Proceedings of the 22nd International Conference on Principles of Knowledge Representation and Reasoning
KR in the Wild

Reasoning About Actual Causality in Answer Set Programming

Daniel Özcan , Dalal Alrajeh , Robert Craven
Imperial College London
danozcan@outlook.com, {dalal.alrajeh, robert.craven}@imperial.ac.uk

Abstract

Causal models provide a formal framework for identifying and reasoning about the causes of observed phenomena, making them valuable for decision-support contexts where understanding causality is essential. Yet applying these models in practice requires automated tools for key reasoning tasks. We present an Answer Set Programming (ASP)-based tool that supports three core capabilities for all acyclic binary causal models: (1) *checking* whether an event is an actual cause of another; (2) *finding* all minimal subsets of a failed candidate that do qualify as causes; and (3) *inferring* all actual causes of an outcome without assuming any candidate. Our tool is the first to support all three tasks within a unified framework, guaranteeing minimal contingency sets and outperforming prior implementations in both runtime and memory. We describe the system's design and report on an empirical evaluation using existing benchmarks.

1 Introduction

Evidence-based policy is essential in fields such as healthcare and policing, where effective interventions depend on understanding cause and effect (Pawson, 2006; Wikström and Sampson, 2006). Policymakers must assess both the causes of current problems and the likely consequences of proposed interventions—often amid uncertainty and limited data (Alrajeh, Chockler, and Halpern, 2020). For example, in addressing community violence, decision-makers may need to judge whether more foot patrols would deter crime, whether gaps in social services are fueling unrest, or whether neighbourhood dynamics are to blame.

Recent advances in the formal study of causality (Pearl, 2000; Halpern and Pearl, 2001, 2005; Halpern, 2015, 2016) have made it increasingly feasible to support decision-makers computationally. Central to this literature is the view that causal claims are best understood counterfactually: that "event X caused Y" means, roughly, "if X had not occurred, Y would not have occurred either." This idea is formalised through the machinery of *causal models*: mathematical objects that assign truth values to claims about causal and counterfactual relationships—statements such as "X caused Y" or "X would be false if we were to do Y" (Pearl, 2022).

A causal model, or *structural equations model*, represents a system as a collection of variables linked by equations. Each variable comes with a domain of possible values. Events then correspond to specific assignments of values to these variables. For example, in a voting scenario with four voters choosing between Billy and Suzy, we might define binary variables $V_1, \ldots, V_4, W$, where $V_i = 1$ if voter i votes for Suzy (0 otherwise), and $W = 1$ if Suzy wins (0 if Billy does). Variables are either *exogenous*, with values fixed by external factors, or *endogenous*, with values determined by the structural equations. Models are generally assumed to be *acyclic*, meaning that, once the values of the exogenous variables are fixed, the rest are uniquely determined, yielding a *causal setting*. Counterfactuals are then evaluated by modifying equations within that setting—for instance, if Suzy wins with three votes and all voters initially vote for her, intervening to set $V_1 = V_2 = 0$ causes her to lose. This machinery underlies the well-known Halpern–Pearl (HP) definitions of causation (Halpern and Pearl, 2001, 2005; Halpern, 2015), which hold that an event $X_1 = x_1 \wedge \ldots \wedge X_n = x_n$ is a cause of φ if, under certain *contingencies*, intervening to change the value of each X_i would prevent φ.

Causal modelling can be viewed from two perspectives: the *input side*—constructing an adequate model of the system—and the *output side*—answering queries about causal and counterfactual relationships between variables[1]. However, while much recent work has focused on causal discovery[2]—the extraction of causal models from data—comparatively little attention has been paid to the problem of *automated causal reasoning*. Yet this capability is essential if causal models are to serve in practical settings, including any high-stakes decision-support systems, where explainability is critical. Legal frameworks such as Article 22 of the EU's GDPR reflect this priority by asserting a "right to explanation," including access to "meaningful information about the logic involved" in automated decisions.[3]

[1] Results of causal queries may inform judgments about a model's adequacy, prompting revisions. Thus, the input and output sides interact. The point here is only to emphasise the distinct roles each plays in modelling; see (Halpern, 2016, Chapter 4).

[2] For a recent general survey, see Zanga, Ozkirimli, and Stella (2022). For ASP-based approaches in particular, see Russo, Rapberger, and Toni (2024); Zhalama et al. (2019); Hyttinen et al. (2017); Hyttinen, Eberhardt, and Järvisalo (2015, 2014)

[3] See Dexe et al. (2022); Goodman and Flaxman (2017); Fandinno and Schulz (2019) for relevant discussion.

Within the HP framework, causal solvers should not only judge the presence of causation but also identify the interventions and minimal contingency sets—the variables that must be fixed for the counterfactual to hold. Without this, the claim of causal dependence lacks full justification and the underlying logic remains opaque.

This paper presents a general-purpose system for automated causal reasoning in binary models—those in which variables take values in $\{0, 1\}$. Our system is grounded in the HP definition of actual causation and implemented in Answer Set Programming (ASP), a declarative paradigm for difficult search problems (Lifschitz, 2008). It handles three core query types: *Checking*, which determines whether a given event is a cause of another; *Finding*, which identifies sub-events of a non-cause that do qualify as causes; and *Inferring*, which enumerates all actual causes of a given outcome. In each case, the solver computes all solutions with minimal contingency sets, supporting transparency.

Our ASP encoding utilises the recent *asprin* system (Brewka et al., 2023), which supports flexible preference handling and is well-suited to the minimality requirements of HP causation. We release an open-source implementation in Python[4], and we provide an empirical evaluation showing that our system outperforms existing state-of-the-art solvers in Ibrahim, Rehwald, and Pretschner (2019) and Ibrahim and Pretschner (2020).

2 Background

2.1 Answer Set Programs

Syntax Symbolic constants, numerals and variables are terms. if $t_1 \ldots, t_n$ are terms and f is a symbolic constant, then $f(t_1, \ldots, t_n)$ is a functional term with $n \geq 1$. Terms are used to construct atomic propositions, or atoms. An atom is an expression of the form $p(t_1, \ldots, t_n)$ where p is a symbolic constant and $t_1, \ldots, t_n$ are terms with $n \geq 0$. A literal is an atom a, a positive literal, or its default negation $not\ a$, a negative literal. A conditional literal is an expression of the form $\ell : \ell_1 \wedge \ldots \wedge \ell_n$ where $\ell, \ell_1 \ldots, \ell_n$ are literals and $n \geq 0$. A choice atom is an expression of the form $\{a_1; \ldots; a_m\}$ where $m \geq 0$ and a_1 through a_m are conditional literals. A rule r is an expression of the form

$$h \leftarrow b_1 \wedge \ldots \wedge b_k \wedge \text{not } b_{k+1} \wedge \ldots \wedge \text{not } b_m \qquad (1)$$

where $m \geq 0$, h is an atom or a choice atom, and $b_1, \ldots, b_m$ are atoms. The *head* of a rule r is given by $H(r) = \{h\}$, the *body* by $B(r) = B^+(r) \cup B^-(r)$, where $B^+(r) = \{b_1, \ldots, b_k\}$ is the positive body and $B^-(r) = \{b_{k+1}, \ldots, b_m\}$, the negative body. If $B(r) = \emptyset$, r is called a fact; if $H(r) = \emptyset$, r is called a constraint, and viewed as shorthand for the rule "$x \leftarrow b_1 \wedge \ldots \wedge b_k \wedge not\ b_{k+1}, \ldots, \text{not } b_m, \text{not } x$" where x is a fresh atom not occurring elsewhere in the program. If h is an atom, r is a normal rule; otherwise, it is a choice rule. Choice rules are viewed as shorthand for a collection of normal rules according to the transformations in Gebser et al. (2022, 18–21).

[4]https://github.com/DanHOzcan/HP_ASPBinary.

A program is a collection of rules. An atom, rule or program is *ground* if it does not contain any variables and non-ground otherwise. A normal logic program is a program containing only normal rules.

Stable Model Semantics Given a program Π, its Herbrand Base $\mathcal{B}_\Pi$ consists of all symbolic constants occurring in Π. An interpretation I of Π is a subset of $\mathcal{B}_\Pi$. Satisfaction with respect to I is defined inductively: for an atom a, $I \models a$ if $a \in I$, and $I \models \text{not } a$ otherwise; for conjunctions of literals, $I \models \ell_1 \wedge \cdots \wedge \ell_n$ if $I \models \ell_1, \ldots, I \models \ell_n$; for disjunctions, $I \models \ell_1 \vee \cdots \vee \ell_n$ if $\{\ell_1, \ldots, \ell_n\} \cap I \neq \emptyset$; for a rule r of form (1), $I \models B(r)$ if $B^+(r) \subseteq I$ and $B^-(r) \cap I = \emptyset$; $I \models r$ if either $I \models H(r)$ or $I \not\models B(r)$. The semantics of a non-ground program Π is given by its ground instantiation $\mathrm{Gr}(\Pi)$, which is obtained by applying all substitutions from variables to constants in Π. An interpretation I satisfies a program Π if $I \models r$ for every $r \in \mathrm{Gr}(\Pi)$. An interpretation I is called a model of Π if $I \models \Pi$.

The Gelfond–Lifschitz reduct (Gelfond and Lifschitz, 1988, 1991) of a ground program Π with respect to $I \subseteq \mathcal{B}_\Pi$, denoted Π^I, is obtained by deleting all rules r with $B^-(r) \cap I \neq \emptyset$ and deleting the negative body from the remaining rules. A model I of Π is a stable model, or answer set, of Π if there is no proper subset of I that satisfies $\mathrm{Gr}(\Pi)^I$. Programs may have multiple, one or no answer sets. The set of answer sets is denoted $\mathrm{AS}(\Pi)$.

The *asprin* System The *asprin* system (Brewka et al., 2023) is a general framework for expressing qualitative and quantitative preferences over the answer sets of a logic program. We review the fragment of *asprin*'s input language used in this paper.

A *preference specification* consists of a finite set S of preference statements and a directive `#optimize`(s), where $s \in S$. Each preference statement has the form `#preference`$(s,t)\{e_1, \ldots, e_m\}$, where s is the name of the statement, t is a preference type, and each e_i is a *preference element*—a weighted formula of the form $t_1, \ldots, t_n : \varphi$, where φ is either a Boolean combination of classical atoms or a naming atom name(s') referring to another statement $s' \in S$. The specification is required to be *closed* ($s \in S$ whenever $name(s)$ occurs in S) and *acyclic* (the dependency relation induced by naming atoms is acyclic).

A preference statement `#preference`$(s,t)\{E\}$ declares a pre-order $\succeq_s$ over answer sets, where $Y \succ_s X$ if $Y \succeq_s X$ and $X \not\succeq_s Y$. To be admissible, E must be in the domain of t – the set of preference elements for which t is well-defined. We use two types:

- `subset`: $X \succeq_s Y$ iff
$$\{\ell \in E \mid X \models \ell\} \subseteq \{\ell \in E \mid Y \models \ell\}.$$
- `lexico`: $X \succeq_s Y$ iff
$$\bigvee_{w:\, \text{name}(s) \in E} \Big((X \succ_s Y) \wedge \bigwedge_{\substack{v:\, \text{name}(s') \in E \\ v > w}} (X =_{s'} Y) \Big)$$
$$\vee \bigwedge_{w:\, \text{name}(s) \in E} (X =_s Y).$$

where the domain of `subset` consists of sets of ground atoms occurring in Π and the domain of `lexico` consists of sets of preference elements of the form $w : \text{name}(s)$, where each name s appears with at most one associated weight, and each weight w applies to at most one name. An answer set X of a program Π is preferred with respect to $\succeq$ if there is no $Y \in \text{AS}(\Pi)$ with $Y \succ X$. We use $\text{AS}_s(\Pi)$ to denote the collection of answer sets preferred with respect to s.

Example 1. *Consider the following preference specification:*

```
#preference(p1, subset){a; b}.
#preference(p2, subset){c}.
#preference(p, lexico){
                1::name(p1); 2::name(p2)}.
#optimize(p).
```

Let $X = \{a, c\}$ and $Y = \{a, b, c\}$ be answer sets. Then $X \succ_{p_1} Y$ since $\{a\} \subset \{a, b\}$, and $X =_{p_2} Y$ since both include c. So, $(X \succ_{p_1} Y) \wedge (X =_{p_2} Y)$, meaning that $X \succeq_p Y$ and $Y \not\succeq_p X$. Therefore, $X \succ_p Y$.

2.2 Actual Causation

A standard distinction in the causality literature is between general (or type) causation and actual (or token) causation. The former concerns general claims such as "smoking causes lung cancer," while the latter concerns specific instances, e.g., "David's smoking caused his lung cancer." The three HP definitions (Halpern and Pearl, 2001, 2005; Halpern, 2015) are formal definitions of actual causation. Each attempts to address the perceived limitations of its predecessors. We adopt the most recent —*modified* —definition (Halpern, 2015), which is simpler, more robust to counterexamples, and yields lower complexity for causality *Checking*. We refer to it simply as "the HP definition". For a comprehensive treatment of actual causation, see Halpern (2016).

Causal Models Let $\mathcal{U}$ and $\mathcal{V}$ denote sets of exogenous and endogenous variables, respectively. Let $\mathcal{R}$ associate with each $Y \in \mathcal{U} \cup \mathcal{V}$ a finite set $\mathcal{R}(Y)$ of possible values called its *range*. We call $\mathcal{S} = (\mathcal{U}, \mathcal{V}, \mathcal{R})$ a *signature*. A causal model is a pair $M = (\mathcal{S}, \mathcal{F})$, where $\mathcal{S}$ is a signature and $\mathcal{F}$ assigns to each $X \in \mathcal{V}$ a function F_X, called a *structural equation*, where

$$F_X : \prod_{Y \in \mathcal{U} \cup \mathcal{V} - \{X\}} \mathcal{R}(Y) \to \mathcal{R}(X).$$

Y depends on X, or X is a parent of Y, if there exist values $x, x' \in \mathcal{R}(X)$ so that $F_Y(x, \vec{z}, \vec{u}) \neq F_Y(x', \vec{z}, \vec{u})$ where $\vec{z}$ is a setting of the variables in $\mathcal{V} - \{X, Y\}$ and $\vec{u}$ is a setting of the variables in $\mathcal{U}$. $Par(X)$, $Par_{exo}(X)$ and $Par_{endo}(X)$ denote the set of all parents of X, its exogenous parents and its endogenous parents, respectively. X affects Y if (X, Y) is in the transitive closure of the dependence relation. We will also say that a set X of variables affects Y if some variable in X affects Y.

Following the literature, we restrict our attention to *acyclic* (or *recursive*) models. In recursive models there is a partial ordering $\preceq$ of $\mathcal{V}$ such that, if X affects Y, then $X \preceq Y$. $\mathcal{M}^{rec}(\mathcal{S})$ denotes the set of acyclic models over $\mathcal{S}$. A model is binary if $\mathcal{R}(Y) = \{0, 1\}$ for all $Y \in \mathcal{U} \cup \mathcal{V}$. $\mathcal{M}^{bin}(\mathcal{S})$ denotes the set of binary acyclic models over $\mathcal{S}$.

If $\vec{X} = (X_1, \ldots, X_n)$ and $\vec{x} \in \Pi_{i \leq n} \mathcal{R}(X_i)$, we write $\vec{x} \in \mathcal{R}(\vec{X})$. If $\vec{Y} \subseteq \mathcal{U} \cup \mathcal{V}$, $\vec{y} \in \mathcal{R}(\vec{Y})$ and $\vec{X} \subseteq \vec{Y}$, $y|_{\vec{X}}$ denotes the restriction of $\vec{y}$ to $\vec{X}$. A *context* $\vec{u}$ is a setting of the exogenous variables. Let $\mathcal{C}(\mathcal{S}) = \prod_{Y \in \mathcal{U}} \mathcal{R}(Y)$ be the set of all contexts over $\mathcal{S}$. A pair $(M, \vec{u}) \in \mathcal{M}^{rec}(\mathcal{S}) \times \mathcal{C}(\mathcal{S})$ is a *causal setting*. Each causal setting has a unique solution—an assignment of values to each variable satisfying both the structural equations and the context. Structural equations let us consider what would happen if we intervened on certain variables in a given setting. Given a model M, if $\vec{X} \subseteq \mathcal{V}$ and $\vec{x} \in \mathcal{R}(\vec{X})$, we can construct a new model $M_{\vec{X} \leftarrow \vec{x}} = (\mathcal{S}, \mathcal{F}_{\vec{X} \leftarrow \vec{x}})$ which is identical to M except that the equation for each variable in X is replaced with $X = \vec{x}|_X$.

Syntax Let $\mathcal{S} = (\mathcal{U}, \mathcal{V}, \mathcal{R})$. For each $Y \in \mathcal{U} \cup \mathcal{V}$ and $y \in \mathcal{R}(Y)$, $Y = y$ is the primitive event which says that the variable Y takes on the value y. $\mathcal{L}(\mathcal{S})$ is the language consisting of all boolean combinations of these propositions and $\mathcal{L}^-(\mathcal{S})$, all such boolean combinations with no occurrences of exogenous variables.[5] Finally, a causal formula over $\mathcal{S}$ is an expression of the form $[Y_1 \leftarrow y_1, \ldots, Y_k \leftarrow y_k]\varphi$, where

- $\varphi \in \mathcal{L}^-(\mathcal{S})$;
- $Y_1 \ldots Y_k$ are distinct variables in $\mathcal{V}$; and
- $y_i \in \mathcal{R}(Y_i)$ for $1 \leq i \leq k$.

Such a formula says "after intervening to set the variables in $\vec{Y}$ to $\vec{y}$, φ holds" and is abbreviated with vector notation, becoming $[\vec{Y} \leftarrow \vec{y}]\varphi$. $\mathcal{L}^+(\mathcal{S})$ is the language consisting of all causal formulas over $\mathcal{S}$.

Semantics A causal formula $\varphi \in \mathcal{L}^+(\mathcal{S})$ is either true or false in a causal setting $(\mathcal{M}, \vec{u}) \in \mathcal{M}^{rec}(\mathcal{S}) \times \mathcal{C}(\mathcal{S})$. We write $(M, \vec{u}) \models \varphi$ if φ is true in $(M, \vec{u})$. $(M, \vec{u}) \models X = x$ if $X = x$ is in the unique solution to the system of equations defined by $\mathcal{F}$, in the context $\vec{u}$. The truth of conjunctions, negations and disjunctions is defined as usual. Finally, $(M, \vec{u}) \models [\vec{X} \leftarrow \vec{x}]\varphi$ if and only if $(M_{\vec{X} \leftarrow \vec{x}}, \vec{u}) \models \varphi$.

The HP Definition As noted in Section 1, counterfactual dependence is the starting point for the HP definitions. That is, we start with the so-called but-for test: *but for X, Y would not have occurred.* But counterfactual dependence alone does not always track actual causation. Consider the following example, well-described by Yablo (2002):

> Two friends throw rocks at a window; Suzy's rock hits, while Billy's sails harmlessly through the now empty frame. It was Suzy's throw, not Billy's, that caused the window to break. Yet counterfactually, the two throws seem on a par: had neither occurred, the window would not have broken; had either occurred without the other, it still would have.

[5] Generally, atomic propositions are taken to involve only endogenous variables. In our case it is useful to be able to include exogenous variables as well since we include them in the ASP encoding to model the setting.

The HP definitions belong to a broader class of views[6] that, in response such cases, replace bare counterfactual dependence with *de facto* (Yablo, 2002) or *contingent* (Halpern and Pearl, 2005) dependence: X caused Y if, had X not occurred, *and had certain suitably chosen factors been held fixed*, Y would not have occurred either (Hall and Paul, 2003, p. 104). In the structural equations framework, what we hold fixed are the values of variables in a given context. In other words, the but-for test is applied under certain contingencies.

As a first pass, in the Suzy–Billy case, we may hold fixed the fact that Billy's rock does not hit the window; under this assumption, if Suzy had not thrown, the window would not have shattered. The obvious issue is that this move can be applied in reverse: make it so Suzy does not throw, or does not hit, and then it holds that, had Billy not thrown, the window does not smash. Clearly, then, there must be constraints on what we may hold fixed. What are the permissible contingencies when evaluating a counterfactual? This is the question for all approaches based on de facto dependence and what divides the HP definitions.

Halpern's most recent answer (Halpern, 2015) is that we may hold fixed only the values of variables in the actual setting. To establish Billy as the cause, we would have to assume Suzy does not hit the window — contrary to fact. By contrast, establishing Suzy as the cause requires holding fixed what actually occurred. It is this asymmetry that justifies treating Suzy's throw, but not Billy's, as the cause. This is the core intuition behind the HP definition, given below. In this and subsequent definitions, a superscript "*" on a variable value x^* indicates that $(M, \vec{u}) \models X = x^*$.

Definition 1 (Actual Causation)**.** *$\vec{X} = \vec{x}$ is an actual cause of φ in $(M, \vec{u})$ iff the following hold:*

AC1. $(M, \vec{u}) \models \vec{X} = \vec{x} \wedge \varphi$.

AC2. *There exist $W \subseteq \mathcal{V}$ and $\vec{x}' \in \mathcal{R}(\vec{X})$ such that*

$$(M, \vec{u}) \models [\vec{X} \leftarrow \vec{x}', \vec{W} \leftarrow \vec{w}^*]\neg\varphi.$$

AC3. *There is no $X' \subset X$ such that $\vec{X}' = \vec{x}|_{\vec{X}'}$ satisfies both AC1 and AC2.*

If $X = x$ is a conjunct in $\vec{X} = \vec{x}$, we say that $X = x$ is in $\vec{X} = \vec{x}$. AC1 is a sanity check: one event cannot cause another if either of them did not occur in the first place. AC3 is a minimality condition, which excludes irrelevant events showing up as parts of causes: it was the arsonist dropping a lit match that caused the forest to burn down, not the arsonist dropping the lit match and, for example, sneezing. AC2, the core of the definition, encodes the qualified but-for test just described, where an empty W corresponds to bare counterfactual dependence.

Example 2 (Rock Throwing)**.** *We model the scenario with the model M_{RT} ("Rock Throwing"), with binary variables ST ("Suzy throws"), SH ("Suzy hits"), BT ("Billy throws"), BH ("Billy hits"), and WS ("window shatters"), and the following structural equations:*

$$SH = ST, \quad BH = BT \wedge \neg SH, \quad WS = SH \vee BH.$$

Exogenous parents of ST and BT are omitted. In the context where $(ST, BT) = (1, 1)$, we can test whether $ST = 1$ caused $WS = 1$ by setting $ST = 0$ and fixing $BH = 0$ (its actual value). This yields $WS = 0$, so AC2 holds with minimal contingency set $W = \{BH\}$. Since $(M_{RT}, (1, 1)) \models ST = 1 \wedge WS = 1$, AC1 holds, and since ST is a singleton, AC3 holds. Thus, $ST = 1$ is an actual cause of $WS = 1$ in $(M_{RT}, (1, 1))$. By contrast, $BT = 1$ is not: establishing counterfactual dependence requires setting ST or SH to 0, neither of which obtained in the actual setting.

Causal Queries In the Introduction, we introduced three kinds of reasoning over causal models: *Checking*, *Finding* and *Inferring*. Now that we have introduced the notation for causal models, we are in a position to define these tasks more precisely.

Definition 2 (Causal Queries)**.** *Fix a signature $\mathcal{S} = (\mathcal{U}, \mathcal{V}, \mathcal{R})$. Let $(M, \vec{u}) \in \mathcal{M}^{\text{rec}}(\mathcal{S}) \times \mathcal{C}(\mathcal{S}), \vec{X} \subseteq \mathcal{V}, \vec{x} \in \mathcal{R}(\vec{X}), \varphi \in \mathcal{L}^{-}(\mathcal{S})$.*

A causal query *$\mathcal{Q}$ is a quadruple*

$$(M, \vec{u},\ \vec{X} = \vec{x},\ \varphi),$$

and we distinguish two basic modes:

(i) ***Checking.*** *Decide whether $\vec{X} = \vec{x}$ is an actual cause of φ in $(M, \vec{u})$.*

(ii) ***Finding(k).*** *Given a bound $k \in N \cup \{\infty\}$, return up to k* restrictions *$\vec{Y} = \vec{x}|_{\vec{Y}}$ of $\vec{X} = \vec{x}$ such that $\vec{Y} = \vec{x}|_{\vec{Y}}$ is an actual cause of φ in $(M, \vec{u})$. When $k = \infty$, this amounts to Finding all causes within $\vec{X} = \vec{x}$.*

Finally, an ***Inference*** *query is a Finding query in which the candidate cause is $\vec{V} = \vec{v}$ where $(M, \vec{u}) \models \vec{V} = \vec{v}$ and V consists of all the endogenous variables that affect some variable in φ. If $M \in \mathcal{M}^{bin}(\mathcal{S})$, we call $\mathcal{Q}$ a binary query.*

2.3 Related Work

Ibrahim, Rehwald, and Pretschner (2019) were the first to automate actual causality queries in a general setting, introducing a SAT-based strategy, SAT_{Min}, for *Checking* causality in binary models. This method guarantees minimal contingency sets by exhaustively enumerating all satisfying assignments of the relevant SAT formula. Later, Ibrahim and Pretschner (2020) proposed partial MaxSAT and Integer Linear Programming (ILP) strategies for *Finding* causes. Both return a single cause, though without guaranteeing minimality of the contingency set. The MaxSAT variant was found to perform better in terms of runtime and memory allocation.

Ibrahim and Pretschner (2020) also introduced ILP_{Why}, an ILP-based strategy for *Inference*, which aims to return a single cause with a cardinality-minimal AC2 intervention. This approach partitions non-effect variables into cause, contingency, or normal classes, while excluding effect variables under the assumption that "variables that appear in the

[6] See, for example, Yablo (2002, 2004); Hitchcock (2001, 2007). The latter approach uses structural equations; the former does not.

effect formula cannot be part of the cause" (Ibrahim and Pretschner, 2020, 9).

However, this assumption leads to incorrect results when an effect variable appears in every AC2-satisfying intervention: $\mathrm{ILP}_{\mathrm{Why}}$ will report no cause—even when one exists.[7] Consequently, Theorem 2 in Ibrahim and Pretschner (2020)—which claims that the relevant formula G^* is satisfiable iff AC2 holds—is incorrect. Allowing effect variables in interventions would fix the problem, but introduces a different issue: if φ itself is the cardinality-minimal cause, then φ is returned as the cause of itself—an uninformative outcome, and likely what motivated the original restriction. Thus, although identifying a cardinality-minimal cause may be a reasonable objective when the details of the model are known, it is, in general, an inadequate formulation of the *Inference* problem.

3 ASP Encodings for the HP definition

Example 2 is illustrative in two respects. First, it is clear that the manual reasoning shown there will quickly become impractical and error-prone as causal models increase in size and queries grow in complexity. For small models, a simple brute-force algorithm may suffice to automate the process. However, Ibrahim (2021) implemented such an algorithm for *Checking* in binary models and found that it failed to scale effectively beyond models with five variables.

Second, identifying a *minimal* contingency set is necessary to fully understand why AC2 is satisfied. Suppose, for example, that a solver returns a *non-minimal* set such as $W = \{ST, BH, BT\}$[8]. Without further analysis, it is unclear which of these variables are genuinely necessary for satisfying AC2—in this case, only BH. With just three variables, one can resolve the ambiguity by manually testing each for relevance. But in larger models with potentially extensive contingency sets, such manual checks become infeasible. In the Evaluation section, for example, we mention a query which returns a minimal contingency set of over 600 variables. A solver that does not guarantee minimality therefore lacks the transparency required to explain why AC2 holds—the conceptual core of the HP definition. To ensure this transparency, a causal solver ought to return interventions with minimal contingency sets.

In this section, we introduce two ASP encodings that jointly automate and generalize the reasoning in Example 2 above while ensuring that every produced contingency set is minimal. The first encoding translates the actual setting of a query into a normal logic program, enabling the checking of AC1. The second encoding jointly models AC2 and AC3 as a logic program with preferences. Together, these encodings give a unified approach to *Checking*, *Finding*, and *Inferring* actual causes in any binary causal model.

Notation

AC1 We start by regimenting the syntactic form of structural equations we allow as input to our tool and describing their translation into ASP.

Let $F_{\mathbf{x}}$ be a structural equation in a binary causal model. $F_{\mathbf{x}}$ will be characterised symbolically as follows:

$$\mathbf{x} = \psi_1 \vee \ldots \vee \psi_n \ (n \geq 1), \tag{2}$$

where each ψ_i is a conjunction such that each conjunct is either a variable in the model or the negation of one. We assume that only parents of $\mathbf{x}$ appear in $F_{\mathbf{x}}$. The equations in Example 2 follow this format, where variables in a causal model are identified with primitive propositions in propositional logic. In this format, any Boolean function can be translated into ASP by having a separate rule for each disjunct.

Let $F_{\mathbf{x}}$ be a structural equation in a binary model written in the form above. Then $\mathcal{T}$, which translates structural equations into fragments of ASP rules, is defined as follows.

(i) $\mathcal{T}(\mathbf{x}) = \mathbf{x}$;

(ii) $\mathcal{T}(\neg\mathbf{x}) = \mathrm{not}\,\mathbf{x}$;

(iii) if each of $\chi_1 \ldots \chi_n$ is a variable or the negation of one, then $\mathcal{T}(\chi_1 \wedge \cdots \wedge \chi_n) = \mathcal{T}(\chi_1) \wedge \ldots \wedge \mathcal{T}(\chi_n)$.

For example, the formula $\mathbf{bt} \wedge \neg\mathbf{sh}$ is translated as $\mathcal{T}(\mathbf{bt} \wedge \neg\mathbf{sh}) = \mathbf{bt} \wedge \mathrm{not}\,\mathbf{sh}$. We now define the full translation of a binary causal setting into a normal logic program. To keep track of disjunctive structural equations, we use superscripts on disjuncts: $\psi_i^{\mathbf{x}}$ denotes the i-th disjunct in the structural equation for $\mathbf{x}$, and $n_{\mathbf{x}}$ denotes the number of disjuncts in the structural equation for $\mathbf{x}$.

Definition 3 ($\Pi_{(M,\vec{u})}$)**.** *Fix a signature $\mathcal{S} = (\mathcal{U}, \mathcal{V}, \mathcal{R})$, and Let $(M, \vec{u}) \in \mathcal{M}^{bin}(\mathcal{S}) \times \mathcal{C}(\mathcal{S})$. Then $\Pi_{(M,\vec{u})}$ is defined as follows.*

$$\{\mathbf{u} \mid \mathbf{u} \in \mathcal{U}, (M, \vec{u}) \models \mathbf{u} = 1\} \tag{3}$$

$$\cup \{\mathbf{x} \leftarrow \mathcal{T}(\psi_i^{\mathbf{x}}) \mid \mathbf{x} \in \mathcal{V}, i \in [1, n_{\mathbf{x}}]\} \tag{4}$$

Example 3 ($\Pi_{(M_{RT},(1,1))}$)**.** *The encoding of $(M_{RT}, (1,1))$, denoted $\Pi_{(M_{RT},(1,1))}$, is given below.*

$\mathbf{st}_{exo}$	$\mathbf{sh} \leftarrow \mathbf{st}$
$\mathbf{bt}_{exo}$	$\mathbf{bh} \leftarrow \mathbf{bt} \wedge \mathrm{not}\ \mathbf{sh}$
$\mathbf{st} \leftarrow \mathbf{st}_{exo}$	$\mathbf{ws} \leftarrow \mathbf{sh}$
$\mathbf{bt} \leftarrow \mathbf{bt}_{exo}$	$\mathbf{ws} \leftarrow \mathbf{bh}$

It can easily be verified that this program has exactly one answer set: namely, $\{\mathbf{st}_{exo}, \mathbf{bt}_{exo}, \mathbf{st}, \mathbf{bt}, \mathbf{sh}, \mathbf{ws}\}$, whose members are exactly the variables that are true in $(M_{RT}, (1,1))$. Proposition 1 generalises this observation.

Proposition 1. *Fix a signature $\mathcal{S} = (\mathcal{U}, \mathcal{V}, \mathcal{R})$, and let $(M, \vec{u}) \in \mathcal{M}^{bin}(\mathcal{S}) \times \mathcal{C}(\mathcal{S})$. Then there exists exactly one $I \in \mathrm{AS}(\Pi_{(M,\vec{u})})$, $I_{(M,\vec{u})}$, such that*

$$I_{(M,\vec{u})} = \{\mathbf{x} \mid \mathbf{x} \in \mathcal{U} \cup \mathcal{V}, (M, \vec{u}) \models \mathbf{x} = 1\}.$$

[7] Let $\varphi \equiv WS = 1 \vee BH = 0$. There are three actual causes of φ in $(M_{RT}, (1,1))$: $ST = 1 \wedge WS = 1$, $SH = 1 \wedge WS = 1$, and $BH = 0 \wedge WS = 1$. The formula G^* in Ibrahim and Pretschner (2020) fails to identify any cause, as it forbids interventions on effect variables.

[8] The SAT strategy in Ibrahim, Rehwald, and Pretschner (2019) defines the contingency set of an AC2-satisfying intervention as the set of all variables that retain their actual value under the intervention.

Proof. See Özcan, Alrajeh, and Craven (2025). □

The correspondence between each $(M, \vec{u}) \in \mathcal{M}^{bin}(\mathcal{S}) \times \mathcal{C}(\mathcal{S})$ and its encoding $\Pi_{(M,\vec{u})}$ allows us to check AC1 with respect to a candidate cause $\vec{\mathbf{x}} = \vec{x}$ and effect φ, as well as all restrictions $\vec{\mathbf{y}} = \vec{x}|_{\vec{\mathbf{y}}}$. For example, if the candidate cause of $\mathbf{ws} = 1$ is $\mathbf{sh} = 1 \wedge \mathbf{bh} = 1$, then we can check AC1 by checking whether $\mathbf{sh}, \mathbf{bh} \in I_{(M_{RT},(1,1))}$ and $\mathbf{ws} \in I_{(M_{RT},(1,1))}$, which in turn means that $(M_{RT}, (1,1)) \models \mathbf{sh} = 1 \wedge \mathbf{bh} = 1 \wedge \mathbf{ws} = 1$ by Proposition 1. Since $\mathbf{sh}, \mathbf{ws} \in I_{(M_{RT},(1,1))}$ but $\mathbf{bh} \notin I_{(M_{RT},(1,1))}$, we know that AC1 is not satisfied with respect to $\mathbf{sh} = 1 \wedge \mathbf{bh} = 1$, but it is satisfied with respect to the restriction $\mathbf{sh} = 1$.

AC2 In the following two sections, we introduce the *logic program with preferences* $(\Pi_{\mathrm{AC2}(\mathcal{Q})}, S_{\min})$, whose preferred answer sets correspond to the AC2-satisfying interventions that are minimal with respect to AC2 and the contingency set. The following definitions formalise the correspondence we aim for.

Definition 4 (Correspondence 1)**.** *Let I be a set of ground atoms $\left(M_{\vec{\mathbf{y}}\leftarrow\vec{y},\ \vec{\mathbf{w}}\leftarrow\vec{w}^*}, \vec{u}\right)$ be a causal setting. We say that I and $\left(M_{\vec{\mathbf{y}}\leftarrow\vec{y},\ \vec{\mathbf{w}}\leftarrow\vec{w}^*}, \vec{u}\right)$ correspond to each other, written $I \cong (M_{\vec{\mathbf{y}}\leftarrow\vec{y},\vec{\mathbf{w}}\leftarrow\vec{w}^*}, \vec{u})$, if, for all $\mathbf{x} \in \mathcal{V}$ and all exogenous parents of mixed-dependency variables, the following conditions hold:*

(i) $\mathrm{in\_x}(\mathbf{y}, |1-y|) \in I$ *iff* $\mathbf{y} = y$ *is in* $\vec{\mathbf{y}} \leftarrow \vec{y}$*;*
(ii) $\mathrm{in\_w}(\mathbf{w}, |1-w|) \in I$ *iff* $\mathbf{w} = w$ *is in* $\vec{\mathbf{w}} \leftarrow \vec{w}^*$*;*
(iii) $I \models \mathbf{x}$ *iff* $(M_{\vec{\mathbf{y}}\leftarrow\vec{y},\vec{\mathbf{w}}\leftarrow\vec{w}^*}, \vec{u}) \models \mathbf{x} = 1$.

Example 4. *Let* $I_1 = \{in\_x(\mathbf{sh}, 1), in\_w(\mathbf{bh}, 0), \mathbf{bt}, \mathbf{st}\}$ *and* $I_2 = \{in\_x(\mathbf{st}, 1), in\_w(\mathbf{bh}, 0), \mathbf{bt}\}$ *Then* $I_1 \cong (M_{RT_{\mathbf{sh}\leftarrow 0,\mathbf{bh}\leftarrow\mathbf{0}},}, (1,1))$ *and* $I_2 \cong (M_{RT_{\mathbf{st}\leftarrow 0,\mathbf{bh}\leftarrow\mathbf{0}},}, (1,1))$

Definition 5 (Correspondence 2)**.** *Let $\mathcal{A}$ be a collection of answer sets and let S be a collection of causal settings. We say that $\mathcal{A}$ corresponds to S, written $\mathcal{A} \cong S$, if and only if:*

(i) for every $I \in \mathcal{A}$, there exists a $(M_{\vec{\mathbf{y}}\leftarrow\vec{y},\vec{\mathbf{w}}\leftarrow\vec{w}^*}, \vec{u}) \in S$ such that $I \cong (M_{\vec{\mathbf{y}}\leftarrow\vec{y},\vec{\mathbf{w}}\leftarrow\vec{w}^*}, \vec{u})$;
(ii) for every $(M_{\vec{\mathbf{y}}\leftarrow\vec{y},\vec{\mathbf{w}}\leftarrow\vec{w}^*}, \vec{u}) \in S$, there exists an $I \in \mathcal{A}$ such that $I \cong (M_{\vec{\mathbf{y}}\leftarrow\vec{y},\vec{\mathbf{w}}\leftarrow\vec{w}^*}, \vec{u})$.

Note that $\mathcal{A} \cong S$ implies that $|\mathcal{A}| = |S|$.[9]

Example 5. *Let $\mathcal{A} = \{I_1, I_2\}$ (see Example 4). Then $\mathcal{A} \cong \{(M_{RT_{\mathbf{sh}\leftarrow 0,\mathbf{bh}\leftarrow\mathbf{0}},}, (1,1)), (M_{RT_{\mathbf{st}\leftarrow 0,\mathbf{bh}\leftarrow\mathbf{0}}}, (1,1))\}$.*

Reduct Returning to the query in Example 2, note that AC2 reasoning focuses only on variables relevant to the intervention under consideration: in that case, $\mathbf{sh}$, $\mathbf{bh}$ and $\mathbf{ws}$. Variables upstream of $\mathbf{sh}$ and $\mathbf{bh}$ ($\mathbf{st}$, $\mathbf{bt}$, $\mathbf{st}_{exo}$, $\mathbf{bt}_{exo}$), or downstream of $\mathbf{ws}$, are not considered. In effect, then, when reasoning about AC2 we operated with a reduced model containing only the endogenous variables $\mathbf{sh}, \mathbf{bh}, \mathbf{ws}$ and the exogenous variables $\mathbf{st}, \mathbf{bt}$. We formalise this feature of AC2-reasoning with the *reduct of a setting with respect to a causal query*, writing $Par^M(\mathbf{x})$ for the parents of $\mathbf{x}$ in M and $Var(\varphi)$ for the variables appearing in φ.

Definition 6 ($(M^{\mathcal{Q}}, \vec{u}^{\mathcal{Q}})$)**.** *Given a causal model $M = ((\mathcal{U}, \mathcal{V}, \mathcal{R}), \mathcal{F})$ and a query $\mathcal{Q} = (M, \vec{u}, \vec{\mathbf{x}} = \vec{x}, \varphi)$, we define the reduct of $(M, \vec{u})$ with respect to $\mathcal{Q}$, denoted $(M^{\mathcal{Q}}, \vec{u}^{\mathcal{Q}})$, as follows.*

- $\mathcal{V}^{\mathcal{Q}} = \{\mathbf{y} \in \mathcal{V} \mid \mathbf{x}$ *affects* $\mathbf{y}$, $\mathbf{y}$ *affects* $Var(\varphi)\}$
- $\mathcal{U}^{\mathcal{Q}} = \{\mathbf{u} \in \mathcal{U} \cup \mathcal{V} \mid \neg(\mathbf{x}$ *affects* $\mathbf{u}), \exists \mathbf{y} \in \mathcal{V}^{\mathcal{Q}}(\mathbf{u} \in Par^M(\mathbf{x}))\}$
- $\mathcal{R}^{\mathcal{Q}} = \mathcal{R}\big|_{\mathcal{U}^{\mathcal{Q}}\cup\mathcal{V}^{\mathcal{Q}}}$, $\mathcal{F}^{\mathcal{Q}} = \mathcal{F}\big|_{\mathcal{V}^{\mathcal{Q}}}$, *and* $(M, \vec{u}) \models \vec{\mathbf{u}}^{\mathcal{Q}} = \vec{u}^{\mathcal{Q}}$.

For example, if $\mathcal{Q} = (M_{RT}, (1,1), \mathbf{bh} = 0, \mathbf{ws} = 1)$, then $\mathcal{V}^{\mathcal{Q}} = \{\mathbf{sh}, \mathbf{bh}, \mathbf{ws}\}, \mathcal{U}^{\mathcal{Q}} = \{\mathbf{st}, \mathbf{bt}\}$ and both $\mathcal{R}$ and $\mathcal{F}$ are adjusted accordingly. Proposition 2, whose proof is straightforward, justifies employing the reduct in place of the full setting.

Proposition 2. *For any query $\mathcal{Q} = (M, \vec{u}, \vec{\mathbf{x}} = \vec{x}, \varphi)$, $\vec{\mathbf{x}} = \vec{x}$ is an actual cause of φ in $(M, \vec{u})$ if and only if it is an actual cause of φ in $(M^{\mathcal{Q}}, \vec{u}^{\mathcal{Q}})$.*

Checking AC2 using the reduct has two advantages: (1) structural equations irrelevant to the query are ignored, and (2) variables that are either unaffected by $\mathbf{x}$ or that don't affect $Var(\varphi)$ are removed as candidates for a minimal contingency set. For small models such as M_{RT}, employing the reduct will make little difference, but in large models the savings can be substantial. In our dataset, for instance, some queries on models with about 8,000 variables reduced to roughly 700 relevant variables. Because contingency interventions are encoded via choice rules (see $\Pi_{\mathrm{AC2\_rules}(\mathcal{Q})}$), this pruning avoids generating more than 14,000 redundant rules.

AC2 By Proposition 2, AC2 holds if there exist $x' \in \mathcal{R}(\vec{\mathbf{x}})$ and $\vec{\mathbf{w}} \subseteq \mathcal{V}^{\mathcal{Q}}$ such that

$$(M^{\mathcal{Q}}_{\vec{\mathbf{x}}\leftarrow\vec{x}',\vec{\mathbf{w}}\leftarrow\vec{w}^*}, \vec{u}^{\mathcal{Q}}) \models \neg\varphi.$$

Modelling this condition requires representing its three components: (1) the reduct $(M^{\mathcal{Q}}, \vec{u}^{\mathcal{Q}})$; (2) the possible interventions on $\vec{\mathbf{x}}$ and $\vec{\mathbf{w}}$; and (3) the formula $\neg\varphi$. $\Pi_{\mathrm{AC2}(\mathcal{Q})}$, introduced in this section, models each with a distinct subprogram: $\Pi_{(M^{\mathcal{Q}},\vec{u}^{\mathcal{Q}})}$ for the reduct, $\Pi_{\mathrm{AC2\_rules}}(\mathcal{Q})$ for interventions, and $\Pi_{\neg\varphi}$ for enforcing the negated effect. We describe each below.

$\Pi_{(M^{\mathcal{Q}},\vec{u}^{\mathcal{Q}})}$, given in Definition 7, is similar to $\Pi_{(M,\vec{u})}$, with three key modifications. First, anticipating (2), we define the candidate contingency set $\vec{\mathbf{w}}^{\mathcal{Q}}$ by removing from $\mathcal{V}^{\mathcal{Q}}$ all variables without endogenous parents, since such variables cannot belong to a minimal contingency set. Second, we omit structural equations for first-level endogenous variables, whose values in $(M^{\mathcal{Q}}, \vec{u}^{\mathcal{Q}})$ are already known from $I_{(M,\vec{u})}$. Third, for variables that may participate in an AC2-satisfying intervention, we append exceptions to their structural equation rules, allowing the solver to override them when needed. Conditions $\mathbf{C1}$–$\mathbf{C4}$ allow us to specify precisely when such exceptions are introduced.

[9] This way of formulating a correspondence definition is inspired by the argumentation literature. See: Egly, Gaggl, and Woltran (2010); Brewka et al. (2020); Dvořák et al. (2011); Gaggl et al. (2015).

Definition 7 ($\Pi_{(M^{\mathcal{Q}},\vec{u}^{\mathcal{Q}})}$)**.** *Let $\mathcal{Q} = (M, \vec{u}, \vec{\mathbf{x}} = \vec{x}, \varphi)$. For any variable $\mathbf{x}$ in $M^{\mathcal{Q}}$, define:*

$\mathbf{C1_x}: Par^{M^{\mathcal{Q}}}_{endo}(\mathbf{x}) = \emptyset$ $\quad$ $\mathbf{C3_x}:$ $\mathbf{x} = 1$ *is in* $\vec{\mathbf{x}} = \vec{x}$

$\mathbf{C2_x}: (M^{\mathcal{Q}}, \vec{u}^{\mathcal{Q}}) \models \mathbf{x} = 1$ $\quad$ $\mathbf{C4_x}:$ $\mathbf{x} = 0$ *is in* $\vec{\mathbf{w}}_{\mathcal{Q}} = \vec{w}^*_{\mathcal{Q}}$

Then $\Pi_{(M^{\mathcal{Q}},\vec{u}^{\mathcal{Q}})}$ is defined as:

$$\{\mathbf{u} \mid \exists \mathbf{x} \in \mathcal{V}(\ \mathbf{u} \in Par_{exo}(\mathbf{x}), Par_{endo}(\mathbf{x}) \neq \emptyset), \mathbf{C2_x}\} \quad (5)$$

$$\cup \{\mathbf{x} \mid \mathbf{C1_x}, \mathbf{C2_x}, \neg\, \mathbf{C3_x}\} \quad (6)$$

$$\cup \{\mathbf{x} \leftarrow \text{not in\_x}(\mathbf{x}, 1) \mid \mathbf{C1_x}, \mathbf{C2_x}, \boldsymbol{C3_x}\} \quad (7)$$

$$\cup \{\mathbf{x} \leftarrow \mathcal{T}(\psi_i^{\mathbf{x}}), \text{not in\_x}(\mathbf{x}, 1) \mid \neg\mathbf{C1_x}, \mathbf{C2_x}, \mathbf{C3_x}\} \quad (8)$$

$$\cup \{\mathbf{x} \leftarrow \mathcal{T}(\psi_i^{\mathbf{x}}), \text{not in\_w}(\mathbf{x}, 0) \mid \mathbf{C4_x}\} \quad (9)$$

$$\cup \{\mathbf{x} \leftarrow \mathcal{T}(\psi_i^{\mathbf{x}}) \mid \neg\mathbf{C1_x}, \neg\mathbf{C4_x}, (\neg\mathbf{C2_x} \vee \neg\mathbf{C3_x})\} \quad (10)$$

Example 6 ($\Pi_{(M^{\mathcal{Q}}_{RT},(1,1)^{\mathcal{Q}})}$)**.** *Let*

$$\mathcal{Q} = (M_{RT}, (1,1), \mathbf{sh} = 1, \mathbf{bh} = 0 \wedge \mathbf{ws} = 1).$$

Then $\Pi_{(M^{\mathcal{Q}}_{RT},(1,1)^{\mathcal{Q}})}$ is given below.

$\mathbf{bh} \leftarrow \mathbf{bt} \wedge \text{not}\, \mathbf{sh} \wedge \text{not in\_w}(\mathbf{bh}, 0)$ $\quad$ $\mathbf{bt}$

$\mathbf{sh} \leftarrow \text{not in\_x}(\mathbf{sh}, 1)$ $\quad$ $\mathbf{ws} \leftarrow \mathbf{sh}$

$\mathbf{ws} \leftarrow \mathbf{bh}$

Note that $\mathbf{bh}$ *is a mixed-dependency variable—it depends on the endogenous* $\mathbf{sh}$ *and the newly exogenous* $\mathbf{bt}$*—so we include its equation and add* $\mathbf{bt}$ *as a fact. Note also that both* $\mathbf{ws}$ *and* $\mathbf{bh}$ *remain candidates for the minimal contingency set, despite being part of the effect. This is crucial: as we know from Example 2,* $\mathbf{bh}$ *is included in the minimal contingency set for the intervention on* $\mathbf{sh}$*. This illustrates the point made in Related Work section: effect variables must not be excluded from candidate contingency sets when Finding, as opposed to Checking.*

$\Pi_{\text{AC2\_rules}}(\mathcal{Q})$ is defined in (11)–(14) below. It uses choice rules to represent the space of possible interventions on $\mathbf{x}$ and $\mathbf{w}$.

$$\{\{\text{in\_x}(\mathbf{x}, x)\} \mid \mathbf{x} = x \text{ is in } \vec{\mathbf{x}} = \vec{x}\} \quad (11)$$

$$\cup \{\{\text{in\_w}(\mathbf{w}, w)\} \mid \mathbf{w} = w \text{ is in } \vec{\mathbf{w}}_{\mathcal{Q}} = \vec{w}^*_{\mathcal{Q}}\} \quad (12)$$

$$\cup \{\mathbf{x} \leftarrow \text{in\_x}(\mathbf{x}, 0) \mid \mathbf{x} = 0 \text{ is in } \vec{\mathbf{x}} = \vec{x}\} \quad (13)$$

$$\cup \{\mathbf{w} \leftarrow \text{in\_w}(\mathbf{w}, 1) \mid \mathbf{w} = 1 \text{ is in } \vec{\mathbf{w}}_{\mathcal{Q}} = \vec{w}^*_{\mathcal{Q}}\}. \quad (14)$$

Finally, if $\varphi \equiv \psi_1 \vee \cdots \vee \psi_m$, $\Pi_{\neg\varphi}$ enforces the negation of φ by including a constraint for each of its disjuncts:

$$\Pi_{\neg\varphi} = \{\leftarrow \mathcal{T}(\psi_i) \mid i \in [1, m]\}. \quad (15)$$

The combined encoding is:

$$\Pi_{\text{AC2}(\mathcal{Q})} = \Pi_{\text{AC2}(M^{\mathcal{Q}},\vec{u}^{\mathcal{Q}})} \cup \Pi_{\text{AC2\_rules}(\mathcal{Q})} \cup \Pi_{\neg\varphi}. \quad (16)$$

We now define the collection of AC2-satisfying settings that $\text{AS}(\Pi_{\text{AC2}(\mathcal{Q})})$ corresponds to.

Definition 8 ($\text{AC2}(\mathcal{Q})$)**.** *Let $\mathcal{Q} = (M, \vec{u}, \vec{\mathbf{x}} = \vec{x}, \varphi)$ be a causal query. Then we define $\text{AC2}(\mathcal{Q})$ as the set of all settings $\left(M^{\mathcal{Q}}_{\vec{\mathbf{y}}\leftarrow\vec{y},\ \vec{\mathbf{w}}\leftarrow\vec{w}^*},\ \vec{u}^{\mathcal{Q}}\right)$ such that the following hold:*

(i) $\left(M^{\mathcal{Q}}_{\vec{\mathbf{y}}\leftarrow\vec{y},\ \vec{\mathbf{w}}\leftarrow\vec{w}^*},\ \vec{u}^{\mathcal{Q}}\right) \models \neg\varphi$*;*

(ii) $\mathbf{y} \subseteq \mathbf{x}$ *and* $\vec{y} \in \mathcal{R}(\vec{\mathbf{y}})$*;*

(iii) $\mathbf{w} \subseteq \mathbf{w}^{\mathcal{Q}}$*;*

(iv) for each $\mathbf{y}_i \in \vec{\mathbf{y}}$*:*

$$\mathbf{y}_i = y_i \text{ is in } \vec{\mathbf{y}} = \vec{y} \text{ iff } \mathbf{y}_i = |1 - y_i| \text{ is in } \vec{\mathbf{x}} = \vec{x}.$$

Condition (i) is just the condition for AC2. The remaining conditions rule out irrelevant interventions. In particular, the second part of condition (ii) excludes the empty intervention, which will satisfy AC2 trivially if φ is false in the actual setting. In our tool, we do not initiate the AC2 program if $I_{(M,\vec{u})} \not\models \varphi$. Condition (iv), anticipating AC3, requires that any part of the candidate cause included in the intervention be negated.[10]

Example 7. *Let* $\mathcal{Q} = (M_{RT}, (1,1), \mathbf{st} = 1 \wedge \mathbf{sh} = 1, \mathbf{ws} = 1)$*. Then*

$$\text{AC2}(\mathcal{Q}) = \left\{ \begin{array}{l} (M^{\mathcal{Q}}_{RT_{\mathbf{sh}\leftarrow 0, \mathbf{bh}\leftarrow 0}}, (1,1)), \\ (M^{\mathcal{Q}}_{RT_{\mathbf{st}\leftarrow 0, \mathbf{bh}\leftarrow 0}}, (1,1)) \end{array} \right\}$$

Proposition 3. *For any binary query $\mathcal{Q}$,*

$$\text{AS}(\Pi_{\text{AC2}(\mathcal{Q})}) \cong \text{AC2}(\mathcal{Q}).$$

Proof. Özcan, Alrajeh, and Craven (2025). □

AC3 Given Proposition 3, it is straightforward with *asprin* to identify interventions minimal with respect to both AC2 and the contingency set. We need three preference statements: (1) a `subset` preference over `in_x/2` atoms; (2) a `subset` preference over `in_w/2` atoms; and (3) a `lexico` preference that minimises `in_x/2` atoms before `in_w/2` atoms. An accompanying `#optimize` directive completes the specification S_{min}, shown below.

```
#preference(min_x, subset)in_x(X,Y).
#preference(min_w, subset)in_w(X,Y).
#preference(min, lexico)
    2::name(min_x), 1::name(min_w).
#optimize(min).
```

Define $\text{AC2}_{\min}(\mathcal{Q})$ so that $\left(M^{\mathcal{Q}}_{\vec{\mathbf{y}}\leftarrow\vec{y},\ \vec{\mathbf{w}}\leftarrow\vec{w}^*}, \vec{u}^{\mathcal{Q}}\right) \in \text{AC2}_{\min}(\mathcal{Q})$ iff $\left(M^{\mathcal{Q}}_{\vec{\mathbf{y}}\leftarrow\vec{y},\ \vec{\mathbf{w}}\leftarrow\vec{w}^*}, \vec{u}^{\mathcal{Q}}\right) \in \text{AC2}(\mathcal{Q})$ and the the following conditions hold.

- $\neg\exists \mathbf{y}' \subset y$ such that $\left(M_{\vec{\mathbf{y}'}\leftarrow\vec{y}|_{\vec{\mathbf{y}'}},\ \vec{\mathbf{w}}\leftarrow\vec{w}^*}, \vec{u}\right) \in \text{AC2}(\mathcal{Q})$;
- $\neg\exists \mathbf{w}' \subset \mathbf{w}$ such that $\left(M_{\vec{\mathbf{y}}\leftarrow\vec{y},\ \vec{\mathbf{w}'}\leftarrow\vec{w}^*_{|\vec{\mathbf{w}'}}}, \vec{u}\right) \in \text{AC2}(\mathcal{Q})$.

Proposition 4. *For any binary causal query $\mathcal{Q}$,*

$$\text{AS}_{\min}(\Pi_{\text{AC2}(\mathcal{Q})}) \cong \text{AC2}_{\min}(\mathcal{Q}).$$

Proof. See Özcan, Alrajeh, and Craven (2025). □

Propositions 5 and 6 establish the soundness of using $\Pi_{(M,\vec{u})}$ and $(\Pi_{\text{AC2}(\mathcal{Q})}, S_{\min})$ to answering *Finding* and *Checking* queries. Since *Inferring* is a special case of *Finding*, these results extend to *Inference*. In practice, a *Checking* query can be resolved by instructing *asprin* to generate at most one preferred answer set (via `--models=1`)

[10] See Ibrahim, Rehwald, and Pretschner (2019, Lemma 1).

and verifying whether it matches the candidate cause. From here, we distinguish two tool variants: $\mathrm{ASP}_{\mathrm{Check}}$, which returns `True` with the associated minimal contingency set if the candidate cause satisfies HP, and `False` otherwise; and $\mathrm{ASP}_{\mathrm{Find}}$.

Proposition 5 (Finding Causes)**.** *For any binary query $\mathcal{Q} = (M, \vec{u}, \vec{\mathbf{x}} = \vec{x}, \varphi)$ and for any $\mathbf{y} \subseteq \mathbf{x}$, $\vec{\mathbf{y}} = \vec{x}|_{\vec{\mathbf{y}}}$ is an actual cause of φ in $(M, \vec{u})$ if and only if*

(i) $I_{(M,\vec{u})} \models \mathcal{T}\big(\vec{\mathbf{y}} = \vec{x}|_{\vec{\mathbf{y}}}\big) \wedge \mathcal{T}(\varphi)$*;*

(ii) there is some $I \in \mathrm{AS}_{\min}(\Pi_{\mathrm{AC2}(\mathcal{Q})})$ *such that* $\mathrm{in\_x}(\mathbf{x}, x) \in I$ *if and only if* $\mathbf{x} = x$ *is a part of* $\vec{\mathbf{y}} = \vec{x}|_{\vec{\mathbf{y}}}$.

Proof. See Özcan, Alrajeh, and Craven (2025). □

Proposition 6 (Checking Causes)**.** *For any binary query $\mathcal{Q} = (M, \vec{u}, \vec{\mathbf{x}} = \vec{x}, \varphi)$ $\vec{\mathbf{x}} = \vec{x}$ is an actual cause of φ in $(M, \vec{u})$ if and only if*

(i) $I_{(M,\vec{u})} \models \mathcal{T}\big(\vec{\mathbf{x}} = \vec{x}\big) \wedge \mathcal{T}(\varphi)$*;*

(ii) there is exactly one $I \in \mathrm{AS}_{\min}(\Pi_{\mathrm{AC2}(\mathcal{Q})})$ *such that* $\mathrm{in\_x}(\mathbf{x}, x) \in I$ *if and only if* $\mathbf{x} = x$ *is a part of* $\vec{\mathbf{x}} = \vec{x}$.

Proof. See Özcan, Alrajeh, and Craven (2025). □

Non-binary Models The framework developed in this section admits a natural extension to the non-binary setting. We replace propositional atoms with atoms of the form $val(\mathbf{x}, x)$, which appear in an answer set when $\mathbf{x} = x$. Structural equations are modified accordingly. Next, we introduce `range/2` atoms, where $range(\mathbf{x}, x)$ appears in answer set if $x \in \mathcal{R}(\mathbf{x})$. Interventions on cause variables can then be expressed as follows:

$$\{val(\mathbf{x}, X) : range(\mathbf{x}, X) \wedge X \neq v\} \leftarrow in\_cause(\mathbf{x}, v).$$[11]

This encoding supports both binary and non-binary variables. However, queries involving variables with large numerical domains will quickly make grounding infeasible (Gebser et al., 2018). In future work we will explore systems like *clingcon* (Banbara et al., 2017) for more efficient reasoning over such models.

4 Evaluation

In this section, we evaluate the performance of our tool in comparison to the current state-of-the-art approaches, namely those presented in Ibrahim, Rehwald, and Pretschner (2019) and Ibrahim and Pretschner (2020), discussed in the Related Work section. The evaluation spans three core tasks: *Checking*, *Finding*, and *Inferring* causes. We focus on two key performance criteria: (1) the ability to process queries within the memory constraints typical of standard computing environments, and (2) execution time relative to practical usability thresholds.

[11]Note that this encoding requires us to allow arithmetic terms into the language described in Section 2.1, in addition to functional terms. To represent linear equations, we would also introduce comparison operators and so-called built-in atoms (see Calimeri et al. (2020) for the definition of these).

Query Size	1	2	3	4	5	10	11	15	20	22	50
Frequency	195	131	95	27	8	8	10	8	8	2	8

Table 1: Frequency of Query Sizes for *Checking*/*Finding* queries.

Benchmark To enable a direct comparison with the Java-implemented SAT, MaxSAT, and ILP strategies evaluated by Ibrahim, Rehwald, and Pretschner (2019) and Ibrahim and Pretschner (2020), we adopt the same benchmark suite. Specifically, we use a total of 500[12] *Checking*/*Finding* queries and 187 *Inferring* queries, spanning 37 binary models of varying sizes. These include 21 smaller models with fewer than 400 endogenous variables and 16 larger models with up to approximately 8,000 variables. Table 1 shows the breakdown of query sizes across *Checking*/*Finding* queries.

Full details of the benchmark models are available at: https://git.io/Jf8iH.

Evaluation Metrics and Research Questions We evaluate our tool using two primary performance metrics: *execution time* and *peak memory usage*. Our evaluation is guided by the following research questions:

RQ1. Is our ASP-based tool able to successfully answer all queries in the benchmark within a given time and memory budget?

RQ2. For queries successfully answered by both toolsets, does our ASP-based tool outperform the Java-based strategies in execution time and memory usage?

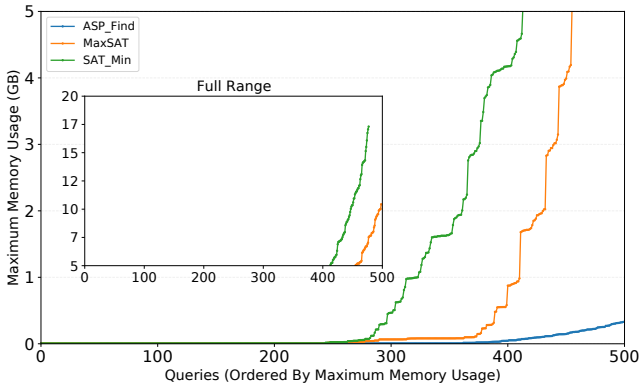
Execution Environment All experiments were conducted on macOS 14.5 with an Apple M1 Pro (32 GB RAM). Our ASP-based tool used Clingo v5.7.1 with asprin v3.1.1 (internally Clingo v5.4.0) under a 25 GB memory cap. Time limits were set to two hours for *Checking* queries (based on preliminary runs of the longest cases) and 10 minutes for *Inference* queries. Java-based tools were tested in two batches: one measuring execution time (10 warm-up runs, 5 measured runs, average reported), and one measuring peak memory usage (single execution, maximum reported). The ASP tool was executed once per query, reporting single-shot time and memory usage.

Due to the error in $\mathrm{ILP}_{\mathrm{Why}}$ identified in the Related Work section, we checked whether there were any queries it would produce an incorrect result for. No such queries were found, indicating that the comparisons with $\mathrm{ILP}_{\mathrm{Why}}$ remain valid.

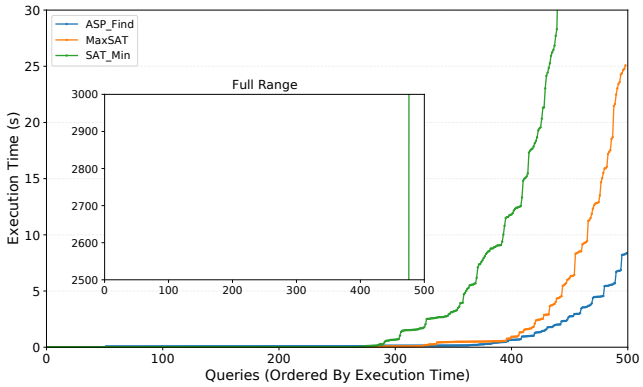
4.1 Results

RQ1 Both $\mathrm{ASP}_{\mathrm{Check}}$ and $\mathrm{ASP}_{\mathrm{Find}}$ answered all 500 *Checking* and *Finding* queries within the allotted time. $\mathrm{SAT}_{\mathrm{Min}}$ failed on 22 out of 500 queries, mostly due to running out of memory. MaxSAT failed on just one query. However, given the speed with which it completed the majority of queries and the relatively short 10-minute cutoff, the remaining query would likely have been solved

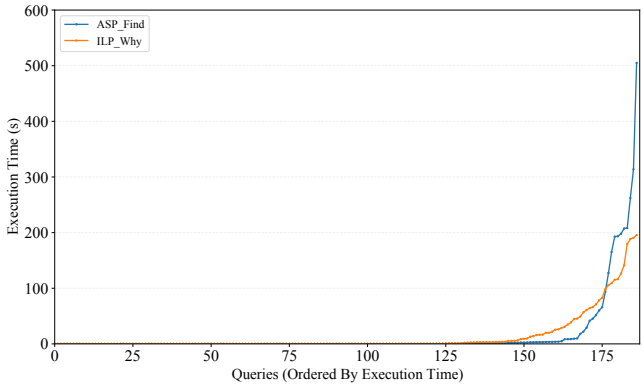
[12]Ibrahim and Pretschner (2020) reports 484 queries, but we found 500 in the authors' benchmarking repository.



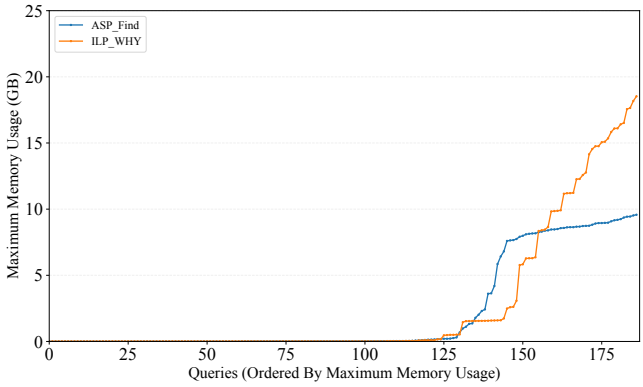
(a) Memory usage for *Checking*/*Finding* queries



(b) Execution times for *Checking*/*Finding* queries



(c) Execution times for *Inference* queries



(d) Memory usage for *Inference* queries

Figure 1: Cactus plots for *Checking* and *Inference* queries.

with slightly more time. For *Inference*, both $\mathrm{ASP}_{\mathrm{Find}}$ and $\mathrm{ILP}_{\mathrm{Why}}$ successfully answered all 187 queries.

RQ2 Figure 1b reports execution times for $\mathrm{ASP}_{\mathrm{Find}}$, $\mathrm{SAT}_{\mathrm{Min}}$, and MaxSAT (we omit $\mathrm{ASP}_{\mathrm{Check}}$, which behaves identically to $\mathrm{ASP}_{\mathrm{Find}}$). $\mathrm{ASP}_{\mathrm{Find}}$ solves all 500 queries in under 10 seconds, while using less than 1 GB of memory (Figure 1a). MaxSAT, which returns at most one cause without enforcing minimality, takes up to 25 seconds and peaks at 10 GB of memory. $\mathrm{SAT}_{\mathrm{Min}}$, though guaranteeing minimality, performs substantially worse, often failing due to memory exhaustion and with the two slowest runs exceeding 50 minutes with memory usage reaching 17 GB.

Figure 1d compares *Inference* performance between $\mathrm{ASP}_{\mathrm{Find}}$ and $\mathrm{ILP}_{\mathrm{Why}}$. As with *Finding* queries, $\mathrm{ASP}_{\mathrm{Find}}$ uses less memory ($\leq$ 10GB vs. 18GB) while solving the harder task of enumerating all causes with minimal contingency sets. In some cases, for example, it finds multiple causes, each with contingency sets exceeding 600 variables. The increased memory usage relative to pure *Finding* reflects the higher complexity of the *Inference* task. Figure 1c shows that $\mathrm{ILP}_{\mathrm{Why}}$ slightly outperforms $\mathrm{ASP}_{\mathrm{Find}}$ on execution time, which solves all but the hardest query in 100–300 seconds, with one outlier at 504 seconds.

5 Conclusion

In this work, we introduced an ASP encoding of the modified HP definition of actual causation, supporting *Checking*, *Finding*, and *Inferring* queries over all binary acyclic causal models while guaranteeing minimal contingency sets. We demonstrated the tool's time and memory efficiency through experimental evaluation, marking a step toward practical causal reasoning tools.

As future work, we plan to extend the tool to handle non-binary causal models and develop an interactive graphical interface to make the tool more accessible and interactive. Additionally, we intend to explore mechanisms for the tool to learn from users' causal interpretations, supporting collaborative reasoning, negotiation, and discussion about possible interventions. We plan to evaluate these enhancements with crime analysts investigating serial sexual offending.

Acknowledgements

This work has been partially supported by the CHEDDAR funded by the UK EPSRC under grant numbers EP/Y037421/1 and EP/X040518/1.

References

Alrajeh, D.; Chockler, H.; and Halpern, J. Y. 2020. Combining experts' causal judgments. *Artificial Intelligence* 288:103355.

Banbara, M.; Kaufmann, B.; Ostrowski, M.; and Schaub, T. 2017. Clingcon: The next generation. *Theory and Practice of Logic Programming* 17(4):408–461.

Brewka, G.; Diller, M.; Heissenberger, G.; Linsbichler, T.; and Woltran, S. 2020. Solving advanced argumentation problems with answer set programming. *Theory and Practice of Logic Programming* 20(3):391–431.

Brewka, G.; Delgrande, J.; Romero, J.; and Schaub, T. 2023. A general framework for preferences in answer set programming. *Artificial Intelligence* 325:104023.

Calimeri, F.; Faber, W.; Gebser, M.; Ianni, G.; Kaminski, R.; Krennwallner, T.; Leone, N.; Maratea, M.; Ricca, F.; and Schaub, T. 2020. Asp-core-2 input language format. *Theory and Practice of Logic Programming* 20(2):294–309.

Dexe, J.; Franke, U.; Söderlund, K.; van Berkel, N.; Jensen, R. H.; Lepinkäinen, N.; and Vaiste, J. 2022. Explaining automated decision-making: a multinational study of the gdpr right to meaningful information. *The Geneva Papers on Risk and Insurance–Issues and Practice* 47(3):669–697.

Dvořák, W.; Gaggl, S. A.; Wallner, J. P.; and Woltran, S. 2011. Making use of advances in answer-set programming for abstract argumentation systems. In *Applications of Declarative Programming and Knowledge Management*. Berlin, Heidelberg: Springer. 114–133.

Egly, U.; Gaggl, S. A.; and Woltran, S. 2010. Answer-set programming encodings for argumentation frameworks. *Argument & Computation* 1(2):147–177.

Fandinno, J., and Schulz, C. 2019. Answering the "why" in answer set programming – a survey of explanation approaches. *Theory and Practice of Logic Programming* 19(2):114–203.

Gaggl, S. A.; Manthey, N.; Ronca, A.; Wallner, J. P.; and Woltran, S. 2015. Improved answer-set programming encodings for abstract argumentation. *Theory and Practice of Logic Programming* 15(4–5):434–448.

Gebser, M.; Leone, N.; Maratea, M.; Perri, S.; Ricca, F.; and Schaub, T. 2018. Evaluation techniques and systems for answer set programming: a survey. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI 2018)*, 5450–5456.

Gebser, M.; Kaminski, R.; Kaufmann, B.; and Schaub, T. 2022. *Answer Set Solving in Practice*. Springer Nature.

Gelfond, M., and Lifschitz, V. 1988. The stable model semantics for logic programming. In *Logic Programming: Proceedings of the Fifth International Conference and Symposium*, 1070–1080. Cambridge: MIT Press.

Gelfond, M., and Lifschitz, V. 1991. Classical negation in logic programs and disjunctive databases. *New Generation Computing* 9:365–385.

Goodman, B., and Flaxman, S. 2017. European union regulations on algorithmic decision-making and a "right to explanation". *AI Magazine* 38(3):50–57.

Hall, N., and Paul, L. A. 2003. Causation and preemption. In *Philosophy of Science Today*. Oxford: Clarendon Press. 100–130.

Halpern, J. Y., and Pearl, J. 2001. Causes and explanations: A structural-model approach. part i: Causes. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence (UAI 2001)*, 194–202.

Halpern, J. Y., and Pearl, J. 2005. Causes and explanations: A structural-model approach. part ii: Explanations. *The British Journal for the Philosophy of Science*.

Halpern, J. Y. 2015. A modification of the halpern–pearl definition of causality. arXiv preprint arXiv:1505.00162.

Halpern, J. Y. 2016. *Actual Causality*. MIT Press.

Hitchcock, C. 2001. The intransitivity of causation revealed in equations and graphs. *The Journal of Philosophy* 98(6):273–299.

Hitchcock, C. 2007. Prevention, preemption, and the principle of sufficient reason. *Philosophical Review* 116:495–532.

Hyttinen, A.; Plis, S. M.; Järvisalo, M.; Eberhardt, F.; and Danks, D. 2017. A constraint optimization approach to causal discovery from subsampled time series data. *International Journal of Approximate Reasoning* 90:208–225.

Hyttinen, A.; Eberhardt, F.; and Järvisalo, M. 2014. Constraint-based causal discovery: Conflict resolution with answer set programming. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, 340–349.

Hyttinen, A.; Eberhardt, F.; and Järvisalo, M. 2015. Do-calculus when the true graph is unknown. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, 395–404.

Ibrahim, A., and Pretschner, A. 2020. From checking to inference: Actual causality computations as optimization problems. In *Proceedings of the International Symposium on Automated Technology for Verification and Analysis (ATVA)*, 343–359. Springer International Publishing.

Ibrahim, A.; Rehwald, S.; and Pretschner, A. 2019. Efficient checking of actual causality with sat solving. In *Engineering Secure and Dependable Software Systems*. IOS Press. 241–255.

Ibrahim, A. 2021. *An actual causality framework for accountable systems*. Ph.D. Dissertation, Technische Universität München.

Lifschitz, V. 2008. What is answer set programming? In *Proceedings of the Twenty-Third National Conference on Artificial Intelligence (AAAI 2008)*, 1594–1597. Chicago, IL: AAAI Press.

Özcan, D. H.; Alrajeh, D.; and Craven, R. 2025. Reasoning about actual causality in answer set programming:

Extended version. Extended version of the paper published in the Proceedings of KR 2025 (KR in the Wild track). Available at https://github.com/DanHOzcan/HP_ASPBinary/releases/download/v1.0.0/KR_Extended.pdf.

Pawson, R. 2006. Evidence based policy: proceed with care. *BMJ* 332(7539):582–585.

Pearl, J. 2000. *Causality: Models, Reasoning, and Inference*. Cambridge University Press.

Pearl, J. 2022. Probabilities of causation: three counterfactual interpretations and their identification. In *Probabilistic and Causal Inference: The Works of Judea Pearl*. 317–372.

Russo, F.; Rapberger, A.; and Toni, F. 2024. Argumentative causal discovery. arXiv preprint arXiv:2405.11250.

Wikström, P.-O. H., and Sampson, R. J., eds. 2006. *The Explanation of Crime: Context, Mechanisms and Development*. Cambridge University Press.

Yablo, S. 2002. De facto dependence. *The Journal of Philosophy* 99(3):130–148.

Yablo, S. 2004. Advertisement for a sketch of an outline of a prototheory of causation. In *Causation and Counterfactuals*. MIT Press. 119–137.

Zanga, A.; Ozkirimli, E.; and Stella, F. 2022. A survey on causal discovery: Theory and practice. *International Journal of Approximate Reasoning* 151:101–129.

Zhalama; Zhang, J.; Eberhardt, F.; Mayer, W.; and Li, M. J. 2019. ASP-based discovery of semi-markovian causal models under weaker assumptions. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI-19)*, 1488–1494.