A Unifying Framework for Semiring-Based Constraint Logic Programming With Negation

Jeroen Spaans[1 [0000−0001−7027−8102]] and Jesse Heyninck[1 [0000−0002−3825−4052]]

Open Universiteit, The Netherlands {jeroen.spaans,jesse.heyninck}@ou.nl

Many problems require the consideration of constraints; examples of such problems can be relatively simple, like sudokus and similar puzzles, or more complex, like the real-worlds applications of resource allocation and automated planning and scheduling. Classically, a problem as such may be formulated as a Constraint Satisfaction Problem (CSP) and solved using Constraint Logic Programming (CLP). We investigate an extension of CLP capable of handling semiring-based constraints with negation.

Consider an informal example constraint logic program, which describes the allocation of a limited number of working hours to two different tasks.

```
% We define two tasks, taking 6 and 4 hours to complete respectively.
task(t1, 6).
task(t2, 4).
% A task is completed if enough time is scheduled.
completed(Task, HoursScheduled) :-
    task(Task, HoursRequired),
    HoursRequired ≤ HoursScheduled.
% We set a time limit of eight hours.
inTimeLimit(Hours1, Hours2) :- Hours1 + Hours2 ≤ 8.
% A schedule is evaluated by the degree to which both tasks are completed
% and the total allotted time does not exceed the time available.
schedule(HoursTask1, HoursTask2) :-
    completed(t1, HoursTask1),
    completed(t2, HoursTask2),
    inTimeLimit(HoursTask1, HoursTask2).
```

Using classical answer set programming semantics to evaluate this program with the goal `schedule(HoursTask1, HoursTask2)` returns *false*. Intuitively, this happens because no schedule can complete the two tasks—totalling ten hours of work—in less than eight hours. Knowing that the two tasks cannot both be completed in the time available, we may wish to optimize their partial completion instead. One way to do this is to replace *false* and *true* with values in $[0, 1]$ (where 1 represents complete truth and 0 complete falsity), replace *or* with max, replace *and* with min, and replace `HoursRequired` $\leq$ `HoursScheduled` with `HoursScheduled / HoursRequired`.

Our example demonstrated that CLP is limited to strict satisfaction or violation of constraints, and that we require an alteration of this framework to solve problems of constraint optimization. The same holds—for example—for problems involving fuzziness, uncertainty, or probability.

Bistarelli et al. [2,3,4] proposed Semiring-based Constraint Logic Programming, a generalization of CLP replacing the Boolean evaluation domain and the associated logical *and* and *or* connectives with semirings—algebraic structures consisting of a set equipped with an additive operator for disjunction and a multiplicative operator for conjunction—much like we did in our example. Since, many related formalisms have likewise been extended to the semiring setting [1,6,7,8,9,10,11] to certain success. Each of these works makes some assumptions about the semirings used, but what exactly those assumptions are and how they relate is left implicit or has not been studied. Herein lies the first major contribution of this work; we perform an analysis of the various families of semirings in relation to semiring-based semantics for CLP, paying special attention to the orderings each family gives rise to.

While some of the above-mentioned works permit negation, most do not, and a general analysis of negation in the semiring setting is so far absent from the literature. Herein lies the second major contribution of this work; a semiring-agnostic form of negation is proposed and the effects of its addition on the semantics of semiring-based constraint logic programming are studied. Notably, and as is to be expected, the addition of negation leads to nonmonotonicity of the immediate consequence operator. To work around this problem we capture the new negation-permitting formalism in Approximation Fixpoint Theory [5], endowing it with AFT's various semantics like Kripke-Kleene, Well-founded, and Stable, which generalize the semantics of ordinary logic programs.

Concretely, the contributions of this work come in five parts: first, a novel notion of model—considering all contributing clauses at once, and specific to the semiring-based setting—is introduced and compared to the traditional notion of model which considers each clause's satisfaction separately. The minimal model semantics based on these notions of model are then compared with the least fixpoint semantics based on an immediate consequence operator. Next, we investigate a generalized method for deriving orderings of semiring elements needed to define models and least fixpoints—but also needed in the later application of approximation fixpoint theory—and find it to be a generalization of the method studied by Bistarelli at al. Then, we study a generalized notion of negation appearing at various points in the literature as applied to our semiring-based framework. Finally, we apply approximation fixpoint theory to our immediate consequence operator—made nonmonotonic by the addition of negation—to define Kripke-Kleene, and well-founded and other stable semantics, studying both ultimate approximation and a novel approximator.

This work studies semiring-based semantics for constraint logic programming with negation. Translating its results to Datalog is future work.

References

1. Belle, V., Raedt, L.D.: Semiring programming: A semantic framework for generalized sum product problems. Int. J. Approx. Reason. **126**, 181–201 (2020). https://doi.org/10.1016/J.IJAR.2020.08.001

2. Bistarelli, S.: Semirings for Soft Constraint Solving and Programming, Lecture Notes in Computer Science, vol. 2962. Springer (2004). https://doi.org/10.1007/B95712
3. Bistarelli, S., Montanari, U., Rossi, F.: Semiring-based constraint satisfaction and optimization. J. ACM **44**(2), 201–236 (1997). https://doi.org/10.1145/256303.256306
4. Bistarelli, S., Montanari, U., Rossi, F.: Semiring-based contstraint logic programming: Syntax and semantics. ACM Trans. Program. Lang. Syst. **23**(1), 1–29 (2001). https://doi.org/10.1145/383721.383725
5. Denecker, M., Marek, V., Truszczynski, M.: Approximations, Stable Operators, Well-Founded Fixpoints And Applications In Nonmonotonic Reasoning. Logic-based Artificial Intelligence (Dec 2001). https://doi.org/10.1007/978-1-4615-1567-8_6
6. Eiter, T., Kiesel, R.: ASP($\mathcal{AC}$): Answer Set Programming with Algebraic Constraints. Theory Pract. Log. Program. **20**(6), 895–910 (2020). https://doi.org/10.1017/S1471068420000393
7. Eiter, T., Kiesel, R.: Weighted LARS for Quantitative Stream Reasoning. In: ECAI 2020 - 24th European Conference on Artificial Intelligence, 29 August-8 September 2020, Santiago de Compostela, Spain, August 29 - September 8, 2020 - Including 10th Conference on Prestigious Applications of Artificial Intelligence (PAIS 2020). pp. 729–736 (2020). https://doi.org/10.3233/FAIA200160
8. Green, T.J., Karvounarakis, G., Tannen, V.: Provenance semirings. In: Proceedings of the Twenty-Sixth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 11-13, 2007, Beijing, China. pp. 31–40 (2007). https://doi.org/10.1145/1265530.1265535
9. Khamis, M.A., Ngo, H.Q., Pichler, R., Suciu, D., Wang, Y.R.: Convergence of datalog over (Pre-) Semirings. J. ACM **71**(2), 8:1–8:55 (Apr 2024). https://doi.org/10.1145/3643027
10. Kimmig, A., den Broeck, G.V., Raedt, L.D.: An Algebraic Prolog for Reasoning about Possible Worlds. In: Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2011, San Francisco, California, USA, August 7-11, 2011. pp. 209–214 (2011). https://doi.org/10.1609/AAAI.V25I1.7852
11. Kimmig, A., den Broeck, G.V., Raedt, L.D.: Algebraic model counting. J. Appl. Log. **22**, 46–62 (2017). https://doi.org/10.1016/J.JAL.2016.11.031