Process Querying in Apromore

Artem Polyvyanyy[1], Luigi Corno[2], Raffaele Conforti[1], Simon Raboczi[1], Marcello La Rosa[1,3], and Giancarlo Fortino[2]

[1] Queensland University of Technology, Brisbane, Australia
[2] University of Calabria, Cosenza, Italy
[3] NICTA Queensland Lab, Brisbane, Australia
{artem.polyvyanyy;raffaele.conforti;m.larosa}@qut.edu.au,
cornoluigi1989@gmail.com, raboczi@gmail.com, giancarlo.fortino@unical.it

Abstract. This paper demonstrates the integration and usage of Process Query Language (PQL), a special-purpose programming language for querying large collections of process models based on process model behavior, in the Apromore open-source process model repository. The resulting environment provides a unique user experience when carrying out process model querying tasks. The tool is useful for researchers and practitioners working with large process model collections, and specifically for those with an interest in model retrieval tasks as part of process compliance, process redesign and process standardization initiatives.

1 Introduction

Process querying addresses the problem of automatically *retrieving* process models from collections thereof on the basis of user-defined *queries*. Process querying can be used to tackle problems of process compliance, reuse, redesign, and standardization [1].

In this paper, we demonstrate a process querying environment that resulted from integrating Process Query Language (PQL) [2] into the Apromore process model repository [3]. PQL is a programming language based upon temporal logic with an intuitive SQL-like syntax for the specification of queries. The semantics of PQL queries is grounded in *process model behavior*. The intent of a PQL query is to retrieve process models from a collection of models based on the arrangements of activities and/or events in the process instances that these models describe. A screencast that demonstrates the environment is available at `https://youtu.be/S_U6frTWd3M`.

In the remainder of this paper, we provide an overview of PQL and its implementation, present the Apromore process model repository, discuss the integration of PQL into Apromore, and demonstrate the use of PQL in Apromore using a typical process querying scenario in the context of a process model collection taken from industry.

2 Querying Process Model Collections

This section discusses PQL (Section 2.1) and its integration into Apromore (Section 2.2).

2.1 Process Query Language

PQL is a special-purpose programming language for managing collections of process models based on information about process instances that these models describe. PQL

programs are called *queries*. PQL is a declarative language that is based upon temporal logic. Temporal logic is an extension of traditional propositional logic with operators that refer to the behavior of systems over time. These behavioral operators, called *predicates* in PQL, provide PQL with a mathematically precise means for expressing properties about the relation between activities and events in process instances.

The abstract syntax of PQL is an extended version of the abstract syntax of "A Process-model Query Language" (APQL) [4]. PQL introduces its own semantics which is based on the behavioral relations of the 4C spectrum and the label unification principle for disambiguation of activity and event names/labels [5]. The 4C spectrum systematizes four fundamental relations that are commonly used to describe the behavior of concurrent systems: co-occurrence, conflict, causality, and concurrency. The systematization is performed by quantifying and, thus, measuring occurrences of activities and events in process instances. The choice of the 4C relations to be included in the first edition of the PQL language is due to the results of interviews with process analysts. The reader is referred to [2] for a detailed discussion on the design of the PQL language.

The concrete syntax of PQL is inspired by SQL—a programming language for managing data stored in relational database management systems [6]. The rationale behind this design decision is threefold: (i) PQL and SQL serve the same overarching purpose—retrieval of information. (ii) SQL is a widely used standard that is well-recognized by technical specialists. (iii) Finally, the SQL syntax was often recommended for PQL by the interviewed process analysts [2].

The PQL language is implemented in the PQL tool, which is publicly available under the open-source GNU Lesser General Public License.[1]

To give the reader a better taste of PQL, below we provide two compliance rules taken from Table 10.3 in [7] and the corresponding PQL queries that formulate search intents to retrieve all process models that *violate* these rules.

c1. Before goods can be produced, components have to be bought.

```
SELECT * FROM * WHERE
CanOccur("produce goods") AND
(CanConflict("produce goods","buy components") OR NOT
TotalCausal("buy components","produce goods"));
```

c2. If an order is confirmed, it must have been received and processed before. Furthermore, once confirmed an order must not be declined afterwards. Conversely, for a declined order no confirmation is possible any longer.

```
x = {"receive order","process order"};
SELECT * FROM * WHERE
(CanOccur("confirm order") AND
(CanConflict("confirm order",x,ALL)
OR NOT TotalCausal(x,"confirm order",ALL)))
OR CanCooccur("confirm order","decline order");
```

The execution speed of PQL queries can be improved by the use of *indexes*—special data structures that can improve the speed of computations of behavioral relations at the cost of time for their construction and space for their storage [1]. The PQL tool uses *model checking* techniques to index behavioral relations. These are automated techniques that given a finite-state model of a system (e.g., a process model) and a formal property (e.g.,

[1] `https://github.com/processquerying/PQL.git`

a temporal logic formula) systematically check whether this property holds for (a given state in) the model [8].

2.2 PQL in Apromore

Apromore is an open-source process model repository [3]. Its source code is available under the GNU Lesser General Public License.[2] Its public version can be accessed at `http://apromore.qut.edu.au`. Apromore's basic features include model import and export, version control and modeling support for a variety of languages, e.g., BPMN, eEPC, YAWL, and Petri nets. In addition, Apromore provides a range of advanced features, e.g., identification of similar process models on the basis of their structural characteristics, merging similar process models into a consolidated model, and comparison of process models on the basis of their behavior. Apromore uses the latest Java J2EE technology, including Spring, OSGi, ZK and Maven.

The integration of PQL into Apromore was carried out in two phases. The first phase focused on embedding PQL into Apromore. As a result, PQL can access all models in the Apromore repository and detect the creation, modification, and deletion of models in order to update its index of behavioral relations. The second phase covered the implementation of a GUI for specifying and issuing queries. This GUI offers a set of features aimed at improving the user experience during the creation of queries. Two features worth mentioning are (i) autocompletion capable of suggesting standard PQL keywords, like `SELECT`, `AND`, `ANY`, or `TotalConcurrent`, and candidate activity/event names, and (ii) drag-and-drop for selecting process models and repository folders to be considered as part of the `FROM` section of a PQL query; for example, "`SELECT id, author FROM "./myModels";`" query requests to find identifiers and authors of all process models in the `"./myModels"` folder of the repository.

3 Process Querying Scenario

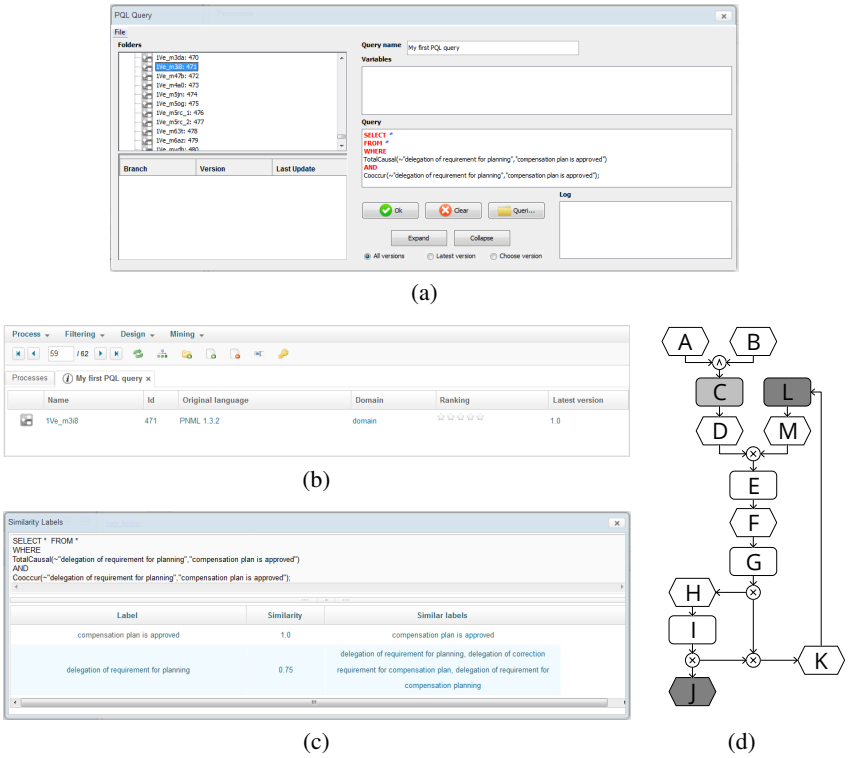
In this section, we demonstrate a process querying scenario of retrieving models that satisfy a given PQL query. The retrieval is performed over a process model repository taken from industry, viz. the SAP Reference Model [9].

Every (version of) process model in Apromore gets indexed by the PQL tool. On average, it takes 51 seconds to index a model from the SAP Reference Model.[3] Once a model is indexed, it can be retrieved as a response to a PQL query.

Process querying in Apromore starts by a user specifying a PQL query. This can be accomplished using the PQL query editor. One can invoke the editor by selecting the "PQL filter" option in the menu of the Apromore's repository view. A screenshot of the PQL query editor is shown in Fig. 1(a). The editor contains a text field for entering variable names and values, i.e., symbolic names as well as sets of activities associated with these names to be used in PQL macros (see query c2 in Section 2.1), a text field for entering PQL queries, the folder tree view control for selecting repository folders and/or (versions of) process models to be used in the `FROM` section of a PQL query, the

[2] `https://github.com/apromore/ApromoreCode.git`

[3] The experiments on indexing times were performed on a laptop with Intel Core i7-3667U CPU 2.00GHz, 8GB of memory, running Windows 7 and Sun JVM 1.7.



(a) (b) (c) (d)

Fig. 1. (a) The PQL query editor, (b) PQL retrieval results, (c) PQL query info, and (d) an EPC with C = "Delegation of requirement for compensation planning", J = "Compensation plan is approved", and L = "Delegation of correction requirement for compensation plan".

PQL syntax error notification box, menu items to call the routines for loading/saving PQL queries from/to files, a text field for specifying names of PQL queries, etc.

After entering a PQL query, pressing the "Ok" button, and passing the syntax check, the PQL tool matches the query against the models specified in its `FROM` section. Once the matching step is finished, the retrieved models, i.e., those process models that describe instances that satisfy the `WHERE` clause of the query, get listed in a fresh tab of the Apromore's repository view, cf. Fig. 1(b). Consider the query given below.

```
SELECT * FROM * WHERE TotalCausal(L,J);
```

This query retrieves one model (listed in Fig. 1(b) and shown in Fig. 1(d)) from the SAP Reference Model.[4] The `TotalCausal(L,J)` predicate requests to retrieve all process models where for every instance in which tasks `L` and `J` both occur, it holds that every occurrence of task `L` precedes every occurrence of task `J`. Indeed, this property holds for the model in Fig. 1(d), cf. EPC function `L` and EPC event `J` highlighted with dark grey background in the figure. Note that the above query does not require tasks `L` and `J` to co-occur in every instance of a matching model. Consider the next query.

```
SELECT * FROM * WHERE TotalCausal(L,J) AND Cooccur(L,J);
```

[4] Note the use of short task names in Fig. 1; full names are given in the caption of the figure.

In this query, the `Cooccur(L,J)` predicate additionally requests to retrieve only those models where it holds that if task `L` occurs in some instance then task `J` occurs in the same instance, and vice versa. Clearly, the above query retrieves no models.

Process model repositories often suffer from inconsistent usage of names/labels. Semantically similar tasks may 'wear' syntactically different names. For these situations, the PQL tool supports semantic name disambiguation.

```
SELECT * FROM * WHERE TotalCausal(~X,J) AND Cooccur(~X,J);
```

Let the short label `X` in the above query refer to the full name "Delegation of requirement for planning". Note that this name is not used in any process model of the SAP Reference Model. The tilde symbol ("~") in the query suggests that in place of `X` the PQL tool can use other labels that are similar with `X` (up to the predefined similarity threshold); refer to [10] for a technique on discovery of similar activity labels. Based on the Levenshtein distance between two sequences of characters (one can configure the PQL tool to use other metrics for string similarity), the PQL tool suggests labels `C` (highlighted with light grey background in Fig. 1(d)) and `L` as being similar to `X`. Consequently, the tool treats labels `C` and `L` as one label. This allows the tool to retrieve the model in Fig. 1(d) again as a response to the last mentioned PQL query. Indeed, `J` always co-occurs either with `C` or `L`, and `C` and `L` together are in the total causal relation with `J`. The above described effect is implemented using the label unification principle proposed in [5].

Finally, the user can review information on every issued PQL query using the PQL query information box. The information box can be invoked by double-clicking the info icon next to the title of the query result tab in the Apromore's repository view. The PQL query information box contains the PQL query, information on all the names/labels used in the query, and all the discovered similar labels, as shown in Fig. 1(c).

References

1. Polyvyanyy, A., La Rosa, M., ter Hofstede, A.H.M.: Indexing and efficient instance-based retrieval of process models using untanglings. In: CAiSE. Vol. 8484 of LNCS., Springer (2014)
2. Polyvyanyy, A., ter Hofstede, A.H.M., La Rosa, M., Ouyang, C.: Process Query Language: Design, implementation and evaluation. BPM Center Report BPM-15-06, BPMcenter.org (2015)
3. La Rosa, M., Reijers, H.A., van der Aalst, W.M.P., Dijkman, R.M., Mendling, J., Dumas, M., García-Bañuelos, L.: APROMORE: An advanced process model repository. ESWA **38**(6) (2011)
4. ter Hofstede, A.H.M., Ouyang, C., La Rosa, M., Song, L., Wang, J., Polyvyanyy, A.: APQL: A process-model query language. In: AP-BPM. Vol. 159 of LNCS., Springer (2013)
5. Polyvyanyy, A., Weidlich, M., Conforti, R., La Rosa, M., ter Hofstede, A.H.M.: The 4C spectrum of fundamental behavioral relations for concurrent systems. In: PETRI NETS. Vol. 8489 of LNCS., Springer (2014)
6. Date, C., Darwen, H.: A Guide to the SQL Standard: A User's Guide to the Standard Database Language SQL. Addison-Wesley (1997)
7. Reichert, M., Weber, B.: Enabling Flexibility in Process-Aware Information Systems: Challenges, Methods, Technologies. Springer (2012)
8. Baier, C., Katoen, J.: Principles of Model Checking. MIT Press (2008)
9. Curran, T., Keller, G., Ladd, A.: SAP R/3 Business Blueprint: Understanding the Business Process Reference Model. Prentice-Hall, Inc., Upper Saddle River, NJ, USA (1998)
10. Awad, A., Polyvyanyy, A., Weske, M.: Semantic querying of business process models. In: EDOC, IEEE Computer Society (2008)