Fiona
A Tool to Analyze Interacting Open Nets

Peter Massuthe and Daniela Weinberg

Humboldt–Universität zu Berlin, Institut für Informatik
Unter den Linden 6, 10099 Berlin, Germany
{massuthe,weinberg}@informatik.hu-berlin.de

Abstract. Fiona is a tool that has been designed to check behavioral correctness of a service and to analyze the interaction of services in service oriented architectures, for instance. It implements very efficient data structures and algorithms, which have partly been adapted from LoLA. Fiona has been proven to be applicable in practice by service designers, service publishers, and service brokers. This tool paper describes the functionality of Fiona and provides an insight into its architecture.

1 Introduction

In this paper we present Fiona (available at `http://www.service-technology.org/fiona`), a tool to analyze the interaction of services. Its features cover the check of controllability and the construction of an operating guideline of a service as well as other derived notions. *Controllability* [1, 2] is a minimal correctness criterion of a service stating the existence of a behaviorally compatible partner for the service. An *operating guideline* (OG) [3, 4] of a service is an operational characterization of *all* behaviorally compatible partners of this service.

As a formal model for services we use *open nets* [5, 4], a special class of Petri nets that extend classical Petri nets by an interface for communication with other open nets. This idea is based on the module concept for Petri nets which was first proposed by Kindler [6].

The development of Fiona started in 2006 as a reimplementation of a tool called Wombat (available at `http://www.informatik.hu-berlin.de/top/wombat`). Wombat was designed to constructively decide controllability of an open net (called workflow module in Wombat). The main reasons for a reimplementation were (1) a completely new theoretical foundation for deciding controllability, (2) a desired focus on efficiency rather than mere effectiveness, (3) the newly developed concept of an operating guideline, and, thus, (4) the need for a separation of computing compatible interactions from reduction rules to ensure efficiency of the algorithms.

Currently, Fiona is developed distributedly by the groups of Wolfgang Reisig (Humboldt-Universität zu Berlin) and Karsten Wolf (University of Rostock, Germany). It is written in C++ and released as free software under the terms of the

GNU General Public License. Fiona's distribution is based on the GNU autotools, which provide the possibility to run Fiona on most operating systems.[1]

The functionality of Fiona comprises (among others) the following features:

Controllability. Controllability of an open net N is decided by synthesizing a partner of N (as an automaton called *interaction graph* (IG) [2]). If no partner can be synthesized (i.e. the IG is empty), then N is un-controllable.

Operating guideline. An operating guideline (OG) [4] is a finite characterization of *all* behaviorally compatible partners by annotating a single partner (as automaton) with Boolean formulae in order to derive all other partners.

Matching. Given an open net M and an operating guideline OG_N of an open net N, Fiona decides whether M is behaviorally compatible to N by matching M with OG_N. Matching is more efficient than composing the two nets and model checking the composition (as proposed by other approaches, like the *public view* approach).

Partner synthesis. Given an open net N, Fiona computes a behaviorally compatible partner open net M (if possible). The synthesis can be triggered to construct a small M (with respect to communication) or a partner M which exhaustively communicates with N.

Substitutability. Given two open nets N and N', Fiona can compare their sets of compatible partners using the corresponding OGs OG_N and $OG_{N'}$. If $OG_{N'}$ comprises OG_N, then N can be substituted by N' without rejecting any compatible partner. Different notions of substitutability are described in [7]. The corresponding decision procedures are implemented in Fiona.

Adapter generation. Given two open nets M and N which are *not* behaviorally compatible, we can consider their composition as an open net and ask Fiona to synthesize a partner. If such a partner exists, it constitutes an *adapter* A, mediating between M and N and making the composition of M, N, and A well-behaving by construction. For more details see [8].

Fiona is a stand-alone tool, designed to be used as a background service of existing service modeling tools. Therefore, Fiona has no graphical interface — the analysis task as well as the input file(s) are given to Fiona via command line options. Fiona then computes and reports the result and, if needed, generates the output file(s).

2 Context

Fiona can be used within the new computing paradigms of service-oriented computing (SOC) and service-oriented architectures (SOA) (see [9] for an overview), as well as other areas of intra- and interorganizational business process modeling and analysis [10]. Therein, a *service* represents a self-contained software unit that offers an encapsulated functionality via a well-defined interface. Services are used as building blocks to implement complex, highly dynamic, and flexible

[1] It compiles on MS Windows® (with Cygwin), Unix (Solaris), Linux, and Mac® OS.

business processes. SOAs introduce a *service broker* to organize the challenges of *service discovery*, i.e. the publishing and management of available services and the introduction of procedures to enable a client to find and use such a service.

Controllability is a minimal requirement for the correctness of a service and is particularly relevant for service designers. Operating guidelines are suited to support service discovery and can be used to decide substitutability of services or to generate adapters for incompatible services [8]. Thus, FIONA is intended to be used by service designers, by service providers, and by service brokers.

3 How to Use FIONA

FIONA accepts two different types of files as its input. The open net file format (`.net`) has been adapted from the LoLA [11] file format. It is fairly easy to read and to comprehend. So, it is possible to model an open net manually. Additionally, there exists a compiler BPEL2oWFN (available at `http://service-technology.org/bpel2owfn`) that implements a feature-complete open net semantics [12] for WS-BPEL and automatically generates an open net file for a given WS-BPEL process. Further, FIONA is able to read a textual representation of an operating guideline stored in a special file format (`.og`).

In the following we describe the main use cases of FIONA.

Controllability. Given an open net as a `.net` file, FIONA decides controllability by constructing an IG. The corresponding command is:

```
> fiona -t ig nets/example.net
```

After computing the IG, FIONA reports whether the net is controllable or not. Furthermore, a graphic file showing the IG is generated by invoking GraphViz Dot (available at `http://www.graphviz.org`).

Operating guideline. Given an open net as a `.net` file, FIONA constructs the OG of the net using the following command.

```
> fiona -t og nets/example.net
```

FIONA reports whether N is controllable or not and a graphic file showing the OG is generated. Further, a file called `example.net.og` is generated that represents the OG of the net textually.

Matching. If a `.net` file and an `.og` file is given, FIONA can decide whether the given net is behaviorally compatible to the net that the given OG corresponds to. The following command is used for invoking the matching algorithm.

```
> fiona -t match nets/client.net ogs/service.net.og
```

FIONA will report whether the net matches with the given OG or not. In case the matching fails, FIONA reports where the failure manifests in.

Partner synthesis. By passing a `.net` file to FIONA it is possible to construct a partner of that net. FIONA can construct two different types of partners: a very small one or a partner that exhaustively communicates with the net.

```
> fiona -t smallpartner nets/example.net
> fiona -t mostpermissivepartner nets/example.net
```

In both cases FIONA will create an open net `example-partner.net` that represents the partner. This is done with the help of the tool Petrify (available at `http://www.lsi.upc.es/~jordicf/petrify/distrib/home.html`).

FIONA provides some more analysis and construction features. A list of all features is available by invoking FIONA with the parameter `-help`. Using the parameter `-d 1`, ..., `-d 5` it is possible to retrieve detailed information of computation progress and debug information. Furthermore, the IG and OG graphics can be enriched to show more details of the graphs. Additionally, there exists a message bound parameter (`-m`) that refers to the *k-limited communication* [4] of open nets. The message bound limits the number of tokens that an interface place can carry simultaneously. The default is `-m 1`, i.e. interface places are safe.

4 Implementation Details

In the following we describe the main architecture of FIONA to construct an IG and an OG, which are the two basic features of FIONA and provide the basis of most other functionalities of FIONA.

In either case, an automaton called *communication graph* (CG) is constructed first, from which both IG and OG will be derived. The CG consists of *nodes* and *events*. The events reflect possible sending or receiving actions of a partner P of the considered net N. Each node q of the CG contains the set of markings of N, which can be reached by consuming and producing the messages along any path from the initial node of the CG to q. That set of markings is called *knowledge* at q ($k(q)$, for short) [4], representing the hypothesis of P about the state of N.

Representation of markings. The markings of N are stored in a data structure which was adapted from LoLA. The more markings it stores, the more the structure converges into a decision tree. It has proven to be very space efficient while allowing to decide the containment of a marking in linear time.

Knowledge calculation. The knowledge $k(q_0)$ of the root node q_0 of the CG consists of all markings reachable from the initial marking m_0 of N. For calculating the knowledge of $q \neq q_0$, $k(q)$ is initially filled with those markings derived from the predecessor nodes' knowledges that represent the occurrence of the event(s) leading to q. Then, FIONA computes the markings m' that are reachable from an $m \in k(q)$ and adds these m' to $k(q)$. The computation of the reachability graph is done as in LoLA. For instance, instead of backtracking, we fire a transition backwards.
The calculation of $k(q)$ can be enhanced by using stubborn sets. Here, only representative markings are stored in $k(q)$ (FIONA parameter `-R`) which may significantly reduce the space required for storing the CG.

Node classification. Each node is classified as either red or blue. The blue nodes constitute the IG/OG later on, whereas a red node represents a state of the partner P that is behaviorally incompatible with the open net N. Such nodes must be avoided and are not part of the IG/OG. Initially, every node q is blue. Then, we analyze $k(q)$ for non-final deadlocks and check for each such deadlock whether it is resolvable by an event of P. If some deadlock is not resolved, q is set to red. If $k(q)$ contains a marking that violates the message bound, q is set to red, too. By backtracking, the red color of q is propagated to its predecessors. To speed up the computation, red nodes are not deleted but stored to avoid a repeated computation of such a node.

Annotating a node. For each node q a Boolean annotation $\phi(q)$ (in conjunctive normal form) is stored. The annotation of q is uniquely defined by $k(q)$: each deadlock $d \in k(q)$ is represented by a clause c where each literal of c represents an event that resolves d. The annotation is part of the OG later on, but is also used to drive the order in which the successor nodes are computed (IG and OG).
$\phi(q)$ can also be used as an early classification of q: each literal x corresponds to an event leading from q to a node q' in the CG. If q' is red, then x is set to *false*. If thereby $\phi(q)$ becomes unsatisfiable, q becomes red immediately.

Successor node computation. For each blue node q we perform each event that occurs in $\phi(q)$, i.e. each event resolving at least one deadlock. For being able to apply the early classification as often as possible, we sort the clauses of $\phi(q)$ by its length and follow events of short clauses first.
In case of IG computation, we only consider a subset of the possible events: several reduction rules [2] for the CG are proven to preserve controllability.

The data structure of the CG is the basis of deciding controllability and for constructing an OG. The OG provides the basis of the matching algorithm. The IG/OG is used to construct a partner (small/most permissive) for a given net.

Controllability. The IG contains the blue nodes of the CG which was computed by applying all reduction rules. The annotations and knowledge values of the nodes are discarded. The open net N is controllable if and only if the root node of the CG is blue. FIONA reports this decision (and statistical numbers of the IG and CG sizes) on the command line; a graphical representation of the IG is generated.

Operating guideline. The OG consists of the blue nodes of the CG, too, this time computed without reductions. Again, the knowledges are discarded, but the annotations remain. In the graphical representation of the OG the annotation of a node q is shown inside of q. FIONA generates a graphical and a textual representation of the OG and stores both in separate files.

Matching. The matching algorithm is a check whether (1) the given net M is simulated by the given operating guideline OG_N and (2) M satisfies all annotations of OG_N. Therefore, FIONA performs a coordinated depth-first search through the state space of M and OG_N and evaluates each formula $\phi(q)$ by interpreting the currently enabled transitions of M as an assignment for $\phi(q)$. Eventually, FIONA will report the matching result. In case that M does not match, FIONA will report the marking of M and the node of OG_N where the simulation or the annotation is violated.

Partner synthesis. FIONA can synthesize two types of partners for a given open net N. A small partner is constructed based on the IG of N, or a most-permissive partner is computed out of the OG of N. Either graph provides the basis of the input of the tool Petrify, which creates the corresponding open net M. M is behaviorally compatible with N by construction.

5 Conclusion

We have presented the tool FIONA that is designed to be used by service designers, service brokers, and service publishers. Some of its data structures and algorithms have been adapted from LoLA. Further, we have put great effort on the theoretical basis of our algorithms in order to make the computations efficient with respect to time and memory consumption. Our case studies show that FIONA is indeed suitable to be used in practice [4, 2].

FIONA has been integrated into the ProM framework, a process mining tool kit with plug-able architecture (available at `http://prom.sourceforge.net/`).

References

1. Schmidt, K.: Controllability of Open Workflow Nets. In: EMISA 2005. LNI, Bonner Köllen Verlag (2005) 236–249
2. Weinberg, D.: Efficient controllability analysis of open nets. In: WS-FM 2008. LNCS, Springer-Verlag (2008) accepted.
3. Massuthe, P., Schmidt, K.: Operating Guidelines – An Automata-Theoretic Foundation for Service-Oriented Architectures. In: QSIC 2005, Melbourne, Australia, IEEE Computer Society Press (2005) 452–457
4. Lohmann, N., Massuthe, P., Wolf, K.: Operating Guidelines for Finite-State Services. In: ICATPN 2007. Volume 4546 of LNCS., Springer-Verlag (2007) 321–341
5. Massuthe, P., Reisig, W., Schmidt, K.: An Operating Guideline Approach to the SOA. AMCT **1**(3) (2005) 35–43
6. Kindler, E.: A compositional partial order semantics for Petri net components. In: ICATPN 1997. Volume 1248 of LNCS., Springer-Verlag (1997) 235–252
7. Stahl, C., Massuthe, P., Bretschneider, J.: Deciding Substitutability of Services with Operating Guidelines. Technical Report 222, Humboldt-Universität zu Berlin, Germany (2008) accepted for a journal.
8. Gierds, C., Mooij, A.J., Wolf, K.: Specifying and generating behavioral service adapter based on transformation rules. Technical Report CS-02-08, Universität Rostock, Germany (2008) submitted to a conference.
9. Papazoglou, M.: Web Services: Principles and Technology. Pearson - Prentice Hall, Essex (2007)
10. Aalst, W.M.P.v.d., Massuthe, P., Stahl, C., Wolf, K.: Multiparty Contracts: Agreeing and Implementing Interorganizational Processes. Technical Report 213, Humboldt-Universität zu Berlin, Germany (2007) submitted to a journal.
11. Schmidt, K.: LoLA: A Low Level Analyser. In: ICATPN 2000. Volume 1825 of LNCS., Springer-Verlag (2000) 465–474
12. Lohmann, N.: A feature-complete Petri net semantics for WS-BPEL 2.0. In: WS-FM 2007. Volume 4937 of LNCS., Brisbane, Australia, Springer-Verlag (2008) 77–91