# Establish Coherence in Logic Programs Modelling Expert Knowledge via Argumentation

Andre Thevapalan[1], Jesse Heyninck[1,2] and Gabriele Kern-Isberner[1]

[1]Technische Universität Dortmund, Germany
[2]University of Cape Town and CAIR, South Africa

**Abstract**

When modelling expert knowledge, default negation is very useful, but might lead to there being no stable models. Detecting the exact causes of the incoherence in the logic program manually can become quite cumbersome, especially in larger programs. Moreover, establishing coherence does require expertise regarding the modelled knowledge as well as technical knowledge about the program and its rules. This paper introduces a method to detect the causes of incoherence in logic programs by using argumentation graphs and proposes strategies to modify the responsible parts of the program in order to obtain a coherent program.

**Keywords**

Logic Programming, Coherence, Argumentation Graphs

## 1. Introduction

With the availability of both strong and default negation, answer set programming offers valuable features for the usage in real-life applications. But often knowledge bases are subject to change. Adding rules to a logic program is not as straightforward as it might seem. Due to its declarative nature, adding new information can easily lead to contradictory knowledge. Such contradictions can result from rules whose conclusions are contradictory while both their premises can potentially hold simultaneously. Several approaches deal with this notion of *contradictions* when updating logic programs [1, 2, 3, 4, 5, 6]. However, by using default negation, another form of inconsistency can appear which is called *incoherence*. Incoherent answer set programs do not possess answer sets even though the cause of the incoherence generally comprises only one or more smaller parts of the program. Several approaches show how these causes can be found and which part of the programs are affected and thus have to be revised in order to restore the coherence of a program [7, 8, 9]. But to the best of our knowledge, there does not exist a method describing how to establish coherence, particularly in collaboration with an expert on the modelled knowledge. To disburden users from dealing with actual logic program rules and from coming up with a solution themselves, we therefore propose

*CAUSAL'21: Workshop on Causal Reasoning and Explanation in Logic Programming, September 20–27, 2021*
andre.thevapalan@tu-dortmund.de (A. Thevapalan); jesse.heyninck@tu-dortmund.de (J. Heyninck); gabriele.kern-isberner@cs.tu-dortmund.de (G. Kern-Isberner)
0000-0001-5679-6931 (A. Thevapalan); 0000-0002-3825-4052 (J. Heyninck); 0000-0001-8689-5391 (G. Kern-Isberner)
Copyright © 2021 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).
CEUR Workshop Proceedings (CEUR-WS.org)

a method to compute possible solutions that reuses the coherent program parts while modifying the incoherence-causing parts as minimally invasive as possible. For that, we adapt results on coherence in argumentation graphs [9] for normal logic programs which in turn enables us to locate those parts of the program (called *SCUPs*) that are responsible for the incoherence. We will then present different methods how the logic program can be *amended* in order to obtain a coherent logic program without performing unncessary modifications.

The paper is organized as follows. After providing some necessary preliminaries in Section 2 and 3, in Section 4, we introduce amendments for argumentation graphs and for normal logic programs and establish results in parallel between both types of amendments. The paper concludes by outlining how the presented approach can serve as part of a framework to interactively establish coherence in logic programs in Section 5, followed by a brief discussion of possible areas of application in Section 7.

## 2. Preliminaries

### 2.1. Normal Logic Programs

In this paper, we look at non-disjunctive *normal logic programs* (NLPs)[1] [10] under the (three-valued) stable semantics [11]. A NLP is a finite set of rules over a set $\mathcal{A}$ of propositional atoms. A *default-negated atom* $A$, also called *default literal*, is written as $\sim A$. A *rule* $r$ is of the form $A_0 \leftarrow A_1, \ldots, A_m, \sim A_{m+1}, \ldots, \sim A_n.$, with atoms $A_0, \ldots, A_n$ and $0 \leq m \leq n$. The atom $A_0$ is the *head* of $r$, denoted by $H(r)$, and $\{A_1, \ldots A_m, \sim A_{m+1}, \ldots \sim A_n\}$ is the *body* of $r$, denoted by $B(r)$. Furthermore, $\{A_1, \ldots, A_m\}$ is denoted by $B^+(r)$ and $\{A_{m+1}, \ldots, A_n\}$ by $B^-(r)$. A rule $r$ with $B(r) = \emptyset$ is called a *fact*, and if $H(r) = \emptyset$, rule $r$ is called a *constraint*. A rule $r$ with $B^-(r) \neq \emptyset$ will be called a *default rule*. A normal logic program $\mathcal{P}$ can then be defined as a set of rules. A set $I$ of atoms *satisfies* a rule $r$ if $H(r) \in I$ whenever $B^+(r) \subseteq I$ and $B^-(r) \cap I = \emptyset$. We say for a set of atoms $I$, rule $r$ *holds* if $I$ satisfies $r$.

A *three-valued interpretation* is a pair of sets of atoms $\langle I, I' \rangle$ with $I \cap I' = \emptyset$, where $I$ represents the atoms which are true and $I'$ represents the atoms which are false. Thus, any atom $A \notin I \cup I'$ is undecided. We will also say, for any atom $A$, $\langle I, I' \rangle(A) = \mathsf{T}$ iff $A \in I$, $\langle I, I' \rangle(A) = \mathsf{F}$ iff $A \in I'$, and $\langle I, I' \rangle(A) = \mathsf{U}$ otherwise. The *truth-order* $\preceq_t$ over $\{\mathsf{T}, \mathsf{F}, \mathsf{U}\}$ is defined as $\mathsf{F} \prec_t \mathsf{U} \prec_t \mathsf{T}$ whereas the *knowledge order* is defined by $\mathsf{U} \prec_k \mathsf{T}, \mathsf{F}$. A three-valued interpretation $J$ satisfies a positive rule $r$ iff $\min_{\preceq_t}\{J(A) \mid A \in B(r)\} \succeq_t J(H(r))$. We furthermore define a constant $\mathtt{U}$ s.t. for every interpretation $J$, $J(\mathtt{U}) = \mathsf{U}$. The *three-valued reduct* $\mathcal{P}^{\langle I, I' \rangle}$ of a program is obtained as follows:

$$\mathcal{P}^{\langle I, I' \rangle} = \{H(r) \leftarrow B^+(r). \mid r \in \mathcal{P}, B^-(r) \subseteq I'\} \cup \tag{1}$$

$$\{H(r) \leftarrow B^+(r), \mathtt{U}. \mid r \in \mathcal{P}, B^-(r) \cap I = \emptyset \text{ and } B^-(r) \setminus I' \neq \emptyset\}. \tag{2}$$

Intuitively, we add $H(r) \leftarrow B^+(r)$ to $\mathcal{P}^{\langle I, I' \rangle}$ when all negative atoms in $r$ are false, whereas we add $H(r) \leftarrow B^+(r), \mathtt{U}$ when all negative atoms in $r$ are not true, and there is some negative atoms in $r$ that is not false. We say $J$ is a stable interpretation for $\mathcal{P}$ iff $J$ is the $\preceq_t$-minimal

[1] Albeit the presented approach aims to be used with answer set programs, for the proof of concept in this paper normal logic programs will suffice.

interpretation of $\mathcal{P}^J$. $J$ is a *regular* interpretation [12] if it is stable and for every stable interpretation $J'$, $J' \preceq_k J$. A set of atoms $I$ is an *answer set* of $\mathcal{P}$ if $\langle I, \mathcal{A} \setminus I \rangle$ is a stable interpretation [10].

**Definition 1** (Coherence). *Let $\mathcal{P}$ be an NLP. $\mathcal{P}$ is* coherent *if $\mathcal{P}$ has at least one answer set. Otherwise, $\mathcal{P}$ will be called* incoherent.

**Example 1.** *The following NLP $\mathcal{P}_1$ will be used as a running example throughout the paper:*

$r_1$: $fever \leftarrow highTemp, {\sim}sunstroke.$
$r_2$: $fatigue \leftarrow lowEnergy, {\sim}stress, {\sim}workedOut.$
$r_3$: $highTemp \leftarrow tempAbove37.$ $r_4$: $sunstroke \leftarrow highTemp, {\sim}fever.$
$r_5$: $allergy \leftarrow pollenSeason, fatigue, {\sim}flu, {\sim}cold.$
$r_6$: $cold \leftarrow fatigue, soreThroat, {\sim}migraine.$
$r_7$: $flu \leftarrow fatigue, {\sim}migraine.$ $r_8$: $migraine \leftarrow headache, {\sim}allergy.$
$r_9$: $soreThroat.$ $r_{10}$: $headache.$ $r_{11}$: $pollenSeason.$ $r_{12}$: $lowEnergy.$ $r_{13}$: $tempAbove37.$

*$\mathcal{P}_1$ has no answer sets and is, therefore, incoherent. It has, however, the three-valued stable interpretations $J_1$, $J_2$ and $J_3$, where:*

$$\begin{aligned}\mathcal{F} =& \{soreThroat, headache, pollenSeason, lowEnergy, tempAbove37\}\\ J_1 =& \langle \mathcal{F} \cup \{fatigue\}, \{stress, workedOut\}\rangle\\ J_2 =& \langle \mathcal{F} \cup \{sunstroke, fatigue\}, \{fever, stress, workedOut\}\rangle\\ J_3 =& \langle \mathcal{F} \cup \{fever, fatigue\}, \{sunstroke, stress, workedOut\}\rangle\end{aligned}$$

*By way of illustration, we explain why $J_1$ is a three-valued stable interpretation. This can be seen by first calculating $\mathcal{P}_1^{J_1}$:*

$r'_1$: $fever \leftarrow highTemp, \mathtt{U}.$ $r'_2$: $fatigue \leftarrow lowEnergy.$
$r'_3$: $highTemp \leftarrow tempAbove37.$ $r'_4$: $sunstroke \leftarrow highTemp, \mathtt{U}.$
$r'_5$: $allergy \leftarrow pollenSeason, fatigue, \mathtt{U}.$
$r'_6$: $cold \leftarrow fatigue, soreThroat, \mathtt{U}.$ $r'_7$: $flu \leftarrow fatigue, \mathtt{U}.$
$r'_8$: $migraine \leftarrow headache, \mathtt{U}.$ $r_9, r_{10}, r_{11}, r_{12}, r_{13}$

*It can be easily checked that $J_1$ is the $\preceq_t$ minimal interpretation of $\mathcal{P}_1^{J_1}$.*

## 3. Argumentation Graphs

In this section, we introduce *assumption-based argumentation* [13] which will allow us to straightforwardly reason about incoherence and its causes in logic programs. The basic idea is the following: given a logic $\mathcal{P}$, an argumentation graph, which is a directed graph consisting of arguments and attacks between these arguments, is constructed by considering sets of default literals as arguments. An attack between two arguments takes place if we can derive a (positive)

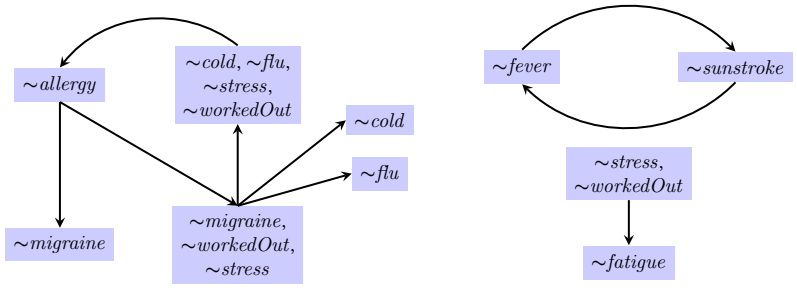



**Figure 1:** Argumentation Graph $\mathsf{AF}(\mathcal{P}_1)$ from Example 1

atom that occurs negated in the attacked set from the attacking set of default literals. Thus, an attack between two sets of default literals occurs if there is one or more sets of rules that allow to derive, from the attacking set of default literals, a positive atom in the attacked set. In other words, the intuition is that every literal is assumed to be false, until and unless we can derive a valid counter-argument against this assumption.

**Definition 2.** *Given a logic program $\mathcal{P}$, we define $\Omega_\mathcal{P} = \{\sim A \mid A \in \bigcup_{r\in\mathcal{P}} B^-(r)\}$. For some $\Delta \subseteq \Omega_\mathcal{P}$, $\vdash_\mathcal{P} \subseteq \wp(\Omega_\mathcal{P}) \times (\mathcal{A}_\mathcal{P} \cup \Omega_\mathcal{P})$ is defined inductively as follows:*

- $\Delta \vdash_\mathcal{P} \sim A$ *iff* $\sim A \in \Delta$,
- $\Delta \vdash_\mathcal{P} A$ *iff there is some* $r \in \mathcal{P}$ *s.t.* $A = H(r)$ *and* $\Delta \vdash_\mathcal{P} \phi$ *for every* $\phi \in B(r)$.

*Given $\Theta \in \wp(\Omega_\mathcal{P})$, we say* $\Theta$ is derivable from $\Delta$ by a set of rules $\Gamma \in \mathcal{P}$ *iff* $\Delta \vdash_\Gamma \Theta$.

**Example 2** (Example 1 continued)**.** *We see that for $\mathcal{P}_1$ in Example 1, the following derivations (among others) hold:*

- $\{\sim fever\} \vdash_{\mathcal{P}_1} \sim fever$ *(in view of the first item of Definition 2).*
- $\emptyset \vdash_{\mathcal{P}_1} tempAbove37.$ *and* $\emptyset \vdash_{\mathcal{P}_1} highTemp.$ *(in view of the second item of Definition 2 and since* $tempAbove37., highTemp \leftarrow tempAbove37. \in \mathcal{P}_1$*).*
- $\{\sim fever\} \vdash_{\mathcal{P}_1} sunstroke$ *(in view of the second item of Definition 2, the previous two items and* $sunstroke \leftarrow highTemp, \sim fever. \in \mathcal{P}_1$*).*

**Definition 3.** *Given a logic program $\mathcal{P}$, $\mathsf{AF}(\mathcal{P}) = \langle \wp(\Omega_\mathcal{P}), \rightsquigarrow_\mathcal{P} \rangle$ is the corresponding* argumentation graph *of $\mathcal{P}$ where $\rightsquigarrow_\mathcal{P} \subseteq \wp(\Omega_\mathcal{P}) \times \wp(\Omega_\mathcal{P})$ s.t. for any $\Delta, \Theta \subseteq \Omega_\mathcal{P}$, $\Delta \rightsquigarrow_\mathcal{P} \Theta$ iff $\Delta \vdash_\mathcal{P} A$ for some $\sim A \in \Theta$. We will also call $\sim A \in \Omega_\mathcal{P}$ an* assumption*, and sets of assumptions* arguments.

**Example 3** (Example 1 continued)**.** *The graphical representation of a fragment of the argumentation graph $\mathsf{AF}(\mathcal{P}_1)$ can be seen in Figure 1. By way of example, we explain the attack from $\{\sim fever\}$ to $\{\sim sunstroke\}$ by pointing to the fact that $\{\sim fever\} \vdash_{\mathcal{P}_1} sunstroke$ (see Example 2). Notice that we have included only the relevant fragment of the argumentation graph. To avoid clutter, attacks like $\{\sim fever\} \rightsquigarrow_{\mathcal{P}_1} \{\sim sunstroke, \sim stress\}$ were left out of Figure 1.*

*Assumption labellings* are used to give semantics to argumentation graphs. An *assumption labelling* $L : \Omega_{\mathcal{P}} \to \{\mathsf{T}, \mathsf{F}, \mathsf{U}\}$ assigns to every assumption a truth-value $\mathsf{T}$ (true), $\mathsf{F}$ (false), or $\mathsf{U}$ (undecided). We denote, for $\dagger \in \{\mathsf{T}, \mathsf{F}, \mathsf{U}\}$, $\dagger(L) = \{\sim A \in \Omega_{\mathcal{P}} \mid L(\sim A) = \dagger\}$. For a set $\Delta \subseteq \Omega_{\mathcal{P}}$ and an assumption labelling $L$, $L(\Delta) = \min_{\preceq_t}\{L(\sim A) \mid \sim A \in \Delta\}$.

**Definition 4** ([14])**.** *Let an argumentation graph* $\mathsf{AF}(\mathcal{P}) = \langle \wp(\Omega_{\mathcal{P}}), \rightsquigarrow_{\mathcal{P}} \rangle$ *and an assumption labelling* $L : \Omega \to \{\mathsf{T}, \mathsf{F}, \mathsf{U}\}$ *be given. Then* $L$ *is:*

- admissible *iff for every* $\sim A \in \Omega_{\mathcal{P}}$*, it holds that (1) if* $L(\sim A) = \mathsf{T}$ *then for every* $\Delta \subseteq \Omega_{\mathcal{P}}$ *s.t.* $\Delta \rightsquigarrow_{\mathcal{P}} \sim A$*, there is some* $\sim B \in \Delta$ *s.t.* $L(\sim B) = \mathsf{F}$*, (2) if* $L(\sim A) = \mathsf{F}$ *then there is some* $\Delta \subseteq \Omega_{\mathcal{P}}$ *s.t.* $\Delta \rightsquigarrow_{\mathcal{P}} \sim A$ *and for every* $\sim B \in \Delta$*,* $L(\sim B) = \mathsf{T}$*, (3) if* $L(\sim A) = \mathsf{U}$ *then for every* $\Delta \subseteq \Omega_{\mathcal{P}}$ *s.t.* $\Delta \rightsquigarrow_{\mathcal{P}} \sim A$*, there is some* $\sim B \in \Delta$ *s.t.* $L(\sim B) \neq \mathsf{T}$.[2]
- complete *if it is admissible and for every* $\sim A \in \Omega_{\mathcal{P}}$*, it holds that if* $L(\sim A) = \mathsf{U}$ *then there is some* $\Delta \subseteq \Omega_{\mathcal{P}}$ *s.t.* $\Delta \rightsquigarrow_{\mathcal{P}} \sim A$ *and for every* $\sim B \in \Delta$*,* $L(\sim B) \neq \mathsf{F}$*.*
- preferred *if it is admissible and* $\mathsf{T}(L)$ *is* $\subseteq$*-maximal among all admissible labellings.*
- stable *if it is admissible and iff* $\mathsf{U}(L) = \emptyset$*.*

**Example 4** (Example 3 continued)**.** *For* $\mathsf{AF}(\mathcal{P}_1)$*, there are three complete labellings,* $L_1$*,* $L_2$ *and* $L_3$*, characterised by (to avoid clutter we leave out set-brackets):*

| | | |
|---|---|---|
| $L_1(\sim fever) = \mathsf{U}$ | $L_2(\sim fever) = \mathsf{T}$ | $L_3(\sim fever) = \mathsf{F}$ |
| $L_1(\sim sunstroke) = \mathsf{U}$ | $L_2(\sim sunstroke) = \mathsf{F}$ | $L_3(\sim sunstroke) = \mathsf{T}$ |
| $L_1(\sim stress) = \mathsf{T}$ | $L_2(\sim stress) = \mathsf{T}$ | $L_3(\sim stress) = \mathsf{T}$ |
| $L_1(\sim workedOut) = \mathsf{T}$ | $L_2(\sim workedOut) = \mathsf{T}$ | $L_3(\sim workedOut) = \mathsf{T}$ |
| $L_1(\sim fatigue) = \mathsf{F}$ | $L_2(\sim fatigue) = \mathsf{F}$ | $L_3(\sim fatigue) = \mathsf{F}$ |
| $L_1(\sim allergy) = \mathsf{U}$ | $L_2(\sim allergy) = \mathsf{U}$ | $L_3(\sim allergy) = \mathsf{U}$ |
| $L_1(\sim flu) = \mathsf{U}$ | $L_2(\sim flu) = \mathsf{U}$ | $L_3(\sim flu) = \mathsf{U}$ |
| $L_1(\sim cold) = \mathsf{U}$ | $L_2(\sim cold) = \mathsf{U}$ | $L_3(\sim cold) = \mathsf{U}$ |
| $L_1(\sim migraine) = \mathsf{U}$ | $L_2(\sim migraine) = \mathsf{U}$ | $L_3(\sim migraine) = \mathsf{U}$ |

*$L_2$ and $L_3$ are also preferred. It can be noticed that there is no stable labelling for* $\mathsf{AF}(\mathcal{P}_1)$*.*

As shown in [15], complete labellings of the argumentation graph $\mathsf{AF}(\mathcal{P})$ actually give an alternative characterisation of three-valued stable interpretation. To describe this characterisation, the following method of moving between labellings of $\mathsf{AF}(\mathcal{P})$ and interpretations of $\mathcal{P}$ are useful:

**Definition 5.** *Given an assumption labelling* $L$*, the corresponding three-valued interpretation* $\texttt{Lab2Int}(L)$ *is defined as* $\texttt{Lab2Int}(L) = \langle I, I' \rangle$ *where* $I = \{A \mid L(\sim A) = \mathsf{F}\}$ *and* $I' = \{A \mid L(\sim A) = \mathsf{T}\}$*. Likewise, given a three-valued interpretation,* $\langle I, I' \rangle$*, the corresponding assumption labelling* $\texttt{Int2Lab}(\langle I, I' \rangle)$ *is defined by:*

$$\texttt{Int2Lab}(\langle I, I' \rangle)(\sim A) = \begin{cases} \mathsf{T} & \text{if } A \in I' \\ \mathsf{F} & \text{if } A \in I \\ \mathsf{U} & \text{otherwise} \end{cases}$$

[2] Notice that in assumption-based argumentation labellings [14], there is a third condition (3) for admissible labellings, which does not occur in abstract argumentation labellings.

The authors in [15] have shown the following representation result:

**Proposition 1.** *Let a normal logic program $\mathcal{P}$ be given. Then:*

- *$\langle I, I' \rangle$ is a three-valued stable interpretation of $\mathcal{P}$ iff $\texttt{Int2Lab}(\langle I, I' \rangle)$ is a complete labelling of $\mathsf{AF}(\mathcal{P})$, and vice versa: $L$ is a complete labelling of $\mathsf{AF}(\mathcal{P})$ iff $\texttt{Lab2Int}(L)$ is a three-valued stable interpretation of $\mathcal{P}$.*
- *$\langle I, I' \rangle$ is a regular interpretation of $\mathcal{P}$ iff $\texttt{Int2Lab}(\langle I, I' \rangle)$ is a preferred labelling of $\mathsf{AF}(\mathcal{P})$, and vice versa: $L$ is a preferred labelling of $\mathsf{AF}(\mathcal{P})$ iff $\texttt{Lab2Int}(L)$ is a regular interpretation of $\mathcal{P}$.*
- *$\langle I, I' \rangle$ is a stable interpretation of $\mathcal{P}$ iff $\texttt{Int2Lab}(\langle I, I' \rangle)$ is a stable labelling of $\mathsf{AF}(\mathcal{P})$, and vice versa: $L$ is a stable labelling of $\mathsf{AF}(\mathcal{P})$ iff $\texttt{Lab2Int}(L)$ is a three-valued stable interpretation of $\mathcal{P}$.*

Notice that the above proposition also implies that every preferred interpretation corresponds to a three-valued stable interpretation.

The problem of incoherence in abstract argumentation has been studied in [9]. In this work, incoherence was analysed using the notion of *strongly connected component (SCC)* [16]. A *path* from a set of assumptions $\Delta$ to a set of assumptions $\Theta$ is a sequence of sets of assumptions $\Delta_1, \ldots, \Delta_n$ s.t. $\Delta_1 = \Delta$, $\Theta = \Delta_n$ and $\Delta_i \rightsquigarrow_{\mathcal{P}} \Delta_{i+1}$ for every $1 \leq i < n$. *Path-equivalence* between two sets of assumptions $\Delta$ and $\Theta$ holds iff $\Delta = \Theta$ or there is a path from $\Delta$ to $\Theta$ and from $\Theta$ to $\Delta$. The equivalence classes of all sets of assumptions under the relation of path-equivalence are called the *strongly connected components* of $\mathsf{AF}(\mathcal{P})$. An SCC $\mathbf{\Delta} \subseteq \wp(\Omega_{\mathcal{P}})$ is an *initial SCC* if there is no SCC $\mathbf{\Theta} \subseteq \wp(\Omega_{\mathcal{P}})$ s.t. $\mathbf{\Delta} \neq \mathbf{\Theta}$ and there is some $\Theta \in \mathbf{\Theta}$ and some $\Delta \in \mathbf{\Delta}$ s.t. $\Theta \rightsquigarrow_{\mathcal{P}} \Delta$. Given a set of assumptions $\Delta \subseteq \Omega_{\mathcal{P}}$, the *restriction of* $\mathsf{AF}(\mathcal{P})$ *to* $\Delta$ is defined as $\mathsf{AF}(\mathcal{P})_{\downarrow \Delta} = \langle \Delta, \rightsquigarrow_{\mathcal{P}} \cap (\Delta \times \Delta) \rangle$.

**Definition 6.** *Given a labelling $L$, $\mathbf{\Delta} \subseteq \wp(\Omega_{\mathcal{P}})$ is a* strongly connected U-part w.r.t. *$L$ (in short, SCUP) iff $\mathbf{\Delta}$ is an initial SCC of $\mathsf{AF}(\mathcal{P})_{\downarrow \mathsf{U}(L)}$.*

The authors in [9] show that SCUPs can be seen as the responsible parts of incoherence:

**Proposition 2** (cf. [9])**.** *There exists a SCUP w.r.t. the preferred labelling $L$ iff $L$ is not stable.*

**Corollary 1** (cf. [9])**.** *$\mathcal{P}$ is incoherent iff for every preferred labelling $L$ of $\mathsf{AF}(\mathcal{P})$ there exists a SCUP.*

They also established a strong connection between SCUPS and *odd attack cycles* between arguments, which we will exploit in this paper. In more detail, a direct consequence of the results in [9] is that every SCUP contains at least one odd-length cycle (see Appendix A.1). SCUPS are, however, a more precise indication of the "cause" of incoherence, since, as we will see, incoherence can be resolved by manipulating SCUPs.

**Example 5** (Example 4 continued)**.** *For $\mathsf{AF}(\mathcal{P}_1)$ (with $\mathcal{P}_1$ as in Example 1) and $L_2$ as in Example 4, let $\Omega^{\dagger} = \{\Delta \in \Omega_{\mathcal{P}_1} \mid \Delta \cap \{\sim allergy, \sim flu, \sim cold, \sim migraine\} \neq \emptyset\}$. Then we see that $\mathsf{AF}(\mathcal{P}_1)_{\downarrow \mathsf{U}(L)} = \langle \Omega^{\dagger}, \rightsquigarrow_{\mathcal{P}_1} \cap (\Omega^{\dagger} \times \Omega^{\dagger}) \rangle$.*

*It can be observed that $\mathsf{AF}(\mathcal{P}_1)_{\downarrow \mathsf{U}(L)}$ contains one SCC, namely: $\mathbf{\Delta} = \{\Delta \subseteq \Omega_{\mathcal{P}_1} \mid \Delta \supseteq \{\sim allergy\}$ or $\Delta \supseteq \{\sim flu, \sim cold, \sim stress, \sim workedOut\}$ or $\Delta \supseteq \{\sim migraine, \sim stress, \sim workedOut\}\}$. Thus, $\mathbf{\Delta}$ is the unique SCUP w.r.t. $L_2$.*

## 4. Coherence Restoration

As pointed out before, argumentation graphs provide insight into whether a logic program is coherent or not. More precisely, SCUPs of an argumentation graph point to the exact sets of attacks between assumptions that are the cause for the incoherence. For this reason, we will describe in the following how to manipulate an argumentation graph in order to obtain an argumentation graph that is free of SCUPs and which, therefore, has a stable labelling. We will specify suitable *SCUP amendment operations*, that are basic modifications that can be used (in combination) to resolve a SCUP and thus restore coherence.

### 4.1. SCUP Amendment

In the remainder of this paper, we will assume as given an incoherent NLP $\mathcal{P}$ over $\mathcal{A}$, its corresponding argumentation graph $\mathsf{AF}(\mathcal{P}) = \langle \wp(\Omega_\mathcal{P}), \rightsquigarrow_\mathcal{P} \rangle$, and the set $\mathsf{SCUPS}(\mathcal{P})$ of all SCUPs in $\mathsf{AF}(\mathcal{P})$ w.r.t. a preferred labelling $L$ for $\mathsf{AF}(\mathcal{P})$.

**Definition 7** (Informativeness). *Let $L$ be a labelling for $\mathsf{AF}(\mathcal{P})$ and $L'$ a labelling for an argumentation graph $\mathsf{AF}^\circledast = \langle \wp(\Omega^\circledast), \rightsquigarrow^\circledast \rangle$ with $\Omega_\mathcal{P}{=}\Omega^\circledast$. We say $L'$ is* more informative *than $L$, denoted by $L \prec_\mathrm{k} L'$, if the following holds: $\mathsf{T}(L) \subseteq \mathsf{T}(L')$, $\mathsf{F}(L) \subseteq \mathsf{F}(L')$, and $\mathsf{U}(L') \subsetneq \mathsf{U}(L)$.*

**Definition 8** (SCUP Amendment). *Given a preferred labelling $L$ for $\mathsf{AF}(\mathcal{P})$ and a SCUP $\mathbf{\Delta} \in \mathsf{SCUPS}(\mathcal{P})$, a* SCUP amendment of $\mathsf{AF}(\mathcal{P})$ w.r.t. $\mathbf{\Delta}$ *is $\mathsf{AF}^\circledast = \langle \wp(\Omega^\circledast), \rightsquigarrow^\circledast \rangle$ such that*

- *$\Delta \rightsquigarrow^\circledast \Theta$ iff $\Delta \rightsquigarrow_\mathcal{P} \Theta$, and $\Theta \cap \bigcup \mathbf{\Delta} = \emptyset$ for any $\Delta, \Theta \subseteq \Omega_\mathcal{P}$.*
- *there exists a preferred labelling $L^\circledast$ for $\mathsf{AF}^\circledast$ such that*
  - *for all $\sim A \in \Omega^\circledast \backslash \Omega$: $L^\circledast(\sim A) = \mathsf{T}$ or $L^\circledast(\sim A) = \mathsf{F}$, and*
  - *$L \prec_\mathrm{k} L^\circledast$.*

*Any such preferred labelling $L^\circledast$ will be called an* amendment labelling of $\mathsf{AF}^\circledast$ w.r.t. $L$.

We explain SCUP amendments as follows: these amendments are changes to the argumentation graph $\langle \wp(\Omega_\mathcal{P}), \rightsquigarrow_\mathcal{P} \rangle$ that are induced by either adding or removing attacks on arguments within the SCUP. A minimal condition for such amendments is that the argumentation graph can only be manipulated by adding or removing attacks on arguments that have at least one assumption in common with the SCUP that is the basis of an amendment. Below, we will give specific methods that lead to SCUP amendments. The amendment should have as a result that there exists a preferred labelling which is more informative than the preferred labelling on which the amendment was based and which "resolves" the SCUP in the sense that now every member of the SCUP gets assigned a definitive label (i.e. a label other than U). From the perspective of information theory, therefore, SCUP amendments reduce the undecidedness of the knowledge represented by the argumentation graph.

As shown in [9], it is not guaranteed that there exists a single SCUP amendment $\mathsf{AF}^\circledast$ of $\mathsf{AF}(\mathcal{P})$ such that a corresponding amendment labelling $L^\circledast$ is a stable labelling.

**Definition 9** (Iterated SCUP Amendment). *Let $\mathsf{AF}^1$ be an argumentation graph with a preferred labelling $L^1$ and at least one SCUP $\mathbf{\Delta}$ w.r.t. $L^1$. An* iterated SCUP amendment *is a sequence $\langle \mathsf{AF}^1, \mathsf{L}^1 \rangle, \ldots, \langle \mathsf{AF}^\mathsf{n}, \mathsf{L}^\mathsf{n} \rangle (\mathsf{n} > 1)$ such that for all $i$ $(1 \leq i < n)$, $\mathsf{AF}^{\mathsf{i}+1}$ is a SCUP amendment of $\mathsf{AF}^\mathsf{i}$ w.r.t. $\mathbf{\Delta}$ and $L^{i+1}$ is an amendment labelling of $\mathsf{AF}^{\mathsf{i}+1}$ w.r.t. $L^i$.*

Next, we will propose different methods on how to manipulate a given argumentation graph in order to obtain a suitable SCUP amendment.

### 4.2. SCUP Amendment Operations

Informally, a SCUP of an argumentation graph $\mathsf{AF}$ contains one or more interdependent odd cycles such that all preferred labellings label all assumptions of these *responsible cycles* as undecided. Notice the importance of the preferred labellings for restoring coherence, as they are "as close" (according to $\leq_i$) to stable labellings as possible. Thus, resolving a SCUP requires the manipulation of the graph such that there exists a preferred labelling $L^{\circledast}$ for the modified graph that is more informative, i. e., where at least one of the arguments that is part of the cycle is labelled either true or false by $L^{\circledast}$. Suppose one of the responsible odd cycles contains the assumption $\sim a$. The most obvious way to obtain such a preferred labelling $L^{\circledast}$ is to force an assumption of a responsible cycle to be true or false, for example to demand that $a$ is true. In argumentation graphs, such an enforcement operation can be implemented by adding an attack from the empty set to $\sim a$ (which, since then $\emptyset \vdash_{\mathcal{P}} a$, in turn corresponds to adding a fact $a$. in logic programs). Other ways to obtain a more informative labelling $L^{\circledast}$ and ultimately a SCUP-free argumentation graph is to modify the responsible odd cycles and manipulate their attacks such that they contain *even subcycles* or break up the responsible cycles by *removing one or more attacks* of said cycles. These preliminary considerations lead us to the following *SCUP amendment operations.*

Given a SCUP $\mathbf{\Delta} \in \mathsf{SCUPS}(\mathcal{P})$ w.r.t. a preferred labelling $L$, the following SCUP amendment operations can be used to (eventually) resolve SCUP $\mathbf{\Delta}$:

**Enforcement** The SCUP amendment operation *(SCUP) enforcement* consists of the manipulation of $\mathsf{AF}$ to $\mathsf{AF}^{\circledast} = \langle \wp(\Omega_{\mathcal{P}}), \rightsquigarrow^{\circledast} \rangle$ w.r.t. $\mathbf{\Delta}$ where $\rightsquigarrow^{\circledast} = \rightsquigarrow_{\mathcal{P}} \cup \{\emptyset \rightsquigarrow^{\circledast} \Delta \mid \sim A \in \Delta\}$ such that $\sim A \in \bigcup \mathbf{\Delta}$.[3]

**Evening Up** The SCUP amendment operation *(SCUP) evening up* consists of the manipulation of $\mathsf{AF}$ to $\mathsf{AF}^{\circledast} = \langle \wp(\Omega_{\mathcal{P}}), \rightsquigarrow^{\circledast} \rangle$ w.r.t. $\mathbf{\Delta}$ where $\rightsquigarrow^{\circledast} = \rightsquigarrow_{\mathcal{P}} \cup \{\Delta \rightsquigarrow^{\circledast} \Theta \mid \sim A \in \Theta\}$ with $\Delta \in \mathbf{\Delta}$, $\sim A \in \bigcup \mathbf{\Delta}$ and $\sim A \notin \Delta$.

**Pacification** The SCUP amendment operation *(SCUP) pacification* consists of the manipulation of $\mathsf{AF}$ to $\mathsf{AF}^{\circledast} = \langle \wp(\Omega_{\mathcal{P}}), \rightsquigarrow^{\circledast} \rangle$ w.r.t. $\mathbf{\Delta}$ such that $\rightsquigarrow^{\circledast} = \rightsquigarrow_{\mathcal{P}} \setminus \{\Delta' \rightsquigarrow_{\mathcal{P}} \Theta \mid \sim A \in \Theta, \Delta' \subseteq \Delta\}$ with $\Delta \in \mathbf{\Delta}$ and $\sim A \in \bigcup \mathbf{\Delta}$.

**Proposition 3.** *There exist argumentation graphs for which: (1) A single application of a SCUP pacification operation might not always lead to a SCUP amendment; (2) no number of applications of evening up leads to a SCUP amendment of* $\mathsf{AF}$ *w.r.t.* $\mathbf{\Delta}$.

*Proof.* Ad 1: this is shown by use of the following example. Consider the following $\mathsf{AF}_1$ drawn in Figure 2a. This argumentation graph has a single complete extension:

$$L(\sim a) = L(\sim b) = L(\sim c) = \mathsf{U}.$$

[3] A more fine-grained version of enforcement would be to allow for any attacks from arguments that are labelled $\mathsf{T}$ by the preferred labelling on which the SCUP amendment is based.


(a) $\mathsf{AF}_1$ (b) $\mathsf{AF}_2$

**Figure 2:** Visualization of argumentation graphs used in proof of Proposition 3

It can be easily checked that any single application of pacification does not result in any change to the complete labellings. For example, deleting $\{\{\sim a\} \rightsquigarrow \Delta \mid \sim b \in \Delta\}$ or $\{\{\sim b\} \rightsquigarrow \Delta \mid \sim b \in \Delta\}$ does not result in a more informative complete labelling.

Ad 2: this is shown by use of the following example. Consider the following argumentation graph $\mathsf{AF}_1$ drawn in Figure 2a. If we apply evening up as much as possible, we end up with the argumentation graph $\mathsf{AF}_2$ drawn in Figure 2b, which has the same unique complete labelling as $\mathsf{AF}_1$:

$$L(\sim a) = L(\sim b) = L(\sim c) = \mathsf{U}.$$

□

However, one can always iterate the amendment operations of enforcement and pacification to obtain a SCUP amendment.

**Proposition 4.** *For any argumentation graph* $\mathsf{AF}$ *with a preferred labelling* $L$ *and a SCUP w.r.t.* $L$*: (1) there exists a SCUP amendment of* $\mathsf{AF}$ *w.r.t.* $\boldsymbol{\Delta}$ *that consists of a sequence of SCUP enforcement operations, and (2) there exists a SCUP amendment of* $\mathsf{AF}$ *w.r.t.* $\boldsymbol{\Delta}$ *that consists of a sequence of SCUP pacification operations.*[4]

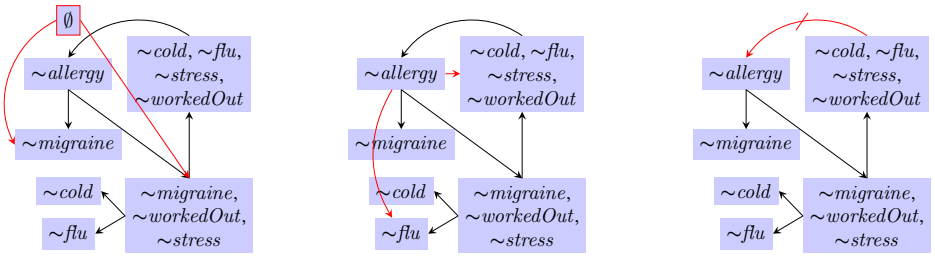
For a SCUP $\boldsymbol{\Delta}$ without self-attacking arguments, evening up is also an adequate amendment operation:

**Proposition 5.** *Let an argumentation graph* $\mathsf{AF}$ *with a preferred labelling* $L$ *and a SCUP* $\boldsymbol{\Delta}$ *w.r.t.* $L$ *s.t. for no* $\Delta \in \boldsymbol{\Delta}$, $\Delta \rightsquigarrow \Delta$*. Then there exists a SCUP amendment of* $\mathsf{AF}$ *w.r.t.* $\boldsymbol{\Delta}$ *that consists of a sequence of SCUP evening up operations.*

**Proposition 6.** *For any finite argumentation graph* $\mathsf{AF}$ *with a preferred labelling* $L$ *and at least one SCUP w.r.t.* $L$*, there exists an iterated SCUP amendment* $\langle \mathsf{AF}, \mathsf{L}\rangle, \ldots, \langle \mathsf{AF}^n, \mathsf{L}^n\rangle$ *s.t.* $L^n$ *is a stable labelling.*

**Corollary 2.** *For any incoherent NLP* $\mathcal{P}$ *and a preferred labelling* $L$ *of* $\mathsf{AF}(\mathcal{P})$*, there exists an iterated SCUP amendment* $\langle \mathsf{AF}(\mathcal{P}), \mathsf{L}\rangle, \ldots, \langle \mathsf{AF}^n, \mathsf{L}^n\rangle$ *s.t. any NLP* $\mathcal{P}^n$ *that induces* $\mathsf{AF}^n$ *is coherent.*

[4] In view of spatial limitations, proofs and further details have been left out but can be found in an online appendix under http://tinyurl.com/thki2021-appndx.




(a) $\mathsf{AF}(\mathcal{P}_1)$ after SCUP enforcement (b) $\mathsf{AF}(\mathcal{P}_1)$ after SCUP evening up (c) $\mathsf{AF}(\mathcal{P}_1)$ after SCUP pacification

**Figure 3:** Visualization of SCUP amendment operations in Example 6

**Example 6** (Example 4 continued). *Consider the NLP $\mathcal{P}_1$, again. $\mathsf{AF}(\mathcal{P}_1)$ has the unique SCUP*

$$\begin{aligned}\mathbf{\Delta} = \{\Delta \subseteq \wp(\Omega_{\mathcal{P}_1}) \mid\ & \Delta \supseteq \{\sim allergy\} \text{ or} \\ & \Delta \supseteq \{\sim flu, \sim cold, \sim stress, \sim workedOut\} \text{ or} \\ & \Delta \supseteq \{\sim migraine, \sim stress, \sim workedOut\}\}.\end{aligned}$$

*Adding the set of attacks $\Gamma_1 = \{\emptyset \rightsquigarrow^{\circledast} \Delta \mid \sim migraine \in \Delta, \Delta \in \mathbf{\Delta}\}$ to $\mathsf{AF}(\mathcal{P}_1)$ would constitute a SCUP enforcement operation and thus $\mathsf{AF}(\mathcal{P}_1^{\circledast}) = \langle \wp(\Omega_{\mathcal{P}_1}), \rightsquigarrow^{\circledast} \rangle$ would be a SCUP amendment w.r.t. $\mathbf{\Delta}$ where $\rightsquigarrow^{\circledast} = \rightsquigarrow \cup \Gamma_1$.*

*Similarly, adding the set of attacks $\Gamma_2 = \{\{\sim allergy\} \rightsquigarrow^{\circledast} \Delta \mid \sim flu \in \Delta, \Delta \in \mathbf{\Delta}\}$ to $\mathsf{AF}(\mathcal{P}_1)$ would constitute a SCUP evening up operation. Again, $\mathsf{AF}(\mathcal{P}_1^{\circledast}) = \langle \wp(\Omega_{\mathcal{P}_1}), \rightsquigarrow^{\circledast} \rangle$ would be a SCUP amendment w.r.t. $\mathbf{\Delta}$ where $\rightsquigarrow^{\circledast} = \rightsquigarrow \cup \Gamma_2$.*

*Finally, $\Gamma_3 = \{\{\sim cold, \sim flu, \sim stress, \sim workedOut\} \rightsquigarrow^{\circledast} \Theta \mid \sim allergy \in \Theta, \Theta \in \mathbf{\Delta}\}$ being removed from $\mathsf{AF}(\mathcal{P}_1)$ constitutes a SCUP pacification operation. Here, $\mathsf{AF}(\mathcal{P}_1^{\circledast}) = \langle \wp(\Omega_{\mathcal{P}_1}), \rightsquigarrow^{\circledast} \rangle$ would be a SCUP amendment w.r.t. $\mathbf{\Delta}$ where $\rightsquigarrow^{\circledast} = \rightsquigarrow \setminus \Gamma_1$. The resulting changes in $\mathsf{AF}(\mathcal{P}_1)$ illustrated in Figure 3.*

Thus, the following methodology for resolving incoherence in an NLP $\mathcal{P}$ has been introduced: first, the argumentation framework $\mathsf{AF}(\mathcal{P})$ is constructed. An iterated SCUP amendment is then applied to this argumentation framework, which consists of a number of SCUP amendments, each of which on its turn consists of a series of applications of SCUP amendment operations of the user's choice. Thus, SCUP amendment operations can be seen as the basic building blocks in our framework for coherence restoration. In the next section, we show how these SCUP amendment operations correspond to changes of the logic program $\mathcal{P}$. With Corollary 2, such changes to the logic program $\mathcal{P}$ can then be iterated to restore coherence of $\mathcal{P}$.

### 4.3. NLP Amendments

By definition, an NLP $\mathcal{P}$ induces a unique argumentation graph $\mathsf{AF}(\mathcal{P})$, whereas $\mathsf{AF}(\mathcal{P})$ can correspond to a vast number of logic programs. We have shown how one can modify an argumentation graph $\mathsf{AF}(\mathcal{P})$ of an incoherent NLP $\mathcal{P}$ via iterated SCUP amendment such that the final argumentation graph $\mathsf{AF}^n$ has a stable labelling. This means, any NLP $\mathcal{P}^n$ that induces

$\mathsf{AF}^{\mathsf{n}}$ is coherent. In this section, we will show how the execution of each amendment operation in $\mathsf{AF}(\mathcal{P})$ can be implemented in $\mathcal{P}$ accordingly.

Let $\mathcal{P}_{\boldsymbol{\Delta}}$ denote the set of rules corresponding to $\boldsymbol{\Delta} \in \mathsf{SCUPS}(\mathcal{P})$, i. e., $\mathcal{P}_{\boldsymbol{\Delta}} = \bigcup\{\Psi \mid \Delta \vdash_{\Psi} A$ for some $\Delta \in \boldsymbol{\Delta}$ and some $\sim A \in \bigcup \boldsymbol{\Delta}\}$, and let $\mathcal{P}_{\tau} = \mathcal{P} \backslash \mathcal{P}_{\boldsymbol{\Delta}}$ be the set of all other rules in $\mathcal{P}$.

**Enforcement in $\boldsymbol{\mathcal{P}}$** Enforcement in $\mathcal{P}$ modifies $\mathcal{P} = \mathcal{P}_{\boldsymbol{\Delta}} \cup \mathcal{P}_{\tau}$ to $\mathcal{P}^{\circledast} = \mathcal{P}_{\boldsymbol{\Delta}'} \cup \mathcal{P}_{\tau}$ such that $\mathcal{P}_{\boldsymbol{\Delta}'} = \mathcal{P}_{\boldsymbol{\Delta}} \cup \{L.\}$ and there exists a rule $r \in \mathcal{P}_{\boldsymbol{\Delta}}$ with $L \in B^{-}(r)$. In that case we also say that $\mathcal{P}^{\circledast}$ is *obtained by application of the program amendment operation enforcement.*

**Evening Up in $\boldsymbol{\mathcal{P}}$** Let $A$ be an atom such that there exists a set $\Theta \in \boldsymbol{\Delta}$ with $\sim A \in \Theta$, and let $\Delta \in \boldsymbol{\Delta}$ be a set in $\boldsymbol{\Delta}$ such that $\sim A \notin \Delta$. Evening up in $\mathcal{P}$ modifies $\mathcal{P} = \mathcal{P}_{\boldsymbol{\Delta}} \cup \mathcal{P}_{\tau}$ to $\mathcal{P}^{\circledast} = \mathcal{P}_{\boldsymbol{\Delta}'} \cup \mathcal{P}_{\tau}$ such that $\mathcal{P}_{\boldsymbol{\Delta}'} = \mathcal{P}_{\boldsymbol{\Delta}} \cup \{r\}$ with $H(r) = A$ and $B(r) = \Delta$. In that case we also say that $\mathcal{P}^{\circledast}$ is *obtained by application of the program amendment operation evening up.*

**Pacification in $\boldsymbol{\mathcal{P}}$** Let $r \in \mathcal{P}_{\boldsymbol{\Delta}}$ be a rule in $\mathcal{P}_{\boldsymbol{\Delta}}$ such that $H(r) = A$ and $\sim A$ an assumption in $\mathcal{P}_{\boldsymbol{\Delta}}$. Let, furthermore, $\boldsymbol{\Gamma} = \{\Gamma \subseteq \mathcal{P}_{\boldsymbol{\Delta}} \mid \Delta \vdash_{\Gamma} A\}$ with $\Delta \in \mathcal{P}_{\boldsymbol{\Delta}}$ be the set of all possible derivations of $A$ from some $\Delta$ in $\mathcal{P}_{\boldsymbol{\Delta}}$. Pacification in $\mathcal{P}$ modifies $\mathcal{P} = \mathcal{P}_{\boldsymbol{\Delta}} \cup \mathcal{P}_{\tau}$ to $\mathcal{P}^{\circledast} = \mathcal{P}_{\boldsymbol{\Delta}'} \cup \mathcal{P}_{\tau}$ by removing at least one rule of every derivation $\Gamma \in \boldsymbol{\Gamma}$ , i. e. $\mathcal{P}_{\boldsymbol{\Delta}'} = \mathcal{P}_{\boldsymbol{\Delta}} \backslash C$, where $C$ is a hitting set of $\boldsymbol{\Gamma}$ ($C$ contains at least one rule of every set in $\boldsymbol{\Gamma}$).[5] In that case we also say that $\mathcal{P}^{\circledast}$ is *obtained by application of the program amendment operation pacification.*

**Proposition 7.** *Let a program $\mathcal{P}$, a preferred labelling $L$ w.r.t. $\mathsf{AF}(\mathcal{P})$, a SCUP $\boldsymbol{\Delta}$ w.r.t. $\mathsf{AF}(\mathcal{P})$ and $L$ be given, and let $\mathcal{P}^{\circledast}$ be obtained by application of the program amendment operation enforcement, evening up or pacification w.r.t. $\boldsymbol{\Delta}$. Then $\mathsf{AF}(\mathcal{P}^{\circledast})$ constitutes an application of the respective SCUP amendment operation to $\mathsf{AF}(\mathcal{P})$ w.r.t. $\boldsymbol{\Delta}$. Conversely, for every application of the SCUP amendment operation enforcement, evening up or pacification to $\mathsf{AF}(\mathcal{P})$ w.r.t. $\boldsymbol{\Delta}$, there exists an application of the respective program amendment operation to $\mathcal{P}$ resulting in a program $\mathcal{P}^{\circledast}$ such that $\mathsf{AF}^{\circledast} = \mathsf{AF}(\mathcal{P}^{\circledast})$, where $\mathsf{AF}^{\circledast}$ is the argumentation graph resulting from applying the mentioned SCUP amendment operation to $\mathsf{AF}(\mathcal{P})$.*

**Definition 10.** *Given a program $\mathcal{P}$, an* iterated program amendment of $\mathcal{P}$ *is a sequence of programs $\langle \mathcal{P}^1, \mathcal{P}^2, \ldots, \mathcal{P}^n \rangle$ s.t. $\mathcal{P}^1 = \mathcal{P}$ and for every $1 \leq i < n$, $\mathcal{P}^{i+1}$ is obtained by application of the program amendment operation enforcement, evening up or pacification.*

**Corollary 3.** *For any incoherent NLP $\mathcal{P}$ there exists an iterated program amendment $\langle \mathcal{P}, \mathcal{P}^1, \ldots, \mathcal{P}^n \rangle$ s.t. $\mathcal{P}^n$ is coherent.*

**Example 7** (Example 6 continued)**.** *Consider the NLP $\mathcal{P}_1$, again. Let $\mathcal{P}_{\boldsymbol{\Delta}} = \{r_2, r_5, r_6, r_7, r_8\}$ be the set of rules corresponding to the unique SCUP $\boldsymbol{\Delta}$ in $\mathcal{P}_1$ (see Example 6).*

*Adding the fact $migraine.$ would then constitute a program enforcement operation in $\mathcal{P}_1$. Let $\mathcal{F}$= {soreThroat, headache, pollenSeason, lowEnergy, tempAbove37}. With $\mathcal{P}_1^{\circledast} = \mathcal{P}_1 \cup \{migraine.\}$, we get $AS(\mathcal{P}_1^{\circledast}) = \{\mathcal{F} \cup \{highTemp, fatigue, migraine, allergy, sunstroke\}\}$.*

[5] A more fine-grained way to alter $\mathcal{P}_{\boldsymbol{\Delta}}$ would be to subsequently replace the removed rule $r$ with a modified rule $r'$ where $H(r') = H(r)$ and $B(r') \neq B(r)$. An analysis and thorough description of the properties that a new rule $r'$ should possess such that the addition of $r'$ eventually leads to a SCUP amendment would be beyond the scope of this paper and will thus be examined in future work.

*Similarly, adding the rule* $r$: *flu* $\leftarrow tempAbove37, \sim allergy.$ *to* $\mathcal{P}_1$ *constitutes a program evening up operation in* $\mathcal{P}_1$. *With* $\mathcal{P}_1^{\circledast\prime} = \mathcal{P}_1 \cup \{r\}$, *we get* $AS(\mathcal{P}_1^{\circledast\prime}) = \{\mathcal{F} \cup \{highTemp, fatigue, fever, flu, migraine\}, \mathcal{F} \cup \{highTemp, fatigue, sunstroke, flu, migraine\}\}$.

*Now, let* $\mathcal{P}_1^{\circledast\prime\prime} = \mathcal{P}_1 \backslash \{r_5\}$ *be the program* $\mathcal{P}_1$ *without rule* $r_5$. *This constitutes a program pacification in* $\mathcal{P}_1$. *We get* $AS(\mathcal{P}_1^{\circledast\prime\prime}) = \{\mathcal{F} \cup \{highTemp, fatigue, migraine, fever\}, \mathcal{F} \cup \{highTemp, fatigue, migraine, sunstroke\}\}$.

In contrast to the program amendment operations pacification and enforcement, evening up was only defined in a purely declarative way. It is easy to see, that given incoherent NLP $\mathcal{P}$, there can exist a vast amount of ways to apply evening up in $\mathcal{P}$ in order to obtain a coherent program $\mathcal{P}^{\circledast}$. In order to illustrate the principle behind the evening up operation, we now propose the constructive definition of one possible evening up implementation.

**Evening Up by Contraposition** The contraposition of a default rule $r$ is a rule $r'$ where $H(r') = A$ such that $\sim A \in B(r)$, and $B(r') = B(r) \setminus \{\sim A\} \cup \{\sim B\}$ such that $H(r) = B$. Evening up by contraposition consists of adding such a contrapositional rule $r'$ of a rule $r \in \mathcal{P}_{\mathbf{\Delta}}$ to $\mathcal{P}_{\mathbf{\Delta}}$.

**Example 8** (Example 6 continued)**.** *For* $\mathcal{P}_1$, *the rule* $r_8'$: $allergy \leftarrow headache, \sim migraine.$ *would be a possible contrapositional rule for* $r_8$ *and adding* $r_8'$ *to* $\mathcal{P}_{\mathbf{\Delta}}$ *would be a SCUP amendment that would lead to a coherent program.*

## 5. Using Amendments in Applications

Given an incoherent program $\mathcal{P}$, the presented approach can lead to a vast amount of possible solutions to obtain a coherent program $\mathcal{P}^{\circledast}$. As mentioned before, deciding which solution is the most adequate for a given application context requires expert knowledge. We, therefore, propose that the SCUP resolution approach is implemented as part of an interactive framework similar to the framework for conflict resolution in [17]. Such a framework can then be used to resolve all SCUPs in $\mathcal{P}$ in interaction with an expert on the knowledge that is modelled in the given program. This expert does not necessarily have to be familiar with logic programming and with coherence in logic programs in particular. Thus, the implementation of such a framework has to perform two main functions: On the one hand, it has to be able to *explain* the exact causes of the incoherence to the expert in a way that is understandable without the technical knowledge surrounding the concept of coherence in logic programs. On the other hand, the framework has to gradually assess in interaction with the expert what part of the knowledge was initially modelled wrongly or insufficiently respectively in order to choose the most suitable solution. The interactive aspect of the framework should consist of asking the user the *right* questions such that they can make the most informed decision. For this, we propose the usage of *argumentative dialogue theory* (see e.g. [18, 19, 20]). Both the causes and possible solutions for incoherence can be explained in an interactive way using argumentative dialogue theory. In these proof theories, the label of an argument can be explained in a dialogue held between the computer and the human-user, where the human-user can ask questions about the status of an argument (e.g. "why is *fatigue* not rejected"?) which the computer then answers by pointing to a counter-argument (e.g. "since we can derive *fatigue* in view of $r_2$). Formally,

these dialogue games consist of a proponent (the computer) and an opponent (the human-user) exchanging arguments and counter-arguments based on the argumentation graph $\mathsf{AF}(\mathcal{P})$. One possible implementation would be to generate not only the possible solutions regarding the SCUP resolution but also a corresponding dialogue tree where each path represents a possible dialogue between the framework and the expert and eventually leads to a coherent program.

## 6. Related Work

The presented approach is similar to methods developed and investigated in the area of *ASP debugging*. Essentially, debugging approaches in [21, 22, 23] aim to modify knowledge bases of any (not necessarily inconsistent) logic programs such that they represent the intended expert knowledge, i. e., they intend to remedy a mismatch between the actual semantics of the program and the semantics intended by the modeller. Generally, the ability to identify the errors in a given program and to compute suggestions crucially depends on information by the expert that is given on top of the original input program. Our approach, however, focuses on a specific class of inconsistent programs and exploits the properties of incoherence. Here, the original program is by itself sufficient to identify the problem causes and to generate suitable solution suggestions. Once possible solutions are available, both, the presented method as well as debugging approaches like those based on the meta-programming [21] technique can be used to successively obtain the most suitable solution in interaction with the user.

## 7. Discussion

We have shown how argumentation theory can be used to establish coherence in an NLP. The presented approach shows different options regarding how the responsible parts of the program can be modified in order to gradually obtain a coherent program. Such an approach is not only useful to analyze the different properties of the possible solutions that resolve the incoherent parts of the program. It can also serve as a motivation to consider using logic programs in real-life applications in numerous areas where not only complex decision processes take place but where the required knowledge is often subject to changes and updates. Embedding the presented approach in a suitable framework as discussed in Section 5 would, therefore, provide a useful extension for decision support systems that are based on logic programs [24], but it also opens up the possibility to use logic program in areas where one is faced with highly contrained problems and continously evolving knowledge like logistics or the medical sector.

## References


[1] T. Eiter, M. Fink, G. Sabbatini, H. Tompits, On properties of update sequences based on causal rejection, Theory Pract. Log. Program. 2 (2002) 711–767.

[2] F. Buccafurri, W. Faber, N. Leone, Disjunctive logic programs with inheritance, in: D. D. Schreye (Ed.), Logic Programming: The 1999 International Conference, Las Cruces, New Mexico, USA, November 29 - December 4, 1999, MIT Press, 1999, pp. 79–93.

[3] J. A. Leite, L. M. Pereira, Generalizing updates: From models to programs, in: J. Dix, L. M. Pereira, T. C. Przymusinski (Eds.), Logic Programming and Knowledge Representation, Third International Workshop, LPKR '97, Port Jefferson, New York, USA, October 17, 1997, Selected Papers, volume 1471 of *Lecture Notes in Computer Science*, Springer, 1997, pp. 224–246.
[4] V. W. Marek, M. Truszczynski, Revision programming, Theor. Comput. Sci. 190 (1998) 241–277.
[5] C. Sakama, K. Inoue, Updating extended logic programs through abduction, in: M. Gelfond, N. Leone, G. Pfeifer (Eds.), Logic Programming and Nonmonotonic Reasoning, 5th International Conference, LPNMR'99, El Paso, Texas, USA, December 2-4, 1999, Proceedings, volume 1730 of *Lecture Notes in Computer Science*, Springer, 1999, pp. 147–161.
[6] Y. Zhang, N. Y. Foo, Updating logic programs, in: H. Prade (Ed.), 13th European Conference on Artificial Intelligence, Brighton, UK, August 23-28 1998, Proceedings., John Wiley and Sons, 1998, pp. 403–407.
[7] S. Costantini, B. Intrigila, A. Provetti, Coherence of updates in answer set programming, in: In Proc. of the IJCAI-2003 Workshop on Nonmonotonic Reasoning, Action and Change, 2003, pp. 66–72.
[8] S. Costantini, On the existence of stable models of non-stratified logic programs, Theory Pract. Log. Program. 6 (2006) 169–212.
[9] C. Schulz, F. Toni, On the responsibility for undecisiveness in preferred and stable labellings in abstract argumentation, Artif. Intell. 262 (2018) 301–335.
[10] M. Gelfond, V. Lifschitz, Classical negation in logic programs and disjunctive databases, New Gener. Comput. 9 (1991) 365–386.
[11] T. C. Przymusinski, The well-founded semantics coincides with the three-valued stable semantics, Fundam. Inform. 13 (1990) 445–463.
[12] J.-H. You, L. Y. Yuan, Three-valued formalization of logic programming: is it needed?, in: Proceedings of the ninth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems, 1990, pp. 172–182.
[13] P. M. Dung, R. A. Kowalski, F. Toni, Assumption-based argumentation, in: Argumentation in artificial intelligence, Springer, 2009, pp. 199–218.
[14] C. Schulz, F. Toni, Labellings for assumption-based and abstract argumentation, International Journal of Approximate Reasoning 84 (2017) 110–149.
[15] M. Caminada, C. Schulz, On the equivalence between assumption-based argumentation and logic programming, Journal of Artificial Intelligence Research 60 (2017) 779–825.
[16] P. Baroni, M. Giacomin, On the role of strongly connected components in argumentation, in: Proceedings of the 10th International Conference on Information Processing and Management of Uncertainty inKnowledge-Based Systems (IPMU 2004), Perugia, Italy, 2004, p. 1887–1894.
[17] A. Thevapalan, G. Kern-Isberner, Towards interactive conflict resolution in asp programs, in: M. Martinez, I. Varcinczak (Eds.), Proceedings of the 18th International Workshop on Non-Monotonic Reasoning, NMR 2020, 2020, pp. 29–36.
[18] S. Modgil, M. Caminada, Proof theories and algorithms for abstract argumentation frameworks, in: Argumentation in artificial intelligence, Springer, 2009, pp. 105–129.
[19] M. Caminada, Argumentation semantics as formal discussion, Journal of Applied Logics 4

(2017) 2457–2492.

[20] D. Walton, E. C. Krabbe, Commitment in dialogue: Basic concepts of interpersonal reasoning, SUNY press, 1995.

[21] M. Gebser, J. Pührer, T. Schaub, H. Tompits, A meta-programming technique for debugging answer-set programs, in: D. Fox, C. P. Gomes (Eds.), Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008, AAAI Press, 2008, pp. 448–453.

[22] J. Oetsch, J. Pührer, H. Tompits, Stepping through an answer-set program, in: J. P. Delgrande, W. Faber (Eds.), Logic Programming and Nonmonotonic Reasoning - 11th International Conference, LPNMR 2011, Vancouver, Canada, May 16-19, 2011. Proceedings, volume 6645 of *Lecture Notes in Computer Science*, Springer, 2011, pp. 134–147.

[23] K. M. Shchekotykhin, Interactive query-based debugging of ASP programs, in: B. Bonet, S. Koenig (Eds.), Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA, AAAI Press, 2015, pp. 1597–1603.

[24] A. Thevapalan, G. Kern-Isberner, D. Howey, C. Beierle, R. G. Meyer, M. Nietzke, Decision support core system for cancer therapies using ASP-HEX, in: K. Brawner, V. Rus (Eds.), Proceedings of the Thirty-First International Florida Artificial Intelligence Research Society Conference, FLAIRS 2018, Melbourne, Florida, USA. May 21-23 2018, AAAI Press, 2018, pp. 531–536.