzed by constructing their reachability graph (which has the reachable markings as nodes, and the steps allowed by the firing rule as edges), and applying model-checking techniques, see e.g. [16, 23]. However, this approach suffers from the well-known state-explosion problem: Even if reachable markings put at most one token in every place, a workflow net with n places can still have $\Theta(2^n)$ reachable markings. For simple qualitative properties, like soundness, the state-explosion problem can be handled very effectively by tools like LoLA [22, 16]. However, the problem becomes acute for workflows enriched with data, time, and/or probabilities.

Since 2013 my co-authors and I have addressed this question by developing novel analysis algorithms that can be applied to general workflow nets, but provide strong runtime guarantees

[*] Supported by the DFG project “Negotiations: A Model for Tractable Concurrency”.

for special classes. The rationale for this approach is that workflow Petri nets modeling real-life business processes tend to have a simpler structure than those found in other application areas, like the analysis of distributed algorithms, concurrent programs, or biological processes. In particular, it has been repeatedly observed that many of these workflow nets are *free-choice* [6]. For example, Workflow Graphs, a simple but effective business process formalism [1, 17, 18, 13], can be translated into free-choice workflow Petri nets, and 1386 of the 1958 workflow nets in the most popular benchmark suite in the literature are free-choice workflow nets [16]. So our goal is to design analysis algorithms that are applicable to arbitrary workflow nets, and provide a better runtime guarantee in the free-choice case. Typically, our algorithms have exponential worst-case complexity in the general case (which is unavoidable due to NP-hardness or PSPACE-hardness results), but polynomial complexity in the free-choice case.

A Petri net is free-choice if every pair of places has either the same set of output transitions or disjoint sets of output transitions. The consequence is that for every reachable marking, if some output transition of a place can fire, then *all* output transitions can fire, that is, the net can *freely choose* which output transition to fire. While free-choice Petri nets have a rich theory, almost all results concerning them are about the basic Petri net model, and do not apply to Petri nets enriched with data, time, or probabilities. In a series of papers, my co-authors and I have developed novel analysis techniques that overcome this problem. In the rest of the note we summarize this work. We first consider our work on reduction algorithms, and then our last paper on decomposition-based algorithms.

▶ Remark. Some of our results are formulated in terms of *deterministic negotiation diagrams*. In a nutshell, negotiation diagrams, introduced in [11], are workflow Petri nets that can be decomposed into communicating sequential Petri nets, a feature that makes them more amenable to theoretical study. A classical theorem of net theory shows that the connection between sound deterministic negotiation diagrams and sound free-choice workflow Petri nets is very tight: There are simple, polynomially computable translations between these two formalisms, which moreover only incur in a linear blow up. More details can be found in [7].

2 Reduction techniques

Reduction algorithms are a very efficient analysis technique for Petri nets and other business modeling formalisms, like EPCs and AND-XOR graphs (see for instance [21, 3, 9, 24]). A reduction algorithm consists of (a) a set of *reduction rules*, whose application allows one to simplify the workflow while preserving important properties, and (b) an algorithmic policy for selecting the next rule to be applied. The algorithm applies the rules exhaustively until it reaches an irreducible workflow net. For certain classes of nets and certain properties, the rules can be *complete*: They can reduce all workflows in the class satisfying the property, and only them, to some unique canonical workflow. Typically, this canonical workflow is the workflow consisting of one single transition with the initial and final places as only input and output place. If the policy followed by the algorithm is guaranteed to reach the canonical workflow, then the reduction algorithm becomes a decision algorithm for the property. Moreover, the algorithm decides the property without having to explore the reachability graph of the net at all.

A set of rules for free-choice Petri nets (not necessarily workflow nets) was presented in [6]. The rules preserve liveness and boundedness, two important properties of Petri nets, and are shown to be complete. In [3] these rules were applied to free-choice workflow Petri nets, and shown to be complete for the soundness property. This leads to a polynomial-time decision procedure for soundness of free-choice workflow nets, in sharp contrast with the PSPACE-hardness of deciding soundness for general workflow nets[1]. However, the rules of [6] have two problems. First, as shown in [13], they do not preserve properties concerning data or timing information. Moreover, as shown in [6], one of the rules is not correct for arbitrary workflow nets. More precisely, applying the rule to a sound, non-free-choice workflow net can make it unsound. In [13] we present a new set of surprisingly simple rules that overcomes these shortcomings. The rules can be applied to Petri nets in which tokens carry data. For example, a token in a certain place can be labeled with a natural number. Transitions collect a tuple of data from their input places, and apply a *transformer* to them, yielding a tuple of data that is sent to the output places.

The rules not only preserve soundness/unsoundness, but also the *input/output relation* of the workflow. This is the relation that assigns to every initial marking the set of final markings reachable from it (observe that initial markings differ only on the value of the token in the initial final place, and final markings on the value of the token in the final place). Therefore, the rules can be applied to decide any property of the input/output relation, and, by suitable reductions, other properties like the worst-case execution time of a given workflow. The rules also solve the second problem: contrary to the original rules, they can be applied to arbitrary workflow nets. Finally, the rules are still complete for free-choice workflow nets, in the sense that they reduce every sound free-choice workflow net to a workflow net with only one transition, but the same input/output relation.

The definitive description of the reduction algorithm, whose correctness proof and complexity analysis are rather complex, is given in [8], an extended and corrected version of [11, 12], which is currently under review. The algorithm completely reduces sound free-choice workflow nets by means of a sequence of rule applications of length at most cubic in the number of places and transitions of the net (in experiments the actual number of rule applications only grows linearly in the size of the net). Further, the sequence of reductions can be found in polynomial time. In [14] we apply this reduction algorithm to the problem of computing the expected cost of a workflow. For this, we define probabilistic workflow nets with costs, and enhance the reduction rules of [13] so that they preserve the expected cost. Using the results of [8] we prove that the expected cost of a free-choice workflow net can be computed in polynomial time.

3 Decomposition-based techniques

Our most recent work is presented in [15]. This paper generalizes the results of [13, 14]. It presents the most versatile analysis algorithm for free-choice workflow nets designed so far, among those that avoid the construction of the reachability graph. In particular, the reduction-based algorithms of [13, 14] can be recast as special cases of this general algorithm. The algorithm can be instantiated to solve problems about the time needed to execute a workflow, and also about its cost[2]. More precisely, the generic algorithm yields polynomial algorithms for the computation of the worst-case and the best-case execution time, and for the worst-case, best-case, and expected cost. It also provides a good algorithm for the computation of the expected time, although in this case the algorithm is not polynomial.

[1] The exact complexity depends on the specifics of the workflow model, for instance whether the workflow Petri net is assumed to be 1-safe or not.

[2] Notice the difference between time and cost: while the cost of executing two concurrent transitions is the sum of the costs, the time is the maximum of the times.

This is however to be expected, since the problem of deciding if the expected time exceeds a given bound is NP-complete, even for acyclic free-choice workflow nets [4].

The paper extends the classical lattice-based formalism for the static analysis of sequential flow-graphs, as presented for example in [20], to workflow Petri nets. A flow-graph consists of a set of nodes, modeling program points, and a set of edges, modeling program instructions, like assignments or guards. In the lattice-based approach one (i) defines a lattice $\mathcal{D}$ of dataflow informations corresponding to the possible results of the analysis to be conducted, (ii) assigns semantic transformers $[\![a]\!] \colon \mathcal{D} \to \mathcal{D}$ to each action a of the flow-graph, (iii) assigns to a path $a_1 \cdots a_n$ of the flow graph the functional composition $[\![a_n]\!] \circ \cdots \circ [\![a_1]\!]$ of the transformers, and (iv) defines the result of the analysis as the "Merge Over all Paths", i.e., the join of the transformers of all execution paths, usually called the MOP-solution or just the MOP of the dataflow problem. So performing an analysis amounts to computing the MOP of the flow-graph for the given lattice and the given transformers.

Katoen *et al.* have recently shown in [19, 10] that in order to adequately deal with quantitative analyses of concurrent systems one needs a semantics that distinguishes between the inherent nondeterminism of each sequential process, and the nondeterminism introduced by concurrency (the choice of the process that should perform the next step). Following these ideas, we introduce a semantics in which the latter is resolved by an external scheduler, and define the MOP for a given scheduler. The result of a dataflow analysis is then given by the infimum or supremum, depending on the application, of the MOPs for all possible schedulers.

In [15] we define the class of *Mazurkiewicz-invariant frameworks*. Loosely speaking, a framework is Mazurkiewicz-invariant if two executions of the workflow net that differ only in the order of execution of concurrent transitions have the same transformer. We prove a theorem showing a first important property of sound free-choice workflow nets, namely that the MOP is independent of the scheduler for Mazurkiewicz-invariant frameworks. This allows to compute the result of the analysis by fixing a scheduler, and computing the MOP for it. The main contribution of the paper is a method to compute the MOP of a framework for a sound free-choice workflow net. This is achieved by proving a novel and very powerful *decomposition theorem* showing that a sound free-choice workflow net is composed of smaller workflow subnets which are also sound. The algorithm iteratively identifies these subnets, computes their MOP, and replaces the complete subnet by one single transition with the same MOP.

Since the algorithm is generic, its complexity depends on the choice of the lattice and the transformers. However, we obtain a "concurrency-for-free" result: The runtime of the algorithm for computing the MOP for a sound free-choice workflow net is within a polynomial factor of the runtime for computing the MOP of a sequential flow-graph. Notice that this is the case even though the reachability graph of the workflow net – which can be seen as the sequential flow-graph equivalent to it – can be exponentially larger than the net itself.

The generic algorithm is more general than the reduction-based algorithms. More precisely, for every reduction-based algorithm, there is an instance of the generic algorithm with at most the same complexity. Further, using the generic algorithm we can solve in polynomial time quantitative problems for which no reduction algorithm existed so far. An example is the computation of the maximal number of tokens of a reachable marking of a sound free-choice workflow net. As shown in [4], the maximal number of tokens corresponds to the minimal number of resources that guarantee successful completion of the workflow, and is therefore an important parameter. Our generic decomposition-based algorithm can be instantiated to compute the maximal number of tokens in polynomial time[3].

[3] This research, conducted with Philipp Meyer and Hagen Völzer, is still unpublished.

References

1 Wil van der Aalst. The application of Petri nets to workflow management. *Journal of Circuits, Systems, and Computers*, 8(1):21–66, 1998.

2 Wil van der Aalst and Kees Max van Hee. *Workflow management: models, methods, and systems.* MIT press, 2004.

3 Wil van der Aalst, Alexander Hirnschall, and Eric Verbeek. An alternative way to analyze workflow graphs. In *Advanced Information Systems Engineering*, volume 2348 of *Lecture Notes in Computer Science*, pages 535–552. Springer, 2002.

4 Mirela Botezatu, Hagen Völzer, and Lothar Thiele. The complexity of deadline analysis for workflow graphs with multiple resources. In *BPM*, volume 9850 of *Lecture Notes in Computer Science*, pages 252–268. Springer, 2016.

5 Jörg Desel and Thomas Erwin. Modeling, simulation and analysis of business processes. In *Business Process Management*, volume 1806 of *Lecture Notes in Computer Science*, pages 129–141. Springer, 2000.

6 Jörg Desel and Javier Esparza. *Free choice Petri nets.* Cambridge University Press, 2005.

7 Jörg Desel and Javier Esparza. Negotiations and Petri nets. *Transactions in Petri Nets and Other Models of Concurrency*, 11:203–225, 2016.

8 Jörg Desel, Javier Esparza, and Philipp Hoffmann. Negotiation as concurrency primitive. *CoRR*, abs/1612.07912, 2016.

9 Boudewijn van Dongen, Wil van der Aalst, and Eric Verbeek. Verification of EPCs: Using reduction rules and Petri nets. In *Advanced Information Systems Engineering*, pages 372–386. Springer, 2005.

10 Christian Eisentraut, Holger Hermanns, Joost-Pieter Katoen, and Lijun Zhang. A semantics for every GSPN. In *PETRI NETS*, volume 7927 of *Lecture Notes in Computer Science*, pages 90–109. Springer, 2013.

11 Javier Esparza and Jörg Desel. On negotiation as concurrency primitive. In *CONCUR*, volume 8052 of *Lecture Notes in Computer Science*, pages 440–454. Springer, 2013.

12 Javier Esparza and Jörg Desel. On negotiation as concurrency primitive II: Deterministic cyclic negotiations. In *FoSSaCS*, volume 8412 of *Lecture Notes in Computer Science*, pages 258–273. Springer, 2014.

13 Javier Esparza and Philipp Hoffmann. Reduction rules for colored workflow nets. In *FASE*, volume 9633 of *Lecture Notes in Computer Science*, pages 342–358, 2016.

14 Javier Esparza, Philipp Hoffmann, and Ratul Saha. Polynomial analysis algorithms for free choice probabilistic workflow nets. In *QEST*, volume 9826 of *Lecture Notes in Computer Science*, pages 89–104. Springer, 2016.

15 Javier Esparza, Anca Muscholl, and Igor Walukiewicz. Static analysis of deterministic negotiations. *CoRR*, abs/1704.04190, 2017. To appear in the Proceedings of LICS'17.

16 Dirk Fahland, Cédric Favre, Barbara Jobstmann, Jana Koehler, Niels Lohmann, Hagen Völzer, and Karsten Wolf. Instantaneous soundness checking of industrial business process models. In *BPM*, volume 5701 of *Lecture Notes in Computer Science*, pages 278–293. Springer, 2009.

17 Cédric Favre, Dirk Fahland, and Hagen Völzer. The relationship between workflow graphs and free-choice workflow nets. *Information Systems*, 47:197–219, 2015.

18 Cédric Favre, Hagen Völzer, and Peter Müller. Diagnostic information for control-flow analysis of Workflow Graphs (a.k.a. Free-Choice Workflow nets). In *TACAS*, volume 9636 of *Lecture Notes in Computer Science*, pages 463–479, 2016.

19 Joost-Pieter Katoen. GSPNs revisited: Simple semantics and new analysis algorithms. In *ACSD*, pages 6–11. IEEE Computer Society, 2012.

20 Flemming Nielson, Hanne Riis Nielson, and Chris Hankin. *Principles of program analysis.* Springer, 1999.

21 Wasim Sadiq and Maria E. Orlowska. Analyzing process models using graph reduction techniques. *Information systems*, 25(2):117–134, 2000.

22 Karsten Schmidt. LoLA: A low level analyser. In *ICATPN*, volume 1825 of *Lecture Notes in Computer Science*, pages 465–474, 2000. Current version available at `http://www.service-technology.org`.

23 Nikola Trcka, Wil M. P. van der Aalst, and Natalia Sidorova. Data-flow anti-patterns: Discovering data-flow errors in workflows. In *CAiSE 2009*, volume 5565 of *Lecture Notes in Computer Science*, pages 425–439, 2009.

24 Eric Verbeek, Moe Thandar Wynn, Wil van der Aalst, and Arthur ter Hofstede. Reduction rules for reset/inhibitor nets. *Journal of Computer and System Sciences*, 76(2):125–143, 2010.