On the Subtlety of Causal Reasoning in Probabilistic Logic Programming: A Bug Report about the Causal Interpretation of Annotated Disjunctions

Kilian Rückschloß[1,†], Felix Weitkämper[1,†]

[1]*Ludwig-Maximilians-Universität München*
Institut für Informatik
Oettingenstraße 67
D-80538 München

Abstract
In this work in progress, we give an example for a logic program with annotated disjunctions where the do-operator does not behave as intended. In particular, we see that the mutual exclusivity of heads in an annotated disjunction is not preserved after intervention.

Bug Report

In this contribution, we study the causal semantics of the LPAD-language [1]. Assume for instance that we throw a die if we decide to play, denoted $play$. In this case, we throw one, two, $three$, $four$, $five$ or six each with a probability of $\frac{1}{6}$. This scenario can now be modelled with the LPAD-clause

$$one : \frac{1}{6}; two : \frac{1}{6}; three : \frac{1}{6}; four : \frac{1}{6}; five : \frac{1}{6}; six : \frac{1}{6} \leftarrow play. \tag{1}$$

To establish a causal semantics for LPAD-programs we rely on the functional causal model semantics or FCM-semantics [2] for ProbLog programs and use the fact that the languages ProbLog and LPAD are expressively equivalent [3, §2.4]. Generally, the FCM-semantics associates to each (acyclic) ProbLog program a system of Boolean equations that is uniquely solvable in terms of some independent predefined random variables, i.e. a functional causal model in the sense of Pearl [4].

To illustrate this, we consider a road, which passes along a field with a sprinkler in it. It is spring or summer, denoted szn_spr_sum with a probability of $\pi_1 := 0.5$. The sprinkler is switched on, written $sprinkler$, by a weather sensor with probability $\pi_2 := 0.7$ in spring or

PLP 2023: The Tenth Workshop on Probabilistic Logic Programming, London, July 09th 2023
[†]These authors contributed equally.
kilian.rueckschloss@lmu.de (K. Rückschloß); felix.weitkaemper@lmu.de (F. Weitkämper)
0000-0002-7891-6030 (K. Rückschloß); 0000-0002-3895-8279 (F. Weitkämper)

summer. Moreover, it rains, denoted by $rain$, with probability $\pi_3 := 0.1$ in spring or summer and with probability $\pi_4 := 0.6$ in fall or winter. If it rains or the sprinkler is on, the pavement of the road gets wet, denoted by wet. And in the case where the pavement is wet we observe that the road is slippery, denoted by $slippery$.

To model this situation we generate mutually independent Boolean random variables u_1-u_4 with $\pi(u_i) = \pi_i$ for all $1 \leq i \leq 4$. The described mechanism is then represented by the following system of equations:

$$\begin{aligned} &szn\_spr\_sum := u1 \\ &sprinkler := szn\_spr\_sum \wedge u2 \\ &rain := (szn\_spr\_sum \wedge u3) \vee (\neg szn\_spr\_sum \wedge u4) \\ &wet := (rain \vee sprinkler) \\ &slippery := wet \end{aligned} \tag{2}$$

Furthermore, the FCM-semantics identifies the System of Equations (2) with the following ProbLog program.

$$\begin{aligned} &0.5 :: szn\_spr\_sum \\ &0.7 :: sprinkler \leftarrow szn\_spr\_sum \\ &0.1 :: rain \leftarrow szn\_spr\_sum \\ &0.6 :: rain \leftarrow \neg szn\_spr\_sum \\ &wet \leftarrow rain \\ &wet \leftarrow sprinkler \\ &slippery \leftarrow wet \end{aligned} \tag{3}$$

Using the FCM-semantics, we can now transfer Pearl's causal reasoning [4] to probabilistic logic programming. This enables us to evaluate queries about the effect of external interventions [2] and about counterfactuals [5]. In this contribution, we focus on modeling external interventions.

What happens in Model 2 if we switch the sprinkler on/off?

As switching the sprinkler on/off basically means that nothing changes but the fact that $sprinkler$ is set to $true/false$, we can model this action by replacing the equation for $sprinkler$ by $sprinkler := true/false$. This can be modelled in Program (6) by erasing the clause $0.7 :: sprinkler \leftarrow szn\_spr\_sum$ and adding the fact $sprinkler \leftarrow$ in case we switched the sprinkler on.

Generally, intervening on a proposition p in a propositional ProbLog program $\mathbf{P}$ is done by erasing all clauses defining p and adding the fact $p \leftarrow$ in the case where p is set to true.

However, remember that we were initially interested in the LPAD-language. As already mentioned earlier to define a semantics for a LPAD-program $\mathbf{P}$, for each proposition p we introduce an auxiliary proposition p^{RC} and transform it to an equivalent ProbLog program according to Riguzzi [3, §2.4]. In our example, we transform the Program (1) to the following ProbLog program.

$$
\begin{aligned}
&\frac{1}{6} :: one^{RC} \leftarrow play \\
&one \leftarrow one^{RC} \\
&\frac{1}{5} :: two^{RC} \leftarrow play, \neg one^{RC} \\
&two \leftarrow two^{RC} \\
&\frac{1}{4} :: three^{RC} \leftarrow play, \neg one^{RC}, \neg two^{RC} \\
&three \leftarrow three^{RC} \\
&\frac{1}{3} :: four^{RC} \leftarrow play, \neg one^{RC}, \neg two^{RC}, \neg three^{RC} \\
&four \leftarrow four^{RC} \\
&\frac{1}{2} :: five^{RC} \leftarrow play, \neg one^{RC}, \neg two^{RC}, \neg three^{RC}, \neg four^{RC} \\
&five \leftarrow five^{RC} \\
&1 :: six^{RC} \leftarrow play, \neg one^{RC}, \neg two^{RC}, \neg three^{RC}, \neg four^{RC}, five^{RC} \\
&six \leftarrow six^{RC}
\end{aligned} \tag{4}
$$

Further, assume we "play" and just turn over the die to show the number four, i.e. we intervene in our model and force the dice to show $four$. As already discussed, this results in the following ProbLog program.

$$
\begin{aligned}
&\frac{1}{6} :: one^{RC} \leftarrow play \\
&one \leftarrow one^{RC} \\
&\frac{1}{5} :: two^{RC} \leftarrow play, \neg one^{RC} \\
&two \leftarrow two^{RC} \\
&\frac{1}{4} :: three^{RC} \leftarrow play, \neg one^{RC}, \neg two^{RC} \\
&three \leftarrow three^{RC} \\
&\frac{1}{3} :: four^{RC} \leftarrow play, \neg one^{RC}, \neg two^{RC}, \neg three^{RC} \\
&four \leftarrow \\
&\frac{1}{2} :: five^{RC} \leftarrow play, \neg one^{RC}, \neg two^{RC}, \neg three^{RC}, \neg four^{RC} \\
&five \leftarrow five^{RC} \\
&1 :: six^{RC} \leftarrow play, \neg one^{RC}, \neg two^{RC}, \neg three^{RC}, \neg four^{RC}, five^{RC} \\
&six \leftarrow six^{RC}
\end{aligned} \tag{5}
$$

Querying the Program (5) for the probability of one should now yield the probability $\pi(one|\,\mathrm{do}(four))$ of the die showing one if we force it to show four. As a die normally cannot show two numbers at the same time we expect $\pi(one|\,\mathrm{do}(four))$ to be zero; however, evaluating (5) yields $\pi(one|\,\mathrm{do}(four)) = \frac{1}{6}$.

It is worth mentioning that CP-logic [6] is another causal semantics for LPAD-programs, which also supports queries about the effect of external interventions. Moreover, the reasoning about external interventions provided by CP-logic is implemented in cplint [7]. However, we showed in Kiesel et al. [5] that evaluating counterfactal queries in CP-logic is the same as translating a LPAD-program **P** into its associated ProbLog program and evaluating the query there under the FCM-semantics. As determining the effect of external interventions is a special case of counterfactual reasoning [4], we obtain the same result $\pi(one|\,\mathrm{do}(four)) = \frac{1}{6}$ if we had evaluated the query under CP-logic.

In particular, let us encode a modified version of Program (1) into the following cplint program.

```
:- use_module(library(pita)).
:- use_rendering(graphviz).
:- pita.
:- begin_lpad.
:- action d/1.
d(1):1/6; d(2):1/6; d(3):1/6; d(4):1/6; d(5):1/6;d(6):1/6.
:- end_lpad.
```

Here, $d(i)$ means that the die shows the number i. As expected, we also obtain the wrong result when querying for the probability that the dice shows one if we force it to show four.

```
?- prob(d(1),do(d(4)),P).
    P = 0.16666666666666666
```

Discussion

The previous discussion reveals that the information about the mutually exclusiveness of the events one, two, $three$, $four$, $five$ and six encoded by the annotated disjunction (1) is lost by intervening in the corresponding ProbLog program (4). In particular, the post-interventional program (5) makes no statement to exclude the possible worlds in which the dice shows both four and one.

In logic programming, one would fix this issue by adding an integrity constraint to the Program (5) that encodes that the events one, two, $three$, $four$, $five$ and six exclude each other. However, in ProbLog, adding such an integrity constraint is equivalent to an observation [8]. This means we would not query the modified ProbLog Program (5) for the probability of one but for the marginal probability of one given that we observe the mutual exclusiveness of all events at hand. Hence, we expect that observing the mutual exclusiveness of the atoms occurring in an annotated disjunction would resolve our bug in some cases.

However, if we reconsider the situation modelled in Program (6) and assume a sensor guarantees that the sprinkler is off if it rains, this could result in the following LPAD-program.

$$
\begin{aligned}
&szn\_spr\_sum : 0.5 && wet \leftarrow rain \\
&rain : 0.1; sprinkler : 0.7 \leftarrow szn\_spr\_sum && wet \leftarrow sprinkler \\
&rain : 0.6 \leftarrow \neg szn\_spr\_sum && slippery \leftarrow wet
\end{aligned} \tag{6}
$$

In this case, switching the sprinkler on does not avoid raining. Hence, it now makes sense that the mutual exclusiveness of an annotated disjunction is destroyed by an intervention.

Therefore, we think that a correct causal interpretation of annotated disjunctions requires to operators to express the "Or" in head of a LPAD-clause. One operator for which the mutual exclusivity pertains under an intervention and one for which it does not. Finally, one then needs to find the right notion of intervention for either operators in the LPAD language.

Acknowledgements

The research leading to this publication was supported by LMUexcellent, funded by the Federal Ministry of Education and Research (BMBF) and the Free State of Bavaria under the Excellence Strategy of the Federal Government and the Länder.

References

[1] J. Vennekens, S. Verbaeten, M. Bruynooghe, Logic programs with annotated disjunctions, in: Logic Programming, Springer Berlin Heidelberg, Berlin, Heidelberg, 2004, pp. 431–445. doi:10.1007/978-3-540-27775-0_30.

[2] K. Rückschloß, F. Weitkämper, Exploiting the full power of pearl's causality in probabilistic logic programming, in: Proceedings of the International Conference on Logic Programming 2022 Workshops co-located with the 38th International Conference on Logic Programming (ICLP 2022), Haifa, Israel, July 31st - August 1st, 2022, volume 3193 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2022. URL: http://ceur-ws.org/Vol-3193/paper1PLP.pdf.

[3] F. Riguzzi, Foundations of Probabilistic Logic Programming: Languages, Semantics, Inference and Learning, River Publishers, 2022. doi:10.1017/9781108770750.

[4] J. Pearl, Causality, 2 ed., Cambridge University Press, Cambridge, UK, 2000. doi:10.1017/CBO9780511803161.

[5] R. Kiesel, K. Rückschloß, F. Weitkämper, "What if?" in probabilistic logic programming, Accepted for TPLP Proceedings of ICLP (2023). URL: https://arxiv.org/abs/2305.15318.

[6] J. Vennekens, M. Denecker, M. Bruynooghe, CP-logic: A language of causal probabilistic events and its relation to logic programming, Theory and Practice of Logic Programming 9 (2009) 245–308. doi:10.1017/S1471068409003767.

[7] F. Riguzzi, G. Cota, E. Bellodi, R. Zese, Causal inference in cplint, International Journal of Approximate Reasoning 91 (2017) 216–232. doi:10.1016/j.ijar.2017.09.007.

[8] D. Fierens, G. Van den Broeck, M. Bruynooghe, L. De Raedt, Constraints for probabilistic logic programming, in: D. Roy, V. Mansinghka, N. Goodman (Eds.), Proceedings of the NIPS Probabilistic Programming Workshop,, 2012. doi:10.1145/377978.377983.