On Belief Dynamics of Dependency Relations for Extended Logic Programs

Patrick Krümpelmann and **Gabriele Kern-Isberner**
Information Engineering Group
Technische Universität Dortmund
Germany

Abstract

Belief operations for non-monotonic formalisms like extended logic programs do not fit into the classic theory of AGM belief change. In this work we make use of dependency theories as abstractions of extended logic programs leading to a monotonic representation. We adapt the basic AGM postulates of revision and contraction for dependency relations in order to obtain a clearer view on the belief change problem for logic programs. We present revision and contraction operations on dependency relations satisfying the proposed postulates. Finally, we show how AGM compliant belief change operations for extended logic programs can be derived.

Introduction

If we want agents to perform intelligently in a dynamic environment we have to enable them to revise their beliefs. The current beliefs have to be modified, new beliefs have to be acquired, old beliefs have to be revised or given up. This is what the theory of belief revision is about; it formalizes changes and operators for these. Properties of these belief change operators have been described thoroughly in literature, with AGM theory (Alchourron, Gardenfors, and Makinson 1985) being the most seminal approach.

In this work we consider dynamic belief bases that are represented by non-monotonic theories and in particular by extended logic programs. We refer to extended logic programs whenever talking about logic programs in the following. Logic programming is intensively used for knowledge representation and proved to play an important role for the development of intelligent systems and advanced reasoning tasks in those (Gelfond and Lifschitz 1988). For the change of belief bases represented by extended logic programs several approaches have been proposed of which the majority proved to be not compliant with the AGM belief change theory due to the non-monotonic nature of the formalism as shown in (Eiter et al. 2002).

In this paper, we define a general framework of dependency relations for extended logic programs in the spirit of (Bondarenko, Toni, and Kowalski 1993) and (Sefranek 2006). Based on this abstraction we investigate the compliance with classic belief revision theory and reformulate the AGM postulates for revision on dependency relations. We transfer results on the connection between contraction and revision operations in classic belief revision theory to dependency relations and define postulates for different types of contractions which can be used for the definition of revision operations. Furthermore, we adapt the notions of kernel-contraction (Hansson 1994) to our framework and show the satisfaction of the proposed postulates by our operators as well as the connection of those to revision operators. Additionally, we show how the operators defined in this paper can be used in order to define new belief operations on logic programs that extend the capabilities of other approaches.

The paper is structured as follows. First, we give preliminaries on belief revision and logic programming. Then we introduce a dependency theory as the basis of our work. Then follows an investigation and adaption of classic belief revision to dependency relations and the use for logic programs. We close with a discussion of related work and ideas for future work.

Preliminaries

In the classic setting of belief change as described by (Alchourron, Gardenfors, and Makinson 1985) beliefs are formulated in form of sentences from a logical language $\mathcal{L}$ which is closed under some boolean connectives. A *belief set* K is a subset of the language that is closed under a consequence operator $Cn(\cdot)$ such that $Cn(K) = K$. Because of the infinite size of belief sets a finite representation is often desirable and given in form of a belief base. A *belief base* B_K for some belief set K is a finite set of sentences such that $Cn(B_K) = K$ holds. If new information is acquired it has to be incorporated into the current set of beliefs. Given a representation as a belief set K and new information being represented as a sentence ϕ of the language $\mathcal{L}$, the operation of adding ϕ to the current beliefs is called *expansion* and denoted by $K + \phi$. An expansion can be performed without problems if the new information ϕ is consistent with K. The expansion is uniquely determined as $K + \phi = Cn(K \cup \phi)$ given the six AGM postulates for expansion. In the case of an inconsistency of K and ϕ, conflicts arising from the addition of ϕ to the current set of beliefs have to be resolved which amounts to a revision of the beliefs. This means that some of the current beliefs have to be given up in order to come to a consistent belief set. The AGM model gives six basic postulates a revision operator $*$ should obey:

$(K*1)$ $K * \phi$ is a belief set.

$(K*2)$ $\phi \in K * \phi$.

$(K*3)$ $K * \phi \subseteq K + \phi$.

$(K*4)$ If $\neg\phi \notin K$, then $K + \phi = K * \phi$.

$(K*5)$ $K * \phi$ is inconsistent *iff* ϕ is inconsistent.

$(K*6)$ If $\phi \equiv \psi$, then $K * \phi = K * \psi$.

An extended logic program (Gelfond and Lifschitz 1988) consists of rules over a set of propositional atoms $\mathcal{A}$ using strong negation $\neg$ and default negation 'not'. A literal L can be an atom A or a negated atom $\neg A$. The complement of a literal L is denoted by $\neg L$ and is A if $L = \neg A$ and $\neg A$ if $L = A$. Let $\mathcal{A}$ be the set of all atoms and Lit the set of all literals $Lit = \mathcal{A} \cup \{\neg A \mid A \in \mathcal{A}\}$. For $X \subseteq Lit$ we define not $X = \{\text{not } L \mid L \in X\}$ and denote the set of all default negated literals by $\mathcal{D} = \text{not } Lit$. $\mathcal{L} = Lit \cup \mathcal{D}$ represents the set of all literals and default negated literals. An *assumption* is a default negated literal. A rule r is written as

$$L \leftarrow L_0, \ldots, L_m, \text{not } L_{m+1}, \ldots, \text{not } L_n.$$

where the head of the rule $H(r) = L$ is either empty or consists of a single literal and the body $\mathcal{B}(r) = \{L_0, \ldots, L_m, \text{not } L_{m+1}, \ldots, \text{not } L_n\}$ is a subset of $\mathcal{L}$. The body consists of a set of literals $\mathcal{B}(r)^+ = \{L_0, \ldots, L_m\}$ and a set of default negated literals not $\mathcal{B}(r)^-$ with $\mathcal{B}(r)^- = \{L_{m+1}, \ldots, L_n\}$. Given this we can write a rule as

$$H(r) \leftarrow \mathcal{B}(r)^+, \text{not } \mathcal{B}(r)^-.$$

If $\mathcal{B}(r) = \emptyset$ we call r a fact. A set of literals which is consistent, i. e., it does not contain complementary literals L and $\neg L$, is called a state I. A literal L is true in I iff $L \in I$ and false otherwise. The body $\mathcal{B}(r)$ of a given rule r is true in I iff each $L \in \mathcal{B}(r)^+$ is true in I and each $L \in \mathcal{B}(r)^-$ is false in I. A rule r is true in I iff $H(r)$ is true in I whenever $\mathcal{B}(r)$ is true in I. A state I is a model of a program P if r is true in I for all $r \in P$. The reduct P^S of a program P relative to a set S of literals is defined as:

$$P^S = \{H(r) \leftarrow \mathcal{B}^+(r) \mid r \in P, \mathcal{B}^-(r) \cap S = \emptyset\}.$$

An answer set of a program P is a state I which is a minimal model of P^I.

Dependency theory

In the following we develop a general dependency theory for non-monotonic formalisms in general and in particular for extended logic programs in the spirit of (Bondarenko, Toni, and Kowalski 1993) and (Sefranek 2006). Here, we focus on the representation of dependencies in extended logic programs and define semantics based on this representation.

Non-monotonic formalisms are characterized by their use of assumptions in what is called a default rule. In extended logic programs these are given in the form of the negative part of the body of a rule. A literal can depend on default negated literals as well as on not default negated literals in the first instance. We elaborate on the differences later on and begin with a general definition of a dependency relation.

Definition 1 (Dependency relation)**.** A *dependency* d is a tuple of the form $d = (L, \mathcal{W})$, where the *dependant* $L \in Lit$ depends on the *premise* $\mathcal{W} \subseteq \mathcal{L}$ of the dependency d. A *dependency relation* R is a set of dependencies. The set of all dependencies, constructible as defined above based on the set of propositional atoms $\mathcal{A}$ is denoted by $\mathcal{R}_\mathcal{A}$.

The dependency of a literal on some, possibly default negated, literals means that the latter have to be known or assumed in order to infer the former. The rules of a logic program resemble basic dependencies on whose basis further dependencies can be entailed. In order to define the entailment of dependencies we facilitate the notion of dependency sequences which are chains of sequential applicable dependencies, as specified in the following definition.

Definition 2 (Entailment relation $\vdash_\sigma$)**.** Let $R \subseteq \mathcal{R}_\mathcal{A}$ be a dependency relation, and $D \subseteq R$ a set of dependencies in R; let $d = (L, \mathcal{W}) \in \mathcal{R}_\mathcal{A}$ be a dependency. D *entails* d, in symbols $D \vdash_\sigma d$, iff there exists a dependency sequence $\sigma = (d_1, \ldots, d_n)$, $n \geq 1$ of elements $d_i = (L_i, \mathcal{W}_i) \in D, 1 \leq i \leq n$, which satisfies the following conditions:

(i) $d_n = (L, \mathcal{W}_n)$

(ii) $\mathcal{W}_1 \subseteq \mathcal{W} \subseteq \mathcal{D}$

(iii) For each $i, 1 \leq i < n : \mathcal{W}_{i+1} \subseteq \mathcal{W} \cup \{L_1, \ldots, L_i\}$

By use of the entailment operator, new dependencies are entailed which result from chaining dependencies such that all non default negated literals of premises occurring in the sequence are only dependent on a, possibly empty, set of assumptions mitigated by the sequence. This behavior is expressed by condition (ii) which states that the premise of the generated dependency is a set of assumptions and condition (iii) which expresses that all non-assumptions occurring in some premise of some dependency in the sequence need to be dependent solely on assumptions by means of some subsequence of the dependency sequence. The premise of the generated dependency can contain any subset of the set of default negated literals as expressed in (ii). This results in the entailment of all dependencies that are constructible using the closure of the premise by consistent assumptions.

Based on the entailment operator for dependencies we define a consequence operator.

Definition 3 (Consequence operator on dependencies)**.**

$$Cn_\sigma(R) = R \cup \{d \mid R \vdash_\sigma d\}$$

This consequence operator on dependencies satisfies the Tarskian axioms for consequence operators, namely iteration $Cn_\sigma(Cn_\sigma(A)) = Cn_\sigma(A)$, inclusion $A \subseteq Cn_\sigma(A)$ and monotony $A \subseteq B$ implies $Cn_\sigma(A) \subseteq Cn_\sigma(B)$.

Given an extended logic program, a dependency relation from this is defined by means of the consequence operator.

Definition 4. For an extended logic program P we define dependencies for rules. The dependency relation for P is then defined as:

$$R_P = Cn_\sigma(\{(H(r), \mathcal{B}(r)) \mid r \in P\})$$

R_P is closed under Cn_σ and in the remainder of this paper we assume a given dependency relation R to be closed under Cn_σ in general.

Example 5.

$$P = \{ \quad A \leftarrow \text{not } B. \quad B \leftarrow \text{not } A. \\ C \leftarrow A. \qquad D \leftarrow B. \}$$

Given this program the following dependency relation is generated for it: $R_P = \{(A, \{\text{not } B\}), (B, \{\text{not } A\}), (C, \{A\}), (D, \{B\}), (C, \{\text{not } B\}), (D, \{\text{not } A\}), (C, \text{not } \{B, D\}), (D, \text{not } \{A, C\}), (D, \text{not } \{A, C, D\}), \dots\}$

Alternative models for a default theory are generated by different sets of assumptions which satisfy certain properties. The dependencies that represent the dependence of a literal on a set of assumptions have a key function as the dependants of these are prone to be part of a model of the theory. These models can therefore be characterized using sets of assumptions which generate the models of the theory. To characterize models of the underlying theory we introduce an operator to project dependencies onto literals with respect to a set of assumptions.

Definition 6 (Projection operator)**.** Given a dependency relation R and a set of assumptions $\Delta \subseteq \mathcal{D}$ we define the projection of R onto the set of literals Lit by:

$$J_R(\Delta) = \{L \mid (L, \Delta) \in R\}$$

Sets of assumptions have to satisfy some conditions in order to be considered a valid characterization of the model of the theory. These are, generally speaking, consistency and maximality of the projection and are specified in the following definition. Later on, we show that the projection of a dependency relation given a constrained set of assumptions corresponds to an answer set of an extended logic program.

Definition 7 (Valid sets of assumptions)**.** A set of assumptions Δ is called a *valid set of assumptions* with respect to a dependency relation R *iff*

(i) for all $L \in Lit$: $L \in J_R(\Delta)$ or not $L \in \Delta$ but not both

and there exists no set of assumptions Δ' with $J_R(\Delta) \subset J_R(\Delta')$ satisfying (i). The set of valid sets of assumptions with respect to R is denoted by $\mathcal{T}_R$.

As an example for valid sets of assumptions consider the following program.

Example 8. Considering Example 5, we get the following valid sets of assumptions: $\Delta_1 = \text{not } \{B, D\}$ and $\Delta_2 = \text{not } \{A, C\}$. The according sets of consequences are $J_{R_P}(\Delta_1) = \{A, C\}$ and $J_{R_P}(\Delta_2) = \{B, D\}$ which are also the only two answer sets of this program.

Definition 9 (Dependency semantics)**.** The semantics of a dependency relation R is given by its set of valid sets of assumptions $\mathcal{T}_R$. The set of extensions $\mathcal{E}_R$ of a dependency relation is given as $\mathcal{E}_R = \{J_R(\Delta) \mid \Delta \in \mathcal{T}_R, J_R(\Delta) \text{ consistent}\}$.

We can show that the defined dependency semantics for extended logic programs under the answer set semantics is sound and complete as stated in the following theorem.

Theorem 10. Given an extended logic program P and the corresponding dependency relation R_P for P. A set of literals $S \subseteq Lit$ is an extension of R_P, $\mathcal{E}_R$, iff S is an answer set of P.

Proofs of all theorems are omitted here due to space restrictions.

AGM Revision of Dependency Relations

In the previous section, we introduced a logical framework consisting of dependency relations and appertaining consequence operators that can be used to model default reasoning. We will now turn to investigate belief operations in this framework, In particular, we will focus on AGM-like belief revision operators for dependency relations. In the following, we will adapt the basic AGM-Postulates (Alchourron, Gardenfors, and Makinson 1985) as far as possible (or reasonable) to the revision of dependencies.

When incorporating new information, the AGM theory distinguishes between belief change operations that are able to resolve conflicts and inconsistencies, and those that simply expand the prior belief set. The latter ones are usually called expansions and are realized by the union operator on sets.

Definition 11 (Expansion)**.** Given a dependency relation R and a single dependency d, then the *expansion* of R by d is defined as $R + d = Cn_\sigma(R \cup \{d\})$.

For the proper revision of a dependency relation R by a new dependency d gives rise to some new dependency relation $R * d$ which is supposed to be closed under the application of Cn_σ:

$(D * 1)$ $Cn_\sigma(R * d) = R * d$

So, closed dependency relations (with respect to Cn_σ) play the role of belief sets in this scenario. Secondly, the belief revision is successful iff d is part of the revised belief set:

$(D * 2)$ $d \in R * d$

The expansion of dependency relations is defined analogously to the classic case, and also for dependencies, we expect expansions to be an upper bound for revisions:

$(D * 3)$ $R * d \subseteq R + d$

The interesting cases of the AGM-Postulates considering dependency relations regard the definition of consistency. The fourth AGM postulate states that the expansion of K by some information α shall be contained of the revision by this information if it is consistent with the belief-set.

$(K * 4)$ if $\neg\alpha \notin K$, then $K + \alpha \subseteq K * \alpha$.

The first part of the postulate $(\neg\alpha \notin K)$ expresses the condition for the consistency of the new information with the set of beliefs. Within the framework of dependencies the complement of a dependency can be defined for a dependency $d = (L, \mathcal{W})$ as $\neg d = (\neg L, \mathcal{W})$ which leads to the definition of conflicts in dependency relations.

Definition 12 (Conflict)**.** Given a dependency relation R, it contains a *conflict* $C \subseteq R$ *iff* for some $L \in Lit$ and some $\Delta \subseteq \mathcal{D}$ it is the case that $C = \{(L, \Delta), (\neg L, \Delta)\} \subseteq R$ and not $\{L, \neg L\} \cap \Delta = \emptyset$. A dependency relation containing conflicts is called *inconsistent*, and *consistent* otherwise. Let $\mathcal{C}_R$ denote the set of all conflicts in R.

Example 13.

$$P = \{ \quad C \leftarrow \text{not } A., \quad B \leftarrow C. \\ \neg B \leftarrow \text{not } A. \quad \}$$

Let R_P be the dependency relation generated by the program P. The only conflict in R_P is $C = \{(B, \text{not } \{A, \neg C, \neg A\}), (\neg B, \text{not } \{A, \neg C, \neg A\})\}$.

Where in classic logics the deductive closure and the classic negation are used to test for conflicts we need to make use of the conflict definition and semantics of dependency relations here. A naive reformulation using the negation of dependencies would be

if $\{d, \neg d\}$ is not a conflict in $Cn_\sigma(R{+}d)$ then $R{+}d \subseteq R{*}d$.

But this does not serve as a sufficient condition as it does not exclude the existence of conflicts in $R + d$ as the following example illustrates.

Example 14. Consider the revision of $R = Cn_\sigma(\{(b, \{a\}), (\neg b, \emptyset)\})$ by $d = (\neg a, \emptyset)$. Here, $\{d, \neg d\}$ *is a conflict in* $Cn_\sigma(R + d)$ does not hold but in spite of that $R + d$ is not conflict free.

In classic logic the consequence operator is more powerful and by use of *contraposition* would lead to the violation of the consistency condition if testing $a \in Cn(a \Rightarrow b, \neg b)$. The appropriate reformulation for dependency relations is therefore

$(D * 4)$ if $R + d$ is consistent then $R + d \subseteq R * d$

The revision of a dependency relation R by some new dependency d should yield a consistent, i. e., conflict free, dependency relation $R' = R * d$ according to the fifth AGM postulate

$(K * 5)$ $K * \alpha$ is inconsistent iff α is inconsistent

Note, that this AGM postulate makes an exception for inconsistent new information. Since we revise by a single dependency which, by Definition 12, cannot be inconsistent we drop this condition demanding that:

$(D * 5)$ $R * d$ is consistent

In summary, the reformulation of the basic AGM-Postulates for revisions of dependency relations yields the following postulates:

$(D * 1)$ $Cn_\sigma(R * d) = R * d$

$(D * 2)$ $d \in R * d$

$(D * 3)$ $R * d \subseteq R + d$

$(D * 4)$ if $R + d$ is consistent then $R + d \subseteq R * d$

$(D * 5)$ $R * d$ is consistent

In the following sections, we will present revision operators that, by and large, comply with these lines of thoughts.

Revision via consolidation

In this section we transfer results from classic belief revision in order to define revision operations by means of consolidation operations. Classic belief revision theory gives means to define revision operations by means of contraction operations via the Levi-Identity $K{*}\alpha = (K{-}\neg\alpha){+}\alpha$. Informally this can be understood as the instruction to first make sure that no information that entails conflicts with α is contained in the belief set. Since K is deductively closed the contraction by $\neg\alpha$ guarantees that the expansion by α will not introduce any conflicts. This is the case since the consequence operator in classic belief revision includes contraposition.

The contraction by the negation of the information to be revised by is not sufficient to grant consistency for consequence operators not satisfying contraposition as elaborated above. For the application of the Levi-Identity to dependency relations we need to make sure that the addition of the information to be incorporated in our beliefs does not introduce any conflicts. In general we have to contract by conflicts, commonly denoted by $\perp$, introduced by the new piece of information.

Our definition for the consistency condition given above is inherently external[1]. That is, in order to determine consistency of a dependency relation R with some dependency d it is not sufficient to consider the original dependency relation R, but we have to consider the extended set $R + d$. Based on this, we need to facilitate a global contraction operator in order to contract by inconsistency. The use of external revision leads to the formulation of the belief revision operator via the reversed Levi-Identity, $K * \alpha = (K + \alpha) - \neg\alpha$. For dependency relations we reformulate this identity as

$$R * d = (R + d){-}_{\mathcal{C}} \perp$$

in general. The contraction by inconsistency is also called a *consolidation operation*, leading to the composition of non-prioritized belief revision operators (Hansson 2001) as we discuss later on.

It should be noted that another striking difference between the closure operator on dependencies and the classical deductive closure becomes evident here. The definition of an external revision operation on classic belief-sets does not make sense due to the identity of all inconsistent belief-sets which leads to the loss of all prior information and to the equality of all revisions. The closure of an inconsistent dependency relation is finite and in general different from the closure of other inconsistent dependency relations. Therefore the definition of external revision for closed dependency relations makes sense.

The formulation of the revision of dependency relations via the reversed Levi-identity leaves us with the need to define a global contraction operator on inconsistency. As a first step we formalize the notion of inconsistency as denoted by $\perp$ in general for dependency relations as the set of conflicts of the dependency relation $\mathcal{C}$. Based on this we define a revision operator for dependency relations using a contraction operation.

Definition 15 (Revision by dependencies)**.** Given a closed dependency relation R and a dependency d we define the revision of R by d as $R * d = Cn_\sigma(R \cup \{d\}) \; {-}_{\mathcal{C}} \; \mathcal{C}_{R+d}$ with $\mathcal{C}_{R+d} = \{C \mid C \text{ is a conflict in } R + d\}$. We call the contraction operation ${-}_{\mathcal{C}}$ a $\mathcal{C}$*-contraction*.

[1]Here and in the following we adopt the terminology used in (Hansson 2001).

This definition gives a formalization of the previous discussion and transfers the problem to the definition of an appropriate contraction operator for sets of conflicts. The result of this operation shall be a subset of R, denoted by R', that is consistent, or more precisely, that does not contain any conflict.

For the further specification of the contraction operation we are going to adapt the basic AGM-postulates for contraction for our setting of conflict driven consolidation of dependency relations as follows:

($D -_{\mathcal{C}} 1$) $Cn_\sigma(R -_{\mathcal{C}} \mathcal{C}_R) = R -_{\mathcal{C}} \mathcal{C}_R$

($D -_{\mathcal{C}} 2$) $R -_{\mathcal{C}} \mathcal{C}_R \subseteq R$

($D -_{\mathcal{C}} 3$) if $\mathcal{C}_R = \emptyset$ then $R -_{\mathcal{C}} \mathcal{C}_R = R$

($D -_{\mathcal{C}} 4$) $\mathcal{C}_{R-_{\mathcal{C}}\mathcal{C}_R} = \emptyset$

($D -_{\mathcal{C}} 5$) $R \subseteq (R -_{\mathcal{C}} \mathcal{C}_R) + \mathcal{C}_R$

The first postulate states, that the result of a contraction should be closed under the consequence operator for dependencies. The second demands that no information is added to the dependency relation by the contraction operation. The third postulate requires the contraction by the empty set of conflicts to be equivalent to the original dependency relation. The fourth one expresses that the result of the contraction by a set of conflicts should be successful in the sense that all conflicts are not contained in the resulting dependency relation. The fifth property requires the result of the expansion of a contracted dependency relation by the contracted set to recover the original dependency relation. This recovery postulate has been heavily discussed in the past and is not desirable in many settings (Hansson 2001). The contraction operations defined in this paper also do not satisfy recovery.

For the connection of the postulates for revision and contraction we get the following relation via the reversed Levi-identity.

Theorem 16. Let $-_{\mathcal{C}}$ be a $\mathcal{C}$-contraction operator on dependency relations. If $-_{\mathcal{C}}$ satisfies the contraction postulates $D -_{\mathcal{C}} 1$, $D -_{\mathcal{C}} 2$, $D -_{\mathcal{C}} 3$ and $D -_{\mathcal{C}} 4$ then the revision operator $*$ based on $-_{\mathcal{C}}$ by Definition 15 satisfies the revision postulates $D * 1$, $D * 3$, $D * 4$ and $D * 5$.

Given the set of postulates for contraction operators by sets of conflicts in dependency relations we continue by describing the construction of such an operator. We define the contraction by a set of conflicts generally as follows.

Definition 17 (Contraction by sets of conflicts)**.** Let R be a dependency relation and $\mathcal{C} = \{C_1, \ldots, C_n\}$, $C_i \subseteq R$ $1 \leq i \leq n$ a set of conflicts. A set of dependencies $\mathcal{I} \subseteq R$ is called an incision set of a set of conflicts $\mathcal{C}$ *iff*

$$C_i \not\subseteq Cn_\sigma(R \setminus \mathcal{I}) \text{ for all } C_i \in \mathcal{C}$$

and no $\mathcal{I}'$ with $\mathcal{I}' \subset \mathcal{I}$ exists.

We define a contraction by sets of conflicts $-_{\mathcal{C}}$ as $R -_{\mathcal{C}} \mathcal{C} = Cn_\sigma(R \setminus \mathcal{I})$ with $\mathcal{I}$ being an incision set for $\mathcal{C}$.

An incision set is basically a set of dependencies such that each conflict persistent in the dependency relation is not existent in the dependency relation without the incision set and will not be reinstated by the consequence operator.

Theorem 18. Let $-_{\mathcal{C}}$ be a $\mathcal{C}$-contraction operator on dependency relations. Then $-_{\mathcal{C}}$ satisfies the postulates for $\mathcal{C}$-contraction $D -_{\mathcal{C}} 1$, $D -_{\mathcal{C}} 2$, $D -_{\mathcal{C}} 3$ and $D -_{\mathcal{C}} 4$.

Revision on base dependencies

As discussed earlier, revision on dependency relations closed under the consequence operator $Cn_\sigma(\cdot)$ differs from classic belief revision on deductively closed belief sets due to the different nature of the consequence operator on dependency relations. In the following, we develop a belief revision operator on a base representation of the dependency relation which corresponds to the use of belief bases and base revision in terms of classic belief change (Hansson 2001). Here, we will solve conflicts based on these dependencies using a base representation of the dependency relation which also has the advantage of being more concise. We will introduce a base representation of dependency relations in the following which basically reduces a dependency relation to the generating elements, i. e., the dependencies directly corresponding to rules.

Definition 19 (Base Dependency)**.** A dependency $d \in R$ is called a base dependency of the dependency relation R *iff*

$$Cn_\sigma(R \setminus d) \neq Cn_\sigma(R).$$

A base relation R^b for a dependency relation R is a set of base dependencies such that $Cn_\sigma(R^b) = Cn_\sigma(R)$. A dependency sequence entirely consisting of base dependencies is called a base sequence $(d_1, \ldots, d_n)$, $d_i \in R^b$, $1 \leq, i, \leq n$.

As an example for the base representation of dependency relations consider the following program and the generated dependency relation.

Example 20.

$$P = \{B. \quad \neg A. \quad A \leftarrow B.\}$$

The dependency relation generated by this program R_P is represented by the following base dependencies.

$$R_P^b = \{(B, \emptyset), (\neg A, \emptyset), (A, \{B\})\}$$

These three dependencies clearly correspond to the three rules in the program. In the closure of these the transitivity leads to the dependency $(A, \emptyset)$. The premise closure would add the following dependencies: $(B, \{\text{not } A\})$, $(B, \{\text{not } \neg A\})$, $(B, \{\text{not } A, \text{not } \neg A\})$, ...

Next, we define incisions which, in contrast to the incision set defined before, address a single conflict. Subsequently this will be rephrased in terms of a base representation.

Definition 21 (Incision I)**.** Let R be a dependency relation and C a conflict. A set of dependencies $I \subseteq R$ is called an incision of a conflict C *iff*

$$C \not\subseteq Cn_\sigma(R \setminus I)$$

and no I' satisfying the property above with $I' \subset I$ exists.

Base dependencies d^b are atomic in that they can be entailed only in a trivial way, i. e., via $d^b \vdash_\sigma d^b$. These are invalidated, i. e., removed, by simply removing d from $R_{\mathcal{P}}$. In

order to invalidate a non-atomic sequence σ one of the dependencies $d' \in \sigma$ has to be invalidated in $R_{\mathcal{P}}$. This underlies the next definition of base incisions for conflicts as every dependency can be invalidated by invalidating base dependencies in the end.

Definition 22 (Base incision)**.** A set of base dependencies I^b is called a *base incision* of a conflict C *iff*

$$C \not\subseteq Cn_\sigma\left(R^b \setminus I^b\right)$$

Kernel contraction

On the level of a base representation of dependencies we can draw some similarities between kernel-contraction operations (Hansson 1994) and the just defined base-contraction on dependency relations. Thus we adapt the notions of kernels and incision functions and to consider all sets of base dependencies that entail one of the dependencies of a conflict.

Definition 23 (Kernel set)**.** Given a dependency relation R and a dependency d, the *kernel set* $R \perp_\sigma d$ consists of all sets $S \subseteq R^b$ with $S \vdash_\sigma d$ and for all $S' \subset S$ it is $S' \nvdash_\sigma d$.

On the basis of kernel sets we can formulate base incisions by means of incision functions.

Definition 24 (Incision function)**.** Let R be a dependency relation. An incision function δ for R is a function $\delta : 2^{2^{\mathcal{R}_\mathcal{A}}} \rightarrow 2^{\mathcal{R}_\mathcal{A}}$ such that the following conditions hold:

(i) $\delta(R \perp d) \subseteq \cup(R \perp d)$

(ii) If $S \in R \perp d$ and $S \neq \emptyset$ then $(S \cap \delta(R \perp d)) \neq \emptyset$

If $R \perp d = \emptyset$ then $\delta(R \perp d) = \emptyset$

Definition 25 (Kernel contraction)**.** Given a dependency relation R and a dependency d the kernel contraction of R by d is given as:

$$R -^k_d d = Cn_\sigma(R^b \setminus \delta(R \perp d))$$

The contraction operation just defined is a general contraction operation by dependencies. For this operation we reformulate the AGM-postulates for contraction.

($D -_d 1$) $Cn_\sigma(R -_d d) = R -_d d$

($D -_d 2$) $R -_d d \subseteq R$

($D -_d 3$) If $d \notin R$, then $R - d = R$

($D -_d 4$) $d \notin R -_d d$

($D -_d 5$) $R \subseteq (R -_d d) + d$

The kernel contraction operator defined above satisfies the postulates which we proposed for d-contraction.

Theorem 26. Let $-^k_d$ be a kernel d-contraction operator on dependency relations. Then $-^k_d$ satisfies the postulates for d-contraction $D -_d 1$, $D -_d 2$, $D -_d 3$ and $D -_d 4$.

The kernel contraction gives means for the contraction of single dependencies from some dependency relation. Before we defined the contraction operation only for sets of conflicts. In the following we are going to define the contraction of sets of conflicts through kernel contractions. For the contraction by sets of conflicts each conflict in the set has to be invalidated. At least one dependency of each conflict cannot be part of the resulting dependency relation. This does not correspond to the removal of one of the dependencies of the conflict since conflicts are only defined on a closed dependency relation but might be generated by other dependencies which also need to be removed to avoid reinstatement. The kernel-contraction operator of Definition 25 allows the contraction by any dependency and performs the contraction on the set of base dependencies. We make use of this for the contraction of sets of conflicts.

Definition 27 (Kernel $\mathcal{C}$-contraction)**.** Given a dependency relation R and a set of conflicts $\mathcal{C} = \{C_1, \ldots, C_n\}$ with $\cup_{1 \leq i \leq n} C_i \subseteq R$, a contraction operator is defined:

$$R -^k_\mathcal{C} \mathcal{C} = Cn_\sigma(R \cap \bigcap_{1 \leq i \leq n} R -^k d_i,\ d_i \in C_i)$$

This contraction operator does satisfy the postulates which we proposed for $\mathcal{C}$-contraction, with the exception of recovery.

Theorem 28. Let $-^k_\mathcal{C}$ be a kernel $\mathcal{C}$-contraction operator on dependency relations. Then $-^k_\mathcal{C}$ satisfies the postulates for $\mathcal{C}$-contraction $D -_\mathcal{C} 1$, $D -_\mathcal{C} 2$, $D -_\mathcal{C} 3$ and $D -_\mathcal{C} 4$.

Revising logic programs

In the preceding section we introduced the means to perform belief revision on dependency relations by dependencies. The dependency relations dealt with are generated by extended logic programs. In the following we give means to transfer the result of such a revision operation back to an extended logic programs. We showed in Theorem 10 that the semantics defined for dependency relations generated by conflict free extended logic programs coincide with the answer set semantics. Here, we generate a logic program from a revised dependency relation.

Given an initial extended logic program P and a single rule $r : H(r) \leftarrow \mathcal{B}(r)$. we generate the dependency relation R_P and revise it by the dependency $d = (L, \{B_1, \ldots, B_n\})$ which results in a new dependency relation $R' = R * d$. From this dependency relation we construct a canonical logic program representing the revision of P by r.

Definition 29 (Dependency Program)**.** Given a dependency relation R we define an extended logic program $P(R)$ as follows: $P(R) = \{L \leftarrow \mathcal{W}. \mid (L, \mathcal{W}) \in R^b)\}$

Based on this transformation we can define the revision of a logic program by a rule as follows.

Definition 30. Given an extended logic program P we define the revision of P by a rule $r : H \leftarrow \mathcal{B}$ as:

$$P * r = P(R_P * (H, \mathcal{B}))$$

Hence, for the revision of a program by a single rule we revise the dependency relation generated by the program by the dependency generated by the rule to be revised by.

Example 31. Consider the following revision of a program $P = \{B. \quad A \leftarrow B.\}$ by a rule $r = \neg A.$: $P * r = P'$. The dependency relation generated by this program R_P and the dependency d_r are represented by the following base dependencies.

$$R^b_P = \{(B, \emptyset), (A, \{B\})\}, d_r = (\neg A, \emptyset)$$

A revision based on a Kernel $\mathcal{C}$-contraction of R_P by d_r will incise one of the two kernels $\{(\neg A, \emptyset)\}$, $\{(\neg A, \{B\}), (B, \emptyset)\}$ for the conflict $C = \{(\neg A, \emptyset), (A, \emptyset)\}$. Hence a valid revision is given by incising the second kernel by $(\neg A, \{B\})$ leading to the resulting dependency relation $R_P * d_r = Cn_\sigma(\{(B, \emptyset), (\neg A, \emptyset)\})$. From this dependency relation we obtain the dependency program $P' = \{B., A.\}$ as the result of the revision.

In particular we hereby perform belief revision of non-monotonic extended logic programs by revising monotonic dependency relations. The latter enable belief operations in correspondence with the AGM theory.

Discussion

In this paper we defined a dependency relation for extended logic programs which complies with the answer set semantics. On the basis of this dependency relation we discussed the AGM postulates for revision. We defined a notion of conflicts in dependency relations yielding a definition of consistency. We defined belief revision and contraction operations of different kinds and analyzed their properties by adapting the AGM-theory. Moreover we moved towards constructive contraction operations by introducing a base representation and base contraction operators. Finally, we showed how these operations can be used to define belief operations for extended logic programs.

In (Flouris, Plexousakis, and Antoniou 2004) the AGM theory has been generalized to a broad class of logics. There a suitable logic is given by some set $\mathcal{S}$ of propositions together with a consequence operator $Cn(\cdot)$. The latter is a mapping from a set of propositions to a set of propositions that satisfies the Tarskian axioms. Any tuple $\langle \mathcal{S}, Cn(\cdot) \rangle$ of this type is considered as a logic. Here, we consider the set of all dependencies of a set of atoms $\mathcal{R}_\mathcal{A}$ together with the consequence operator on dependencies $Cn_\sigma(\cdot)$. The consequence operator on dependencies satisfies the Tarskian axioms and therefore $\langle \mathcal{R}_\mathcal{A}, Cn_\sigma(\cdot) \rangle$ is a logic in the sense defined in (Flouris, Plexousakis, and Antoniou 2004). From this follows directly the existence of an AGM contraction operation $\langle \mathcal{R}_\mathcal{A}, Cn_\sigma(\cdot) \rangle$ that satisfies the AGM contraction postulates (AGM-1)–(AGM-5). Belief revision of extended logic programs is performed by most approaches through syntactic rewriting of programs as surveyed in (Eiter et al. 2002). There, it has also been shown that those approach fall short in satisfying any non-trivial AGM-postulates. AGM belief operations on Horn clause rules have been studied recently, e. g. in (Delgrande 2008; Booth, Meyer, and Varzinczak 2009) or (Alechina, Jago, and Logan 2008), which show the satisfaction of AGM style postulates but are restricted to monotonic logics.

For future work will clearly lie in further elaboration and the analysis of properties of our approaches. Especially, the comparison to other approaches is interesting. Also, constructive methods shall be developed further and implications for the resulting extended logic programs need to be investigated.

References

Alchourron, C. E.; Gardenfors, P.; and Makinson, D. 1985. On the logic of theory change: Partial meet contraction and revision functions. *The Journal of Symbolic Logic* Vol. 50, No. 2 (Jun., 1985):510–530.

Alechina, N.; Jago, M.; and Logan, B. 2008. Preference-based belief revision for rule-based agents. *Synthese* 165(2):159–177.

Bondarenko, A.; Toni, F.; and Kowalski, R. A. 1993. An assumption-based framework for non-monotonic reasoning. In *Proc. 2nd Intern. Workshop on Logic Programming and Non-monotonic Reasoning*, 171–189. MIT Press.

Booth, R.; Meyer, T.; and Varzinczak, I. J. 2009. Next steps in propositional horn contraction. In *IJCAI'09: Proc. of the 21st intl. Jont Conference on Artifical intelligence*, 702–707. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Delgrande, J. P. 2008. Horn clause belief change: Contraction functions. In Brewka, G., and Lang, J., eds., *Principles of Knowledge Representation and Reasoning: Proc. of the 11th Intern. Conference, KR 2008, Sydney, Australia, September 16-19, 2008*, 156–165. AAAI Press.

Eiter, T.; Fink, M.; Sabbatini, G.; and Tompits, H. 2002. On properties of update sequences based on causal rejection. *Theory Pract. Log. Program.* 2(6):711–767.

Flouris, G.; Plexousakis, D.; and Antoniou, G. 2004. Generalizing the AGM postulates: preliminary results and applications. In *Proc. of 10th Intern. Workshop on Non-Monotonic Reasoning (NMR 2004)*, 171–179.

Gelfond, M., and Lifschitz, V. 1988. The stable model semantics for logic programming. In *ICLP/SLP*, 1070–1080. MIT Press.

Hansson, S. O. 1994. Kernel contraction. *J. Symb. Log.* 59(3):845–859.

Hansson, S. O. 2001. *A Textbook of Belief Dynamics*. Kluwer Academic Publishers.

Sefranek, J. 2006. Rethinking semantic of dynamic logic programming. In *Proc. of the 11'th Intl. Workshop on Non-Monotonic Reasoning (NMR-06)*.