

The QBFEVAL web portal

Massimo Narizzano, Luca Pulina, and Armando Tacchella *

Laboratory of Systems and Technologies for Automated Reasoning (STAR-Lab)
DIST, Università di Genova, Viale Causa, 13 – 16145 Genova, Italy
{mox,pulina,tac}@dist.unige.it

Abstract. In this paper we describe the QBFEVAL web portal, an on-line resource supporting the participants and the organizers of the yearly evaluation of QBF solvers and instances.

1 Introduction

The implementation of effective automated reasoning tools for deciding the satisfiability of Quantified Boolean Formulas (QBFs) is attracting increasing attention, e.g., in formal verification, planning, and reasoning about knowledge (see, e.g., [1] for relevant references). In this context, the yearly evaluation of QBF solvers and instances [1] has been established with the aim of assessing the advancements in the field of QBF reasoning. The QBFEVAL web portal, integrated into QBFLIB [2], is motivated by the need of organizing the increasing amount of data produced by the evaluations and making it available for the community perusal.

Currently QBFEVAL automates several tasks, ranging from the submission of solvers and instances, to the generation of hyper-textual reports describing different views about the QBF evaluations. To offer these features in a flexible and scalable way, we implemented the portal on top of a three-tier architecture [3] using web services [4] to connect underlying components distributed across different hardware platforms. Although QBFEVAL is not an automated reasoning system per se, we believe that it provides an essential tool to improve the state of the art in QBF research and applications. From this point of view, the key feature of QBFEVAL is its extensive support to the manual extraction of information.

QBFEVAL is available for on-line browsing at www.qbflib.org/qbfeval, and the source code of the portal is downloadable from QBFLIB.

2 Architecture and implementation

The architecture of QBFEVAL is based on the three-tier paradigm [3], whereby three separate software layers provide user interface, process logic, and data manipulation, respectively. Figure 1 presents an overview of the hardware/software

* The authors wish to thank the Italian Ministry of University and Research (MIUR) for its financial support, and the anonymous reviewers who helped to improve the original manuscript.

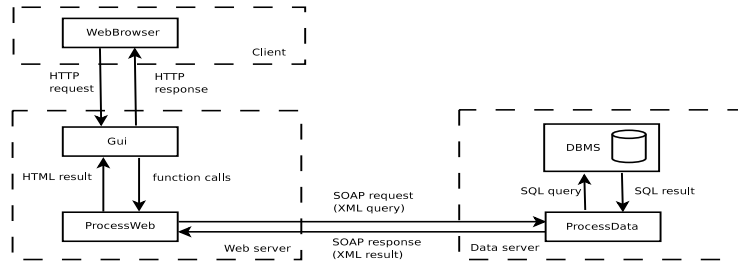


Fig. 1. QBFEVAL architecture

architecture using the following notation: dashed-outline boxes denote hardware components, solid-outline boxes denote relevant software modules, and the arrows denote communications occurring between the modules. With reference to Figure 1, QBFEVAL involves three pieces of hardware:

Client : a user's PC, i.e., a thin client containing a **WebBrowser**.

Web server : the front-end server, containing the **Gui** and a part of the process logic (**ProcessWeb**).

Data server : the back-end server of QBFEVAL, containing the bulk of the process logic (**ProcessData**), the data manipulation functionality (**DBMS**).

The data flow across the modules can be summarized as follows: (i) the user navigating QBFEVAL with her browser originates HTTP requests that the web server satisfies using the content dynamically generated by the scripts of the **Gui** module; (ii) the presentation offered by the **Gui** module is supported by the functions of the process layer, in particular by the set of scripts **ProcessWeb** that resides on the web server; (iii) the **ProcessWeb** module relies on the web services offered by the module **ProcessData** to perform heavy computations and data manipulations: the two process modules communicate using SOAP, whereby each service is negotiated and data is exchanged using XML; (iv) the **ProcessData** scripts are interfaced with the data base through a socket connection whereby SQL queries and results can be exchanged.

The clear cut interfaces between the layers **Gui**, **Process[Web,Data]** and **DBMS** allow us to modify extensively the implementation of each one without hurting the stability of the whole system. Moreover, a distinctive advantage of our implementation based on web services is that, by paying a small increase in complexity, we are able to decouple substantially the part of data presentation (hosted in the web server) and the part of data storage/manipulation (hosted in the data server). This could allow us, e.g., to transparently distribute the data part across several remote servers.

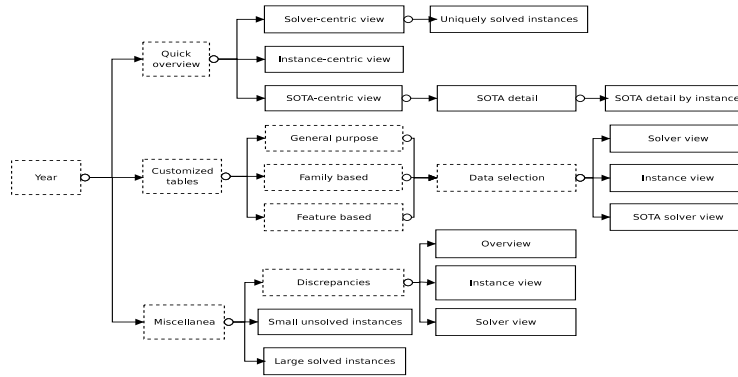


Fig. 2. QBFEVAL structure.

3 Interface

Figure 2 shows a hierarchical map of the QBFEVAL interface using the following notation: solid-outline boxes stand for pages containing tables with specific views of QBFEVAL data, dotted-outline boxes represent menus and forms by which the user can interact with QBFEVAL, and the arrows denote hyper-textual links between the components. With reference to Figure 2:

Quick Overview The tables in this section are available according to a categorization of the instances in two classes: probabilistic structure and fixed structure (see [1] for more details). With reference to Figure 2, for each class, we report tables focusing on the relative performance of the solvers (Solver-centric view) and the relative hardness of the instances (Instance-centric view). In the SOTA solver section (SOTA-centric view), further details about the contribution of each solver to the SOTA solver can be dug out (SOTA detail), going from a general overview, down to the data related to each instance (SOTA detail by instance).

Customized tables The tables of this section are generated dynamically according to the user’s preferences. With reference to Figure 2, the **General purpose** form allows the user to extract specific information grouping instances by several parameters. The **Family based** form offers a specialized selection centered around family grouping. Finally, the **Feature based** form is a (still experimental) section that allows the user to extract data using features, i.e., attributes of the instances. The combination of parameters that can be specified using the **Data selection** wizard allows the user to obtain solver-centric, instance-centric and SOTA solver views (Solver view, Instance view, SOTA solver view).

Miscellanea This section groups some views that should be particularly useful for developers. In particular, the **Discrepancies** form enables to obtain data

about those instances where at least two solvers reported a different satisfiability result; the data can be arranged to give the global picture ([Overview](#)) or to obtain specific information about instances ([Overview by instance](#)) or about solvers ([Overview by solver](#)). The [Small unsolved instances](#) and the [Large unsolved instances](#) tables report data about small instances that could not be solved and about large instances that have been solved, respectively.

4 Conclusions

As far as we know, QBFEVAL is the first system in its genre to be presented in the context of automated reasoning for QBFs. QBFEVAL builds on and improves QBFLIB [2] which aims to become to the QBF community what TPTP [5] is for the automated theorem proving community. Although the work of QBFEVAL has been inspired by SATEX [6] and the on-line reporting of the SAT competition [7], we believe that our portal differs substantially from the aforementioned contributions. SATEX is a framework for continuous experimentation in SAT, i.e., developers can submit SAT solvers and instances at any time, and such submissions enter the evaluation process which is constantly summarized on the web pages. On the other hand, QBFEVAL offers a historical series of in-depth snapshots about the state of the art in QBF considering specific events; therefore, even if QBFEVAL replicates some of the underlying machinery of SATEX, the kind of reporting available in QBFEVAL is not meant to be available in SATEX. As for the on-line reporting of the SAT competition, we believe that our portal offers more customizable report generation tools, and more extensive support of the event. It is also important to notice that QBFEVAL is the only such portal featuring a complete source-code distribution available for free download. Although objectivity and reproducibility of the results made available through QBFEVAL are not its main focus, the open source distribution allows other researchers to peruse our scripts, customize them, and identify possible bugs that we could have overlooked, thus contributing to the overall robustness of the platform.

References

1. M. Narizzano, L. Pulina, and A. Tacchella. The third QBF solvers comparative evaluation. *Journal on Satisfiability, Boolean Modeling and Computation*, 2:145–164, 2006. Available online at <http://jsat.ewi.tudelft.nl/> [2006-6-2].
2. E. Giunchiglia, M. Narizzano, and A. Tacchella. Quantified Boolean Formulas satisfiability library (QBFLIB), 2001. www.qbflib.org.
3. W. W. Eckerson. Three Tier Client/Server Architecture: Achieving Scalability, Performance, and Efficiency in Client Server Applications. *Open Information Systems*, 3, 1995.
4. D. Booth, H. Haas, F. McCabe, E. Newcomer, M. Champion, C. Ferris, and D. Orchard. Web Services Architecture, February 2004. W3C Working Group Note.
5. G. Sutcliffe and C. Suttner. The tptp problem library for automated theorem proving. Available from <http://www.cs.miami.edu/~tptp/> [2006-6-19].
6. L. Simon and P. Chatalic. SATEX: a Web-based Framework for SAT Experimentation. In *Workshop on Theory and Applications of Satisfiability Testing*, 2001.
7. D. Le Berre and L. Simon. The SAT Competition. <http://www.satcompetition.org> [2006-6-2].