COMBAS: A Semantic-Based Model Checking Framework

Eduardo González-López de Murillas, Javier Fabra, Pedro Álvarez, Joaquín Ezpeleta
Aragón Institute of Engineering Research (I3A)
Department of Computer Science and Systems Engineering
University of Zaragoza, Spain
Email: {edugonza, jfabra, alvaper, ezpeleta}@unizar.es

***Abstract*—The introduction of semantic aspects in scientific workflows is a powerful approach that allows the analysis of the workflow prior to its development and deployment. In this paper, the COMBAS framework for the semantic-based model checking processing is presented. COMBAS integrates the required languages and tools and implements its own algorithms in order to allow the verification of properties on a model specified with the U-RDF-PN formalism, a high-level Petri net-based formalism, which introduces parametric semantic annotations in the model. COMBAS facilitates the generation of temporal logic formulae to express the properties that are going to be verified in the model as well as it provides system designers with an RDF and CTL adapted environment to browse and review the results. The suitability of the proposed framework is demonstrated by means of its application to the analysis of the EBI InterProScan scientific workflow.**

***Keywords*-Semantic Annotated Processes; RDF; SMT; High-level Petri Nets.**

I. INTRODUCTION

Scientific computing applications are being used in a broad spectrum of domains related to science and human life such as geography, biology, or the public sector, for instance. Scientific workflows are a special type of workflows, which often underlies many large-scale complex e-science applications, such as climate modelling, structural biology and chemistry, medical surgery or disaster recovery simulation, among others. Scientific workflows have been progressively improved by means of the introduction of new paradigms and technologies in order to achieve more complex challenges. Once the deployment and execution of such workflows has been carried out, the next challenge is focused on the incorporation of semantic Web techniques [1], [2] in order to analyse their behaviour.

With the development of semantic technologies, the incorporation of semantic aspects allows scientists to more efficiently browse, query, integrate and compose relevant cross discipline datasets and services [1]. Scientific workflow executions are expensive in the use of execution resources, as well as a time consuming activity. For this reason, it is of special interest to dispose of tools and techniques making possible the analysis of the workflow behaviour prior to its execution. The aim of such analysis would be to ensure a good behaviour (It is a waste of time and money to realize after 20 hours executing a task that the output has not the correct information to feed the next task!) as well as facilitating having a very efficient (from the budget and time points of view) resource utilization. The result of the analysis should allow predicting the quality of the results and also identifying those parameters suitable to get the expected outcome.

The introduction of semantic aspects in workflows requires new models and analysis techniques, able to deal with such semantic aspects, to be considered. With this respect, in this paper, the COMBAS framework is presented. COMBAS seeks at helping system designers (scientists and business process developers, among others) in the task of validation and property analysis of workflows that include semantic aspects in the task specification. The analysis process should be as follows. First, the designer models the workflow by means of a Petri net (the advantages of using Petri nets for this purpose has been widely discussed in the literature [3], [4]). The transitions of the Petri net model would correspond to system actions changing the workflow state (mainly, the execution of tasks the workflow requires). Semantic information is attached to the transitions, corresponding to the formal specification of the task and including the description of the input and output parameters. Since some of the outputs could correspond to data received or computed at runtime, formal symbolic parameters (as usual in procedure specification) will be used to represent such data. On the other hand, preconditions and postconditions for tasks should be considered and described as expressions involving the parameters in the task specification.

The U-RDF-PN formalism (Unary RDF Annotated Petri Net formalism) [5], is a subclass of Petri nets defined to consider this class of systems. Semantic annotations are specified using the RDF (Resource Description Framework) [6]. With respect to pre- and post-conditions, SMT (Satisfiability Modulo Theories) [7] solvers represent a very useful tool. They are able to work with logical predicates and solve the decision problem when using background theories. Therefore, they are able to determine the satisfiability of a collection of logical predicates, according to a certain combination of such theories. Finally, it is necessary then to have a tool, which allows us to verify the satisfiability of certain predicates, and also a language to express such predicates. The analysis of the workflow is carried out

 ISBN: 978-1-61208-237-0

using model checking techniques, which define the way to verify the model through its reachability graph by means of queries expressed in terms of CTL (Computation Tree Logic) predicates.

The remainder of this paper is organized as follows. The design and implementation details of COMBAS are first depicted in Section II, and its application to a real problem in the scientific computing area is conducted in Section III. Section IV briefly introduces the related standards and tools for model checking analysis. Finally, Section V concludes the paper and addresses future directions of the work.

II. COMBAS: A SEMANTIC-BASED MODEL CHECKER

The architectural view of the COMBAS framework is depicted in Figure 1. COMBAS integrates a set of tools and techniques to cover the full cycle for the model checking-based analysis of semantically annotated workflows: from the generation of U-RDF-PN models, the corresponding reachability graph and its Kripke-structure, the creation and edition of queries and CTL formulae, the execution of the model checking process and, finally, the results browsing and reviewing. Let us describe the comprehensive steps a system designer must perform in order to achieve the verification process using the framework:

1) First, the scientist must design the workflow or the process using a Petri net. For that purpose, the graphical tool Renew [8] can be used. The resulting model will be then exported to the PNML standard (Petri Net Markup Language, ISO/IEC 15909), an XML based description for Petri nets. The initial marking of the model as well as the model itself must be semantically annotatedby means of RDF, building a set of XML specification files (RDF graphs and patterns). A graphical assistant will help in this process.
2) Now some steps are taken in a transparent way for the user, who can review the results at each stage. The reachability graph generator uses the previous files as an input and generates two outputs: the reachability graph (RG), which is stored in the RDF Triple Store, and a set of XML files that contain the relation between states and their markings and a collection of RDF/XML files representing the RDFGs (RDF Graph) corresponding to the marking. The graphical representation of the reachability graph, which can be viewed with the tool Graphviz, is also generated at this step. Both the results from the previous processing and the reachability graph can be browsed by means of a Web viewer.
3) An annotated Kripke structure is constructed from this RG (RDF-KS). The reachability graph of a U-RDF-PN system can be easily transformed into an RDF Annotated Kripke Structure [5].
4) Now, it is necessary to create the CTL formula to be verified with the model checker. The designer can build this formula by means of a user-oriented interface provided by the Web editor, which will export it as an XML file compliant with the corresponding schema.
5) Finally, the model checker uses the CTL formula along with the RDF Kripke structure (RDF-KS) to compute and generate the output. This output consists of a collection of XML files that represent and relate states on the RG validated with the corresponding excerpts from the CTL formula. Both input files and results from the analysis at this stage can be viewed and browsed through the Web interface in a very convenient and intuitive manner.

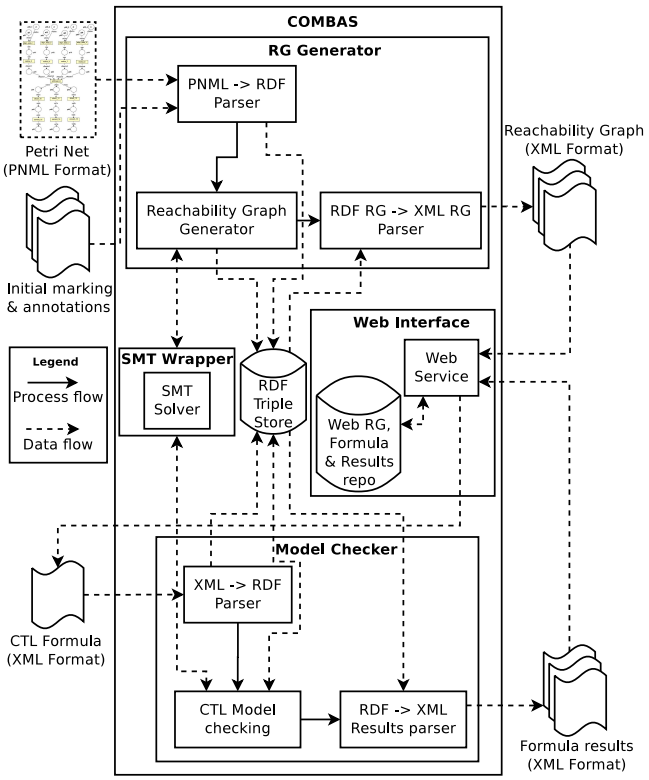

Figure 1. Architecture of the COMBAS model-checking framework.

All components in COMBAS expose an easy, flexible and usable interface, and the complexity of the graph generation, storage in the semantic triple store and verification processes are hidden from the user's perspective. During the model checking process, an RDF database (a triple store) is used. The COMBAS framework allows using several different RDF databases, such as the *AllegroGraph RDFStore* database or the *Virtuoso RDF Store*, for instance. The only requirement of a candidate RDF store is that it must allow to be accessed through a SPARQL interface. In this work the Virtuoso RDF Store has been used. As a result from the processing, the truth about the verification of the formula is obtained. Moreover, a graph depicting the

 ISBN: 978-1-61208-237-0

reachability graph states can be browsed using a graphical Web-based enabled interface. Doing so, it is possible to find the specific situations in which a predicate violates some wanted condition, having a better insight of the workflow behaviour, and thus facilitating the way to improve the workflow.

Internally, the generation of the reachability graph is based on the classical algorithm used for computation of the reachability graph in Petri nets and making the necessary modifications to adapt to the semantic nature of annotations [5]. Then, the implementation of our model checker is based on an adaptation of the labelling algorithm proposed in [9]. The inputs are an RDF-KS and an RDF CTL formula. Both inputs are stored into the RDF database following the corresponding RDF schemas. Basically, the algorithm computes the set of states of the input system $\mathcal{M}$ that satisfy the given CTL formula ϕ. This process consists of three steps. Initially, the formula ϕ is translated into an equivalent formula in terms of the connectives AF, EU, EX, $\top$, $\wedge$ and $\neg$. The equivalence rules applied are defined as part of the labelling algorithm. Secondly, the states of $\mathcal{M}$ satisfying subformulas of ϕ (ψ) are labelled, starting with the smallest subformulas and finishing with the original formula. Finally, the algorithm returns the states labelled with ϕ.

A. Reachability graph generation

The reachability graph generation process needs a valid model to be used as an input: an U-RDF-PN class Petri net, with an initial marking corresponding to the system's initial state [5]. The semantic annotations can be linked to three different elements of the net: *arcs*, where it is possible to find RDF patterns; *transitions*, where guards can be defined; and *places*, where tokens are stored as RDF graphs. The generated reachability graph is stored also as RDF triples, according to the ontology depicted in Figure 2.

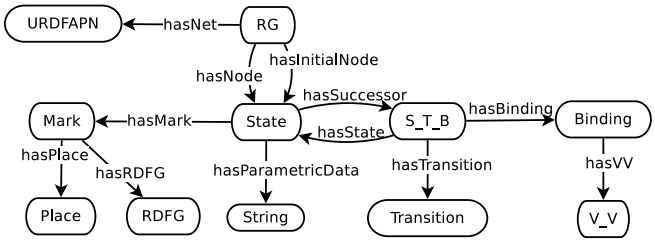

Figure 2. Ontology for the Parametric Reachability Graph.

In this paper, we are considering the parametric extension over U-RDF-PN as defined in [5]. The concept is very simple: to add parameters to the model, instead of static values. This way, representing general cases of a process or workflow consists of the addition of logical propositions to the information, which annotates each state in the reachability graph. Also the guards in transitions must be specified making use of such annotations.

Let us now briefly describe the main differences of the parametric approach with respect to the ordinary model checking approach presented in [5] regarding to the reachability graph generation. First of all, a parametric U-RDF-PN, which is an ordinary U-RDF-PN annotated with parametric statements in some of its transitions (as guards), places (as initial marking) and arcs (as part of the RDFG-Patterns), is used as an input. It has been necessary to implement a wrapper in order to make use of an SMT solver, crucial when working with logic and parameters. This way, the generation process is similar to the one of an ordinary U-RDF-PN, except for the following considerations:

1) The initial state is formed not only by the initial marking of every place in the net, but also by the parametric initial marking of each state. In case there is not such marking, the logical statement $true$ will be considered as initial parametric marking.
2) For every transition that has a parametric guard, the validity of such guard must be verified. This is performed making use of a SMT solver to check if the conjunction of the logical statements in the guard and the statements of the current state are satisfiable.
3) When generating a new state for the RG, its parametric marking is formed by the conjunction of the one of the parent state and the guard of the triggered transition.
4) Also when introducing a new state in the store, it is necessary to check if it is unique. To do so, the generator must compare the semantic part, and also the parametric part. When comparing the last one, an SMT solver is used to check the equivalence of both logical statements (P and Q), observing the satisfiability of the formula $P \rightarrow Q \wedge Q \rightarrow P$.

B. Edition and visualization service

The basic features of this Web user interface are: 1) reachability graph visualization; 2) creation and edition of CTL formulae used by the Model Checker; and 3) review of Model checking results. The initial aim was to integrate all this functionalities in a single interface. Let us now detail the components of this interface.

RG visualization. As previously mentioned, the output of the RG generator consists of a main XML file, which describes the RG graph structure and a collection of XML files containing the RDF graphs that mark every state in the RG. In case we are dealing with a parametric model, also several *SMT-Lib* files will be generated. These files contain the parametric predicates corresponding to each state. The generated graph will contain a big amount of states, around hundreds in case it is a simple model, but it will raise exponentially as the complexity increases. Therefore, the review of the graph can be a tedious task. This was one of the principal aims to develop a viewer to represent the graph as a diagram, so it makes easier to check the marking of every state.

 ISBN: 978-1-61208-237-0

The developed application makes possible to select a certain reachability graph, and visualize its structure in a graphical way, allowing to consult the marking in an easy and intuitive way.

Formula edition. Due to the purpose of this project, it is also important to provide an effective tool to edit CTL formulae. It represents an important benefit to the verification process, avoiding typing mistakes, and improving user experience because it reduces the learning curve from the user point of view and increases the formula creation and editing speed. That is why it is important to design a tool as intuitive as possible. It seemed right to develop a formulae constructor, which could provide a list of components and a mechanism to include them in the formula in an interactive way. Also, we thought it would be interesting to include a verification phase to make sure that all the input is well constructed, avoiding syntactic errors.

Among the features of this part, we find interesting the ability to create/edit formulae, visualize them in a graphical environment and the interactive creating using intuitive user interfaces.

Model Checker results review. A very important stage in the model checking process is the result review. In this stage, the user needs to verify the results and analyse them in order to identify the errors, if any, in the model. Therefore, it is crucial to provide the tools to manage the output of our Model Checker.

The developed interface makes possible to select a graph and inspect the execution result of a certain formula, using the reachability graph visualization, and checking the marking and satisfiability of the formula components in every state in the RG.

C. Model checking process

The ontology represented in Figure 3 is used to specify the input CTL formula for the model cheker. It may contain any of the unary CTL operators $AF, EF, AX, EX, AG, EG, NOT$, or binary ones AND, OR, AU, EU, IMP, or any of the terminal nodes $TRUE, FALSE, RDFG, RDFGP, SMT-EXP$ being RDFG a RDF-Graph, RDFGP a RDF-Graph Pattern, and SMT-EXP a logical statement in SMT-Lib format.

The other input of our model checker is the reachability graph, which generation process has been previously explained. Because this RG may contain parametric data, we still need to make use of an SMT solver. So, the same SMT wrapper mentioned in II-A is required.

The verification process is similar to the one used with ordinary U-RDF-PN and CTL formulae as described in [5]. The main difference lies in the terminal node $SMT-EXP$, which is represented by a logical statement in SMT-Lib format. In order to know if a state satisfies such statement, we need to make use of the SMT wrapper. Therefore, the parametric marking of such state must be compared to the parametric statement in the formula. Being P the parametric data stored in the state, and Q the content of the $SMT-EXP$ node in the formula, the logic expression $P \vee Q$ is used to check the validity of the state parametric marking (P) and the formula parametric expression (Q). This way, we will know if both logic statements are compatible or contradictory.

III. Analysis of the EBI InterProScan workflow

In order to show the suitability of the COMBAS framework, in this section the InterProScan workflow is going to be analysed. This workflow relies on the use of the EBIs WSInterProScan service [10], an its workflow can be checked out at the Myexperiment.org community [11]. The workflow receives as an input the protein sequence to be processed, a user email address for notification purposes and a few more parameters required for the analysis. Starting with this input, a protein sequence is searched inside a set of protein families and domain signature databases integrated in InterPro [12]. As a result, a set of matches are properly formatted and returned. These matches are also annotated with the corresponding InterPro and GO term assignments (for further explanations about these assignments, please refer to the experiment page). In order to execute the `runInterProScan` activity, two different Web services, `runInterProScan1` and `runInterProScan2`, are available in a repository. Both services are able to carry out the protein analysis, which represents the more expensive part of the experiment.

Figure 4 depicts the workflow modelled as a High-Level Petri net using the Renew tool [8] and semantically annotated according to the parametric U-RDF-PN formalism. All the data flowing through the workflow have been semantically annotated using instances based on the Protein Ontology [13]. Nodes corresponding to the original workflow are in dark grey, whereas some additional structures, which have been included in order to provide more generality in the generated reachability graph (representing a higher variety of cases and states to analyse) are in light grey.

Let us now briefly describe the parametric net. The seven annotated places at the top of the net are the main inputs of the workflow. Five of those inputs, which represent the job parameters and are also necessary to configure properly the service *InterProScan*, are grouped in the left side: goterms_default, async_default, crc_default, seqtype_default and email_address. These inputs are required for transition *job_params* to be enabled and then fired. In the right side, two places represent the protein sequence data: sequence_or_ID and input_datatype_default. They are also required to fire the transition *Input_data*. From the morphology of the net, we can observe that all mentioned parameters are necessary for the execution of the service *InterProScan*. When this service is executed, the process enters the bottom half of the net, in which the output is converted to the proper

 ISBN: 978-1-61208-237-0

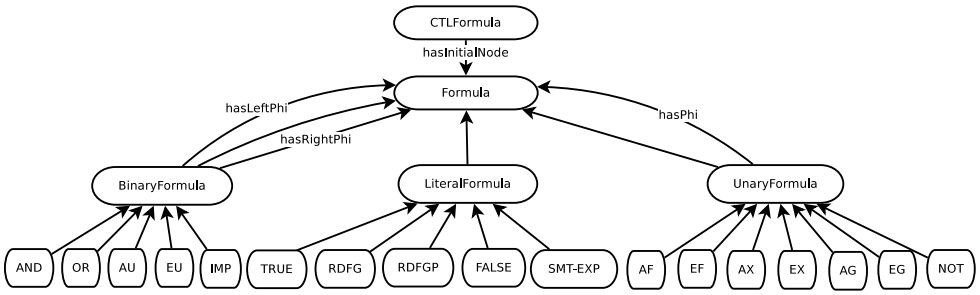

Figure 3. CTL formula ontology with parametric elements.

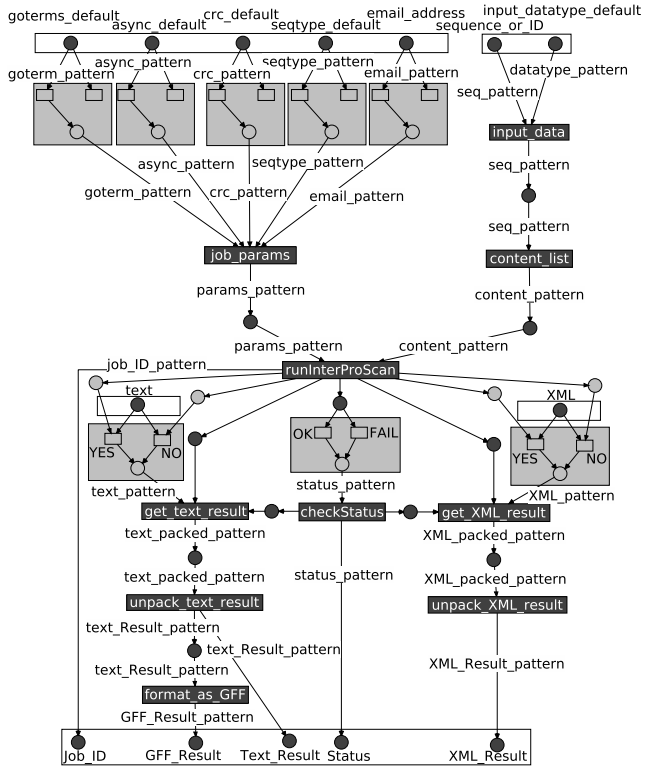

Figure 4. Parametric U-RDF-PN modeling the InterProScan workflow for the analysis of a protein sequence.

format. The are two places in charge of selecting the proper format, and they also represent an input of the workflow: 1) text and 2) XML. These places are semantically annotated, containing, each of them, the reference to a parameter. Below both places, a light grey section has been added. The purpose of this structure is to assign two different values to the mentioned parameters (true or false), so it is possible to analyse the behaviour of the workflow in any situation, using a single reachability graph. The same technique has been used in the top part of the net, with the five parameters described earlier. At the bottom, there are five output places: Job_ID, GFF_Result, Text_Result, XML_Result and Status. These are the places in which the results of the workflow will be stored. They represent the five different output data types we can obtain: the job identifier, the result in GFF, text and XML format, and the status value, respectively.

This model is then processed by the Reachability Graph Generator to compute the graph. The resulting reachability graph has been built in 1343 seconds, and it is composed of 798 states.

A. Properties checking

Due to the size of the generated reachability graph, a manual checking would not be feasible, resulting in a tedious and complex task. This represents the perfect situation to apply the proposed solution to analyse the problem. Let us now design and formulate some different queries over the model in order to verify its correctness. These queries are expressed as formulae in CTL language and have been implemented using the corresponding XML format, which is ready to be processed by COMBAS. The Model Checker is then in charge of verifying the satisfiability of the formula.

The first query to be consider when verifying a workflow could be to satisfy if the process is able, in some situation, to end in a proper way. This will happen if the output places in the net are marked with a resulting graph. Five different output places are distinguished: Job_ID place; Text_Result place; GFF_Result place; Status place and, finally, XML_Result place. These places correspond to the ones located in the bottom of the net presented in Figure 4. Therefore, to verify if those places are reached, it is possible to check if there exists any state in which such places are marked by any graph (the empty graph represents the 'any graph' concept). This query corresponds to the following CTL predicate: $EF(RDFG_{Text\_Result} \vee RDFG_{GFF\_Result} \vee RDFG_{XML\_Result} \vee RDFG_{Status} \vee RDFG_{Job\_ID})$

This formula, when executed in the Model Checker, provides the next output:

```
INFO [main](CombasApp.java) Model satisfies the formula!
INFO [main](CombasApp.java) Output files generated.
INFO [main](CombasApp.java) Checked in 17159 millis
INFO [main](CombasApp.java) Formula: formula_1h9wbb3xwwj7c
INFO [main](CombasApp.java) Model: netId1337458105565_RG
```

This means that the model satisfies the formula and, therefore, it is possible to reach a state where any of the

specified states is marked. The next step would be to verify if the workflow is always able to finish. The corresponding formula is expressed as follows: $AF(RDFG_{Text\_Result} \vee RDFG_{GFF\_Result} \vee RDFG_{XML\_Result} \vee RDFG_{Status} \vee RDFG_{Job\_ID})$

The results from the model checking state that this formula is not satisfied, which means that the model will not always reach any of those places in the net. This can happen, for instance, when the net reaches a deadlock state with no transitions ready to fire.

Due to the morphology of the net, in order to trigger the transition named *runInterProScan* its source places must be marked. These places, which are referred to as *params* and *content_list* places, will be marked in some situation. Otherwise, the first query would not be satisfied (the one asking if it is possible to reach any of the output places). Then, it would be interesting to know if every time these two places are marked the workflow will reach any of the output places. This is the corresponding CTL query: $AG((RDFG_{params} \wedge RDFG_{content}) \rightarrow AF(RDFG_{GFF\_Result} \vee RDFG_{Text\_Result} \vee RDFG_{Status})$. When processed, the Model Checker states that the model satisfies the property.

Another useful query is related to verifying if every time a result is obtained (at least one of the output places is reached), the value of the status result is *true* (so the token in the place *status* has the value *true* for the *status* variable). The mentioned property in CTL is: $AG((RDFG_{Text\_Result} \vee RDFG_{GFF\_Result} \vee RDFG_{XML\_Result}) \rightarrow RDFGP_{Status(true)}$. The predicate satisfies the model, which means that every time we get a result, the status variable is set to *true*.

As shown, the model is very suitable to be analysed by means of CTL queries related to the properties we want to verify. Once the input model and its corresponding annotations have been defined, COMBAS allows to easily perform a model checking process, browse through the reachability graph checking in which states the conditions are not being verified, review the results, etc. No prior knowledge of the model checking algorithm is required for the scientist nor internal or technological details. Just using the interface with the framework allows a set of advanced features and tools for the analysis of complex systems.

IV. Related work

There already exist numerous Model Checking tools, which can be classified according to the language they use to express models and properties. Some of the tools allow us to define properties in Computation Tree Logic and Linear Temporal Logic (CTL and LTL, respectively), although in this paper we will focus on the most widespread alternatives supporting CTL as the language to express properties. BANDERA uses code analysis techniques to verify properties on models defined in Java, while CADENCE SMV uses plain model checking to verify models expressed in Cadence SMV, SMV or Verilog languages [14]. Another example is NuSMV, which supports CTL, LTL and PSL to define properties, in order to do plain model checking on SMV models. APMC uses an approximate probabilistic technique on Reactive Modules, and properties are expressed in PCTL and PLTL languages [15]. PRISM supports CSL in addition to both previous languages, and uses a probabilistic model checking technique on models expressed in a big variety of languages, like PEPA, PRISM language or Plain MC [15]. Between the probabilistic model checkers, we find MCMR, which supports properties defined in CSL, CSRL, PCTL or PRCTL languages, and uses real time and probabilistic model checking on Plain MC models.

Other tools support the language μ-calculus to describe properties, like TAPAs, which also supports CTL on CCSP models. ARC performs plain model checking of CTL* properties on AltaRica models. GEAR is an alternative tool, which accepts μ-calculus properties too, among others like AFMC and CTL [16]. CWB-NC is a similar tool, and uses plain and temporized model checking on CCS, CSP, LOTOS and TCCS models, using AFMC, CTL and GCTL properties. Finally, MCMAS is a plain and epistemic model checker on ISPL models, which verifies CTL and CTLK properties [17].

Regarding the use of SMT solvers, there exists a wide variety solvers to choose from. The main characteristic aspects to choose among them are the collection of supported theories and languages, the programming language they are implemented in and also its portability and reusability features. However, there are other points to take into account, like the activity of the user community, how often new versions are released, and the quality of the documentation. Regarding to these concepts, the list of options is considerably reduced. CVC, OpenSMT and STP represent the most active projects. STP only supports formulas over the theory of bit-vectors and arrays, therefore it does not represent a valid solution for our problem, because the main theory we need support for is linear arithmetic, specifically over reals, integers and booleans. Although OpenSMT is a valid option, we choose CVC instead because it is a more stable and production ready alternative. More precisely, we choose CVC3, which is the stable version of CVC due to the instability of CVC4, which is in beta phase.

With respect to the description of predicates to perform the model checking process, the Satisfiability Modulo Theories Competition (SMT-COMP) is celebrated each year, since 2005, with the purpose of encouraging the development of SMT solvers, and also as an impulse to the adoption of the Satisfiability Modulo Theories Library standard (SMT-LIB) [18]. SMT-LIB is a format designed by the community that tries to unify the description of the background theories and the inputs/outputs for SMT solvers as well as to provide a collection of benchmarks to boost the development of this kind of tools. Therefore,

 ISBN: 978-1-61208-237-0

compared to other formats like CVC language or DIMACS, SMT-LIB represents the most recommendable option to keep compatibility and portability for our predicates.

To sum up, there are a great variety of standards and tools that are involved in the model checking process. However, there are no frameworks that integrate them in a comprehensive way in order to allow scientists to analyse their scientific workflows prior to their deployment and execution, becoming COMBAS a suitable approach for this kind of scenarios.

V. Conclusions and future work

In this work, the COMBAS framework to carry out model checking of semantically annotated processes and workflows has been presented. COMBAS represents a novel approach to add RDF semantic information to the source model as well as to achieve its processing and analysis. This will allow the scientists community to take advantages of the new semantic technologies and also to facilitate sharing workflows and tasks as well as reasoning about the results and behaviours. The framework allows verifying and viewing the intermediate structures and the results by means of a visual environment able to handle RDF and CTL aspects. The use of the Unary RDF Annotated Petri Net formalism (U-RDF-PN) for the modeling of processes has been extended with the addition of parametric aspects, allowing to consider a more flexible and powerful analysis for complex systems. Our proposal represents a novel approach to manage semantic-based computation problems by means of the integration of several model checking related standards and tools, and its suitability has been demonstrated in the analysis of the InterProScan workflow.

The COMBAS framework is currently being extended in order to integrate other standards and tools used in the scientific community. Also, it is being applied to solve some open challenges in the scientific workflow area, and the reachability graph generator is being adapted in order to be executed in grid environments, therefore improving the overall execution costs.

Acknowledgments

This work has been supported by the research project TIN2010-17905, granted by the Spanish Ministry of Science and Innovation, and the regional project DGA-FSE, granted by the European Regional Development Fund (ERDF).

References

[1] C. Berkley, S. Bowers, M. Jones, B. Ludäscher, M. Schildhauer, and J. Tao, "Incorporating Semantics in Scientific Workflow Authoring," in *SSDBM 2005*, 2005, pp. 75–78.

[2] C. Goble, J. Bhagat, S. Aleksejevs, D. Cruickshank, D. Michaelides, D. Newman, M. Borkum, S. Bechhofer, M. Roos, P. Li, and D. De Roure, "myExperiment: a repository and social network for the sharing of bioinformatics workflows," *Nucleic Acids Research*, 2010.

[3] W. M. P. van der Aalst, "The application of Petri nets to workflow management." *Journal of Circuits, Systems, and Computers*, vol. 8, no. 1, pp. 21–66, 1998.

[4] T. Gubala, D. Herezlak, M. Bubak, and M. Malawski, "Semantic composition of scientific workflows based on the Petri nets formalism," in *E-SCIENCE'06*.

[5] M. J. Ibáñez, J. Fabra, P. Álvarez, and J. Ezpeleta, "Model checking analysis of semantically annotated business processes," *Systems, Man, and Cybernetics – Part A: Systems and Humans*, 2012.

[6] P. Hayes, "RDF Semantics," W3C, Tech. Rep., February 2004, http://www.w3.org/TR/rdf-mt/.

[7] C. Barrett, R. Sebastiani, S. Seshia, and C. Tinelli, "Satisfiability modulo theories," *Handbook of Satisfiability*, vol. 4, 2009.

[8] O. Kummer and F. Wienberg, "Renew - the Reference Net Workshop," in *Tool Demonstrations, 21st International Conference on Application and Theory of Petri Nets*, 2000, pp. 87–89.

[9] E. A. Emerson, "Temporal and Modal Logic," *Handbook of Theoretical Computer Science*, vol. Formal Models and Semantics, pp. 995–1072, 1990.

[10] WSInterProScan, available at http://www.ebi.ac.uk /Tools/webservices/services/archive/pfa/wsinterproscan [retrieved: September, 2012].

[11] EBI's WSInterProScan workflow at MyExperiment community, available at http://www.myexperiment.org/workflows/814.html [retrieved: September, 2012].

[12] InterPro protein sequence analysis & classification, available at http://www.ebi.ac.uk/interpro/ [retrieved: September, 2012].

[13] Protein Ontology, available at http://bioportal.bioontology.org/ontologies/1052 [retrieved: September, 2012].

[14] D. Garlan, S. Khersonsky, and J. S. Kim, "Model checking publish-subscribe systems," in *SPIN'03*, 2003, pp. 166–180.

[15] M. Duflot, L. Fribourg, T. Hérault, R. Lassaigne, F. Magniette, S. Messika, S.Peyronnet, and C. Picaronny, "Probabilistic model checking of the csma/cd protocol using prism and apmc," *Electr. Notes Theor. Comput. Sci.*, vol. 128, no. 6, pp. 195–214, 2005.

[16] O. Grumberg and H. Veith, Eds., *25 Years of Model Checking - History, Achievements, Perspectives*, ser. Lecture Notes in Computer Science, vol. 5000. Springer, 2008.

[17] A. Lomuscio, H. Qu, and F. Raimondi, "Mcmas: A model checker for the verification of multi-agent systems," in *CAV*, 2009, pp. 682–688.

[18] SMT-Lib: The Satisfiability Modulo Theories Library, available at http://www.smtlib.org/ [retrieved: September, 2012].

 ISBN: 978-1-61208-237-0