Künstliche Intelligenz manuscript No.
(will be inserted by the editor)

The Potsdam Answer Set Solving Collection 5.0

Martin Gebser · Roland Kaminski · Benjamin Kaufmann · Patrick Lühne · Philipp Obermeier · Max Ostrowski · Javier Romero · Torsten Schaub · Sebastian Schellhorn · Philipp Wanko

Received: date / Accepted: date

Abstract The Potsdam Answer Set Solving Collection, or *Potassco* for short, bundles various tools implementing and/or applying Answer Set Programming. The article at hand succeeds an earlier description of the Potassco project published in [6]. Hence, we concentrate in what follows on the major features of the most recent, fifth generation of the ASP system *clingo* and highlight some recent resulting application systems.

1 Answer Set Programming Systems

Answer Set Programming (ASP) offers a declarative and effective technology for solving knowledge-intense combinatorial (optimization) problems. As such, it is often designated as a *model*, *ground*, *and solve paradigm* that features a high-level first-order specification language, which is turned by a grounder into a propositional format that is finally used by a solver to compute (optimal) solutions of the original problem. This traditional workflow is also followed by the ASP system *clingo*, which relies on the grounder *gringo* and the solver *clasp*. Their basic functioning is described in [7,11]. *clingo*'s input language slightly extends the ASP language standard [3] and has been put on firm semantic foundations in its full extent [5]. In practice, often a more flexible approach is needed to capture evolving or heterogeneous problem specifications. The fifth generation of *clingo* addresses both via means for controlling solving processes and for extending them with constraints foreign to ASP. Both corresponding techniques, *multi-shot* and *theory solving*, are sketched next, and detailed in [10].

T. Schaub
University of Potsdam E-mail: torsten@cs.uni-potsdam.de

Multi-shot solving allows for solving continuously changing logic programs in an operative way. This can be controlled via APIs implementing reactive procedures that loop on grounding and solving while reacting, for instance, to outside changes or previous solving results. Such reactions may entail the addition or retraction of rules that *clingo*'s operative approach can accommodate while leaving unaffected program parts intact within the solver. This avoids re-grounding and benefits from heuristic scores and constraints learned over time. *clingo* supports this by two language constructs. Programs can be partitioned into (parametrized) subprograms with directive `#program` and externally determined atoms can be declared with `#external`. While the former enables grounding and solving procedures to concentrate on subprograms, for instance, when iteratively unfolding a transition function, the latter allows us to control the truth value of atoms, for instance, when incorporating exogenous information. Paired with APIs for grounding, solving, assigning truth values, etc. this provides us with fine-grained control over the ASP solving process and offers a high degree of customization. Use cases of multi-shot solving include incremental and reactive reasoning in general, and more specifically, complex optimization, planning and monitoring, multi-agent systems, sensor data handling, etc.

Theory solving allows us to extend the range of ASP beyond its native constraints. Such extensions concern the entire workflow and affect not only the actual solver but as well the modeling language and its grounder. *clingo* provides generic means for adding theory solving capacities. On the one hand, it offers theory grammars for expressing theory languages whose expressions are seamlessly integrated in the grounding process. On the other hand, a simple API consisting of four methods offers an easy integration of theory propagators into the solver, either in C, C++, Lua, or Python. Meanwhile this framework has been instantiated in various ways. Of interest are for example extensions with linear constraints over integers and reals [9]:

- *clingo*[DL] extends *clingo* with difference constraints (of form $x - y \leq k$) over reals and integers.
- *clingo*[LP] extends *clingo* with linear constraints over reals and integers via an interface to linear programming solvers such as *cplex* or *lpsolve*.
- *clingcon* extends *clingo* with linear constraints over integers and global constraints like *distinct*.

For example, in *clingo*[DL], the rule

```
&diff{ end(T)-ini(T) } <= D :- duration(T,D).
```

can be used to express that a task `T` respects its duration `D`. The atom in the rule head represents an (aforementioned) difference constraint, in which `ini(T)` and `end(T)` are real variables indicating a task's start and end times; `D` is a real number. Theory atoms begin with '`&`' and are defined by the respective theory grammar. The ASP solver treats them as common atoms; their subatomic structure is interpreted by theory propagator, e.g., handling difference constraints in case of *clingo*[DL].

2 ASP-based Application Systems

The Potassco suite is in use in various application areas in academia as well as industry worldwide. One of the highlights was presumably the use of *clasp* in the US-wide auction to re-purpose radio spectrum from broadcast television to wireless internet [12]. Here is a selection of applications hosted at `potassco.org`:

- *aspcud* [8], a solver for Linux package configuration.
- *asprin* [2], a general framework for (combined) qualitative and quantitative optimization in ASP.
- *chasp* [13], a composer for musical harmonies.
- *plasp* [4], a PDDL-based planning system.
- *teaspoon* [1], a course timetabling system.

All software, further information, and resources are available at `potassco.org`.

References

1. Banbara, M., Inoue, K., Kaufmann, B., Okimoto, T., Schaub, T., Soh, T., Tamura, N., Wanko, P.: teaspoon: Solving the curriculum-based course timetabling problems with answer set programming. Annals of Operations Research (2018). To appear.
2. Brewka, G., Delgrande, J., Romero, J., Schaub, T.: asprin: Customizing answer set preferences without a headache. In: Proceedings of the Twenty-Ninth National Conference on Artificial Intelligence (AAAI'15), pp. 1467–1474. AAAI Press (2015).
3. Calimeri, F., Faber, W., Gebser, M., Ianni, G., Kaminski, R., Krennwallner, T., Leone, N., Ricca, F., Schaub, T.: ASP-Core-2: Input language format.
4. Dimopoulos, Y., Gebser, M., Lühne, P., Romero, J., Schaub, T.: plasp 3: Towards effective ASP planning. In: Proceedings of the Fourteenth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'17), pp. 286–300. Springer-Verlag (2017)
5. Gebser, M., Harrison, A., Kaminski, R., Lifschitz, V., Schaub, T.: Abstract Gringo. Theory and Practice of Logic Programming **15**(4-5), 449–463 (2015).
6. Gebser, M., Kaminski, R., Kaufmann, B., Ostrowski, M., Schaub, T., Schneider, M.: Potassco: The Potsdam answer set solving collection. AI Communications **24**(2), 107–124 (2011)
7. Gebser, M., Kaminski, R., Kaufmann, B., Schaub, T.: Answer Set Solving in Practice. Morgan and Claypool Publishers (2012)
8. Gebser, M., Kaminski, R., Schaub, T.: aspcud: A Linux package configuration tool based on answer set programming. In: Proceedings of the Second International Workshop on Logics for Component Configuration (LoCoCo'11), *Electronic Proceedings in Theoretical Computer Science (EPTCS)*, vol. 65, pp. 12–25 (2011)
9. Janhunen, T., Kaminski, R., Ostrowski, M., Schaub, T., Schellhorn, S., Wanko, P.: Clingo goes linear constraints over reals and integers. Theory and Practice of Logic Programming **17**(5-6), 872–888 (2017)
10. Kaminski, R., Schaub, T., Wanko, P.: A tutorial on hybrid answer set solving with clingo. In: Proceedings of the Thirteenth International Summer School of the Reasoning Web, pp. 167–203. Springer-Verlag (2017)
11. Kaufmann, B., Leone, N., Perri, S., Schaub, T.: Grounding and solving in answer set programming. AI Magazine **37**(3), 25–32 (2016)
12. Newman, N., Fréchette, A., Leyton-Brown, K.: Deep optimization for spectrum repacking. Communications of the ACM **61**(1), 97–104 (2018)
13. Opolka, S., Obermeier, P., Schaub, T.: Automatic genre-dependent composition using answer set programming. In: Proceedings of the Twenty-first International Symposium on Electronic Art (ISEA'15) (2015).