COLIBRI2

François Bobot* Hichem Rami Ait-El-Hara†
Christophe Junke*

June 4, 2026

COLIBRI2[1] is the successor of COLIBRI [3, 2]. Unlike traditional DPLL(T) based SMT solvers, it employs a Constraint Programming (CP) solving architecture centered on domain-based reasoning without learning. It manages equivalence classes using a union-find environment and coordinates search through specialized scheduling phases (for COLIBRI2: propagation, decision, last-effort, and fix-model).

Compared to the base solver COLIBRI, COLIBRI2 is completely redesigned and implemented in OCaml instead of ECLiPSe Prolog. This architectural shift provides the scheduling fairness and flexibility needed for SMT problems, alongside clear modular boundaries. Furthermore, COLIBRI2 introduces support for quantifiers and algebraic datatypes integrated through domain-based reasoning. We also acknowledge the use of the external OCaml library `dolmen` for parsing and typing SMT-LIB files, and the library `ocplib-simplex` for the simplex.

For the competition, the system is mainly focused on FP logics, though it also participates in logics that include bit-vectors. The reasoning techniques used by COLIBRI2 are detailed in the technical report [1]. In summary:

- For floating-point numbers:
  - Forward and backward propagation of unions of intervals of floating-point numbers.
  - Linearization of floating-point operations into real operations.
- For bit-vectors:
  - Forward and backward propagation of unions of intervals of integers with information about congruence.
  - Two possible extended Shostak theories: boolean ring or BDD.
  - Bit-vector operations are mapped to integer operations using modulo.
- For integers:
  - Forward and backward propagation of unions of intervals of integers with information about congruence.
  - Shostak theory of linear arithmetic.

*Université Paris-Saclay, CEA, List, F-91120, Palaiseau, France
†OCamlPro SAS, 21 Rue de Chatillon, 75014, Paris, France
[1] `http://colibri.frama-c.com`

  - Extended Shostak theory for products of terms.

- For quantifiers:
  - Efficient E-matching, using eager and delayed instantiation.
  - For universal quantification on floating-points, an optional mechanism can try to disprove them. In parallel, for every universal quantifier, it uses domains to reduce the domain for which the quantifier is true.

References

[1] François Bobot, Hichem Rami Ait-El-Hara, and Bruno Marre. *colibri2: A CP solver for SMT problems*. Tech. rep. CEA List, 2026. URL: `https://cea.hal.science/view/index/docid/5644448`.

[2] Zakaria Chihani et al. "Sharpening Constraint Programming approaches for Bit-Vector Theory". In: *CPAIOR 2017. International Conference on AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*. Integration of AI and OR Techniques in Constraint Programming. Padova, Italy, June 2017. URL: `https://hal-cea.archives-ouvertes.fr/cea-01795779`.

[3] Bruno Marre, François Bobot, and Zakaria Chihani. "Real Behavior of Floating Point Numbers". In: *The SMT Workshop*. SMT 2017, 15th International Workshop on Satisfiability Modulo Theories. Heidelberg, Germany, July 2017. URL: `https://hal-cea.archives-ouvertes.fr/cea-01795760`.