Constraint-based Causal Discovery:
Conflict Resolution with Answer Set Programming

Antti Hyttinen and **Frederick Eberhardt**
California Institute of Technology
Pasadena, CA, USA

Matti Järvisalo
HIIT & Department of Computer Science
University of Helsinki, Finland

Abstract

Recent approaches to causal discovery based on Boolean satisfiability solvers have opened new opportunities to consider search spaces for causal models with both feedback cycles and unmeasured confounders. However, the available methods have so far not been able to provide a principled account of how to handle conflicting constraints that arise from statistical variability. Here we present a new approach that preserves the versatility of Boolean constraint solving *and* attains a high accuracy despite the presence of statistical errors. We develop a new logical encoding of (in)dependence constraints that is both well suited for the domain and allows for faster solving. We represent this encoding in Answer Set Programming (ASP), and apply a state-of-the-art ASP solver for the optimization task. Based on different theoretical motivations, we explore a variety of methods to handle statistical errors. Our approach currently scales to cyclic latent variable models with up to seven observed variables and outperforms the available constraint-based methods in accuracy.

1 INTRODUCTION

The search for causal relations underlies many scientific fields. Unlike mere correlational information, causal relations support predictions of how a system will behave when it is subject to an intervention. In the causal Bayes net framework (Spirtes et al., 1993; Pearl, 2000) the causal structure is represented in terms of a directed graph (see Figure 1). One of the most widely applicable approaches to discovering the causal structure uses independence and dependence constraints obtained from statistical tests to narrow down the candidate graphs that may have produced the data. Such an inference relies on the now well-understood assumptions of *causal Markov* and *causal faithfulness* (Spirtes et al., 1993). Unlike many other approaches, these constraint-based causal discovery methods can allow for the presence of latent confounders, feedback cycles and the utilisation of several (partially overlapping) observational or experimental data sets.

Even without experimentation (or additional assumptions, such as time order), and despite the generality of the model space, constraint-based methods can infer some causal orientations on the basis of *v-structures* (unshielded colliders). A v-structure in a graph is a triple of variables, such as $\langle x, z, y \rangle$ in Figure 1, where z is a common child of x and y, but x and y are non-adjacent in the graph. V-structures can be identified because of the specific (in)dependence relations they imply (here, $x \not\perp\!\!\!\perp z$, $z \not\perp\!\!\!\perp y$ and $x \perp\!\!\!\perp y$ are jointly sufficient to identify the v-structure). The edges that are thus oriented provide the basis for all further orientation inferences in constraint-based algorithms such as PC and FCI (Spirtes et al., 1993); e.g. identifying the v-structure in Figure 1 enables the additional orientation of the zw-edge. However, when processing sample data, the above inference is often prone to error. Establishing the further dependence $x \not\perp\!\!\!\perp y \mid z$ would confirm the inference. If this dependence does not hold, we have a case of a *conflict*: There is no causal graph that satisfies all available (in)dependence constraints (while respecting Markov and faithfulness).

The problem of conflicting constraints is exacerbated when trying to integrate multiple observational and experimental data sets in which the sets of measured variables *overlap* only partially. Unlike the case for one passively observed data set, the characterization of the class of graphs consistent with the (in)dependence constraints is more difficult in this setting: the graphs may disagree on orientations, adjacencies, and ancestral relations (Tsamardinos et al., 2012). Triantafillou et al. (2010) and Hyttinen et al. (2013) have started using general Boolean satisfiability (SAT) solvers (Biere et al., 2009) to integrate the various constraints. The basic idea of these methods is to convert the (in)dependence constraints found in the data into logical constraints on the presence and absence of certain pathways in the underlying causal structure, and to use a

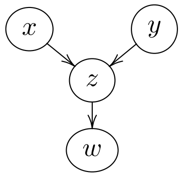


Figure 1: Example causal graph (see text for details).

SAT-solver to find the causal structures consistent with all constraints.

For pure SAT methods, conflicted constraints imply unsatisfiability. For these methods to work at all in practice, conflict handling is paramount. We address this problem at a variety of levels. We represent the task as a constraint optimization problem, and discuss in Section 3 different weighting schemes of the constraints and the theoretical motivations that support them. Then, we present in Section 5 a new encoding that relates (in)dependence constraints directly to one another via the operations of *intervening*, *conditioning* and *marginalization*. The encoding naturally captures the central ideas of constraint integration as it more directly connects constraints that refer to the same graphical neighborhood. We represent this encoding in the constraint optimization paradigm of Answer Set Programming (ASP) (Gelfond and Lifschitz, 1988; Niemelä, 1999; Simons et al., 2002). Finally, we compare the reliability of our proposed methods with available existing algorithms in Section 7.

2 PRELIMINARIES

For notational simplicity, we restrict the presentation to the setting with a single passive observational data set. However, the approach extends naturally to the general case of multiple overlapping experimental data sets; details are provided in the supplement.

We consider the class $\mathcal{G}$ of *causal graphs* of the form $G = (\mathbf{V}, \mathbf{E})$, where $\mathbf{V}$ is the set of nodes (associated with random variables) of G, and the edge relation $\mathbf{E} = \mathbf{E}_{\rightarrow} \cup \mathbf{E}_{\leftrightarrow}$ is composed of a set $\mathbf{E}_{\rightarrow}$ of directed edges and a set $\mathbf{E}_{\leftrightarrow}$ of bi-directed edges. A bi-directed edge $\leftrightarrow$ represents a *latent confounder*, i.e. an unmeasured common cause of two or more of the observed variables. In other words, we allow for the presence of feedback cycles and do not assume causal sufficiency. We define a *path* as a sequence of consecutive edges in the graph, without any restrictions on the types or orientations of the edges involved. A vertex appears as a *collider* on a path if both its adjacent edges on the path point into the vertex.

(In)dependence constraints can be brought into correspondence with structural properties of the graph using the d-separation criterion (Pearl, 2000). A path in graph G is *d-connecting* with respect to a conditioning set $\mathbf{C}$ if every collider c on the path is in $\mathbf{C}$ and no other nodes on the path are in $\mathbf{C}$; otherwise the path is *d-separated* (or "blocked"). A pair of nodes are d-connected given a conditioning set $\mathbf{C}$ if there is at least one d-connecting path between them; otherwise they are d-separated.[1] Under the causal Markov and faithfulness assumptions, two variables x and y are independent conditional on a set of variables $\mathbf{C}$ iff x and y are d-separated given $\mathbf{C}$ in the graph G.[2]

Let $\mathbf{D}$ be an i.i.d. data set sampled from a distribution that is Markov and faithful to an underlying "true" causal graph $G_t = (\mathbf{V}, \mathbf{E})$. Overall, the aim of causal discovery is to recover as many properties of G_t as possible from the data $\mathbf{D}$. There are a variety of ways to proceed. In light of the generality of the search space we consider, we focus on the graphical constraints implied by the conditional (in)dependencies found in the data. Our proposal is that the causal discovery problem is addressed well by solutions to the following abstract constrained optimization problem.

Problem Statement

INPUT: A set $\mathbf{K}$ of conditional independence and dependence constraints over a set $\mathbf{V}$ of variables, and a non-negative weight (cost) $w(k)$ for each $k \in \mathbf{K}$.

TASK: Find a causal graph G^* over the vertex set $\mathbf{V}$ such that

$$G^* \in \underset{G \in \mathcal{G}}{\arg\min} \sum_{k \in \mathbf{K}\,:\,G \not\models k} w(k). \tag{1}$$

In words, our goal is to find a single representative graph G^* that minimizes the sum of the weights of the given conditional independence and dependence constraints *not* implied by G^*. The idea is that the constraints can be weighted according to their reliability, and that conflicts among the constraints are well-resolved when the sum of the weights of the constraints not satisfied by the output graph are minimized. We take the set $\mathbf{K}$ to be the set of all (in)dependence constraints testable in the data $\mathbf{D}$.[3]

This formalization poses two key challenges: 1) How to define the weighting scheme $w(k)$ such that the solutions (causal graphs) are "as similar as possible" to the true graph? 2) How to actually find a representative graph G^*, i.e., how to represent the constraints efficiently such that one can minimize the objective function (an NP-hard optimization problem) defined above? In the following, we will

[1] This definition is equivalent to Pearl's standard definition (Studený, 1998; Koster, 2002).

[2] See Spirtes (1995), Pearl and Dechter (1996) and Neal (2000) for discussions of d-separation in cyclic models.

[3] When considering experimental data sets, the weights of (in)dependence constraints of the *manipulated* distributions enter into the minimization as well; see Appendix A.

propose solutions to both of these challenges. We note that once a representative graph G^* has been found, the independence oracle methods of Hyttinen et al. (2013) can be used to understand the common properties, such as adjacencies and ancestral relations, of all solutions that satisfy the same set of constraints.

3 PREVIOUS WORK

Statistical variability results in incorrect test results, but such errors are only detectable when the test results have conflicting implications. Conflicts can arise directly between test results or, more commonly, in combination with search space assumptions. Whether or not conflicts are detected depends on the tests that have been performed. Thus, one way of "handling" conflicts is to avoid performing tests whose results can conflict, or not to draw the full set of possible inferences from a set of test results. This is what the standard PC-algorithm does; it basically avoids conflicts.[4] This approach proved to be unreliable for the detection of v-structures, so a *conservative* extension was added that checks additional tests to verify the inference to the v-structures (see example in Section 1). The resulting cPC-algorithm (Ramsey et al., 2006) marks any conflicted v-structures and abstains from the orientation inference, i.e. it returns a "don't know" for this part of the graph. The FCI and cFCI algorithms are analogous in this regard to PC and cPC, respectively, only that the search space is enlarged to allow for latent confounders. A more generic approach, not focused on v-structures, reduces the occurrence of conflicts by controlling which test results are accepted. Li and Wang (2009) control the false discovery rate of the tests that PC performs, while Tsamardinos et al. (2012) use different p-value thresholds to infer independence vs. dependence constraints. Very recently, Triantafillou and Tsamardinos (2014) developed a scheme that uses the p-values from tests performed by FCI to rank (a specific class of) structural constraints on the underlying graph, and then use a SAT-based procedure to satisfy as many constraints as possible. The selection of tests by FCI depends on the results of earlier tests. Thus the conflict resolution (in terms of the ranked constraints) only handles conflicts between tests that were selected. This scales very well, but the theoretical account of the accuracy of the output model is unclear, as the selection of tests interacts with the conflict resolution.

Score-based algorithms take a very different approach to inconsistencies in data (Cooper and Herskovits, 1992; Chickering, 2002). Instead of explicitly determining the (in)dependence constraints (and finding them to be inconsistent) the conflicts are implicitly resolved by a direct integration of the data points into a score that identifies the graph (or equivalence class) that maximizes the Bayesian posterior. Such an approach provides a clear theoretical account in which sense the output is "closest" to the true graph (equivalence class). Although mostly restricted to DAGs, Claassen and Heskes (2012) have transfered some of the advantages of the Bayesian approach to a constraint-based search method over models with latent confounders. Their BCCD algorithm builds on the skeleton search of PC, and computes Bayesian-style probabilities for the (conditional) independence constraints.

4 WEIGHTING SCHEMES

Given the general search space we are considering, we take a constraint-based approach. But unlike other such methods we do not select tests to perform based on previous test results. Instead, for a given data set, we consider *all* independence tests that can be performed on the set of variables, and apply the following weighting schemes to the resulting constraints.

4.1 Controlling False Negatives

One of the problems for constraint based causal discovery are false negative results, i.e. variables that are truly dependent but test as independent due to low sample sizes or several cancelling d-connecting paths between them (violations of faithfulness). Strictly speaking, classical statistical tests do not license the inference to independence when the null-hypothesis H_0 of independence fails to be rejected. Schulte et al. (2010) have developed a search procedure for causal DAGs based on *dependence* constraints, which are licensed by classical tests when H_0 is rejected. Independencies enter only as a result of a simplicity assumption. Analogously, we propose to control the false negatives with a given sufficiently low p-value threshold on the independence tests. We enforce the detected dependencies as hard constraints. Dependence constraints on their own cannot conflict, as the complete graph will satisfy all possible dependencies. But the principle of Occam's Razor recommends choosing the simplest among the models able to produce the data. We take this here to amount to maximizing the number of independencies given the dependencies (which is closely related to the dependence minimality proposed by Pearl (2000)). Thus, if we partition the set of constraints $\mathbf{K}$ into the independence constraints $\mathbf{K}_{\perp\!\!\!\perp}$ and dependence constraints $\mathbf{K}_{\not\perp\!\!\!\perp}$, then the causal discovery problem over the class of causal models $\mathcal{G}$ amounts to solving the constrained optimization problem in (1) with a weight function

$$w(k) = \begin{cases} \infty & \text{if } k \in \mathbf{K}_{\not\perp\!\!\!\perp} \\ 1 & \text{if } k \in \mathbf{K}_{\perp\!\!\!\perp}. \end{cases} \qquad (2)$$

[4] Some PC implementations do in some cases infer the presence of a latent confounder (violating the model space) when v-structures are incorrectly detected.

4.2 Controlling False Positives and Negatives

In a slight but common abuse of classical statistics, one can treat the failure to reject H_0 as the acceptance of independence. Then one can simply find the graph that minimizes the number of disagreements between the (in)dependence constraints implied by the graph and the test results. Such an approach would then also be able to recover from false positive errors, i.e. true independencies that test as dependencies. Such false positive errors may occur, for example, if the true structure is a chain $x \to z \to y$ and we condition on a measured, but noisy, version of z, and we still get that $x \not\perp\!\!\!\perp y \mid z$. The corresponding weight-function for the the constrained optimization problem in (1) is then simply

$$w(k) = 1 \quad \forall k \in \mathbf{K}. \tag{3}$$

4.3 Weighted Constraints

So far we have treated the test results as binary, but in many cases one has reason to be more confident about some test results than others, and the constraints could be weighted accordingly. One would like to associate a probability as a measure of confidence with each constraint, since then the independence test could be treated as a probabilistic classifier without a hard decision between independence and dependence. When ground truth is available the quality of probabilistic classifiers is often compared by proper scoring rules, such as the log-score. Proper scoring rules assign costs that are minimized when the tests or classifier return the true probability of class membership. We use such scoring rules here as cost functions: we find the graph G^* such that if it were the ground truth, the results of the probabilistic classifier would be optimal – minimizing the cost assigned by the proper score. Given a data set $\mathbf{D}$, for each constraint k the classifier returns the probability $P(k \mid \mathbf{D})$ for k to hold in G^*. If in fact k holds in G^*, then the classifier should only suffer the cost $-\log P(k \mid \mathbf{D})$, otherwise the cost is $-\log[1 - P(k \mid \mathbf{D})]$. As the minimum of these costs will always be suffered, it is sufficient to let

$$w(k) = \log P(k \mid \mathbf{D}) - \log[1 - P(k \mid \mathbf{D})], \tag{4}$$

which is positive, since only the constraint with higher probability is included in $\mathbf{K}$.[5]

It is not straightforward, neither in terms of theoretical foundation nor actual implementation, to turn p-values from classical tests into probability estimates, as the distribution of p-values under H_0 is uniform and only known to be decreasing under H_1.[6] Instead, Margaritis and Bromberg (2009) use a Bayesian paradigm to assign probabilities to the (in)dependence statements.[7] Following them, for each independence statement $x \perp\!\!\!\perp y \mid \mathbf{C}$, we consider two models $M_{\perp\!\!\!\perp}$ and $M_{\not\perp\!\!\!\perp}$, where $M_{\perp\!\!\!\perp} : P(x, y \mid \mathbf{C}) = P(x \mid \mathbf{C})P(y \mid \mathbf{C})$ postulates independence, while $M_{\not\perp\!\!\!\perp} : P(x, y \mid \mathbf{C}) = P(x \mid \mathbf{C})P(y \mid x, \mathbf{C})$ postulates dependence. Given data $\mathbf{D}$ and a prior $P(M_{\perp\!\!\!\perp}) = \alpha$ the probability associated with $k = x \perp\!\!\!\perp y \mid \mathbf{C}$ simplifies to

$$P(k \mid \mathbf{D}) = \frac{P(y \mid \mathbf{C})\alpha}{P(y \mid \mathbf{C})\alpha + P(y \mid x, \mathbf{C})(1 - \alpha)}.$$

The marginal likelihoods $P(y \mid \mathbf{C})$ and $P(y \mid x, \mathbf{C})$ correspond to local scores in the score-based learning framework and have a closed form for categorical variables using a Dirichlet prior (Cooper and Herskovits, 1992) or for continuous variables with linear relations and Gaussian disturbances using an inverse Wishart Gaussian prior (Geiger and Heckerman, 1994).

5 A RECURSIVE VIEW TO CAUSAL GRAPHS

Given a set of (in)dependence constraints, finding an optimal graph G^* (in the sense of the problem statement's objective function Eq. 1) requires a formulation of the d-connection property that is suitable for constraint solvers. Hyttinen et al. (2013) provided such a formulation in terms of propositional logic, but that proves to be inefficient for the computationally more demanding case with conflicted constraints (see Section 7).

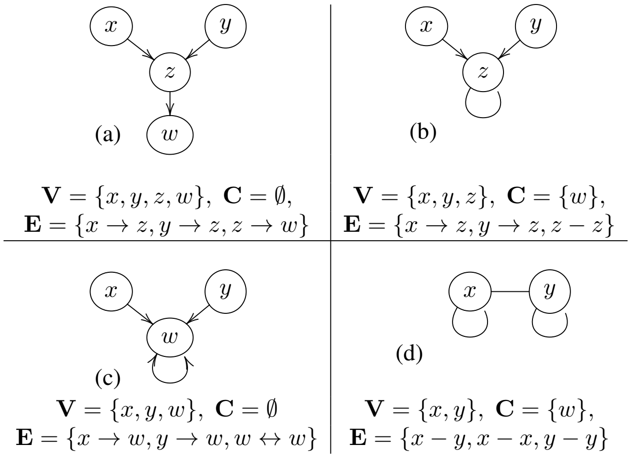
Our new formulation is based on the two central graph operations that relate the underlying causal graph to the (in)dependence constraints obtained from an observational data set: *conditioning* and *marginalization*. (Intervening is treated in the supplementary material.) We define these operations over objects that we call *d-connection graphs* rather than MAGs/PAGs (Richardson and Spirtes, 2002), as we want allow for the complete generality of the model space.

A *d-connection graph* $H = (\mathbf{V}, \mathbf{E})_{\mathbf{C}}$ is defined relative to a set $\mathbf{C}$ such that: (i) $\mathbf{V}$ is the set of variables in the graph, (ii) the edge relation $\mathbf{E} = \mathbf{E}_{\rightarrow} \cup \mathbf{E}_{\leftrightarrow} \cup \mathbf{E}_{-}$ is composed of directed, bidirected and undirected edges among $\mathbf{V}$, and (iii) the set $\mathbf{C}$ denotes the set of conditioned variables with the restriction that $\mathbf{V} \cap \mathbf{C} = \emptyset$. The disjoint sets $\mathbf{V}$ and $\mathbf{C}$ are used to keep track of which variables have been subject to the two operations. Given a causal graph $G = (\mathbf{V}, \mathbf{E})$ (Section 2), the corresponding d-connection graph H has the same sets $\mathbf{V}$, $\mathbf{E}_{\rightarrow}$ and $\mathbf{E}_{\leftrightarrow}$, but in addition the set $\mathbf{E}_{-} = \emptyset$ (no undirected edges) and $\mathbf{C} = \emptyset$ (no conditioning). The d-connection graph of the causal graph in

[5] In Appendix B we show how the log-weights can be interpreted probabilistically.

[6] Some proposals in this direction appear in a recent unpublished paper by Triantafillou and Tsamardinos (2014).

[7] Claassen and Heskes (2012) also describe a way of obtaining Bayesian probabilities for independence statements, which is particularly accurate for finding minimal independencies in an acyclic domain.



(a) $\mathbf{V} = \{x, y, z, w\},\ \mathbf{C} = \emptyset,$ $\mathbf{E} = \{x \to z, y \to z, z \to w\}$

(b) $\mathbf{V} = \{x, y, z\},\ \mathbf{C} = \{w\},$ $\mathbf{E} = \{x \to z, y \to z, z - z\}$

(c) $\mathbf{V} = \{x, y, w\},\ \mathbf{C} = \emptyset$ $\mathbf{E} = \{x \to w, y \to w, w \leftrightarrow w\}$

(d) $\mathbf{V} = \{x, y\},\ \mathbf{C} = \{w\},$ $\mathbf{E} = \{x - y, x - x, y - y\}$

Figure 2: Graph operations on d-connection graphs: (a) original graph, (b) after conditioning on w, (c) after marginalizing z, (d) after conditioning on w and marginalizing z in either order. The xy-edge in (d) shows that x and y are dependent when conditioning on z and marginalizing w.

Figure 1 is shown in Figure 2a; the edges correspond exactly. In general, an edge $x_{a} - {}_{b}y$ in a d-connection graph $H = (\mathbf{V}', \mathbf{E}')_{\mathbf{C}'}$ denotes the existence of a path p with the given edge ends a and b at x and y in the underlying causal graph G, such that p is d-connecting with respect to $\mathbf{C}'$ and does not go through other variables in $\mathbf{V}'$. Consequently, the d-connection property (as defined in Section 2) can be directly applied to any d-connection graph.

Given a d-connection graph $H = (\mathbf{V}, \mathbf{E})_{\mathbf{C}}$, the *conditioning* operation $c(H, w)$ on a variable $w \in \mathbf{V}$ results in a d-connection graph $H' = (\mathbf{V} \setminus w, \mathbf{E}')_{\mathbf{C} \cup w}$, where $\mathbf{E}'$ is related to $\mathbf{E}$ by (i) including in $\mathbf{E}'$ any edges in $\mathbf{E}$ not involving w; (ii) adding to $\mathbf{E}'$ an edge $x_{a} - {}_{b}y$ if there are edges $x_{a} \to w$ and $w \leftarrow {}_{b}y$ in $\mathbf{E}$, and (iii) not permitting any other edges in $\mathbf{E}'$ (x and y can be equal above).

The graphs in the right column of Figure 2 are formed by applying the operation $c(\cdot, w)$ to the graphs in the left column of Figure 2. The conditioning operation is a little unusual because the conditioned variable is removed from the graph and undirected edges and undirected self-cycles are introduced. As is well-known, conditioning on a collider or a child of a collider results in a d-connection between the collider's parents. For example, if w is conditioned on in $x \to w \leftarrow y$, we have a d-connecting path between x and y with a tail at either end. Since the encoding also removes the conditioned variable from the graph, we just have an undirected edge $x - y$, the parents are "moralized". The undirected self-loop represents the same idea, just in case of the child of a collider: each parent of a conditioned variable receives an undirected self-loop to indicate that it can provide a d-connection between two incoming paths. In Figure 2a-b this is illustrated for variable z when w is added to the conditioning set (and removed from the graph). The graph in (b) indicates that there is a tail to tail d-connection at z, which implies that x and y are now d-connected when w, the child of collider z, is in the conditioning set. The (perhaps unintuitive) removal of the conditioning variable from the graph achieves two goals: it reduces the size of the graph to be encoded, but more importantly, it incrementally represents the effect of conditioning on the d-connections among the other variables.

Given a d-connection graph $H = (\mathbf{V}, \mathbf{E})_{\mathbf{C}}$, the *marginalization* operation $m(H, z)$ for variable $z \in \mathbf{V}$ results in a d-connection graph $H' = (\mathbf{V} \setminus z, \mathbf{E}')_{\mathbf{C}}$, where $\mathbf{E}'$ is related to $\mathbf{E}$ by (i) including in $\mathbf{E}'$ any edges in $\mathbf{E}$ not involving z; (ii) adding to $\mathbf{E}'$ an edge $x_{a} - {}_{b}y$ if there are edges $x_{a} - z$ and $z_{c} - {}_{b}y$ in $\mathbf{E}$; (iii) adding to $\mathbf{E}'$ an $x_{a} - {}_{b}y$ if there are edges $x_{a} \to z$, $z - z$ and $z \leftarrow {}_{b}y$ in $\mathbf{E}$, and (iv) not permitting any other edges in $\mathbf{E}'$ (x and y can be equal above). Marginalization follows the standard graphical procedures used elsewhere, except that a little more book-keeping is required to track the d-connections resulting from self-loops and undirected edges (see Figure 2 top to bottom).

The following theorem shows that the operations preserve the d-connection properties among the variables still present in the graph after the operation (proof in Appendix D).

Theorem 1 *Let $H' = (\mathbf{V}', \mathbf{E}')_{\mathbf{C}'}$ be the d-connection graph obtained from a d-connection graph $H = (\mathbf{V}, \mathbf{E})_{\mathbf{C}}$ by applying the conditioning operation on w, i.e. $H' = c(H, w)$ (or by applying the marginalization operation on z, thus $H' = m(H, z)$). Then there is a path of type $x_{a} \cdots {}_{b}y$ that is d-connecting given $\mathbf{C}'' \supseteq \mathbf{C}'$ in H' if and only if there is a path of type $x_{a} \cdots {}_{b}y$ that is d-connecting given $\mathbf{C}''$ in H.*

Consequently, a dependence $x \not\perp\!\!\!\perp y \mid \mathbf{C}$ in (the true causal graph) $G = (\mathbf{V}, \mathbf{E})$ is equivalent to having an edge of some type in the d-connection graph $H = (\mathbf{V}', \mathbf{E}')_{\mathbf{C}}$ when $\mathbf{V}' = \{x, y\}$. This d-connection graph H can be obtained by consecutively applying the conditioning operation to all variables in $\mathbf{C}$, and the marginalization operation to the rest in $\mathbf{V} \setminus (\mathbf{C} \cup \{x, y\})$. For example, in Figure 2d the undirected edge between x and y represents the fact that $x \not\perp\!\!\!\perp y \mid w$ in the underlying causal graph (Figure 1).

Given a causal graph G, we can calculate all of its implied (in)dependence relations by applying the operations in any order. However, we need not apply all operations for each (in)dependence relation, since we can exploit the compact intermediary representations of the d-connections in the d-connection graphs obtained earlier. Figure 3 shows the "encoding DAG" for one set of applied operations by which we obtain all (in)dependence relations of the four variable graph in Figure 1. The true causal graph corresponds to the node in the middle, and we use the operations to move

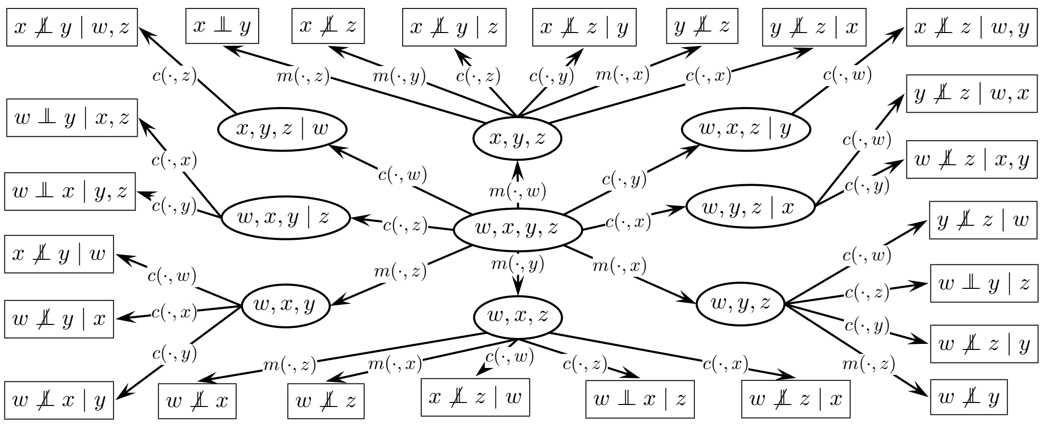

$x \not\perp\!\!\!\perp y \mid w,z$
$x \perp\!\!\!\perp y$
$x \not\perp\!\!\!\perp z$
$x \not\perp\!\!\!\perp y \mid z$
$x \not\perp\!\!\!\perp z \mid y$
$y \not\perp\!\!\!\perp z$
$y \not\perp\!\!\!\perp z \mid x$
$x \not\perp\!\!\!\perp z \mid w,y$
$c(\cdot,z)$
$m(\cdot,z)$
$m(\cdot,y)$
$c(\cdot,z)$
$c(\cdot,y)$
$m(\cdot,x)$
$c(\cdot,x)$
$c(\cdot,w)$
$w \perp\!\!\!\perp y \mid x,z$
$x,y,z \mid w$
x,y,z
$w,x,z \mid y$
$y \not\perp\!\!\!\perp z \mid w,x$
$c(\cdot,x)$
$c(\cdot,w)$
$m(\cdot,w)$
$c(\cdot,y)$
$c(\cdot,w)$
$w \not\perp\!\!\!\perp z \mid x,y$
$c(\cdot,y)$
$w \perp\!\!\!\perp x \mid y,z$
$c(\cdot,y)$
$w,x,y \mid z$
$c(\cdot,z)$
w,x,y,z
$c(\cdot,x)$
$w,y,z \mid x$
$y \not\perp\!\!\!\perp z \mid w$
$x \not\perp\!\!\!\perp y \mid w$
$m(\cdot,z)$
$m(\cdot,y)$
$m(\cdot,x)$
$c(\cdot,w)$
$c(\cdot,w)$
w,x,y
w,y,z
$c(\cdot,z)$
$w \perp\!\!\!\perp y \mid z$
$w \not\perp\!\!\!\perp y \mid x$
$c(\cdot,x)$
w,x,z
$c(\cdot,y)$
$c(\cdot,y)$
$w \not\perp\!\!\!\perp z \mid y$
$m(\cdot,z)$
$m(\cdot,x)$
$c(\cdot,w)$
$c(\cdot,z)$
$c(\cdot,x)$
$m(\cdot,z)$
$w \not\perp\!\!\!\perp x \mid y$
$w \not\perp\!\!\!\perp x$
$w \not\perp\!\!\!\perp z$
$x \not\perp\!\!\!\perp z \mid w$
$w \perp\!\!\!\perp x \mid z$
$w \not\perp\!\!\!\perp z \mid x$
$w \not\perp\!\!\!\perp y$

Figure 3: An "encoding DAG": a tree connecting the causal graph (represented by the middle node) to (in)dependence constraints (leaves). The conditioning/marginalization operations are marked on the arrows. The circular nodes represent the sets $\mathbf{V} \mid \mathbf{C}$ of the corresponding d-connection graphs and the square nodes show the implied independencies for the true causal graph shown in Figure 1.

outward until we reach the implied (in)dependencies at the leaves.

For causal discovery we want to perform the backward inference. We know the (in)dependence relations at the leaves (Figure 3) and our aim is to find a causal graph in the middle node. Again we exploit the same tree structure. But this time we will have at each node several different graphs that satisfy the constraints, which are downstream from it. Moving towards the center, nodes can be combined to find the options that satisfy all constraints downstream from the node. For example, in the node 'x, y, z' all the tests relevant to identifying the v-structure $x{\scriptstyle a} \rightarrow z \leftarrow {\scriptstyle b} y$ are available locally. In the general case of weighted and possibly conflicting (in)dependence constraints, the inference becomes hard. We use off-the-shelf solvers for this job. Nevertheless, as shown by the simulations, the recursive structure allows the backwards inference to work faster than for the logical formulations of d-connection in Hyttinen et al. (2013).

6 CAUSAL DISCOVERY VIA ASP

Building on Section 5, we describe here an ASP-based constraint optimization approach to optimally solving the causal structure discovery task defined in (1).

Answer set programming (ASP) is a rule-based declarative constraint satisfaction paradigm that is well-suited for representing and solving various computationally hard problems (Gelfond and Lifschitz, 1988; Niemelä, 1999; Simons et al., 2002). ASP offers an expressive declarative modelling language in terms of first-order logical rules, allowing for intuitive and compact representations of NP-hard optimization tasks. When using ASP, the first task is to model the problem in terms of ASP rules (constraints) so that the set of solutions implicitly represented by the ASP rules corresponds to the solutions of the original problem. One or multiple solutions of the original problem can then be obtained by invoking an off-the-shelf ASP solver on the constraint declaration.

6.1 An ASP Encoding of Causal Discovery

As a self-contained explanation of ASP syntax and semantics would exceed the page limit, we only aim to give an intuitive reading of our ASP encoding. The ASP encoding, outlined in Figure 4, is based on exactly representing the conditioning and marginalization operations (defined in Section 5) in ASP.

Answer set programming can be viewed as a data-centric constraint satisfaction paradigm, in which the input data, represented as "facts" that are true via input predicates, express the instance of the original problem at hand. In our case, the problem instance consists of a set of independence and dependence constraints and their associated weights, represented via the predicates *indep* and *dep*, and the "encoding DAG", describing which d-connection graphs can be mapped from one to the other via the conditioning and marginalization operations (predicates *cond* and *marg*). Concretely, the input predicates $indep(x, y, \{x, y\}, \mathbf{C}, W)$ and $dep(x, y, \{x, y\}, \mathbf{C}, W)$ represent as facts that the input contains an independence (resp., dependence) constraint with weight W over the variables x and y given the conditioning set $\mathbf{C}$.[8] The input predicates $cond(\mathbf{V}, \mathbf{C}, z)$ and $marg(\mathbf{V}, \mathbf{C}, z)$ enable the conditioning and marginalizing of a variable z, respectively, in a d-connection graph that has exactly the variables $\mathbf{V}$ and conditioning set $\mathbf{C}$. Es-

[8] In practice, the ASP language requires integer-valued weights. For sufficient precision, in our experiments we multiply the weights by 1000, and then truncate to integers.

sentially, $cond$ and $marg$ represent edges in the encoding DAG (recall Figure 3).

The other predicates $tt(x, y, \mathbf{V}, \mathbf{C})$, $th(x, y, \mathbf{V}, \mathbf{C})$, and $hh(x, y, \mathbf{V}, \mathbf{C})$ present the existence of different types of edges (tt: tail-tail, th: tail-head, hh: head-head) in the d-connection graph with variable set $\mathbf{V}$ and conditioning set $\mathbf{C}$. The rules associated with the conditioning and marginalization operations encode how the different edges in the d-connection graphs are derived from edges in other d-connection graphs through the conditioning and marginalization operations. The restrictions that the (in)dependence constraints put on the set of solutions (causal graphs) follow from these rules. As an example, the first marginalization rule

$$th(x, y, \mathbf{V} \setminus z, \mathbf{C}) \text{ :- } th(x, y, \mathbf{V}, \mathbf{C}),\ marg(\mathbf{V}, \mathbf{C}, z).$$

allows to derive, for any choice of $\mathbf{C}$, $\mathbf{V}$, $z \in \mathbf{V}$, and $x, y \in \mathbf{V} \setminus z$, that $th(x, y, \mathbf{V} \setminus z, \mathbf{C})$ is true (i.e., that there is an edge $x \to y$ in the d-connection graph over $\mathbf{V} \setminus z$ relative to $\mathbf{C}$) given that (i) $th(x, y, \mathbf{V}, \mathbf{C})$ is true (there is an edge $x \to y$ in the d-connection graph over $\mathbf{V}$ relative to $\mathbf{C}$), *and* (ii) the input contains the fact $marg(\mathbf{V}, \mathbf{C}, z)$, i.e., marginalizing z in the d-connection graph over $\mathbf{V}$ relative to $\mathbf{C}$ is allowed by the encoding DAG. Note that the set of derivation rules for $tt(x, y, \mathbf{V}, \mathbf{C})$, $th(x, y, \mathbf{V}, \mathbf{C})$, and $hh(x, y, \mathbf{V}, \mathbf{C})$ are very similar to each other.

Conditioning on a variable $z \in \mathbf{V}, \forall x, y \in \mathbf{V} \setminus z$:

$$\begin{aligned}
th(x, y, \mathbf{V} \setminus z, \mathbf{C} \cup z) &\text{ :- } th(x, y, \mathbf{V}, \mathbf{C}), \mathit{cond}(\mathbf{V}, \mathbf{C}, z).\\
th(x, y, \mathbf{V} \setminus z, \mathbf{C} \cup z) &\text{ :- } th(x, z, \mathbf{V}, \mathbf{C}), hh(z, y, \mathbf{V}, \mathbf{C}),\\
&\qquad \mathit{cond}(\mathbf{V}, \mathbf{C}, z).
\end{aligned}$$

$$\begin{aligned}
hh(x, y, \mathbf{V} \setminus z, \mathbf{C} \cup z) &\text{ :- } hh(x, y, \mathbf{V}, \mathbf{C}), \mathit{cond}(\mathbf{V}, \mathbf{C}, z).\\
hh(x, y, \mathbf{V} \setminus z, \mathbf{C} \cup z) &\text{ :- } hh(x, z, \mathbf{V}, \mathbf{C}), hh(z, y, \mathbf{V}, \mathbf{C}),\\
&\qquad \mathit{cond}(\mathbf{V}, \mathbf{C}, z).
\end{aligned}$$

$$\begin{aligned}
tt(x, y, \mathbf{V} \setminus z, \mathbf{C} \cup z) &\text{ :- } tt(x, y, \mathbf{V}, \mathbf{C}), \mathit{cond}(\mathbf{V}, \mathbf{C}, z).\\
tt(x, y, \mathbf{V} \setminus z, \mathbf{C} \cup z) &\text{ :- } th(x, z, \mathbf{V}, \mathbf{C}), th(y, z, \mathbf{V}, \mathbf{C}),\\
&\qquad \mathit{cond}(\mathbf{V}, \mathbf{C}, z).
\end{aligned}$$

Marginalizing a variable $z \in \mathbf{V}, \forall x, y \in \mathbf{V} \setminus z$:

$$\begin{aligned}
th(x, y, \mathbf{V} \setminus z, \mathbf{C}) &\text{ :- } th(x, y, \mathbf{V}, \mathbf{C}), marg(\mathbf{V}, \mathbf{C}, z).\\
th(x, y, \mathbf{V} \setminus z, \mathbf{C}) &\text{ :- } tt(x, z, \mathbf{V}, \mathbf{C}), th(z, y, \mathbf{V}, \mathbf{C}),\\
&\qquad marg(\mathbf{V}, \mathbf{C}, z).\\
th(x, y, \mathbf{V} \setminus z, \mathbf{C}) &\text{ :- } th(x, z, \mathbf{V}, \mathbf{C}), th(z, y, \mathbf{V}, \mathbf{C}),\\
&\qquad marg(\mathbf{V}, \mathbf{C}, z).\\
th(x, y, \mathbf{V} \setminus z, \mathbf{C}) &\text{ :- } tt(x, z, \mathbf{V}, \mathbf{C}), hh(z, y, \mathbf{V}, \mathbf{C}),\\
&\qquad marg(\mathbf{V}, \mathbf{C}, z).\\
th(x, y, \mathbf{V} \setminus z, \mathbf{C}) &\text{ :- } th(x, z, \mathbf{V}, \mathbf{C}), tt(z, z, \mathbf{V}, \mathbf{C}),\\
&\qquad hh(z, y, \mathbf{V}, \mathbf{C}), marg(\mathbf{V}, \mathbf{C}, z).
\end{aligned}$$

$$\begin{aligned}
hh(x, y, \mathbf{V} \setminus z, \mathbf{C}) &\text{ :- } hh(x, y, \mathbf{V}, \mathbf{C}), marg(\mathbf{V}, \mathbf{C}, z).\\
hh(x, y, \mathbf{V} \setminus z, \mathbf{C}) &\text{ :- } th(z, x, \mathbf{V}, \mathbf{C}), th(z, y, \mathbf{V}, \mathbf{C}),\\
&\qquad marg(\mathbf{V}, \mathbf{C}, z).\\
hh(x, y, \mathbf{V} \setminus z, \mathbf{C}) &\text{ :- } hh(x, z, \mathbf{V}, \mathbf{C}), th(z, y, \mathbf{V}, \mathbf{C}),\\
&\qquad marg(\mathbf{V}, \mathbf{C}, z).\\
hh(x, y, \mathbf{V} \setminus z, \mathbf{C}) &\text{ :- } th(z, x, \mathbf{V}, \mathbf{C}), hh(z, y, \mathbf{V}, \mathbf{C}),\\
&\qquad marg(\mathbf{V}, \mathbf{C}, z).\\
hh(x, y, \mathbf{V} \setminus z, \mathbf{C}) &\text{ :- } hh(x, z, \mathbf{V}, \mathbf{C}), tt(z, z, \mathbf{V}, \mathbf{C}),\\
&\qquad hh(z, y, \mathbf{V}, \mathbf{C}), marg(\mathbf{V}, \mathbf{C}, z).
\end{aligned}$$

$$\begin{aligned}
tt(x, y, \mathbf{V} \setminus z, \mathbf{C}) &\text{ :- } tt(x, y, \mathbf{V}, \mathbf{C}), marg(\mathbf{V}, \mathbf{C}, z).\\
tt(x, y, \mathbf{V} \setminus z, \mathbf{C}) &\text{ :- } tt(x, z, \mathbf{V}, \mathbf{C}), tt(z, y, \mathbf{V}, \mathbf{C}),\\
&\qquad marg(\mathbf{V}, \mathbf{C}, z).\\
tt(x, y, \mathbf{V} \setminus z, \mathbf{C}) &\text{ :- } th(x, z, \mathbf{V}, \mathbf{C}), tt(z, y, \mathbf{V}, \mathbf{C}),\\
&\qquad marg(\mathbf{V}, \mathbf{C}, z).\\
tt(x, y, \mathbf{V} \setminus z, \mathbf{C}) &\text{ :- } tt(x, z, \mathbf{V}, \mathbf{C}), th(y, z, \mathbf{V}, \mathbf{C}),\\
&\qquad marg(\mathbf{V}, \mathbf{C}, z).\\
tt(x, y, \mathbf{V} \setminus z, \mathbf{C}) &\text{ :- } th(x, z, \mathbf{V}, \mathbf{C}), tt(z, z, \mathbf{V}, \mathbf{C}),\\
&\qquad th(y, z, \mathbf{V}, \mathbf{C}), marg(\mathbf{V}, \mathbf{C}, z).
\end{aligned}$$

Inferring failures to sat. (in)dep. $\forall x \forall y > x, \forall \mathbf{C}, \mathbf{V} = \{x, y\}$:

$$\begin{aligned}
\mathit{fail}(x, y, \mathbf{V}, \mathbf{C}, W) &\text{ :- } tt(x, y, \mathbf{V}, \mathbf{C}), \mathit{indep}(x, y, \mathbf{V}, \mathbf{C}, W).\\
\mathit{fail}(x, y, \mathbf{V}, \mathbf{C}, W) &\text{ :- } th(x, y, \mathbf{V}, \mathbf{C}), \mathit{indep}(x, y, \mathbf{V}, \mathbf{C}, W).\\
\mathit{fail}(x, y, \mathbf{V}, \mathbf{C}, W) &\text{ :- } th(y, x, \mathbf{V}, \mathbf{C}), \mathit{indep}(x, y, \mathbf{V}, \mathbf{C}, W).\\
\mathit{fail}(x, y, \mathbf{V}, \mathbf{C}, W) &\text{ :- } hh(x, y, \mathbf{V}, \mathbf{C}), \mathit{indep}(x, y, \mathbf{V}, \mathbf{C}, W).\\
\mathit{fail}(x, y, \mathbf{V}, \mathbf{C}, W) &\text{ :- } \text{not } th(x, y, \mathbf{V}, \mathbf{C}), \text{not } th(y, x, \mathbf{V}, \mathbf{C}),\\
&\qquad \text{not } hh(x, y, \mathbf{V}, \mathbf{C}), \text{not } tt(x, y, \mathbf{V}, \mathbf{C}),\\
&\qquad \mathit{dep}(x, y, \mathbf{V}, \mathbf{C}, W).
\end{aligned}$$

Weak constraints $\forall x \forall y > x, \forall \mathbf{C}, \mathbf{V} = \{x, y\}$:

$$:\sim \mathit{fail}(x, y, \mathbf{V}, \mathbf{C}, W).\ [W]$$

Figure 4: The ASP encoding.

Finally, the objective function under minimization (Equation 1) is expressed using the so-called *weak constraints* offered by the ASP language. First, the predicate *fail* is derived whenever the candidate solution disagrees with the input. The first four rules for *fail* denote cases where the constraints tested from the data suggest independence but an edge *tt*/*th*/*hh* is derived. The fifth rule derives *fail* whenever the input constraints suggest dependence but there are no d-connecting paths. The last rule of the encoding denotes the fact that whenever the *fail* predicate is derived, then the cost W is incurred. This implies that any solution produced by an ASP solver on the encoding is guaranteed to present an optimal solution to the causal discovery task, as stated by the following theorem.

Theorem 2 *Given a set $\mathbf{K}$ of conditional independence and dependence constraints over a set $\mathbf{V}$ of variables, and a non-negative weight $w(k)$ for each $k \in \mathbf{K}$, for any encoding DAG connecting the (in)dependence constraints, it holds that any optimal solution (minimizing the sum of the unsatisfied weak constraints) to the ASP encoding corresponds to a causal graph that minimizes the objective function Equation 1.*

To find optimal solutions to the encoded problem, the input facts together with the derivation rules and weak constraints are given as input to an ASP solver. The actual complete search for solutions is then performed over a propositional instantiation of the first-order rules; this process is automatized within the ASP solver and does not require involvement of the user. The search procedure implemented within state-of-the-art ASP solvers, such as `Clingo` (Gebser et al., 2011) used in this paper, is then based on the successful and highly-efficient Boolean satisfiability solver technology (Biere et al., 2009).

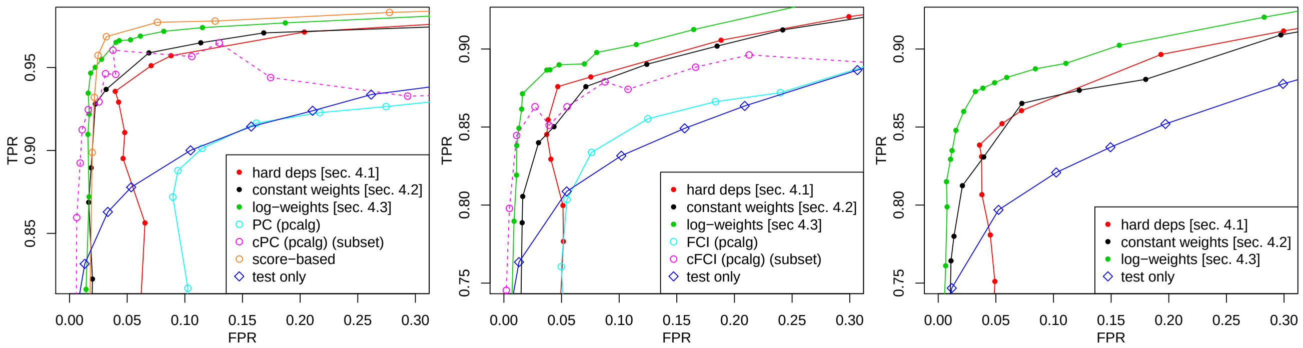


Figure 5: ROC given causally sufficient acyclic models (left), causally insufficient acyclic models (middle) and causally insufficient cyclic models (right).

7 SIMULATIONS

We first test the accuracy of our method against several competing algorithms under the restricting assumption of acyclicity (and causal sufficiency), since no other algorithms are presently available for the most general model space that we can handle. In the tests for accuracy we draw 200 linear Gaussian models over 6 variables, where the edge coefficients are drawn uniformly from $\pm[0.2, 0.8]$. The directed edges were drawn randomly such that the average degree of each node was 2 for the causally sufficient models. For models with latent confounders the average degree of the nodes for directed edges was 1, and the covariance matrix of the disturbances corresponded to the passively observed covariance matrix of another similar causally sufficient linear Gaussian model. For all methods we use a correlation based t-test, or the corresponding Bayesian test. Although our methods straightforwardly allow for experimental data sets, we only used 500 samples of passively observed data here to adhere to the restrictions of the competing methods.

Different methods represent the uncertainty in their output differently: cPC and cFCI return parts of the graphs as unknown, while score-based methods, as well as the approaches in this paper, may be used to return several high scoring graphs. We compared the methods regarding only the single 'highest scoring' graph (or Markov equivalence class). To account for the fact that each method returns an equivalence class based on its model space assumptions, we evaluate the d-separation and d-connection relations of the learned result against those of the true data generating graph. This includes all possible d-connection/separation relations in the passively observed setting. Each of the methods takes a parameter (such as a p-value-threshold) that adjusts the sparsity of the output. To avoid effects of a specific parameter choice, we plot the accuracy of the different methods run with different parameters in the ROC space.

Figure 5 (left) shows the accuracy of the methods for acyclic and causally sufficient data generating models. The blue line shows the performance of the tests alone (classic t-tests almost exactly equal the Bayesian tests in the ROC space). The plain PC algorithm (implemented by Kalisch et al. (2012)) does not exceed the performance of these tests and cannot achieve high true positive rates for acceptable false positive rates. cPC returns an equivalence class of graphs without unknown parts in only $58/200$ cases for the optimal p-value threshold of 0.1 (only outputs from these runs are considered in the plot for cPC). Its performance on these 'easy instances' is quite good. The score-based approach achieves much better accuracy (BIC score; the MAP DAG is found using exact methods). All our approaches were run by restricting the model space to acyclic and causally sufficient graphs. The surprising finding here is that the constraint-based approach using 'log-weights', introduced in Section 4.3, seems to be able to roughly match the performance of the score-based method. This suggests that the constraint-optimization resolves many conflicts that arise from erroneous tests, and so it is an accurate approach for causal discovery. Also, the other suggested approaches, 'hard deps' (Section 4.1) and 'constant weights' (Section 4.2), seem to be able to perform quite well. Clearly, their results are not restricted to that of the test performance. The conflict resolution is able to correct many erroneous test results.

Figure 5 (middle) shows the accuracy of the methods in the ROC-space assuming acyclicity but not causal sufficiency. Our approaches (now only restricted by acyclicity), especially the 'log-weights', achieve higher true positive rates than the competing methods. Again cFCI does better than FCI but only returns a fully determined result for $61/200$ of the cases for the optimal p-value threshold of 0.1. For these 'easier instances' (again only results without unknowns are plotted for cFCI) its performance is quite good.[9] Figure 5 (right) shows the accuracy of the proposed methods in the

[9] The BCCD algorithm (Claassen and Heskes, 2012) and the (still unpublished) approach of Triantafillou and Tsamardinos (2014) would provide the most suitable comparison in this setting. We hope to perform the comparison when implementations of the methods are made available.

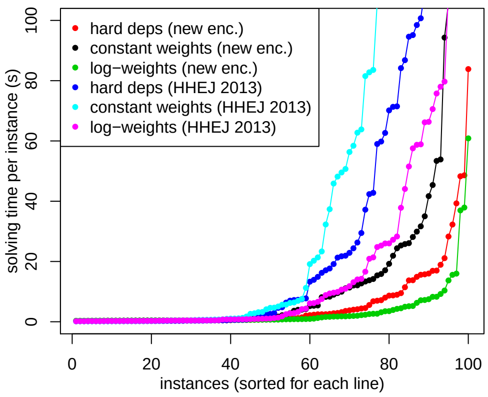


Figure 6: Solving times for (possibly) cyclic causally insufficient models (right).

most general model space allowing for cycles and latent confounders. There currently do not exist other methods that apply to such a general model space.

Figure 6 shows the solving times of `Clingo` on a 2.4-GHz Intel Core i5 processor for 100 instances of constraints obtained from 500 samples of passively observed data generated by different possibly cyclic and causally insufficient models over six variables. The solving times are sorted for each algorithm individually. The plot shows nicely the rather large variance of the solving times. Easy problems (left on the plot) are solved almost instantly, while harder problems may take considerably longer. This is a general feature of exact algorithms solving very complex problems: in the worst case the NP-complexity of the problem kicks in, but still a large number of instances are relatively easy to solve. In the figure we also compare the present encoding against a straightforward ASP-implementation of the encoding of Hyttinen et al. (2013). For all weighting schemes, and especially for the harder instances, the encoding presented in this paper seems to allow for much faster solving. Different weighting schemes also clearly affect the solving time. 'log-weights' and 'hard deps' are considerably faster than 'constant weights'. For graphs with seven observed variable the solving times take up to half an hour (see supplement), for graphs with eight variables many instances take several hours to solve.

Finally we analyzed what actually happens in the conflict resolution. We generated data from 100 random parameterizations of the graph in Figure 1, and ran the inference with log-weights for different sample sizes (Figure 7). The redness of the background color denotes how many times the specific independence tests most prone to error (listed on the right axis) produced an incorrect result. The blue line counts the number of incorrect tests obtained from the data, which serve as input to our method. The black line shows the number of (in)dependence relations that were incorrect in the output graph of our method. The tests produce errors throughout different sample sizes, but we are able to offer an output with significantly fewer errors.

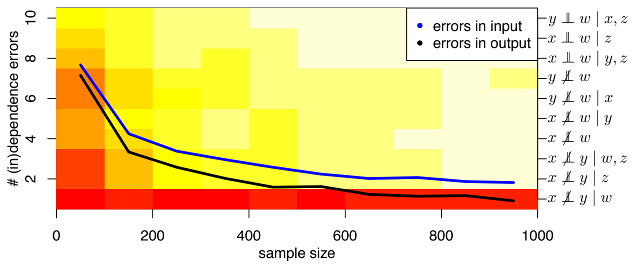


Figure 7: Conflict resolution for the graph in Figure 1.

8 CONCLUSION

We presented a method for causal discovery that works in a completely general search space that includes models with feedback *and* latent confounders. It accepts as input possibly inconsistent (in)dependence constraints obtained from overlapping experimental or observational data sets. It returns an *exact* output, in the sense that it finds the graph that minimizes a well-defined objective function, which characterizes how conflicted constraints should be resolved. We have considered a variety of theoretical motivations for different conflict resolution schemes, and tested them successfully against several extant algorithms.[10] Although not shown explicitly in this article, the direct encoding of the d-connection properties ensures that in the infinite sample limit, our algorithm retains the completeness properties of Hyttinen et al. (2013).

The scalability of the present procedure is still quite limited, but it is similar to that of the exact graph structure discovery methods exploiting ASP of Corander et al. (2013), who search for (undirected) Markov networks. We have emphasized the exactness of our method, which provides theoretical guarantees and a very high accuracy. All other methods we are aware of that consider similar search spaces take a greedy approach in one way or another. Our results suggest that the resulting scalability may come at the price of accuracy for realistic sample sizes on anything but very sparse causal structures. We hope to use our approach as basis to explore the trade-off between scalability and accuracy in the future.

Acknowledgements

A.H. was supported by the Finnish Foundation for Technology Promotion (TES), and M. J. by the Academy of Finland under grants 251170 (COIN Finnish Centre of Excellence in Computational Inference Research) and 276412.

[10] The supplementary material and the code package reproducing the simulation results is available from the authors' websites.

References

Biere, A., Heule, M., van Maaren, H., and Walsh, T., editors (2009). *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*. IOS Press.

Chickering, D. M. (2002). Optimal structure identification with greedy search. *Journal of Machine Learning Research*, 3:507–554.

Claassen, T. and Heskes, T. (2012). A bayesian approach to constraint based causal inference. In *Proceedings of UAI*, pages 207–216. AUAI Press.

Cooper, G. F. and Herskovits, E. (1992). A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347.

Corander, J., Janhunen, T., Rintanen, J., Nyman, H. J., and Pensar, J. (2013). Learning chordal Markov networks by constraint satisfaction. In *Proceedings of NIPS*, pages 1349–1357.

Gebser, M., Kaufmann, B., Kaminski, R., Ostrowski, M., Schaub, T., and Schneider, M. T. (2011). Potassco: The Potsdam answer set solving collection. *AI Communications*, 24(2):107–124.

Geiger, D. and Heckerman, D. (1994). Learning Gaussian networks. Technical Report MSR-TR-94-10, Microsoft Research.

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

Hyttinen, A., Hoyer, P. O., Eberhardt, F., and Järvisalo, M. (2013). Discovering cyclic causal models with latent variables: A general SAT-based procedure. In *Proceedings of UAI*, pages 301–310. AUAI Press.

Kalisch, M., Mächler, M., Colombo, D., Maathuis, M. H., and Bühlmann, P. (2012). Causal inference using graphical models with the R package pcalg. *Journal of Statistical Software*, 47(11):1–26.

Koster, J. T. A. (2002). Marginalizing and conditioning in graphical models. *Bernoulli*, 8(6):817–840.

Li, J. and Wang, Z. J. (2009). Controlling the false discovery rate of the association/causality structure learned with the PC algorithm. *Journal of Machine Learning Research*, 10:475–514.

Margaritis, D. and Bromberg, F. (2009). Efficient Markov network discovery using particle filters. *Computational Intelligence*, 25(4):367–394.

Neal, R. (2000). On deducing conditional independence from d-separation in causal graphs with feedback. *Journal of Artificial Intelligence Research*, 12:87–91.

Niemelä, I. (1999). Logic programs with stable model semantics as a constraint programming paradigm. *Annals of Mathematics and Artificial Intelligence*, 25(3-4):241–273.

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

Pearl, J. and Dechter, R. (1996). Identifying independencies in causal graphs with feedback. In *Proceedings of UAI*, pages 420–426. Morgan Kaufmann.

Ramsey, J., Zhang, J., and Spirtes, P. (2006). Adjacency-faithfulness and conservative causal inference. In *Proceedings of UAI*, pages 401–408. AUAI Press.

Richardson, T. and Spirtes, P. (2002). Ancestral graph Markov models. *Annals of Statistics*, 30(4).

Schulte, O., Luo, W., and Greiner, R. (2010). Mind change optimal learning of Bayes net structure from dependency and independency data. *Information and Computation*, 208(1):63 – 82.

Simons, P., Niemelä, I., and Soininen, T. (2002). Extending and implementing the stable model semantics. *Artificial Intelligence*, 138(1-2):181–234.

Spirtes, P. (1995). Directed cyclic graphical representation of feedback models. In *Proceedings of UAI*, pages 491–498. Morgan Kaufmann.

Spirtes, P., Glymour, C., and Scheines, R. (1993). *Causation, Prediction, and Search*. Springer-Verlag.

Studený, M. (1998). Bayesian networks from the point of view of chain graphs. In *Proceedings of UAI*, pages 496–503. Morgan Kaufmann.

Triantafillou, S. and Tsamardinos, I. (2014). Constraint-based causal discovery from multiple interventions over overlapping variable sets. arXiv:1403.2150 (unpublished).

Triantafillou, S., Tsamardinos, I., and Tollis, I. G. (2010). Learning causal structure from overlapping variable sets. In *Proceedings of AISTATS*, pages 860–867. JMLR.

Tsamardinos, I., Triantafillou, S., and Lagani, V. (2012). Towards integrative causal analysis of heterogeneous data sets and studies. *Journal of Machine Learning Research*, 13:1097–1157.

Constraint-based Causal Discovery: Conflict Resolution with Answer Set Programming
Paper Supplement

Antti Hyttinen and **Frederick Eberhardt**
California Institute of Technology
Pasadena, CA, USA

Matti Järvisalo
HIIT & Department of Computer Science
University of Helsinki, Finland

A EXTENSION TO MULTIPLE OVERLAPPING DATA SETS

A set of data sets is said to be overlapping if the sets of measured variables share some, but not all the same variables. Tillman et al. (2009), Tillman and Spirtes (2011) and Triantafillou et al. (2010) have explored causal discovery algorithms in this setting under the assumption that the data is passive observational in each data set and that the underlying causal model is acyclic. In particular, the approach in Tillman and Spirtes (2011) uses R.A. Fisher's technique of pooling p-values to integrate test results from multiple tests on the same set of variables. Hyttinen et al. (2013) extend these approaches to include experimental data sets where a subset of the variables has been subject to an intervention. They also allow for feedback cycles in the generating models. However, their algorithm could not handle conflicting constraints that arise from statistical data.

Our present procedure extends naturally to the completely general setting of multiple overlapping experimental or observational data sets, and handles conflicted constraints. In this setting the set of constraints $\mathbf{K}$ that enters into the constraint optimization is expanded to (in the general case) include all possible constraints that can be obtained from each of the individual data sets. Consider a constraint $k \in \mathbf{K}$ obtained from an (experimental) data set $\mathbf{D}$ where the (possibly empty) set of variables $\mathbf{J} \subset \mathbf{V}$ was subject to an intervention. The weight $w(k)$ now enters into the sum of weights for a graph G if the graph $G_\mathbf{J} \not\models k$, where $G_\mathbf{J}$ is the same as G except that all edges incident on any variable in $\mathbf{J}$ are removed, i.e. $G_\mathbf{J}$ is the manipulated version of G. The basic idea is that one simply has to keep track of the experimental setting that the constraint was obtained from, and use that as the basis for the minimization. Consequently, the graph that our method returns is optimal (in the sense of the problem statement) across the available data sets.

Variables that are measured in one of the overlapping data sets, but not in another, are treated as marginalized variables in the d-connection graphs for the latter data set. Below, in Appendices C and F, we specify how to add an intervention operation to the ASP-encoding while ensuring that the d-connection properties are appropriately preserved for the d-connection graphs that specify the test results for experimental data sets.

B PROBABILISTIC INTERPRETATION OF THE LOG-WEIGHTS

Often the log-score can be interpreted in a probabilistic way. Here we investigate the log-scores and their relation to the problem formulation in Section 2 with this aim. Given a set of constraints $\mathbf{K}$, we set out to find G^* such that

$$\begin{aligned} G^* &\in \underset{G\in\mathcal{G}}{\arg\min} \sum_{k\in\mathbf{K}\,:\,G\not\models k} w(k) \\ w(k) &= \log P(k \mid \mathbf{D}) - \log[1 - P(k \mid \mathbf{D})] \end{aligned}$$

Now, let vector $\mathbf{q}$ be a binary vector indicating whether a constraint is satisfied by a graph G with $q_i = 1$ if the i:th constraint is satisfied and $q_i = 0$ if it is not, i.e. vector $\mathbf{q}$ describes the equivalence class of graphs with respect to the constraints $\mathbf{K}$. Let vector $\mathbf{p}$ be the probability estimates of constraints calculated from the data, i.e. $p_i = P(k_i \mid \mathbf{D})$. The optimization problem can now be re-described as

$$\underset{\mathbf{q}}{\arg\min} \sum_{i\,:\,q_i=0} \log p_i - \log(1-p_i),$$

where we still (implicitly) require that vector $\mathbf{q}$ has to correspond to some graph in the considered model space that satisfies causal Markov and faithfulness. This formulation can now be converted into an equivalent maximization problem

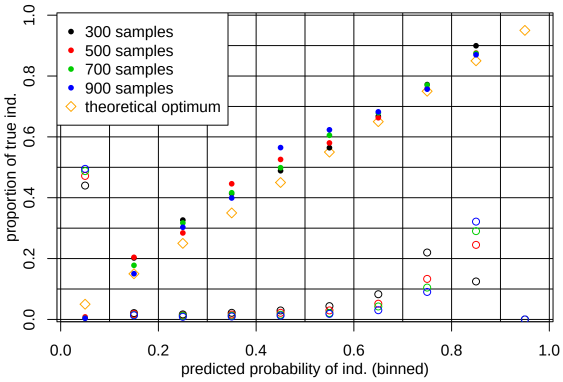


Figure 1: Calibration of probabilities. See text for details.

(adding constants does not change the optimal answer):

$$\begin{aligned}
& \arg\min_{\mathbf{q}} \sum_{i\,:\,q_i=0} \log p_i - \log(1-p_i) \\
\Leftrightarrow\quad & \arg\max_{\mathbf{q}} \sum_{i\,:\,q_i=0} -\log p_i + \log(1-p_i) \\
\Leftrightarrow\quad & \arg\max_{\mathbf{q}} \sum_{i\,:\,q_i=0} [-\log p_i + \log(1-p_i)] + \sum_{i} \log p_i \\
\Leftrightarrow\quad & \arg\max_{\mathbf{q}} \sum_{i\,:\,q_i=1} \log p_i + \sum_{i\,:\,q_i=0} \log(1-p_i) \\
\Leftrightarrow\quad & \arg\max_{\mathbf{q}} \prod_{i} p_i^{q_i}(1-p_i)^{1-q_i}
\end{aligned}$$

Thus, we are finding the vector $\mathbf{q}$ (representing an equivalence class of graphs) that maximizes the probability distribution $P(\mathbf{q} \mid \mathbf{p})$, with the approximation that the elements q_i are distributed mutually independently, with the corresponding probabilities in the vector $\mathbf{p}$. In fact, some (in)dependence relations imply others, especially under the assumption of Markov and faithfulness. But independently run independence tests are not able to share this information with each other. Instead, this is exactly the task of conflict resolution: One has to find a $\mathbf{q}$ that corresponds to at least on graph in the model space (and respects all the implications between (in)dependences).

Suppose we further assume that the vector $\mathbf{p}$ exhausts all information on $\mathbf{q}$ available in $\mathbf{D}$, so that it forms a 'sufficient statistic', that is (i) $\mathbf{D} \perp\!\!\!\perp \mathbf{q} \mid \mathbf{p}$ and (ii) $\mathbf{p}$ is a deterministic function of $\mathbf{D}$. In that case we have

$$P(\mathbf{q}|\mathbf{D}) \quad = \quad P(\mathbf{q} \mid \mathbf{p}).$$

So in our minimization we are essentially – modulo the assumptions and approximations just specified – finding the equivalence class of graphs that maximizes the posterior probability.

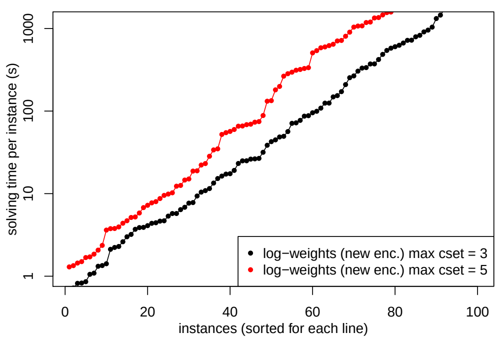


Figure 2: Solving times for the log-weighting scheme.

C INTERVENTION

Given a d-connection graph $H = (\mathbf{V}, \mathbf{E})_{\mathbf{C}}$, the *intervention* operation $i(H, t)$ for variable $t \in \mathbf{V}$ results in d-connection graph $H' = (\mathbf{V}, \mathbf{E}')_{\mathbf{C}}$, where $\mathbf{E}'$ is related to $\mathbf{E}$ by (i) including in $\mathbf{E}'$ any edges in $\mathbf{E}$ not involving t; (ii) including in $\mathbf{E}'$ any edge $x_a\!-\!t$ if there is an edge $x_a\!-\!t$ in $\mathbf{E}$; and (iii) not permitting any other edges in $\mathbf{E}'$. Thus, as is standard in the representation of interventions, edges incident on the intervened variable are omitted (they are not copied to $\mathbf{E}'$). Note that the intervention operation cannot be applied to a variable previously conditioned on or marginalized, however these operation can still be applied to a variable after intervention. In the case of conditioning, the purpose of allowing an intervention before conditioning is obvious: One wants to be able to express (in)dependence constraints that condition on an intervened variable. In the case of marginalization the possibility of intervening is necessary in this representation of d-connection graphs in order to reach (in)dependence constraints between non-intervened variables in a data set where some variables have been subject to intervention. Thus, with regard to the "encoding DAG" in Figure 3, there are still multiple options of where the intervention operations could be included, since the order of intervention before marginalization and conditioning only applies to each variable individually. A variety of considerations may make one arrangement of operations in the encoding DAG more useful than another: One may prefer to integrate constraints within one data set first, or one may want to integrate similar constraints across data sets first.

D PROOF OF THEOREM 1

Proof: Assume there is a path p of type $x_a \cdots{}_b y$ that is d-connecting given $\mathbf{C}'' \supseteq \mathbf{C}'$ in H. Since p is d-connecting, every collider on p is in $\mathbf{C}''$ and every non-collider is not in $\mathbf{C}''$. In the conditioning operation all edges not adjacent to w are preserved. So the only parts of p that may be broken in H' are due to edges connected to the newly conditioned variable w. Since $w \in \mathbf{C}''$ and the path p is d-connecting given $\mathbf{C}''$, it follows that w must be a collider on p, which means it can only appear in components of the type $s_a\!\rightarrow w[\leftrightarrow w]^*\!\leftarrow_b t$ on the path p, where $s \neq w, t \neq w$ and the part in the brackets may appear zero or more times. Let p' be the path between x and y where all such components are replaced by $s_a -_b t$. This replacement edge is in H' due to rule (ii) in the definition of the conditioning operation. Furthermore, since there is no other way to break a path, p' as a whole is present in H', and p' is still d-connecting given $\mathbf{C}''$. The path p' will have the same end points as path p, and thus a path of type $x_a \cdots_b y$ is present in H' that is d-connecting given $\mathbf{C}''$.

Assume there is a path p' of type $x_a \cdots_b y$ that is d-connecting given $\mathbf{C}'' \supseteq \mathbf{C}'$ in H'. As all other edges in H' are just copies of edges in H, for the path p' not to exist in H there must be edges $s_a -_b t$ on the path p' that are added by rule (ii) in the definition of the conditioning operation. But then for each such edge there has to be a triple $s_a\!\rightarrow w \leftarrow_b t$ in H, as otherwise rule (ii) would not have added $s_a -_b t$ to H'. Such a triple cannot be blocked since $w \in \mathbf{C}''$. Replacing all edges not present in H by such triples similarly results in a path p in H that is d-connecting given $\mathbf{C}''$ and still of the type $x_a \cdots_b y$. $\square$

The proof for the marginalization and intervention operations follow the exact same idea.

E FURTHER SIMULATIONS

Figure 1 shows the probability calibration plot for the estimates of the probability of independence determined by the model comparison described in Section 4.3. The estimates are based on 300 causally insufficient and possibly cyclic models under passive observation. The predictions were divided into 10 bins with equal width, denoted by the black lines in the plot. We then calculated how many times the true result was in fact an independence. The filled circles mark these true proportions against the predicted probability for different sample sizes. The probabilities seem to be roughly calibrated for the different sample sizes when the prior is set to $\alpha = 0.5$ (and the equivalent sample size prior for the local-score is set to 20). The unfilled circles show the proportions of predictions in each bin. Note that for these sample sizes, the test does not yet predict a probability over 0.9 for an independence. This seems natural since it is very hard to be sure of an independence, it might just be a very weak dependence. Otherwise the high and low probabilities are predicted more often when the sample size grows, while the probabilities in the middle bins are predicted somewhat more often for lower sample sizes.

Figure 2 further explores the scalability of the method using log-weights for 7-variable graphs. The running times are in log-scale, and again the instances are sorted according to their solving times. Limiting the maximimum conditioning set size to 3, which already allows most inferences to be made, cuts down the number of constraints and allows for faster solving.

F FULL ASP ENCODING

Finally, we give the full ASP-encoding with the *intervention*, *conditioning* and *marginalization* operations, presented in Figure 3. In the full encoding, an additional set $\mathbf{J}$ is introduced that represents the intervention set. In analogy with $cond(\mathbf{V}, \mathbf{C}, \mathbf{J}, z)$ and $marg(\mathbf{V}, \mathbf{C}, \mathbf{J}, z)$, the input predicate $intervene(\mathbf{V}, \mathbf{C}, \mathbf{J}, z)$ enables intervening on a variable z in a d-connection graph that has exactly the variables $\mathbf{V}$ and conditioning set $\mathbf{C}$.

References

Hyttinen, A., Hoyer, P. O., Eberhardt, F., and Järvisalo, M. (2013). Discovering cyclic causal models with latent variables: A general SAT-based procedure. In *Proceedings of UAI*, pages 301–310. AUAI Press.

Tillman, R. E., Danks, D., and Glymour, C. (2009). Integrating locally learned causal structures with overlapping variables. In *Proceedings of NIPS 2008*, pages 1665–1672.

Tillman, R. E. and Spirtes, P. (2011). Learning equivalence classes of acyclic models with latent and selection variables from multiple datasets with overlapping variables. In *Proceedings of AISTATS*. JMLR.

Triantafillou, S., Tsamardinos, I., and Tollis, I. G. (2010). Learning causal structure from overlapping variable sets. In *Proceedings of AISTATS*, pages 860–867. JMLR.

Conditioning on variable $z \in \mathbf{V}, \forall x, y \in \mathbf{V} \setminus z$:

$$\begin{aligned}
\mathit{th}(x,y,\mathbf{V}\setminus z,\mathbf{C}\cup z,\mathbf{J}) &\text{ :- } \mathit{th}(x,y,\mathbf{V},\mathbf{C},\mathbf{J}), \mathit{cond}(\mathbf{V},\mathbf{C},\mathbf{J},z).\\
\mathit{th}(x,y,\mathbf{V}\setminus z,\mathbf{C}\cup z,\mathbf{J}) &\text{ :- } \mathit{th}(x,z,\mathbf{V},\mathbf{C},\mathbf{J}), \mathit{hh}(z,y,\mathbf{V},\mathbf{C},\mathbf{J}),\\
&\qquad \mathit{cond}(\mathbf{V},\mathbf{C},\mathbf{J},z).
\end{aligned}$$

$$\begin{aligned}
\mathit{hh}(x,y,\mathbf{V}\setminus z,\mathbf{C}\cup z,\mathbf{J}) &\text{ :- } \mathit{hh}(x,y,\mathbf{V},\mathbf{C},\mathbf{J}), \mathit{cond}(\mathbf{V},\mathbf{C},\mathbf{J},z).\\
\mathit{hh}(x,y,\mathbf{V}\setminus z,\mathbf{C}\cup z,\mathbf{J}) &\text{ :- } \mathit{hh}(x,z,\mathbf{V},\mathbf{C},\mathbf{J}), \mathit{hh}(z,y,\mathbf{V},\mathbf{C},\mathbf{J}),\\
&\qquad \mathit{cond}(\mathbf{V},\mathbf{C},\mathbf{J},z).
\end{aligned}$$

$$\begin{aligned}
\mathit{tt}(x,y,\mathbf{V}\setminus z,\mathbf{C}\cup z,\mathbf{J}) &\text{ :- } \mathit{tt}(x,y,\mathbf{V},\mathbf{C},\mathbf{J}), \mathit{cond}(\mathbf{V},\mathbf{C},\mathbf{J},z).\\
\mathit{tt}(x,y,\mathbf{V}\setminus z,\mathbf{C}\cup z,\mathbf{J}) &\text{ :- } \mathit{th}(x,z,\mathbf{V},\mathbf{C},\mathbf{J}), \mathit{th}(y,z,\mathbf{V},\mathbf{C},\mathbf{J}),\\
&\qquad \mathit{cond}(\mathbf{V},\mathbf{C},\mathbf{J},z).
\end{aligned}$$

Marginalizing a variable $z \in \mathbf{V}, \forall x, y \in \mathbf{V} \setminus z$:

$$\begin{aligned}
\mathit{th}(x,y,\mathbf{V}\setminus z,\mathbf{C},\mathbf{J}) &\text{ :- } \mathit{th}(x,y,\mathbf{V},\mathbf{C},\mathbf{J}), \mathit{marg}(\mathbf{V},\mathbf{C},\mathbf{J},z).\\
\mathit{th}(x,y,\mathbf{V}\setminus z,\mathbf{C},\mathbf{J}) &\text{ :- } \mathit{tt}(x,z,\mathbf{V},\mathbf{C},\mathbf{J}), \mathit{th}(z,y,\mathbf{V},\mathbf{C},\mathbf{J}),\\
&\qquad \mathit{marg}(\mathbf{V},\mathbf{C},\mathbf{J},z).\\
\mathit{th}(x,y,\mathbf{V}\setminus z,\mathbf{C},\mathbf{J}) &\text{ :- } \mathit{th}(x,z,\mathbf{V},\mathbf{C},\mathbf{J}), \mathit{th}(z,y,\mathbf{V},\mathbf{C},\mathbf{J}),\\
&\qquad \mathit{marg}(\mathbf{V},\mathbf{C},\mathbf{J},z).\\
\mathit{th}(x,y,\mathbf{V}\setminus z,\mathbf{C},\mathbf{J}) &\text{ :- } \mathit{tt}(x,z,\mathbf{V},\mathbf{C},\mathbf{J}), \mathit{hh}(z,y,\mathbf{V},\mathbf{C},\mathbf{J}),\\
&\qquad \mathit{marg}(\mathbf{V},\mathbf{C},\mathbf{J},z).\\
\mathit{th}(x,y,\mathbf{V}\setminus z,\mathbf{C},\mathbf{J}) &\text{ :- } \mathit{th}(x,z,\mathbf{V},\mathbf{C},\mathbf{J}), \mathit{tt}(z,z,\mathbf{V},\mathbf{C},\mathbf{J}),\\
&\qquad \mathit{hh}(z,y,\mathbf{V},\mathbf{C},\mathbf{J}), \mathit{marg}(\mathbf{V},\mathbf{C},\mathbf{J},z).
\end{aligned}$$

$$\begin{aligned}
\mathit{hh}(x,y,\mathbf{V}\setminus z,\mathbf{C},\mathbf{J}) &\text{ :- } \mathit{hh}(x,y,\mathbf{V},\mathbf{C},\mathbf{J}), \mathit{marg}(\mathbf{V},\mathbf{C},\mathbf{J},z).\\
\mathit{hh}(x,y,\mathbf{V}\setminus z,\mathbf{C},\mathbf{J}) &\text{ :- } \mathit{th}(z,x,\mathbf{V},\mathbf{C},\mathbf{J}), \mathit{th}(z,y,\mathbf{V},\mathbf{C},\mathbf{J}),\\
&\qquad \mathit{marg}(\mathbf{V},\mathbf{C},\mathbf{J},z).\\
\mathit{hh}(x,y,\mathbf{V}\setminus z,\mathbf{C},\mathbf{J}) &\text{ :- } \mathit{hh}(x,z,\mathbf{V},\mathbf{C},\mathbf{J}), \mathit{th}(z,y,\mathbf{V},\mathbf{C},\mathbf{J}),\\
&\qquad \mathit{marg}(\mathbf{V},\mathbf{C},\mathbf{J},z).\\
\mathit{hh}(x,y,\mathbf{V}\setminus z,\mathbf{C},\mathbf{J}) &\text{ :- } \mathit{th}(z,x,\mathbf{V},\mathbf{C},\mathbf{J}), \mathit{hh}(z,y,\mathbf{V},\mathbf{C},\mathbf{J}),\\
&\qquad \mathit{marg}(\mathbf{V},\mathbf{C},\mathbf{J},z).\\
\mathit{hh}(x,y,\mathbf{V}\setminus z,\mathbf{C},\mathbf{J}) &\text{ :- } \mathit{hh}(x,z,\mathbf{V},\mathbf{C},\mathbf{J}), \mathit{tt}(z,z,\mathbf{V},\mathbf{C},\mathbf{J}),\\
&\qquad \mathit{hh}(z,y,\mathbf{V},\mathbf{C},\mathbf{J}), \mathit{marg}(\mathbf{V},\mathbf{C},\mathbf{J},z).
\end{aligned}$$

$$\begin{aligned}
\mathit{tt}(x,y,\mathbf{V}\setminus z,\mathbf{C},\mathbf{J}) &\text{ :- } \mathit{tt}(x,y,\mathbf{V},\mathbf{C},\mathbf{J}), \mathit{marg}(\mathbf{V},\mathbf{C},\mathbf{J},z).\\
\mathit{tt}(x,y,\mathbf{V}\setminus z,\mathbf{C},\mathbf{J}) &\text{ :- } \mathit{tt}(x,z,\mathbf{V},\mathbf{C},\mathbf{J}), \mathit{tt}(z,y,\mathbf{V},\mathbf{C},\mathbf{J}),\\
&\qquad \mathit{marg}(\mathbf{V},\mathbf{C},\mathbf{J},z).\\
\mathit{tt}(x,y,\mathbf{V}\setminus z,\mathbf{C},\mathbf{J}) &\text{ :- } \mathit{th}(x,z,\mathbf{V},\mathbf{C},\mathbf{J}), \mathit{tt}(z,y,\mathbf{V},\mathbf{C},\mathbf{J}),\\
&\qquad \mathit{marg}(\mathbf{V},\mathbf{C},\mathbf{J},z).\\
\mathit{tt}(x,y,\mathbf{V}\setminus z,\mathbf{C},\mathbf{J}) &\text{ :- } \mathit{tt}(x,z,\mathbf{V},\mathbf{C},\mathbf{J}), \mathit{th}(y,z,\mathbf{V},\mathbf{C},\mathbf{J}),\\
&\qquad \mathit{marg}(\mathbf{V},\mathbf{C},\mathbf{J},z).\\
\mathit{tt}(x,y,\mathbf{V}\setminus z,\mathbf{C},\mathbf{J}) &\text{ :- } \mathit{th}(x,z,\mathbf{V},\mathbf{C},\mathbf{J}), \mathit{tt}(z,z,\mathbf{V},\mathbf{C},\mathbf{J}),\\
&\qquad \mathit{th}(y,z,\mathbf{V},\mathbf{C},\mathbf{J}), \mathit{marg}(\mathbf{V},\mathbf{C},\mathbf{J},z).
\end{aligned}$$

Intervening on a variable $z \in \mathbf{V}, \forall x, y \in \mathbf{V}$:

$$\begin{aligned}
\mathit{th}(x,y,\mathbf{V},\mathbf{C},\mathbf{J}\cup z) &\text{ :- } \mathit{th}(x,y,\mathbf{V},\mathbf{C},\mathbf{J}), z \neq y,\\
&\qquad \mathit{intervene}(\mathbf{V},\mathbf{C},\mathbf{J},z).\\
\mathit{hh}(x,y,\mathbf{V},\mathbf{C},\mathbf{J}\cup z) &\text{ :- } \mathit{hh}(x,y,\mathbf{V},\mathbf{C},\mathbf{J}), x \neq z, y \neq z,\\
&\qquad \mathit{intervene}(\mathbf{V},\mathbf{C},\mathbf{J},z).\\
\mathit{tt}(x,y,\mathbf{V},\mathbf{C},\mathbf{J}\cup z) &\text{ :- } \mathit{tt}(x,y,\mathbf{V},\mathbf{C},\mathbf{J}), \mathit{intervene}(\mathbf{V},\mathbf{C},\mathbf{J},z).
\end{aligned}$$

Inferring failures to satisfy (in)dependencies $\forall x \forall y > x, \forall \mathbf{C}, \mathbf{V} = \{x, y\}$:

$$\begin{aligned}
\mathit{fail}(x,y,\mathbf{V},\mathbf{C},\mathbf{J},W) &\text{ :- } \mathit{tt}(x,y,\mathbf{V},\mathbf{C},\mathbf{J}), \mathit{indep}(x,y,\mathbf{V},\mathbf{C},\mathbf{J},W).\\
\mathit{fail}(x,y,\mathbf{V},\mathbf{C},\mathbf{J},W) &\text{ :- } \mathit{th}(x,y,\mathbf{V},\mathbf{C},\mathbf{J}), \mathit{indep}(x,y,\mathbf{V},\mathbf{C},\mathbf{J},W).\\
\mathit{fail}(x,y,\mathbf{V},\mathbf{C},\mathbf{J},W) &\text{ :- } \mathit{th}(y,x,\mathbf{V},\mathbf{C},\mathbf{J}), \mathit{indep}(x,y,\mathbf{V},\mathbf{C},\mathbf{J},W).\\
\mathit{fail}(x,y,\mathbf{V},\mathbf{C},\mathbf{J},W) &\text{ :- } \mathit{hh}(x,y,\mathbf{V},\mathbf{C},\mathbf{J}), \mathit{indep}(x,y,\mathbf{V},\mathbf{C},\mathbf{J},W).\\
\mathit{fail}(x,y,\mathbf{V},\mathbf{C},\mathbf{J},W) &\text{ :- } \text{not } \mathit{th}(x,y,\mathbf{V},\mathbf{C},\mathbf{J}), \text{not } \mathit{th}(y,x,\mathbf{V},\mathbf{C},\mathbf{J}),\\
&\qquad \text{not } \mathit{hh}(x,y,\mathbf{V},\mathbf{C},\mathbf{J}), \text{not } \mathit{tt}(x,y,\mathbf{V},\mathbf{C},\mathbf{J}),\\
&\qquad \mathit{dep}(x,y,\mathbf{V},\mathbf{C},\mathbf{J},W).
\end{aligned}$$

Weak constraints $\forall x \forall y > x, \forall \mathbf{C}, \mathbf{J}, \mathbf{V} = \{x, y\}$:

$$:\sim \mathit{fail}(x,y,\mathbf{V},\mathbf{C},\mathbf{J},W). \, [W]$$

Figure 3: The ASP encoding, including the encoding of the intervention operation.