Probabilistic Deduction with Conditional Constraints over Basic Events

Thomas Lukasiewicz
Institut für Informatik, Universität Gießen
Arndtstr. 2, D-35392 Gießen, Germany
E-mail: lukasiewicz@informatik.uni-giessen.de

Abstract

We show that probabilistic deduction with conditional constraints over basic events is NP-hard. We then focus on the special case of probabilistic deduction in conditional constraint trees. We elaborate very efficient and globally complete techniques for probabilistic deduction. More precisely, for exact conditional constraint trees, we present a local approach to globally complete probabilistic deduction, which runs in linear time in the size of the conditional constraint trees. For general conditional constraint trees, we show that globally complete probabilistic deduction can be done by solving global nonlinear programs. We elaborate how these nonlinear programs can be transformed into equivalent linear programs, which are solvable in polynomial time in the size of the conditional constraint trees.

1 INTRODUCTION

Representing and reasoning with uncertain knowledge has gained growing importance in the recent decades. The literature contains many different formalisms and methodologies for tackling uncertainty. Most of them are directly or indirectly based on probability theory.

In this paper, we focus on probabilistic deduction with conditional constraints (that is, interval restrictions for conditional probabilities). The considered probabilistic deduction problems consist of a probabilistic knowledge-base and a probabilistic query. We give a classical example. As a probabilistic knowledge-base, we may take the probabilistic knowledge that all ostriches are birds, that ostriches do not fly, that at least 95 per cent of all birds fly, and that not more than 10 per cent of all birds are ostriches. As a probabilistic query, we may wonder about the entailed greatest lower bound and the entailed least upper bound for the rate of all birds that are ostriches. The solution to this probabilistic deduction problem is 0 per cent for the entailed greatest lower bound, and 5 per cent for the entailed least upper bound.

There is a long controversy in the probabilistic community about whether to solve this kind of probabilistic deduction problems in a global approach by linear programming or in a local approach by the iterative application of inference rules. The global approach by linear programming (see, for example, [23], [13], [10], [17], [16], [3], [22], and [20]) can be performed within rich probabilistic languages capable of representing many facets of probabilistic knowledge (see especially [10]). Crucially, probabilistic deduction by linear programming is globally complete, that is, it really produces the requested tightest bounds entailed by the whole probabilistic knowledge-base. However, it generally runs in exponential time in the size of the probabilistic deduction problems. Furthermore, it cannot provide any explanatory informations on how the deduced results are obtained.

Mainly to overcome these deficiencies, researchers started to work on local techniques based on inference rules. The local approach (see, for example, [7], [9], [2], [8], [28], [11], [15], and [18]) is generally performed within more restricted probabilistic languages. The iterative application of inference rules is very rarely and only within very restricted languages globally complete (see [11] for an example of globally complete local probabilistic deduction in a very restricted framework). Whenever the languages become more expressive, the global completeness of local probabilistic deduction cannot be guaranteed anymore. Local techniques are generally expected to be more efficient than global ones. Moreover, they can elucidate the deduction process by the sequence of applied inference rules.

Probabilistic deduction with conditional constraints over *propositional events* is immediately NP-hard, since it is a generalization of the satisfiability problem for probabilistic logic [23], which is known to be NP-complete from [13]. As a first crucial contribution of this paper, we even show that probabilistic deduction with conditional constraints over *basic events* (like in the introductory example) is also NP-hard. It is surprising that this quite restricted class of probabilistic deduction problems is still computationally so difficult.

Hence, it is unlikely that there is an algorithm that efficiently solves all probabilistic deduction problems with conditional constraints over basic events. However, we can still hope that there are efficient special-case, average-case, or approximation algorithms.

In this paper, we focus on elaborating an efficient special-case algorithm. We concentrate on probabilistic deduction in conditional constraint trees. It is an interesting subclass of all probabilistic deduction problems with conditional constraints over basic events. Conditional constraint trees are undirected trees with basic events as nodes and with bidirectional conditional constraints over basic events as edges between the nodes. Like Bayesian networks, conditional constraint trees represent a well-structured probabilistic knowledge-base. However, differently from Bayesian networks, they do not require any knowledge about probabilistic independence.

For exact conditional constraint trees, we present functions for deducing tightest lower and upper bounds in linear time in the size of the conditional constraint trees. For general conditional constraint trees, we show that greatest lower bounds can be deduced in the same way, in linear time in the size of the conditional constraint trees. However, computing least upper bounds turns out to be computationally more difficult. It can be done by solving special nonlinear programs. We show how these nonlinear programs can be transformed into equivalent linear programs. Crucially, these linear programs have a number of variables and of linear inequalities being linear and polynomial, respectively, in the size of the conditional constraint trees. Thus, our way of deducing least upper bounds still runs in polynomial time in the size of the conditional constraint trees, since linear programming runs in polynomial time in the size of the linear programs.

Another contribution of this paper is related to the controversy about whether to perform probabilistic deduction with conditional constraints by linear programming or by the iterative application of inference rules. On the one hand, we provide a very efficient technique for globally complete probabilistic deduction in exact conditional constraint trees. However, on the other hand, we also show that extending this local technique to general conditional constraint trees already results in solving global linear programs.

Previous work on similar deduction problems is rare in the literature, the few existing does not provide any or just very few results towards global completeness. In [2], the rule of quantified syllogism and the generalized Bayes' rule are applied to sets of bidirectional conditional constraints over basic events. This deduction technique is generally not globally complete. In [27], trees of bidirectional conditional constraints over basic events are examined within the framework of deductive database systems. A system of inference rules is presented, which is globally complete only in deducing lower bounds.

The rest of this paper is organized as follows. In Section 2, we formulate the probabilistic deduction problems considered in this work. Section 3 focuses on complete probabilistic deduction in exact and general conditional constraint trees. In Section 4, we summarize the main results of this work.

This paper is a revised extract of own work from [19], which we extended by new results on the computational complexity of probabilistic deduction with conditional constraints over basic events. Moreover, we improved significantly the probabilistic deduction technique for general conditional constraint trees.

2 FORMULATING THE DEDUCTION PROBLEMS

In this section, we introduce the syntactic and semantic notions related to probabilistic knowledge in general and to conditional constraint trees in particular.

2.1 PROBABILISTIC KNOWLEDGE

Before focusing on the details of conditional constraint trees, we give a general introduction to the kind of probabilistic knowledge considered in this work. We deal with conditional constraints over propositional events. They represent interval restrictions for conditional probabilities of propositional events. The formal background introduced in this section is commonly accepted in the literature (see, for example, [11] for other work in the same spirit).

We assume a nonempty and finite set of *basic events* $\mathcal{B} = \{B_1, B_2, \ldots, B_n\}$. The set of *conjunctive events* $\mathcal{C}_\mathcal{B}$ is the closure of $\mathcal{B}$ under the Boolean operation $\wedge$. We abbreviate the conjunctive event $C \wedge D$ by CD. The set of *propositional events* $\mathcal{G}_\mathcal{B}$ is the closure of $\mathcal{B}$

under the Boolean operations $\wedge$ and $\neg$. We abbreviate the propositional events $G \wedge H$ and $\neg G$ by GH and $\overline{G}$, respectively. The *true event* $B_1 \vee \neg B_1$ and the *false event* $B_1 \wedge \neg B_1$ are abbreviated by $\top$ and $\bot$, respectively. *Conditional constraints* are expressions of the form $(H|G)[u_1, u_2]$ with real numbers $u_1, u_2 \in [0,1]$ and propositional events G and H. In $(H|G)[u_1, u_2]$, we call G the *premise* and H the *conclusion*.

To define probabilistic interpretations of propositional events and of conditional constraints, we introduce atomic events and the binary relation $\Rightarrow$ between atomic and propositional events. The set of *atomic events* $\mathcal{A}_{\mathcal{B}}$ is defined by $\mathcal{A}_{\mathcal{B}} = \{E_1 E_2 \cdots E_n \mid E_i = B_i$ or $E_i = \overline{B}_i$ for all $i \in [1\!:\!n]\}$. The atomic events of our framework coincide with the more commonly known possible worlds from probabilistic logic [23]. For all atomic events A and all propositional events G, let $A \Rightarrow G$ iff $A\overline{G}$ is a propositional contradiction.

A *probabilistic interpretation* Pr is a mapping from $\mathcal{A}_{\mathcal{B}}$ to $[0,1]$ such that all $Pr(A)$ with $A \in \mathcal{A}_{\mathcal{B}}$ sum up to 1. Pr is extended in a well-defined way to propositional events G by $Pr(G) = \sum_{A \in \mathcal{A}_{\mathcal{B}},\, A \Rightarrow G} Pr(A)$. Pr is extended to conditional constraints by:

$$\begin{aligned} &Pr \models (H|G)[u_1, u_2] \text{ iff} \\ &\qquad u_1 \cdot Pr(G) \leq Pr(GH) \leq u_2 \cdot Pr(G)\,. \end{aligned}$$

Note that $Pr(G) = 0$ entails $Pr \models (H|G)[u_1, u_2]$ (see, for example, [14] and [11] for other work with the same semantics of conditional probability statements).

The notions of models, satisfiability, and logical consequence for conditional constraints are defined in the classical way. A probabilistic interpretation Pr is a *model* of a conditional constraint $(H|G)[u_1, u_2]$ iff $Pr \models (H|G)[u_1, u_2]$. Pr is a *model* of a set of conditional constraints KB, denoted $Pr \models KB$, iff Pr is a model of all $(H|G)[u_1, u_2] \in KB$. KB is *satisfiable* iff a model of KB exists. $(H|G)[u_1, u_2]$ is a *logical consequence* of KB, denoted $KB \models (H|G)[u_1, u_2]$, iff each model of KB is also a model of $(H|G)[u_1, u_2]$.

For a conditional constraint $(H|G)[u_1, u_2]$ and a set of conditional constraints KB, let $\boldsymbol{u}$ denote the set of all real numbers $u \in [0,1]$ for which there exists a model Pr of KB with $u \cdot Pr(G) = Pr(GH)$ and $Pr(G) > 0$. Now, we easily verify that $(H|G)[u_1, u_2]$ is a logical consequence of KB iff $u_1 \leq \inf \boldsymbol{u}$ and $u_2 \geq \sup \boldsymbol{u}$.

This observation yields a canonic notion of tightness for logical consequences of conditional constraints. The conditional constraint $(H|G)[u_1, u_2]$ is a *tight logical consequence* of KB, denoted $KB \models_{\mathit{tight}} (H|G)[u_1, u_2]$, iff $u_1 = \inf \boldsymbol{u}$ and $u_2 = \sup \boldsymbol{u}$.

The set $\boldsymbol{u}$ is a closed interval in the real numbers (see, for example, [11] and [19]). Note that for $\boldsymbol{u} = \emptyset$, we assume $\inf \boldsymbol{u} = 1$ and $\sup \boldsymbol{u} = 0$. Thus, we get $\boldsymbol{u} = \emptyset$ iff $KB \models (G|\top)[0,0]$ iff $KB \models_{\mathit{tight}} (H|G)[1,0]$ iff $KB \models (H|G)[u_1, u_2]$ for all $u_1, u_2 \in [0,1]$.

Based on the just introduced notion of tightness, probabilistic deduction problems and their solutions are more formally specified as follows.

A *probabilistic knowledge-base* $(\mathcal{B}, KB)$ consists of a set of basic events $\mathcal{B}$, and a set of conditional constraints KB over $\mathcal{G}_{\mathcal{B}}$ with $u_1 \leq u_2$ for all $(H|G)[u_1, u_2] \in KB$. A *probabilistic query* to a probabilistic knowledge-base $(\mathcal{B}, KB)$ is an expression of the form $\exists(F|E)[x_1, x_2]$ with $E, F \in \mathcal{G}_{\mathcal{B}}$ and two different variables x_1 and x_2. Its *tight answer* is the substitution $\sigma = \{x_1/u_1,\, x_2/u_2\}$ with $u_1, u_2 \in [0,1]$ such that $KB \models_{\mathit{tight}} (F|E)[u_1, u_2]$. A *correct answer* is a substitution $\sigma = \{x_1/u_1,\, x_2/u_2\}$ with $u_1, u_2 \in [0,1]$ such that $KB \models (F|E)[u_1, u_2]$.

Finally, we define the notions of soundness and of completeness related to inference rules and to techniques for probabilistic deduction. An inference rule $KB \vdash (H|G)[u_1, u_2]$ is *sound* iff $KB \models (H|G)[u_1, u_2]$, where $(H|G)[u_1, u_2]$ is a conditional constraint and KB is a set of conditional constraints. It is sound and *locally complete* iff $KB \models_{\mathit{tight}} (H|G)[u_1, u_2]$. A technique for probabilistic deduction is *sound* iff it computes a correct answer to any given probabilistic query. It is sound and *globally complete* iff it computes the tight answer to any given probabilistic query.

2.2 COMPUTATIONAL COMPLEXITY

In the framework of conditional constraints over propositional events, the problem of computing the tight answer to a probabilistic query is NP-hard, since it generalizes the satisfiability problem for probabilistic logic, which is known to be NP-complete from [13].

Surprisingly, the problem of computing the tight answer to a probabilistic query remains NP-hard even if we just use conditional constraints over basic events.

Theorem 2.1 *The problem of computing the tight answer to a probabilistic query over basic events that is directed to a probabilistic knowledge-base over basic events is NP-hard.*

Proof. The NP-complete problem of graph 3-colorability [12] can be polynomially-reduced to the problem of computing the tight answer to a probabilistic query over basic events that is directed to a probabilistic knowledge-base over basic events. The proof follows similar lines to the proof of NP-hardness of 2PSAT given in [13].

Let (V, E) be an undirected graph. We construct a probabilistic knowledge-base $(\mathcal{B}, KB)$ as follows. We initialize $(\mathcal{B}, KB)$ with $(\{B\}, \emptyset)$. For each node $v \in V$, we increase $\mathcal{B}$ by the new basic events B_v^1, B_v^2, and B_v^3. For each node $v \in V$ and for each $i \in \{1,2,3\}$, we increase KB by $(B|B_v^i)[1,1]$ and $(B_v^i|B)[1/3,1/3]$. For each node $v \in V$ and for each $i, j \in \{1,2,3\}$ with $i < j$, we increase KB by $(B_v^j|B_v^i)[0,0]$. For each edge $\{u, v\} \in E$ and for each $i \in \{1,2,3\}$, we increase KB by $(B_v^i|B_u^i)[0,0]$. Now, we show that (V, E) is 3-colorable iff $\{x_1/1,\ x_2/1\}$ is the tight answer to the probabilistic query $\exists(B|B)[x_1, x_2]$ to $(\mathcal{B}, KB)$, or equivalently, iff KB is satisfiable:

If (V, E) is 3-colorable, then there exists a mapping c_1 from V to $\{1,2,3\}$ with $c_1(u) \neq c_1(v)$ for all edges $\{u, v\} \in E$. Thus, if π is a cyclic permutation of the members in $\{1,2,3\}$ and if $c_2, c_3 : V \to \{1,2,3\}$ are defined by $c_2(v) = \pi(c_1(v))$ and $c_3(v) = \pi(c_2(v))$ for all nodes $v \in V$, then also $c_2(u) \neq c_2(v)$ and $c_3(u) \neq c_3(v)$ for all edges $\{u, v\} \in E$. For $j \in \{1,2,3\}$, let $A_j \in \mathcal{A}_{\mathcal{B}}$ such that $A_j \Rightarrow B$ and $A_j \Rightarrow B_v^i$ iff $c_j(v) = i$ for all nodes $v \in V$ and $i \in \{1,2,3\}$. If $Pr\colon \mathcal{A}_{\mathcal{B}} \to [0,1]$ is defined by $Pr(A) = 1/3$ for all $A \in \{A_1, A_2, A_3\}$ and by $Pr(A) = 0$ for all $A \in \mathcal{A}_{\mathcal{B}} \setminus \{A_1, A_2, A_3\}$, then Pr is a model of KB. Conversely, if there is a model Pr of KB, then there is an atomic event $A \in \mathcal{A}_{\mathcal{B}}$ with $Pr(A) > 0$. Thus, if $c : V \to \{1,2,3\}$ is defined by $c(v) = i$ iff $A \Rightarrow B_v^i$ for all nodes $v \in V$, then $c(u) \neq c(v)$ for all edges $\{u, v\} \in E$. Hence, (V, E) is 3-colorable. $\Box$

Hence, it is unlikely that there is an efficient algorithm for computing the tight answer to all probabilistic queries over basic events that are directed to any given probabilistic knowledge-base over basic events. However, there may be still efficient algorithms for solving more specialized probabilistic deduction problems.

The rest of this work deals with probabilistic deduction in conditional constraint trees. The next section provides a motivating example, which gives evidence of the practical importance of this kind of probabilistic deduction problems.

2.3 MOTIVATING EXAMPLE

A senior student in mathematics describes his experience about being successful at the university as follows. The success of a student (su) is influenced by how well-informed (wi) and how well-prepared (wp) the student is. Well-informedness can be reached by interviewing professors (pr) or by asking senior students (st). Being well-prepared is influenced by how much time is invested in books (bo), exercises (ex), and hobbies (ho). It is estimated that between 60 and 70 per cent of all well-informed students are successful, that more than 85 per cent of all successful students are well-informed, that more than 95 per cent of all successful students are well-prepared, and that more than 95 per cent of all well-prepared students are successful. This probabilistic knowledge completed by further probabilistic estimations is given by the probabilistic knowledge-base $(\mathcal{B}, KB)$ in Fig. 1. $\mathcal{B}$ is the set of nodes $\{\mathsf{su}, \mathsf{wi}, \mathsf{wp}, \mathsf{pr}, \mathsf{st}, \mathsf{bo}, \mathsf{ex}, \mathsf{ho}\}$. KB is the least set of conditional constraints containing $(Y|X)[u_1, u_2]$ for each arrow from X to Y labeled with u_1, u_2.

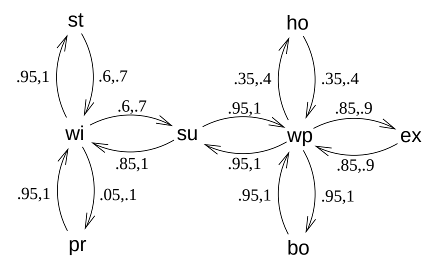

Figure 1: A Conditional Constraint Tree

Wondering, if it is useful for being successful at the university to interview the professors, to study on books, to spend the time on one's hobbies, or to do both studying on books and spending the time on one's hobbies, we get the probabilistic queries $\exists(\mathsf{su}|\mathsf{pr})[x_1, x_2]$, $\exists(\mathsf{su}|\mathsf{bo})[x_1, x_2]$, $\exists(\mathsf{su}|\mathsf{ho})[x_1, x_2]$, and $\exists(\mathsf{su}|\mathsf{bo}\,\mathsf{ho})[x_1, x_2]$ yielding the tight answers $\{x_1/0.00,\ x_2/1.00\}$, $\{x_1/0.90,\ x_2/1.00\}$, $\{x_1/0.30,\ x_2/0.46\}$, and $\{x_1/0.71,\ x_2/1.00\}$, respectively. The deduced results are quite intuitive, since studying on books has a positive effect on the success of a student, while investing time in hobbies is acting in a negative way.

Wondering, if successful students at the university interviewed their professors, if they studied on books, if they spent their time with their hobbies, or if they both studied on books and spent their time with their hobbies, we get the probabilistic queries $\exists(\mathsf{pr}|\mathsf{su})[x_1, x_2]$, $\exists(\mathsf{bo}|\mathsf{su})[x_1, x_2]$, $\exists(\mathsf{ho}|\mathsf{su})[x_1, x_2]$, and $\exists(\mathsf{bo}\,\mathsf{ho}|\mathsf{su})[x_1, x_2]$ yielding the tight answers $\{x_1/0.00,\ x_2/0.17\}$, $\{x_1/0.90,\ x_2/1.00\}$, $\{x_1/0.30,\ x_2/0.45\}$, and $\{x_1/0.25,\ x_2/0.45\}$, respectively. Again, the results make sense, since the biggest part of all successful students really studied, and just a small part of them spent their time with their hobbies.

Note that in this example, probabilistic interpretations of propositional events represent relative cardinalities of sets of objects. That is, more generally, in probabilistic propositional logics, we do not have the difference between probabilities on a set of possible worlds and those on a domain, like in probabilistic first-order logics (see, for example, [4] and [14]).

2.4 CONDITIONAL CONSTRAINT TREES

We formally define conditional constraint trees and queries to conditional constraint trees. We provide some additional examples, which are subsequently used as running examples.

A *(general) conditional constraint tree* is a probabilistic knowledge-base $(\mathcal{B}, KB)$ for which an undirected tree (a singly connected undirected graph) $(\mathcal{B}, \leftrightarrow)$ exists such that KB contains exactly one pair of conditional constraints $(B|A)[u_1, u_2]$ and $(A|B)[v_1, v_2]$ with $u_1, v_1 > 0$ for each pair of adjacent nodes A and B. A conditional constraint tree is *exact* iff $u_1 = u_2$ for all $(B|A)[u_1, u_2] \in KB$.

A *query to a conditional constraint tree* is a probabilistic query $\exists(F|E)[x_1, x_2]$ with conjunctive events E, F that are disjoint in their basic events and such that all paths from a basic event in E to a basic event in F have at least one basic event in common. A query $\exists(F|E)[x_1, x_2]$ to a conditional constraint tree is *premise-restricted* iff E is a basic event. It is *conclusion-restricted* iff F is a basic event. It is *complete* iff EF contains exactly the leaves of $(\mathcal{B}, \leftrightarrow)$.

Fig. 2 shows two conditional constraint trees of which the one on the left side is exact. $\exists(\mathsf{GHI}|\mathsf{ABEF})[x_1, x_2]$ is a query, while $\exists(\mathsf{AG}|\mathsf{EI})[x_1, x_2]$ is not a query to the conditional constraint trees of Fig. 2. Furthermore, $\exists(\mathsf{GHI}|\mathsf{A})[x_1, x_2]$ is a premise-restricted query, $\exists(\mathsf{C}|\mathsf{EFGHI})[x_1, x_2]$ a conclusion-restricted query, and $\exists(\mathsf{EFGHI}|\mathsf{A})[x_1, x_2]$ a premise-restricted complete query to the conditional constraint trees of Fig. 2.

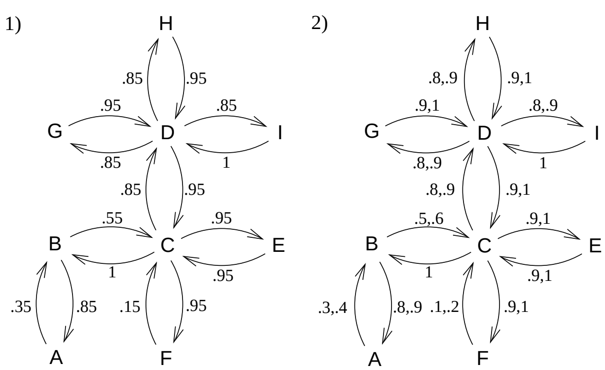

Figure 2: Two Conditional Constraint Trees

3 PROBABILISTIC DEDUCTION

The problem of computing the tight answer to a query to a conditional constraint tree can be divided into the subproblems of computing the tight answer to a premise-restricted query and the tight answer to a conclusion-restricted query.

For example, given the query $\exists(\mathsf{GHI}|\mathsf{ABEF})[x_1, x_2]$ to the conditional constraint trees in Fig. 2, we first compute the tight answer $\{y_1/u_1,\ y_2/u_2\}$ to the premise-restricted query $\exists(\mathsf{ABEF}|\mathsf{C})[y_1, y_2]$ and the tight answer $\{z_1/v_1,\ z_2/v_2\}$ to the conclusion-restricted query $\exists(\mathsf{C}|\mathsf{ABEF})[z_1, z_2]$. We then generate a new conditional constraint tree by replacing the subtree over the nodes A, B, C, E, and F by the new pair of conditional constraints $('\mathsf{ABEF}'|\mathsf{C})[u_1, u_2]$ and $(\mathsf{C}|'\mathsf{ABEF}')[v_1, v_2]$ over the nodes $'\mathsf{ABEF}'$ and C (in this new conditional constraint tree, ABEF acts as a new basic event $'\mathsf{ABEF}'$). Finally, we just compute the tight answer to the premise-restricted query $\exists(\mathsf{GHI}|'\mathsf{ABEF}')[x_1, x_2]$ to the new conditional constraint tree.

In the sequel, we restrict the considerations of this work to computing tight answers to premise-restricted queries. This subproblem reveals the main key ideas and the main theoretically interesting results of computing tight answers to queries. For more details on computing tight answers to queries, the interested reader may refer to our results in [19].

The problem of computing tight answers to premise-restricted queries can be reduced to the more specialized problem of calculating tight answers to premise-restricted complete queries. More precisely, for each premise-restricted query $\exists(F|E)[x_1, x_2]$ to a conditional constraint tree $(\mathcal{B}, KB)$, an equivalent premise-restricted complete query to a slightly transformed conditional constraint tree is generated as follows (see [19] for the correctness of this reduction):

In a first step, the conditional constraint tree is transformed by iteratively removing leaves not contained in EF, until all leaves are in EF. In a second transformation step, new basic events and new conditional constraints are added to the conditional constraint tree. For each non-leaf B contained in EF, $\mathcal{B}$ is increased by a new leaf B', and KB is increased by the new pair of conditional constraints $(B'|B)[1, 1]$ and $(B|B')[1, 1]$. The query is then transformed by replacing all non-leaves in E and F by the corresponding new leaves.

Hence, without loss of generality, we can restrict the subsequent considerations of this work to computing tight answers to premise-restricted complete queries.

3.1 EXACT CONDITIONAL CONSTRAINT TREES

In this section, let $(\mathcal{B}, KB)$ be an exact conditional constraint tree. To compute the tight answer to the premise-restricted complete query $\exists(F|E)[x_1, x_2]$, we start by defining a directed tree (that is, a directed acyclic graph in which each node has exactly one parent, except for the root that does not have any):

$$A \to B \quad \text{iff} \quad A \leftrightarrow B \text{ and } A \text{ is closer to } E \text{ than } B.$$

The directed tree $(\mathcal{B}, \to)$ is uniquely determined by the conditional constraint tree and the premise-restricted complete query. Fig. 3 shows $(\mathcal{B}, \to)$ for the premise-restricted complete query $\exists(\mathsf{EFGHI}|\mathsf{A})[x_1, x_2]$ to the exact conditional constraint tree in Fig. 2, left side.

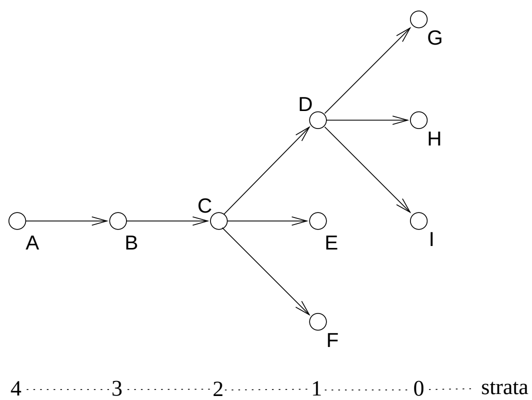

Figure 3: The Directed Tree $(\mathcal{B}, \to)$

The set of nodes $\mathcal{B}$ is partitioned into several strata. The lowest stratum contains only nodes with no children in $(\mathcal{B}, \to)$, the highest stratum contains the nodes with no parents in $(\mathcal{B}, \to)$ (that is, exactly the node of the premise E of the query). Fig. 3 also shows the different strata in our example.

At each node of $(\mathcal{B}, \to)$, we compute certain tightest bounds that are logically entailed by KB. More precisely, the tightest bounds at a node B are computed locally, by exploiting the tightest bounds that have been previously computed at the children of B. Hence, we iteratively compute the tightest bounds at the nodes of each stratum, starting with the nodes of the lowest stratum and terminating with the nodes of the highest stratum. We distinguish three different ways of computing tightest bounds at a node:

- initialization of a leaf (Leaf),
- chaining of an arrow and a subtree via a common node (Chaining),
- fusion of subtrees via a common node (Fusion).

Let us consider again the premise-restricted complete query $\exists(\mathsf{EFGHI}|\mathsf{A})[x_1, x_2]$ to the exact conditional constraint tree in Fig. 2, left side. Fig. 4 illustrates the three different ways of computing tightest bounds at a node (the common nodes for Chaining and Fusion are filled black). Table 1 shows the greatest lower and the least upper bounds that are computed at each node B of each stratum st. More precisely, $\alpha_1 = \inf Pr(BC)/Pr(B)$, $\alpha_2 = \sup Pr(BC)/Pr(B)$, $\beta_2 = \sup Pr(\overline{B}C)/Pr(B)$, and $\gamma_2 = \sup Pr(C)/Pr(B)$ subject to $Pr \models KB$ and $Pr(B) > 0$ (in Table 1, we abbreviate Leaf, Chaining, and Fusion by Le, Ch, and Fu, respectively). Table 1 also shows the requested tight answer $\{x_1/0.02, x_2/0.17\}$, which is given by the tightest bounds computed at the premise A.

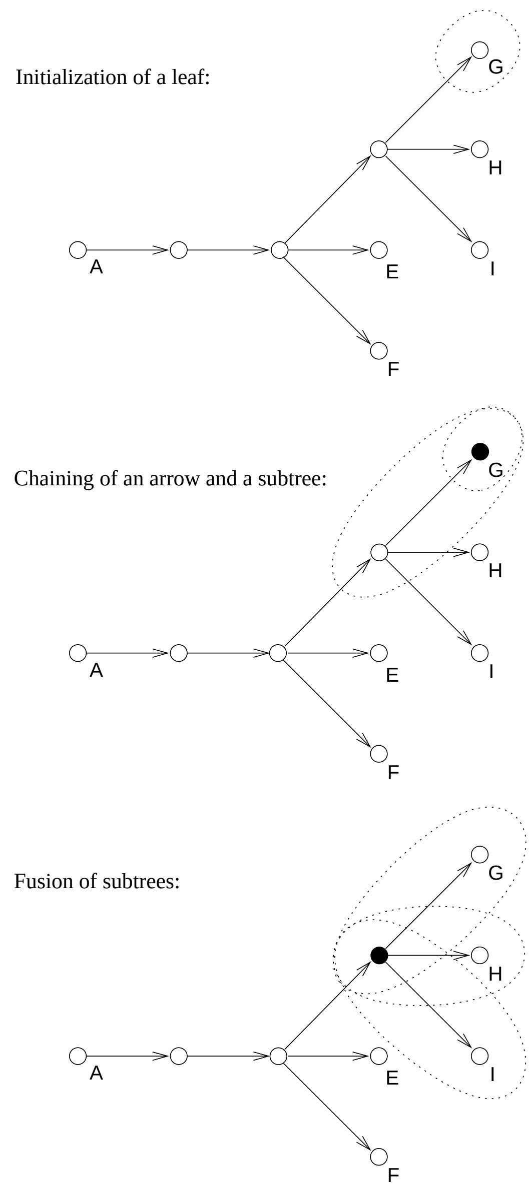

Figure 4: Local Computations in $(\mathcal{B}, \to)$

Table 1: Locally Computed Tightest Bounds

st	B	C	α_1	α_2	β_2	γ_2	
	G	G	1.0000	1.0000	0.0000	1.0000	(LE)
0	H	H	1.0000	1.0000	0.0000	1.0000	(LE)
	I	I	1.0000	1.0000	0.0000	1.0000	(LE)
	D	G	0.8500	0.8500	0.0447	0.8947	(CH)
1	D	H	0.8500	0.8500	0.0447	0.8947	(CH)
	D	I	0.8500	0.8500	0.0000	0.8500	(CH)
	D	GHI	0.5500	0.8500	0.0000	0.8500	(FU)
1	E	E	1.0000	1.0000	0.0000	1.0000	(LE)
	F	F	1.0000	1.0000	0.0000	1.0000	(LE)
	C	GHI	0.4474	0.7605	0.0447	0.7605	(CH)
2	C	E	0.9500	0.9500	0.0500	1.0000	(CH)
	C	F	0.9500	0.9500	5.3833	6.3333	(CH)
2	C	EFGHI	0.3474	0.7605	0.0447	0.7605	(FU)
3	B	EFGHI	0.1911	0.4183	0.0246	0.4183	(CH)
4	A	EFGHI	0.0169	0.1722	0.0719	0.1722	(CH)

Note that computing the requested tight answer by the classical linear programming approach would result in solving two linear programs, each of them having 512 variables and 34 linear inequalities.

Finally, we focus on the technical details. We present the functions P_1 and P_2, which compute the described greatest lower and least upper bounds. For this purpose, we need the following preparative definitions.

Let $P(BC, B) = u$ for all $(C|B)[u, u] \in KB$.

A node B is a *leaf* if it does not have any children. For all leaves B let $B^{\uparrow} = B$. For all the other nodes B let $B^{\uparrow}$ be the conjunction of all the children of B. For all leaves C let $L(C) = C$. For all the other conjunctive events C let $L(C)$ be the conjunction of all the leaves that are in C or that are descendants of a node in C.

In the sequel, let B be a node and let $C = B^{\uparrow}$. The case $C = B$ refers to the initialization of the leaf B, the case $C = B_1$ with a node $B_1 \neq B$ to the chaining of the arrow $B \to B_1$ and a subtree via the common node B_1, and the case $C = B_1 B_2 \ldots B_k$ with $k > 1$ nodes $B_1, B_2, \ldots, B_k$ to the fusion of k subtrees via the common node B.

We now define the function P_1 for the computation of greatest lower bounds. Note that its existence is already known from [27] in the framework of deductive database systems. Let $P_1(BL(C), B) = \alpha_1$, where α_1 in LEAF, CHAINING, and FUSION is given as follows:

LEAF ($C = B$):

$$\alpha_1 \;=\; 1$$

CHAINING ($C = B_1$):

$$\alpha_1 \;=\; \max(0, P(BC, B) \cdot (1 + \tfrac{P_1(CL(C^{\uparrow}), C) - 1}{P(CB, C)}))$$

FUSION ($C = B_1 B_2 \ldots B_k$):

$$\alpha_1 \;=\; \max(0, 1 - k + \sum_{i=1}^{k} P_1(BL(B_i), B))$$

To formulate that P_1 computes greatest lower bounds, we need the following definitions. Let $\mathcal{B}(B, C)$ comprise B, all nodes in C and all descendants of a node in C. Let $KB(B, C)$ be the set of all conditional constraints of KB over $\mathcal{B}(B, C)$. Let $Mo(B, C)$ be the set of all models of $KB(B, C)$ over $\mathcal{B}(B, C)$.

Now, the function P_1 is sound and globally complete with respect to B and C iff $P_1(BL(C), B) = \alpha_1$ is the infimum of $Pr(BL(C)) \,/\, Pr(B)$ subject to $Pr \in Mo(B, C)$ and $Pr(B) > 0$. Thus, the next theorem shows soundness and global completeness of P_1.

Theorem 3.1

a) $\alpha_1 \cdot Pr(B) \leq Pr(BL(C))$ *for all* $Pr \in Mo(B, C)$.

b) There exists $Pr \in Mo(B, C)$ *with* $Pr(B) > 0$ *and* $\alpha_1 \cdot Pr(B) = Pr(BL(C))$.

Proof. The proof is given in full detail in [19]. □

Next, we present the function P_2 for computing least upper bounds. Note that it has been unknown in the literature so far if such a function exists. This function P_2 provides the crucial result that for exact conditional constraint trees, there are local probabilistic deduction techniques that are sound and globally complete. In detail, let $P_2(BL(C), B) = \alpha_2$, $P_2(\overline{B}L(C), B) = \beta_2$, and $P_2(L(C), B)) = \gamma_2$, where α_2, β_2, and γ_2 in LEAF, CHAINING, and FUSION are given as follows:

LEAF ($C = B$):

$$(\alpha_2, \beta_2, \gamma_2) \;=\; (1, 0, 1)$$

CHAINING ($C = B_1$):

$$\begin{aligned}\alpha_2 \;=\; \min(1, &P(BC, B) \cdot \tfrac{P_2(L(C^{\uparrow}), C)}{P(CB, C)},\\ &1 - P(BC, B) \cdot (1 - \tfrac{P_2(CL(C^{\uparrow}), C)}{P(CB, C)}),\\ &P(BC, B) \cdot (1 + \tfrac{P_2(\overline{C}L(C^{\uparrow}), C)}{P(CB, C)}))\end{aligned}$$

$$\begin{aligned}\beta_2 \;=\; \min(&P(BC, B) \cdot (\tfrac{P_2(\overline{C}L(C^{\uparrow}), C) + 1}{P(CB, C)} - 1),\\ &P(BC, B) \cdot \tfrac{P_2(L(C^{\uparrow}), C)}{P(CB, C)})\end{aligned}$$

$$\gamma_2 \;=\; P(BC, B) \cdot \tfrac{P_2(L(C^{\uparrow}), C)}{P(CB, C)}$$

FUSION ($C = B_1 B_2 \ldots B_k$):

$$\alpha_2 \;=\; \min_{i \in [1:k]} P_2(BL(B_i), B)$$

$$\beta_2 \;=\; \min_{i \in [1:k]} P_2(\overline{B}L(B_i), B)$$

$$\gamma_2 = \min(\min_{i\in[1:k]} P_2(L(B_i),B)$$
$$\min_{i,j\in[1:k],i\neq j}(P_2(BL(B_i),B)+$$
$$P_2(\overline{B}L(B_j),B)))$$

The function P_2 is sound and globally complete with respect to B and C iff α_2, β_2, and γ_2 are the supremum of $Pr(BL(C))\,/\,Pr(B)$, $Pr(\overline{B}L(C))\,/\,Pr(B)$, and $Pr(L(C))\,/\,Pr(B)$, respectively, subject to $Pr \in Mo(B,C)$ and $Pr(B) > 0$. Hence, the following theorem shows soundness and global completeness of P_2 (actually, it shows even more to enable a proof by induction on the recursive definition of P_2).

Theorem 3.2

a) $Pr(BL(C)) \leq \alpha_2 \cdot Pr(B)$, $Pr(\overline{B}L(C)) \leq \beta_2 \cdot Pr(B)$, *and* $Pr(L(C)) \leq \gamma_2 \cdot Pr(B)$ *for all* $Pr \in Mo(B,C)$.

b) There exists $Pr \in Mo(B,C)$ *with* $Pr(B) > 0$, $Pr(BL(C)) = \alpha_2 \cdot Pr(B)$, *and* $Pr(L(C)) = \gamma_2 \cdot Pr(B)$.

c) There exists $Pr \in Mo(B,C)$ *with* $Pr(B) > 0$, $Pr(\overline{B}L(C)) = \beta_2 \cdot Pr(B)$, *and* $Pr(L(C)) = \gamma_2 \cdot Pr(B)$.

Proof. The proof is given in full detail in [19]. □

Note that Theorem 3.2 also shows that $P_2(L(B_i),B) \leq P_2(BL(B_i),B)+P_2(\overline{B}L(B_i),B)$ for all $i \in [1:k]$. Thus, $\min_{i,j\in[1:k],i\neq j}(P_2(BL(B_i),B)+P_2(\overline{B}L(B_j),B))$ in the definition of γ_2 in FUSION can be replaced by $\alpha_2+\beta_2$ for an increased efficiency in computing γ_2 by exploiting the already computed values of α_2 and β_2.

Briefly, by Theorems 3.1 and 3.2, the tight answer to the premise-restricted complete query $\exists(F|E)[x_1,x_2]$ is $\{x_1/P_1(EL(E^{\uparrow}),E),\, x_2/P_2(EL(E^{\uparrow}),E)\}$.

The time complexity of computing the tight answer to a premise-restricted complete query this way, by using the functions P_1 and P_2, is analyzed in Section 3.3.

3.2 CONDITIONAL CONSTRAINT TREES

In this section, we focus on computing the tight answer to premise-restricted complete queries to general conditional constraint trees. In the sequel, let $(\mathcal{B},KB)$ be a conditional constraint tree and let $\exists(F|E)[x_1,x_2]$ be a premise-restricted complete query.

We may think that our local deduction technique for exact conditional constraint trees of the previous section can easily be generalized to conditional constraint trees. In fact, this is true as far as the computation of greatest lower bounds is concerned. However, the computation of least upper bounds cannot be generalized that easily from exact conditional constraint trees to conditional constraint trees. More precisely, generalizing the computation of least upper bounds results in solving global nonlinear programs. These nonlinear programs and our way to solve them are illustrated by the following example.

Let the conditional constraint tree $(\mathcal{B},KB)$ be given by $\mathcal{B}=\{\mathsf{A},\mathsf{B},\mathsf{C},\mathsf{D}\}$ and $KB=\{(\mathsf{B}|\mathsf{A})[u_1,u_2], (\mathsf{A}|\mathsf{B})[v_1,v_2], (\mathsf{C}|\mathsf{B})[x_1,x_2], (\mathsf{B}|\mathsf{C})[y_1,y_2], (\mathsf{D}|\mathsf{C})[r_1,r_2], (\mathsf{C}|\mathsf{D})[s_1,s_2]\}$, and let the premise-restricted complete query be defined by $\exists(\mathsf{D}|\mathsf{A})[z_1,z_2]$.

The requested least upper bound is the maximum of z subject to $u \in [u_1,u_2]$, $x \in [x_1,x_2]$, and the following nonlinear inequalities in (1) to (5):

$$z \leq 1 \tag{1}$$
$$z \leq 1-u+\tfrac{u}{v_1}-\tfrac{ux}{v_1}+\tfrac{uxr_2}{v_1y_1} \tag{2}$$
$$z \leq 1-u+\tfrac{ux}{v_1}-\tfrac{uxr_2}{v_1y_1}+\tfrac{uxr_2}{v_1y_1s_1} \tag{3}$$
$$z \leq u-\tfrac{ux}{v_1}+\tfrac{ux}{v_1y_1}-\tfrac{uxr_2}{v_1y_1}+\tfrac{uxr_2}{v_1y_1s_1} \tag{4}$$
$$z \leq \tfrac{uxr_2}{v_1y_1s_1} \tag{5}$$

We transform this nonlinear program into an equivalent linear program as follows. The maximum of z subject to $u \in [u_1,u_2]$, $x \in [x_1,x_2]$, and the nonlinear inequalities in (1) to (5) is equal to the maximum of z subject to the following system of linear inequalities over z and $x_B \geq 0$ ($B \in \mathcal{B}$):

$$\begin{array}{lllllll}
z \leq x_{\mathsf{A}} & & & & & & \\
z \leq x_{\mathsf{A}} & + & \frac{1-v_1}{v_1}\cdot x_{\mathsf{B}} & - & \frac{y_1}{v_1y_1}\cdot x_{\mathsf{C}} & + & \frac{s_1}{v_1y_1s_1}\cdot x_{\mathsf{D}} \\
z \leq x_{\mathsf{A}} & - & \frac{v_1}{v_1}\cdot x_{\mathsf{B}} & + & \frac{y_1}{v_1y_1}\cdot x_{\mathsf{C}} & + & \frac{1-s_1}{v_1y_1s_1}\cdot x_{\mathsf{D}} \\
z \leq & & \frac{v_1}{v_1}\cdot x_{\mathsf{B}} & + & \frac{1-y_1}{v_1y_1}\cdot x_{\mathsf{C}} & + & \frac{1-s_1}{v_1y_1s_1}\cdot x_{\mathsf{D}} \\
z \leq & & & & & & \frac{1}{v_1y_1s_1}\cdot x_{\mathsf{D}}
\end{array}$$

$$1 \leq x_{\mathsf{A}} \leq 1, \qquad u_1\cdot x_{\mathsf{A}} \leq x_{\mathsf{B}} \leq u_2\cdot x_{\mathsf{A}}$$
$$x_1\cdot x_{\mathsf{B}} \leq x_{\mathsf{C}} \leq x_2\cdot x_{\mathsf{B}}, \quad r_1\cdot x_{\mathsf{C}} \leq x_{\mathsf{D}} \leq r_2\cdot x_{\mathsf{C}}$$

More generally, least upper bounds to premise-restricted complete queries to conditional constraint trees can be expressed by similar nonlinear programs, which can similarly be transformed into linear programs.

For example, let us consider the premise-restricted complete query $\exists(\mathsf{EFGHI}|\mathsf{A})[x_1,x_2]$ to the conditional constraint tree in Fig. 2, right side. The requested least upper bound is the maximum of x subject to the system of linear inequalities in Fig. 5 (note that we actually generated 72 linear inequalities of which 31 were trivially subsumed by others).

Thus, in this example, the requested least upper bound is computed by solving a linear program that has 10 variables and 72 linear inequalities.

$$
\begin{array}{rcl@{\qquad}rcl}
x & \leq & x_{\mathsf{A}} & x & \leq & \frac{5}{4}x_{\mathsf{C}} + \frac{5}{36}x_{\mathsf{E}} \\
x & \leq & x_{\mathsf{A}} + \frac{1}{4}x_{\mathsf{B}} - \frac{5}{4}x_{\mathsf{C}} + \frac{5}{4}x_{\mathsf{D}} & x & \leq & \frac{5}{4}x_{\mathsf{C}} + \frac{45}{4}x_{\mathsf{F}} \\
x & \leq & x_{\mathsf{A}} + \frac{1}{4}x_{\mathsf{B}} - \frac{5}{4}x_{\mathsf{C}} + \frac{5}{4}x_{\mathsf{E}} & x & \leq & \frac{5}{4}x_{\mathsf{D}} + \frac{5}{36}x_{\mathsf{E}} \\
x & \leq & x_{\mathsf{A}} + \frac{1}{4}x_{\mathsf{B}} - \frac{5}{4}x_{\mathsf{C}} + \frac{5}{4}x_{\mathsf{F}} & x & \leq & \frac{5}{36}x_{\mathsf{D}} + \frac{5}{4}x_{\mathsf{E}} \\
x & \leq & x_{\mathsf{A}} + \frac{1}{4}x_{\mathsf{B}} - \frac{5}{4}x_{\mathsf{D}} + \frac{25}{18}x_{\mathsf{G}} & x & \leq & \frac{5}{36}x_{\mathsf{D}} + \frac{5}{4}x_{\mathsf{F}} \\
x & \leq & x_{\mathsf{A}} + \frac{1}{4}x_{\mathsf{B}} - \frac{5}{4}x_{\mathsf{D}} + \frac{25}{18}x_{\mathsf{H}} & x & \leq & \frac{5}{36}x_{\mathsf{E}} + \frac{5}{4}x_{\mathsf{F}} \\
x & \leq & x_{\mathsf{A}} + \frac{1}{4}x_{\mathsf{B}} - \frac{5}{4}x_{\mathsf{D}} + \frac{25}{18}x_{\mathsf{I}} & x & \leq & \frac{25}{18}x_{\mathsf{E}} \\
x & \leq & x_{\mathsf{A}} - x_{\mathsf{B}} + \frac{5}{4}x_{\mathsf{C}} + \frac{5}{36}x_{\mathsf{F}} & x & \leq & \frac{25}{2}x_{\mathsf{F}} \\
x & \leq & x_{\mathsf{B}} + \frac{1}{9}x_{\mathsf{D}} & x & \leq & \frac{25}{18}x_{\mathsf{G}} \\
x & \leq & x_{\mathsf{B}} + \frac{1}{9}x_{\mathsf{E}} & x & \leq & \frac{25}{18}x_{\mathsf{H}} \\
x & \leq & x_{\mathsf{B}} + \frac{1}{9}x_{\mathsf{F}} & x & \leq & \frac{25}{18}x_{\mathsf{I}} \\
x & \leq & \frac{5}{4}x_{\mathsf{C}} + \frac{5}{36}x_{\mathsf{D}} & & &
\end{array}
$$

$$1 \leq x_{\mathsf{A}} \leq 1$$

$$
\begin{array}{ccccc@{\qquad}ccccc}
\frac{3}{10}x_{\mathsf{A}} & \leq & x_{\mathsf{B}} & \leq & \frac{2}{5}x_{\mathsf{A}}, & \frac{1}{2}x_{\mathsf{B}} & \leq & x_{\mathsf{C}} & \leq & \frac{3}{5}x_{\mathsf{B}} \\
\frac{9}{10}x_{\mathsf{C}} & \leq & x_{\mathsf{F}} & \leq & x_{\mathsf{C}}, & \frac{9}{10}x_{\mathsf{C}} & \leq & x_{\mathsf{E}} & \leq & x_{\mathsf{C}} \\
\frac{4}{5}x_{\mathsf{C}} & \leq & x_{\mathsf{D}} & \leq & \frac{9}{10}x_{\mathsf{C}}, & \frac{4}{5}x_{\mathsf{D}} & \leq & x_{\mathsf{G}} & \leq & \frac{9}{10}x_{\mathsf{D}} \\
\frac{4}{5}x_{\mathsf{D}} & \leq & x_{\mathsf{H}} & \leq & \frac{9}{10}x_{\mathsf{D}}, & \frac{4}{5}x_{\mathsf{D}} & \leq & x_{\mathsf{I}} & \leq & \frac{9}{10}x_{\mathsf{D}}
\end{array}
$$

Figure 5: The Generated Linear Inequalities

Note that computing the requested least upper bound by the classical linear programming approach would result in solving a linear program that has 512 variables and 34 linear inequalities.

Let us now focus on the technical details. We subsequently generalize the function P_1 of Section 3.1 in a straightforward way to compute greatest lower bounds in conditional constraint trees. Moreover, we present a linear program for computing the requested least upper bound in conditional constraint trees.

Let $P_1(BC, B) = u_1$ for all $(C|B)[u_1, u_2] \in KB$.

In the sequel, let B be a node and let $C = B^{\uparrow}$. Like in Section 3.1, $C = B$, $C = B_1$ with a node $B_1 \neq B$, and $C = B_1 B_2 \ldots B_k$ with $k > 1$ nodes $B_1, B_2, \ldots, B_k$ refer to Leaf, Chaining, and Fusion, respectively.

We define the generalized function P_1 for computing greatest lower bounds in conditional constraint trees. Note that its existence is already known from [27] in the framework of deductive database systems. Let $P_1(BL(C), B) = \alpha_1$, where α_1 in Leaf, Chaining, and Fusion is given as follows:

Leaf ($C = B$):

$$\alpha_1 \;=\; 1$$

Chaining ($C = B_1$):

$$\alpha_1 \;=\; \max(0, P_1(BC, B) \cdot (1 + \tfrac{P_1(CL(C^{\uparrow}), C) - 1}{P_1(CB, C)}))$$

Fusion ($C = B_1 B_2 \ldots B_k$):

$$\alpha_1 \;=\; \max(0, 1 - k + \sum_{i=1}^{k} P_1(BL(B_i), B))$$

The function P_1 is sound and globally complete with respect to B and C iff $P_1(BL(C), B) = \alpha_1$ is the infimum of $Pr(BL(C)) \,/\, Pr(B)$ subject to $Pr \in Mo(B, C)$ and $Pr(B) > 0$. Thus, the next theorem shows soundness and global completeness of P_1.

Theorem 3.3

a) $\alpha_1 \cdot Pr(B) \leq Pr(BL(C))$ *for all* $Pr \in Mo(B, C)$.

b) There exists $Pr \in Mo(B, C)$ *with* $Pr(B) > 0$ *and* $\alpha_1 \cdot Pr(B) = Pr(BL(C))$.

Proof. The claims follow from Theorem 3.1. □

Next, we concentrate on computing the requested least upper bound. More precisely, we formulate a linear program similar to the ones described in the two examples. We do this by defining the function I over the variables x_B ($B \in \mathcal{B}$). Each admissible instantiation of the variables corresponds to an exact conditional constraint tree as instance of the considered conditional constraint tree. Given such an admissible instantiation of the variables, I now provides least upper bounds for the corresponding exact conditional constraint tree. Hence, to obtain least upper bounds for the conditional constraint tree, we must maximize the value of I subject to all admissible instantiations of the variables. In detail, let $I(BL(C), B) = \alpha_2$, $I(\overline{B}L(C), B) = \beta_2$, and $I(L(C), B)) = \gamma_2$, where α_2, β_2, and γ_2 in Leaf, Chaining, and Fusion are given as follows:

Leaf ($C = B$):

$$(\alpha_2, \beta_2, \gamma_2) \;=\; (x_B, 0, x_B)$$

Chaining ($C = B_1$):

$$\alpha_2 \;=\; \min(x_B, \tfrac{I(L(C^{\uparrow}), C)}{P_1(CB, C)}, x_C + \tfrac{I(\overline{C}L(C^{\uparrow}), C)}{P_1(CB, C)}, x_B - x_C + \tfrac{I(CL(C^{\uparrow}), C)}{P_1(CB, C)})$$

$$\beta_2 \;=\; \min(\tfrac{1 - P_1(CB, C)}{P_1(CB, C)} \cdot x_C + \tfrac{I(\overline{C}L(C^{\uparrow}), C)}{P_1(CB, C)}, \tfrac{I(L(C^{\uparrow}), C)}{P_1(CB, C)})$$

$$\gamma_2 \;=\; \tfrac{I(L(C^{\uparrow}), C)}{P_1(CB, C)}$$

Fusion ($C = B_1 B_2 \ldots B_k$):

$$\alpha_2 \;=\; \min_{i \in [1:k]} I(BL(B_i), B)$$

$$\beta_2 = \min_{i\in[1:k]} I(\overline{B}L(B_i), B)$$

$$\gamma_2 = \min(\min_{i\in[1:k]} I(L(B_i), B), \min_{i,j\in[1:k], i\neq j} (I(BL(B_i), B) + I(\overline{B}L(B_j), B)))$$

It remains to specify the admissibility of an instantiation of the variables. This is done by the system of linear inequalities $J(E)$, which is defined as the least set of linear inequalities over $x_B \geq 0$ ($B \in \mathcal{B}$) that contains $1 \leq x_E \leq 1$ and $u_1 \cdot x_B \leq x_C \leq u_2 \cdot x_B$ for all $(C|B)[u_1, u_2] \in KB$ with $B \rightarrow C$ (that is, B is the parent of C). Note that $J(E)$ is just related to the final computation of the requested least upper bound.

We implicitly performed the variable transformation described in the two examples. The next lemma shows that this variable transformation is indeed correct for conditional constraint trees.

Lemma 3.4

a) If x_B ($B \in \mathcal{B}$) satisfies $J(E)$, then for all conditional constraints $(C|B)[u_1, u_2] \in KB$ such that $B \rightarrow C$, there exists $u_C \in [u_1, u_2]$ with $x_C = u_C \cdot x_B$.

b) Let $u_C \in [u_1, u_2]$ for all $(C|B)[u_1, u_2] \in KB$ such that $B \rightarrow C$. There exists x_B ($B \in \mathcal{B}$) with $J(E)$ and $x_C = u_C \cdot x_B$ for all nodes C with parent B.

Proof. a) For all nodes C with parent B, let u_C be defined by $u_C = x_C \,/\, x_B$.

b) Let $x_E = 1$, and for all nodes C with parent B, let x_C be defined by $x_C = u_C \cdot x_B$. □

We are now ready to formulate an optimization problem for computing the requested least upper bound.

Theorem 3.5 *Let X_2 be the maximum of x subject to $x \leq I(EL(E^{\uparrow}), E)$ and $J(E)$.*

a) $Pr(EL(E^{\uparrow})) \leq X_2 \cdot Pr(E)$ for all $Pr \in Mo(E, E^{\uparrow})$.

b) There exists $Pr \in Mo(E, E^{\uparrow})$ with $Pr(E) > 0$ and $Pr(EL(E^{\uparrow})) = X_2 \cdot Pr(E)$.

Proof. Let $P(CB, C) = v_1$ for all $(B|C)[v_1, v_2] \in KB$ such that $B \rightarrow C$. By Theorem 3.2, the requested least upper bound is the maximum of x subject to $x \leq P_2(EL(E^{\uparrow}), E)$ and $P(BC, B) = u_C \in [u_1, u_2]$ for all $(C|B)[u_1, u_2] \in KB$ such that $B \rightarrow C$. By Lemma 3.4, we can equivalently maximize x subject to $x \leq I(EL(E^{\uparrow}), E)$ and $J(E)$. □

We wonder how to solve the generated optimization problem, since $I(EL(E^{\uparrow}), E)$ may still contain min-operations that cannot be tackled by linear programming. Given a technique for solving this optimization problem, we are also interested in a rough idea on the overall time complexity of computing the requested least upper bound this way. Moreover, we are interested in possible improvements to increase efficiency. These topics are discussed in the following.

If $I(EL(E^{\uparrow}), E)$ does not contain any min-operations at all, then the generated optimization problem is already a linear program. Otherwise, it can easily be transformed into a linear program. In a first transformation step, all inner min-operations are eliminated. This can easily be done due to the well-structuredness of $I(EL(E^{\uparrow}), E)$. In a second step, the only remaining outer min-operation is eliminated by introducing exactly one linear inequality for each contained operand. In these linear inequalities, the operands of the outer min-operation are upper bounds of x.

To get a rough idea on the time complexity of computing the requested least upper bound this way, we must analyze the size of the generated linear programs. It is given by the number of variables, the number of linear inequalities in $J(E)$, and the number of linear inequalities extracted from $x \leq I(EL(E^{\uparrow}), E)$. The latter number is quite worrying, since γ_2 seems to produce many min-operands in Fusion. Moreover, γ_2 contains $I(BL(B_i), B)$ in Fusion, and α_2 contains $I(L(C^{\uparrow}), C)$ in Chaining. So, due to this crossed dependency, the overall number of generated linear inequalities is likely to 'explode' for trees that branch very often.

In order to avoid these problems, we now introduce the auxiliary function I' over the variables x_B ($B \in \mathcal{B}$). Let $I'(BL(C), B) = \alpha'_2$, $I'(\overline{B}L(C), B) = \beta'_2$, and $I'(L(C), B)) = \gamma'_2$, where α'_2, β'_2, and γ'_2 in Leaf, Chaining, and Fusion are given as follows:

Leaf ($C = B$):

$$(\alpha'_2, \beta'_2, \gamma'_2) = (x_B, 0, x_B)$$

Chaining ($C = B_1$):

$$\alpha'_2 = \min(x_B,\, x_C + \tfrac{I'(\overline{C}L(C^{\uparrow}), C)}{P_1(CB, C)},\, x_B - x_C + \tfrac{I'(CL(C^{\uparrow}), C)}{P_1(CB, C)})$$

$$\beta'_2 = \tfrac{1 - P_1(CB, C)}{P_1(CB, C)} \cdot x_C + \tfrac{I'(\overline{C}L(C^{\uparrow}), C)}{P_1(CB, C)}$$

$$\gamma'_2 = \tfrac{I'(L(C^{\uparrow}), C)}{P_1(CB, C)}$$

Fusion ($C = B_1 B_2 \ldots B_k$):

$$\alpha'_2 = \min_{i\in[1:k]} I'(BL(B_i), B)$$

$$\beta'_2 = \min_{i\in[1:k]} I'(\overline{B}L(B_i), B)$$

$$\gamma'_2 = \min(\min_{i\in[1:k]} I'(L(B_i),B), \min_{i,j\in[1:k],i\neq j}(I'(BL(B_i),B)+ I'(\overline{B}L(B_j),B)))$$

Note that in the definition of α'_2 in CHAINING, we can consider the cases $C^\uparrow = C$ and $C^\uparrow \neq C$ separately. Since simply $\alpha'_2 = x_C$ for $C^\uparrow = C$, we reduce the number of generated linear inequalities this way.

The next lemma shows that the function I can be expressed in terms of the auxiliary function I'.

Lemma 3.6 *For all x_B $(B \in \mathcal{B})$ that satisfy $J(E)$:*

$$\alpha_2 = \min(\alpha'_2, \gamma'_2),\ \beta_2 = \min(\beta'_2, \gamma'_2),\ \textit{and}\ \gamma_2 = \gamma'_2\,.$$

Proof sketch. The claim can be proved by induction on the recursive definition of I. □

Briefly, by Theorem 3.3, Theorem 3.5, and Lemma 3.6, the tight answer to the premise-restricted complete query $\exists(F|E)[x_1,x_2]$ is $\{x_1/P_1(EL(E^\uparrow),E),\ x_2/X_2\}$, where X_2 is given by the maximum of x subject to $x \leq I'(EL(E^\uparrow),E)$, $x \leq I'(L(E^\uparrow),E)$, and $J(E)$.

In our example, we get $\{x_1/0.00,\ x_2/0.27\}$ as the tight answer to the query $\exists(\mathsf{EFGHI}|\mathsf{A})[x_1,x_2]$ to the conditional constraint tree in Fig. 2, right side.

The time complexity of this way of computing the requested greatest lower bound and especially the requested least upper bound is analyzed in Section 3.3.

3.3 COMPLEXITY

For exact conditional constraint trees, our approach to compute the tight answer in several strata by the functions P_1 and P_2 runs in time $O(n)$, where n denotes the number of nodes. This result is a consequence of the following observations. The directed tree can be computed in time $O(n)$. An initialization of a leaf with a constant number of assignments is performed exactly for each leaf of the directed tree, a chaining with a constant number of arithmetic operations is performed exactly for each arrow of the directed tree. Hence, initializing all leaves and performing all chainings runs in time $O(n)$. A fusion is done for each branching of the directed tree, using linear time in the number of branches. Thus, all fusions together run in time $O(n)$.

For general conditional constraint trees, the requested greatest lower bound is analogously computed in time $O(n)$. The requested least upper bound is computed in polynomial time in n. This result follows from the fact that linear programming runs in polynomial time in the size of the linear programs (see, for example, [24] and [26]). The size of the generated linear programs is given by the number of variables and the number of linear inequalities. The number of variables is $n+1$. The number of linear inequalities in $J(E)$ is $2n$. By induction on the recursive definition of I', it can be shown that the number of min-operands in α'_2, β'_2, and γ'_2 is limited by $|\mathcal{B}(B,C)|^2$, $|\mathcal{B}(B,C)|$, and $|\mathcal{B}(B,C)|^4$, respectively. Hence, the number of linear inequalities extracted from $x \leq I'(EL(E^\uparrow),E)$ and $x \leq I'(L(E^\uparrow),E)$ is limited by $|\mathcal{B}(E,E^\uparrow)|^2 + |\mathcal{B}(E,E^\uparrow)|^4 = n^2+n^4$. Thus, the overall number of generated linear inequalities l is limited by $l_u = 2n + n^2 + n^4$.

Note that l_u is a very rough upper bound for l, in many conditional constraint trees (especially in those that branch very rarely), l is much lower than l_u. For example, taking a complete binary tree with $n = 127$ nodes, we get only $l = 19\,964$ compared to $l_u = 260\,161\,024$. In the example of Section 3.2 with $n = 9$ nodes, we get only $l = 72$ compared to $l_u = 6\,660$. Another example is a tree that is degenerated to a chain of basic events. In this case, we get $l = 5n + 1$, that is, the overall number of generated linear inequalities is linear in n.

As a comparison, we shortly focus on the classical linear programming approach to probabilistic deduction. It runs in exponential time in n. In the special case of computing tight answers to queries to exact and to general conditional constraint trees, two linear programs are to be solved, each of them having 2^n variables and $4n-2$ linear inequalities.

4 SUMMARY AND CONCLUSIONS

We showed that probabilistic deduction with conditional constraints over basic events is NP-hard. We then presented very efficient and globally complete techniques for probabilistic deduction in conditional constraint trees. For exact conditional constraint trees, greatest lower and least upper bounds are deduced in linear time in the size of the conditional constraint trees. For general conditional constraint trees, greatest lower bounds are deduced in the same way, in linear time in the size of the conditional constraint trees. However, computing least upper bounds seems to be computationally more difficult. It is done by solving global nonlinear programs. We showed how to transform these nonlinear programs into equivalent linear programs, which are solvable in polynomial time in the size of the conditional constraint trees.

Conditional constraint trees and the analyzed probabilistic deduction problems related to them seem to be very interesting for many practical applications. Like Bayesian networks, they represent well-structured probabilistic knowledge-bases, which have an intuitive graphical representation. However, differently from Bayesian networks, conditional constraint trees do not require any probabilistic independence assumptions. So, rather than being in competition with Bayesian networks, conditional constraint trees are to be understood as a complement to Bayesian networks, useful for applications in which well-structured independence assumptions do not hold or are difficult to access.

Acknowledgements

I am very grateful to Thomas Eiter for valuable comments on an earlier version of this paper.

References

[1] E. W. Adams. *The Logic of Conditionals*, volume 86 of *Synthese Library*. D. Reidel Publishing Company, Dordrecht, Holland, 1975.

[2] S. Amarger, D. Dubois, and H. Prade. Constraint propagation with imprecise conditional probabilities. In *Proc. of the 7th Conference on Uncertainty in Artificial Intelligence*, pages 26–34. Morgan Kaufmann Publishers, 1991.

[3] K. A. Andersen and J. N. Hooker. Bayesian logic. *Decision Support Systems*, 11:191–210, 1994.

[4] F. Bacchus. *Representing and Reasoning with Probabilistic Knowledge: A Logical Approach to Probabilities.* The MIT Press, Cambridge, USA, 1990.

[5] R. Carnap. *Logical Foundations of Probability.* University of Chicago Press, Chicago, 1950.

[6] B. de Finetti. *Theory of Probability.* Wiley, New York, 1974.

[7] D. Dubois and H. Prade. On fuzzy syllogisms. *Computational Intelligence*, 4(2):171–179, 1988.

[8] D. Dubois, H. Prade, L. Godo, and R. L. de Màntaras. A symbolic approach to reasoning with linguistic quantifiers. In *Proc. of the 8th Conference on Uncertainty in Artificial Intelligence*, pages 74–82. Morgan Kaufmann Publishers, 1992.

[9] D. Dubois, H. Prade, and J.-M. Touscas. Inference with imprecise numerical quantifiers. In Z. W. Ras and M. Zemankova, editors, *Intelligent Systems*, chapter 3, pages 53–72. Ellis Horwood, 1990.

[10] R. Fagin, J. Y. Halpern, and N. Megiddo. A logic for reasoning about probabilities. *Information and Computation*, 87:78–128, 1990.

[11] A. M. Frisch and P. Haddawy. Anytime deduction for probabilistic logic. *Artificial Intelligence*, 69:93–122, 1994.

[12] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Freeman & Co., 1979.

[13] G. Georgakopoulos, D. Kavvadias, and C. H. Papadimitriou. Probabilistic satisfiability. *Journal of Complexity*, 4(1):1–11, 1988.

[14] J. Y. Halpern. An analysis of first-order logics of probability. *Artifical Intelligence*, 46:311–350, 1990.

[15] J. Heinsohn. Probabilistic description logics. In *Proc. of the 10th Conference on Uncertainty in Artificial Intelligence*, pages 311–318. Morgan Kaufmann Publishers, 1994.

[16] B. Jaumard, P. Hansen, and M. P. de Aragão. Column generation methods for probabilistic logic. *ORSA Journal of Computing*, 3:135–147, 1991.

[17] D. Kavvadias and C. H. Papadimitriou. A linear programming approach to reasoning about probabilities. *Annals of Mathematics and Artificial Intelligence*, 1:189–205, 1990.

[18] T. Lukasiewicz. Uncertain reasoning in concept lattices. In *Proc. of the 3rd European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty*, volume 946 of *LNCS/LNAI*, pages 293–300. Springer, 1995.

[19] T. Lukasiewicz. *Precision of Probabilistic Deduction under Taxonomic Knowledge*. Doctoral Dissertation, Universität Augsburg, 1996.

[20] T. Lukasiewicz. Efficient global probabilistic deduction from taxonomic and probabilistic knowledge-bases over conjunctive events. In *Proc. of the 6th International Conference on Information and Knowledge Management*, pages 75–82. ACM Press, 1997.

[21] T. Lukasiewicz, W. Kießling, G. Köstler, and U. Güntzer. Taxonomic and uncertain integrity constraints in object-oriented databases - the TOP approach. In *Proc. of the 4th International Conference on Information and Knowledge Management*, pages 241–249. ACM Press, 1995.

[22] C. Luo, C. Yu, J. Lobo, G. Wang, and T. Pham. Computation of best bounds of probabilities from uncertain data. *Comput. Intell.*, 12(4):541–566, 1996.

[23] N. J. Nilsson. Probabilistic logic. *Artificial Intelligence*, 28:71–88, 1986.

[24] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization, Algorithms and Complexity.* Prentice-Hall, Englewood Cliffs, New Jersey, 1982.

[25] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, San Mateo, California, 1988.

[26] A. Schrijver. *Theory of Linear and Integer Programming.* Wiley, New York, 1986.

[27] H. Thöne. *Precise Conclusion under Uncertainty and Incompleteness in Deductive Database Systems.* Doctoral Dissertation, Universität Tübingen, 1994.

[28] H. Thöne, U. Güntzer, and W. Kießling. Towards precision of probabilistic bounds propagation. In *Proc. of the 8th Conf. on Uncertainty in Artificial Intelligence*, pages 315–322. Morgan Kaufmann Publishers, 1992.