Integrating multiple sources to answer questions in Algebraic Topology.⋆

Jónathan Heras, Vico Pascual, Ana Romero, and Julio Rubio

Departamento de Matemáticas y Computación, Universidad de La Rioja,
Edificio Vives, Luis de Ulloa s/n, E-26004 Logroño (La Rioja, Spain).
{jonathan.heras, vico.pascual, ana.romero, julio.rubio}@unirioja.es

Abstract. We present in this paper an evolution of a tool from a user interface for a concrete Computer Algebra system for Algebraic Topology (the Kenzo system), to a front-end allowing the interoperability among different sources for computation and deduction. The architecture allows the system not only to interface several systems, but also to make them cooperate in shared calculations.

1 Different questions, different sources

When working in Mathematics, usually the researcher uses different sources of information. Typically, he can consult some papers or textbooks, make some computations with a Computer Algebra system, check the results against some known tables or, more rarely, verify some conjectures with a proof assistant tool. That is to say, Mathematical Knowledge is dispersed among several sources.

Our aim in this work is to mechanize, in some particular cases, the management of these multiple-source information systems. Since it would be too pretentious to try to solve fully this problem, we work in a very specific context. Thematically, we restrict ourselves to (a subset of) Algebraic Topology. With respect to the sources, in order to have a representation wide enough, we have chosen two Computer Algebra systems (Kenzo and GAP), a theorem prover (ACL2) and a small expert system developed by us. The objective of the expert system is computing *homotopy* groups. Kenzo and GAP can compute *homology* groups of different spaces, but the calculation of *homotopy* is in general much harder. Our homotopy expert system tries to take profit of theoretical knowledge contained in theorems (tables have been excluded up to now, since they are considered less difficult to integrate, from a technological point of view), and can ask computational results to Kenzo, if needed.

This paper is a natural continuation of [17] and [18]. There are three main contributions in the paper: an architecture based on the Broker pattern [8] (proven as an open, flexible and adaptable tool); an Homotopy Expert System (HES) that allows non-trivial computations (and explanations) interacting with Kenzo; and the automation of the interoperability between Kenzo and GAP.

From the symbolic computation literature, we looked for inspiration in different projects and frameworks such as the MathWeb software bus [5], its successor the MathServe Framework [4], the MoNET project [9,6] or the MathBroker [2] and MathBroker II [3] projects, as well as in other works as [13] or [22].

2 General view of the system

The *Broker* architectural pattern [8] can be used to structure software systems with decoupled components that interact through service invocations. The Broker pattern defines three kinds of participants: *clients*, *components*, and the *broker* itself. A scheme of our architecture based on this pattern is depicted in Figure 1. The mediator (broker) component embeds an *Internal Memory* where a strategy of *memoization* has been systematically implemented (based on the same idea used in GAP for attributes, see [16]). The system stores the results in the internal memory when a computation is executed for the first time, and if the same computation is asked again later, the result is simply looked up and returned, without further computation.
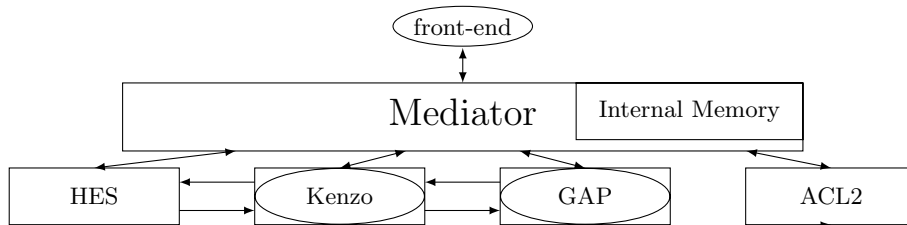

Fig. 1. Broker architectural pattern

The decorator pattern [8] is used to wrap objects of our system with information, like the type of the object (simplicial set, group,. . .) or the reduction degree [17] if the object is a topological space. This information guides the mediator to decide which component to use. Namely, Kenzo [11] (a Symbolic Computation system devoted to Algebraic Topology) is the core for computations related to homology groups of spaces, GAP [1] (a Computer Algebra system in the area of Computational Group Theory) and HAP [12] (a GAP homological algebra library) are the core for computations related to group homology, ACL2 [19] (a first order logic theorem prover) is the kernel for verifying the correctness of statements and, finally, the Homotopy Expert System (a small module developed by us described in the next paragraph and from now on called HES) is in charge of computing homotopy groups.

HES is a rule-based expert system. The structure of a rule-based expert system, see [15], consists of, and the HES is no exception, the following components: the *Working memory (the facts)*, the *Knowledge base (the rules)*, the *Inference engine*, a *Knowledge acquisition* module and an *Explanation facility* module. In the scope of the HES, the facts are properties associated with the objects (for instance, "$\forall n \in \mathbb{N} : \Delta^n$ is a contractible space"). The current knowledge base is

made up of 23 rules (such as, "$\texttt{if}$ X is contractible $\texttt{and}$ $n \geq 1$ $\texttt{then}$ $\pi_n(X) = 0$"). The inference engine uses the *forward chaining* method for reasoning, see [15]. To grapple with the knowledge acquisition aspects, the HES takes profit from both the RuleML markup language [7] and the OMDoc format [20], the former one is used to specify rules in a declarative way and the second one to store concrete functionalities. Last but not least, gathering the applied rules and the facts that decorate each object, our HES is able to provide a trace containing the reasoning followed by it in order to reach a conclusion.

However, the power of our system does not lie in gathering several computer algebra systems and theorem provers and use them separately with the same front end, but interconnecting them to reach new results. The communication among modules is performed by means of the OpenMath language [10], used to represent the objects in a common language for all the systems.

In [21] an approach to coordinate GAP and Kenzo was presented. In that work GAP and Kenzo cooperate in order to compute homology groups of some spaces. These spaces with their homology can then be used in other constructions and applications. Some enhancements of that tool provided by our system are: avoidance of the installation of several programs and packages, automation of communication steps (here the SCSCP protocol [14] plays a key role) and concealment of the details to mix the systems.

The general procedure and technology to connect with the ACL2 system explained in [18] is now applied to the context of group homology. The Common Lisp code used in Kenzo to represent a group is sent to ACL2 as an instance of an ACL2 *encapsulate* (a mechanism to introduce new functions symbols by axioms constraining them to have certain properties) by means of our broker, which is also in charge of invoking ACL2 in a way transparent for the user.

In another line, Kenzo and the HES cooperate to compute homotopy groups of spaces. In this case, the HES requests Kenzo to compute homology groups which can be used to obtain homotopy groups. Whereas Kenzo communicates with the HES in order to send it results. The idea consists of gathering the knowledge stored in the HES and chaining several tools available in Kenzo to get results which are not reachable by anyone of them working in an independent way.

3   Putting all together

Our current front-end has evolved from the user interface for Kenzo presented in [17]. Its presentation layer is kept, but its internal mediator has been enriched to support different sources of information. Figure 2 displays some computations which took profit of the following interactions: $H_4(\Omega^2(S^4)) = \mathbb{Z}$ (computed with Kenzo), $H_5(C_5) = \mathbb{Z}/5\mathbb{Z}$ (computed with GAP), $\pi_4(\Delta^4 \times \Delta^5) = 0$ (obtained with the Homotopy Expert System), $H_5(K(C_5, 1)) = \mathbb{Z}/5\mathbb{Z}$ (computed with Kenzo + GAP), $\pi_4(S^4) = \mathbb{Z}$ (obtained with the Homotopy Expert System + Kenzo).

It is worth noting that all results are shown to the user in a unique screen and that the computations are asked from a sole menu, then the user does not know
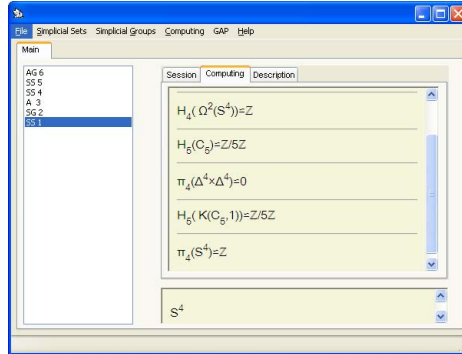
Fig. 2. The front end for computations

the system in charge of computing neither the collaboration among computer algebra systems, he only receives the desired result. The technical details are hidden to the user. The results related to ACL2 are shown in a different tab to split the computations from the deductions.

4   Conclusions and Further Work

In this paper an architecture to integrate different tools for computing and logical reasoning in Algebraic Topology is presented. Even if our proposal has a limited extend, both thematically and from the point of view of the core systems, we think it shows a solid line of research that could be exported to other areas of mathematical knowledge management. OpenMath technologies are the essential tool ensuring the interoperability among systems (even integration in some cases). This interoperability has a vertical dimension (from the mediator to the kernel systems) as well as a horizontal axis (allowing direct interconnection of kernel systems). The modules can be taken from their sources (as in the cases of Kenzo and ACL2), invoked in a remote manner (like the GAP server, connected via the SCSCP protocol) or even developed in an ad-hoc way (as our Homotopy Expert System).

Several research lines are still open. The most important ones are related to giving more resources to the user to manage the interaction. Moreover, it would be also necessary to improve the interaction with the ACL2 system. At this moment the queries must be pre-processed; a comfortable way of introducing questions about the truth of properties of intermediary objects, dynamically generated during a computing session, should be provided. Last, and the most difficult one, a meta-language should be designed to specify how and when a new kernel system can be plugged in the framework. This capability and the necessity of orchestrating the different services suppose a real challenge, which will be explored by means of OpenMath technologies.

References

1. GAP - Groups, Algorithms, Programming - System for Computational Discrete Algebra. `http://www.gap-system.org`.
2. MathBroker: A Framework for Brokering Distributed Mathematical Services. http://www.risc.uni-linz.ac.at/projects/basic/mathbroker/.
3. MathBroker II: Brokering Distributed Mathematical Services. http://www.risc.uni-linz.ac.at/projects/mathbroker2/.
4. MathServe Framework. http://www.ags.uni-sb.de/jzimmer/mathserve.html.
5. MATHWEB-SB:A Software Bus for MathWeb. http://www.ags.uni-sb.de/jzimmer/mathweb-sb/.
6. MoNET: Mathematics on the Net. http://monet.nag.co.uk/cocoon/monet/index.html.
7. H. Boley et al. Rule Markup Language (RuleML) version 0.91, 2004. `http://ruleml.org/`.
8. F. Buschmann et al. *Pattern-Oriented Software Architecture. A Pattern Language for Distributed Computing*, volume 4 of *Software Design Patterns*. Wiley, 2007.
9. A.M. Cohen, O. Caprotti, H. Cuypers, M. N. Riem, and H. Sterk. Using OpenMath Servers for Distributing Mathematical Computations. *Proceedings of the Fifth Asian Technology Conference in Mathematics*, pages 325–336, 2000.
10. The OpenMath Consortium. Openmath standard 2.0, 2004. `http://www.openmath.org/standard/om20-2004-06-30/omstd20.pdf`.
11. X. Dousson, F. Sergeraert, and Y. Siret. The Kenzo program. Institut Fourier, Grenoble, 1998. `http://www-fourier.ujf-grenoble.fr/~sergerar/Kenzo/`.
12. G. Ellis. HAP package for GAP, 2009. `http://www.gap-system.org/Packages/hap.html`.
13. S. Freundt et al. Symbolic Computation Sofware Composability. *Lectures Notes in Computer Science*, 5144:285–295, 2008.
14. S. Freundt, P. Horn, A. Konovalov, S. Lindon, and D. Roozemond. Symbolic Computation Software Composability Protocol (SCSCP) specification, version 1.3, 2009. `http://www.symbolic-computation.org/scscps`.
15. J. C. Giarratano and G. D. Riley. *Expert Systems: Principles and Programming*. PWS Publishing Company, 2005.
16. The GAP group. *GAP a Tutorial*, chapter 8: Operations and Methods, pages 72–76. 2008.
17. J. Heras, V. Pascual, and J. Rubio. Mediated Access to Symbolic Computation Systems. *Lectures Notes in Computer Science*, 5144:446–461, 2008.
18. J. Heras, V. Pascual, and J. Rubio. Using Open Mathematical Documents to Interface Computer Algebra and Proof Assistant Systems. *Lectures Notes in Computer Science*, 5625:467–473, 2009.
19. M. Kaufmann and J. S. Moore. ACL2 version 3.6, 2009. `http://www.cs.utexas.edu/users/moore/acl2/`.
20. M. Kohlhase. *OMDoc − An open markup format for mathematical documents [Version 1.2]*. Springer Verlag, 2006.
21. A. Romero, G. Ellis, and J. Rubio. Interoperating between Computer Algebra systems: computing homology of groups with Kenzo and GAP. *Proceedings of International Symposium on Symbolic and Algebraic Computation*, pages 303–310, 2009.
22. E. Smirnova, C. M. So, and S. M. Watt. An architecture for distributed mathematical web services. *Lectures Notes in Computer Science*, 3119:363–377, 2004.