(Hrsg.): Studierendenkonferenz Informatik (SKILL 2018),
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2018 1

Wissensrevision in der reaktiven Antwortmengenprogrammierung

Jonas Philipp Haldimann[1]

Abstract: Reaktive Antwortmengenprogrammierung [Ge11; Ge15] ist eine neuere Erweiterung der Antwortmengenprogrammierung, welche sich wiederum für Planungs- und andere Suchprobleme verwenden lässt. Die Erweiterung zur reaktiven Antwortmengenprogrammierung ermöglicht es, Antwortmengenprogramme zu ergänzen oder zu ändern, noch nachdem sie bereits grundiert wurden. Dadurch lässt sie sich auch in dynamischen Umgebungen effizient einsetzen.

Eine solche Anpassung ist eine Form der Wissensrevision. Diese Revision soll hier für Multi-Shot Solver [Ge15], eine Form der reaktiven Antwortmengenprogrammierung, genauer untersucht werden. Dazu werden wir zunächst die AGM-Kriterien [AGM85], die häufig für die Untersuchung der Revision auf Mengen logischer Aussagen verwendet werden, an die reaktive Antwortmengenprogrammierung anpassen. Anschließend untersuchen wir die Revision bei der reaktiven Antwortmegenprogrammierung mit den angepassten AGM-Kriterien und den Basisrevisionskriterien aus [KK12].

Keywords: (Reaktive) Antwortmengenprogrammierung; Wissensrevision; AGM-Theorie; Basisrevisionskriterien

1 Einleitung

Reaktive Antwortmengenprogrammierung [Ge11; Ge15] ist eine neuere Erweiterung der Antwortmengenprogrammierung. Diese wiederum ist eine Form der logischen Programmierung, die auf der Suche nach Antwortmengen zu gegebenen Programmen basiert. Antwortmengenprogrammierung lässt sich für Planungs- und andere Suchprobleme verwenden. Die Erweiterung zur reaktiven Antwortmengenprogrammierung ermöglicht es, Antwortmengenprogramme zu ergänzen oder zu ändern, noch nachdem sie bereits grundiert wurden. Dadurch lässt sich Antwortmengenprogrammierung auch in dynamischen Umgebungen effizient einsetzen.

Betrachten wir beispielsweise einen Roboter, der Kisten in einer Lagerhalle transportiert. Wir wollen Antwortmengenprogrammierung nutzen um die Aktionen des Roboters zu planen. Die Regeln, welche die Bewegung des Roboters beschreiben, bleiben konstant. Einige Informationen, wie der Startpunkt des Roboters oder die abzuarbeitenden Aufträge, ändern sich jedoch. Reaktive Antwortmengenprorammierung ermöglicht es, diese Informationen effizient zur Laufzeit anzupassen.

[1] TU Dortmund, Fakultät für Informatik, Deutschland; Kontakt: jonas.haldimann@tu-dortmund.de

Wenn so eine Anpassung stattfindet, findet eine Wissensrevision statt. Diese Revision soll für Multi-Shot Solver [Ge15], eine Form der reaktiven Antwortmengenprogrammierung, genauer untersucht werden. Anders als Arbeiten die sich mit Wissensrevision in der Antwortmengenprogrammierung allgemein beschäftigen, wird in dieser Arbeit besonders die Wissensrevision in dem Multi-Shot Ansatz zur reaktiven Antwortmengenprogrammierung untersucht. Nachdem die notwendigen Grundlagen zur reaktiven Antwortmengenprogrammierung (Kapitel 2) und zur Wissensrevision (Kapitel 3) vorgestellt wurden, werden wir dazu zunächst die AGM-Kriterien [AGM85], die für die Untersuchung der Revision auf Mengen logischer Aussagen entworfen wurden, an die reaktive Antwortmengenprogrammierung anpassen (Kapitel 4). Anschließend untersuchen wir die Wissensrevision bei der reaktiven Antwortmegenprogrammierung mit den angepassten AGM-Kriterien und den Basisrevisionskriterien aus [KK12] (Kapitel 5).

2 Reaktive Antwortmengenprogrammierung

Die *Antwortmengenprogrammierung* [BK14, Kap. 9], englisch *answer set programming* oder kurz *ASP*, ist eine Form der deklarativen Programmierung und basiert auf einer Semantik sogenannter stabiler Modelle (Antwortmengen). Als Grundlage der Antwortmengenprogrammierung dienen *erweiterte logische Programme* [BK14, Kap. 9.5].

Definition 1 (Erweitertes logisches Programm)**.** *Ein* erweitertes logisches Programm $\mathcal{P}$ *ist eine endliche Menge von* Regeln *der Form*

$$r: \quad H \leftarrow P_1, \ldots, P_n, not\ N_1, \ldots, not\ N_m.$$

Dabei sind $H, P_1, \ldots, P_n$ *und* $N_1, \ldots, N_m$ Literale*, d.h. prädikatenlogische Atome aus der Herbrandbasis* $\mathcal{H}(\mathcal{P})$ *oder deren Negation. Mit* $head(r) := \{H\}$ *bezeichnet man den* Kopf *der Regel,* $P_1, \ldots, P_n$ *heißen* positive Rumpfliterale *und* $N_1, \ldots, N_m$ negative Rumpfliterale*. Die Menge aller Rumpfliterale ist* $body(r) := \{P_1, \ldots, P_n, not\ N_1, \ldots, not\ N_m\}$*. Regeln ohne Rumpfliterale heißen* Fakten *und werden ohne* $\leftarrow$ *notiert. Es dürfen auch Regeln mit leerem Kopf vorkommen, diese heißen* Constraints*.*

Eine Regel eines erweiterten logischen Programms ist anschaulich so zu verstehen, dass der Kopf einer Regel erfüllt sein muss, wenn alle positiven Rumpfliterale erfüllt sind und für alle negativen Rumpfliterale nicht sicher ist, dass sie erfüllt sind. Der Rumpf eines Constraints darf nicht erfüllt sein. Um die Semantik formal zu erklären, benötigen wir zunächst den Begriff der *Gelfond-Lifschitz-Reduktion* [BK14, Kap. 9.5]. Sie entfernt sämtliche *Default-Negationen* (*not*-negierte Literale) aus einem erweiterten logischen Programm und reduziert letzteres somit auf ein klassisches logisches Programm.

Definition 2 (Gelfond-Lifschitz-Reduktion)**.** *Sei* $\mathcal{P}$ *ein erweitertes logisches Programm und* S *ein* Zustand*, d.h. eine endliche Menge von Literalen, in der es keine zwei Literale* $A, B \in S$

mit $A = \neg B$ gibt. Die Gelfond-Lifschitz-Reduktion *bildet $\mathcal{P}$ und S auf das folgende* Redukt *ab:*

$$\mathcal{P}^S = \{H \leftarrow P_1, \ldots, P_n. \mid H \leftarrow P_1, \ldots, P_n, not\ N_1, \ldots, not\ N_m. \in \mathcal{P} \text{ und } S \cap \{N_1, \ldots, N_m\} = \emptyset\} \ .$$

Die Reduktion entfernt alle Regeln aus $\mathcal{P}$, bei denen ein negatives Rumpfliteral auch in S vorkommt. Bei den übrigen Regeln werden die negativen Rumpfliterale entfernt. Übrig bleibt ein Programm ohne Default-Negation. Eine *Antwortmenge* [BK14, Kap. 9.7] kann nun als Fixpunkt unter Anwendung des *Cl*-Operators auf das entsprechende Redukt definiert werden. Der *Cl*-Operator bildet dabei ein logisches Programm auf sein minimalen geschlossenen Zustand, d.h. die kleinste Menge von Literalen, die jede Regel erfüllt, ab [BK14, Kap. 9.6].

Definition 3 (Antwortmenge)**.** *Es sei $\mathcal{P}$ ein erweitertes logisches Programm und S ein Zustand. S ist eine* Antwortmenge *von $\mathcal{P}$, wenn $S = Cl(\mathcal{P}^S)$ gilt.*

Anders als bei klassisch logischen Programmen gibt es bei erweiterten logischen Programmen nicht immer eine eindeutige Antwortmenge.

Beispiel 1. *Seien $\mathcal{P}_1 = \{P \leftarrow not\ Q.,\ Q \leftarrow not\ P.\}$ und $\mathcal{P}_2 = \{P \leftarrow not\ Q.,\ Q \leftarrow P.\}$ erweiterte logische Programme. Sowohl $S_a = \{P\}$ als auch $S_b = \{Q\}$ sind Antwortmengen für das Programm $\mathcal{P}_1$: $Cl(\mathcal{P}_1^{S_a}) = Cl(\{P.\}) = \{P\} = S_a$ und $Cl(\mathcal{P}_1^{S_b}) = S_b$. Das Programm $\mathcal{P}_2$ hingegen hat keine Antwortmenge.*

Regeln, die Literale mit Variablen beinhalten, sind als Schemata zu verstehen. Vor dem Bestimmen von Antwortmengen werden diese Regeln *grundiert*, d.h. die Variablen werden durch die zur Verfügung stehenden Konstanten ersetzt. Ist beispielsweise die Konstantenmenge $\{1, 2\}$ gegeben, so wird die Regel $a(X) \leftarrow b(X), c(X).$ durch die beiden Regeln $a(1) \leftarrow b(1), c(1).$ und $a(2) \leftarrow b(2), c(2).$ ersetzt. Ein Programm, das diesen Vorverarbeitungsschritt durchführt, heißt *Grounder*. Der Grounder vereinfacht dabei das entstehende Programm direkt. Käme $b(2)$ beispielsweise nicht im Kopf einer Regel des Programms vor, würde er die Regel $a(2) \leftarrow b(2), c(2).$ nicht erzeugen.

Mittels Kardinalitätsschranken lassen sich Regeln um die Möglichkeit erweitern, mehrere Literale im Kopf zu beinhalten. Eine solche Regel hat dann die Form $k\{H_1, ..., H_j\}l \leftarrow P_1, ..., P_n, not\ N_1, ..., not\ N_m.$ mit natürlichen Zahlen k und l. Wenn der Rumpf der Regel erfüllt ist, kann die Belegung der Literale $H_1, ..., H_j$ im Kopf frei gewählt werden, solange mindestens k und höchstens l der Literale erfüllt sind. Dabei ist es auch möglich, eine oder beide Grenzen wegzulassen. Fehlt die obere Grenze, können beliebig viele Literale im Kopf der Regel wahr sein. Fehlt die untere Grenze, dürfen beliebig wenig Literale erfüllt sein. Insbesondere ist es möglich, dass alle Literale mit „Falsch“ belegt werden. Die Regel $\{H\}.$ bedeutet also, dass H möglicherweise erfüllt ist.

Für manche Anwendungen ist es notwendig, das logische Programm oft zu ändern, beispielsweise wenn Fakten eine sich verändernde Umgebung darstellen. Für „normale“

Antwortmengenprogramme bedeutet eine Änderung, das Programm anzupassen und erneut dem Grounder und dem Solver[2] zu übergeben. *Reaktive Antwortmengenprogrammierung* gestaltet solche Änderungen einfacher und effizienter.

Ein Ansatz zur reaktiven Antwortmengenprogrammierung ist die Multi-Shot Methodik [Ge15]. Dieser Ansatz wurd auch in Clingo[3], einem der wichtigsten Programme zum Lösen von Antwortmengenprogrammen, umgesetzt.

Die Multi-Shot Methodik erlaubt es, die Belegung einiger Literale zu ändern, auch nachdem das Program bereits grundiert wurde. Diese Literale müssen vor dem Grundieren gekennzeichnet werden.

Definition 4 (erweiterbares logisches Programm)**.** *Ein* erweiterbares logisches Programm $\mathcal{P}$ *darf neben Regeln zusätzlich* externe Deklarationen *der Form*

`#external` a `:` B.

mit einem Atom a und einem Regelrumpf B enthalten. Man bezeichnet a als externe Variable.

Eine externe Variable, die nicht im Kopf einer Regel des erweiterbaren logischen Programms vorkommt, wird vor dem Lösen des Programms mit „Wahr“, „Falsch“ oder „Unbekannt“ belegt. Ohne entsprechende Kennzeichnung würde der Grounder annehmen, dass a falsch ist, weil es nicht im Kopf einer Regel vorkommt und das Programm entsprechend vereinfachen.

Definition 5 (partielle Belegung)**.** *Eine* partielle Belegung *einer Menge A von grundierten Atomen ist eine Funktion* $i : A \to \{t, f, u\}$ *die jedem Atom einen der Werte „Wahr“, „Falsch“ oder „Unbekannt“ (bzw. t, f oder u) zuordnet. Wir schreiben* $A^t = \{a \in A \mid i(a) = t\}$ *und analog* $A^f = \{a \in A \mid i(a) = f\}$ *und* $A^u = \{a \in A \mid i(a) = u\}$*. Üblicherweise gibt man eine partielle Belegung durch* $\langle A^t, A^f \rangle$ *oder* $\langle A^t, A^u \rangle$ *an.*

Der Zustand eines Programmes wird in Clingo während der Ausführung durch ein Tripel $(\mathcal{Q}, \mathcal{P}, I)$ repräsentiert. Dabei ist $\mathcal{Q}$ ein (noch nicht grundiertes) logisches Programm. $\mathcal{Q}$ dient lediglich dazu, Regeln zu sammeln, die zum Programm hinzugefügt werden sollen, bevor sie grundiert werden. $\mathcal{P}$ ist ein grundiertes logisches Programm. $\mathcal{P}$ enthält die Regeln, die das Verhalten des Programms bestimmen. Wenn die Regeln in $\mathcal{Q}$ grundiert wurden, werden sie zu $\mathcal{P}$ hinzugefügt. I schließlich ist die Menge der Input-Atome, d.h. der externen Variablen von $\mathcal{P}$, die nicht im Kopf einer Regel vorkommen, zusammen mit einer partielle Belegung $\langle I^t, I^u \rangle$ dieser Atome. Die Kombination von $\mathcal{P}$ mit der Belegung $\langle I^t, I^u \rangle$, die beim Lösen dem Solver übergeben wird, ist $P_{\langle I^t, I^u \rangle} := P \cup \{a. \mid a \in I^t\} \cup \{\{a\}. \mid a \in I^u\}$.

Um den Zustand eines Programms zu beeinflussen, stehen verschiedene Operationen zur Verfügung. Hier sind vor allem die Operationen add, ground und assignExternal interessant. In [Ge15] sind ausserdem die Operationen create und releaseExternal, sowie solve zum Lösen von Programmen beschrieben.

[2] Der Solver ist die Software, die ein Antwortmengenprogramm löst indem es passende Antwortmengen bestimmt.
[3] Clingo ist Teil der Softwaresammlung *Potassco* die an der Universität Potsdam entwickelt wird (potassco.org)

add($\mathcal{R}$) $(\mathcal{Q}, \mathcal{P}, I) \mapsto (\mathcal{Q} \cup \mathcal{R}, \mathcal{P}, I)$ für ein (nicht grundiertes) Programm $\mathcal{R}$
Fügt nicht grundierte Regeln zu dem Programm hinzu.

ground() $(\mathcal{Q}, \mathcal{P}_1, I_1) \mapsto (\emptyset, \mathcal{P}_2, I_2)$ mit

- $D = \{a \leftarrow B, \varepsilon. \mid \texttt{\#external}\ a\ \texttt{:}\ B. \in \mathcal{Q}\}$
  $\mathcal{P} = \{r \in grd(\mathcal{Q} \cup D \cup \{\{\varepsilon\}.\}) \setminus \{\{\varepsilon\}.\} \mid \varepsilon \notin body(r)\}$
  $E = \{head(r) \mid r \in grd(\mathcal{Q} \cup D \cup \{\{\varepsilon\}.\}),\ \varepsilon \in body(r)\}$
  $\mathcal{P}$ enthält die grundierten Regeln aus $\mathcal{Q}$ und E die in $\mathcal{Q}$ deklarierten externen Variablen. ε ist ein spezielles Atom, das Regeln aus externen Deklarationen markiert und in $\mathcal{Q}$ nicht vorkommt. Weitere Details hierzu finden sich in [Ge15].
- $\mathcal{P}_2 = \mathcal{P}_1 \cup \mathcal{P}$
- $I_2 = (I_1 \cup E) \setminus head(\mathcal{P}_2)$
- $I_2^t = I_2 \cap I_1^t, \qquad I_2^u = I_2 \cap I_1^u, \qquad I_2^f = I_2 \setminus I_2^t \setminus I_2^u.$

Grundiert das Programm in $\mathcal{Q}$ bzgl. $\mathcal{P}_1$ und I_1, fügt es zu $\mathcal{P}$ hinzu und passt die partielle Belegung der Input-Variablen I an.
Bei der Grundierung von $\mathcal{Q}$ müssen $\mathcal{P}_1$ und I_1 einfließen, da sie zur Herbrandbasis (d.h. die verfügbaren Konstanten) beitragen. Zu I werden die neuen externen Variablen aus $\mathcal{Q}$ hinzugefügt. Die externen Variablen, die in einem Regelkopf von $\mathcal{Q}$ vorkommen, werden aus den Inputatomen entfernt.

assignExternal(a, x) $(\mathcal{Q}, \mathcal{P}, I_1) \mapsto (\mathcal{Q}, \mathcal{P}, I_2)$ mit

- $I_2 = I_1$
- $I_2^t = I_1^t \cup \{a\}$ wenn $x = t$ und $a \in I_1$ ist
  $I_2^t = I_1^t$ wenn $x = t$ und $a \notin I_1$ ist
  $I_2^t = I_1^t \setminus \{a\}$ wenn $x \in \{f, u\}$
- $I_2^u = I_1^u \cup \{a\}$ wenn $x = u$ und $a \in I_1$ ist
  $I_2^u = I_1^u$ wenn $x = u$ und $a \notin I_1$ ist
  $I_2^u = I_1^u \setminus \{a\}$ wenn $x \in \{t, f\}$
- $I_2^f = I_2 \setminus I_2^t \setminus I_2^u$.

Belegt das Atom a mit x, wobei x „wahr“ (t), „falsch“ (f) oder „unbekannt“ (u) sein kann. Für $x = t$ oder $x = f$ kommt a in jeder bzw. keiner Antwortmenge vor. Für $x = u$ ist nicht festgelegt, ob a in den Antwortmengen des Programms vorkommen muss oder nicht.

3 Wissensrevision

Es kommt vor, dass Software-Agenten ihr Wissen anpassen müssen. Die AGM-Theorie [AGM85] nimmt an, dass das Wissen eines Agenten eine Menge von (logischen) Aussagen

ist. Ist diese Menge über dem Cn-Operator[4] abgeschlossen, bezeichnet man sie auch als Wissensmenge.

Auf einer Menge von Aussagen sind die Operationen Expansion (+), Kontraktion (−) und Revision (∗) möglich. Bei der Expansion werden neue Informationen zu dem vorhandenen Wissen hinzugefügt. Bei der Kontraktion werden bestimmte Aussagen aus dem vorhanden Wissen entfernt, sollen also vergessen werden. Bei der Revision sollen neue Informationen hinzugefügt werden. Dabei soll jedoch das vorhandene Wissen so angepasst werden, dass das Wissen nach der Operation wieder konsistent ist, auch wenn die neuen Informationen dem alten Wissen wiedersprechen.

Um verschiedene Verfahren für diese Operationen zu bewerten, werden die sogenannten AGM-Kriterien verwendet. Diese sind wünschenswerte Eigenschaften von Verfahren für die beschriebenen Operationen.

Definition 6 (AGM-Revisionskriterien)**.** *Es sei A eine Menge von logischen Aussagen und x eine neue Aussage. Die AGM-Kriterien für die Revision nach [AGM85] lauten:*

(*1) $A * x$ *ist eine Wissensmenge.*

(*2) $x \in A * x$

(*3a) $A * x \subseteq Cn(A \cup \{x\})$

(*3b) *Wenn* $\neg x \notin Cn(A)$ *gilt, dann ist* $Cn(A \cup \{x\}) \subseteq A * x$.

(*4) *Wenn x erfüllbar ist, dann ist* $A * x$ *konsistent.*

(*5) *Wenn* $Cn(x) = Cn(y)$ *gilt, dann ist* $A * x = A * y$.

(*6) *Wenn A eine Wissensmenge ist, dann ist* $(A * x) \cap A = A - \neg x$.

(*7) $A * (x \wedge y) \subseteq Cn((A * x) \cup \{y\})$

(*8) *Wenn* $\neg y \notin A * x$ *gilt, dann ist* $Cn((A * x) \cup \{y\}) \subseteq A * (x \wedge y)$.

*Die Kriterien (*3a) und (*3b) sind in der Originalarbeit zusammengefasst, werden in einigen Quellen (z.B. [Ke17]) jedoch als einzelne Kriterien aufgeführt.*

Das Kriterium AGM (*2) beispielsweise fordert, dass das neue Wissen x in dem Wissen nach der Revision enthalten ist; AGM (*5) fordert, dass semantisch äqivalente Informationen den gleichen Einfluss auf die Menge der Aussagen nach der Revision haben.

[4] Der Cn-Operator bildet eine Menge M von logischen Aussagen auf die Menge $Cn(M) := \{a \mid M \models a\}$ ab.

Man sieht, dass die AGM-Kriterien nicht für Antwortmengenprogrammierung vorgesehen sind. Ein logisches Programm ist eine Menge von Regeln und nicht von Aussagen. In Kapitel 4 werden wir die AGM-Revisionskriterien an reaktive Antwortmengenprogrammierung anpassen. Eine andere Alternative zu den AGM-Revisionskriterien sind die Basisrevisionskriterien für Antwortmengenprogramme, die in [KK12] vorgestellt wurden.

Definition 7 (Basisrevisionskriterien)**.** *Im Folgenden sei P das vorhandene logische Programm und Q eine Menge von neuen Regeln, die hinzugefügt werden sollen. Dann sind die Basisrevisionskriterien für Antwortmengenprogrammierung:*

Success $Q \subseteq P * Q$

Inclusion $P * Q \subseteq P \cup Q$

Vacuity *Wenn $P \cup Q$ konsistent*[5] *ist, dann gilt $P \cup Q \subseteq P * Q$.*

Consistency *Wenn Q konsistent ist, dann ist $P * Q$ konsistent.*

NM-Consistency *Wenn es ein konsistentes Programm X gibt, so dass $Q \subseteq X \subseteq P \cup Q$ gilt, dann ist $P * Q$ konsistent.*

Relevance *Wenn es eine Regel $r \in (P \cup Q) \setminus (P * Q)$ gibt, dann existiert ein Programm H mit $P * Q \subseteq H \subseteq P \cup Q$, so dass H konsistent ist, aber $H \cup \{r\}$ nicht.*

Fullness *Wenn es eine Regel $r \in (P \cup Q) \setminus (P * Q)$ gibt, dann ist $P * Q$ konsistent, $(P * Q) \cup \{r\}$ aber nicht.*

Uniformity *Wenn für alle $P' \subseteq P$ gilt, dass $P' \cup Q$ inkonsistent ist genau dann wenn $P' \cup R$ inkonsistent ist, dann ist $P \cap (P * Q) = P \cap (P * R)$.*

Dabei gilt, dass Fullness strenger ist als Relevance und Relevance strenger ist als Vacuity: Fullness $\Rightarrow$ Relevance $\Rightarrow$ Vacuity. Außerdem folgt Consistency aus NM-Consistency. Die unterschiedlich starken Versionen der Kriterien erlauben es untersuchten Operationen genauer einzuordnen.

4 Anpassung der Revisionskriterien für die reaktive Antwortmengenprogrammierung

Die AGM-Kriterien sind zur Beurteilung der Wissensrevision auf Mengen von Aussagen entworfen. Deshalb lassen sich die meisten Kriterien nicht auf die Operationen der reaktiven

[5] Ein (erweitertes) logisches Programm $\mathcal{P}$ heißt genau dann *konsistent* oder *erfüllbar*, wenn $\mathcal{P}$ mindestens eine Antwortmenge hat.

Antwortmengenprogrammierung anwenden. Im Folgenden versuchen wir die Revisionskriterien so anzupassen, dass sie auch auf die Operationen ground und assignExternal der Multishotsolver anwendbar sind. Die Herausforderung ist dabei, die den Kriterien zugrundeliegende Struktur auf die reaktive Antwortmengenprogrammierung zu übertragen, obwohl viele Konzepte wie beispielsweise die Abgeschlossenheit von Aussagenmengen bei Antwortmengenprogrammen nicht anwendbar sind.

Dafür sei $T_1 = (Q_1, \mathcal{P}_1, I_1)$ das Clingo-Tripel vor und $(Q_2, \mathcal{P}_2, I_2)$ das Tripel nach Anwendung der jeweiligen Operation. Q sei ein logisches Programm, welches das neu hinzuzufügende Wissen repräsentiert. Während bei dem Kriterium (*1') die Tripel-Struktur berücksichtigt werden kann, werden wir bei den meisten Kriterien statt der Clingo-Tripel die Mengen $\mathcal{P}_{1\langle I_1^t, I_1^u\rangle}$ und $\mathcal{P}_{2\langle I_2^t, I_2^u\rangle}$ betrachten:

$$T_1 = (Q_1, \mathcal{P}_1, I_1) \xrightarrow{\quad *Q \quad} T_2 = (Q_2, \mathcal{P}_2, I_2)$$
$$\mathcal{P}_{1\langle I_1^t, I_1^u\rangle} \dashrightarrow \mathcal{P}_{2\langle I_2^t, I_2^u\rangle}$$
Untersuchte Revision

Im Folgenden sind beispielhaft die Überlegungen zu den Anpassungen der AGM-Revisionskriterien (*3b), (*5) und (*6) dargestellt.

AGM-Kriterium (*3b): Wenn $\neg x \notin Cn(A)$ gilt, dann ist $Cn(A \cup \{x\}) \subseteq A * x$.
Die Vorraussetzung $\neg x \notin Cn(A)$ lässt sich als „x wiederspricht A nicht“ interpretieren. Für Antwortmengenprogramme P, Q lässt sich dass durch „$P \cup Q$ ist konsistent“ ausdrücken. Die Folgerung $Cn(A \cup \{x\}) \subseteq A * x$ drückt aus, das alles, was sich aus aus $A \cup \{x\}$ folgern lässt, auch in dem revidierten Wissen $A * x$ enthalten ist. Der Cn-Operator lässt sich nicht auf Antwortmengenprogramme anweden, aber wir können fordern, dass alle alten und neuen Regeln bei der Revision erhalten bleiben: $P \cup Q \subseteq P * Q$.

Schließlich ersetzen wir die Mengen A und $A * x$, wie beschrieben, durch $\mathcal{P}_{1\langle I_1^t, I_1^u\rangle}$ vor und $\mathcal{P}_{2\langle I_2^t, I_2^u\rangle}$ nach der Operation. Das neue Kriterium lautet:
AGM (*3b'): Wenn $\mathcal{P}_{1\langle I_1^t, I_1^u\rangle} \cup Q$ konsistent ist, dann ist $\mathcal{P}_{1\langle I_1^t, I_1^u\rangle} \cup Q \subseteq \mathcal{P}_{2\langle I_2^t, I_2^u\rangle}$.

AGM-Kriterium (*5): Wenn $Cn(x) = Cn(y)$ gilt, dann ist $A * x = A * y$.
$Cn(x) = Cn(y)$ bedeutet, dass x und y äquivalent sind. Für Antwortmengenprogramme P, Q gibt es verschiedene Arten von Äquivalenz. Eine der einfachsten ist $C_{AS}(P) = C_{AS}(Q)$ mit $C_{AS}(P) := \{L \mid L \text{ liegt in allen Antwortmengen von } P\}$.

Sei $T_2 = (Q_2, \mathcal{P}_2, I_2)$ das Ergebnis der Revision mit Q und $T_3 = (Q_3, \mathcal{P}_3, I_3)$ das Ergebnis der Revision von T_1 mit R. Dann ist das angepasste Kriterium:
AGM (*5'): Wenn $C_{AS}(Q) = C_{AS}(R)$ gilt, dann ist $\mathcal{P}_{2\langle I_2^t, I_2^u\rangle} = \mathcal{P}_{3\langle I_3^t, I_3^u\rangle}$ mit $T_3 = T_1 * Q$.

AGM-Kriterium (*6): Wenn A eine Wissensmenge ist, dann ist $(A * x) \cap A = A - \neg x$. Dieses Kriterium fordert, dass von den Aussagen in der alten Wissensmenge bei der Revision nur die entfernt werden, die für das Vergessen von $\neg x$ entfernt werden müssen. Es gibt aber keine Negation und keine Kontraktion für allgemeine logische Programme. Dieses Kriterium können wir zur Untersuchung der Antwortmengenprogramme also nicht sinvoll anpassen. Die AGM (*6) zugrundeliegende Idee, dass bei der Revision keine Aussagen unnötigerweise entfernt werden, wird am ehesten in den Basisrevisionskriterien Fullness und Relevance ausgedrückt.

Die anderen AGM-Revisionskriterien lassen sich in ähnlicher Weise anpassen. Wir haben uns für die folgenden angepassten Kriterien entschieden:

(*1') Die Revisionsoperation gibt ein Clingo-Tripel zurück.

(*2') $Q \subseteq \mathcal{P}_{2\langle I_2^t, I_2^u\rangle}$

(*3a') $\mathcal{P}_{2\langle I_2^t, I_2^u\rangle} \subseteq \mathcal{P}_{1\langle I_1^t, I_1^u\rangle} \cup Q$

(*3b') Wenn $\mathcal{P}_{1\langle I_1^t, I_1^u\rangle} \cup Q$ konsistent ist, dann ist $\mathcal{P}_{1\langle I_1^t, I_1^u\rangle} \cup Q \subseteq \mathcal{P}_{2\langle I_2^t, I_2^u\rangle}$.

(*4') Wenn Q konsistent ist, dann ist $\mathcal{P}_{2\langle I_2^t, I_2^u\rangle}$ konsistent.

(*5') Wenn $C_{AS}(Q) = C_{AS}(R)$ gilt, dann ist $\mathcal{P}_{2\langle I_2^t, I_2^u\rangle} = \mathcal{P}_{3\langle I_3^t, I_3^u\rangle}$ mit $T_3 = T_1 * Q$.

Bei den Kriterien (*7) und (*8) hängen die Anpassungen von der Operation ab:

(*7') $\mathcal{P}_{5\langle I_5^t, I_5^u\rangle} \subseteq \mathcal{P}_{2\langle I_2^t, I_2^u\rangle} \cup R$ mit $T_5 = (T_1 * Q) * R$ bei der Untersuchung von *assignExternal*

(*7'') $\mathcal{P}_{4\langle I_4^t, I_4^u\rangle} \subseteq \mathcal{P}_{2\langle I_2^t, I_2^u\rangle} \cup R$ mit $T_4 = T_1 * (Q \cup R)$ bei der Untersuchung von *ground*

(*8') Wenn $\mathcal{P}_{2\langle I_2^t, I_2^u\rangle} \cup R$ konsistent ist, dann gilt $\mathcal{P}_{2\langle I_2^t, I_2^u\rangle} \cup R \subseteq \mathcal{P}_{5\langle I_5^t, I_5^u\rangle}$ mit $T_5 = (T_1 * Q) * R$ für die Untersuchung von *assignExternal*

(*8'') Wenn $(\mathcal{P}_{2\langle I_2^t, I_2^u\rangle} \cup R$ konsistent ist, dann gilt $\mathcal{P}_{2\langle I_2^t, I_2^u\rangle} \cup R \subseteq \mathcal{P}_{4\langle I_4^t, I_4^u\rangle}$ mit $T_4 = T_1 * (P \cup Q)$ für die Untersuchung von *ground*.

Die Basisrevisionskriterien sind bereits an die Verwendung mit Antwortmengenprogrammen angepasst. Sie sind jedoch nicht für die Untersuchung von Clingo-Tripeln vorgesehen. Seien $T_1 = (Q_1, \mathcal{P}_1, I_1)$ und $T_2 = (Q_2, \mathcal{P}_2, I_2)$ die Clingo-Tripel vor und nach der Operation. Es werden daher bei den Untersuchungen mit den Basisrevisionskriterien statt der Clingo-Tripel T_1 und T_2 die Mengen $P := \mathcal{P}_{1\langle I_1^t, I_1^u\rangle}$ und $P * Q := \mathcal{P}_{2\langle I_2^t, I_2^u\rangle}$ betrachtet.

Damit sind einige Basisrevisionskriterien äquivalent zu angepassten AGM-Kriterien. Success ist äquivalent zu AGM-Kriterium (*2'), Inclusion ist äquivalent zu AGM-Kriterium (*3a'), Vacuity ist äquivalent zu AGM-Kriterium (*3b') und Consistency ist äquivalent zu AGM-Kriterium (*4').

5 Formale Untersuchung der reaktiven Antwortmengenprogrammierung auf Erfüllung der Revisionskriterien

Untersuchen wir zunächst die Operation *assignExternal(v, t)* mit den angepassten AGM-Revisionskriterien aus Kapitel 4 und den Basisrevisionskriterien. Diese Operation belegt die externen Variable v mit „wahr“. Das neu dazugewonnene Wissen ist also $Q := \{v.\}$.

Satz 1. *Die Operation* assignExternal(v, t) *erfüllt die angepassten AGM-Kriterien (*1'), (*2'), (*3a'), (*5') und (*7'). Die Kriterien (*3b'), (*4') und (*8') sind nicht erfüllt.*

Die Operation erfüllt die Basisrevisionskriterien Success und Inclusion. Die Kriterien Vacuity, Relevance, Fullness, (NM-)Consistency und Uniformity sind nicht erfüllt.

Beweis. Für diese Operation gilt $\mathcal{P}_{2\langle I_2^t, I_2^u\rangle} = \mathcal{P}_{1\langle I_1^t, I_1^u\rangle} \cup \{v.\} \setminus \{\{v\}.\}$.

Die Operation *assignExternal(v, t)* gibt ein Clingo-Tripel zurück. AGM (*1') ist also erfüllt.
Nach der Operation gilt $I_2(v) = t$. Damit ist $Q = \{v.\} \subseteq \mathcal{P}_2 \cup \{a. \mid a \in I_2^t\} \cup \{\{a\}. \mid a \in I_2^u\} = \mathcal{P}_{2\langle I_2^t, I_2^u\rangle}$. Also ist AGM (*2') erfüllt.
Wenn $I_1(v) = t$ oder $I_1(v) = f$ gilt, dann ist $\mathcal{P}_{2\langle I_2^t, I_2^u\rangle} = \mathcal{P}_{1\langle I_1^t, I_1^u\rangle} \cup \{v.\} = \mathcal{P}_{1\langle I_1^t, I_1^u\rangle} \cup Q$ und das Kriterium (*3a') ist erfüllt. Wenn aber $I_1(v) = u$ gilt, dann ist $\mathcal{P}_{2\langle I_2^t, I_2^u\rangle} = \mathcal{P}_{1\langle I_1^t, I_1^u\rangle} \cup \{v.\} \setminus \{\{v\}.\} \subseteq \mathcal{P}_{1\langle I_1^t, I_1^u\rangle} \cup Q$ und das Kriterium (*3a') ist ebenfalls erfüllt.
AGM (*3b') ist nicht erfüllt. Wenn vor der Operation $I_1(v) = u$ gilt, ist $\mathcal{P}_{2\langle I_2^t, I_2^u\rangle} = \mathcal{P}_{1\langle I_1^t, I_1^u\rangle} \cup Q \setminus \{\{v\}.\} \subsetneq \mathcal{P}_{1\langle I_1^t, I_1^u\rangle} \cup Q$.
Ein Gegenbeispiel für AGM (*4') ist:

Beispiel 2. *Sei* $Q_1 = \emptyset$ *und* $\mathcal{P}_1 = \{\leftarrow v.\}$ *und* $I_1 = \{v \mapsto f\}$. *Die Antwortmenge von* $\mathcal{P}_{1\langle I_1^t, I_1^u\rangle}$ *ist* $\emptyset$. *Nach der Operation* assignExternal(v, t) *erhalten wir das Tripel* $(Q_2, \mathcal{P}_2, I_2)$ *mit* $I_2(v) = t$. *Das Programm* $\mathcal{P}_{2\langle I_2^t, I_2^u\rangle} = \{\leftarrow v., v.\}$ *hat keine Antwortmenge, obwohl* $Q = \{v.\}$ *konsistent ist.*

Für zwei einelementige Programme $Q = \{v.\}$ und $R = \{w.\}$ gilt $C_{AS}(Q) = C_{AS}(R)$ genau dann, wenn $v = w$ gilt. Wenn $v = w$ gilt, dann auch $\mathcal{P}_{2\langle I_2^t, I_2^u\rangle} = \mathcal{P}_{3\langle I_3^t, I_3^u\rangle}$. AGM (*5') ist also erfüllt.
AGM (*7') ist erfüllt: $\mathcal{P}_{5\langle I_5^t, I_5^u\rangle} = \mathcal{P}_5 \cup \{a. \mid a \in I_5^t\} \cup \{\{a\}. \mid a \in I_5^u\}$
$= \mathcal{P}_2 \cup \{a. \mid a \in I_2^t\} \cup \{\{a\}. \mid a \in I_2^u\} \cup \{w.\} \setminus \{\{w\}.\} \subseteq \mathcal{P}_{2\langle I_2^t, I_2^u\rangle} \cup \{w.\}$
AGM (*8') ist nicht erfüllt. Wenn vor der Operation $I_1(w) = u$ war, dann ist $\{w\}. \in \mathcal{P}_{2\langle I_2^t, I_2^u\rangle}$ aber $\{w\}. \notin \mathcal{P}_{5\langle I_5^t, I_5^u\rangle}$.

Die Basisrevisionskriterien Success, Inclusion, Vacuity und Consistency sind jeweils äquivalent zu einem bereits untersuchten angepassten AGM-Kriterium. Die Kriterien Fullness und Relevance sind nicht erfüllt, da schon das schwächere Kriterium Vacuity nicht erfüllt ist. Ebenso kann NM-Consistency nicht erfüllt sein, da Consistency nicht erfüllt ist.

Uniformity ist nicht erfüllt, wie dieses Gegenbeispiel zeigt:

Beispiel 3. *Sei* $\mathcal{P}_1 = \mathcal{Q}_1 = \emptyset$ *und* $I_1 = \{v \mapsto u, w \mapsto u\}$*. Die Anwendung der Operation* assignExternal(v, t) *(also* $Q := \{v.\}$*) führt zu dem Tripel* $(\mathcal{P}_2, \mathcal{Q}_2, I_2)$*, die Anwendung von* assignExternal(w, t) *(also* $R := \{w.\}$*) führe zu* $(\mathcal{P}_3, \mathcal{Q}_3, I_3)$*. Entsprechend sind* $P * Q := \mathcal{P}_{2\langle I_2^t, I_2^u \rangle}$ *und* $P * R := \mathcal{P}_{3.\langle I_3^t, I_3^u \rangle}$*.*

Es ist $P = \mathcal{P}_{1\langle I_1^t, I_1^u \rangle} = \{\{v\}., \{w\}.\}$*. Damit ist für jede Teilmenge* P' *von* P *sowohl* $P' \cup Q$ *als auch* $P' \cup R$ *konsistent. Trotzdem ist* $P \cap (P * Q) = \{\{v\}., \{w\}.\} \cap \{v., \{w\}.\} = \{\{w\}.\} \neq \{\{v\}.\} = \{\{v\}., \{w\}.\} \cap \{\{v\}., w.\} = P \cap (P * R)$*.*

□

Diese Untersuchungen lassen sich analog für *ground()* und *assignExternal(v, f)* mit $\mathcal{Q} := \mathcal{P}$ bzw. $\mathcal{Q} := \{\neg v\}$ durchführen. Man erhält die Ergebnisse:

Satz 2. *Die Operation* ground() *erfüllt die angepassten AGM-Kriterien (*1'), (*2'), (*3a') und (*7'). Die Kriterien (*3b'), (*4'), (*5) und (*8') sind nicht erfüllt.*

ground() *erfüllt nur die Basisrevisionskriterien Success und Inclusion. Die Kriterien Vacuity, Relevance, Fullness, (NM-)Consistency und Uniformity sind nicht erfüllt.*

Satz 3. *Die Operation* assignExternal(v, f) *erfüllt die angepassten AGM-Kriterien (*1'), (*3a'), (*5') und (*7'). Die Kriterien (*2'), (*3b'), (*4') und (*8') sind nicht erfüllt.*

assignExternal(v, f) *erfüllt nur das Basisrevisionskriterium Inclusion. Die Kriterien Success, Vacuity, Relevance, Fullness, (NM-)Consistency und Uniformity sind nicht erfüllt.*

Die Beweise zu den Aussagen in Satz 2 und 3 befinden sich in [Ha17]. Die Operation *assignExternal(v, u)* ist eine Kontraktion und wurde daher nicht mit den Revisionskriterien untersucht.

6 Zusammenfassung und Ausblick

In diesem Paper wurde die Wissensrevision in der Multi-Shot Methodik untersucht. Dabei kamen hier angepasste AGM-Revisionskriterien und die Basisrevisionskriterien aus [KK12] zum Einsatz. Eine Übersicht über die Ergebnisse findet sich in Tabelle 1. Viele Kriterien sind nicht erfüllt. Die reaktive Antwortmengenprorammierung ist also kein besonders gute geeignetes Werkzeug zum Wissensmanagement.

Offen bleibt die Untersuchung der Operationen mit den AGM-Kontraktionskriterien. Diese wurde in [Ha17] durchgeführt.

Danksagung. Dieses Paper ist auf Grundlage meiner Bachelorarbeit [Ha17] entstanden, die ich unter der Betreuung von Prof. Gabriele Kern-Isberner und Marco Wilhelm verfasst habe. Ich danke allen, die mich beim Verfassen dieser Bachelorarbeit unterstützt haben.

	assignExternal(v, t)	*assignExternal(v, f)*	*ground()*
(*1')	j	j	j
(*2')	j	n	j
(*3a')	j	j	j
(*3b')	n	n	n
(*4')	n	n	n
(*5')	j	j	n
(*7')	j	j	j
(*8')	n	n	n
Success	j	n	j
Inclusion	j	j	j
Vacuity	n	n	n
Relevance	n	n	n
Fullness	n	n	n
Consistency	n	n	n
NM-Consistency	n	n	n
Uniformity	n	n	n

Tab. 1: Übersicht über die Ergebnisse der Untersuchung

Literatur

[AGM85] Alchourrón, C. E.; Gärdenfors, P.; Makinson, D.: On the logic of theory change: Partial meet contraction and revision functions. The Journal of Symbolic Logic 50/02, S. 510–530, 1985.

[BK14] Beierle, C.; Kern-Isberner, G.: Methoden wissensbasierter Systeme: Grundlagen, Algorithmen, Anwendungen. Springer-Verlag, 2014.

[Ge11] Gebser, M.; Grote, T.; Kaminski, R.; Schaub, T.: Reactive answer set programming. In: International Conference on Logic Programming and Nonmonotonic Reasoning. Springer, S. 54–66, 2011.

[Ge15] Gebser, M.; Kaminski, R.; Obermeier, P.; Schaub, T.: Ricochet robots reloaded: A case-study in multi-shot ASP solving. In: Advances in Knowledge Representation, Logic Programming, and Abstract Argumentation. Springer, S. 17–32, 2015.

[Ha17] Haldimann, J. P.: Reaktive Antwortmengenprogrammierung - formale Eigenschaften und logistische Anwendung, Bachelorarbeit, TU Dortmund, 2017.

[Ke17] Kern-Isberner, G.: Darstellung, Verarbeitung und Erwerb von Wissen, Vorlesungsfolien, WS 16/17, URL: https://ls1-www.cs.tu-dortmund.de/images/courses/ie/ws1617/dvew/folien/, Stand: 05. 09. 2017.

[KK12] Krümpelmann, P.; Kern-Isberner, G.: Belief base change operations for answer set programming. In: European Workshop on Logics in Artificial Intelligence. Springer, S. 294–306, 2012.