An Application of Disjunctive Logic Programming with Incomplete Information

Dietmar Seipel, Helmut Thöne
University of Tübingen
Sand 13, D – 72076 Tübingen, Germany
{seipel,thoene}@informatik.uni-tuebingen.de

Abstract

Despite their different perspectives, probabilistic reasoning and disjunctive logic programming strive for similar goals. Both address the problem of representing incomplete information and providing conclusions under reasonable assumptions such as independences or closed world assumptions.

This paper investigates the evaluation of independence assumptions which can be characterized by graphs (Markov and Bayesian networks). The underlying independence axioms are well–known – they correspond to a set of disjunctive clauses. Based on a compact representation of disjunctive clauses, called clause trees, we provide an efficient Δ–iteration technique with subsumption, which allows us to compute the least fixpoint of the program or alternatively the verification of particular independences.

Experimental results revealed that our approach is much more efficient than the conventional evaluation techniques.

Keywords: Disjunctive logic programming, incomplete information, probabilistic reasoning, fixpoint computation, subsumption.

1 Introduction

Reasoning about **incomplete information** is a major research topic in Artificial Intelligence with the goal to provide reasonable conclusions when knowledge is lacking. As expert systems have moved beyond definite problems to complex applications, an adequate treatment of incompleteness has become increasingly important. Flexible and reliable systems are required to give plausible explanations for derived results. This is, however, crucial when the deduction relies on some implicit assumptions about a real–world domain. For instance, some events may be regarded as independent unless we can establish a connection between these events, e.g. [ThöKiGü 94]. Recent research on probabilistic reasoning has brought a deep insight into **independence assumptions**, cf. [PeGeVe 89], even we know of some promising systems, cf. Hugin [AnOlJeJe 89] and System Z^+ [GoPe 92], which heavily rely on them.

Our concern here is to make the assumptions explicit when required e.g. for explanation purposes. In particular, we want to describe the evaluation of independence assumptions which are incompletely specified, i.e. we will consider disjunctions of independences. But instead of introducing different levels of independence, we prefer to provide a cautious approach of computing all independences which are a logical consequence of a given set of assumptions. By exploiting the advanced techniques of disjunctive logic programming we are trying to tackle the reasoning task.

In general, a **disjunctive, normal logic program**, cf. [LoMiRa 92], has rules with heads given by a disjunction of positive atoms and bodies given by a conjunction of positive and negated atoms:

$$\begin{array}{l} a_1(X_1) \vee \ldots \vee a_k(X_k) :- \\ \quad b_1(Y_1) \wedge \ldots \wedge b_m(Y_m) \wedge \\ \quad \neg b_{m+1}(Y_{m+1}) \wedge \ldots \wedge \neg b_{m+n}(Y_{m+n}). \end{array}$$

where $a_i(X_i)$ and $b_j(Y_j)$ are atoms with argument vectors X_i and Y_j, respectively. As usual, a rule is called definite horn clause, iff $k = 1$ and $n = 0$. In our case, we consider a disjunctive logic program P without negated body atoms (i.e. $n = 0$). The disjunctive consequence operator T_P^S applied to a set S of ground disjunctive facts – S is called a Herbrand state – searches for all disjunctive facts $B_1 \vee C_1$, …, $B_n \vee C_n$ in S that can be hyperresolved with a ground instance

$A_1 \vee \ldots \vee A_k$:- $B_1, \ldots, B_n$ of a rule $r \in P$, yielding the disjunctive fact $A_1 \vee \ldots \vee A_k \vee C_1 \vee \ldots \vee C_n$.

We use a bottom–up Δ–iteration approach that iteratively derives the set $T_P^S \uparrow \omega$ of disjunctive facts logically implied by the disjunctive program P. The intermediate Herbrand states are organized in a compact tree structure, called **clause trees**, cf. [Sei 93], that is used for a fast hyperresolution of further disjunctive facts by T_P^S. This will speed up the combinatorial search for resolvable disjunctive facts in a T_P^S–hyperresolution step. The necessary operations of eliminating clauses which are subsumed by subclauses (**syntactic subsumption**) during the Δ–iteration process can be performed efficiently based on the clause trees. Fast algorithms have been developed using certain heuristics for constructing compact tree representations.

Moreover, we can employ suitable heuristics defined by some further, **semantic subsumption** information, a powerful formalism introduced in [KöKiThöGü 93]: Δ–iteration combined with subsumption is sound and complete for operators which are monotonous w.r.t. the subsumption ordering. In our case, the semantic subsumption information is derived from the program P itself. All Herbrand states S are purified during the iteration by discarding subsumed disjunctive facts. This yields an enormous reduction of the intermediately derived Herbrand states.

This paper is organized as follows: In section 2, we describe the basic notions of disjunctive logic programming and discuss a Δ–iteration technique for evaluating disjunctive logic programs. Based on these techniques, we show how to evaluate some independence assumptions for a probabilistic network in section 3. We also exemplify the use of subsumption information by integrating it into a logic program. The paper ends with a short conclusion and ideas for further research.

2 Disjunctive Logic Programming

A well–known paradigm of recursive query processing is Δ–iteration or semi–naive iteration, cf. [BaRa 86], [GüKiBa 87]. Up to now it has been used mainly for the fixpoint computation of definite consequence operators. It can of course be combined with the tree–based evaluation of disjunctive consequence operators as well. For extending it to the fixpoint computation of a disjunctive logic program P, we need the Δ–version of the disjunctive consequence operator T_P^S.

Definition 2.1 *Let P be a disjunctive logic program and HB_P its Herbrand base.*

(i) *The* **disjunctive Herbrand base** DHB_P *of P is given by the set of positive ground clauses*

$$DHB_P = \{\ A_1 \vee \ldots \vee A_k \mid k \geq 1,\ A_i \in HB_P,\ 1 \leq i \leq k\ \}.$$

A **Herbrand state** *S of P is a subset $S \subseteq DHB_P$. The canonical form of S consists of*

$$can(S) = \{\ C \in S \mid \text{there is no proper subclause } C' \text{ of } C \text{ in } S\ \}.$$

The **set difference** *with subclause–subsumption of two Herbrand sets S_1 and S_2 is given by*

$$S_1 \setminus_s S_2 = \{\ C \in S_1 \mid \text{there is no subclause } C' \text{ of } C \text{ in } S_2\ \}.$$

(ii) *The disjunctive* **Δ–consequence operator** $T_P^S : 2^{\mathrm{DHB}_P} \times 2^{\mathrm{DHB}_P} \rightarrow 2^{\mathrm{DHB}_P}$ *is defined as*

$$T_P^S(S,\Delta S) = \{\ C \in DHB_P \mid C' \text{ :- } B_1, \ldots, B_n \text{ is a ground instance of a rule in } P,\ B_1 \vee C_1,\ \ldots,\ B_n \vee C_n \text{ are in } S \cup \Delta S, \text{ some } B_i \vee C_i \text{ is in } \Delta S, \text{ and } C \text{ is the smallest factor of } C' \vee C_1 \vee \ldots \vee C_n\ \}.$$

for two Herbrand states S and ΔS.

The algorithm Δ–ITERATION, cf. [Sei 93], as described in figure 1, formulates the disjunctive Δ–iteration based on the Δ–version of T_P^S. It transforms the Herbrand state S containing the given disjunctive facts to the least fixpoint of $T_{P \cup S}^S$. The basic operations of Δ–ITERATION are the following: Δ–**computation** w.r.t. subsumption is the computation of the Herbrand state Aux containing all derived disjunctive facts in $T_P^S(S, \Delta S)$ which are not subsumed by already known disjunctive facts in S. **Canonization** with the *can*–operator is the computation of all clauses in Aux which are not properly subsumed by other clauses – i.e. subclauses – in Aux. The time complexity of Δ–computation is $n_1 \cdot n_2$, where $n_1 = |T_P^S(S, \Delta S)|$, $n_2 = |S|$, and canonization takes time n^2, where $n = |Aux|$. Since n_1 is much larger than n, it is essential, that the canonization is applied *after* computing the set difference $\setminus_s$ with subclause–subsumption.

Algorithm Δ–ITERATION (P, S)

{ initialization }

$\Delta S := S$;

{ iteration }

while ($|\Delta S| > 0$)

$\quad Aux := T_P^S(S, \Delta S) \setminus_s S$;

$\quad \Delta S := can(Aux)$;

$\quad S := can(S \cup \Delta S)$;

end

Figure 1: Disjunctive Δ–ITERATION

3 Independence Assumptions in Probabilistic Networks

The independences embedded into probabilistic networks, such as MARKOV networks or BAYESIAN networks, cf. [Pe 88], can, in general, be identified by some graphical criterion. For instance, in a MARKOV network, separated vertices correspond to a genuine independence condition.[1] Let us formalize this criterion. (Identification of independences by vertex–separation.)

Definition 3.1 *Let $G = (V, E)$ be an undirected graph and X, Y, Z be disjoint subsets of vertices in G. Then X and Y are separated by Z, written $I(X, Z, Y)$, iff for every path from a vertex in X to a vertex in Y there is a vertex in Z on that path.*

The deduction problem can be stated a follows: Given an incomplete list of independences, specified by a disjunction of $I(X, Z, Y)$–atoms, we want to deduce the set of all independences which follow from graph separation. As shown in [Pe 88] (theorem 2, p. 94), the axioms in program P_1, cf. figure 2, are necessary and sufficient for vertex separation.

For instance, the transitivity axiom states that if X and Y are separated by Z, then for each vertex $N \in V \setminus (X \cup Y \cup Z)$ the following holds: X and N are separated by Z, or N and Y are separated by Z. Obviously, for a definite I–atom this axiom yields a disjunction of I–atoms. In general, given a set of independences for a graph, these axioms generate a set of derived independences with a cardinality that is exponential in the number of vertices in the graph. For instance, consider the independence $I(X, Z, Y)$ with $X = \{1\}$, $Y = \{2, \ldots, m\}$, $Z = \{m+1, \ldots, n\}$ and $2 < m < n$. By applying decomposition and strong union we get the set $\{ I(X, Z', Y') \mid Z \subseteq Z', Y' \subseteq Y \}$. However, to specify the derived set, it suffices to consider only the maximal elements w.r.t. the following partial **subsumption ordering**:

$$\begin{aligned} &subsumes(I(X_1, Z_1, Y_1), I(X_2, Z_2, Y_2)) \ :- \\ &\quad (X_2 \subseteq X_1, Z_1 \subseteq Z_2, Y_2 \subseteq Y_1) \ or \\ &\quad (X_2 \subseteq Y_1, Z_1 \subseteq Z_2, Y_2 \subseteq X_1). \end{aligned}$$

When we evaluate the disjunctive logic program P_1 together with the subsumption rule, then we may not get an equivalent fixpoint. To guarantee completeness, Δ–iteration with semantic subsumption must be applied to a modified, monotonous disjunctive logic program derived from P_1. The effect of discarding "superfluous" independences on the fly can be achieved by evaluating program P_2, as depicted in figure 3, with the above subsumption rule.

The new program allows for an efficient evaluation of the least fixpoint, which is, in general, not exponential in the number of vertices in the graph.

Theorem 3.2 (Program equivalence)
Program P_1 is equivalent to program P_2 w.r.t. the subsumption rule.

Sketch of the Proof:
Soundness of Intersection*: Suppose we have

$$I(X_1, Z_1, Y_1) \quad \text{and} \quad I(X_2, Z_2, Y_2). \qquad (1)$$

Let $Y_1' = Y_1 \setminus Y_2$ and $X = X_1 \cap X_2$. Then, given (1), symmetry and decomposition imply

$$I(X, Z_1, Y_1') \quad \text{and} \quad I(X, Z_2, Y_2). \qquad (2)$$

Now let $Y = Y_1' \cup Y_2$ and $Z = (Z_1 \cup Z_2) \setminus Y$. Since $Z \cup Y_2 = (Z_1 \cup Z_2) \setminus Y_1' \supseteq Z_1$ and $Z \cup Y_1' = (Z_1 \cup Z_2) \setminus Y_2 \supseteq Z_2$, we get from (2) by strong union:

$$I(X, Z \cup Y_2, Y_1') \quad \text{and} \quad I(X, Z \cup Y_1', Y_2). \quad (3)$$

Finally, given (3), intersection yields $I(X, Z, Y)$ which is equal to

$$I(X_1 \cap X_2, (Z_1 \cup Z_2) \setminus Y, Y). \qquad (4)$$

That is, the first rule in intersection* is sound. The soundness of the second and third rule in intersection* can be shown analogously.

[1] The graphical criterion for BAYESIAN networks is called d–separation. Due to space limitations the evaluation of independences there is omitted here.

Symmetry:	$I(X,Z,Y)$	:-	$I(Y,Z,X)$.
Decomposition:	$I(X,Z,Y)$	:-	$I(X,Z,Y\cup W)$.
Intersection:	$I(X,Z,Y\cup W)$	:-	$I(X,Z\cup W,Y), I(X,Z\cup Y,W)$.
Strong union:	$I(X,Z\cup W,Y)$	:-	$I(X,Z,Y)$.
Transitivity:	$I(X,Z,\{N\})\vee I(\{N\},Z,Y)$	:-	$I(X,Z,Y), N\in V\setminus(X\cup Y\cup Z)$.

Figure 2: Disjunctive logic program P_1

Intersection*:	$I(X_1\cap X_2,(Z_1\cup Z_2)\setminus Y,Y)$	:-	$I(X_1,Z_1,Y_1), I(X_2,Z_2,Y_2), Y=Y_1\cup Y_2$.
	$I(X_1\cap X_2,(Z_1\cup Z_2)\setminus Y,Y)$	:-	$I(X_1,Z_1,Y_1), I(Y_2,Z_2,X_2), Y=Y_1\cup Y_2$.
	$I(X_1\cap X_2,(Z_1\cup Z_2)\setminus Y,Y)$	:-	$I(Y_1,Z_1,X_1), I(Y_2,Z_2,X_2), Y=Y_1\cup Y_2$.
Transitivity:	$I(X,Z,\{N\})\vee I(\{N\},Z,Y)$	:-	$I(X,Z,Y), N\in V\setminus(X\cup Y\cup Z)$.

Figure 3: Disjunctive logic program P_2

Completeness w.r.t. symmetry, decomposition and strong union follows from the subsumption rule.

Completeness w.r.t. intersection: Given

$$J_1 = I(X,Z\cup W,Y) \text{ and } J_2 = I(X,Z\cup Y,W). \quad (5)$$

By induction assumption, there are

$$K_1 = I(X_1,Z_1,Y_1) \text{ and } K_2 = I(X_2,Z_2,Y_2) \quad (6)$$

such that $subsumes(K_1,J_1), subsumes(K_2,J_2)$. Since all cases are covered by intersection*, w.l.o.g., we can assume $X\subseteq X_1$, $X\subseteq X_2$, and $Z\cup W\supseteq Z_1$, $Z\cup Y\supseteq Z_2$, and $Y\subseteq Y_1$, $W\subseteq Y_2$. For $Y'=Y_1\cup Y_2$, we get $X\subseteq X_1\cap X_2$ and $(Z_1\cup Z_2)\setminus Y'\subseteq(Z\cup W\cup Y)\setminus(Y\cup W)=Z$ and $Y\cup W\subseteq Y'$. From (6), intersection* yields

$$K = I(X_1\cap X_2,(Z_1\cup Z_2)\setminus Y',Y'). \quad (7)$$

Now let

$$J = I(X,Z,Y\cup W). \quad (8)$$

Then $subsumes(K,J)$. Therefore, we can model intersection (which yields J from J_1 and J_2) by intersection* and subsumption.

Completeness w.r.t. transitivity: Given

$$J = I(X,Z,Y) \text{ and } N\in V\setminus(X\cup Y\cup Z). \quad (9)$$

By induction assumption, there is a

$$K = I(X',Z',Y') \quad (10)$$

such that $subsumes(K,J)$. If $N\in V\setminus(X'\cup Y'\cup Z')$ then transitivity applied to K yields a disjunction which subsumes the disjunction derived from J:

$$C = I(X,Z,\{N\})\vee I(\{N\},Z,Y). \quad (11)$$

If $N\notin V\setminus(X'\cup Y'\cup Z')$ then C can not be derived from K, but obviously C is subsumed by K. ■

Example 3.3 *Suppose that we have the Herbrand state*

$$S_0 = \{\ I(\{1\},\{2\},\{3,4\}),\ I(\{2\},\{1,3\},\{4\})\vee I(\{2\},\{1,4\},\{3\})\ \}.$$

The disjunctive Δ–iteration with program P_2 yields the following states:

$$S_1 = \{\ I(\{1\},\{2\},\{3,4\}),\ I(\{4\},\{3\},\{1,2\})\vee I(\{2\},\{1,4\},\{3\}),\ I(\{2\},\{1,3\},\{4\})\vee I(\{3\},\{4\},\{1,2\})\ \},$$
$$S_2 = \{\ I(\{1\},\{2\},\{3,4\}),\ I(\{4\},\{3\},\{1,2\})\vee I(\{3\},\{4\},\{1,2\})\ \}.$$

Thus, the initial Herbrand state S_0 is even simplified by the least fixpoint S_2 of $T^S_{P_2\cup S_0}$.

We have efficiently implemented the disjunctive consequence operator T^S_P based on a compact data structure of so–called clause trees for disjunctive facts. It speeds up the combinatorial search in a T^S_P–hyperresolution step. Furthermore, the necessary operations of subsumption elimination can be performed efficiently based on the clause trees.

Example 3.4 *Given the Herbrand state – a set $\{i_1,\ldots,i_k\}$ is denoted by the string $i_1\cdots i_k$ –*

$$S = \{\ I(35,14,2)\vee I(1,4,3)\vee I(25,4,1),\ I(35,14,2)\vee I(1,5,2)\vee I(1,2,345),\ \ldots,\ I(35,14,2)\vee I(1,5,2)\vee I(34,5,12),\ I(34,5,1)\vee I(1,5,2)\vee I(5,4,123),\ I(34,5,1)\vee I(2,14,35)\vee I(1,3,245),\ \ldots,\ I(34,5,1)\vee I(2,14,35)\vee I(2,5,134)\ \}.$$

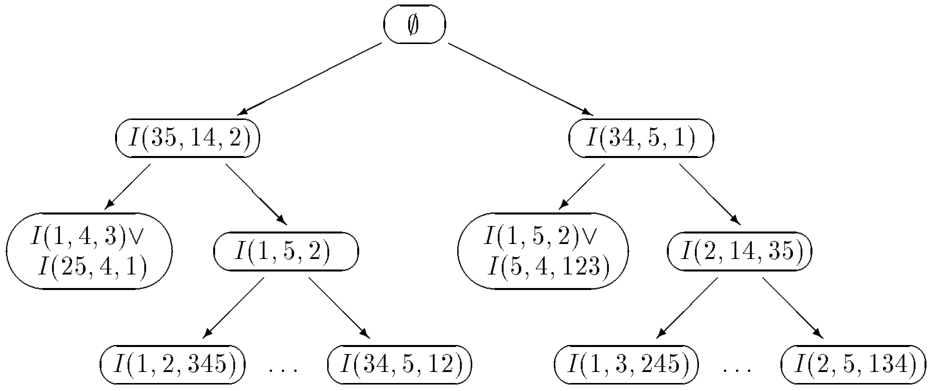

Figure 4: Tree representation

From each fact

$$I(35,14,2) \vee I(1,5,2) \vee C_i,$$

and each fact

$$I(34,5,1) \vee I(2,14,35) \vee C_j,$$

the $T^S_{P_2}$–operator derives a new fact

$$I(3,45,12) \vee I(1,5,2) \vee I(2,14,35) \vee C_i \vee C_j$$

using the same ground instance of the first Intersection rule, namely*

$$I(3,45,12) :- I(35,14,2), I(34,5,1).$$

As suggested, repeated application of the T^S_P–operator tends to rapidly increase the amount of intersections between derived disjunctive facts. Because of the special structure of the independence axioms the amount of intersections between the disjunctive facts is rapidly growing. Thus, a **tree representation** of Herbrand states turned out to be a suitable choice for supporting the deduction efficiently, cf. [SeiAr 93].

Definition 3.5 *Given a Herbrand state S. Let $\mathcal{A}_S = \cup_{C \in S} C$ denote the set of all atoms occurring in the clauses of S. A* **clause tree** *$\mathcal{T} = (V, E)$ with a node labeling $l : V \to 2^{\mathcal{A}_S}$* **represents** *$S$, iff*

(i) *each node $v \in V$ represents a clause $c(v)$ as follows: $c(v)$ is the union of the sets $l(v')$ which label the nodes v' on the path from the root to v,*

(ii) *the clauses $c(v)$ of the leaves $v \in V$ are in S, and every clause $C \in S$ is represented by exactly one leaf of the clause tree.*

Example 3.6 *For the transitive closure example 3.4 the clause tree $\mathcal{T}$ – given in figure 4 – representing the set S is unique. This usually is not the case. The node identifiers are omitted, only the labels are displayed.*

Based on a clause tree representation $\mathcal{T}$ of a Herbrand state S, the T^S_P–operator is evaluated in two steps. **Tree search** determines nodes in the tree, such that there is a ground instance of a rule from P whose body atoms are contained in the tree nodes. **Tree resolution** resolves the clauses for the leaf nodes in the subtrees of the determined nodes w.r.t the ground instance of the rule found in tree search. Interleaved with the T^S_P–operator the **tree update** can be performed by insertion of the portion of newly derived facts which are not subsumed by already known facts in the old clause tree.

By using the clause tree $\mathcal{T}$, it can be determined very efficiently if the represented intermediate Herbrand state S **subclause–subsumes** a given disjunctive fact: we essentially have to search for a complete path in the tree representation from the root to a leaf, such that all labeling clauses of the nodes on the path are subclauses of the investigated clause. Then, the disjunctive fact represented by the determined leaf is in S and is a subclause of the considered disjunctive fact.

4 Conclusion

We have compared Prolog–implementations of the basic version of Δ–Iteration and a tree–based version using clause trees for resolution and subsumption. We achieved very encouraging results for the independence example, even without incorporating all possible optimization techniques. E.g. tree update was done by inserting a clause tree for the derived new facts at the root of the old clause tree, instead of inserting the new facts into the old tree. For Herbrand states with many overlapping facts the tree–based version of Δ–ITERATION is very efficient and superior to the basic version.

The intermediately derived Herbrand states S are always purified by removing disjunctive facts, which are syntactically or semantically **subsumed** by other disjunctive facts. This yields an enormous reduction of the Herbrand states. But, to guarantee the completeness of the evaluation procedure, the disjunctive logic program has to be transformed to an equivalent disjunctive logic program which is **monotonous** w.r.t. the semantic subsumption rule.

The presented techniques can be used for computing the extended generalized closed world assumption of a disjunctive logic program and the generalized disjunctive well–founded semantics, cf. [LoMiRa 92], of a disjunctive normal logic program as well. Thus, even with negative information, we can jump to reasonable conclusions. Details will be presented in a forthcoming paper.

References

[AnOlJeJe 89] *S.K. Andersen, K.G. Olesen, F.V. Jensen, F. Jensen*: HUGIN – A Shell for Building BAYESIAN Belief Universes for Expert Systems, Proc. Intl. Joint Conf. on Artificial Intelligence 1989, pp. 1080-1086.

[BaRa 86] *F. Bancilhon, R. Ramakrishnan:* An Amateur's Introduction to Recursive Query Processing Strategies, Proc. ACM SIGMOD 1986, pp. 16-52.

[CeGoTa 90] *S. Ceri, G. Gottlob, L. Tanca:* Logic Programming and Databases, Springer, 1990.

[CheWa 92] *W. Chen, D.S. Warren:* A Goal–Oriented Approach to Computing Well–Founded Semantics, Proc. Joint Intl. Conf. and Symp. on Logic Programming 1992, pp. 589-603.

[CheWa 93] *W. Chen, D.S. Warren:* Query Evaluation under the Well–Founded Semantics, Proc. ACM PODS 1993.

[Da 92] *S.K. Das:* Deductive Databases and Logic Programming, Addison–Wesley, 1992.

[GoPe 92] *M. Goldszmidt, J. Pearl:* Rank–Based Systems: A Simple Approach to Belief Revision, Belief Update, and Reasoning about Evidence and Actions, Proc. Intl. Conf. on Principles of Knowledge Representation and Reasoning 1992, Cambridge.

[GüKiBa 87] *U. Güntzer, W. Kießling, R. Bayer:* On the Evaluation of Recursion in (Deductive) Database Systems by Efficient Differential Fixpoint Iteration, Proc. Intl. Conf. on Data Engineering 1987, pp. 120-129.

[KöKiThöGü 93] *G. Köstler, W. Kießling, H. Thöne, U. Güntzer:* The Differential Fixpoint Operator with Subsumption, Proc. Intl. Conf. on Deductive and Object–Oriented Databases 1993.

[Llo 87] *J.W. Lloyd:* Foundations of Logic Programming, Springer, second edition, 1987.

[LoMiRa 92] *J. Lobo, J. Minker, A. Rajasekar:* Foundations of Disjunctive Logic Programming, MIT Press, 1992.

[Pe 88] *J. Pearl:* Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, Morgan Kaufman, 1988.

[PeGeVe 89] *J. Pearl, D. Geiger, T. Verma*: Conditional Independence and its Representations, Kybernetika, vol. 25(2), 1989.

[SeiAr 93] *D. Seipel, H. Argenton*: Evaluation Techniques for Disjunctive Logic Programs, Proc. Symposium on Operations Research 1993.

[Sei 93] *D. Seipel*: Tree–Based Fixpoint Iteration for Disjunctive Logic Programs, Proc. Workshop on Logic Programming with Incomplete Information at the Intl. Symp. on Logic Programming 1993, pp. 177-189.

[ThöGüKi 92] *H. Thöne, U. Güntzer, W. Kießling*: Towards Precision of Probabilistic Bounds Propagation, Conference on Uncertainty in Artificial Intelligence 1992, Stanford, pp. 315-322.

[ThöKiGü 94] *H. Thöne, W. Kießling, U. Güntzer*: On Cautious Probabilistic Inference and Default Detachment, to appear: Annals of Operations Research, 1994.

[Ul 88,89] *J.D. Ullman*: Principles of Database and Knowledge–Base Systems, Volume I,II, Computer Science Press, 1988/89.

[VaGeRoSchli 91] *A. Van Gelder, K.A. Ross, J.S. Schlipf*: The Well–Founded Semantics for General Logic Programs, JACM, vol. 38(3), 1991, pp. 620-650.