Answer Set Programming for Actual Causality: The Role of Negations

Thomas Eiter, Tobias Geibinger, Xinghan Liu
{thomas.eiter, tobias.geibinger, xinghan.liu}@tuwien.ac.at

We propose an ASP approach to reasoning about Halpern-Pearl actual causes and argue that our encoding is simpler, conceptually more appropriate, and more general than the existing approach in literature.

Causality study: from philosophy to computer science Together with Judea Pearl, Joe Halpern [9, 8] has profoundly reshaped the study of causality, a field of growing importance in computer in relation to explainable artificial intelligence [4, 16]. Arguably, the best scientific explanations are causal, and "perhaps the main reason people are interested in causality is that they want 'explanations'", as Halpern [8] said.

To better appreciate the significance of his work, however, it is necessary to take a brief look at the history of philosophy, where the study of causality originated. Defining causality has been notoriously difficult ever since David Hume [10] questioned its nature. The use of *counterfactual conditionals* has proved particularly promising. It was already latent in Hume's writing, namely explaining causation by statements of the form *if the cause did not occur, the effect would not occur*.

This approach was challenged once by Goodman [7] as to whether counterfactuals capture the *non-monotonicity* of causal reasoning. His famous example is "if the match were struck (S), it would light (L)"$(\dagger)$ is true; but "if the oxygen were not present $(\neg O)$ and the match were struck, then it would light" $(\ddagger)$ is false. In fact, Goodman's counterargument did not extinguish, but rather promoted the approach of counterfactuals, as it turned out that they support non-monotonic reasoning well. David Lewis [13] formalized a logical theory of counterfactuals in terms of his comparative similarity. The key point is that, ceteris paribus, worlds without oxygen are less similar to the actual world compared to worlds with oxygen, and therefore excluded when evaluating $(\dagger)$. Nevertheless, Lewis' radical standpoint of reducing causality to counterfactuals forced him to establish the similarity relation without appealing to causal knowledge. To answer why $S \wedge L$ is more similar to the actual world $\neg S \wedge \neg L$ compared to $S \wedge \neg L$, he had to rely on vague concepts such as big and small miracles [14].

The Halpern-Pearl (HP) approach represents a paradigm shift away from the philosophical inquiry of nature of causality. The so-called structural equation model (SEM) takes atomic causal dependencies for granted (by commonsense causal knowledge), e.g. the positive influence of S on L. The task is then to identify causes in different settings. Moreover, the HP approach goes beyond Lewis' theory in the following respects.

a) *Complete-information modeling*. The causal model is presumed to contain all

relevant causal dependencies and can be represented as a causal graph with a set of structural equations for its variables. Thus, if the presence of oxygen O is included as a variable, then the evaluation of (†) is sensitive to the value of O. By contrast, if the match's dryness D is not a variable, then it is not taken into account in the evaluation.

b) *Counterfactuals under intervention*. An intervention sets selected variables to specified values and removes affected structural equations. In the resulting counterfactual settings is the effect formula evaluated. Non-monotonicity remains essential, as expanding an intervention may disrupt causal chains and alter outcomes. [1]

Based on causal models, the notion of *actual cause* has formally defined and undergone several revisions. We concentrate on the most recent version, the *modified definition*. We omit the definition, adopt the notation, and refer the reader to [8, pp. 23–25, Def. 2.2.1].

Answer set programming Let us turn to another field, logic programming, where non-monotonicity is also a key issue in the context of closed world assumption and default negation (negation as failure). Answer set programming (ASP) is the best known solution to handle the tension between expressiveness and computational complexity. Given a set $\mathcal{A}$ of propositional atoms, a *normal program* P has rules of the form

$$r : a_0 \leftarrow a_1, \ldots, a_m, not\ a_{m+1}, \ldots not\ a_n., \tag{1}$$

where $a_0, \ldots, a_n \in \mathcal{A}$, a_0 is denoted as $head(r)$, $\{a_1, \ldots, a_m\}$ ($\{a_{m+1}, \ldots a_n\}$) as $body^+(r)$ ($body^-(r)$), and $body(r) = body^+(r) \cup body^-(r)$. We call r a *fact* if $body(r) = \emptyset$, and a *constraint* if $head(r)$ is $\bot$ (and therefore omitted).

An interpretation $I \subseteq \mathcal{A}$ is a *stable model* of P, if I is a *minimal model of* P^I. P^I is the *Gelfond-Lifschitz reduct* of P relative to I defined as

$$P^I = \left\{\ head(r) \leftarrow body^+(r) : r \in grd(P), body^-(r) \cap I = \emptyset\ \right\}. \tag{2}$$

where $grd(P)$ is the grounding of P, grounding is the sense of first-order logic. (When P has only propositional formulas, $grd(P) = P$.) The non-monotonicity of stable model semantics can be illustrated as follows: adding a new fact $abnormal.$ into $\{fly \leftarrow bird, not\ abnormal.,\ bird.\}$ changes $\{fly, bird\}$ from stable to unstable.

Related work Since both SEMs and ASP are non-monotonic rule-based systems, it is unsurprising that their connection has been studied before. However, most work [6, 3, 17] focuses on extending HP framework rather than implementing it. Only recently, ÖAC [15] presented an ASP implementation for actual causality, which significantly outperforms previous approaches based on SAT or integer linear programming [12, 11]. They use a two-step translation which we illustrate with the famous example below.

Example 1. *Suzy and Billy decided to hit a bottle. They both threw a rock aiming towards the bottle. Suzy's rock hit the bottle before Billy's arrived, and the bottle shattered. Halpern [8] modeled the causal setting* $(M, \vec{u})$ *with the equations*

$$ST = SD, \quad BT = BD, \quad SH = ST, \quad BH = BT \wedge \neg SH, \quad BS = BH \vee SH,$$

[1] Thus HP approach can be viewed as a special, computable Lewisian semantics: comparative similarity is determined by both the set of different variables value and the set of removed rules, as [1] argued.

and $\vec{u}$ being $(SD, BD) = (1, 1)$. Namely, the actual value of exogenous variables is (SD, BD), and the actual evaluation is $\vec{u} \cup (ST, BT, SH, \neg BH, BS)$. Note that we identify $SH = 1$ with SH and $BH = 0$ with $\neg BH$, as the model here is Boolean.

ÖAC's encoding will first translate the equations into program $\{SD.,\ BD.,\ ST \leftarrow SD.,\ BT \leftarrow BD.,\ SH \leftarrow ST.\ BH \leftarrow BT, not\ SH.,\ BS \leftarrow SH.,\ BS \leftarrow BH.\}$ in dealing with **AC1**. Then in the next step for **AC2**, the translation depends on both the given causal setting and candidate cause. For instance, setting $\vec{u} = (SD, BD)$ and considering the candidate cause (ST, BT) result in the following program:

$$BD., \quad BH \leftarrow BD \wedge not\ SH \wedge not\ in_w(BH, 0)., \quad SH \leftarrow not\ in_x(SH, 1).,$$
$$BS \leftarrow SH., \quad BS \leftarrow BH.,$$

where $in_w(BH, 0)$ means intervening BH whose actual value is 0, and $in_x(SH, 1)$ means intervening SH whose actual value is 1. Then more rules are added to ensure that $in_x(SH, 1)$ changes the actual value of SH while $in_w(BH, 0)$ fixes its actual value (the contingency set).

Their encoding, in our view, has two weaknesses apart from the practical limitation that it relies on *asprin*, whose documented compatibility is tied to *clingo* 5.4.0 and maintenance is less active after 2022.

1) The two-step translation is tailored to both context $\vec{u}$ and candidate $\vec{x}$, which makes it not only unnecessarily compliated, but also hard to generalize.

2) It is a conceptual mismatch to use default negation to translate variables' values. Default negation handles indeterminacy, and is epistemologically interpreted as *not known to be true*. But as mentioned earlier, in HP approach everything is deterministic as information is completely given by its modeling. The $\neg SH$ in equation for BH shall be interpreted as it is known that $SH = 0$ rather than it is unknown that $SH = 1$.

Our approach: strong negation for values, default negation for interventionism In light of the conceptual analysis, we consider *extended programs* where *strong negation* "$-$" is introduced. Then, for a rule r, $body^+(r)$ has form $\{a_1, \ldots, a_k, -a_{k+1}, \ldots, -a_m\}$. Note strong-negated atoms $-a$ are in the positive body, since it is interpreted as *a is known to be false*, and behaves similar to an atom.

The following definition presents the ASP translation at a meta level; the implementation-level encoding is given in Figure 1. Note **AC3** the minimality condition is ensured by the heuristic on Line 45 and 46.

Definition 2. *Let $(M, \vec{u})$ be a binary recursive causal setting and $X = \bigvee_{i=1}^{m} \delta_i$ be an equation in M, where $\bigvee_{i=1}^{m} \delta_i$ is in disjunction normal form. Our binary ASP translation is inductively defined as shown in Table 1.*

Default negation is only used to let possible intervention override the equation. Choice rules $\{x; -x\}$ are used for the law of excluded middle. To extend the translation to multi-valued models such as voting examples in [8], we simply change the atomic case from $\text{Tr}(X) = x$ to $\text{Tr}(X = x) = \texttt{val(X, x)}$, and add a numerical constraint that X takes exactly one value. It echoes our argument that $\neg SH$ shall be viewed as an atom $\texttt{val(SH, 0)}$ rather than a negative literal $not\ \texttt{val(SH, 1)}$.

$\mathrm{Tr}(X)$	x
$\mathrm{Tr}(\ell)$	p if $\ell = p$, $-p$ if $\ell = \neg p$
$\mathrm{Tr}(\bigwedge_{j\in 1,\ldots k} \ell_j)$	$\mathrm{Tr}(\ell_1), \ldots, \mathrm{Tr}(\ell_k)$
$\mathrm{Tr}((X, \delta))$	$\mathrm{Tr}(X) \leftarrow \mathrm{Tr}(\delta), not\ int(X, \_).$
$\mathrm{Tr}(X = \bigvee_i \delta_i)$	$\bigcup_{i\in\{1,\ldots,m\}} \mathrm{Tr}((X, \delta_i))$
$\mathrm{Tr}(M)$	$\bigcup\{\mathrm{Tr}(X = \bigvee_i \delta_i) : X = \bigvee_i \delta_i \in M\}$
$\mathrm{Tr}(\vec{u})$	$\{u. : \mathbf{u} = 1 \in \vec{u}\} \cup \{-u. : \mathbf{u} = 0 \in \vec{u}\}$
$\mathrm{Tr}((M, \vec{u}))$	$\mathrm{Tr}(M) \cup \mathrm{Tr}(\vec{u}) \cup \{\{x; -x\} : X \in \vec{V}\}$
$\mathrm{Tr}(Int(\vec{y}))$	$\bigcup_{\mathbf{y}=1\in\vec{y}}\{y \leftarrow int(\mathbf{y}, 1).\}$
	$\cup \bigcup_{\mathbf{y}=0\in\vec{y}}\{-y \leftarrow int(\mathbf{y}, 0).\}$
$\mathrm{Tr}((M, \vec{u})^{\vec{y}})$	$\mathrm{Tr}((M, \vec{u})) \cup \mathrm{Tr}(Int(\vec{y}))$

Table 1: Translation of binary SEMs. $(M, \vec{u})^{\vec{y}}$ denotes intervening $(M, \vec{u})$ by setting $\vec{Y} = \vec{y}$

As one can see, our encoding is aligned with the conceptual foundations of both interventionism and ASP. Thanks to its simplicity and modularity, it can be generalized to reason about multi-valued models, non-recursive models, and in particular notions that do not require $\vec{W}$ to take its actual value as ÖAC require, e.g. the original und updated definitions and sufficient cause.

Some theoretical results The correctness of encoding is not hard to shown and omitted. To guarantee the efficiency of our encoding, we also investigate the following two complexity problems which, to the best of our knowledge, has not been studied. [2]

Theorem 3. *Checking whether some actual cause for a given formula under a given causal setting exists is* $\mathbf{NP}$*-complete. The hardness holds already for binary models.*

Computing some actual cause for a given formula under a given causal setting is $\mathbf{FP}^{\mathbf{NP}}[log, wit]$*-complete. The hardness holds already for binary models.*

Note that no actual cause exists when φ holds necessarily in $(M, \vec{u})$. Hardness for binary models follows by reduction from SAT checking of $\neg\varphi$ as a Boolean function. The class $\mathbf{FP}^{\mathbf{NP}}[log, wit]$ contains all search problems solvable in P with access to an NP witness oracle log-many often. Hardness is proven by reduction from computing maximal model of QBF $\exists\vec{B}.\neg\varphi(\vec{A}, \vec{B})$, where intuitively $\vec{B}$ is the contingency set $\vec{W}$. The result matches our encoding, which requires log-many calls in order to satisfy **AC3**.

The last finding presented here is the connection between actual causes and constastive explanation for ASP defined by EGO [5].

Observation 4. *Given a binary causal setting* $(M, \vec{u})$*,* $P = Tr((M, \vec{u}), \{I\} = AS(P)$*, and* E *a conjunction of literals, then* $\vec{X} = \vec{x}$ *is an actual cause of* E *under* $(M, \vec{u})$ *iff the rules for* $\vec{X}$ *in* P *explains the contrastive question "Why* E *holds in* I *(fixing facts and constraints in* P*, and some addable facts from* I*), but not the opposite of* $\vec{X} = \vec{x}$*?"*

[2] In [8], deciding whether a *given* candidate is an actual cause was studied, and in [2], computing the degree of responsibility under modified definition was informally discussed.

Implementation and experiments We used a cluster with 10 nodes, each having 2 Intel Xeon Silver 4314, running Ubuntu 22.04. Clingo 5.8.0 and Python 3.12.8 were utilized. The total benchmark contains 512 instances for checking a candidate actual causes (500 for binary models from the literature [11, 15], 12 newly designed for non-binary models) and 181 instances for computing an actual cause (175 from the literature and 6 new for non-binary models). Our experiments show that checking a candidate actual cause takes 11.16 secs at most with the mean being 1.49 secs (standard deviation 2.38 secs). Computing an actual cause takes 22.13 secs with the mean being 2.6 secs (standard deviation 4.86 secs). The memory usage for both types of problems is between 40 MB and 732 MB for each instance. We remark that our encoding can also be used to compute sufficient causes by the saturation technique which, to the best of our knowledge, has not been implemented.

```
1  term_sat_act(D,A,V) :- struct_term(D,A,V),
2                             val_act(A',V') : in_pos(A',V',D);
3                          not val_act(A',V') : in_neg(A',V',D).
4  impl_val_act(A,V) :- struct_term(D,A,V), term_sat_act(D,A,V).
5
6  vt_cnt_act(D,V,N) :- vote(D,A1), domain(A1,V), #count{ A2 : in(A2,D), val_act(A2,V) } = N.
7  impl_val_act(A,V) :- vote(D,A), vt_cnt_act(D,V,N), N = #max{ M : vt_cnt_act(D,_,M) }.
8
9  impl_val_act(A,V) :- context(A,V).
10
11 val_act(A,V) :- impl_val_act(A,V).
12
13 val_act(A,V) :- default_value(A,V), not impl_val_act(A,_).
14
15 query_term_sat(D) :- query_term(D),
16                          val_act(A,V) : in_pos(A,V,D);
17                       not val_act(A,V) : in_neg(A,V,D).
18 :- not query_term_sat(_).
19
20 term_sat_int(D,A,V) :- struct_term(D,A,V),
21                             val_int(A',V') : in_pos(A',V',D);
22                           not val_int(A',V') : in_neg(A',V',D).
23 impl_val_int(A,V) :- struct_term(D,A,V), term_sat_int(D,A,V).
24
25 vt_cnt_int(D,V,N) :- vote(D,A1), domain(A1,V), #count{ A2 : in(A2,D), val_int(A2,V) } = N.
26 impl_val_int(A,V) :- vote(D,A), vote_count_int(D,V,N), N = #max{ M : vt_cnt_int(D,_,M) }.
27
28 impl_val_int(A,V) :- context(A,V).
29
30 val_int(A,V) :- intervention(A,V).
31 val_int(A,V) :- impl_val_int(A,V), not intervention(A,_).
32 val_int(A,V) :- default_value(A,V), not intervention(A,_), not impl_val_int(A,_).
33
34 { intervention(A,V) : domain(A,V) } <= 1 :- domain(A,_), not context(A,_).
35
36 cause(A,V1) :- intervention(A,V2), val_act(A,V1), V1 != V2.
37
38 query_term_viol(D) :- query_term(D), in_pos(A,V,D), not val_int(A,V).
39 query_term_viol(D) :- query_term(D), in_neg(A,V,D), val_int(A,V).
40 :- not query_term_viol(_).
41
42 check_candidate :- candidate_cause(_,_).
43 :- check_candidate, cause(A,V), not candidate_cause(A,V).
44
45 #heuristic cause(A,V) : domain(A,V).[1, false]
46 #show cause/2.
```

Figure 1: ASP encoding for actual causes

Acknowledgments

This research was funded in part by the Austrian Science Fund (FWF) 10.55776/COE12 and benefited from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101034440 (LogiCS@TUWien). Tobias Geibinger is a recipient of a DOC Fellowship of the Austrian Academy of Sciences.

References

[1] Carlos Aguilera-Ventura, Xinghan Liu, Emiliano Lorini, and Dmitry Rozplokhas. A non-interventionist approach to causal reasoning based on lewisian counterfactuals. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence*, pages 4301–4310, 2025.

[2] Gadi Aleksandrowicz, Hana Chockler, Joseph Y Halpern, and Alexander Ivrii. The computational complexity of structure-based causality. *Journal of Artificial Intelligence Research*, 58:431–451, 2017.

[3] Chitta Baral, Michael Gelfond, and Nelson Rushton. Probabilistic reasoning with answer sets. *Theory and Practice of Logic Programming*, 9(1):57–144, 2009.

[4] Sander Beckers. Causal explanations and xai. In *Conference on causal learning and reasoning*, pages 90–109. PMLR, 2022.

[5] Thomas Eiter, Tobias Geibinger, and Johannes Oetsch. Contrastive explanations for answer-set programs. In *European Conference on Logics in Artificial Intelligence*, pages 73–89. Springer, 2023.

[6] Michael Gelfond, Jorge Fandinno, and Evgenii Balai. Embracing background knowledge in the analysis of actual causality: An answer set programming approach. *Theory and Practice of Logic Programming*, 23(4):715–729, 2023.

[7] Nelson Goodman. *Fact, fiction, and forecast*. Harvard University Press, 1983(1955).

[8] Joseph Y Halpern. *Actual causality*. MiT Press, 2016.

[9] Joseph Y Halpern and Judea Pearl. Causes and explanations: A structural-model approach. Part I: Causes. *The British journal for the philosophy of science*, 2005.

[10] David Hume. *An enquiry concerning the human understanding: And an enquiry concerning the principles of morals*. Clarendon Press, 1894(1748).

[11] Amjad Ibrahim and Alexander Pretschner. From checking to inference: Actual causality computations as optimization problems. In Dang Van Hung and Oleg Sokolsky, editors, *Automated Technology for Verification and Analysis - 18th International Symposium, ATVA 2020, Hanoi, Vietnam, October 19-23, 2020, Proceedings*, volume 12302 of *Lecture Notes in Computer Science*, pages 343–359. Springer, 2020.

[12] Amjad Ibrahim, Simon Rehwald, and Alexander Pretschner. Efficient checking of actual causality with sat solving. In *Engineering Secure and Dependable Software Systems*, pages 241–255. IOS Press, 2019.

[13] David K Lewis. Counterfactuals. 1973.

[14] David K. Lewis. Counterfactual dependence and time's arrow. *Noûs*, 13(4):455–476, 1979.

[15] Daniel Özcan, Dalal Alrajeh, and Robert Craven. Reasoning about actual causality in answer set programming. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, volume 22, pages 610–620, 2025.

[16] Gesina Schwalbe and Bettina Finzel. A comprehensive taxonomy for explainable artificial intelligence: a systematic survey of surveys on methods and concepts. *DMKD*, 38(5):3043–3101, 2024.

[17] Joost Vennekens. Actual causation in cp-logic. *Theory and Practice of Logic Programming*, 11(4-5):647–662, 2011.