Proceedings of the 21st International Conference on Principles of Knowledge Representation and Reasoning
Reasoning, Learning & Decision-Making Track

Argumentative Causal Discovery

Fabrizio Russo , **Anna Rapberger** , **Francesca Toni**
Imperial College London, Department of Computing
{fabrizio, a.rapberger, ft}@imperial.ac.uk

Abstract

Causal discovery amounts to unearthing causal relationships amongst features in data. It is a crucial companion to causal inference, necessary to build scientific knowledge without resorting to expensive or impossible randomised control trials. In this paper, we explore how reasoning with symbolic representations can support causal discovery. Specifically, we deploy *assumption-based argumentation (ABA)*, a well-established and powerful knowledge representation formalism, in combination with *causality theories*, to learn graphs which reflect causal dependencies in the data. We prove that our method exhibits desirable properties, notably that, under natural conditions, it can retrieve ground-truth causal graphs. We also conduct experiments with an implementation of our method in *answer set programming (ASP)* on four datasets from standard benchmarks in causal discovery, showing that our method compares well against established baselines.

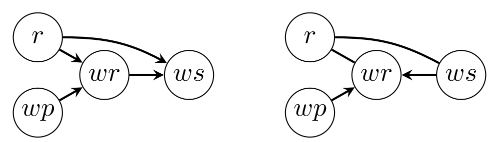
1 Introduction

Causal Discovery is the process of extracting causal relationships amongst variables in data, represented as graphs. These graphs are crucial for understanding causal effects and perform causal inference (Peters, Janzing, and Schölkopf, 2017; Pearl, 2009; Spirtes, Glymour, and Scheines, 2000), e.g. to determine the impact of an action or treatment on an outcome. Causal effects are ideally discovered through interventions or randomised control trials, but these can be expensive, time consuming or outright impossible, e.g. in healthcare, trying to establish whether smoking causes cancer through a randomised control trial would require the study group to take up smoking to measure its (potentially deadly) effect. Hence the need to use observational, as opposed to interventional, data to study causes and effects (Peters, Janzing, and Schölkopf, 2017; Schölkopf et al., 2021).

Prominent approaches to perform causal discovery include constraint-based, score-based and functional causal model-based methods (see e.g. (Glymour, Zhang, and Spirtes, 2019; Vowels, Camgoz, and Bowden, 2022; Zanga, Ozkirimli, and Stella, 2022) for overviews). These approaches employ statistical methods to retrieve the causal relations between variables. However, statistical methods, even if consistent with infinite data, are prone to errors due to finite data. As a result, the extracted causal relations can deviate from the ground truth and, crucially, also from the observed data. Let us consider an example.

Example 1.1. *We set out to discover the causal relations between* rain (r)*,* wet roof terrace (wr)*,* wet street (ws) *and* watering plants (wp) *(on the roof terrace). After collecting sufficient data, we carry out conditional independence tests. These correctly return that r and wp are independent (written $r \perp\!\!\!\perp wp$); but also find r and wp independent when conditioned on $\{wr\}$ (written $r \perp\!\!\!\perp wp \mid \{wr\}$) which goes against our intuition: since something must have caused wr, we can infer r when knowing $\neg wp$ and vice versa. That is, r and wp become dependent when conditioning on $\{wr\}$.*

Below, we depict the ground truth causal graph (left) and the output of Majority-PC (right), proven to be sound and complete with infinite data (Colombo and Maathuis, 2014).



A directed edge is interpreted as cause, e.g., wp causes wr; the absence of an edge indicates causal independence; an undirected edge indicates a causal relationship but the direction of the cause and effect relation remains unclear.

Since the conditional independence test wrongly rendered r and wp independent given $\{wr\}$, it is impossible to retrieve the ground truth whilst satisfying all reported causal relations between the variables. In fact, it can happen that no graph exists that faithfully captures the results of the tests.

To account for the issues observed in the example, researchers have investigated several methods to handle conflicting data; e.g., Corander et al. (2013) utilised Answer Set Programming (ASP) to learn chordal Markov networks; Hyttinen, Eberhardt, and Järvisalo (2014) provide an encoding of graphical interventions to compute causal graphs; Rantanen, Hyttinen, and Järvisalo (2020) use constraint programming. However, argumentative methods, which are ideally suited for conflict resolution, have not received much attention in the context of causal discovery so far. A notable exception is the work by Bromberg and Margaritis (2009) who employ a form of preference-based argumentation (Amgoud and Cayrol, 2002), instantiated with deductive argumentation (see (Philippe Besnard, 2018) for an overview), to choose a set of tests to use within the PC algorithm (Spirtes, Glymour, and Scheines, 2000). Their method

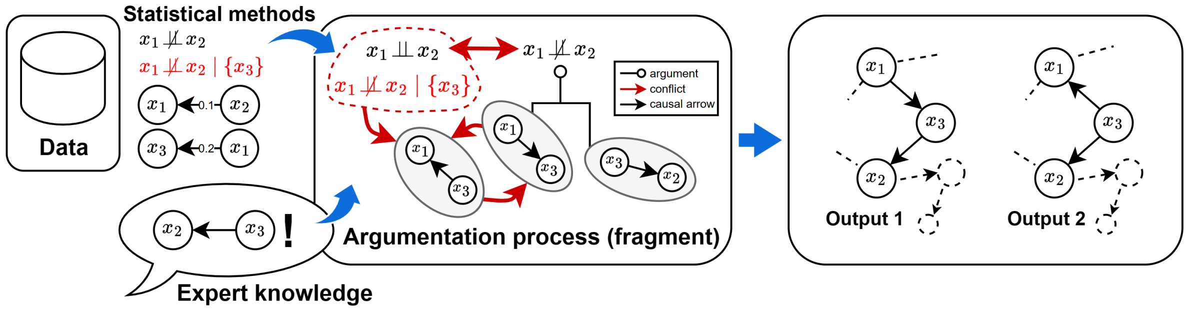


Figure 1: Overview of the workflow of our *Causal ABA algorithm*, which combines statistical methods and expert domain knowledge with non-monotonic reasoning and performs argumentative reasoning to output causal graphs consistent with the reported causal relationships.

is however based on Pearl's graphoid axioms which are incomplete; thus, some inconsistencies between the reported (in)dependences might not be detected by their approach.

In this paper, we provide a novel argumentative approach to account for inconsistencies in the reported tests and reflect a consistent subset of them into a *directed acyclic graph (DAG)*. In line with the causal discovery literature, we assume *faithfulness* of the data, i.e., that all the independencies in the data are compatible with some DAG structure (Spirtes, Glymour, and Scheines, 2000) as well as *sufficiency* i.e. there are no latent confounders. To handle conflicts in data, we employ *assumption-based argumentation (ABA)* which is a versatile non-monotonic reasoning formalism (Čyras et al., 2018) based on assumptions (i.e., defeasible elements) and inference rules. ABA has been studied under numerous semantics, which are criteria to determine the acceptance of assumption sets and their conclusions. A single ABA framework can possess several different extensions, i.e., sets of acceptable assumptions w.r.t. a given semantics, which reflect the different viewpoints that exist within a single framework.

Fig. 1 summarises the workflow of our method. Based on (i) the output of statistical methods and (ii) domain knowledge provided by experts, we construct an ABA framework whose extensions provide all the DAGs compatible with (i) and (ii). Overall, our contributions are as follows:

- We formalise causal graphs in the language of ABA (*Causal ABA*). We use rules to model the d-separation criterion (Pearl, 2009), which characterises conditional independence in DAGs; and assumptions to model conditional independence and causal relations.
- We provide an ASP implementation of our theoretical framework using the independence tests from the Majority-PC algorithm (Colombo and Maathuis, 2014) as hard or weak constraints, resulting in *ABA-PC*. We employ weights for fact selection when necessary.
- We experimentally evaluate our ABA-PC algorithm with four (standard) datasets. Our experiments show that our proposed framework improves on current state-of-the-art baselines in Causal Discovery. In particular, we reconstruct the ground-truth causal DAG better than Majority-PC using the same set of independence relations.

A long version of this article including supplementary material can be found in (Russo, Rapberger, and Toni, 2024).

2 Preliminaries

Graphs are crucial for causal and argumentation theories. A graph $G = (\mathbf{V}, E)$ has nodes $\mathbf{V}$ and edges $E \subseteq \mathbf{V} \times \mathbf{V}$; G is *directed* if either $(x, y) \in E$ or $(y, x) \in E$; *undirected* if $(x, y) \in E$ and $(y, x) \in E$; and *partially directed* otherwise. The *skeleton* of G is the result of replacing all directed edges with undirected ones. $x, y \in \mathbf{V}$ are *adjacent* iff $(x, y) \in E$ or $(y, x) \in E$. A *(x_1-x_n-)path path* is a sequence of distinct nodes $x_1 \dots x_n$ s.t. for $1 \leq i < n$, x_i and x_{i+1} are adjacent. We omit 'x_1-x_n' if it is clear from the context. Given a path $\mathcal{p} = x_1 \dots x_n$ and a node x, we sometimes abuse notation and write $x \in \mathcal{p}$ to specify that x is contained in $\mathcal{p}$, i.e., there is $i \leq n$ s.t. $x = x_i$. A path $x_1 \dots x_n$ is *directed* if $(x_i, x_{i+1}) \in E$ for all $i \leq n$; *cyclic* if it is directed and $x_1 = x_n$. A *directed acyclic graph (DAG)* is a directed graph without cycles.

2.1 Causal Graphs

A causal graph represents causal relations between variables (Pearl, 2009; Spirtes, Glymour, and Scheines, 2000). In this paper, we focus on causal graphs that admit a DAG structure. Pearl's *d-separation criterion* establishes the link between DAGs and conditional independence.

Conditional (In)dependence We consider a finite set of *variables* $\mathbf{V}$. For pairwise disjoint sets $\mathbf{X}, \mathbf{Y}, \mathbf{Z} \subseteq \mathbf{V}$ we let $(\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z})$ indicate that $\mathbf{X}$ and $\mathbf{Y}$ are *independent* given the conditioning set $\mathbf{Z}$; $(\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \emptyset)$ is simply written as $(\mathbf{X} \perp\!\!\!\perp \mathbf{Y})$ and singleton sets $\{x\}$ are denoted by x (e.g., $(\{x\} \perp\!\!\!\perp \{y\} \mid \emptyset)$ is written as $(x \perp\!\!\!\perp y)$. Also, $(\mathbf{X} \not\perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z})$ means that $\mathbf{X}$ and $\mathbf{Y}$ are *dependent* given $\mathbf{Z}$. A fundamental property of conditional independence is symmetry (Pearl and Paz, 1986); we identify $(\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z})$ and $(\mathbf{Y} \perp\!\!\!\perp \mathbf{X} \mid \mathbf{Z})$ with each other (analogously for dependence statements).

A triple (x_i, x_j, x_k) of variables in a DAG is an *Unshielded Triple (UT)* if two variables are not adjacent but each is adjacent to the third. An UT (x, y, z) is a *v-structure* iff $(x, y) \in E$ and $(z, y) \in E$; y is a *collider* (w.r.t. x, z).

Definition 2.1. *Let $G = (\mathbf{V}, E)$ be a DAG. A x-y-path $\mathcal{p}$, $x, y \in \mathbf{V}$, $x \neq y$, is* $\mathbf{Z}$-active *for a set $\mathbf{Z} \subseteq \mathbf{V} \setminus \{x, y\}$ in G iff for each node $z \in \mathcal{p}$: if z is a collider in $\mathcal{p}$, then $z \in \mathbf{Z}$ or there is a descendant z' of z s.t. $z' \in \mathbf{Z}$; otherwise, $z \notin \mathbf{Z}$.*

Two variables $x, y \in \mathbf{V}$ are *independent*, conditioned on a set $\mathbf{Z} \subseteq \mathbf{V} \setminus \{x, y\}$, if fixing the values of the variables in $\mathbf{Z}$ does not provide additional information about x or y (resp.). Independence in DAGs is captured by *d-separation*.

Definition 2.2. *Let $G = (\mathbf{V}, E)$ be a DAG. Two nodes $x, y \in \mathbf{V}$ are* d-connected *given $\mathbf{Z} \subseteq \mathbf{V}$ iff G contains a $\mathbf{Z}$-active x-y-path $\not p$. The nodes $x, y \in \mathbf{V}$ are* d-separated *given $\mathbf{Z}$ iff x, y are not d-connected given $\mathbf{Z}$. Two variables x, y are* independent *w.r.t. $\mathbf{Z}$ in G iff they are d-separated given $\mathbf{Z}$, denoted by $x \perp\!\!\!\perp_G y \mid \mathbf{Z}$.*

Causal Graphs and Statistics Causal Discovery couples statistical and graphical methods to extract causal graphs from data. The nodes $\mathbf{V} = \{X_1, \ldots, X_d\}$ in a causal graph $G = (\mathbf{V}, E)$ correspond to random variables (in our running Example 1.1, 'rain' can be a random variable when associated with observed data) and the edges represent causal relationships between them. A joint probability distribution P factorizes according to a DAG G if $P(\mathbf{V}) = \prod_{i=1}^{d} P(X_i \mid \mathrm{pa}(G, X_i))$, where $\mathrm{pa}(G, X_i)$ denotes the set of parents of X_i in G. A distribution P is *Markovian* w.r.t. G if it respects the conditional independence relations entailed by G via d-separation. In turn, P is *faithful* to G if DAG G reflects all conditional independences in P. Different DAGs can imply the same set of conditional independences, in which case they form a Markov Equivalence Class (MEC) (Richardson and Spirtes, 1999). DAGs in a MEC present the same adjacencies and v-structures and are uniquely represented by a *Completed Partially* DAG (CPDAG) (Chickering, 2002) which is a partially directed graph that has a directed edge if every DAG in the MEC has it, and an undirected edge if both directions appear in the MEC.

A *Conditional Independence Test (CIT)*, e.g. Fisher's Z (Fisher, 1970), HSIC (Gretton et al., 2007), or KCI (Zhang et al., 2011), is a procedure to measure independence with a known asymptotic distribution under the null hypothesis $\mathcal{H}_0$ of independence. Calculating the test statistic for a dataset allows to estimate the test's observed significance level (p-value), under $\mathcal{H}_0$. This is a measure of evidence against $\mathcal{H}_0$ (Hung et al., 1997). Under $\mathcal{H}_0$, p is uniformly distributed in the interval $[0, 1]$, which allows to set a significance level α that represents the pre-experiment Type I error rate (rejecting $\mathcal{H}_0$ when it is true), whose expected value is at most α. A CIT, denoted by $I(X_i, X_j \mid \mathbf{Z})$, outputs a p-value. If $I(X_i, X_j \mid \mathbf{Z}) = p \geq \alpha$ then $X_i \perp\!\!\!\perp X_j \mid \mathbf{Z}$. Instead, if $I(X_i, X_j \mid \mathbf{Z}) = p < \alpha$ then we can reject $\mathcal{H}_0$ and declare the variables dependent: $X_i \not\perp\!\!\!\perp X_j \mid \mathbf{Z}$.

2.2 Assumption-based Argumentation

We recall basics of assumption-based argumentation (ABA); for a comprehensive introduction we refer to (Čyras et al., 2018). We assume a deductive system $(\mathcal{L}, \mathcal{R})$, where $\mathcal{L}$ is a formal language, i.e., a set of sentences, and $\mathcal{R}$ is a set of rules over $\mathcal{L}$. A rule $r \in \mathcal{R}$ has the form $a_0 \leftarrow a_1, \ldots, a_n$ with $a_i \in \mathcal{L}$, $head(r) = a_0$ and $body(r) = \{a_1, \ldots, a_n\}$.

Definition 2.3. *An ABA framework (ABAF) is a tuple $(\mathcal{L}, \mathcal{R}, \mathcal{A}, ^-)$, where $(\mathcal{L}, \mathcal{R})$ is a* deductive system*, $\mathcal{A} \subseteq \mathcal{L}$ a set of* assumptions*, and $^- : \mathcal{A} \to \mathcal{L}$ is a function mapping assumptions $a \in \mathcal{A}$ to sentences $\mathcal{L}$ (contrary function).*

A sentence $q \in \mathcal{L}$ is *tree-derivable* from $S \subseteq \mathcal{A}$ and rules $R \subseteq \mathcal{R}$, denoted by $S \vdash_R q$, if there is a finite rooted labeled tree T which, intuitively, corresponds to the structure of the derivation: the root of T is labeled with q; the set of labels for the leaves of T is equal to S or $S \cup \{\top\}$; and for every inner node v of T there is a rule $r \in R$ such that v is labelled with $head(r)$, the number of successors of v is $|body(r)|$ and every successor of v is labelled with a distinct $a \in body(r)$ or $\top$ if $body(r) = \emptyset$. We often drop R and write $S \vdash_R q$ simply as $S \vdash q$ if it does not cause confusion.

Definition 2.4. *Let $D = (\mathcal{L}, \mathcal{R}, \mathcal{A}, ^-)$ be an ABAF. A set $S \subseteq \mathcal{A}$* attacks *$T \subseteq \mathcal{A}$ if there is $S' \subseteq S$, $a \in T$, s.t. $S' \vdash \overline{a}$. A set S is* conflict-free *in an ABAF D ($S \in cf(D)$) if it does not attack itself; S* defends *T iff it attacks each attacker of T; S is* closed *iff $S \vdash a$ implies $a \in S$; S is* admissible *($S \in ad(D)$) if it is conflict-free and defends itself.*

With a little notational abuse we say a set S of assumptions attacks an assumption a if S attacks the singleton $\{a\}$; we let $\overline{S} = \{\overline{a} \mid a \in S\}$.

An ABAF D is called *flat* iff each set S of assumptions is closed. We call an ABAF *non-flat* if it does not belong to the class of flat ABAFs.

We next recall grounded, complete, preferred, and stable ABA semantics (abbr. gr, co, pr, stb).

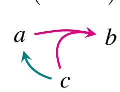
Definition 2.5. *Let D be an ABAF and let $S \in ad(D)$.*

- *$S \in co(D)$ iff S contains every assumption set it defends;*
- *$S \in gr(D)$ iff S is $\subseteq$-minimal in $co(D)$;*
- *$S \in pr(D)$ iff S is $\subseteq$-maximal in $co(D)$;*
- *$S \in stb(D)$ iff S attacks each $\{x\} \subseteq \mathcal{A} \setminus S$.*

Given a semantics σ, we call $\sigma(D)$ the set of σ-*extensions* of the ABAF D. We drop 'σ' if it is clear from context.

Graphical ABA Representation *Argumentation frameworks with collective attacks (SETAFs)* (Nielsen and Parsons, 2006) are ideally suited to depict the attack structure between the assumptions in ABAFs as outlined by König, Rapberger, and Ulbricht (2022). In brief, a SETAF is a pair (A, R) consisting of a set of arguments A and an attack relation $R \subseteq 2^A \times A$. We can instantiate an ABAF $D = (\mathcal{L}, \mathcal{A}, \mathcal{R}, ^-)$ as SETAF by setting $A = \mathcal{A}$ and R is the induced attack relation: $S \subseteq A$ attacks $a \in A$ if $S \vdash \overline{a}$.

Example 2.6. *Consider an ABAF with assumptions a, b, c, their contraries $\overline{a} = s$, $\overline{b} = p$, $\overline{c} = q$, and rules $(p \leftarrow a, c)$ and $(s \leftarrow c)$. We can represent the ABAF as SETAF:*



- *$\{a, c\}$ collectively attacks b as $\{a, c\} \vdash p$;*
- *c attacks a since $\{c\} \vdash s$.*

The graph depicts the attack structure between the assumptions; the collective attack is depicted as a joint arrow.

3 Capturing Causal Graphs with ABA

We formalise causal graphs in ABA. We assume a fixed but arbitrary set of variables $\mathbf{V}$ with $|\mathbf{V}| = d$. We refrain from explicitly mentioning the language $\mathcal{L}$. Each assumption a below has a distinct contrary a_c; for convenience, we write $\overline{a}$ instead of a_c if it does not cause confusion.

3.1 Causal ABA

The class of causal relations we aim to capture are characterised by two factors: acyclicity and d-separation.

Acyclicity We formalise graph-theoretic properties since our expected outcome, i.e., the resulting extensions, are graphs. Thus, the assumptions in our ABAF are arrows:

$$\mathcal{A}_{arr} = \{arr_{xy} \mid x, y \in \mathbf{V}, x \neq y\}.$$

Then, we define acyclicity as follows.

Definition 3.1. *Let $D_{dag} = (\mathcal{A}_{dag}, \mathcal{R}_{dag}, \overline{})$ where*

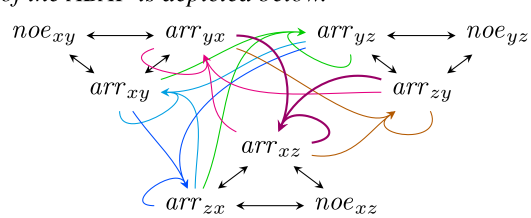
$$\mathcal{A}_{dag} = \mathcal{A}_{arr} \cup \{noe_{xy} \mid x, y \in \mathbf{V}, x \neq y\}$$

and $\mathcal{R}_{dag}$ contains the following rules:

- $\overline{a} \leftarrow b$, $a \neq b$, $a, b \in \{arr_{xy}, arr_{yx}, noe_{xy}\}$, $x, y \in \mathbf{V}$;
- $\overline{arr_{x_i x_{i+1}}} \leftarrow arr_{x_1 x_2}, \dots, arr_{x_{k-1} x_k}$ *for each sequence* $x_1 \dots x_k$ *with* $x_1 = x_k$, *for each* $1 \leq i < k$.

Intuitively, noe_{xy} stands for "no edge between x and y." Note that we define only one atom noe_{xy} for each pair of variables x, y. The first set of rules enables the choice between noe_{xy}, arr_{xy} and arr_{yx}. The second ensures that no extension contains a cycle.

Example 3.2. *Consider $\mathbf{V} = \{x, y, z\}$. The corresponding ABAF D_{dag} contains 9 assumptions: for each pair of variables $u, v \in \mathbf{V}$, we have arr_{uv}, arr_{vu} and noe_{uv}. We observe that we have precisely two cyclic sequences of length > 2, namely (from x) $c_1 = xyzx$ and $c_2 = xzyx$. Both cycles attack each arrow it contains; the attack structure of the ABAF is depicted below.*



The joint arcs represent collective attacks; e.g., the thick, purple arrows pointing to arr_{xz} represent the attack from set $\{arr_{yx}, arr_{xz}, arr_{zy}\}$ on the assumption arr_{xz} based on the derivation $\{arr_{yx}, arr_{xz}, arr_{zy}\} \vdash \overline{arr_{xz}}$.

We show that D_{dag} correctly captures the set of all DAGs of fixed size d. The correspondence is true for all (except gr) argumentation semantics under consideration. Below, we use the assumption arr_{xy} to stand for the arrow (x, y). All proofs of this section are provided in (Russo, Rapberger, and Toni, 2024, Appendix §A).

Proposition 3.3. $\{(\mathbf{V}, S \cap \mathcal{A}_{arr}) \mid S \in \sigma(D_{dag})\} = \{G \mid G$ *is a DAG*$\}$ *for* $\sigma \in \{co, pr, stb\}$.

Note that the grounded extension corresponds to the fully disconnected graph $G = (\mathbf{V}, \emptyset)$ since the empty set is complete. Note also that the correspondence between DAGs and the extensions of the ABAF is one-to-many for complete, admissible and conflict-free assumption sets since a single acyclic graph corresponds to several complete extensions. Accepting the absence of an edge between two variables x, y can be realised by accepting noe_{xy} or simply by accepting none of noe_{xy}, arr_{xy}, arr_{yx} in the extension.

Example 3.4. *In the ABAF from Example 3.2, the fully disconnected graph $(\mathbf{V}, \emptyset)$ corresponds to 2^3 complete extensions; i.e, to each subset of $\{noe_{xy}, noe_{yz}, noe_{zx}\}$.*

For preferred and stable semantics, the correspondence is one-to-one; the semantics coincide in D_{dag}, as stated below.

Lemma 3.5. $\sigma(D_{dag}) = \tau(D_{dag})$ *for* $\sigma, \tau \in \{pr, stb\}$.

Corollary 3.6. *Let $\sigma \in \{pr, stb\}$. Each DAG G corresponds to a unique set $S \in \sigma(D_{dag})$ and vice versa.*

D-separation The first step to represent d-separation is to extend our ABAF with independence statements. We do so by assuming independence between variables. We let

$$\mathcal{A}_{ind} = \{(x \perp\!\!\!\perp y \mid \mathbf{Z}) \mid \mathbf{Z} \subseteq \mathbf{V}, x, y \in \mathbf{V} \setminus \mathbf{Z}, x \neq y\}.$$

The conditional independence $x \perp\!\!\!\perp y \mid \mathbf{Z}$ is violated if the variables x, y are d-connected, given the conditioning set $\mathbf{Z}$. Intuitively, we want to formalise

$$\overline{x \perp\!\!\!\perp y \mid \mathbf{Z}} \text{ if there exists a } \mathbf{Z}\text{-active path between } x, y.$$

To capture this, it is convenient to formalise directed paths and we do so by letting $\mathcal{R}_{graph}$ contain the following rules:

$$dpath_{xy} \leftarrow arr_{xy} \qquad dpath_{xz} \leftarrow dpath_{xy}, arr_{yz}$$
$$e_{xy} \leftarrow arr_{xy} \qquad e_{xy} \leftarrow arr_{yx} \qquad \overline{noe_{xy}} \leftarrow e_{xy}$$

where, intuitively, e_{xy} stands for "edge between x and y."

To formalise d-connectedness in the context of ABA, we introduce *collider-trees*, which generalise the notion of path by adding branches from collider nodes.

Definition 3.7. *Let $G = (\mathbf{V}, E)$ be a DAG, $x, y \in \mathbf{V}$. A* x-y-collider-tree $\mathcal{t}$ *is a sub-graph of G satisfying:*

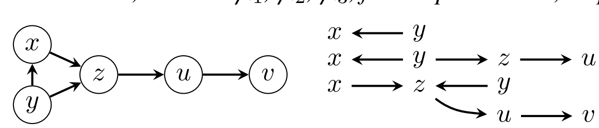
(a) $\mathcal{t}$ contains an x-y-path $\mathcal{p}_t$;

(b) for all $u \in \mathcal{t}$, if $u \notin \mathcal{p}_t$ then there is $v \in \mathcal{t}$ such that v is a collider in $\mathcal{p}_t$ and u is a descendant of v.

A c-t-path from collider node c (of $\mathcal{p}_t$) to a leaf node t, $t \notin \{x, y\}$, is called a branch *of $\mathcal{t}$. For a set of variables $\mathbf{Z} \subseteq \mathbf{V}$, we call $\mathcal{t}$* $\mathbf{Z}$-active *iff $\mathcal{p}_t$ is active w.r.t. $\mathcal{t} \cup (\mathbf{V}, \emptyset)$.*

In the remainder of the paper, we drop 'x-y' and simply say collider-tree whenever it does not cause confusion.

Example 3.8. *Consider a causal graph G with five variables x, y, z, u, v as depicted below (left), and some collider-trees, denoted $\mathcal{p}_1$, $\mathcal{p}_2$, $\mathcal{p}_3$, from top to bottom, resp.:*



The collider-trees $\mathcal{p}_1$ and $\mathcal{p}_2$ are active w.r.t. $\emptyset$; both paths have no collider so they are active w.r.t. every set not intersecting them; $\mathcal{p}_3$ is active for sets containing z, u or v.

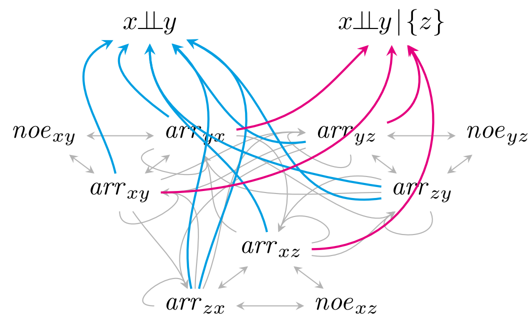
We are now ready to define our causal ABA framework.

Definition 3.9. *A* causal ABAF $D_{ds} = (\mathcal{A}_{ds}, \mathcal{R}_{ds}, \overline{})$ *is characterised by*

$$\mathcal{A}_{ds} = \mathcal{A}_{dag} \cup \mathcal{A}_{ind}, \text{ and } \mathcal{R}_{ds} = \mathcal{R}_{dag} \cup \mathcal{R}_{graph} \cup \mathcal{R}_{act},$$

where $\mathcal{R}_{act}$ contains rules $(\overline{x \perp\!\!\!\perp y \mid \mathbf{Z}} \leftarrow \mathcal{t})$ for each $\mathbf{Z}$-active x-y-collider-tree $\mathcal{t}$ with $x \neq y$, and $\mathbf{Z} \subseteq \mathbf{V} \setminus \{x, y\}$.

Example 3.10. *Let us consider again Example 3.2 with* $\mathbf{V} = \{x, y, z\}$*. We extend our ABAF with six independence assumptions and add the corresponding contraries. Below, we depict all arguments and attacks for the pair* x, y*; i.e., all attacks on the new assumptions* $(x \perp\!\!\!\perp y)$ *and* $(x \perp\!\!\!\perp y \,|\, \{z\})$*.*



The assumption $(x \perp\!\!\!\perp y)$ *is attacked by* arr_{xy}*,* arr_{yx} *and by all x-y-paths with inner node* z *except for the collider;* $(x \perp\!\!\!\perp y \mid \{z\})$ *is attacked by* arr_{xy}*,* arr_{yx} *and the collider* $\{arr_{xz}, arr_{yz}\}$*. Attacks for the other pairs are analogous.*

The formalization correctly captures independence, as stated in the following proposition.

Proposition 3.11. *Let* $\sigma \in \{pr, stb\}$*,* $S \in \sigma(D_{ds})$*,* $x, y \in \mathbf{V}$*,* $\mathbf{Z} \subseteq \mathbf{V} \setminus \{x, y\}$*. Then* $(x \perp\!\!\!\perp y \,|\, \mathbf{Z}) \in S$ *iff* $(x \perp\!\!\!\perp_G y \mid \mathbf{Z})$*.*

Note that we cannot guarantee the correspondence for complete semantics, as illustrated next.

Example 3.12. *In the ABAF from Example 3.10,* $S = \emptyset$ *is complete; indeed,* D_{ds} *does not contain assumptions that are unattacked. The corresponding graph is* $G = (\mathbf{V}, \emptyset)$ *(cf. Example 3.4). In* G*, each pair of variables is independent; however,* S *does not contain any independence statement.*

The example above shows that the correspondence between independence assumptions and independencies entailed by a DAG via d-separation is not preserved when dropping $\subseteq$-maximality of the extensions. Interestingly, the other direction of Proposition 3.11 still holds for complete semantics; i.e., no incorrect independence statements are included in a complete extension.

Proposition 3.13. *Let* $S \in co(D_{ds})$*,* $x, y \in \mathbf{V}$*,* $\mathbf{Z} \subseteq \mathbf{V} \setminus \{x, y\}$*. Then* $(x \perp\!\!\!\perp y \,|\, \mathbf{Z}) \in S$ *implies* $(x \perp\!\!\!\perp_G y \,|\, \mathbf{Z})$*.*

3.2 Integrating Causal Knowledge

So far, we have introduced an ABAF that faithfully captures conditional independence in causal models. We have shown that an independence statement $(x \perp\!\!\!\perp y \,|\, \mathbf{Z})$ is contained in an extension S if and only if it is consistent with graph corresponding to S. This, in turn translates to the dependencies of the graph: x and y are dependent given $\mathbf{Z}$ iff $(x \perp\!\!\!\perp y \,|\, \mathbf{Z}) \notin S$.

Our proposed ABAF can be integrated in any causal discovery pipeline to add formal guarantees that the graph discovered corresponds to the independences in the data. We integrate information from external sources, might they be statistical methods or experts, as *facts*.[1]

[1]Assuming sufficient accuracy of the data as a first step; later on, we will assign weights to the reported independence statements in our final system to account for statistical errors.

In the remainder of this section, we write $D \cup \{r\} = (\mathcal{A}, \mathcal{R} \cup \{r\}, \overline{})$ for an ABAF $D = (\mathcal{A}, \mathcal{R}, \overline{})$ and rule r.

Let us consider again our three-variables example from before (cf. Example 3.10). First, suppose we have learned that x and y are marginally independent. We incorporate this information simply by adding the rule $(x \perp\!\!\!\perp y \leftarrow)$. This rule ensures that each extension contains $(x \perp\!\!\!\perp y)$. Since each extension must be closed, no active path between x and y can be accepted. We can proceed similarly when incorporating specific causal relations (directed edges).

Proposition 3.14. *Let* $x, y, a, b \in \mathbf{V}$*,* $\mathbf{Z} \subseteq \mathbf{V} \setminus \{x, y\}$*,* $\mathbf{X} \subseteq \mathbf{V} \setminus \{a, b\}$*,* $\sigma \in \{pr, na, ss, stg, stb\}$*,* $r \in \{(x \perp\!\!\!\perp y \,|\, \mathbf{Z} \leftarrow), (arr_{xy} \leftarrow)\}$*. For each* $S \in \sigma(D_{ds} \cup \{r\})$*, it holds that*

$$(a \perp\!\!\!\perp b \,|\, \mathbf{X}) \in S \text{ iff } (a \perp\!\!\!\perp_G b \,|\, \mathbf{X}).$$

Crucially, we observe that adding external facts comes at a cost: the ABAF is not flat anymore; indeed, the independence and arrow literals might appear in the head of rules.

Now, what happens if we incorporate test results or causal relation from an external source? Suppose we discovered x and y are marginally dependent. When we add the rule $(\overline{x \perp\!\!\!\perp y} \leftarrow)$ we successfully render $(x \perp\!\!\!\perp y)$ false; however, we lose the correspondence between the (in)dependence statements and the graph of a given extension: when adding the contrary of $(x \perp\!\!\!\perp y)$ nothing (in the framework presented so far) prevents us from accepting one of the arrows arr_{xy} or arr_{yx}. We need to generalise the framework to ensure that our ABAF is sound when adding dependencies as facts to the framework, as discussed next.

Blocked paths The ABAF D_{ds} successfully captures that an active path implies dependence. To guarantee soundness, it remains to formalise the other direction: independence between two nodes x, y implies that each path linking them is blocked. For this, we introduce new assumptions

$$\mathcal{A}_{bp} = \{bp_{\mathscr{p}|\mathbf{Z}} \mid \mathscr{p} \text{ is a x-y-path}, \mathbf{Z} \subseteq \mathbf{V} \setminus \{x, y\}\}$$

with contraries $\overline{bp_{\mathscr{p}|\mathbf{Z}}} = ap_{\mathscr{p}|\mathbf{Z}}$.

Furthermore, we require two new sets of rules: the first set of rules formalises that the independence between two variables x and y given $\mathbf{Z}$ requires that each path between x, y is blocked; the second set specifies when a path is $\mathbf{Z}$-active.

Definition 3.15. *For* $x, y \in \mathbf{V}$*,* $x \neq y$*,* $\mathbf{Z} \subseteq \mathbf{V} \setminus \{x, y\}$*, we define* $\mathcal{R}_{ext} = \mathcal{R}_{ds} \cup \mathcal{R}_{xyZ}$ *with* $\mathcal{R}_{xyZ}$ *containing the rules*

- $x \perp\!\!\!\perp y \,|\, \mathbf{Z} \leftarrow bp_{\mathscr{p}_1|\mathbf{Z}}, \ldots, bp_{\mathscr{p}_k|\mathbf{Z}}$ *where* $\mathscr{p}_1, \ldots, \mathscr{p}_k$ *denote all paths between* x *and* y*;*
- $ap_{\mathscr{p}|\mathbf{Z}} \leftarrow \mathscr{p}_t$ *for each* $\mathbf{Z}$*-active x-y-collider-tree* $\mathscr{t}$ *with underlying x-y-path* $\mathscr{p}_t$*.*

Let us consider the effect of these rules with an example.

Example 3.16. *Consider again Example 3.10; suppose we observed* $x \not\!\perp\!\!\!\perp y \mid \{z\}$*. We add the independence* $(\overline{x \perp\!\!\!\perp y \,|\, \{z\}} \leftarrow)$ *which prevents us from accepting all* $bp_{\mathscr{p}|\{z\}}$ *assumptions at the same time (since each extension* S *is closed, we also accept* $(x \perp\!\!\!\perp y \mid \{z\})$*, therefore, this leads to a conflict). Consequently, one of the* $bp_{\mathscr{p}|\{z\}}$ *assumptions is attacked, i.e., some path between* x, y *is active.*

It can be checked that the paths $\wp_1$, $\wp_2$, $\wp_3$ depicted below are $\{z\}$-active:

$$x \longrightarrow y \qquad x \longleftarrow y \qquad x \longrightarrow z \longleftarrow y$$

As visualised in Example 3.10, each of these paths attack $(x \perp\!\!\!\perp y \,|\, \{z\})$. Due to the new rules from Definition 3.15 each path $\wp_i$ also derives $ap_{\wp_i|\mathbf{Z}}$ which attacks $bp_{\wp_i|\mathbf{Z}}$. Therefore, each extension S must contain one of these paths.

We note that it suffices to add rules only for the dependence fact that we want to add. That is, when introducing fact $(\overline{x \perp\!\!\!\perp y \,|\, \mathbf{Z}} \leftarrow)$ it suffices to add the rules from Definition 3.15 for x, y, $\mathbf{Z}$. We define the extended ABAF.

Definition 3.17. *For $x, y \in \mathbf{V}$, $\mathbf{Z} \subseteq \mathbf{V} \setminus \{x, y\}$, the extended causal ABAF $D^{xy\mathbf{Z}}_{csl} = (\mathcal{A}_{csl}, \mathcal{R}_{csl}, \bar{\ })$ is characterised by $\mathcal{A}_{csl} = \mathcal{A}_{ds} \cup \mathcal{A}_{bp}$ and $\mathcal{R}_{csl} = \mathcal{R}_{ext} \cup \{\overline{x \perp\!\!\!\perp y \,|\, \mathbf{Z}} \leftarrow\}$.*

The ABAF is sound and complete, as stated below.

Proposition 3.18. *Let $x, y, a, b \in \mathbf{V}$, let $\mathbf{Z} \subseteq \mathbf{V} \setminus \{x, y\}$ and $\mathbf{X} \subseteq \mathbf{V} \setminus \{a, b\}$, let $\sigma \in \{pr, stb\}$ and let $S \in \sigma(D^{xy\mathbf{Z}}_{csl})$. It holds that $(a \perp\!\!\!\perp b \,|\, \mathbf{X}) \in S$ iff $(a \perp\!\!\!\perp_G b \,|\, \mathbf{X})$.*

Together, Propositions 3.14 and 3.18 guarantee that causal knowledge can be integrated in a faithful way. We obtain that this fine-tuned specification allows us to add (in)dependence facts and arrows whilst guaranteeing consistency of the causal ABAF. Independence facts and arrows can be added without further changes to the framework; when adding dependence facts, we require additional rules as specified in Definition 3.15. Below, we denote by D^{T}_{csl}, where T is a set of (in)dependence and arrow facts, the ABAF obtained by the iterative update of the ABAF D_{ds} with $D^{xy\mathbf{Z}}_{csl}$ for all dependence facts $(\overline{x \perp\!\!\!\perp y \,|\, \mathbf{Z}} \leftarrow) \in T$.

Corollary 3.19. *Let T be a set of (in)dependence and arrow facts, $\sigma \in \{pr, stb\}$, $x, y \in \mathbf{V}$, $\mathbf{Z} \subseteq \mathbf{V} \setminus \{x, y\}$ and $S \in \sigma(D^{T}_{csl})$. Then $(x \perp\!\!\!\perp y \,|\, \mathbf{Z}) \in S$ iff $(x \perp\!\!\!\perp_G y \mid \mathbf{Z})$.*

4 Implementation

In this section, we present an instance of our *Causal ABA algorithm* which combines our causal ABAF with heuristic approaches to select the independence facts that it can take in input. The workflow of Algorithm 1 is as follows:

1. The main function of the algorithm is what we name **`causalaba`** (line 9 and 12 of Algorithm 1). The causal ABAF instance is determined by the number of nodes in the graph d and a set of facts $\mathbf{T}$. We *generate the causal ABAF* $D^{\mathbf{T}}_{csl}$ presented in §3, using an ASP implementation in clingo (Gebser et al., 2019). We then compute the stable extensions of the causal ABAF. Our ASP encoding is detailed in §4.1.
2. The main input of Algorithm 1 is a set of independence facts ($\mathcal{I}$), alongside the significance threshold α and the number of nodes d. We discuss *sourcing facts* in §4.2.
3. As shown in §3, each stable extension corresponds to a DAG compatible with the fixed set of independence tests. However, statistical methods can return erroneous results, in which case our causal ABAF might output no stable

Algorithm 1: Causal ABA (with independence facts)

Input: $\mathcal{I}, \alpha, |\mathbf{V}| = d$

1: $\mathbf{T} \leftarrow [\ ]$
2: **for** $p = I(x, y \mid \mathbf{Z}) \in \mathcal{I}$ **do**
3: $\quad s \leftarrow |\mathbf{Z}|$
4: $\quad$ **if** $p > \alpha$ **then**
5: $\quad\quad \mathbf{T} \leftarrow \mathbf{T} + [(\texttt{indep}(x, y, \mathbf{Z}), \mathcal{S}(p, \alpha, s, d))]$
6: $\quad$ **else**
7: $\quad\quad \mathbf{T} \leftarrow \mathbf{T} + [(\texttt{dep}(x, y, \mathbf{Z}), \mathcal{S}(p, \alpha, s, d))]$
8: $\mathbf{T} \leftarrow \text{sort}(\mathbf{T}, \mathcal{S})$ ▷ Sort elements of $\mathbf{T}$ by strength $\mathcal{S}$
9: $\mathbf{M} = \texttt{causalaba}(d, \mathbf{T})$
10: **while** $\mathbf{M} = \emptyset$ **do**
11: $\quad \mathbf{T} \leftarrow \mathbf{T}[2 \ldots |\mathbf{T}|]$ ▷ Drop fact with lowest $\mathcal{S}$
12: $\quad \mathbf{M} = \texttt{causalaba}(d, \mathbf{T})$
13: **for** $G \in \mathbf{M}$ **do**
14: $\quad \mathcal{S}_G = 0$
15: $\quad$ **for** $p = I(x, y \mid \mathbf{Z}) \in \mathcal{I}$ **do**
16: $\quad\quad s \leftarrow |\mathbf{Z}|$
17: $\quad\quad$ **if** $p > \alpha$ & $x \perp\!\!\!\perp_G y \mid \mathbf{Z}$ **then**
18: $\quad\quad\quad \mathcal{S}_G \leftarrow \mathcal{S}_G + \mathcal{S}(p, \alpha, s, d)$
19: $\quad\quad$ **else**
20: $\quad\quad\quad \mathcal{S}_G \leftarrow \mathcal{S}_G - \mathcal{S}(p, \alpha, s, d)$
21: $G \leftarrow \text{argmax}(G \in \mathbf{M}, \mathcal{S}_G)$ ▷ Select G with max $\mathcal{S}_G$
return G

extension at all. To overcome this problem, we *select facts* by assigning them appropriate weights (lines 2-12) and use these weights both to optimise (using weak constraints within **`causalaba`**) and rank (possibly several) output extensions (lines 10-20). We discuss this in §4.3.

Our proposed Algorithm 1 is a sound procedure to extract DAGs given a consistent set of independencies.

Proposition 4.1. *Given a set $\mathbf{V}$ of variables and a set of (in)dependencies $\mathcal{I}$, compatible with a (set of) MEC(s), Algorithm 1 outputs a DAG consistent with $\mathcal{I}$.*

In this work, we instantiate our Algorithm 1 using the Majority-PC algorithm (MPC) (Colombo and Maathuis, 2014) to source facts, resulting in the *ABA-PC algorithm*. In the following subsections, we detail our implementation.

Remark 4.2. *The causal ABAF D^{T}_{csl} from Definition 3.17 is potentially* non-flat *since assumptions can be derived: independence assumptions as well as arr and ap assumptions may appear in the head of rules. Thus, it lies in a broader ABA class, affecting semantical properties known for flat ABAFs; for instance, complete extensions may not always exist (Čyras et al., 2018; Ulbricht et al., 2024). As a consequence, standard ABA solvers are not applicable to our case since they typically focus on the class of flat ABAFs. In this work, we therefore propose an Answer Set Programming (ASP) encoding of our causal ABAF under stable semantics. This also allows us to exploit ASP's grounding abilities to obtain causal ABAFs from concise schemata representations (see (Proietti and Toni, 2022) for the presentation of ABA in terms of schemata).*

Listing 1: Module Π_{col}

```
1 collider(Y,X,Z) ← arrow(X,Y),arrow(Z,Y),X!=Y,var(X),
      var(Y),var(Z).
2 coll_desc(N,Y,X,Z) ← collider(Y,X,Z),dpath(Y,N).
3 nb(N,X,Y,S) ← in(N,S), collider(N,X,Y).
4 nb(N,X,Y,S) ← not in(N,S),not collider(N,X,Y),var(N),
      var(X),var(Y),set(S),N!=X,N!=Y,X!=Y.
5 nb(N,X,Y,S) ← not in(N,S),coll_desc(Z,N,X,Y),in(Z,S),
      var(N),var(X),var(Y).
```

Listing 2: Module $\Pi_{ap}(\wp, (v_i)_{i \leq k})$

```
ap(v_1,v_k,℘,S) ← (arrow(v_i,v_{i+1}))_{i<k}, not in(v_1,S), not
      in(v_k,S), set(S), (nb(v_i,v_{i-1},v_{i+1},S))_{1<i<k}.
dep(v_1,v_k,S) ← ap(v_1,v_k,℘,S).
```

4.1 Encoding Causal ABA in ASP

Stable ABA semantics and stable semantics for Logic Programs (LP) are closely related (Caminada et al., 2015; Schulz and Toni, 2015); crucially, their correspondence has recently been extended to non-flat instances (Rapberger, Ulbricht, and Toni, 2024). In standard ABA-LP translations, assumptions are associated with their default negated contraries: an assumption $a \in \mathcal{A}$ with contrary $a_c \in \mathcal{L}$ corresponds to the default negated literal *not* a_c. These translations, however, consider only the case where the underlying logical language is atomic. To exploit the full power of ASP, we slightly deviate from standard translations, when appropriate, whilst guaranteeing consistency with our model. We also make use of more descriptive contrary names to enable a more intuitive reading.

For a set of variables $\mathbf{V}$, we express the causal ABAF by

1. encoding DAGs: each answer set corresponds to a DAG;
2. encoding d-separation: nodes x and y are independent given $\mathbf{Z}$ *iff* x and y are not linked via an active path.

Following the standard translation, each ABA atom arr_{xy} is translated to *not* $\overline{arr_{xy}}$. Here, we identify "not arr_{yx}" simply with **arrow**(x,y) and "not noe_{xy}" with **edge**(x,y). In our encoding, each answer set corresponds to precisely one DAG of size $|\mathbf{V}| = d$ for a given set $\mathbf{V}$ of variables. The encoding of acyclicity and further DAG-specific elements is given in (Russo, Rapberger, and Toni, 2024, Appendix §B).

To link causality and DAGs we encode the d-separation criterion. To handle sets in ASP, we encode the (k-th) set $S \subseteq \mathbf{V}$ with predicates **in**(k,x). Module Π_{col} in Listing 1 encodes collider and collider descendant (with natural specifications of the **arrow** and **dpath** (directed path) predicates). Next, we introduce *non-blocking* nodes: node $v \notin \{x, y\}$ in an x-y-path is non-blocking, given $\mathbf{Z}$, iff

- v is a collider (with respect to its neighbours in the path) and either $v \in \mathbf{Z}$ or a descendant of v is in $\mathbf{Z}$; or
- v is not a collider and $v \notin \mathbf{Z}$.

Lines 3-5 in Module Π_{col} in Listing 1 encode these rules. Now, for each pair $x, y \in \mathbf{V}$, for each set $S \setminus \{x, y\}$, for each x-y-path $\wp = v_1 \ldots v_n$ with $x = v_1$ and $y = v_n$, we add rules $\Pi_{ap}(\wp, (v_i)_{i \leq k})$ as specified in Listing 2. This

Listing 3: Module $\Pi_{bp}(x, y, (\wp_i)_{i \leq k})$

```
indep(x,y,S) ← (not ap(x,y,℘_i,S))_{i≤k}, not in(x,S),
      not in(y,S), set(S).
```

guarantees the '*if*'-direction: if x and y are connected via a $\mathbf{Z}$-active path then they are dependent. For the '*only if*'-direction, we require Module $\Pi_{bp}(x, y, (\wp_i)_{i \leq k})$: for each pair of variables x and y, we add the rule detailed in the listing to ensure that the absence of an active path between x and y implies independence between them; $(\wp_i)_{i \leq k}$ denotes the list of all paths between x and y. The Module Π_{bp} in Listing 3 encodes the blocked path rules defined in Definition 3.15. The **indep**- and **dep**-predicates take two variables x, y, and a set S of variables as arguments. We note that, in general, the number of paths between two variables can be exponential (up to $\lfloor (d-2)!e \rfloor$). To lower the number of paths, we make use of the observation that fixing independence facts amounts to removing edges between nodes.[2] When fixing independence facts ($a \perp\!\!\!\perp b \,|\, \mathbf{X} \leftarrow$), we thus consider only the paths in the skeleton that do not contain (a, b).

As outlined in Proposition 3.18, fixing dependence facts requires only the addition of the blocked path rules corresponding to the fact. That is, adding the fact ($a \not\!\perp\!\!\!\perp b \mid \mathbf{X}$) only requires including the rules $\Pi_{ap}(\wp, (v_i)_{i \leq k})$ and $\Pi_{bp}(x, y, (\wp_i)_{i \leq k})$ to guarantee correctness.

As discussed in §3, our proposed ABAF returns all the DAGs compatible with some fixed facts, representing relations amongst nodes, may these be conditional/marginal independencies and/or (un)directed causal relations (arrows and edges). In the proposed instantiation of Causal ABA, ABA-PC, we input a set of facts in the form of independence relations and weight them according to their p-value.

4.2 Sourcing Facts

A DAG with d nodes is fully characterised by $\frac{1}{2}d(d-1)2^{d-2}$ independence relations, growing exponentially in the number of nodes. Therefore, it is not computationally efficient to carry out all possible tests, as in (Hyttinen, Eberhardt, and Järvisalo, 2014). Several solutions to this problem have been proposed in the Causal Discovery literature, e.g., (Spirtes, Glymour, and Scheines, 2000; Tsamardinos, Brown, and Aliferis, 2006; Colombo and Maathuis, 2014). Spirtes, Glymour, and Scheines (2000); Tsamardinos, Brown, and Aliferis (2006); Colombo and Maathuis (2014) all use conditional independence tests such as (Fisher, 1970; Zhang et al., 2011; Gretton et al., 2007). Other strategies to recover causal graphs from data, referred to as score-based methods, e.g., (Chickering, 2002; Ramsey et al., 2017) involve the use of statistical metrics that measure the added-value of adding/removing an arrow in terms of fit to the data. Hence they would return arrow weights. In this work, we use the MPC algorithm (Colombo and Maathuis, 2014), which provably[3] recovers the underlying CPDAG from data,

[2]This observation is key for constraint-based causal discovery algorithms such as PC (Spirtes, Glymour, and Scheines, 2000).

[3]under the assumptions of sufficiency (no unmeasured confounders), faithfulness (data represents a DAG) and perfect inde-

to source facts. Let us illustrate the input facts we consider through our running example.

Example 4.3. *We run the MPC algorithm in Example 1.1 which performs 23 out of 24 tests, including the following.*

$$r \perp\!\!\!\perp wp \,|\, \{ws\} \quad wp \perp\!\!\!\perp ws \,|\, \{r\} \quad r \perp\!\!\!\perp wp$$
$$r \perp\!\!\!\perp wp \,|\, \{wr\} \quad wp \not\perp\!\!\!\perp ws \,|\, \{r, wr\} \quad r \perp\!\!\!\perp wp \,|\, \{wr, ws\}$$

However, only $r \perp\!\!\!\perp wp$ is correct; the only other independence $wp \perp\!\!\!\perp_G ws \mid \{r, wr\}$ in G is wrongly classified. All other tests result in dependencies.

Based on this erroneous results, MPC yields the graph shown in Example 1.1 (right), deviating from the ground truth. Crucially, the graph does not capture the independence relations listed above. In fact, there is no graph that satisfies the test results because it is not possible that r and wp are independent conditioned on any set, but r and ws, as well as wp and ws, are dependent. The dependencies indicate a path between r and wp, leading to a contradiction.

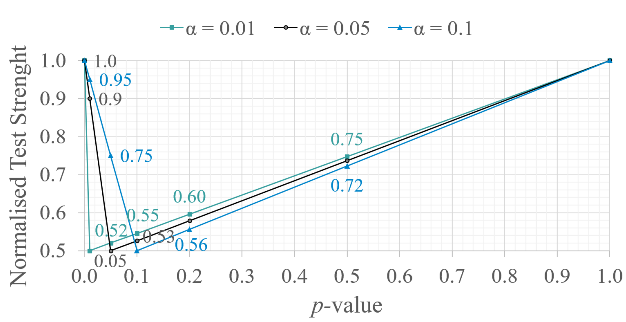
Note that Algorithm 1 is flexible to the choice of facts' source, e.g. we could have used the tests performed by (Tsamardinos, Brown, and Aliferis, 2006) or, with a slight modification, the arrow weights from (Ramsey et al., 2017).

4.3 Weighting Facts

Here we outline our strategy to weight independence tests results, based on their p-value and the size of the conditioning set. We use these weights as weak constraints and to rank facts and extensions. As a result of using stable semantics, wrong tests can render empty extensions if they contradict another (set of) test(s). Our aim is thus to exclude the wrong tests that create inconsistencies and cause our ABAF to output no extension. To this end, we define a simple heuristic to rank p-values from independence tests, given significance level α, but insensitive to whether they fall below or above it. Firstly, we define the following normalising function:

$$\gamma(p, \alpha) = \begin{cases} 2p\alpha - 1 & \text{iff } p < \alpha \\ \frac{2\alpha - p - 1}{2(\alpha - 1)} & \text{otherwise} \end{cases}$$

Below is a plot of the function γ across the p-value interval, for three commonly chosen levels of α.



pendence information, see (Colombo and Maathuis, 2014) for detail and formal definitions. Note that the original PC strategy is based on the assumption that there will be no inconsistencies and therefore the algorithm does not test a pair of variables anymore once an independence is found. However, inconsistencies might arise when erroneous results are obtained.

The output of γ, for a given α and p, follows the intuition that the most uncertainty is around the significance threshold $p = \alpha$ (Sellke, Bayarri, and Berger, 2001; Berger, 2003), which we make correspond to the lowest value of $\gamma = 0.5$.

The final strength of the (in)dependence facts is obtained by weighting the output of the normalising function γ by a factor penalising bigger sizes of the conditioning set $\mathbf{Z}$:

$$\mathcal{S}(p, \alpha, s, d) = \frac{(1 - s)}{(d - 2)} \gamma(p, \alpha) \tag{1}$$

where $s = |\mathbf{Z}|$ the cardinality of the conditioning set and $d = |\mathbf{V}|$, the cardinality of the set of nodes in the graph. The reason for weighting γ by s and d follows the intuition that the accuracy of independence test lowers as the conditioning set size increases (Sellke, Bayarri, and Berger, 2001).

We use our final weights $\mathcal{S}$ to rank the test carried out by MPC. Then, our strategy is simple: exclude an incremental number of the lowest ranked tests until the returned extension is not empty. Let us illustrate our strategy.

Example 4.4. *Consider again Example 1.1. The results of the independence tests from MPC (using Fisher's Z (1970) and $\alpha = 0.05$) have the following p-values (we show the same subset of the 23 tests carried out, as in Example 4.3):*

$r \perp\!\!\!\perp wp$	$p = 0.45$	$\mathcal{S} = 0.71$
$r \perp\!\!\!\perp wp \mid \{ws\}$	$p = 0.52$	$\mathcal{S} = 0.37$
$r \perp\!\!\!\perp wp \mid \{wr\}$	$p = 0.33$	$\mathcal{S} = 0.32$
$wp \perp\!\!\!\perp ws \mid \{r\}$	$p = 0.05$	$\mathcal{S} = 0.25$
$r \perp\!\!\!\perp wp \mid \{wr, ws\}$	$p = 0.39$	$\mathcal{S} = 0.00$
$wp \not\perp\!\!\!\perp ws \mid \{r, wr\}$	$p = 0.03$	$\mathcal{S} = 0.00$

We apply Eq. 1 to calculate $\mathcal{S}$. Ranking tests by $\mathcal{S}$, as shown above, the right test is the highest scoring one. Fixing all the tests returns no solution. We thus start excluding the test with the lowest strength and progressively more until we find a model. In this example, the right DAG is obtained by excluding 9 of the performed tests, including the bottom five of the above list, and keeping the 14 strongest ones.

Here, we obtain exactly one DAG when excluding 40% of the tests carried out by MPC. If we obtain multiple models, we score each of them as in Algorithm 1, lines 14-19. In addition, we encode the (in)dependence facts as *weak constraints*, treated as optimisation statements (Gebser, Kaminski, and Schaub, 2011), to sort out sub-optimal extensions.

Our weighting function is similar to the one proposed by Bromberg and Margaritis (2009), with two differences: we re-base around 0.5 instead of $1 - \alpha$, to allow for more discrimination; and use the conditioning set size irrespective of the test's result, instead of including it only in the case of dependence (trusting that p-values accurately reflect the probability of wrongly rejecting the null hypothesis).

We emphasize that classical independence tests are asymmetric in nature, and inference of dependence is only possible if there is enough evidence against the null hypothesis ($p < \alpha$) with an expected Type I error (rejecting independence when it is true) corresponding to α. Conversely, no inference is possible when $p \geq \alpha$, i.e., when the null hypothesis $\mathcal{H}_0$ cannot be rejected, since p-values are distributed uniformly in $[0, 1]$ under $\mathcal{H}_0$.

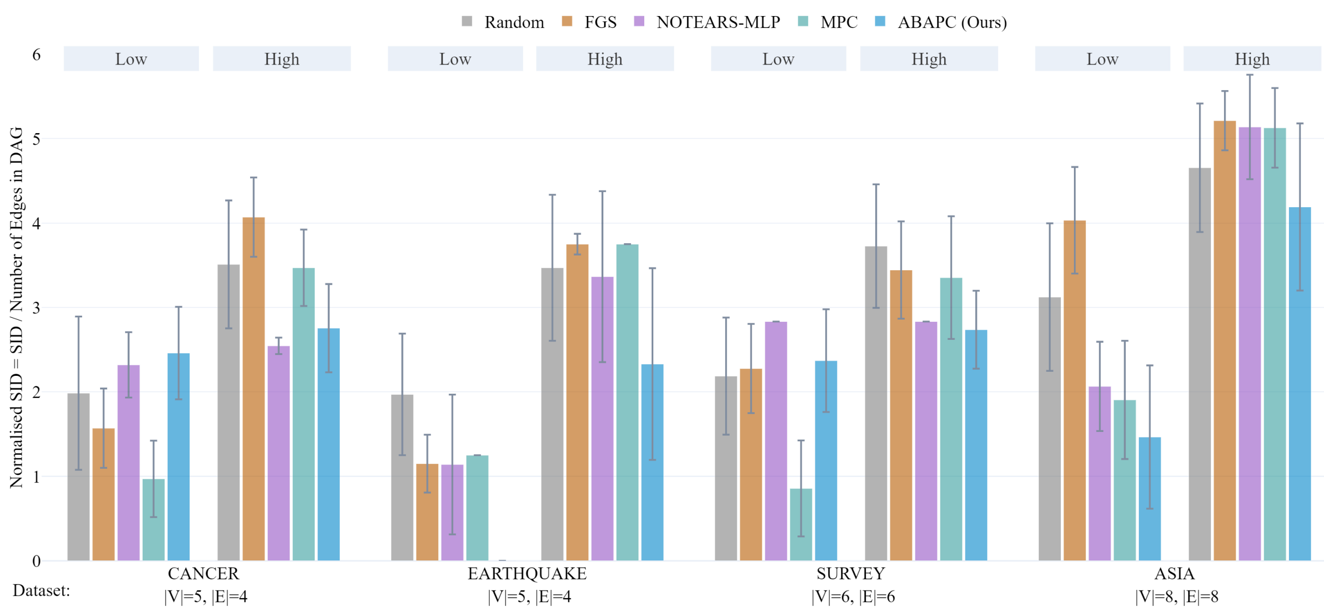


Figure 2: Normalised Structural Interventional Distance for four datasets from the `bnlearn` repository. Lower is better. Low (resp. High) is the SID for the best (resp. worst) DAG in the estimated CPDAG.

As pointed out in (Bromberg and Margaritis, 2009; Hyttinen, Eberhardt, and Järvisalo, 2014), using p-values directly as strength is common but neither sound nor consistent. Transforming p-values to probability estimates, e.g. as in (Jabbari et al., 2017; Claassen and Heskes, 2012; Triantafillou, Tsamardinos, and Roumpelaki, 2014), would address this point, but out of scope for this work. A possible alternative to weighting and excluding facts might also be the use of less strict ABA semantics, left out of our experiments since not available in the ASP implementation used.

5 Empirical Evaluation

We evaluate our ABA-PC algorithm on four datasets from the `bnlearn` repository (Scutari, 2014), which hosts commonly used benchmarks in Causal Discovery, some of which based on real published experiments or expert opinions. We use the Asia, Cancer, Earthquake and Survey datasets, which represent problems of decision making in the medical, law and policy domains. We provide further details in Appendix §C.1; implementation and computing infrastructure details are in §C.2 in (Russo, Rapberger, and Toni, 2024).

Evaluation Metrics and Baselines For evaluation, we use a prominent metric in causal discovery: Structural Interventional Distance (SID) (Peters and Bühlmann, 2015) measures the deviation in the causal effects estimation deriving from a mistake in the estimated graph. SID works as a "downstream task" error rate for the causal inference task, which has causal graphs as a pre-requisite. We calculate SID between the estimated and the true CPDAG and repeat the experiments 50 times per dataset to record confidence intervals. Given that a CPDAG is a mixed graph, SID is calculated for the worst and best scenarios. In order to compare across graphs with different number of edges, we normalise SID (NSID) dividing it by the number of edges in the true DAG. NSID can go above 100% since extra edges could be introduced. We provide details on the metrics in §C.3 and results based on additional metrics (SHD, F1 score, precision and recall) in §C.5 in (Russo, Rapberger, and Toni, 2024).

We compare ABA-PC to four baselines: a Random sample of graphs of the right dimensions ($\mathbf{V}$,E); Fast Greedy Search (FGS) (Ramsey et al., 2017) and NOTEARS-MLP (Zheng et al., 2020) which use, resp., the Bayesian Information Criterion and Multilayer Perceptrons with a continuous formulation of acyclicity to optimise the graph's fit to the data; and MPC (Colombo and Maathuis, 2014),[4] see (Russo, Rapberger, and Toni, 2024, §C.4) for more details.

Results The results of our experiments are in Fig. 2. Best and worst SID are in the (Low, resp. High) sections for each dataset; the number of edges and nodes in each dataset is given below the x-axis labels. ABA-PC ranks 1^{st} in the worst case SID (High) for all datasets. It performs significantly (w.r.t. t-tests of difference in means, see (Russo, Rapberger, and Toni, 2024, §C.6)) better than all baselines on three out of four datasets (Cancer, Earhquake and Asia) and is on par with NOTEARS-MLP for the Survey data. Furthermore, ABA-PC performs significantly better than MPC for all datasets. This demonstrates how, with the same underlying information from the data, our method returns more accurate CPDAGs in the worst case scenario. For the best case SID (Low), ABA-PC is significantly better than all baselines for two datasets (Earthquake and Asia). Overall, we observe that ABA-PC performs well on benchmark data compared to a varied selection of baselines from the literature.

[4]We would have liked to compare to the method closest to our work, i.e. (Bromberg and Margaritis, 2009) but unfortunately there is no implementation available.

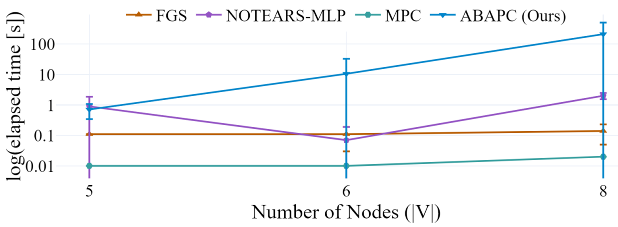


Figure 3: Mean and Standard Deviation of the elapsed time in log scale by number of nodes averaged over 50 runs.

Scalability In Fig. 3 we show the elapsed time (on a log scale) by the number of nodes. This are the recorded times for the experiments in Fig. 2 with 50 repetitions per dataset. As we can see, ABA-PC is the least efficient method. The main reasons for this are the complexity of both the grounding of logical rules and the calculation of the extensions, which clingo carries out exactly and efficiently, but still constitute a bottleneck. We already identified avenues of future work, discussed next, to address scaling limitations of our implementation, given the promising results shown in Fig. 2.

6 Conclusion

We proposed a novel argumentation-based approach to Causal Discovery, targeting the resolution of inconsistencies in data, and showed that it outperforms existing statistics-based methods on four (standard) datasets. Our approach uses independence tests and their p-values to narrow down DAGs most fitting to the data, drawn from stable extensions of ABA frameworks. Other methods to identify and resolve inconsistencies in data for causal discovery have been proposed, e.g. by Ramsey, Spirtes, and Zhang (2006); Colombo and Maathuis (2014), but they focus on marking orientations as ambiguous in the presence of inconsistencies, rather than actually resolving the inconsistencies as we do.

Our proposed framework allows for the introduction of weighted arrows and edges, on top of independencies, which would allow to integrate, as future work, other data-centric methods like score-based causal discovery algorithms (e.g. (Chickering, 2002; Ramsey et al., 2017; Claassen and Heskes, 2012). As for the scalability, we cannot process more than 10 variables at the current state. We are currently working on making the processing of extensions more efficient and on incremental solving to avoid re-grounding when deleting independence facts. Additionally, we would like to extend our approach to deal with latent confounders, in line with (Colombo et al., 2012; Hyttinen, Eberhardt, and Järvisalo, 2014) and cycles (Rantanen, Hyttinen, and Järvisalo, 2020; Richardson and Spirtes, 1999; Hyttinen, Eberhardt, and Järvisalo, 2014) and experiment with other argumentation semantics in the literature, making use of a recently developed solver for non-flat ABA (Lehtonen et al., 2024). Finally, we plan to explore the explainability capabilities intrinsic in an ABA framework (Čyras et al., 2018), which we believe may bring great value to causal discovery in a collaborative human-AI discovery process (Russo and Toni, 2023; Russo, 2023).

Acknowledgemets

Russo was supported by UK Research and Innovation (grant number EP/S023356/1), in the UKRI Centre for Doctoral Training in Safe and Trusted Artificial Intelligence (www.safeandtrustedai.org). Rapberger and Toni were funded by the ERC under the EU's Horizon 2020 research and innovation programme (grant number 101020934) and Toni also by J.P. Morgan and by the Royal Academy of Engineering under the Research Chairs and Senior Research Fellowships scheme.

References

Amgoud, L., and Cayrol, C. 2002. A reasoning model based on the production of acceptable arguments. *Annals of Mathematics and Artificial Intelligence* 34:197–215.

Berger, J. O. 2003. Could fisher, jeffreys and neyman have agreed on testing? *Statistical Science* 18(1):1–32.

Bromberg, F., and Margaritis, D. 2009. Improving the reliability of causal discovery from small data sets using argumentation. *Journal of Machine Learning Research* 10:301–340.

Caminada, M.; Sá, S.; Alcântara, J. F. L.; and Dvorák, W. 2015. On the difference between assumption-based argumentation and abstract argumentation. *IfCoLog Journal of Logics and their Applications* 2(1):15–34.

Chickering, D. M. 2002. Learning equivalence classes of bayesian-network structures. *Journal of Machine Learning Research* 2:445–498.

Claassen, T., and Heskes, T. 2012. A bayesian approach to constraint based causal inference. In *Proceedings of the 28th Conference on Uncertainty in Artificial Intelligence (UAI'12)*, 207–216. AUAI Press.

Colombo, D., and Maathuis, M. H. 2014. Order-independent constraint-based causal structure learning. *Journal of Machine Learning Research* 15(1):3741–3782.

Colombo, D.; Maathuis, M. H.; Kalisch, M.; and Richardson, T. S. 2012. Learning high-dimensional directed acyclic graphs with latent and selection variables. *The Annals of Statistics* 40(1):294–321.

Corander, J.; Janhunen, T.; Rintanen, J.; Nyman, H.; and Pensar, J. 2013. Learning chordal markov networks by constraint satisfaction. *Proceedings of the 27th Annual Conference on Neural Information Processing Systems (NeurIPS'13)* 1349–1357.

Čyras, K.; Fan, X.; Schulz, C.; and Toni, F. 2018. Assumption-based argumentation: Disputes, explanations, preferences. In *Handbook of Formal Argumentation*. College Publications. chapter 7, 365–408.

Fisher, R. A. 1970. Statistical methods for research workers. In *Breakthroughs in statistics: Methodology and distribution*. Springer. 66–70.

Gebser, M.; Kaminski, R.; Kaufmann, B.; and Schaub, T. 2019. Multi-shot ASP solving with clingo. *Theory and Practice of Logic Programming* 19(1):27–82.

Gebser, M.; Kaminski, R.; and Schaub, T. 2011. Complex optimization in answer set programming. *Theory and Practice of Logic Programming* 11(4-5):821–839.

Glymour, C.; Zhang, K.; and Spirtes, P. 2019. Review of causal discovery methods based on graphical models. *Frontiers in genetics* 10:524.

Gretton, A.; Fukumizu, K.; Teo, C. H.; Song, L.; Schölkopf, B.; and Smola, A. J. 2007. A kernel statistical test of independence. In *Proceedings of the 21st Annual Conference on Neural Information Processing Systems (NeurIPS'07)*, 585–592. Curran Associates, Inc.

Hung, H. M. J.; O'Neill, R. T.; Bauer, P.; and Kohne, K. 1997. The behavior of the p-value when the alternative hypothesis is true. *Biometrics* 53(1):11–22.

Hyttinen, A.; Eberhardt, F.; and Järvisalo, M. 2014. Constraint-based causal discovery: conflict resolution with answer set programming. In *Proceedings of the 30th Conference on Uncertainty in Artificial Intelligence (UAI'14)*, 340–349. AUAI Press.

Jabbari, F.; Ramsey, J.; Spirtes, P.; and Cooper, G. 2017. Discovery of causal models that contain latent variables through bayesian scoring of independence constraints. In *Proceedings of the European Conference of Machine Learning and Knowledge Discovery in Databases (ECML PKDD 2017)*, 142–157. Springer.

König, M.; Rapberger, A.; and Ulbricht, M. 2022. Just a matter of perspective. In *Proceedings of the 9th International Conference on Computational Models of Argument (COMMA'22)*, volume 353 of *FAIA*, 212–223. IOS Press.

Lehtonen, T.; Rapberger, A.; Toni, F.; Ulbricht, M.; and Wallner, J. P. 2024. Instantiations and computational aspects of non-flat assumption-based argumentation. *CoRR* abs/2404.11431.

Nielsen, S. H., and Parsons, S. 2006. A generalization of dung's abstract framework for argumentation: Arguing with sets of attacking arguments. In *3rd International Workshop on Argumentation in Multi-Agent Systems (ArgMAS'06), Revised Selected and Invited Papers*, volume 4766 of *LNCS*, 54–73. Springer.

Pearl, J., and Paz, A. 1986. Graphoids: Graph-based logic for reasoning about relevance relations or When would x tell you more about y if you already know z? *Probabilistic and Causal Inference*.

Pearl, J. 2009. *Causality*. Cambridge University Press, 2 edition.

Peters, J., and Bühlmann, P. 2015. Structural intervention distance for evaluating causal graphs. *Neural computation* 27(3):771–799.

Peters, J.; Janzing, D.; and Schölkopf, B. 2017. *Elements of causal inference: foundations and learning algorithms*. The MIT Press.

Philippe Besnard, A. H. 2018. A review of argumentation based on deductive arguments. In *Handbook of Formal Argumentation*. College Publications. chapter 9, 437–484.

Proietti, M., and Toni, F. 2022. Learning assumption-based argumentation frameworks. In Muggleton, S. H., and Tamaddoni-Nezhad, A., eds., *Proceedings of the 31st International Conference on Inductive Logic Programming (ILP'22)*, volume 13779 of *LNCS*, 100–116. Springer.

Ramsey, J.; Glymour, M.; Sanchez-Romero, R.; and Glymour, C. 2017. A million variables and more: the fast greedy equivalence search algorithm for learning high-dimensional graphical causal models, with an application to functional magnetic resonance images. *International journal of data science and analytics* 3:121–129.

Ramsey, J.; Spirtes, P.; and Zhang, J. 2006. Adjacency-faithfulness and conservative causal inference. In *Proceedings of the 22nd Conference on Uncertainty in Artificial Intelligence (UAI'06)*, 401–408. AUAI Press.

Rantanen, K.; Hyttinen, A.; and Järvisalo, M. 2020. Discovering causal graphs with cycles and latent confounders: An exact branch-and-bound approach. *International Journal of Approximate Reasoning* 117:29–49.

Rapberger, A.; Ulbricht, M.; and Toni, F. 2024. On the correspondence of non-flat assumption-based argumentation and logic programming with negation as failure in the head. *CoRR* abs/2405.09415.

Richardson, T., and Spirtes, P. 1999. Automated Discovery of Linear Feedback Models. In *Computation, Causation, and Discovery*. AAAI Press.

Russo, F., and Toni, F. 2023. Causal discovery and knowledge injection for contestable neural networks. In *Proceedings of the 26th European Conference on Artificial Intelligence (ECAI'23)*, volume 372 of *FAIA*, 2025–2032. IOS Press.

Russo, F.; Rapberger, A.; and Toni, F. 2024. Argumentative causal discovery. *CoRR* abs/2405.11250.

Russo, F. 2023. Argumentation for interactive causal discovery. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI 2023*, 7091–7092. ijcai.org.

Schölkopf, B.; Locatello, F.; Bauer, S.; Ke, N. R.; Kalchbrenner, N.; Goyal, A.; and Bengio, Y. 2021. Toward causal representation learning. *Proceedings of the IEEE* 109(5):612–634.

Schulz, C., and Toni, F. 2015. Logic programming in assumption-based argumentation revisited - semantics and graphical representation. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI'15)*, 1569–1575. AAAI Press.

Scutari, M. 2014. Bayesian network repository. http://www.bnlearn.com/bnrepository.

Sellke, T.; Bayarri, M.; and Berger, J. O. 2001. Calibration of ρ values for testing precise null hypotheses. *The American Statistician* 55(1):62–71.

Spirtes, P.; Glymour, C. N.; and Scheines, R. 2000. *Causation, prediction, and search*. MIT press.

Triantafillou, S.; Tsamardinos, I.; and Roumpelaki, A. 2014. Learning neighborhoods of high confidence in constraint-based causal discovery. In *Proceedings of the 7th European Workshop on Probabilistic Graphical Models (PGM'14)*, 487–502. Springer.

Tsamardinos, I.; Brown, L. E.; and Aliferis, C. F. 2006. The max-min hill-climbing bayesian network structure learning algorithm. *Machine learning* 65:31–78.

Ulbricht, M.; Potyka, N.; Rapberger, A.; and Toni, F. 2024. Non-flat ABA is an instance of bipolar argumentation. In *Proceedings of the 38th AAAI Conference on Artificial Intelligence (AAAI'24)*, 10723–10731. AAAI Press.

Vowels, M. J.; Camgoz, N. C.; and Bowden, R. 2022. D'ya like dags? a survey on structure learning and causal discovery. *ACM Computing Surveys* 55(4):1–36.

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

Zhang, K.; Peters, J.; Janzing, D.; and Schölkopf, B. 2011. Kernel-based conditional independence test and application in causal discovery. In *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence (UAI'11)*, 804–813. AUAI Press.

Zheng, X.; Dan, C.; Aragam, B.; Ravikumar, P.; and Xing, E. 2020. Learning sparse nonparametric DAGs. In *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS'20)*, volume 108 of *PMLR*, 3414–3425. PMLR.