

# Rule-based Graph Repair using Minimally Restricted Consistency-Improving Transformations

Alexander Lauer

January 25, 2023

## **Abstract**

# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Related Work</b>	<b>3</b>
<b>3</b>	<b>Preliminaries</b>	<b>3</b>
3.1	Graphs and Graph morphisms . . . . .	3
3.2	Nested Graph Conditions and Constraints . . . . .	4
3.3	Rules and Graph Transformations . . . . .	7
3.4	Concepts of Consistency . . . . .	9
<b>4</b>	<b>Consistency Increase and Maintenance</b>	<b>10</b>
4.1	Universally quantified ANF . . . . .	11
4.2	Conditions up to Layer . . . . .	11
4.3	Consistency Increasing and Maintaining Transformations and Rules . . . . .	16
4.4	Direct Consistency Maintaining and Increasing Transformations . . . . .	19
4.5	Comparison with other concepts of Consistency . . . . .	23
<b>5</b>	<b>Application Conditions</b>	<b>27</b>
5.1	General Application Conditions . . . . .	28
5.2	Basic Increasing and Maintaining Rules . . . . .	34
5.3	Application Conditions for Basic Rules . . . . .	40
<b>6</b>	<b>Rule-based Graph Repair</b>	<b>42</b>
6.1	Conflicts within Conditions . . . . .	42
6.2	Repairing rule Sets . . . . .	46
6.3	Construction of Repairing Sets . . . . .	49
6.4	Rule-based Graph Repair for one Constraint . . . . .	51
6.5	Rule-based Graph Repair for multiple Constraints . . . . .	54
<b>7</b>	<b>Conclusion</b>	<b>58</b>

## 1 Introduction

## 2 Related Work

## 3 Preliminaries

Our graph repair process is based on the concept of the double-pushout approach [1]. In this chapter we introduce some formal prerequisites such as graphs, graph morphisms, nested graph conditions and constraints, and graph transformations.

### 3.1 Graphs and Graph morphisms

We start by introducing graphs and graph morphisms according to [1].

**Definition 3.1 (graph).** A graph  $G = (V, E, \text{src}, \text{tar})$  consists of a set of vertices (or nodes)  $V$ , a set of edges  $E$  and two mappings  $\text{src}, \text{tar} : E \rightarrow V$  that assign the source and target vertices to an edge. The edge  $e \in E$  connects the vertices  $\text{tar}(e)$  and  $\text{src}(e)$ .

If no tuple as above is given,  $V_G$ ,  $E_G$ ,  $\text{tar}_G$  and  $\text{src}_G$  denote the sets of vertices, edges and target and source mappings, respectively.

For the rest of this paper we will assume that all graphs are finite, i.e. given a graph  $G$ , the sets  $V_G$  and  $E_G$  are finite.

**Definition 3.2 (graph morphism).** Let the graphs  $G$  and  $H$  be given. A graph morphism  $f : G \rightarrow H$  consists of two mappings  $f_V : V_G \rightarrow V_H$  and  $f_E : E_G \rightarrow E_H$  such that the source and target functions are preserved. This means

$$\begin{aligned} f_V \circ \text{src}_G &= \text{src}_H \circ f_E \wedge \\ f_V \circ \text{tar}_G &= \text{tar}_H \circ f_E \end{aligned}$$

holds. A graph morphism  $f$  is called injective (surjective) if  $f_E$  and  $f_V$  are injective (surjective) mappings. If  $f$  is injective, it is denoted with  $f : G \hookrightarrow H$ . Two morphisms  $f_1 : G_1 \rightarrow H$  and  $f_2 : G_2 \rightarrow H$  are called jointly surjective if for each element  $e$  of  $H$  either an element  $e' \in G_1$  with  $f_1(e') = e$  or an element  $e' \in G_2$  with  $f_2(e') = e$  exists.

For our newly introduced notions of consistency increase and maintenance, we also need to consider *subgraphs*, *overlaps* of graphs, and so-called *intermediate graphs*. Intuitively, intermediate graphs are graphs  $G'$  which lie between two given graphs  $G$  and  $H$ . That is,  $G$  is a subgraph of  $G'$  and  $G'$  is a subgraph of  $H$ .

**Definition 3.3 (subgraph).** Let the graphs  $G$  and  $H$  be given. Then  $G$  is called a subgraph of  $H$  if an injective morphism  $p : G \hookrightarrow H$  exists.

Note that since the injective morphism can also be surjective, by this definition every graph  $G$  is a subgraph of itself.

**Definition 3.4 (intermediate-graph).** Let  $G$  and  $H$  be graphs such that  $G$  is a subgraph of  $H$ . A graph  $C$  is called an intermediate-graph of  $G$  and  $H$ , if  $G$  is a subgraph of  $C$  and  $C$  is a subgraph of  $H$ . The set of intermediate-graphs of  $G$  and  $H$  is denoted by  $\text{IG}(G, H)$ .

**Definition 3.5 (overlap).** Let the graphs  $G_1$  and  $G_2$  be given. An overlap  $P = (H, i_{G_1}, i_{G_2})$  consists of a graph  $H$  and a jointly surjective pair of injective morphisms  $i_{G_1} : G_1 \hookrightarrow H$  and  $i_{G_2} : G_2 \hookrightarrow H$  with  $i_{G_1}(G_1) \cap i_{G_2}(G_2) \neq \emptyset$ . The set of all overlaps of  $G_1$  and  $G_2$  is denoted by  $\text{ol}(G_1, G_2)$ . If a tuple as above is not given, then  $G_P$ ,  $i_{G_1}^P$  and  $i_{G_2}^P$  denote the graph and morphisms of a given overlap  $P \in \text{ol}(G_1, G_2)$ .

Note that  $(H, i_{G_1}, i_{G_2})$  with  $i_{G_1}$  and  $i_{G_2}$  being jointly surjective and  $i(G_1) \cap i'(G_2) = \emptyset$  could also be considered as an overlap of  $G_1$  and  $G_2$ . In this paper we only need to consider overlaps with  $i_{G_1}(G_1) \cap i_{G_2}(G_2) \neq \emptyset$ . So we have embedded this property directly into the definition.

As mentioned above, our approach also considers intermediate graphs. Therefore a notion of restricted graph morphisms is needed. For this, we introduce the notion of *restricted morphisms*, which intuitively is the restriction of the domain and co-domain of a morphism  $p : G \hookrightarrow H$  with subgraphs of  $G$  and  $H$  respectively.

**Definition 3.6 (restricted morphism).** Let the graphs  $G$ ,  $H$  and a morphism  $f : G \rightarrow H$  be given. Then, a morphism  $f' : G' \rightarrow H'$  is called a restricted morphism of  $p$  if morphisms  $i : G' \hookrightarrow G$  and  $i' : H' \hookrightarrow H$  exist ( $G'$  is a subgraph of  $G$  and  $H'$  is a subgraph of  $H$ ) such that

$$\begin{aligned} i'_E \circ f'_E &= f_E \circ i_E \wedge \\ i'_V \circ f'_V &= f_V \circ i_V. \end{aligned}$$

A restricted morphism of  $p$  is denoted by  $p^r$ .

Note that given a morphism  $p : G \rightarrow H$  a restriction  $p^r : G' \rightarrow H'$  of  $p$  is uniquely determined by  $G'$  and  $H'$ .

### 3.2 Nested Graph Conditions and Constraints

*Nested graph constraints* are useful for specifying graph properties. The more general notion of *nested graph conditions* allows the specification of properties for graph morphisms and the definition of graph conditions and constraints in a recursive manner. Within these conditions, only quantifiers and Boolean operators are used [4].

**Definition 3.7 (nested graph condition).** A nested graph condition over a graph  $C_0$  is defined recursively as

1. *true* is a graph condition over every graph.
2.  $\exists(a_0 : C_0 \hookrightarrow C_1, d)$  is a graph condition over  $C_0$  if  $a_0$  is an injective graph morphism and  $d$  is a graph condition over  $C_1$ .

3.  $\neg d$ ,  $d_1 \wedge d_2$  and  $d_1 \vee d_2$  are graph conditions over  $C_0$  if  $d$ ,  $d_1$  and  $d_2$  are graph conditions over  $C_0$ .

Conditions over the empty graph  $\emptyset$  are called constraints. We use the abbreviations  $\forall(a_0 : C_0 \hookrightarrow C_1, d) := \neg\exists(a_0 : C_0 \hookrightarrow C_1, \neg d)$  and  $\text{false} = \neg \text{true}$ .

Conditions of the form  $\exists(a_0 : C_0 \hookrightarrow C_1, d)$  are called existentially bound, the graph  $C_1$  is also called existentially bound. Conditions of the form  $\forall(a_0 : C_0 \hookrightarrow C_1, d)$  are called universally bound, the graph  $C_1$  is also called universally bound.

Since these are the only types of conditions that will be used in this paper, we will refer to them only as *conditions* and *constraints*. We will use the more compact notations  $\exists(C_1, d)$  for  $\exists(a_0 : C_0 \hookrightarrow C_1, d)$  and  $\forall(C_1, d)$  for  $\forall(a_0 : C_0 \hookrightarrow C_1, d)$  if  $C_0$  and  $a_0$  are clear from the context.

**Definition 3.8 (semantic of graph conditions).** Given a graph  $G$ , a condition  $c$  over  $C_0$  and a graph morphism  $p : C_0 \hookrightarrow G$ . Then  $p$  satisfies  $c$ , denoted by  $p \models c$ , if

1. If  $c = \text{true}$ .
2. If  $c = \exists(a_0 : C_0 \hookrightarrow C_1, d)$ : There does exists an injective morphism  $q : C_1 \hookrightarrow G$  with  $p = q \circ a_0$  and  $q \models d$ .
3. If  $c = \neg d$ :  $p \not\models d$ .
4. If  $c = d_1 \wedge d_2$ :  $p \models d_1$  and  $p \models d_2$ .
5. If  $c = d_1 \vee d_2$ :  $p \models d_1$  or  $p \models d_2$ .

A graph  $G$  satisfies a constraint  $c$ , denoted by  $G \models c$ , if the morphism  $p : \emptyset \hookrightarrow G$  satisfies  $c$ .

Our approach is designed to repair a specific type of constraint, constraints without any boolean operators. Each of these conditions can be transformed into an equivalent condition in so-called *alternating quantifier normal form* [7]. As the name suggests, these are conditions with alternating quantifiers and without any boolean operators.

**Definition 3.9 (alternating quantifier normal form (ANF)).** Conditions in alternating quantifier normal form (ANF) are defined recursively as

1. *true* and *false* are conditions in ANF.
2.  $\exists(a_0 : C_0 \hookrightarrow C_1, d)$  is a condition in ANF if either  $d$  is an universally bound condition over  $C_1$  in ANF or  $d = \text{true}$ .
3.  $\forall(a_0 : C_0 \hookrightarrow C_1, d)$  is a condition in ANF if either  $d$  is an existentially bound condition over  $C_1$  in ANF or  $d = \text{false}$ .

$$\begin{array}{ccc}
C_0 & \xhookrightarrow{i} & C'_0 \\
\downarrow a_0 & & \downarrow a'_0 \\
C_1 & \xhookrightarrow{i'} & C'_1
\end{array}$$

Figure 1: Diagram for the Shift operator.

In both cases,  $d$  is called a subcondition of  $\exists(a : C_0 \hookrightarrow C_1, d)$  or  $\forall(a : C_0 \hookrightarrow C_1, d)$  respectively. All subconditions of  $d$  are also subconditions of  $\exists(a : C_0 \hookrightarrow C_1, d)$  or  $\forall(a : C_0 \hookrightarrow C_1, d)$  respectively. The nesting level  $\text{nl}(c)$  of a condition  $c$  is recursively defined as  $\text{nl}(\text{true}) = \text{nl}(\text{false}) = 0$  and  $\text{nl}(\exists(a : P \hookrightarrow Q, d)) = \text{nl}(\forall(a : P \hookrightarrow Q, d)) := \text{nl}(d) + 1$ .

In the literature, conditions in ANF also allow conditions that end with conditions of the form  $\exists(C_1, \text{false})$  or  $\forall(C_1, \text{true})$ . We exclude these cases so that conditions in ANF can only end with conditions of the form  $\exists(C_1, \text{true})$  or  $\forall(C_1, \text{false})$ , since it is easily seen that every morphism  $p : C_0 \hookrightarrow G$  satisfies  $\forall(C_1, \text{true})$  and does not satisfy  $\exists(C_1, \text{false})$ . Therefore, these conditions can be replaced by **true** and **false** respectively.

In the following, we assume that all conditions are finite. As a direct consequence, the nesting level is also finite.

Using the *shift over morphism* construction, we are able to transform a nested condition over  $C$  into a nested condition over  $C'$  via an injective morphism  $i : C \hookrightarrow C'$  [4].

**Definition 3.10 (shift over morphism).** Let a condition  $c$  over  $C_0$  and a morphism  $i : C_0 \hookrightarrow C'_0$  be given. The shift of  $c$  over  $i$ , denoted by  $\text{Shift}(c, i)$ , is given by

1. If  $c = \text{true}$ ,  $\text{Shift}(c, i) = \text{true}$ .
2. If  $c = \exists(a_1 : C_0 \hookrightarrow C_1, d)$ ,  $\text{Shift}(c, i) = \bigvee_{(a', i') \in \mathcal{F}} \exists(a', \text{Shift}(d, i'))$  with  $\mathcal{F}$  being the set of all pairs  $(a', i')$  of injective morphisms that are jointly surjective and  $i' \circ a = a' \circ i$ , i.e., the diagram shown in Figure 1 commutes.
3. If  $c = \neg d$ ,  $\text{Shift}(c, i) = \neg \text{Shift}(d, i)$
4. If  $c = d_1 \wedge d_2$ ,  $\text{Shift}(c, i) = \text{Shift}(d_1, i) \wedge \text{Shift}(d_2, i)$
5. If  $c = d_1 \vee d_2$ ,  $\text{Shift}(c, i) = \text{Shift}(d_1, i) \vee \text{Shift}(d_2, i)$

**Lemma 3.11.** Let a condition  $c$  over  $C_0$  and a morphism  $i : C_0 \hookrightarrow C'_0$  be given. Then, for each morphism  $m : C'_0 \hookrightarrow G$ ,

$$m \models \text{Shift}(c, i) \iff m \circ i \models c$$

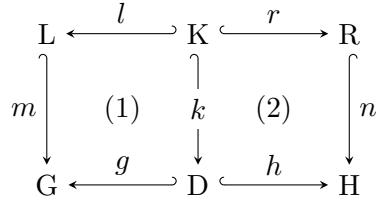


Figure 2: Diagram of a transformation in the double-pushout approach.

### 3.3 Rules and Graph Transformations

Via *rules* and *graph transformation* graphs can be modified by inserting or deleting nodes and edges. We will use the concept of the double-pushout approach for rules and transformations, which is based on category theory [1]. A rule consists of the three graphs  $L$ , called the *left-hand side*,  $K$ , called *context*, and  $R$ , called *right-hand side*, where  $K$  is a subgraph of  $L$  and  $R$ . During a transformation, denoted by  $G \Rightarrow H$ , elements of  $L \setminus K$  are removed and elements of  $R \setminus K$  are inserted so that a new morphism  $p : R \hookrightarrow H$  is created. In addition, the so-called *dangling edge condition* must be satisfied, which means that for every edge  $e \in E_H$  there are vertices  $u, v \in V_H$  such that  $\text{tar}(e) = u$  and  $\text{src}(e) = v$  or vice versa. We also define application conditions. These are nested conditions over  $L$  and  $R$  that prevent the transformation if they are not satisfied. Later, we will use application conditions to ensure that transformations cannot reduce consistency. For example, application conditions that prevent a transformation if  $G \models c$  and  $H \not\models c$ .

**Definition 3.12 (rules and application conditions).** A plain rule  $\rho = L \xleftarrow{l} K \xrightarrow{r} R$  consists of graphs  $L, K, R$  and injective graph morphisms  $l : K \hookrightarrow L$  and  $r : K \hookrightarrow R$ . The rule  $\rho^{-1} = R \xleftarrow{r} K \xleftarrow{l} L$  is called the inverse rule of  $\rho$ .

An application condition is a nested condition over  $L$  or  $R$  respectively. A rule  $(\text{ap}_L, \rho, \text{ap}_R)$  consists of a plain rule  $\rho$  and application conditions  $\text{ap}_L$  over  $L$ , called left application condition, and  $\text{ap}_R$  over  $R$ , called right application condition respectively.

**Definition 3.13 (graph transformation).** Let a rule  $\rho = (\text{ap}_L, \rho', \text{ap}_R)$ , a graph  $G$  and a morphism  $m : L \hookrightarrow G$ , called the match, be given. Then, a graph transformation  $t : G \Rightarrow_{\rho, m} H$  is given in Figure 2 if the squares (1) and (2) are pushouts in the sense of category theory,  $m \models \text{ap}_L$  and the morphism  $n : L \hookrightarrow H$ , called the co-match of  $t$ , satisfies  $\text{ap}_R$ .

The presence of right applications conditions leads to unpleasant side effects. The satisfaction of a right application condition can only be checked after the transformation. The transformation must therefore be reversed if the co-match does not satisfy this condition. To avoid this, we introduce the *shift over rule* operation, which is capable of transforming a right into an equivalent left application condition [4].

**Definition 3.14 (shift over rule).** Let a plain rule  $\rho = L \xleftarrow{l} K \xrightarrow{r} R$  and a right

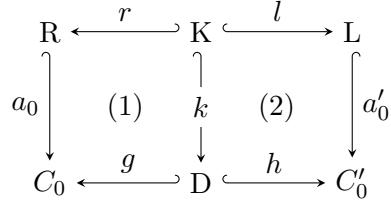


Figure 3: Transformation for the shift over rule operator.

application condition  $\text{ap}$  of  $\rho$  be given. The shift of  $\text{ap}$  over  $\rho$ , denoted with  $\text{Left}(\text{ap}, \rho)$ , is defined as

1. If  $\text{ap} = \text{true}$ ,  $\text{Left}(\text{ap}, \rho) := \text{true}$ .
2. If  $\text{ap} = \neg d$ ,  $\text{Left}(\text{ap}, \rho) := \neg \text{Left}(d, \rho)$ .
3. If  $\text{ap} = d_1 \wedge d_2$ ,  $\text{Left}(\text{ap}, \rho) := \text{Left}(d_1, \rho) \wedge \text{Left}(d_2, \rho)$ .
4. If  $\text{ap} = d_1 \vee d_2$ ,  $\text{Left}(\text{ap}, \rho) := \text{Left}(d_1, \rho) \vee \text{Left}(d_2, \rho)$ .
5. If  $\text{ap} = \exists(a_0 : R \hookrightarrow C_1, d)$ ,  $\text{Left}(\text{ap}, \rho) := \exists(a'_0 : L \hookrightarrow C'_0, \text{Left}(d, \rho'))$  where  $\rho' = C_0 \xleftarrow{g} D \xleftarrow{h} C'_0$  is the rule derived in Figure 3 by applying  $\rho^{-1}$  at match  $a_0$ . If this transformation does not exist, we set  $\text{Left}(\text{ap}, \rho) := \text{false}$ .

**Lemma 3.15.** Let a plain rule  $\rho = L \xleftarrow{l} K \xleftarrow{r} R$ , a right application condition  $\text{ap}$  for  $\rho$  and a transformation  $t : G \Rightarrow_{\rho, m} H$  be given. Then,

$$m \models \text{Left}(\text{ap}, \rho) \iff n \models \text{ap}.$$

Shift over rule produces an equivalent left application condition, meaning that, given a right application condition  $\text{ap}$  and a plain rule  $\rho$ , a match of a transformation satisfies  $\text{Left}(\text{ap}, \rho)$  if and only if the co-match satisfies  $\text{ap}$  [4]. Since every right application condition can be transformed into an equivalent left application condition, we will assume from now on that each rule contains only left application conditions. These rules are denoted by  $(\text{ap}, \rho)$ .

Via the *track morphism* it is possible to track elements across a transformation [6].

**Definition 3.16 (track morphism).** Consider the transformation  $t$  shown in figure 2. The track morphism,  $\text{tr}_t : G \rightarrow H$ , of  $t$  is defined as

$$\text{tr}_t = \begin{cases} h(g^{-1}(e)) & \text{if } e \in g(D) \\ \text{undefined} & \text{otherwise.} \end{cases}$$

For example, given a transformation  $t : G \Rightarrow H$ , the track morphism can be used to check whether a morphism  $p : C \hookrightarrow G$  still exists in the derived graph  $H$  by checking

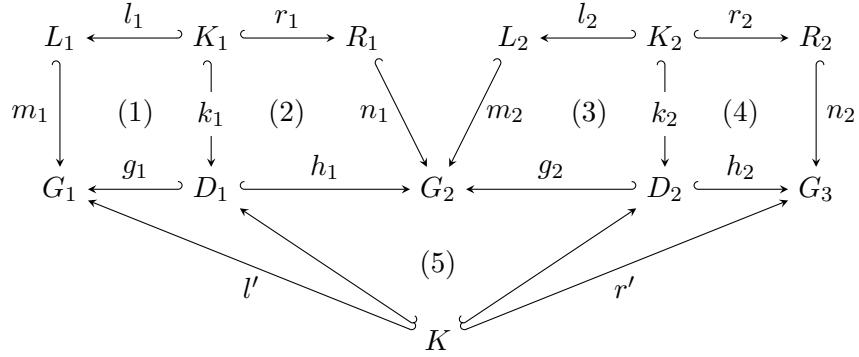


Figure 4: Pushout diagram of the transformation sequence  $G_1 \xrightarrow{\rho_1, m_1} G_2 \xrightarrow{\rho_2, m_2} G_3$  using the rules  $\rho_1 = L_1 \xleftarrow{l_1} K_1 \xrightarrow{r_1} R_1$  and  $\rho_2 = L_2 \xleftarrow{l_2} K_2 \xrightarrow{r_2} R_2$ .

whether  $\text{tr}_t \circ p$  is total, or whether a new morphism  $q : C \hookrightarrow H$  has been inserted by checking that no morphism  $p : C \hookrightarrow H$  with  $q = \text{tr}_t \circ p$  exists.

Given a sequence of transformations, the notion of *concurrent rules* can be used to describe this sequence by a rule. In other words, any sequence of transformations can be replaced by a transformation via its concurrent rule [2].

**Definition 3.17 (concurrent rule).** Let the rules  $\rho_1 = L_1 \xleftarrow{l_1} K_1 \xrightarrow{r_1} R_1$ ,  $\rho_2 = L_2 \xleftarrow{l_2} K_2 \xrightarrow{r_2} R_2$  and a sequence of transformations

$$G_1 \xrightarrow{\rho_1, m_1} G_2 \xrightarrow{\rho_2, m_2} G_3$$

be given. Then,  $\rho' = G_1 \xleftarrow{l'} K \xrightarrow{r'} G_3$  is called the concurrent rule of the transformation sequence if the square (5) in 4 is a pullback.

A transformation sequence  $G_1 \xrightarrow{\rho_1, m_1} G_2 \xrightarrow{\rho_2, m_2} G_3$  can be replaced by a transformation  $G_1 \xrightarrow{\rho', \text{id}} G_3$  via its concurrent rule. By inductive application, a concurrent rule for a transformation sequence  $G_1 \xrightarrow{\rho_0} \dots \xrightarrow{\rho_n} G_n$  of arbitrary finite length can be derived.

### 3.4 Concepts of Consistency

Now, we will introduce familiar consistency concepts. Namely, the notions of consistency preserving and guaranteeing transformations [4] and the notions of (direct) consistency sustaining and improving transformations [5]. Later, we will examine how these concepts differ from and share with our newly introduced concept of consistency.

**Definition 3.18 (consistency preserving and guaranteeing transformations).** Let a constraint  $c$  and a transformation  $t : G \Rightarrow H$  be given. Then,  $t$  is called  $c$ -preserving if

$$G \models c \Rightarrow H \models c.$$

The transformation  $t$  is called  $c$ -guaranteeing if  $H \models c$ .

While consistency preserving and guaranteeing transformations are defined for nested conditions, the finer-grained notions of (direct) consistency sustaining and improving transformations are defined only for conditions in ANF.

**Definition 3.19 (consistency sustaining and improving transformations).** Let a constraint  $c$  in ANF and a transformation  $t : G \Rightarrow_{\rho} H$  be given. If  $c$  is existentially bound,  $t$  is called consistency sustaining w.r.t.  $c$  if it is  $c$ -preserving and  $t$  is called consistency improving w.r.t.  $c$  if it is  $c$ -guaranteeing. If  $c = \forall(a_0 : \emptyset \hookrightarrow C_1, d)$  is universally bound,  $t$  is called consistency sustaining w.r.t.  $c$  if

$$|\{p : C_1 \hookrightarrow G \mid p \not\models d\}| \geq |\{p : C_1 \hookrightarrow H \mid p \not\models d\}|$$

and  $t$  is called consistency improving w.r.t.  $c$  if

$$|\{p : C_1 \hookrightarrow G \mid p \not\models d\}| > |\{p : C_1 \hookrightarrow H \mid p \not\models d\}|.$$

The number of elements of these sets are called the number of violations in  $G$  and number of violations in  $H$  respectively.

The even stricter notion of direct sustaining and improving transformations prohibits the insertion of new violations altogether.

**Definition 3.20 (direct sustaining and improving transformations).** Let a constraint  $c$  in ANF and a transformation  $t : G \Rightarrow_{\rho} H$  be given. If  $c$  is existentially bound,  $t$  is called direct consistency sustaining w.r.t.  $c$  if  $t$  is  $c$ -preserving and  $t$  is called direct consistency improving w.r.t.  $c$  if  $t$  is  $c$ -guaranteeing.

If  $c = \forall(a_0 : \emptyset \hookrightarrow C_1, d)$ ,  $t$  is called consistency sustaining w.r.t.  $c$  if

$$\begin{aligned} \forall p : C_0 \hookrightarrow G((p \models d \wedge \text{tr}_t \circ p \text{ is total}) \implies \text{tr}_t \circ p \models d) \wedge \\ \forall p' : C_0 \hookrightarrow H(\neg \exists q : C_0 \hookrightarrow G(p' = \text{tr}_t \circ q) \implies p' \models d) \end{aligned}$$

and  $t$  is called consistency improving w.r.t.  $c$  if additionally

$$\begin{aligned} \exists p : C_0 \in G(p \not\models d \wedge \text{tr}_t \circ p \text{ is total} \wedge \text{tr}_t \circ p \models c) \vee \\ \exists p : C \hookrightarrow G(p \not\models d \wedge \text{tr}_t \circ p \text{ is not total}). \end{aligned}$$

## 4 Consistency Increase and Maintainment

**Definition 4.1 (layer of a subcondition).** Let a condition  $c$  in ANF and a subcondition  $d$  of  $c$  be given. The layer of  $d$  is defined as  $\text{lay}(d) := \text{nl}(c) - \text{nl}(d)$ .

Our approach is based on the idea that the consistency of a constraint increases layer by layer, and that even small improvements, such as inserting single elements of existentially bound graphs, should be detectable as increasing. To formalise this, we introduce the notions of *consistency increasing* and *consistency maintaining* transformations and rules, where consistency increasing indicates that the consistency has actually increased and consistency maintaining indicates that the consistency has not decreased.

## 4.1 Universally quantified ANF

The definition of consistency increase and maintenance requires that each condition begins with a universal quantifier. Otherwise, case discrimination is required. Therefore, we will only consider a subset of the set of conditions in ANF, namely the set of universally quantified conditions in ANF, called *universally quantified ANF* (UANF). Furthermore, we will show that these sets are expressively equivalent by showing that every condition in ANF can be transformed into an equivalent condition in UANF.

**Definition 4.2 (universally quantified alternating quantifier normal form).** A conditions  $c$  in ANF is in universally quantified ANF (UANF) if it is universally bound.

Note that, given a condition  $c$  in UANF, any subcondition of  $c$  at layer  $0 \leq k \leq \text{nl}(c)$  is universally bound if  $k$  is an even number and existentially bound if  $k$  is an odd number. Furthermore, a graph  $C_k$  of  $c$  is universally bound if  $k$  is an odd number and existentially bound if  $k$  is an even number. It is already known that an existentially bound condition  $c$  can be extended to the equivalent condition  $\exists(a_0 : \emptyset \hookrightarrow C_0, d)$  [3]. We will use this to show that every condition in ANF has an equivalent condition in UANF.

**Lemma 4.3.** Any condition in ANF can be transformed into an equivalent condition in UANF.

*Proof.* Let a graph  $G$  and a constraint  $c$  in ANF be given. If  $c$  is universally bound, then  $c$  is already in UANF. If  $c = \exists(a_0 : C_0 \hookrightarrow C_1, d)$ , we show that  $c$  is equivalent to  $c' := \forall(\text{id}_{C_0} : C_0 \hookrightarrow C_0, c)$ .

1. Let  $p : C_0 \hookrightarrow G$  be a morphism such that  $q \models c$ . Then  $p \models c'$ , because  $p$  is the only morphism from  $C_0$  to  $G$  with  $p = p \circ \text{id}_{C_0}$  and  $p \models c$ .
2. Let  $p : C_0 \hookrightarrow G$  be a morphism with  $p \models c'$ , then all morphisms  $q : C_0 \hookrightarrow G$  with  $p = q \circ \text{id}_{C_0}$  satisfy  $c$ . Since  $p = p \circ \text{id}_{C_0}$ , it immediately follows that  $p \models c$ .

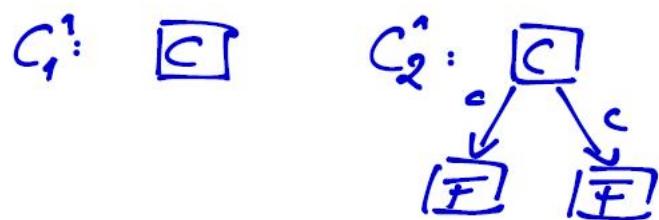
□

For the rest of this thesis, given a condition  $c = \forall(a_0 : C_0 \hookrightarrow C_1, d)$  in UANF, we assume that no morphism in  $c$ , except  $a_0$ , is bijective, since it can be shown that every condition in ANF can be transformed into an equivalent condition in ANF that satisfies this property, by showing that  $\exists(a_0 : C_0 \hookrightarrow C_0, \forall(a_1 : C_0 \hookrightarrow C_2, d))$  is equivalent to  $\forall(a_1 \circ a_0 : C_0 \hookrightarrow C_2, d)$  and that  $\forall(a_0 : C_0 \hookrightarrow C_0, \exists(a_1 : C_0 \hookrightarrow C_2, d))$  is equivalent to  $\exists(a_1 \circ a_0 : C_0 \hookrightarrow C_2, d)$ .

## 4.2 Conditions up to Layer

The goal of our approach is to increase the consistency of a constraint layer by layer, as we have already mentioned. To do this, we introduce a notion of partial consistency, called *satisfaction at layer*, which allows us to check whether a constraint is satisfied at a particular layer by checking whether the so-called *truncated condition* is satisfied at that layer.

$$G = \forall C_1 \exists C_2$$



$$C_2 = \forall C_1 \exists C_2 \forall C_3 \exists C_4$$

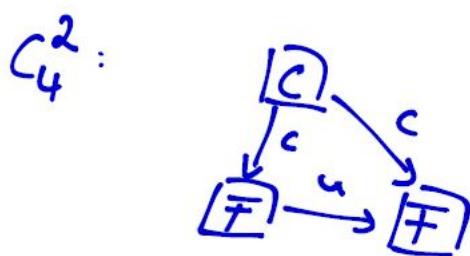
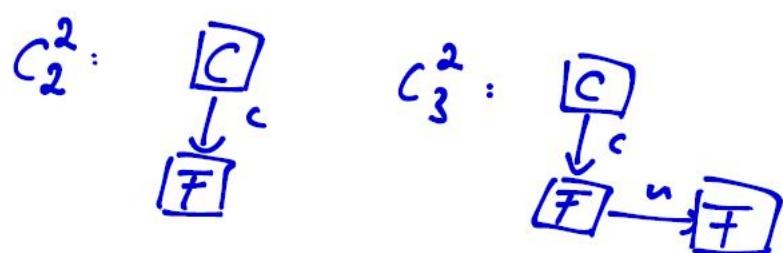


Figure 5: constraints

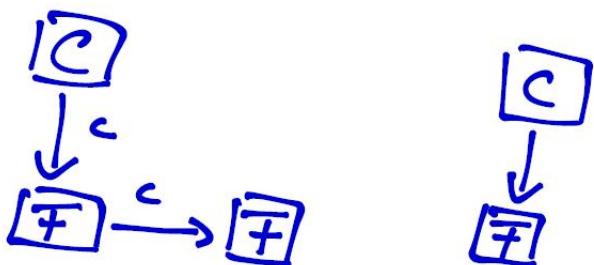


Figure 6: graph

**Definition 4.4 (subcondition at layer).** Let a condition  $c$  in ANF be given. The subcondition at layer  $0 \leq k \leq \text{nl}(c)$ , denoted by  $\text{sub}_k(c)$ , is the subcondition  $d$  of  $c$  with  $\text{lay}(d) = k$ .

**Example 4.1.** Consider the condition  $c = \forall(a_0 : C_0 \hookrightarrow C_1, \exists(a_1 : C_1 \hookrightarrow C_2, \forall(a_2 : C_2 \hookrightarrow C_3, \text{false})))$ . Then,  $\text{sub}_1(c) = \exists(a_1 : C_1 \hookrightarrow C_2, \forall(a_2 : C_2 \hookrightarrow C_3, \text{false}))$ .

Let us first introduce an operator which allows to replace a subcondition  $\text{sub}_k(c)$  by an arbitrary condition over  $C_k$ , called *replacement at layer*.

**Definition 4.5 (replacement at layer).** Given a condition  $c = Q(a_0 : C_0 \hookrightarrow C_1, d)$ , with  $Q \in \{\forall, \exists\}$ , in ANF, and a condition  $e$  over  $C_k$  in ANF. The replacement of layer  $k$  in  $c$  by  $e$ , denoted by  $\text{rep}_k(c, e)$ , is defined recursively as

$$\text{rep}_k(c, e) := \begin{cases} e & \text{if } k = 0 \\ Q(a_0 : C_0 \hookrightarrow C_1, \text{rep}_{k-1}(d, e)) & \text{otherwise.} \end{cases}$$

**Example 4.2.** Consider the conditions  $c := \forall(a_0 : C_0 \hookrightarrow C_1, \exists(a_1 : C_1 \hookrightarrow C_2, \text{true}))$  and  $d = \exists(a'_1 : C_1 \hookrightarrow C_3, e)$ . The replacement of layer 1 in  $c$  by  $d$  is given by

$$\text{rep}_1(c, d) = \forall(a_0 : C_0 \hookrightarrow C_1, \exists(a'_1 : C_1 \hookrightarrow C_3, e)).$$

We now define *truncated conditions* using the concept of replacement at layer. Intuitively, a condition is truncated at a particular layer by replacing the subcondition at that layer with **true** or **false**, depending on which quantifier the replaced subcondition is bound by.

**Definition 4.6 (truncated condition).** Let a condition  $c$  in UANF be given. The truncated condition of  $c$  at layer  $-1 \leq k < \text{nl}(c)$ , denoted by  $\text{cut}_k(c)$ , is defined as

$$\text{cut}_k(c) := \begin{cases} \text{true} & \text{if } k = -1 \\ \text{rep}_{k+1}(c, \text{true}) & \text{if } d \text{ is existentially bound, i.e. } k \text{ is odd} \\ \text{rep}_{k+1}(c, \text{false}) & \text{if } d \text{ is universally bound, i.e. } k \text{ is even.} \end{cases}$$

**Example 4.3.** Consider constraint  $c_2$  given in Figure 5. The truncated condition of  $c$  at layer 1 is given by  $\text{cut}_1(c_2) = \forall(C_1^1, \exists(C_2^1, \text{true}))$  and the truncated condition of  $c$  at layer 0 is given by  $\text{cut}_0(c_2) = \forall(C_1^1, \text{false})$ .

Note that the truncated condition of a condition  $c$  at layer  $\text{nl}(c) - 1$  is  $c$  itself. With these prerequisites we can now introduce *satisfaction at layer*, which allows us to check whether a condition is satisfied at a given layer. A morphism or graph satisfies a condition or constraint at that layer if it satisfies the truncated condition at that layer.

**Definition 4.7 (satisfaction at layer).** Let a graph  $G$  and a condition  $c$  in UANF be given. A morphism  $p : C_0 \hookrightarrow G$  satisfies  $c$  at layer  $-1 \leq k < \text{nl}(c)$ , denoted by  $p \models_k c$ , if

$$p \models \text{cut}_k(c).$$

$p \models_k c$	$p \models_{j < k} c$		$p \models_{j > k} c$		$p \models c$
	$j$ even	$j$ odd	$j$ even	$j$ odd	
$k$ even	?	✓	✓	✓	✓
$k$ odd	?	✓	?	?	?

Table 1: Overview of the conclusions made via satisfaction at layer with “✓” indicating that  $p \models_j$  and  $p \models c$ , respectively, if  $p \models_k$ . And “?” indicating that it cannot be concluded from  $p \models_k c$  whether  $p \models_j c$  or  $p \not\models_j c$ .

A graph  $G$  satisfies a constraint  $c$  at layer  $-1 \leq k < \text{nl}(c)$ , denoted by  $G \models_k c$ , if  $q : \emptyset \hookrightarrow G$  satisfies  $\text{cut}_k(c)$ . The largest  $-1 \leq k < \text{nl}(c)$  such that  $G \models_k c$  and there is no  $k < j < \text{nl}(c)$  with  $G \models_j c$ , called the biggest satisfied layer, is denoted by  $\text{k}_{\max}(c, G)$ . When  $c$  and  $G$  are clear from the context, we use the abbreviation  $\text{k}_{\max}$ .

Note that given a graph  $G$  and a constraint  $c$ ,  $\text{k}_{\max}(c, G)$  always exists, since  $\text{cut}_{-1}(c) = \text{true}$  and every graph satisfies  $\text{true}$ . Moreover, if  $p \models_{\text{nl}(c)-1} c$ , it immediately follows that  $p \models c$ .

**Example 4.4.** Consider the graph  $G$  given in Figure 6 and the constraint  $c_2$  given in Figure 5. This graph does not satisfy  $c_2$  because the second occurrence of **Class** does not satisfy  $\text{sub}_1(c_2) = \exists(C_2^2, (\forall C_3^2, (\exists C_4^2, \text{true})))$ , but it does satisfy  $\text{cut}_1(c_2) = \forall(C_1^1, \exists(C_2^2, \text{true}))$  and therefore

$$G \models_1 c_2 \text{ and } \text{k}_{\max} = 1.$$

Let a graph  $G$ , a condition  $c$  and a morphism  $p : C_0 \hookrightarrow G$  be given. Assume that  $p \models_k c$  for any  $0 \leq k < \text{nl}(c)$ . Then, we are able to conclude results for the satisfaction at other layers. If  $k$  is even, that means  $\text{sub}_k(c)$  is universally bound, we can conclude that  $p \models_j c$  for all  $k < j < \text{nl}(c)$  and in particular  $p \models c$ . For any  $0 \leq k < \text{nl}(c)$  with  $p \models_k c$  we can conclude that  $p \models_j c$  for all odd  $0 \leq j < k$ . An overview of these conclusion is shown in Table 1. We show these results within the following lemmas.

We start by investigating the conclusions for satisfaction at layer  $j > k$  if  $p \models_k c$ . Our first result shows that the replacement of the sub-condition  $\text{sub}_{k+1}(c)$  by any arbitrary condition over  $C_{k+1}$  leads to a condition that is satisfied by  $p$  if  $k$  is even.

**Lemma 4.8.** Let a graph  $G$ , a condition  $c$  in UANF and a morphism  $p : C_0 \hookrightarrow G$  with  $p \models_k c$ , such that  $0 \leq k < \text{nl}(c)$  is even, be given. Then, for any condition  $f$  over  $C_{k+1}$  it holds that

$$p \models \text{rep}_{k+1}(c, f).$$

*Proof.* Let  $0 \leq j \leq \text{nl}(c)$  be the smallest number with  $\text{sub}_j(c) = \forall(a_j : C_j \hookrightarrow C_{j+1}, d)$  being universally bound and  $p \models_j c$ . This must exist, since at least one of these exists due to the assumption. Let  $q : C_j \hookrightarrow G$  be a morphism such that  $q \models \forall(a_j : C_j \hookrightarrow C_{j+1}, \text{false})$ . This must exist, since  $p \models_j c$  and  $j$  is the smallest even number such that  $p \models_j c$ . Therefore, there does not exist a morphism  $q' : C_{j+1} \hookrightarrow G$  with  $q = q' \circ a_j$ . Hence, for every condition  $f$  over  $C_{j+1}$  a morphism  $q' : C_{j+1} \hookrightarrow G$  with  $q \not\models f$  and  $q = q' \circ a_j$

cannot exist. It follows immediately that  $q \models \forall(a_j : C_j \hookrightarrow C_{j+1}, f)$  and with that  $p \models \text{rep}_{j+1}(c, f)$ .

We can now conclude that for every even  $j < k \leq \text{nl}(c)$ , such that  $p \models_k c$ , and every condition  $d$  over  $C_{k+1}$  it holds that  $p \models \text{rep}_{j+1}(c, f)$  with  $f = \text{sub}_{j+1}(\text{rep}_{k+1}(c, d))$ . Since  $\text{rep}_{j+1}(c, f) = \text{rep}_{k+1}(c, d)$  it follows that  $p \models \text{rep}_{k+1}(c, d)$ .  $\square$

As a direct consequence of the previous lemma, a morphism satisfying a condition at layer  $k$  with  $k$  being even also satisfies the condition at layer  $j$  for all  $j > k$ .

**Lemma 4.9.** *Let a graph  $G$ , a morphism  $p : C_0 \hookrightarrow G$  and a condition  $c$  in UANF be given. If  $0 \leq k < \text{nl}(c)$  is even, i.e.  $\text{sub}_k(c)$  is universally bound, then for all  $k < j < \text{nl}(c)$  it holds that*

$$p \models_k c \implies p \models_j c.$$

*Proof.* Follows immediately by using lemma 4.8 and setting  $f$  equal to  $\text{sub}_{k+1}(\text{cut}_j(c))$ .  $\square$

Since a morphism  $p$  satisfies a condition  $c$  in UANF if and only if  $p$  satisfies  $c$  at layer  $\text{nl}(c) - 1$ , because  $\text{cut}_{\text{nl}(c)-1}(c) = c$ , we can conclude the following.

**Corollary 4.10.** *Let a graph  $G$ , a morphism  $p : C_0 \hookrightarrow G$  and a condition  $c$  in UANF be given. If  $0 \leq k < \text{nl}(c)$  is even it holds that*

$$p \models_k c \implies p \models c.$$

Furthermore, this allows us to make statements about the satisfaction of other conditions. Let a graph  $G$ , a morphism  $p : C_0 \hookrightarrow G$  and a condition  $c$  be given such that  $p \models_k c$  for an even  $0 \leq k < \text{nl}(c)$ . Then,  $p \models c$  and for every condition  $c'$  with  $\text{cut}_k(c) = \text{cut}_k(c')$ . With Lemma 4.8, it follows that  $p \models c'$ .

Let us now investigate the satisfaction at layer  $j$  with  $j < k_{\max}$ . If  $j$  is odd, i.e.  $\text{sub}_j(c)$  is existentially bound, we can conclude that  $p \models_j c$  as shown in Lemma 4.11. If  $j$  is even, i.e.  $\text{sub}_j(c)$  is universally bound, we are only able to make statements depending on  $k_{\max}$ . If  $k_{\max} < \text{nl}(c) - 1$ , it follows that  $p \not\models_j c$ . Otherwise  $p \models c$  and therefore  $k_{\max} = \text{nl}(c) - 1$  would follow immediately with Corollary 4.10. If  $k_{\max} = \text{nl}(c) - 1$  we can only state that at least one even  $j \leq k_{\max}$  with  $p \models_j c$  exists if  $c$  ends with a condition of the form  $\forall(a_k : C_k \hookrightarrow C_{k+1}, \text{false})$ .

**Lemma 4.11.** *Let a graph  $G$ , a morphism  $p : C_0 \hookrightarrow G$  and a constraint  $c$  in UANF be given. Then, for all odd  $0 \leq j < k_{\max}$ , i.e.  $\text{sub}_j(c)$  is existentially bound, it holds that*

$$p \models_j c.$$

*Proof.* If an even  $0 \leq j < k_{\max}$  with  $p \models_j c$  exists such that  $\text{sub}_j(c)$  is universally bound, let  $j'$  be the smallest of these. With lemma 4.8 follows that  $p \models_\ell c$  for all  $j' \leq \ell < \text{nl}(c)$ . Otherwise, if no such  $j'$  exists we set  $j' = k_{\max}$ .

Let  $\ell < j'$ , such that  $\text{sub}_\ell(c)$  is existentially bound and let  $d = \text{sub}_\ell(\text{cut}_{j'}(c)) = \exists(a_\ell : C_\ell \hookrightarrow C_{\ell+1}, e)$  be the sub-condition at layer  $\ell$  of the condition up to layer  $j'$  of  $c$ .

Since  $\ell < j'$ , a morphism  $q : C_\ell \hookrightarrow G$  with  $q \models d$  must exists and therefore a morphism  $q' : C_{\ell+1} \hookrightarrow G$  with  $q = q' \circ a_\ell$  must exists. It follows that  $q \models \exists(a_\ell : C_\ell \hookrightarrow C_{\ell+1}, \text{true})$  and with that  $p \models_\ell c$ .  $\square$

Through satisfaction at layer, an increase of consistency can be detected in the following way: Let  $t : G \implies H$  be a transformation. If  $k_{\max}(c, G) < k_{\max}(c, H)$ , we consider the transformation as consistency increasing, since  $H$  satisfies more layers of the constraint than  $G$ . But, the notion of consistency increasement should also be able to detect the smallest changes, performed by a transformation, that lead to an increase of consistency, namely the inserting of a single edge or node of an existentially bound graph. To remedy this issue, we introduce *intermediate conditions*, which will be used to recognize these types of increasements by checking whether an intermediate condition not satisfied by  $G$  is satisfied by  $H$ . Obviously, an decrease of consistency can be detected in a similar manner, by checking whether an intermediate condition satisfied by  $G$  is not satisfied by  $H$ . Intuitively, given a constraint  $c$  in UANF with  $\text{sub}_k(c) = \exists(a_k : C_k \hookrightarrow C_{k+1}, d)$  and  $0 \leq k < \text{nl}(c)$ , the condition  $\text{sub}_k(c)$  is replaced by  $\exists(a_k^r : C_k \hookrightarrow C', \text{true})$  with  $C' \in \text{IG}(C_k, C_{k+1})$ .

The construction of intermediate conditions is designed to only replace graphs in existentially bound layers, since the replacement in an universally bound layer would lead to a more restrictive constraint than the original condition up to layer. That means, given the condition  $c = \forall(a_0 : C_0 \hookrightarrow C_1, \text{false})$ , let  $C' \in \text{IG}(C_0, C_1)$ . If the condition  $c' = \forall(a_0^r : C_0 \hookrightarrow C', \text{false})$  is satisfied the satisfaction of  $c$  is implied but the backwards implication does not hold.

**Definition 4.12 (intermediate condition).** Let a condition  $c$  in UANF be given. Let  $0 \leq k < \text{nl}(c)$  such that  $k$  is odd, i.e.  $\text{sub}_k(c)$  is existentially bound. The intermediate condition, denoted by  $\text{IC}_k(c, C')$ , of  $c$  at layer  $k$  with  $C' \in \text{IG}(C_k, C_{k+1})$  is defined as

$$\text{IC}_k(c, C') := \text{rep}_k(c, \exists(a_k^r : C_k \hookrightarrow C', \text{true})).$$

**Example 4.5.** Consider constraint  $c_1$  given in figure 5. Since  $C_2^2 \in \text{IG}(C_1^1, C_2^1)$ , we can construct a intermediate condition of  $c_1$  at layer 1 with  $C_2^2$  as  $\text{IC}_1(c_1, C_2^2) = \forall C_1^1 \exists C_2^2$ . Whereas  $c_1$  checks whether each node of type **Class** is connected to at least two nodes of type **Feature**, the intermediate condition checks whether each node of type **Class** is connected to at least one node of type **Feature** which is trivially satisfied if  $c_1$  is satisfied.

Given a graph  $G$  and a constraint  $c$  in UANF with  $k_{\max} < \text{nl}(c) - 3$ , note that in this case  $\text{sub}_{k_{\max}}(c)$  has to be existentially bound, it holds that  $G \not\models_{k_{\max}+2} c$  and there does exist at least one graph  $C' \in \text{IG}(C_{k_{\max}+2}, C_{k_{\max}+3})$  such that  $G \models \text{IC}_{k_{\max}+2}(c, C')$  since  $G$  always satisfies  $\text{IC}_{k_{\max}+2}(c, C_{k_{\max}+2})$ .

### 4.3 Consistency Increasing and Maintaining Transformations and Rules

With the results above, we are now ready to define the notions of *consistency increase-ment* and *maintainment*, with increasement being a special case of maintainment. A

transformation  $t$  is considered as consistency maintaining if the consistency is not decreased, whereas  $t$  is considered as consistency increasing if the consistency has been increased.

These notions are designed to only detect transformations that maintain (or increase) the consistency of the first two unsatisfied layer of a constraint  $c$ . That means, given a graph  $G$  and a constraint  $c$ , let  $k = k_{\max} + 1$  and  $\text{sub}_k(c) := \forall(a_k : C_k \hookrightarrow C_{k+1}, \exists(a_{k+1} : C_{k+1} \hookrightarrow C_{k+2}, d))$ . A transformation  $t : G \Rightarrow H$  is considered as consistency maintaining if  $k_{\max}(c, G) \leq k_{\max}(c, H)$ , i.e. the satisfaction up to layer is not decreased, and at least the same amount of increasing insertions or deletions have been performed than decreasing ones. An increasing deletion is the deletion of an occurrence of  $C_{k+1}$  that does not satisfy  $\exists(a_{k+1} : C_{k+1} \hookrightarrow C_{k+2}, \text{true})$ , an increasing insertion is the insertion of elements of  $C_{k+2}$ , such that for at least one occurrence  $p$  of  $C_{k+1}$  it holds that  $p \not\models \exists(a_{k+1}^r : C_{k+1} \hookrightarrow C', \text{true})$  and  $\text{tr}_t op \models \exists(a_{k+1}^r : C_{k+1} \hookrightarrow C', \text{true})$  for a graph  $C' \in \text{IG}(C_{k+1}, C_{k+2})$ . A decreasing insertion is the creation of an occurrence of  $C_{k+1}$  not satisfying  $\exists(a_{k+1} : C_{k+1} \hookrightarrow C_{k+2}, \text{true})$  and a decreasing deletion is the deletion of elements of  $C_{k+2}$  such that for an occurrence  $p$  of  $C_{k+1}$  with  $p \models \exists(a_{k+1}^r : C_{k+1} \hookrightarrow C', \text{true})$  it holds that  $\text{tr}_t op \not\models \exists(a_{k+1}^r : C_{k+1} \hookrightarrow C', \text{true})$  for a graph  $C' \in \text{IG}(C_{k+1}, C_{k+2})$ . If  $k_{\max}(c, G) < k_{\max}(c, H)$  or the number of increasing insertions and deletions is greater than the number of decreasing ones,  $t$  is considered as consistency increasing.

To evaluate this, we define the *number of violations*. Intuitively, for all occurrences  $p$  of  $C_{k+1}$  the number of graphs  $C' \in \text{IG}(C_{k+1}, C_{k+2})$  with  $p \not\models \exists C'$  is add up and it can be determined whether more increasing or decreasing actions have been performed by a transformation.

Note, that the number of violations is defined for each layer of the constraint, but only for the first unsatisfied layer the sum is calculated as described above. For all layers  $k$  with  $k \leq k_{\max}$  it is set to 0 and for all layers  $k$  with  $k > k_{\max} + 1$  it is set to  $\infty$ . Through this, a transformation  $t : G \Rightarrow H$  that increases the satisfaction up to layer can easily detected since the number of violations in  $H$  at layer  $c_{\max}^G + 1$  will be set to 0.

**Definition 4.13 (number of violations).** Let a graph  $G$  and a constraint  $c$  in UANF be given. Let  $e = \text{sub}_{k_{\max} + 2}(c)$ . The number of violations  $\text{nvc}_j(G)$  at layer  $0 \leq j < \text{nl}(c)$  in  $G$  is defined as:

$$\text{nvc}_j(G) := \begin{cases} 0 & \text{if } j < k_{\max} + 1 \\ \sum_{C' \in \text{IG}(C_{j+1}, C_{j+2})} |\{q \mid q : C_{j+1} \hookrightarrow G \wedge q \not\models \text{IC}_0(e, C')\}| & \text{if } e \neq \text{false} \text{ and } j = k_{\max} + 1 \\ |\{q \mid q : C_{j+1} \hookrightarrow G\}| & \text{if } e = \text{false} \text{ and } j = k_{\max} + 1 \\ \infty & \text{if } j > k_{\max} + 1 \end{cases}$$

Note that the second and third case of Definition 4.13 only apply if  $G \not\models c$  and  $\text{sub}_{k_{\max}}(c)$  is existentially bound. Therefore  $e$  is also existentially bound or equal to **false**, if  $c$  ends with  $\forall(a_{\text{nl}(c)-1} : C_{\text{nl}(c)-1} \hookrightarrow C_{\text{nl}(c)}, \text{false})$ . Via the number of violations, we now define *consistency maintaining* and *increasing* transformations and rules, by checking whether the number of violations has not been increased, or in case of consistency increasing, has not been decreased for any layer of the constraint.

**Definition 4.14 (consistency maintaining and increasing transformations and rules).** Let a graph  $G$ , a rule  $\rho$  and a constraint  $c$  in UANF be given. A transformation  $t : G \Rightarrow_{\rho,m} H$  is called consistency maintaining w.r.t.  $c$ , if

$$\text{nvc}_k(H) \leq \text{nvc}_k(G)$$

for all  $0 \leq k < \text{nl}(c)$ . The transformation  $t$  is called consistency increasing w.r.t.  $c$  this inequality is strict. A rule  $\rho$  is called consistency maintaining/increasing w.r.t  $c$ , if all of its transformations are.

Note that if  $G \models c$  there does not exist a consistency increasing transformation  $G \Rightarrow H$  w.r.t  $c$ , since  $\text{nvc}_j(G) = 0$  for all  $0 \leq j < \text{nl}(c)$ . Also, no plain rule  $\rho$  is consistency increasing w.r.t  $c$ , since a graph  $G$  satisfying  $c$ , such that a transformation  $t : G \Rightarrow_{\rho,m} H$  exists can always be constructed. Therefore, each consistency increasing rule has to be equipped with at least one application condition.

As mentioned above, a transformation should be detected as consistency increasing if it increases the satisfaction up to layer, which is shown by the following theorem.

**Theorem 4.1.** Let a graph  $G$ , a rule  $\rho$  and a constraint  $c$  in UANF with  $G \not\models c$  be given. A transformation  $t : G \Rightarrow_{\rho,m} H$  is consistency increasing w.r.t.  $c$  if

$$k_{\max}(c, G) < k_{\max}(c, H)$$

*Proof.* No  $\ell > k_{\max}(c, G)$  with  $G \models_{\ell} c$  exists. Hence,  $\text{nvc}_{k_{\max}(c,G)+1}(G) > 0$  and  $\text{nvc}_{k_{\max}(c,G)+1}(G) \neq \infty$ . Since  $k_{\max}(c, H) > k_{\max}(c, G)$ ,  $\text{nvc}_{k_{\max}(c,G)+1}(H) = 0$  and it follows immediately that  $t$  is consistency increasing w.r.t.  $c$ .  $\square$

Since no consistency increasing transformation originating in consistent graphs exist, there do not exist infinite long sequences of consistency increasing transformations.

**Theorem 4.2.** Let a constraint  $c$  in UANF be given. Every sequence of consistency increasing transformations w.r.t  $c$  is finite.

*Proof.* Let  $G_0$  be a graph and

$$G_0 \Rightarrow_{\rho_0, m_0} G_1 \Rightarrow_{\rho_1, m_1} G_2 \Rightarrow_{\rho_2, m_2} \dots$$

be a sequence of consistency increasing transformations w.r.t  $c$ . We assume that  $k_{\max}(c, G_0) < \text{nl}(c)$ , otherwise  $\text{nvc}_j(G_0) = 0$  for all  $0 \leq j < \text{nl}(c)$  and no consistency increasing transformation  $G_0 \Rightarrow H$  w.r.t.  $c$  exists.

We show that after at most  $j := \text{nvc}_{k_{\max}(c,G_0)+1}(G_0)$  transformations  $G_j \models_{k_{\max}(c,G_0)+2} c$ . Note that  $j$  has to be finite, since  $G_0$  contains only a finite number of occurrences of  $C_{j+1}$ . Since each transformation is consistency increasing w.r.t.  $c$  it holds that  $\text{nvc}_{k_{\max}(c,G_i)+1}(G_{i+1}) \leq \text{nvc}_{k_{\max}(c,G_i)+1}(G_i) - 1$  after each transformation. Therefore, after at most  $j$  transformations,  $\text{nvc}_{k_{\max}(c,G_0)+1}(G_j) \leq \text{nvc}_{k_{\max}(c,G_0)+1}(G_0) - j = 0$  and  $G_j \models_{k_{\max}(c,G_0)+2} c$ . By iteratively applying this, it follows that after a finite number of transformations a graph  $G_k$  with  $G_k \models c$  has to exist. Since no consistency increasing transformation  $G_k \Rightarrow_{\rho_k, m_k} G_{k+1}$  exists, the sequence has to be finite.  $\square$

#### 4.4 Direct Consistency Maintaining and Increasing Transformations

Let a constraint  $c$  in UANF and graphs  $G$  with  $G \not\models c$  and  $H$  with  $H \models c$  be given. The transformation  $t : G \Longrightarrow_{\rho, \text{id}_G} H$  via the rule  $\rho = G \xleftarrow{l} \emptyset \xrightarrow{r} H$  is a consistency increasing transformation. Therefore, the notions of consistency increase- and maintenance, a similar example for a consistency maintaining transformation can easily be constructed, does allow insertions or deletions that are unnecessary in order to increase or maintain consistency. That is, the deletion of occurrences of existentially bound graphs, the deletions of occurrences  $p : C_k \hookrightarrow G$  of universally bound graphs  $C_k$  such that  $p \models \exists(a_k : C_k \hookrightarrow C_{k+1}, \text{true})$  or the insertion of occurrences of universally bound graphs and the insertion of occurrences  $p$  of intermediate graphs  $C' \in \text{IG}(C_{k-1}, C_k)$  with  $C_k$  being existentially bound, such that each occurrence  $q$  of  $C_{k-1}$  with  $q = p \circ a_{k-1}^r$  already satisfied  $\exists(a_{k-1}^r : C_{k-1} \hookrightarrow C_k, \text{true})$ .

*Direct consistency increasing and maintaining* transformations are more restricted, in the sense that these unnecessary deletions and insertions are leading to a transformation not being direct consistency increasing or maintaining, respectively. The presence of these unnecessary actions can be checked via second-order logic formulas. Additionally, it is secured that no new violations are introduced since these can always be considered as an unnecessary insertion or deletion. With that, the removal of one violation is sufficient to state that the transformation is (direct) consistency increasing, which can also be checked via a second order logic formula. We start by introducing *direct consistency maintaining* transformations. Let  $t : G \Longrightarrow H$  be a transformation.

Intuitively, formulas (4.1) and (4.2) check that no new violations of the first two unsatisfied layers are introduced, that means, every occurrence  $p$  of  $C_{k_{\max}(c,G)+2}$  that is not destroyed by  $t$  and satisfies  $\exists C'$  still satisfies  $\exists C'$  in  $H$  with the intermediate graph  $C' \in \text{IG}(C_{k_{\max}(c,G)+2}, C_{k_{\max}(c,G)+3})$ . To check that  $p$  has not been destroyed by  $t$ , we use the notion of total morphisms, since it has been shown that  $\text{tr}_t op$  is total if and only if  $p$  has not been destroyed by  $t$  [5]. Additionally, every newly inserted occurrence of  $C_{k_{\max}(c,G)+2}$  satisfies  $d$  with  $d = \text{false}$  if  $\text{sub}_{k_{\max}(c,G)+2}(C) = \text{false}$  and  $d = \exists C_{k_{\max}(c,G)+3}$  otherwise. This case discrimination arises as a consequence of the fact that conditions in UANF are also allows to end with a condition of the form  $\forall(a : C_0 \hookrightarrow C_1, \text{false})$ . Formulas (4.3) and (4.4) ensure that the satisfaction at layer has not been decreased by checking that no occurrences of universally bound graphs  $C_j$  with  $j < k_{\max}$  have been inserted and that no occurrences of existentially bound graphs  $C_j$  with  $j \leq k_{\max}$  have been destroyed. Only in these cases, the satisfaction at layer can be decreased. Of course, this does not always lead to a decrease of satisfaction at layer but it can always be considered as an unnecessary insertion or deletion.

The third formula secures that at least one violation has been removed and the last formulas secures that the satisfaction up to layer is not decreased.

**Definition 4.15 (direct consistency maintaining transformations).** Let  $G$  be a graph  $\rho$  a rule and  $c$  a constraint in UANF. If  $G \models c$ , a transformation  $t : G \Longrightarrow_{\rho, m} H$  is called direct consistency maintaining w.r.t.  $c$  if  $H \models c$ . Otherwise, if  $G \not\models c$ , let  $k = k_{\max}(c, G) + 2$ ,  $e = \text{sub}_k(c)$ . A transformation  $t : G \Longrightarrow_{\rho, m} H$  is called direct

consistency maintaining w.r.t.  $c$  if the following equations hold.

1. If  $e \neq \text{false}$ , every occurrence of  $C_k$  in  $G$  that satisfies  $\text{IC}_0(e, C')$  for any  $C' \in \text{IG}(C_k, C_{k+1})$  still satisfies  $\text{IC}_0(e, C')$  in  $H$ .

$$\forall p : C_k \hookrightarrow G \left( \bigwedge_{C' \in \text{IG}(C_k, C_{k+1})} (p \models \text{IC}_0(e, C') \wedge \text{tr}_t \circ p \text{ is total}) \implies \text{tr}_t \circ p \models \text{IC}_0(e, C') \right) \quad (4.1)$$

Otherwise, if  $e = \text{false}$ , (4.1) is set equal to true.

2. Let  $d = \text{IC}_0(e, C_{k+1})$  if  $e \neq \text{false}$  and  $d = \text{false}$  otherwise. Every newly inserted occurrence of  $C_k$  satisfies  $d$ .

$$\forall p' : C_k \hookrightarrow H (\neg \exists p : C_k \hookrightarrow G (p' = \text{tr}_t \circ p) \implies p' \models d) \quad (4.2)$$

3. No occurrence of a universally bound graph  $C_j$  with  $j \leq k_{\max}$  gets inserted.

$$\bigwedge_{\substack{i < k_{\max} \\ C_i \text{ universally}}} \forall p : C_i \hookrightarrow H (\exists p' : C_i \hookrightarrow G (p = \text{tr}_t \circ p')) \quad (4.3)$$

4. No occurrence of an existentially bound graph  $C_j$  with  $j \leq k_{\max}$  gets deleted.

$$\bigwedge_{\substack{i \leq k_{\max} \\ C_i \text{ existentially}}} \forall p : C_i \hookrightarrow G (\text{tr}_t \circ p \text{ is total}) \quad (4.4)$$

Before we continue with the definition of direct consistency increasing, let us first show that each direct consistency maintaining transformation is indeed consistency maintaining. For this, we start by showing that the satisfaction of formulas (4.3) and (4.4) guarantee that the satisfaction at layer has not been decreased.

**Lemma 4.16.** *Let a transformation  $t : G \implies H$  and a constraint  $c$  in UANF be given, such that (4.3) and (4.4) are satisfied. Then,*

$$H \models_{k_{\max}(c, G)} c.$$

*Proof.* Assume that  $H \not\models_{k_{\max}(c, G)} c$ . Then, either a new occurrence of an universally bound graph  $C_k$  with  $k < k_{\max}(c, G)$  has been inserted or an occurrence of an existentially bound graph  $C_k$  with  $k \leq k_{\max}(c, G)$  has been destroyed. Therefore, the following holds:

$$\exists p : C_i \hookrightarrow H (\neg \exists p' : C_i \hookrightarrow G (p = \text{tr}_t \circ p')) \vee \exists p : C_j \hookrightarrow G (\text{tr}_t \circ p \text{ is not total})$$

with  $i, j \leq k_{\max}(c, G)$ ,  $i$  being even and  $j$  being odd, i.e.  $C_i$  is universally and  $C_j$  is existentially bound. It follows immediately that either (4.3) or (4.4) is not satisfied. This is a contradiction.  $\square$

With this, we will now show that direct consistency maintaining transformation is also consistency maintaining.

**Theorem 4.3.** *Let a graph  $G$ , a constraint  $c$  in UANF, a rule  $\rho$  and a direct consistency maintaining transformation  $t : G \Rightarrow_{\rho,m} H$  w.r.t.  $c$  be given. Then,  $t$  is also a consistency maintaining transformation.*

*Proof.* With Lemma 4.16 follows that  $k_{\max}(c, G) \leq k_{\max}(c, H)$  and it also holds that  $nvc_{k_{\max}(c, G)+1}(H) \neq \infty$ . It remains to show that  $nvc_k(H) \leq nvc_k(G)$  for all  $0 \leq k < nl(c)$ . In particular, we only need to show that  $nvc_{k_{\max}(c, G)+1}(H) \leq nvc_{k_{\max}(c, G)+1}(G)$  since for all  $0 \leq j < k_{\max}(c, G) + 1$  it holds that  $nvc_j(H) = nvc_j(G) = 0$ . Also, since  $nvc_j(G) = \infty$  for all  $k_{\max}(c, G) + 1 < j < nl(c)$  it follows that  $nvc_j(H) \leq nvc_j(G)$ .

Let  $k = k_{\max}(c, G) + 1$  and  $d = \text{sub}_{k+1}(c)$ . We show that (4.1) and (4.2) imply that  $nvc_k(H) \leq nvc_k(G)$ . Assume that  $nvc_k(H) > nvc_k(G)$ .

Therefore, a morphism  $p : C_{k+1} \hookrightarrow H$  with  $p \not\models \text{IC}_0(d, C')$  for any  $C' \in \text{IG}(C_{k+1}, C_{k+2})$  exists, such that either 1 or 2 is satisfied. Note that this is only the case if  $d \neq \text{false}$ . Otherwise, a morphism  $p$  satisfying 2 must exist.

1. There does exist a morphism  $q' : C_k \hookrightarrow G$  with  $q' \models \text{IC}_0(d, C')$  and  $p = \text{tr}_t \circ q'$ .
2. There does not exist a morphism  $q : C_k \hookrightarrow G$  with  $p = \text{tr}_t \circ q$ .

This is a contradiction, if 1 is satisfied,  $q'$  does not satisfy equation (4.1) and if (2) is satisfied  $q$  does not satisfy equation (4.2) since  $q$  only satisfies  $\text{IC}_0(d, C_{k+2})$  if  $q$  satisfies  $\text{IC}_0(d, C')$  for all  $C' \in \text{IG}(C_{k+1}, C_{k+2})$ . It follows that

$$nvc_k(H) \leq nvc_k(G).$$

Therefore,  $t$  is a consistency maintaining transformation. □

Let us now introduce the notion of *direct consistency increasing* transformations. Similar to the definition of consistency maintaining and increasing transformation, again this notion is based on the notion of direct consistency maintaining transformations, in the sense that a direct consistency increasing transformation is also a direct consistency maintaining one. Since a direct consistency maintaining transformation  $t$  does not insert any new violations it is sufficient that  $t$  deletes at least one violation to state that  $t$  is direct consistency increasing. For this, (4.5) checks that either an occurrence  $p$  of  $C_{k_{\max}+2}$  has been deleted or  $\text{tr}_t \circ p \models \exists C'$  for an intermediate graph  $C' \in \text{IG}(C_{k_{\max}+2}, C_{k_{\max}+3})$  such that  $p \not\models \exists C'$ .

**Definition 4.17 (direct consistency increasing).** *Let a graph  $G$ , a rule  $\rho$  and a constraint  $c$  in UANF with  $G \not\models c$  be given. Let  $e = \text{sub}_{k_{\max}+2}(c)$  and*

*A transformation  $t : G \Rightarrow_{\rho,m} H$  is called direct consistency increasing w.r.t.  $c$  if it is direct consistency maintaining w.r.t.  $c$  and the following holds:*

$$\exists p : C_{k_{\max}+2} \hookrightarrow G(d) \tag{4.5}$$

with

$$d := \begin{cases} \text{tr}_t \circ p \text{ is not total} \\ \bigvee_{C' \in \text{IG}(k_{\max} + 2, k_{\max} + 3)} (p \not\models \text{IC}_0(e, C') \wedge (\text{tr}_t \circ p \text{ is not total} \vee \text{tr}_t \circ p \models \text{IC}_0(e, C'))) \end{cases} \quad \begin{array}{l} \text{if } e = \text{false} \\ \text{otherwise.} \end{array}$$

Note that (4.3) and (4.4) not only secure that the satisfaction up to layer does not decrease, as shown in Lemma 4.16, but also prevent further unnecessary insertions and deletions, since the insertion of a universally and the deletion of a existentially bound graph will never lead to an increase of consistency.

Now, we will show the already indicated relationship between direct consistency increasing and consistency increasing, namely that a direct consistency increasing transformation is also consistency increasing. Counterexamples, that the inversion of the implication does not hold can easily be constructed, showing that these notions are not identical but related.

**Theorem 4.4.** *Let a graph  $G$ , a constraint  $c$  in UANF with  $G \not\models c$ , a rule  $\rho$  and a direct consistency increasing transformation  $t : G \implies_{\rho, m} H$  w.r.t.  $c$  be given. Then,  $t$  is also a consistency increasing transformation.*

*Proof.* With Theorem 4.3 follows that  $t$  is a consistency maintaining transformation. Therefore, it is sufficient to show that  $\text{nvc}_{k_{\max}(c, G)+1}(H) < \text{nvc}_{k_{\max}(c, G)+1}(G)$ . Let  $k = k_{\max}(c, G) + 2$  and  $d = \text{sub}_k(c)$  with  $d \neq \text{false}$ .

Since (4.5) is satisfied, a morphism  $p : C_k \hookrightarrow G$  with  $p \not\models \text{IC}_0(d, C')$ , such that either  $\text{tr } op$  is total and  $\text{tr}_t \circ p \models \text{IC}_0(d, C')$  or  $\text{tr } op$  is not total exists, for a graph  $C' \in \text{IG}(C_k, C_{k+1})$ . In both cases the following holds:

$$\begin{aligned} p \in \{q \mid q : C_k \hookrightarrow G \wedge q \not\models \text{IC}_0(d, C')\} \wedge \\ \text{tr } op \notin \{q \mid q : C_k \hookrightarrow H \wedge q \not\models \text{IC}_0(d, C')\} \end{aligned}$$

Since  $t$  is direct consistency maintaining it follows that

$$|\{q \mid q : C_k \hookrightarrow G \wedge q \not\models \text{IC}_0(d, C')\}| \leq |\{q \mid q : C_k \hookrightarrow H \wedge q \not\models \text{IC}_0(d, C')\}|.$$

for all  $C \in \text{IG}(C_k, C_{k+1})$ . Additionally, this inequality is strictly satisfied if  $C = C'$ . Therefore, it follows that  $\text{nvc}_k(H) < \text{nvc}_k(G)$  and  $t$  is a consistency increasing transformation.

If  $d = \text{false}$  it holds that

$$|\{q \mid q : C_k \hookrightarrow G\}| \leq |\{q \mid q : C_k \hookrightarrow H\}|$$

since  $t$  is a direct consistency maintaining transformation and it can be shown in a similar manner as above that (4.5) implies

$$|\{q \mid q : C_k \hookrightarrow G\}| < |\{q \mid q : C_k \hookrightarrow H\}|.$$

In total it follows that  $t$  is a consistency increasing transformation.  $\square$

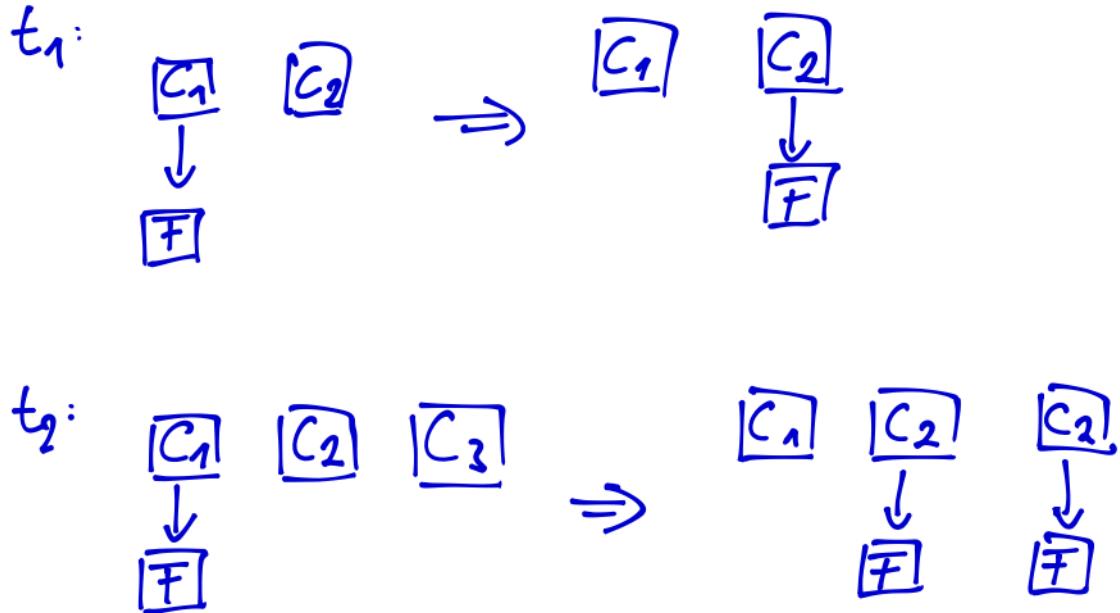


Figure 7: example

**Example 4.6.** Consider the transformations  $t_1$  and  $t_2$  given in Figure 7 and constraint  $c_1$  given in Figure 5. Then,  $t_1$  is a consistency maintaining transformation since the number of violations in both graphs is equal to 2. But,  $t_1$  is not a direct consistency maintaining transformation since one occurrence of a node of type Class satisfying  $\exists C_2^2$  in the first but not in the second graph of the transformation exists. Therefore (4.1) is not satisfied.

The transformation  $t_2$  is consistency increasing w.r.t.  $c_1$  since the number of violations is equal to 4 in the first and equal to 2 in the second graph. But,  $t_2$  is not a direct consistency increasing transformation since (4.1) is not satisfied.

#### 4.5 Comparison with other concepts of Consistency

In this chapter, the notions of (direct) consistency increase- and maintenance are compared to the already known concepts of consistency guaranteeing, consistency preserving [4], (direct) consistency increasing and sustaining [5] in order to reveal relationships between them and ensure that (direct) consistency increase- and maintenance are indeed a new notions of consistency. These relations are displayed in Figure 8.

First, we compare (direct) consistency increase- and maintenance with the notions of consistency-guaranteeing and -preserving. We start by showing that consistency maintaining implies preserving but the backwards implication does not hold.

**Lemma 4.18.** Let a constraint  $c$  in UANF, graphs  $G$  and  $H$  and a transformation

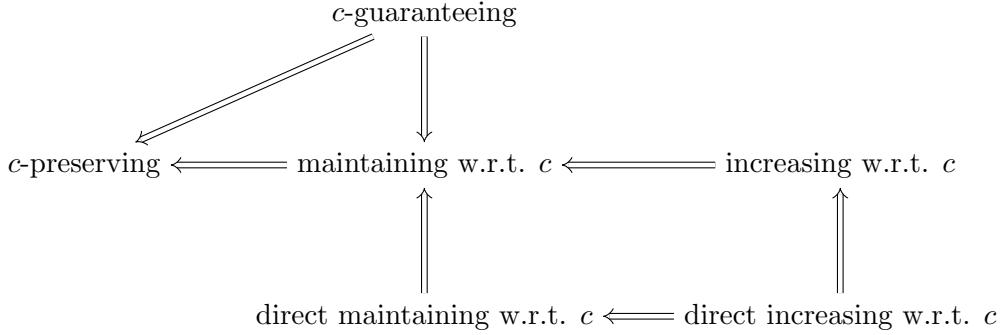


Figure 8: Relations of consistency notions.

$t : G \Rightarrow H$  be given. Then,

$$\begin{aligned} t \text{ is maintaining w.r.t. } c &\implies t \text{ is } c\text{-preserving} && \text{and} \\ t \text{ } c\text{-preserving} &\not\implies t \text{ is maintaining w.r.t. } c \end{aligned}$$

- Proof.*
1. Let  $t$  be a consistency maintaining transformation w.r.t.  $c$ . If  $G \not\models c$ ,  $t$  is a  $c$ -preserving transformation. If  $G \models c$ , it holds that  $\text{nvc}_j(G) = 0$  for all  $0 \leq j < \text{nl}(c)$ . Since  $t$  is consistency maintaining it follows that  $\text{nvc}_j(H) = 0$  for all  $0 \leq j < \text{nl}(c)$  and therefore  $H \models c$ . It follows that  $t$  is a  $c$ -preserving transformation.
  2. Consider graphs  $C_1^1, C_2^2$  and constraint  $c_1$  given in Figure 5. Then, the transformation  $t : C_2^2 \Rightarrow C_1^1$  is  $c$ -preserving, since  $C_2^2 \not\models c_1$ , but not consistency maintaining w.r.t.  $c$  since  $\text{nvc}_0(C_2^2) = 2$  and  $\text{nvc}_0(C_1^1) = 5$ .

□

Obviously, guaranteeing implies consistency maintaining, since this property is embedded in the definition of consistency maintenance. The inversion of this implication does not hold, since maintaining is a way stricter notion, in the sense, that the number of removed violations has to be greater or equal than the number of introduced violations. For guaranteeing transformations this is not the case, an arbitrary number of violations can be inserted, as long as the derived graph satisfies the constraint and therefore guaranteeing does not imply direct increasing, since a direct increasing transformations is not allowed to introduce any new violations.

**Lemma 4.19.** Let a constraint  $c$  in UANF, graphs  $G$  and  $H$  and a transformation  $t : G \Rightarrow H$  be given. Then,

$$\begin{aligned} t \text{ is } c\text{-guaranteeing} &\implies t \text{ is maintaining w.r.t. } c && \text{and} \\ t \text{ is } c\text{-guaranteeing} &\not\implies t \text{ is direct maintaining w.r.t. } c \text{ and} \\ t \text{ is maintaining w.r.t. } c &\not\implies t \text{ is } c\text{-guaranteeing} \end{aligned}$$



Figure 9: example

- Proof.*
1. Let  $t$  be a  $c$ -guaranteeing transformation, then  $H \models c$ . It holds that  $nvc_j(H) = 0$  for all  $0 \leq j < nl(c)$  and since the number of violations cannot be negative,  $t$  is a maintaining transformation w.r.t.  $c$ .
  2. Let  $t$  be a  $c$ -guaranteeing transformation. Then,  $t$  is also a  $c$ -preserving transformation and since each direct maintaining transformation is also a maintaining one the statement follows directly with Lemma 4.18.
  3. Consider the transformation  $t : G \implies H$  shown in Figure 9 and constraint  $c_1$  shown in Figure 5. Then,  $t$  is a maintaining transformation w.r.t  $c$ , in particular,  $c$  is a consistency increasing transformation w.r.t.  $c$  since  $nvc_0(G) = 10$  and  $nvc_0(H) = 7$ . But,  $t$  is not  $c$ -guaranteeing since both occurrences of nodes of type Class do not satisfy  $\exists C_2^1$ . □

Let  $t : G \implies H$  be a  $c$ -guaranteeing transformation. If  $G \models c$ , the transformation is, by definition, not consistency increasing. If  $G \not\models c$ ,  $t$  is always also a consistency increasing transformation w.r.t.  $c$ .

**Lemma 4.20.** *Let graphs  $G, H$ , a constraint  $c$  with  $G \not\models c$  and a transformation  $t : G \implies H$  be given. Then*

$$t \text{ is } c\text{-guaranteeing} \implies t \text{ is increasing w.r.t } c.$$

*Proof.* Let  $t$  be a  $c$ -guaranteeing transformation, then  $H \models c$ . Since  $G \not\models c$ , it holds that  $nvc_{k_{\max}(c,G)+1}(G) > 0$  and  $nvc_{k_{\max}(c,G)+1}(H) = 0$ . Therefore,  $t$  is consistency increasing. □

The definition of consistency improvement only differs from guaranteeing if the corresponding constraint is universally bound and these notions are identical for existentially bound constraint. Therefore, with Lemmas 4.19 and 4.20, we can state the following.

**Corollary 4.21.** *Let an existentially bound constraint  $c$  in ANF and a transformation  $t : G \implies H$  be given. Then,*

$$\begin{aligned} t \text{ is consistency improving w.r.t } c &\implies t \text{ is consistency maintaining w.r.t } c \text{ and} \\ t \text{ is consistency maintaining w.r.t } c &\not\implies t \text{ is consistency improving w.r.t } c. \end{aligned}$$

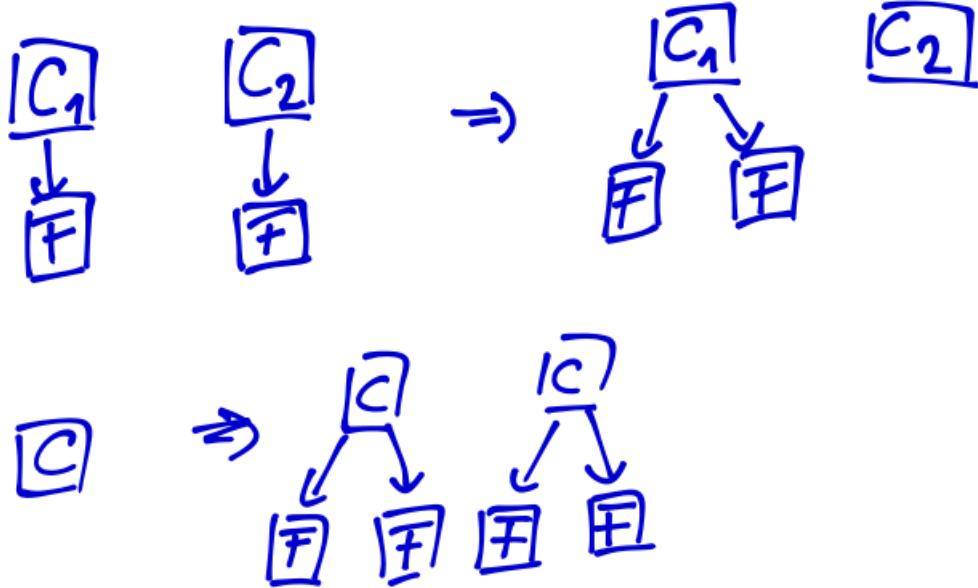


Figure 10: example

If  $G \not\models c$ , we can also state that

$$t \text{ is consistency improving w.r.t } c \implies t \text{ is consistency increasing w.r.t } c$$

The notions of increase- and improvement are equivalent for universally bound constraints with nesting level 1. Note that, with corollary 4.21, this equivalence does not hold for existentially bound constraints with nesting level 1.

**Lemma 4.22.** Let a universally bound constraint  $c$  in UANF with  $\text{nl}(c) = 1$ , a graph  $G$  with  $G \not\models c$  and a transformation  $t : G \Rightarrow H$  be given. Then,

$$t \text{ is consistency improving w.r.t } c \iff t \text{ is consistency increasing w.r.t } c$$

*Proof.* Let  $c = \forall(a : \emptyset \hookrightarrow C, \text{false})$ . Since  $\text{sub}_1(c) = \text{false}$ ,  $\text{nvc}_0(G)$  is the number of occurrences of  $C$  in  $G$ . This is exactly the definition of the number of violations for consistency improving transformations and the statement follows immediately.  $\square$

For universally bound constraints  $c$  with  $\text{nl}(c) \geq 2$ , the notions of (direct) consistency increase- and maintainment are not related to (direct) consistency improve- and sustainment. By definition, (direct) consistency improvement implies (direct) consistency sustainment [5]. Therefore, it is sufficient to show that direct improvement does not imply maintainment and that direct increasement does not imply consistency sustainment.

- ~~delete~~
- insert

move Feature :



assign Feature :



Figure 11: rules

**Lemma 4.23.** Let a universally bound constraint  $c$  in UANF with  $\text{nl}(c) \geq 2$ , a graph  $G$  with  $G \not\models c$  and a transformation  $t : G \Rightarrow H$  be given. Then,

$t$  is direct consistency improving w.r.t  $c \iff t$  is consistency maintaining w.r.t  $c$  and  
 $t$  is direct increasing w.r.t  $c \iff t$  is consistency sustaining w.r.t  $c$

*Proof.* 1. Consider transformation  $t_1$  given in Figure 10 and constraint  $c_1$  given in Figure 5. The transformation  $t_1$  is direct consistency improving but not maintaining since  $\text{nvc}_0(G) = 4$  and  $\text{nvc}_0(H) = 5$ .

2. Consider the constraint  $c = \forall(C_1^1, \exists(C_2^1, \forall(C_4^2, d)))$  with  $d$  being an existentially bound constraint in ANF with  $d \neq \text{false}$  composed of the graphs given in Figure 5 and transformation  $t_2$  given in Figure 10. Then,  $t$  is direct consistency increasing, (4.1), (4.2), (4.3) and (4.4) are trivially satisfied and (4.5) is satisfied since one occurrence of  $C_1^1$  that did not satisfy  $\exists C_2^1$  in  $G$  satisfies  $\exists C_2^1$  in  $H$ , but not consistency sustaining since the number of occurrences of  $C_1^1$  that not satisfying  $\exists(C_2^1, \forall(C_4^2, d))$  in  $H$  is greater than the number of occurrences of  $C_1^1$  in  $G$  not satisfying  $\exists(C_2^1, \forall(C_4^2, d))$ .  $\square$

## 5 Application Conditions

To guarantee that each transformation  $t$  is (direct) consistency increasing or maintaining w.r.t to a constraint  $c$ , we present applications conditions ensuring this property. Given a constraint  $c$ , these application conditions are designed to only consider graph of  $c$  up to a certain layer. In particular, this is useful to reduce the restrictiveness of these application conditions, since all graphs  $C_j$  of  $c$  with  $j > k_{\max} + 2$  do not affect whether a transformation is considered as consistency maintaining or increasing. Additionally, in the case of consistency increasing application conditions, this design is necessary since it has to be ensured that violations at layer  $k_{\max} + 1$  will be removed. Therefore,

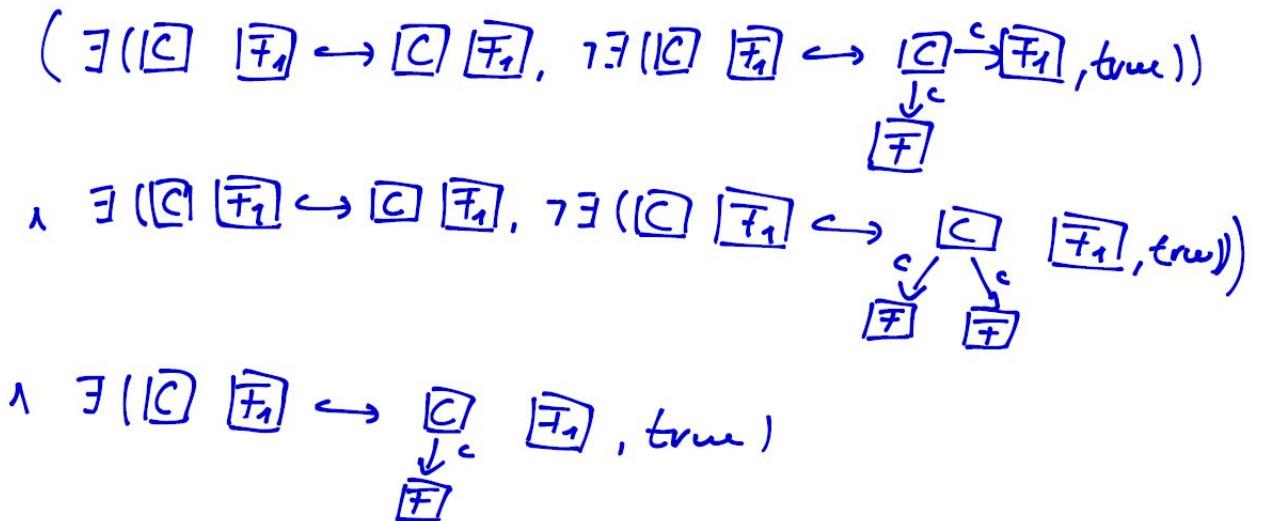


Figure 12: application condition

let us introduce a weaker notion of consistency increasing and maintaining rules, namely *consistency increasing and maintaining rules at layer*. As the name suggests, given a constraint  $c$ , a rule is consistency increasing or maintaining at layer  $0 \leq k < \text{nl}(c)$  if all of its applications at graphs  $G$ , with  $k_{\max} = k$ , are consistency increasing or maintaining w.r.t.  $c$ , respectively.

**Definition 5.1 ((direct) consistency increasing and maintaining rule at layer).** Let a constraint  $c$  be given. A rule  $\rho$  is called (direct) consistency maintaining at layer  $-1 \leq k < \text{nl}(c)$  w.r.t.  $c$  if all transformations  $t : G \Rightarrow_{\rho} H$ , with  $k_{\max}(G, c) = k$ , are (direct) consistency maintaining w.r.t.  $c$ . Additionally,  $\rho$  is called (direct) consistency increasing at layer  $0 \leq k < \text{nl}(c)$  w.r.t.  $c$  if all transformations  $t : G \Rightarrow_{\rho} H$ , with  $k_{\max}(G, c) = k$ , are (direct) consistency increasing w.r.t.  $c$ .

Note, that a consistency maintaining rule at layer  $\text{nl}(c)-1$  w.r.t.  $c$  is also a consistency maintaining or increasing rule w.r.t.  $c$ . But, a consistency increasing rule at layer  $\text{nl}(c)-1$  must not necessarily be a consistency increasing rule w.r.t.  $c$ .

### 5.1 General Application Conditions

We start by introducing consistency maintaining application conditions, i.e. a rule equipped with this application condition will only be applicable if the corresponding transformation is consistency maintaining. In particular, we will show that this transformation is even direct consistency maintaining.

The maintaining application condition consists of the three parts  $\text{ned}_k(\rho')$ ,  $\text{nui}_k(\rho')$  and  $\text{ndsk}(\rho')$  that are connected via the boolean  $\wedge$ -operator with  $\rho'$  being a plain rule and  $0 \leq k < \text{nl}(c)$  for a given constraint  $c$ . As already discussed, the satisfaction at layer is decreased if and only if occurrences of existentially bound graphs  $C_j$  have been deleted

or occurrences of universally bound graphs  $C_j$  have been inserted, with  $0 \leq j \leq k$ . To check that no existentially bound graph  $C_j$ ,  $0 \leq j \leq k$ , will be deleted, we use the condition constructed by  $\text{ned}_k(\rho')$  that checks that no overlap  $P$  of the left-hand-side of  $\rho'$  with  $C_j$ , such that the application of  $\rho'$  at  $P$  leads to a deletion of  $C_j$ , exists in the originating graph. To ensure that no universally bound graph  $C_j$ ,  $0 \leq j \leq k$ , will be inserted, the condition constructed by  $\text{nui}_k(\rho')$  checks that no overlap  $P$  of the right-hand-side of  $\rho'$  and  $C_j$ , such that the application of  $\rho'^{-1}$  at  $P$  would destroy the occurrence of  $C_j$ , exists in the derived graph. Note, that a condition constructed in this way is a right application condition. Therefore, we use the shift over rule operator to construct an equivalent left application condition. To verify that the number of violations is not decreased, we use the condition constructed by  $\text{nds}_k(\rho')$ , that checks, in the same manner as  $\text{nui}_k(\rho')$ , that no occurrences of graphs  $C' \in \text{IG}(C_{k+2}, C_{k+3})$  will be deleted if  $k < \text{nl}(c) - 2$ . Otherwise, this condition is set to **true**.

**Definition 5.2 (consistency maintaining application condition).** Let a rule  $\rho = (\text{ac}, \rho')$  with  $\rho' = L \longleftrightarrow K \hookrightarrow R$  and a constraint  $c$  in UANF be given. The maintaining application condition of  $c$  for  $\rho$  at layer  $-1 \leq k < \text{nl}(c)$  is defined as  $\text{ac} \wedge \text{main}_k(\rho')$  with

$$\text{main}_k(\rho') := \text{ned}_k(\rho') \wedge \text{nui}_k(\rho') \wedge \text{nds}_k(\rho')$$

and

1. Let  $E$  be the set of all existentially bound graphs  $C_j$  with  $j \leq k+1$  and  $\mathbf{P}_{C_j}$  be the set all overlaps  $P'$  of  $L$  and  $C_j$  with  $i_L^{P'}(L \setminus K) \cap i_{C_j}^{P'}(C_j) \neq \emptyset$ :

$$\text{ned}_k(\rho') := \bigwedge_{C \in E} \bigwedge_{P' \in \mathbf{P}_{C_j}} \neg \exists(i_L^{P'} : L \hookrightarrow P', \text{true})$$

2. Let  $U$  be the set of all universally bound graphs  $C_j$  with  $j \leq k+2$ , and  $\mathbf{P}_{C_j}$  be the set of all overlaps  $P'$  of  $R$  and  $C_j$  with  $i_R^{P'}(R \setminus K) \cap i_{C_j}^{P'}(C_j) \neq \emptyset$ :

$$\text{nui}_k(\rho') := \bigwedge_{C \in U} \bigwedge_{P' \in \mathbf{P}_{C_j}} \text{Left}(\neg \exists(i_R^{P'} : R \hookrightarrow P', \text{true}), \rho')$$

3. If  $k > \text{nl}(c) - 3$  or  $\text{sub}_k(c)$  is universally bound,  $\text{nds}_k(\rho') = \text{true}$ . Otherwise, Let  $E$  be the set of all overlaps of  $L$  and  $C'$  with  $i_L^{P'}(L \setminus K) \cap i_{C'}^{P'}(C') \neq \emptyset$  for all  $C' \in \text{IG}(C_{k+2}, C_{k+3})$ :

$$\text{nds}_k(\rho') := \bigwedge_{P \in E} \neg \exists(i_L^P : L \hookrightarrow P', \text{true})$$

### Example 5.1.

Let us now show that each rule equipped with the according application condition is a consistency maintaining rule at layer.

**Theorem 5.1.** Let a constraint  $c$  in UANF be given. Each rule  $\rho = (\text{ac}', \rho')$  with  $\text{ac}' = \text{ac} \wedge \text{main}_k(\rho')$  and  $-1 \leq k < \text{nl}(c)$  is a consistency maintaining rule at layer  $k$  w.r.t.  $c$ .

*Proof.* Let a graph  $G$ , such that  $k_{\max} = k$ , and a transformation  $t : G \Rightarrow_{\rho} H$  be given. It is sufficient to show that  $t$  is a direct consistency maintaining transformation. If  $k < \text{nl}(c) - 1$ , it follows that  $G \not\models c$  and that  $k$  is odd. We show that  $t$  satisfies (4.1), (4.2), (4.3) and (4.4).

1. Assume that (4.1) does not hold. Then,  $e = \text{sub}_{k+2}(c) \neq \text{false}$  and a morphism  $p : C_{k+2} \hookrightarrow G$  exists, such that  $p \models \text{IC}_0(e, C')$ ,  $\text{tr}_t op$  is total and  $\text{tr}_t op \not\models \text{IC}_0(e, C')$  for a graph  $C' \in \text{IG}(C_{k+2}, C_{k+3})$ . Therefore, an overlap  $P$  of  $L$  and  $C'$  such that  $i_{C_{k+2}}^P \models \exists(a_{k+2}^r : C_{k+2} \hookrightarrow C', \text{true})$  with  $i_L^P(L \setminus K) \cap i_{C'}^P(C' \setminus C_{k+2}) \neq \emptyset$  must exist and  $m \models \exists(i_L^P : L \hookrightarrow P, \text{true})$  holds. Thus,  $\text{nds}_k(\rho')$  and consequently also  $\text{main}_k(\rho')$  cannot be satisfied.
2. Assume that (4.2) does not hold and let

$$d := \begin{cases} \text{IC}_0(\text{sub}_{k+2}(c), C_{k+3}) & \text{if } \text{sub}_{k+2}(c) \neq \text{false} \\ \text{false} & \text{otherwise.} \end{cases}$$

Then, a morphism  $p' : C_{k+2} \hookrightarrow H$  with  $p' \not\models d$  exists, such that no morphism  $p : C_{k+2} \hookrightarrow G$  with  $\text{tr}_t op = p'$  exists. Therefore, an overlap  $P$  of  $R$  and  $C_{k+2}$  with  $i_R^P(R \setminus K) \cap i_{C_{k+2}}^P(C_{k+2}) \neq \emptyset$  exists, such that  $m \models \text{Left}(\exists(i_R^P : R \hookrightarrow P, \text{true}), \rho')$ . Hence,  $m$  does not satisfy  $\text{nui}_k(\rho')$ .

3. Assume that (4.3) does not hold. Then, a morphism  $p : C_j \hookrightarrow H$  with  $C_j$  being universally bound and  $j < k$  exists, such that no morphism  $p' : C_j \hookrightarrow G$  with  $\text{tr}_t op' = p$  exists. Then, an overlap  $P$  of  $C_j$  and  $R$  with  $i_R^P(R \setminus K) \cap i_{C_j}^P(C_j) \neq \emptyset$  exists, such that  $m \models \text{Left}(\exists(i_R^P : R \hookrightarrow P, \text{true}), \rho')$ . Hence,  $m \not\models \text{nui}_k(\rho')$ .
4. Assume that (4.4) does not hold. Then, a morphism  $p : C_j \hookrightarrow G$  with  $C_j$  being existentially bound and  $j \leq k$  exists, such that  $\text{tr}_t op$  is not total. Then, an overlap  $P$  of  $C_j$  and  $L$  with  $i_L^P(L \setminus K) \cap i_{C_j}^P(C_j) \neq \emptyset$  exists, such that  $m \models \exists(i_L^P : L \hookrightarrow P, \text{true})$ . Hence,  $m \not\models \text{ned}_k(\rho')$ .

If  $k = \text{nl}(c) - 1$ , it follows immediately that  $G \models c$ . Therefore, we need to show that  $H \models c$ . It follows that  $\text{nds}_k(\rho') = \text{true}$ . Assume that  $H \not\models c$ . Therefore, either an occurrence  $p : C_j \hookrightarrow H$  of an universally bound graph  $C_j$  exists such that no  $q : C_j \hookrightarrow G$  with  $p = \text{tr}_t op$  exists or an occurrence  $p' : C_{j'} \hookrightarrow G$  of an existentially bound graph  $C_{j'}$  exists, such that  $\text{tr}_t op'$  is not total. If the first case applies, an overlap  $P$  of  $C_j$  and  $R$  with  $i_{C_j}^P(C_j) \cap i_R^P(R \setminus K) \neq \emptyset$  exists, such that  $m \models \text{Left}(\neg\exists(i_R^P : R \hookrightarrow P, \text{true}), \rho')$  and therefore  $m \not\models \text{nui}_k(\rho')$ . If the second case applies, an overlap  $P$  of  $C_{j'}$  and  $L$  with  $i_{C_{j'}}^P(C_{j'}) \cap i_L^P(L \setminus K) \neq \emptyset$  exists, such that  $m \models \neg\exists(i_L^P : L \hookrightarrow P, \text{true})$  and therefore  $m \not\models \text{ned}_k(\rho')$ . By contradiction follows that  $H \models c$ .

In total follows that  $\rho$  is a consistency maintaining rule at layer  $k$  w.r.t.  $c$ .  $\square$

For an application condition, such that a rule equipped with it is consistency increasing at layer, we have additionally to ensure that at least one violation will be removed. To check this via an application condition, it is necessary (a) to check that an occurrence  $p$  of the universally bound graph  $C_{k_{\max}+2}$  exists, such that  $p$  and the match  $m$  do overlap, i.e.  $p(C_{k_{\max}+2}) \cap m(L) \neq \emptyset$ , and, if the sub-condition at layer  $k_{\max}+2$  is not equal to `false`, (b) that  $p$  does not satisfy  $c' := \exists C_{k_{\max}+3}$ . Only in this case, it is possible that the transformation does remove a violation. To ensure that  $p$  does not satisfy  $c'$ , the non-existence of all possible overlaps  $P$  of  $L$  and  $C_{k_{\max}+3}$  such that  $p \models c'$  has to be checked. For this, we introduce *extended overlaps*. Intuitively, given an overlap  $C$  of  $L$  and  $C_{k_{\max}+2}$  with  $p : C_{k_{\max}+2} \hookrightarrow C$ , the overlap is extended with elements of  $C_{k_{\max}+3}$  such that  $p \models C_{k_{\max}+3}$ .

**Definition 5.3 (extended overlaps).** Let graphs  $C_0, C_1, G$  and morphisms  $i_{C_0}^G : C_0 \hookrightarrow G$  and  $i_{C_0}^{C_1} : C_0 \hookrightarrow C_1$  be given. The set of extended overlaps of  $G$  with  $i_{C_0}^G$  and  $i_{C_0}^{C_1}$ , denoted by  $\text{eol}(G, i_{C_0}^G, i_{C_0}^{C_1})$  is defined as:

$$\text{eol}(G, i_{C_0}^G, i_{C_0}^{C_1}) := \{P \in \text{ol}(G, C_1) \mid i_G^P \circ i_{C_0}^G \models \exists(i_{C_0}^{C_1} : C_0 \hookrightarrow C_1, \text{true})\}$$

Via the notion of extended overlaps we are now able to check that a violation exists, to decide whether a transformation is able to remove a violation at all. It remains to check whether such a violation will be removed.

In the definition below,  $\text{exv}(\cdot, \cdot)$  and  $\text{remv}(\cdot, \cdot)$  ensure that a violation will be removed, with  $\text{exv}(\cdot, \cdot)$  ensuring that a violation is present and  $\text{remv}(\cdot, \cdot)$  ensuring that this violation will be removed. Note, that the construction of these is divided in two cases. Firstly, either  $k \leq \text{nl}(c) - 3$  and secondly,  $k = \text{nl}(c) - 2$  and the constraint ends with a condition of the form  $\forall(a : C_0 \hookrightarrow C_1, \text{false})$ .

For  $\text{exv}(\cdot, \cdot)$ , the first case will be checked via extended overlaps as already described above. In the second case, it is sufficient to check whether an occurrence  $p$  of  $C_{\text{nl}(c)}$  with  $m(L) \cap p(C_{\text{nl}(c)}) \neq \emptyset$  exists.

For  $\text{remv}(\cdot, \cdot)$ , in the first case, a violation can be removed by either deleting an occurrence  $p$  of  $C_{k+2}$  or inserting elements of  $C_{k+3}$ , such that  $p \not\models \exists C'$  and  $\text{tr}_t \circ p \models \exists C'$  for a graph  $C' \in \text{IG}(C_k k + 2, C_{k+3})$ . In the second case, a violation can only be removed by deleting an occurrence  $p$  of  $C_1$ . This will only occur if  $m(L \setminus K) \cap p(C_1) \neq \emptyset$ .

**Definition 5.4 (consistency increasing application condition).** Let a rule  $\rho = (\text{ac}, \rho')$  with  $\rho' = L \hookrightarrow K \hookrightarrow R$  and a constraint  $c$  in UANF be given. Let  $0 \leq k < \text{nl}(c)$  be even, i.e.  $\text{sub}_k(c)$  is universally bound, and  $C \in \text{IG}(C_{k+1}, C_{k+2})$  if  $\text{sub}_{k+1}(c) \neq \text{false}$  and  $C = C_{k+1}$  otherwise. The increasing application condition of  $c$  for  $\rho$  at layer  $k$  with  $C$  is defined as

$$\text{incr}_k(C, \rho) := \text{ac} \wedge \text{main}_{k-1}(\rho) \wedge \left( \bigvee_{P \in \text{ol}(L, C_{k+1})} \text{exv}(P, C) \wedge \text{remv}(P, C) \right) \quad (5.1)$$

with

1. Let  $a^r : C_{k+1} \hookrightarrow C$  be the restricted morphism of  $a_{k+1}$  and  $i_L^P$  and  $i_P^Q$  the inclusions of  $L$  in  $P$  and  $P$  in  $Q$ , respectively:

$$\text{exv}(P, C') := \begin{cases} \exists(i_L^P : L \hookrightarrow P, \text{true}) & \text{if } \text{sub}_{k+1}(c) = \text{false} \\ \exists(i_L^P : L \hookrightarrow P, \bigwedge_{Q \in \text{eol}(P, i_{C_{k+1}}^P, a^r)} \neg \exists(i_P^Q : P \hookrightarrow Q, \text{true})) & \text{otherwise} \end{cases}$$

2. If  $i_L^P(L \setminus K) \cap i_{C_{k+1}}^P(C_{k+1}) \neq \emptyset$ , we set

$$\text{remv}(P, C') := \text{true}$$

Otherwise, let  $P'$  be the graph derived by the transformation  $P \Rightarrow_{\rho, m} P'$ . Then,  $P'$  is an overlap of  $R$  and  $C_{k+1}$ . If this transformation does not exist, we set  $\text{remv}(P, C') := \text{false}$ . Let  $a^r : C_{k+1} \hookrightarrow C$  be the restricted morphism of  $a_{k+1}$ , then

$$\text{remv}(P, C') := \begin{cases} \text{false} & \text{if } \text{sub}_{k+2}(c) = \text{false} \\ \bigvee_{Q \in \text{eol}(P', i_{C_{k+1}}^{P'}, a^r)} \text{Left}(\forall(i_R^Q : R \hookrightarrow P', \exists(i_{P'}^Q : P' \hookrightarrow Q, \text{true})), \rho) & \text{otherwise.} \end{cases}$$

Note, that, in case that no occurrence of  $C_{k+1}$  not satisfying  $\exists C_{k+2}$  will be removed,  $\text{incr}_k(C', \rho)$  for any  $C' \in \text{IG}(C_{k+1}, C_{k+2})$  will only be evaluated with **true** if an occurrence  $p$  of  $C_{k+1}$  with  $p \not\models \exists(a_{k+1}^r : C_{k+1} \hookrightarrow C', \text{true})$  and  $\text{tr}_t op \models \exists(a_{k+1}^r : C_{k+1} \hookrightarrow C', \text{true})$  exists. For any smaller improvements, i.e a similar improvement for a sub-graph  $C'' \in \text{IG}(C_{k+1}, C_{k+2})$  of  $C'$ ,  $\text{incr}_k(C', \rho)$  would be evaluated with **false**. For any bigger improvements, i.e the same improvement for a super-graph  $C'' \in \text{IG}(C_{k+1}, C_{k+2})$  of  $C'$ ,  $\text{incr}_k(C', \rho)$  would also be evaluated with **false**, if  $p \models \exists(a_{k+1}^r : C_{k+1} \hookrightarrow C', \text{true})$ . In both cases, the application condition would prohibit the transformation, even if it would be consistency increasing. To resolve this problem, multiple application conditions could be combined by

$$\bigvee_{C' \in \text{IG}(C_{k+1}, C_{k+2})} \text{incr}_k(C', \rho).$$

This application condition will be evaluated with **true** if the cases described above do appear, with the drawback that this leads to a huge condition, even if duplicate conditions are removed. At least all duplicates of **main()** can be removed, since they are identical for each  $\text{incr}_k(C', \rho)$  and only need to be constructed once.

In general, these application conditions are a trade-off between conditions-size and restrictiveness. They are very restrictive, since they do not allow any deletions of occurrences of existentially bound and insertions of universally bound graphs. For example, any of these application conditions with the rule **moveFeature** and constraint  $c_1$  will be equivalent to **false**; the maintaining part of the condition will always be evaluated with **false** since **moveFeature** does remove elements of the existentially bound graph  $C_2^1$ . A change of the conditions constructed by **main()** such that it is checked whether two

nodes of the type `Feature` are connected to a node `Class` will yield application conditions that are satisfiable with `moveFeature`, but for a similar rule moving two nodes of type `Feature`, this newly constructed `nds()` would still be evaluated with `false`. Therefore, this only leads to a slight decrease of restrictiveness.

The conditions constructed by `ned()` and `nui()` could be changed in a similar fashion. For `ned()` and the universally bound graph  $C_j$ , by checking whether there does exist an additional occurrence  $p$  of  $C_{j+1}$  such that  $p \models \text{sub}_{j+2}(\text{cut}_{k_{\max}}(c))$  and for `nui()`, by checking whether an introduced occurrence  $p$  of  $C_j$  does satisfy  $\text{sub}_{j+1}(\text{cut}_{k_{\max}}(c))$ . The construction of these is similar to the construction of consistency guaranteeing application conditions as introduced by Habel and Pennemann [4], which is known to construct huge application conditions. Also, they do get more and more restrictive for increasing  $k$ , since the number of conditions constructed by `ned()` and `nui()` also increases.

**Example 5.2.** Consider the rule `assignFeature` of figure 11 and constraint  $c_1$  of figure 5. The application condition of  $c_1$  at layer 1 with  $C_2^1$  for `assignFeature` constructed by definition 5.4 is shown in figure 12. The first two rows are conditions constructed by `exv( $\cdot, \cdot$ )` and the third row is the condition constructed by `remv( $\cdot, \cdot$ )`. Note that `ned()`, `nui()` and `nwo()` did not construct any conditions since `assignFeature` does not create elements of  $C_1^1$  and does not delete elements of  $C_2^1$ .

Additionally, this application condition is also a consistency improving application condition w.r.t  $c_2$ .

Let us now show that the construction above generates consistency increasing application conditions.

**Theorem 5.2.** Let a constraint  $c$  in UANF be given. Each rule  $\rho = (\text{ac}', \rho')$  with  $\text{ac}' = \text{ac} \wedge \text{incr}_k(C, \rho)$  with  $0 \leq k < \text{nl}(c)$  being even and  $C = Ck + 1$  if  $\text{sub}_{k+1}(c) = \text{false}$  and  $C \in \text{IG}(C_{k+1}, C_{k+2})$  otherwise. Then,  $\rho$  is a consistency increasing rule at layer  $k - 1$  w.r.t.  $c$ .

*Proof.* Let a transformation  $t : G \implies_{\rho} H$  with  $\text{k}_{\max}(c, G) = k - 1$  be given. Since  $\text{main}_{k-1}(\rho)$  is contained in  $\text{incr}_k(C, \rho)$ ,  $t$  is a consistency maintaining transformation at layer  $k - 1$  with Theorem 5.1. It remains to show that  $t$  satisfies (4.5).

Let  $\text{sub}_k(c) = \forall(a_k : C_k \hookrightarrow C_{k+1}, e)$  be the sub-condition of  $c$  at layer  $k$ .

1. If  $e = \text{false}$ , assume that (4.5) does not hold, then, no morphism  $p : C_{k+1} \hookrightarrow G$  exists, such that  $\text{tr}_t \circ p$  is not total. Therefore, no overlap  $P$  of  $L$  and  $C_{k+1}$  with  $i_L^P(L \setminus K) \cap i_{C_{k+1}}^P(C_{k+1}) \neq \emptyset$  exists. It follows that  $\text{remv}(P, C') = \text{false}$  and  $m \not\models \text{incr}_k(C, \rho)$
2. Otherwise, let  $P \in \text{ol}(L, C_{k+1})$ . We show that  $m \models \text{exv}(P, C) \wedge \text{remv}(P, C)$  implies that (4.5) holds. If  $m \models \text{exv}(P, C)$ , there does exist a morphism  $p : P \hookrightarrow G$  with  $m = p \circ i_L^P$  and  $p \models \neg \exists(i_P^Q : P \hookrightarrow Q, \text{true})$  for all  $Q \in \text{eol}(P, a^r)$ . Therefore,  $q \not\models \exists(a_{k+1} : C_{k+1} \hookrightarrow C_{k+2}, \text{true})$  with  $q = p \circ i_{C_{k+1}}^P$ .

If  $i_L^P(L \setminus K) \cap i_{C_{k+1}}^P(C_{k+1}) \neq \emptyset$ ,  $\text{tr}_t \circ q$  is not total, since all occurrence  $i$  of  $C_{k+1}$  with  $i = p' \circ i_{C_{k+1}}^P$ ,  $p' : P \rightarrow G$  and  $m = p' \circ i_L^P$  in  $G$  will be removed by  $t$ . Otherwise,  $\text{tr}_t \circ q$  is total and there does exist a morphism  $p : P' \rightarrow H$  such that  $\text{tr}_t \circ q = p \circ i_{C_{k+1}}^{P'}$ . Since  $m \models \text{remv}(P, C)$ , all morphisms  $p \circ i_{C_{k+1}}^{P'}$  with  $n = p \circ i_R^Q$  satisfy  $\text{IC}_0(e, C)$ . Therefore,  $\text{tr}_t \circ q \models \text{IC}_0(e, C)$ . It follows that (4.5) is satisfied.

Therefore,  $\rho$  is a consistency increasing rule at layer  $k - 1$ .  $\square$

## 5.2 Basic Increasing and Maintaining Rules

The construction of the application conditions introduced in the previous section, as well as the constructed application conditions itself, are very complex. For a certain set of rules, which we will call *basic consistency increasing rules*, for which, we are able to construct application conditions with the same property, namely that a rule equipped with this application condition is consistency increasing at layer, in a less complex manner. The main idea is, that these rules are (a) not able to delete occurrences of existentially or insert occurrences of universally bound graphs and (b) are able to increase consistency at a certain layer. That means, given a basic increasing rule  $\rho$ , there does exist a transformation  $t : G \Rightarrow_\rho H$  such that  $t$  is a consistency increasing transformation w.r.t to a constraint  $c$ .

To ensure that (a) is met, we firstly introduce *basic consistency maintaining rules up to layer*, which means that, given a constraint, a plain rule is not able to delete existentially bound and insert universally bound graphs up to a certain layer. For the definition, we use the notion of consistency maintaining rules up to layer. The set of basic consistency maintaining rules up to layer is indeed a subset of the set of consistency maintaining rules up to layer since these rules have to be plain rules, whereas consistency maintaining rules up to layer are allowed to be equipped with application conditions, i.e.  $\text{main}(\cdot, \cdot)$ .

**Definition 5.5 (basic consistency maintaining rule up to layer).** Let a plain rule  $\rho$  and a constraint  $c$  in UANF be given. Then,  $\rho$  is called *basic consistency maintaining rule up to layer*  $-1 \leq k < \text{nl}(c)$  w.r.t.  $c$  if  $\rho$  is a direct consistency maintaining rule at layer  $k$ .

**Example 5.3.** Consider the rules `moveFeature` and `assignFeature` given in Figure 11 and constraint  $c_1$  given in Figure 5. The rule `assignFeature` is a basic consistency maintaining rule up to layer 1 w.r.t.  $c$ , whereas `moveFeature` is not a basic consistency maintaining rule.

Since infinite many transformations via a plain rule  $\rho$  exist, it is impossible to check whether  $\rho$  is a basic consistency maintaining rule up to layer based on the definition above. Therefore, we present a characterisation of basic consistency maintaining rules, which only relies on  $\rho$  itself. First, let us assume that  $\rho$  is able to create occurrences of a universally bound graph  $C_j$ . This is possible if (a)  $\rho$  does insert an edge of  $C_j \setminus C_{j-1}$  which connects already existing nodes of  $C_j$ , since it is unclear whether this would create

a new occurrence of  $C_j$ , or (b) if  $\rho$  does insert a node  $v$  of  $C_j$  such that all edges  $e \in E_{C_j}$  with  $\text{src}(e) = v$  or  $\text{tar}(e) = v$  are also inserted. If at least one of these edges is not inserted, it is guaranteed that this insertion does not create an occurrence of  $C_j$  since  $v$  is only connected to edges that have also been inserted by  $\rho$ .

Second, let us assume that  $\rho$  is able to delete occurrences of a existentially bound graph  $C_j$ . This is possible if (a)  $\rho$  does delete an edge of  $C_j \setminus C_{j-1}$  or (b)  $\rho$  deletes a node  $v$  of  $C_j \setminus C_{j-1}$  such that all edges  $e \in E_{C_j}$  with  $\text{src}(e) = v$  or  $\text{tar}(e) = v$  are also deleted. If  $\rho$  deletes a node  $c$  of  $C_j \setminus C_{j-1}$  without all its connected edges in  $C_j$  there does not exist a transformation via  $\rho$  such that an occurrence of  $C_j$  is deleted by deleting this node since the dangling edge condition would not be satisfied. A rule satisfying this properties does not decrease the satisfaction at layer. Additionally, we have to ensure that the number of violations will not be increased. For this, we have to check that  $\rho$  is not able to insert occurrences of the corresponding universally bound, as described above, and that  $\rho$  is not able to remove occurrence of any intermediate graph. This is only ensured, if  $\rho$  does not remove any elements of  $C'$  if the set of intermediate graphs is given by  $\text{IG}(C, C')$ .

To check that a plain rule satisfies these properties, we make use of the dangling edge condition, or in other words, we check that the rule is not applicable at certain overlaps of  $L$  and an existentially bound graph or that the inverse rule is not applicable at certain overlaps of  $R$  and an universally bound graph.

**Lemma 5.6.** *Let a plain rule  $\rho = L \xleftarrow{l} K \xrightarrow{r} R$  and a constraint  $c$  in EANF be given. Then,  $\rho$  is a basic consistency maintaining rule up to layer  $-1 \leq k < \text{nl}(c)$  w.r.t  $c$  if 1 and 2 apply for all  $k$  and 3 applies if  $k \leq \text{nl}(c) - 3$  and  $\text{sub}_k(c)$  is existentially bound, i.e.  $k$  is odd.*

1. Let  $E$  be the set of all existentially bound graphs  $C_j$  with  $0 \leq j \leq k+1$ . For each graph  $C \in E$  and each overlap  $P \in \text{ol}(L, C)$  with  $i_L^P(L \setminus K) \cap i_C^P(C) \neq \emptyset$ , the transformation

$$t : P \Longrightarrow_{\rho, i_L^P} H$$

does not exist.

2. Let  $U$  be the set of all universally bound graphs  $C_j$  with  $0 \leq j \leq k+2$ . For each graph  $C \in U$  and each overlap  $P \in \text{ol}(R, C)$  with  $i_R^P(R \setminus K) \cap i_C^P(C) \neq \emptyset$  the transformation

$$t : P \Longrightarrow_{\rho^{-1}, i_R^P} H$$

does not exist.

3. For all graphs  $P \in \text{ol}(L, C_{k+3})$  it holds that

$$i_L^P(L \setminus K) \cap i_{C_{k+3}}^P(C_{k+3} \setminus C_{k+1}) = \emptyset$$

*Proof.* Let  $\rho = L \xleftarrow{l} K \xrightarrow{r} R$  be a rule that satisfies the characterisations listed in Lemma 5.6 with  $-1 \leq k < \text{nl}(c)$ . Let us assume that  $\rho$  is not a direct consistency maintaining

rule up to layer  $k$  w.r.t.  $c$ . Then, there exists a transformation  $t : G \Rightarrow_{\rho, m} H$  with  $k_{\max}(c, G) = k$  such that  $t$  is not direct consistency maintaining w.r.t.  $c$ . Therefore, either (4.1), (4.2), (4.3) or (4.4) is not satisfied.

1. If (4.1) is not satisfied, then  $k < \text{nl}(c) - 2$ , let  $j = k_{\max}(c, G) + 2$ , there does exist an occurrence  $p : C_j \hookrightarrow G$  such that  $p \models \text{IC}_0(\text{sub}_j(c), C)$  and  $\text{tr}_t \circ p \models \text{IC}_0(\text{sub}_j(c), C)$  with  $C \in \text{IG}(C_j, C_{j+1})$ . Since  $i_L^P(L \setminus K) \cap i_{C_{k+3}}^P(C_{k+3} \setminus C_{k+1}) = \emptyset$  for all  $P \in \text{ol}(L, C_{k+3})$  this case can to apply since  $\rho$  does not delete any elements of  $C_{j+1} \setminus C_j$ .
2. If (4.2) is not satisfied, there does exist an occurrence  $p : C_j \hookrightarrow H$  such that no morphism  $q : C_j \hookrightarrow G$  with  $p = \text{tr}_t \circ q$  exists and  $p \not\models \text{false}$  if  $\text{sub}_j(c) = \text{false}$  and  $p \not\models \text{IC}_0(\text{sub}_j(c), C_j + 1)$  otherwise. Since  $\rho$  satisfies 2  $G$  must have dangling edges and therefore,  $G$  is not a graph.
3. If (4.3) is not satisfied, there does exist an occurrence  $p : C_j \hookrightarrow H$  of an universally bound graph with  $j \leq k_{\max}(c, G)$  such that no morphism  $q : C_j \hookrightarrow G$  with  $\text{tr}_t \circ q = p$  exists. Again, since  $\rho$  satisfies 2  $G$  must have dangling edges and therefore,  $G$  is not a graph.
4. If (4.4) is not satisfied, there does exist an occurrence  $p : C_j \hookrightarrow H$  of an existentially bound graph with  $j \leq k_{\max}(c, G)$  such that  $\text{tr}_t \circ p$  is not total. Since  $\rho$  satisfied 1 the dangling edge condition would not bee satisfied and  $H$  would contain dangling edges.

In total follows that  $\rho$  is a direct basic consistency maintaining rule up to layer  $k$ .  $\square$

Now, we are ready to introduce *basic increasing rules at layer  $k$*  with  $k$  being odd. The set of basic increasing rules is a subset of the set of maintaining rules at layer  $k$  which ensures that the satisfaction at layer as well as the number of violation will not be decreased. Additionally, the left-hand side of these rule do contain an occurrence  $p$  of the universally bound graph  $C_{k+2}$ , such that this occurrence either will be removed, i.e. elements of  $C_{k+2} \setminus C_{k+1}$  will be deleted, or an intermediate graph  $C \in \text{IG}(C_{k+2}, C_{k+3})$  will be inserted. Of course, this second case only occurs if  $k < \text{nl}(c) - 3$  with  $c$  being the respective constraint. This property yields the advantage that application conditions for basic increasing rules are less complex and smaller, since it can be exactly determined how this rule removes a violation and therefore, no overlaps have to be considered.

This, on first sight seems like a restriction of the set of basic increasing rules but the context of each rule  $\rho$  that does satisfy all properties of a basic increasing rule excluding that  $C_{k+2}$  is a sub-graph of the left-hand side can be expanded such that  $\rho$  is a basic increasing rule and the semantic of  $\rho$  is not increased. Later on, a method to derive this rules will be presented.

Basic increasing rules at layer  $k$  are called *deleting basic increasing rules* if  $p$  will be removed and *inserting basic increasing rules* if  $\text{sub}_k(c)$  an intermediate graph will be inserted. For our repairing process, we will introduce the restriction that deleting basic increasing rules are only allowed to delete edges but no nodes of  $C_{k+2}$  since otherwise it is not possible, given a rule set and a constraint to decide whether this rules set is able

to repair an arbitrary graph based only on deleting basic increasing rules. For example, consider a rule that deletes a node of  $C_{k+2}$ . Then, it is unknown whether this node is connected to nodes not belonging to  $C_{k+2}$  and it is unclear whether all occurrence of  $C_{k+2}$  could be destroyed by  $\rho$  since the dangling edge condition might be unsatisfied.

**Definition 5.7 (basic increasing rule).** Let a constraint  $c$  in UANF and a direct consistency maintaining rule  $\rho = (\text{ac}, L \xleftarrow{l} K \xrightarrow{r} R)$  up to layer  $-1 \leq k < \text{nl}(v) - 1$ , with  $k$  odd, be given. Then,  $\rho$  is called basic increasing w.r.t  $c$  at layer  $k$  if a morphism  $p : C_{k+2} \hookrightarrow L$ , called the increasing morphism, exists such that either 1 or 2 applies.

1.  $r \circ l^{-1} \circ p$  is not total. Then,  $\rho$  is called a deleting basic increasing rule.
2. If  $k < \text{nl}(c) - 2$ , there does exist an intermediate graph  $C \in \text{IG}(C_{k+2}, C_{k+3})$  such that  $p \not\models \exists(a_{k+2}^r : C_{k+2} \hookrightarrow C, \text{true})$ ,  $r \circ l^{-1} \circ p$  is total and  $r \circ l^{-1} \circ p \models \exists(a_{k+2}^r : C_{k+2} \hookrightarrow C, \text{true})$ . Then,  $\rho$  is called a inserting basic increasing rule with  $C$ .

**Example 5.4.** Consider the rule `assignFeature` given in Figure 11 and constraint  $c_1$  given in Figure 5. Then, `assign Feature` is a inserting basic rule with  $C_2^2 \in \text{IG}(C_1^1, C_2^1)$  w.r.t.  $c_1$  but not a inserting basic rule with respect to the constraint  $\forall(C_2^2, \exists C_2^1)$  since the left-hand side of `assignFeature` does not contain an occurrence of  $C_2^2$ .

Again, 5.7 relies on every transformation of a rule  $\rho$ . Therefore, we present an alternative method to determine whether  $\rho$  satisfies 5.7 or not by checking that  $\rho$  does not delete any edges or nodes of  $C_{k+1} \setminus C_k$ .

As mentioned above, given a consistency maintaining rule  $\rho$  we can derive basic increasing rules that are only applicable if at a match and graph if  $\rho$  is applicable.

**Definition 5.8 (derived rules).** Let a constraint  $c$  in UANF and an rule  $\rho = (\text{ac}, L \xleftarrow{l} K \xrightarrow{r} R)$  be given. The set of derived rules of  $\rho$  at layer  $0 \leq k \leq \text{nl}(c) - 1$  contains rules characterized in the following way: Let

$$\mathbf{G} := \begin{cases} \text{IG}(C_k, C_{k+1}) & \text{if } \text{sub}_k(c) \text{ is existentially bound} \\ \{C_k\} & \text{otherwise} \end{cases}$$

For each  $P \in \mathbf{G}$  and  $L' \in \text{ol}(L, P)$ : If the diagram shown in Figure 13 is a transformation, i.e. (1) and (2) are pushouts, and for the characterisations of Definition 5.7 holds that

$$\begin{aligned} \rho \text{ satisfies 1} &\implies L' \xleftarrow{l'} K' \xrightarrow{r'} R' \text{ satisfies 1} \wedge \\ \rho \text{ satisfies 2} &\implies L' \xleftarrow{l'} K' \xrightarrow{r'} R' \text{ satisfies 2} \end{aligned}$$

the rule

$$\rho' = (\text{Shift}(\text{ac}, i_L^{L'}), L' \xleftarrow{l'} K' \xrightarrow{r'} R')$$

is a derived rule of  $\rho$  at layer  $k$ .

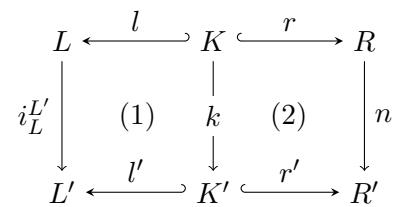


Figure 13: Pushout diagram for the construction of basic increasing rules.

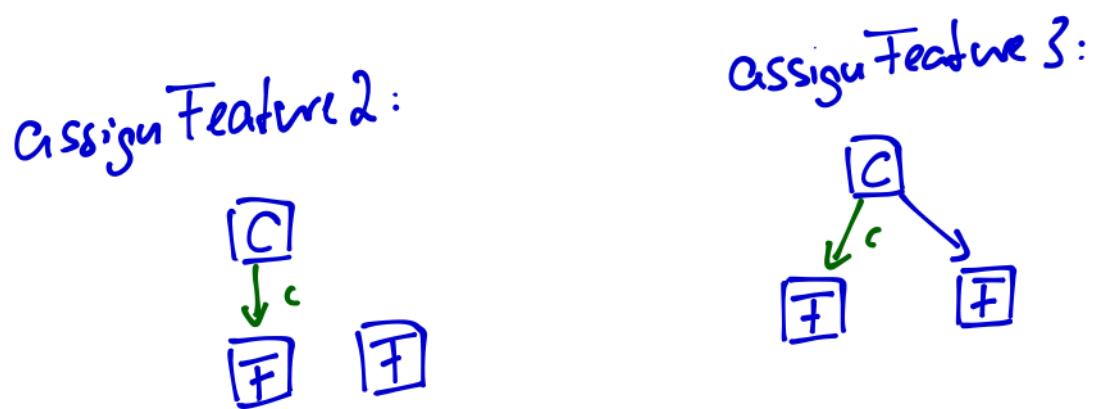


Figure 14: Derived rules of `assignFeature` and  $c_1$ .

**Example 5.5.** Consider the rule `assignFeature` given in Figure 11 and constraint  $c_1$  given in Figure 5. The set of derived rules of  $\rho$  at layer 1 is given in Figure 14.

Obviously, a rule  $\rho'$  contained in the set of derived rules of a rule  $\rho$  is only applicable at a match  $m'$  if  $\rho$  is applicable at match  $m$  with  $m = m' \circ i$  with  $i$  being the inclusion of the left-hand side of  $\rho$  into the left-hand side of  $\rho'$ . Therefore, given a rules set  $\mathcal{R}$ , the extension of  $\mathcal{R}$  by the set of all derived rules for each rule of  $\mathcal{R}$  does not extend the expressiveness of  $\mathcal{R}$ . The main idea of the concept of derived rules is the extension of a given rule set by as many basic increasing rules as possible without extending the expressiveness of this set.

**Lemma 5.9.** Let a constraint  $c$  in UANF, and a rule  $\rho = (\text{ac}, L \xleftarrow{l} K \xrightarrow{r} R)$  be given, such that  $\rho$  is a maintaining rule up to layer  $-1 \leq k < \text{nl}(c) - 1$ , with  $k$  odd, and satisfies 1 and 2 of Definition 5.7. Then, each rule contained in the set of derived rules of  $\rho$  at layer  $k$  is a basic increasing rule.

*Proof.* Let  $\rho' = (\text{ac}', L' \xleftarrow{l'} K' \xrightarrow{r'} R')$  be one of those derived rules. Since  $\rho'$  deletes an inserts exactly the same elements as  $\rho$  and  $m' \models \text{ac}' \iff m' \circ i_L^{L'} \models$ ,  $\rho'$  is a consistency maintaining rule up to layer  $k$  and satisfies 1 and 2 of Definition 5.7. Since a morphism  $i : C_k \hookrightarrow L$  exists,  $\rho'$  is a basic increasing rule at layer  $k$ .  $\square$

In transformations via a rule  $\rho$ , such that the match intersects with an occurrence of an universally bound graph  $C_k$ ,  $\rho$  can be replaced by a derived rule of  $\rho$  at layer  $k$ .

**Lemma 5.10.** Let a constraint  $c$  in UANF and a rule  $\rho = (\text{ac}, L \xleftarrow{l} K \xrightarrow{r} R)$  be given. Then, for each transformation

$$t : G \Longrightarrow_{\rho, m} H$$

such that an occurrence  $p : C_k \hookrightarrow G$  of a universally bound graph  $C_k$  with  $p(C_k) \cap m(L) \neq \emptyset$  exists, there does exist a transformation

$$t' : G \Longrightarrow_{\rho', m'} H$$

with  $\rho'$  being a derived rule of  $\rho$  at layer  $k$ .

*Proof.* Since  $p(C_k) \cap m(L) \neq \emptyset$  there does exist an overlap  $P \in \text{ol}(C_k, L)$  such that a morphism  $q : \hookrightarrow G$  with  $m = q \circ i_L^P$  and  $p = q \circ i_{C_k}^P$  exists. Since  $t$  exists, there does exist a derived rule  $\rho' = (\text{ac}', L' \xleftarrow{l'} K' \xrightarrow{r'} R)$  with  $L' = P$ . We set  $m' = p$ , since  $m = m' \circ i_L L' \models \text{ac}$  it follows that  $m' = \text{ac}'$ . Since  $\rho$  does removes and inserts the same elements as  $\rho'$ , there does exist a transformation  $t' : G \Longrightarrow_{\rho', m'} H$ .  $\square$

Via this, we are able to replace consistency increasing transformations via a rule  $\rho$  that is consistency maintaining up to layer  $k$  and satisfies satisfies 1 and 2 of Definition 5.7 by a derived rule of  $\rho$  at layer  $k$ , that is, a basic increasing rule at layer  $k$ .

### 5.3 Application Conditions for Basic Rules

Let us now introduce the application conditions for basic increasing rules. Since these rules are maintaining rules up to a certain layer  $k$ , given a basic increasing rule  $\rho$ , it is sufficient to check whether  $m \circ i \not\models \exists C_{k+1}$  with  $i$  is a deleting rule and do check whether  $m \circ i \models \exists C$  if  $\rho$  is a inserting rule with  $C$  with  $i$  being the increasing morphism of  $\rho$ .

**Definition 5.11 (application conditions for basic increasing rules).** Let a constraint  $c$  in UANF and a basic increasing rule  $\rho = (\text{ac}, L \xleftarrow{l} K \xrightarrow{r} R)$  w.r.t  $c$  at layer  $-1 \leq k < \text{nl}(c) - 1$ , with  $k$  odd, be given. The basic application condition of  $\rho$  w.r.t.  $c$  at layer  $-1 \leq j < \text{nl}(c)$  is given by

$$\text{ac}' = \text{ac} \wedge \text{basic}_j(\rho)$$

with

$$\text{basic}_j(\rho) := \begin{cases} \bigwedge_{P \in \text{eol}(L, a_{k+2}^r, i)} \neg \exists(i_L^P : L \hookrightarrow P, \text{true}) & \text{if } j = k \text{ and } k < \text{nl}(c) - 2 \\ \text{true} & \text{if } k = \text{nl}(c) - 2 \\ \text{false} & \text{otherwise} \end{cases}$$

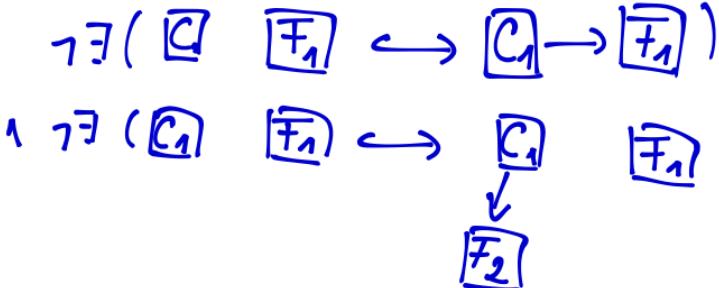
and  $a_{k+2}^r = a_{k+2}$ , if  $\rho$  is a deleting rule,  $a_{k+2}^r : C_{k+2} \hookrightarrow C$  if  $\rho$  is an inserting rule with  $C$  and  $i$  being the increasing morphism of  $\rho$ .

This application conditions are way easier to construct and smaller than the ones constructed by Definition 5.4. Note, that in case of a inserting basic rule  $\rho$  that inserts an intermediate graph  $C$ , the application condition only checks whether the increasing morphism does not satisfy  $\exists C$ . But, an application of this rule could also lead to a consistency increasing transformation w.r.t.  $c$  if the increasing morphism satisfied  $\exists C$ , if another intermediate graph  $C'$  will be inserted. To check this, conditions similar to the ones constructed via Definition 5.4 need to be constructed. On first sight, this seams like a restriction, but via the notion of derived rules we are able to dissolve this restriction, since the set of derived rules of  $\rho$  will contain a inserting basic increasing rule with  $C'$ , such that this rule, equipped with the according basic application condition can be used to perform this consistency increasing transformation. For example, consider the rule `assignFeature`, there does exist a consistency increasing transformation  $t : C_2^2 \Rightarrow_{\text{assignFeature}, m} C_2^1$  such that  $m \not\models \text{basic}_1(\text{assignFeature})$ , but there does also exist a transformation  $t : C_2^2 \Rightarrow_{\text{assignFeature3}, m'} C_2^1$  with  $m' \models \text{basic}_1(\text{assignFeature3})$ .

Let us now show that basic increasing rules equipped with the application condition constructed by Definition 5.11 are direct consistency increasing rules at layer.

**Theorem 5.3.** Let a constraint  $c$  in UANF and a basic increasing rule  $\rho = (\text{ac}, L \xleftarrow{l} K \xrightarrow{r} R)$  w.r.t  $c$  at layer  $k$  be given. Then,  $\rho' = (\text{ac} \wedge \text{basic}_k(\rho), L \xleftarrow{l} K \xrightarrow{r} R)$  is a direct consistency increasing rule at layer  $k$ .

*assign Feature:*



*assignFeature3:*

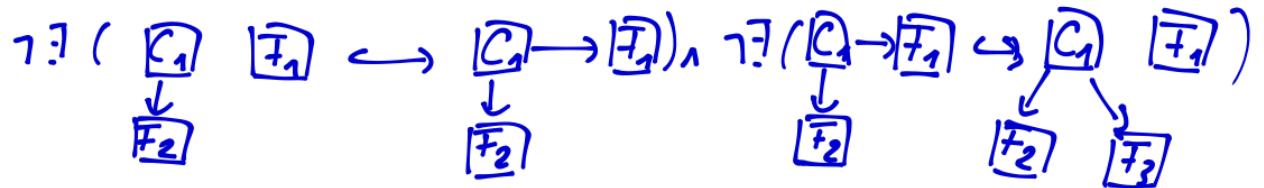


Figure 15: Application condition for `assignFeature` and `assignFeature3` with  $c_1$  at layer 1.

*Proof.* Let  $G$  be a graph with  $k_{\max} = k$ . We show that each transformation  $t : G \Rightarrow_{\rho',m} H$  is a direct consistency increasing transformation. Since,  $\rho$  is a direct basic increasing rule at layer  $k$ ,  $\rho'$  is also a consistency maintaining transformation at layer  $k$  and  $t$  satisfies (4.1), (4.2), (4.3) and (4.4). Therefore, we only need to show that  $t$  satisfies (4.5).

1. If  $\rho$  is a deleting rule,  $r \circ l^{-1} \circ i$  is not total, with  $i$  being the increasing morphism of  $\rho$ . If  $k = nl(c) - 2$ , this transformation is satisfied (4.5) since one occurrence of  $C_{k+2}$  has been destroyed. If  $k < nl(c) - 2$ , since  $m \models \text{basic}_k(\rho)$  the morphism  $m \circ i$  does not satisfy  $\exists C_{k+3}$ . Since this occurrence will be destroyed,  $t$  satisfies (4.5).
2. If  $\rho$  is a inserting rule with  $C \in \text{IG}(C_{k+2}, C_{k+3})$ , the morphism  $m \circ i$  does not satisfy  $\exists C$  since  $m \models \text{basic}_k(\rho)$ . But it holds that  $\text{tr}_t \circ m \circ i \models \exists C$  and therefore  $t$  satisfies (4.5).

In total follows that  $\rho'$  is a direct consistency increasing rule at layer  $k$ .  $\square$

**Example 5.6.** Again, consider the rule `assignFeature`, its derived rule `assignFeature3` and  $c_1$ . The application condition for these rules at layer  $k$  is given in Figure 15.

## 6 Rule-based Graph Repair

In the following we present our rule-based graph repair approach. First, we propose an graph repair process for one constraint in UANF and second, a repair process for a set of constraints in UANF, both based on a given set of rules  $\mathcal{R}$ . Additionally, we need to make further assumptions for these constraints and constraint sets, namely, that they are *circular conflict free* to guarantee that our approach terminates.

### 6.1 Conflicts within Conditions

During a repair process, the insertion of elements of  $C_j \setminus C_{j-1}$ , with  $C_j$  being a existentially bound graph of a given constraint  $c$  in UANF, could also insert new occurrence of universally bound graphs  $C_i$  of  $c$ . This insertion is not problematic if  $i > k_{\max}$ , but if  $i \leq k_{\max}$  this could either lead to the insertion of new violations or a decrease of satisfaction at layer. Additionally, the removal of elements of  $C_j \setminus C_{j-1}$ , with  $C_j$  being a universally bound graph, could destroy occurrences of a existentially bound graph  $C_i$ . Again, this case can lead to an insertion of new violations or a decrease of the satisfaction at layer.

We will now introduce the notion of *conflicts within conditions*, which states that  $C_j$  has a conflict with  $C_i$  if and only if one of the cases described above can occur. Note, that conflicts can only occur between existentially and universally bound graphs and vice versa. There cannot exist a conflict between two existentially or two universally bound graphs since the insertion of elements cannot destroy occurrences of existentially bound and the removal of elements cannot insert new occurrences of universally bound graphs, respectively.

**Definition 6.1 (conflicts within conditions).** Let a condition  $c$  in UANF be given. An existentially bound graph  $C_k$  has a conflict with an universally bound graph  $C_j$  if a transformation  $t : G \xrightarrow{\rho} H$  with  $\rho = C_{k-1} \xleftarrow{\text{id}} C_{k-1} \xrightarrow{a_{k-1}} C_k$  exists such that

$$\exists p : C_j \hookrightarrow H (\neg \exists q : C_j \hookrightarrow G(\text{tr}_t \circ q = p)).$$

An universally bound graph  $C_k$  has a conflict with an existentially bound graph  $C_j$  if a transformation  $t : G \xrightarrow{\rho} H$  with  $\rho = C_k \xleftarrow{a_{k-1}^r} C \xrightarrow{\text{id}} C$  for any  $C \in \text{IG}(C_{k-1}, C_k)$  exists such that

$$\exists p : C_j \hookrightarrow G(\text{tr}_t \circ p \text{ is not total}).$$

Additionally, we introduce *conflicts graphs*, which represent the conflicts within a condition via a graph. With these, we are able to define *transitive conflicts*, *circular conflicts* and the absence of these, which will be an necessary property for the termination of our repairing process. Intuitively, as the name suggests, a condition  $c$  contains a circular conflict if a graph  $C_k$  has a conflict with itself or there does exist a sequence  $C_k = C_{j_1}, \dots, C_{j_n} = C_k$  of graphs such that  $C_{j_i}$  has a conflict with  $C_{j_{i+1}}$ . We can check this property by checking whether the conflict graph does contain specific cycles.

**Definition 6.2 (conflict graph, circular conflicts).** Let a condition  $c$  in UANF be given. The conflict graph of  $c$  is constructed in the following way. For each graph  $C_k$ ,  $0 \leq k \leq \text{nl}(c)$ , in  $c$ , there does exist a node labelled with  $k$ . For each odd  $0 \leq k < \text{nl}(c)$ , i.e.  $C_k$  is universally bound, there does exist edge  $e, e'$  of type **constraint** with  $\text{src}(e) = k$ ,  $\text{tar}(e) = k + 1$ ,  $\text{src}(e') = k + 1$  and  $\text{tar}(e') = k$ . If a conflict between  $C_k$  and  $C_j$  exists, there does exist an edge  $e$  of type **conflict** with  $\text{tar}(e) = k$  and  $\text{src}(e) = j$ .

A graph  $C_k$  has a transitive conflict with  $C_j$  if a path from  $k$  to  $j$  exists. A graph  $C_k$  has a circular conflict if  $C_k$  has a transitive conflict with itself and there does exist a cycle that contains one or more than two edges or every edge in the cycle is of type **conflict**. A condition  $c$  is called circular conflict free if  $c$  does not contain a circular conflict.

In other words, a condition  $c$  is *circular conflict free* if the conflict graph does not contain any cycles that contains more than two edges and all cycles with exactly two edges contain at least one edge of type **constraint**.

**Example 6.1.** Consider constraint  $c_3$  and the transformations  $t_1$  and  $t_2$  shown in Figure 16. Transformation  $t_1$  shows that  $C_1$  has a conflict with  $C_2$  since the rule  $\rho = C_1 \xleftarrow{\text{id}} C_1 \xrightarrow{a_1} C_2$  has been applied and there does exist a newly inserted occurrence of  $C_1$  not satisfying  $\exists(C_2, \text{true})$ . Transformation  $t_2$  shows that  $C_2$  has a conflict with  $C_1$  since the rule  $C_2 \xleftarrow{a_1} C_1 \xrightarrow{\text{id}} C_1$  has been applied and one occurrence of  $C_1$  has been destroyed. Therefore,  $c_3$  contains a circular conflict, the conflict graph of  $c_3$  is shown in Figure 17.

In general, the statement “ $C_j$  has a conflict with  $C_k$ ” does not imply that “ $C_k$  has a conflict with  $C_j$ ” as shown by constraint  $c_4$  shown in Figure 16. The conflict graph of  $c_4$  is also shown in Figure 17. Constraint  $c_4$  is circular conflict free since the only cycle in this graph contains exactly two edges and one has type **constraint**.

We will now introduce two characterisations of conflicts. One based on overlaps and the second one based on rules. For  $C_k$  being existentially and  $C_j$  being universally bound, the overlap based characterisation checks whether for each overlap of  $C_k$  and  $C_j$ , such that the inclusions of  $C_k \setminus C_{k+1}$  and  $C_j$  do overlap, the rule that only deletes  $C_k \setminus C_{k-1}$  is applicable. If this is not possible, there does not exist a transformation as described in Definition 6.2. For  $C_k$  being universally and  $C_j$  being existentially bound, the characterisation checks whether for each overlap of  $C_k$  and  $C_j$ , such that the elements of  $C_k \setminus C_{k-1}$  and  $C_j \setminus C_{j-1}$  do intersect, a rule only removing elements of  $C_k \setminus C_{k-1}$  is applicable. Again, if this is not possible, there does not exist a transformation as described in Definition 6.2.

**Lemma 6.3.** Let a constraint  $c$  in UANF be given.

- Let  $C_k$  be an existentially and  $C_j$  an universally bound graph of  $c$ . Then,  $C_k$  has a conflict with  $C_j$ , if and only if an overlap  $P \in \text{ol}(C_k, C_j)$  exists, such that

$$i_{C_k}^P(C_k \setminus C_{k-1}) \cap i_{C_j}^P(C_j \setminus C_{j-1}) \neq \emptyset$$

and the rule  $\rho = C_k \xleftarrow{a_{k-1}} C_{k-1} \xrightarrow{\text{id}} C_{k-1}$  is applicable at match  $i_{C_k}^P$ .

$$c_3 : \vee \underbrace{\square \rightarrow \top \rightarrow \perp}_{C_1}, (\exists \underbrace{\square \rightarrow \top \rightarrow \perp \rightarrow \top}_{C_2}, \text{true})$$

$$t_1 : \begin{array}{ccc} \square & \xrightarrow{\square} & \top \\ & \downarrow & \\ & \square \rightarrow \top \rightarrow \perp & \end{array} \Rightarrow \begin{array}{ccc} \square & \xrightarrow{\square} & \top \\ & \downarrow & \\ & \square \rightarrow \top \rightarrow \perp & \end{array}$$

$$t_2 : \begin{array}{ccc} \square & \xrightarrow{\square} & \top \\ & \downarrow & \\ & \square \rightarrow \top \rightarrow \perp \rightarrow \top & \end{array} \Rightarrow \begin{array}{ccc} \square & \xrightarrow{\square} & \top \\ & \downarrow & \\ & \square \rightarrow \top \rightarrow \perp & \end{array}$$

$$c_4 : \vee \square \exists \square \rightarrow \square$$

Figure 16: Constraint  $c_3$  and the transformation that show the existence of conflicts between  $C_1$  and  $C_2$  and  $C_2$  and  $C_1$ .

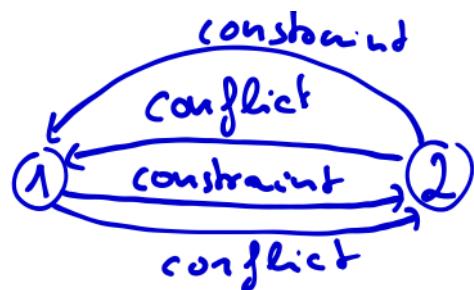


Figure 17: Conflict graphs of  $c_3$  and  $c_4$ .

2. Let  $C_k$  be an universally and  $C_j$  be an existentially bound graph of  $c$ . Then,  $C_k$  has a conflict with  $C_j$  if and only if an overlap  $P$  of  $C_k$  and  $C_j$  exists such that

$$i_{C_k}^P(C_k \setminus C_{k-1}) \cap i_{C_j}^P(C_j \setminus C_{j-1}) \neq \emptyset$$

and a rule  $\rho = C_k \xleftarrow{a_{k-1}^r} C \xrightarrow{\text{id}} C$  with  $C \in \text{IG}(C_{k-1}, C_k)$  and  $i_{C_k}^P(C_k \setminus C) \cap i_{C_j}^P(C_j \setminus C_{j-1})$  is applicable at match  $i_{C_k}^P$ .

*Proof.* Let a condition  $c$  in UANF be given.

1. “ $\implies$ ”: Let  $C_k$  be an existentially bound graph that has a conflict with an universally bound graph  $C_j$ . Then, there does exists a transformation  $t : G \implies_\rho H$  with  $\rho = C_{k-1} \xleftarrow{\text{id}} C_{k-1} \xleftarrow{a_{k-1}^r} C_k$  such that a new occurrence  $p$  of  $C_j$  has been inserted. Since only elements of  $C_k \setminus C_{k-1}$  have been inserted, it holds that  $p(C_j) \cap n(C_k \setminus C_{k-1}) \neq \emptyset$ , with  $n$  being the co-match of  $t$ . The graph  $p(C_j) \cup n(C_k)$  is the searched for overlap and the rule  $\rho^{-1} = C_k \xleftarrow{a_{k-1}^r} C_{k-1} \xrightarrow{\text{id}} C_{k-1}$  has to be applicable at match  $n$ .  
“ $\Leftarrow$ ”: Let  $C_k$  be an existentially and  $C_j$  an universally bound graph such that an overlap  $P \in \text{ol}(C_k, C_j)$  with  $i_{C_k}^P(C_k \setminus C_{k-1}) \cap i_{C_j}^P(C_j \setminus C_{j-1}) \neq \emptyset$  exists such that the rule  $\rho = C_k \xleftarrow{a_{k-1}^r} C_{k-1} \xrightarrow{\text{id}} C_{k-1}$  is applicable at match  $i_{C_k}^P$ . Then, the inverse transformation of  $t : P \implies_{\rho, i_{C_k}^P} H$  is the searched for transformation and  $C_k$  has a conflict with  $C_j$ .
2. “ $\implies$ ”: Let  $C_k$  be an universally bound graph that has a conflict with an existentially bound graph  $C_j$ . Then, a transformation  $t : G \implies_\rho H$  with  $\rho = C_k \xleftarrow{a_{k-1}^r} C \xrightarrow{\text{id}} C$  with  $C \in \text{IG}(C_{k-1}, C_k)$  exists such that  $\text{tr}_t \circ p$  is no total for one occurrence  $p : C_j \hookrightarrow G$ . Then, the graph  $p(C_j) \cup m(C_k)$  is the searched for overlap and  $i_{C_k}^P(C_k \setminus C_{k-1}) \cap i_{C_j}^P(C_j \setminus C_{j-1}) \neq \emptyset$  has to hold since  $\rho$  only deletes elements of  $C_k \setminus C_{k-1}$ .  
“ $\Leftarrow$ ”: Let  $C_k$  be universally and  $C_j$  existentially bound such that an overlap  $P$  of  $C_k$  and  $C_j$  with  $i_{C_k}^P(C_k \setminus C_{k-1}) \cap i_{C_j}^P(C_j \setminus C_{j-1}) \neq \emptyset$  exists such that a rule  $\rho = C_k \xleftarrow{a_{k-1}^r} C \xrightarrow{\text{id}} C$  with  $C \in \text{IG}(C_{k-1}, C_k)$  is applicable at match  $i_{C_k}^P$ . Then, the transformation of  $t : P \implies_{\rho, i_{C_k}^P} H$  is the searched for transformations and  $C_k$  has a conflict with  $C_j$ .

□

Our second characterisation of conflicts is based on the notion of basic maintaining rules.

**Lemma 6.4.** *Let a condition  $c$  in UANF be given.*

1. Let  $C_k$  be an existentially and  $C_j$  be an universally bound graph of  $c$ . Then,  $C_k$  has a conflict with  $C_j$  if and only if the rule  $\rho = C_{k-1} \xleftarrow{\text{id}} C_{k-1} \xrightarrow{a_{k-1}} C_k$  is not basic consistency maintaining rule up to layer 1 w.r.t.  $\forall(a_{j-1} \circ \dots \circ a_0 : C_0 \hookrightarrow C_j, \text{false})$ .
2. Let  $C_k$  be an universally and  $C_j$  be an existentially bound graph of  $c$ . Then,  $C_k$  has a conflict with  $C_j$  if and only if each rule  $\rho = C_k \xleftarrow{a_{k-1}^r} C \xrightarrow{\text{id}} C$  with  $C \in \text{IG}(C_{k-1}, C_k)$  is not a basic consistency maintaining rule up to layer 1 w.r.t.  $\exists(a_{j-1} \circ \dots \circ a_0 : C_0 \hookrightarrow C_j, \text{true})$ .

*Proof.* 1. Let  $C_k$  be an existentially and  $C_j$  an universally bound graph of  $c$ .

“ $\implies$ ”: Assume that  $C_k$  has a conflict with  $C_j$ . Therefore, there does exist a transformation  $t : G \implies_\rho H$  with  $\rho = C_{k-1} \xleftarrow{\text{id}} C_{k-1} \xrightarrow{a_{k-1}} C_k$  such that a new occurrence  $p : C_j \hookrightarrow H$  has been inserted. Therefore,  $t$  does not satisfy (4.2) and  $\rho$  is not a basic maintaining rule up to layer 1.

“ $\impliedby$ ”: Assume that  $\rho = C_{k-1} \xleftarrow{\text{id}} C_{k-1} \xrightarrow{a_{k-1}} C_k$  is not a basic maintaining rule up to layer 1 w.r.t.  $\forall(a_{j-1} \circ \dots \circ a_0 : C_0 \hookrightarrow C_j, \text{false})$ . Since this constraint only contains universally bound graphs, there must exist a transformation  $t : G \implies_\rho$  that does not satisfy (4.2). Therefore, a new occurrence of  $C_j$  has been inserted by  $t$  and with Definition 6.1 follows that  $C_k$  has a conflict with  $C_j$ .

2. Let  $C_k$  be an universally and  $C_j$  be an existentially bound graph of  $c$  and  $c = \exists(a_{j-1} \circ \dots \circ a_0 : C_0 \hookrightarrow C_j, \text{true})$ .

“ $\implies$ ”: Assume that  $C_k$  has a conflict with  $C_j$ . Therefore, there does exist a transformation  $t : G \implies_\rho H$  with  $\rho = C_k \xleftarrow{a_{k-1}^r} C \xrightarrow{\text{id}} C$ , for a  $C \in \text{IG}(C_{k-1}, C_k)$  such that an occurrence of  $C_j$  has been destroyed. Then,  $t$  does not satisfy (4.4), since it must hold that  $G \models c$ . Therefore,  $\rho$  is not a basic consistency maintaining rule w.r.t.  $c$  up to layer 1.

“ $\impliedby$ ”: Assume that  $\rho = C_k \xleftarrow{a_{k-1}^r} C \xrightarrow{\text{id}} C$  is a not a basic increasing rule w.r.t.  $c$  up to layer 1. The rule  $\rho$  is only applicable to graphs that satisfy  $c$ . Therefore, there must exist a transformation  $t : G \implies_\rho H$  that does not satisfy (4.4). Therefore, an occurrence of  $C_j$  has been removed by  $t$  and with Definition 6.1 follows that  $C_k$  has a conflict with  $C_j$ .

□

## 6.2 Repairing rule Sets

If a set of rules and a constraint is given, it is unclear whether it is possible to repair a graph using rules of this set or not. Therefore, we introduce the notion of *repairing rule sets* which is a characterisation rule sets that are capable of repairing a graph w.r.t. to a circular conflict free constraint. First, we introduce the notion of *repairing sequences*. A repairing sequence is a sequence of rule application that either destroys an occurrence of

a universally or inserts an occurrence of an existentially bound graph and is applicable at each occurrence of these graphs. To ensure that these sequences are applicable at each occurrence it needs to be ensured that no nodes of these occurrences will be removed and that the left hand side of the rule of the repairing sequence is contained in this occurrence. In other words, each repairing sequence of  $C_k$  starts with a transformation originating in  $C_k$  if  $C_k$  is universally bound and  $C_{k-1}$  if  $C_k$  is existentially bound.

**Definition 6.5 (repairing sequence).** Let a constraint  $c$  in UANF and a set of rules  $\mathcal{R}$  be given.

1. If  $C_k$  is existentially bound, a sequence of transformations

$$C_{k-1} = G_0 \xrightarrow{\rho_1, m_1} G_1 \xrightarrow{\rho_2, m_2} \dots \xrightarrow{\rho_n, m_n} G_n$$

with  $\rho_i \in \mathcal{R}$  is called a repairing sequence of  $C_k$  if  $G_n \models_k c$ ,  $\text{tr}_{t_n} \circ \dots \circ \text{tr}_{t_1} \circ \text{id}_{C_{k-1}}$  is total and the concurrent rule  $\rho = G_0 \xleftarrow{\text{id}} G_0 \xrightarrow{\text{tr}_{t_n} \circ \dots \circ \text{tr}_{t_1}} G_n$  is a basic consistency maintaining rule w.r.t.  $\forall(C_j, \text{false})$  for all graph universally bound graphs  $C_j$  such that  $C_k$  has no conflict with  $C_j$ .

2. If  $C_k$  is existentially bound, a sequence of transformations

$$C_k = G_0 \xrightarrow{\rho_1, m_1} G_1 \xrightarrow{\rho_2, m_2} \dots \xrightarrow{\rho_n, m_n} G_n$$

with  $\rho_i \in \mathcal{R}$  is called a repairing sequence of  $C_k$  if  $G \models_k c$ , for each node  $v \in V_{G_0}$  there does exist a node  $v' \in V_{G_n}$  with  $v' = \text{tr}_{t_n}(\dots \text{tr}_{t_1}(v))$  and the concurrent rule  $\rho = G_0 \xleftarrow{\text{tr}_{t_n}(\dots \text{tr}_{t_1}(G_0))} G_n$  is a basic consistency maintaining rule w.r.t.  $\forall(C_j, \text{true})$  for universally bound graphs  $C_j$ .

In both cases, the insertion of additional elements, i.e.  $G_n \neq C_{k+1}$  if  $C_k$  is existentially and  $G_n \neq C$  for all  $C \in \text{IG}(C_{k-1}, C_k)$  if  $C_k$  is universally bound, could lead to the insertion of universally bound graphs. For existentially bound graph this can occur if an overlap with an universally bound graph in a similar manner as shown in Figure 18 exists. To ensure that this case does not occur, we need the additional condition that the concurrent rule is a basic consistency maintaining rule w.r.t. certain constraints. If  $G_n = C_{k+1}$  if  $C_k$  is existentially or  $G_n \neq C$  with  $C \in \text{IG}(C_{k-1}, C_k)$  if  $C_k$  is universally bound, this condition is not needed.

**Theorem 6.1.** Let a constraint  $c$  in UANF and a set of rules  $\mathcal{R}$  be given.

1. If  $C_k$  is existentially bound and there does exist sequence

$$C_{k-1} \xrightarrow{\rho_1, m_1} \dots \xrightarrow{\rho_n, m_n} C_k$$

with  $\rho_i \in \mathcal{R}$  such that  $\text{tr}_{t_n} \circ \dots \circ \text{tr}_{t_1} \circ \text{id}_{C_{k-1}}$  is total and  $C_k \models_{k+1} c$ . Then, this is a repairing sequence for  $C_k$ .

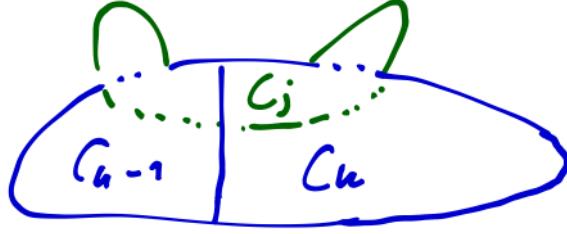


Figure 18: Scheme of a overlap of existentially bound graph  $C_k$  and universally bound graph  $C_j$  that could lead to an insertion of  $C_j$  via repairing sequences.

2. If  $C_k$  is universally bound and there does exist a sequence

$$C_k \Rightarrow_{\rho_1, m_1} \dots \Rightarrow_{\rho_n, m_n} C$$

with  $\rho_i \in \mathcal{R}$ ,  $C \in \text{IG}(C_{k-1}, C_k)$  and for each node  $v \in V_{G_0}$  there does exist a node  $v' \in V_{G_n}$  with  $v' = \text{tr}_{t_n}(\dots \text{tr}_{t_1}(v))$ . Then, this is a repairing sequence for  $C_k$ .

*Proof.* 1. If  $C_k$  is existentially bound, the concurrent rule is given by  $\rho = C_{k-1} \xleftarrow{\text{id}} C_{k-1} \xrightarrow{a_k} C_k$ . Let  $C_j$  be a universally bound graph such that  $C_k$  has no conflict with  $C_j$  and  $\rho$  is not a basic consistency maintaining rule w.r.t.  $\forall(C_j, \text{true})$ . With Lemma 6.4 follows immediately that  $C_k$  has a conflict with  $C_j$ , this is a contradiction.

2. If  $C_k$  is universally bound, the concurrent rule is given by  $\rho = C_k \xleftarrow{a_{k-1}^T} C \xrightarrow{\text{id}} C$ . Then,  $\rho$  is a basic consistency maintaining rule w.r.t.  $\forall(C_j, \text{true})$  for all universally bound graphs since  $\rho$  does not insert any elements.

□

**Definition 6.6 (repairing rule set).** Let a set of rules  $\mathcal{R}$  and a circular conflict free constraint  $c$  in UANF be given. Then,  $\mathcal{R}$  is called a repairing rule set of  $c$  if there does exist a repairing sequence for each existentially bound graph of  $c$  and, if  $\text{nl}(c)$  is odd, i.e.  $c$  ends with a condition of the form  $\forall(C_{\text{nl}(c)}, \text{false})$ ,  $\mathcal{R}$  contains a repairing sequence for  $C_{\text{nl}(c)}$ .

Note that there cannot exist a repairing sequence for universally bound graphs  $C_k$  such that  $C_k \setminus C_{k-1}$  does not contain any edges. Therefore, there does not exist a repairing set for all constraints of the form  $\forall(C_1, \text{false})$  such that  $E_{C_1} = \emptyset$ .

**Theorem 6.2.** Let a circular conflict free constraint  $c$  in UANF and a repairing set  $\mathcal{R}$  of  $c$  be given. Then, for each graph  $G$  with  $G \not\models c$ , there does exist a sequence of transformations

$$G = G_0 \Rightarrow_{\rho_1, m_1} \dots \Rightarrow_{\rho_n, m_n} G_n$$

with  $\rho_i \in \mathcal{R}$  such that  $G_n \models c$ .

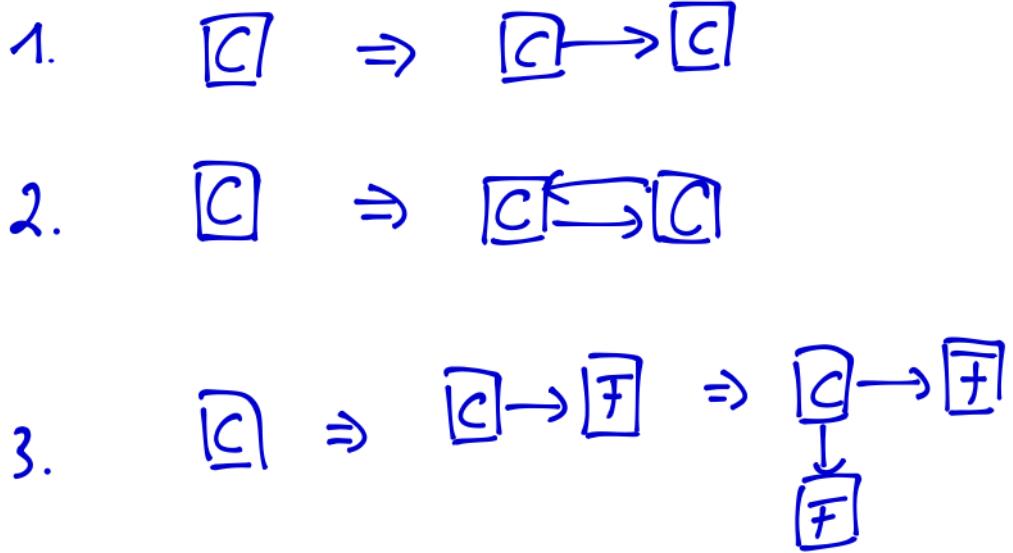


Figure 19: Repairing sequences for  $c_1$  and  $c_4$ .

We postpone the proof of this Theorem, since it will follow immediately from the termination of our repairing process.

**Example 6.2.** Consider constraints  $c_1$ ,  $c_4$  and the sequences shown in Figure 19. The first sequence is not a repairing sequence for the existentially bound graph of  $c_4$  since  $G_1 \not\models_1 c_4$  and therefore, a rule set containing only this rule is not a repairing set w.r.t.  $c_4$ . The second sequence is a repairing sequence for the existentially bound graph of  $c_4$ , since the last graph satisfies  $c_4$  and there does not exist an the existentially bound graph has a conflict with the universally bound graph. Therefore, the condition for the concurrent rule is also satisfied and a rule set containing this rule is a repairing set w.r.t.  $c_4$ .

The third sequence is a repairing sequence for  $c_1$  since the last graph satisfies  $c_1$  and the sequence satisfies the criteria given in Theorem 6.1. Note, that this sequence contains of two applications of the same rule. A rule set containing this rule is a repairing set w.r.t.  $c_1$ .

### 6.3 Construction of Repairing Sets

In the following we want to introduce a construction of these repairing sets for a given circular conflict free constraint in UANF.

**Definition 6.7 (constructed repairing set).** Let a circular conflict free constraint  $c$  in UANF with  $\text{nl}(c) > 1$  or  $c = \forall(C_1, \text{false})$  and  $E_{C_1} \neq \emptyset$  be given. The constructed repairing set,  $\mathcal{R}$ , for  $c$  is constructed in the following way. For each universally bound

graph  $C_k$  of  $c$  and every pair of graphs  $C, C' \in \text{IG}(C_{k-1}, C_k)$  with  $C$  being a sub-graph of  $C'$  there does exist a rule

$$C' \xleftarrow{a_{k-1}^r} C \xrightarrow{\text{id}} C$$

in  $\mathcal{R}$ . For each existentially bound graph  $C_k$  of  $c$  and every pair of graphs  $C, C' \in \text{IG}(C_{k-1}, C_k)$  with  $C$  being a sub-graph of  $C'$  there does exist a rule

$$C \xleftarrow{\text{id}} C \xrightarrow{a_{k-1}^r} C'$$

in  $\mathcal{R}$ .

Note that for each element of existentially bound graphs,  $\mathcal{R}$  contains a rule that only inserts this element, and for each element of universally bound graphs  $\mathcal{R}$  contains a rule that only removes this element.

**Theorem 6.3.** Let a circular conflict free constraint  $c$  in UANF with  $\text{nl}(c) > 1$  or  $c = \forall(C_1, \text{false})$  and  $E_{C_0} \neq \emptyset$  be given. The set of rules constructed by Definition 6.7 is a repairing set w.r.t.  $c$ .

*Proof.* Let  $\mathcal{R}$  be the constructed rule set. If  $\text{nl}(c) = 1$  and  $E_{C_1} \neq \emptyset$ , there does exist a rule  $\rho = C_1 \xleftarrow{a_0^r} C' \xrightarrow{\text{id}} C$  with  $C \in \text{IG}(\emptyset, C_1)$  such that  $V_C = V_{C_0}$ . Then,

$$C_1 \xrightarrow{\rho, \text{id}} C$$

is a repairing sequence for  $C_1$  and  $\mathcal{R}$  is a repairing rule set for  $c$ .

If  $\text{nl}(c) \geq 2$ , consider an existentially bound graph  $C_k$  and the transformation sequence

$$C_{k-1} \xrightarrow{\rho_1, m} C_k$$

with  $\rho_1 = C_{k-1} \xleftarrow{\text{id}} C_{k-1} \xrightarrow{a_{k-1}^r} C_k$ . If  $C_k$  has no conflict with an universally bound graph  $C_j$  with  $j < k$ ,  $C_k \models_k c$  and this is a repairing sequence. If,  $C_k \not\models_k c$ , an occurrence  $p$  of an universally bound graph  $C_j$  with  $j < k$  has been inserted. Then, the repairing sequence can be extended by

$$C_k \xrightarrow{\rho_2, m} G_1$$

with  $\rho_2 = C \xleftarrow{\text{id}} C \xrightarrow{a_j^r} C_{j+1}$  such that  $C \in \text{IG}(C_j, C_{j+1})$  and  $V_C = V_{C_{k+1}}$ . with  $p = m \circ a_{j-1}^r$ . Since  $j < k$ , such a match and transformation must exist. If this inserts new occurrences of universally bound graphs  $C_{j'}$  with  $j' < k$ , we repeat this process.

□

**Example 6.3.**

## 6.4 Rule-based Graph Repair for one Constraint

In the following, we present our graph repair process for one circular conflict free constraint in UANF. The process is shown in Algorithm 1 and proceeds in the following way. The algorithm starts by finding all occurrences of  $C_{k_{\max}+2}$  that do not satisfy  $\text{cut}_0(\text{sub}_{k_{\max}+2}(c))$  (line 2). This condition is equal to false if  $k_{\max}+2 = \text{nl}(c) - 2$  and equal to  $\exists(C_{k_{\max}+2}, \text{true})$  otherwise. If  $P$  is empty, it must follow that  $G \models_{k_{\max}+2} c$  and therefore, we will apply repairing sequences at these occurrences. It might be enough to only repair some of these occurrences. Since it is unknown which of these are able to increase the satisfaction at layer, we choose one uniformly at random (line 3). For example, for existentially bound constraints  $d$ , that means, its equivalent constraint in UANF is equal to  $\forall(\emptyset, d)$ , there might exist occurrences of  $C_{k_{\max}+2}$ , whose repair will not lead to an increase of the satisfaction at layer.

There are two possible ways to repair the chosen occurrence, either by destroying it, or by inserting elements such that the occurrence satisfies  $\text{cut}_0(\text{sub}_{k_{\max}+2}(c))$ . The Algorithm chooses one of these options (line 4) and applies the corresponding repairing sequence (line 5–11). Note that a repairing sequence for  $C_{k_{\max}+2}$  might not exist, since this graph is universally bound. If this is the case we use the repairing sequence for  $C_{k_{\max}+3}$ . This must exist since  $C_{k_{\max}+3}$  is existentially bound.

If the repairing sequence for  $C_{k_{\max}+2}$  has been applied, occurrences of existentially bound graphs might have been destroyed. Note that this can only be occurrences of graphs  $C_i$  such that  $C_{k_{\max}+2}$  has a conflict with  $C_i$ . This might lead to a decrease of satisfaction of layer. Therefore, the algorithm finds all of these destroyed occurrences, in particular it finds all occurrences  $p$  of universally bound graphs  $C_i$  such that an occurrence  $q$  of  $C_{i+1}$  with  $p = q \circ a_j$  has been removed (line 7).

If the repairing sequence for  $C_{k_{\max}+3}$  has been applied, occurrences of universally bound graphs might have been inserted. Again, this can only be occurrences of graphs  $C_i$  such that  $C_{k_{\max}+2}$  has a conflict with  $C_i$  and this might lead to a decrease of satisfaction at layer. Again, the algorithm finds all inserted occurrences of universally bound graphs (line 10). If the satisfaction at layer has not been decreased, the algorithm chooses the next occurrence in  $P$ .

Otherwise, the satisfaction at layer needs to be restored. For this, the occurrences contained in  $M$  need to be repaired. The repair of these occurrences might again lead to an insertion of existentially bound graphs or the removal of universally bound ones. These occurrences are added to  $H$  and this process repeats until the satisfaction at layer is restored, i.e.  $H \models_{k_{\max}} c$  (line 12 – 25). This whole process will repeat until a graph satisfying  $c$  is derived.

Form this, it becomes clear, why  $c$  has to be circular conflict free. For a constraint with circular conflicts, during the restore phase, new occurrence of  $C_{k_{\max}+2}$  can be inserted an occurrence of  $C_{k_{\max}+3}$  can be removed. In particular cases, this could lead to an infinite loop and therefore, there is no guarantee that this algorithm will terminate. For example, consider constraint  $c_3$  given in Figure 16. The set of rules that are used for the transformations  $t_1$  and  $t_2$  in Figure 16 forms a repairing set. During a repair process using Algorithm 1 with the starting graph being the first graph of  $t_1$  it might be

possible that Algorithm 1 runs into an infinite loop, by alternately applying  $t_1$  and  $t_2$ .

A optimization of the repair algorithm in terms of the number of inserted or deleted elements can be performed by using partial repairing sequences if possible. For example, consider the repairing sequence

$$C_k \Rightarrow C_1 \Rightarrow \dots \Rightarrow C_{k+1}$$

with  $C_1 \in \text{IG}(C_k, C_{k+1})$ . For an occurrence  $p$  of  $C_k$  that already satisfies  $\exists(C_1, \text{true})$  it might be sufficient to only apply the sequence

$$C_1 \Rightarrow \dots \Rightarrow C_{k+1}$$

at  $p$ . But, after this, it needs to be checked that no occurrences of existentially bound graphs have been destroyed and that no occurrence of universally bound graphs  $C_i$  such that  $C_k$  has no conflict with  $C_i$  have been inserted. If this is the case, the transformations need to be undone and another (partial) repairing sequence needs to be used. Even if this would lead to an optimization in terms of the number of inserted and deleted elements, due to the reversion of transformations, this will lead to an increase of runtime.

For each circular conflict free constraint, Algorithm 1 will always terminate as shown by the following Theorem.

**Theorem 6.4.** *Let a graph  $G$ , a circular conflict free condition in UANF and a repairing set  $\mathcal{R}$  be given. Then, Algorithm 1 with input  $G, c$  and  $\mathcal{R}$  terminates and returns a graph  $H$  with  $H \models c$ .*

*Proof.* If Algorithm 1 terminates, it returns a graph satisfying  $c$ . Therefore, it is sufficient to show that Algorithm 1 will terminate. Since  $G$  is finite, the set  $P$  must also be finite. If a repairing sequence has been applied, the set  $M$  only contains occurrences of graphs  $C_j$  such that  $C_{k_{\max}+2}$  has a (transitive) conflict with  $C_j$  since the repairing sequence is not able to destroy or insert occurrences of  $C_i$  such that  $C_{k_{\max}+2}$  has no (transitive) conflict with  $C_i$ . Because  $G$  is finite,  $|M|$  must also be finite.

If the derived graph does not satisfy  $\text{cut}_{k_{\max}(c,G)}(c)$ , we need to restore the satisfaction at layer. Because the satisfaction at layer only decreases if an occurrence of an existentially bound graph has been destroyed or an occurrence of universally bound graphs has been inserted and  $M$  does contain all these occurrences, we only need to consider the occurrences contained in  $M$ . The application of repairing sequences at occurrences  $p : C_j \hookrightarrow H \in M$  could again lead to an insertion of universally bound or an removal of existentially bound graphs. The set  $M'$  contains all these occurrences and again, this are only occurrences of  $C_i$  such that  $C_j$  has a (transitive) conflict with  $C_i$ . Since  $c$  is circular conflict free,  $M'$  cannot contain any occurrences of  $C_{k_{\max}+2}$ , otherwise,  $C_j$  would have a (transitive) conflict with  $C_{k_{\max}+2}$  and therefore  $C_{k_{\max}+2}$  has circular conflict. Therefore no occurrences of  $C_{k_{\max}+3}$  will be destroyed and no occurrences of  $C_{k_{\max}+2}$  will be inserted. Additionally,  $C_{k_{\max}+2}$  has a (transitive) conflict with  $C_i$  and the repair of any  $p \in M'$  will not lead to an insertion of an occurrence of  $C_{k_{\max}+2}$  or the removal of an occurrence of  $C_{k_{\max}+3}$ .

---

**Algorithm 1:** Repair for one circular conflict free constraint

---

**Data:** A graph  $G$ , a circular conflict free constraint  $c$  in UANF and a repairing set  $\mathcal{R}$  for  $c$ .

**Result:** A graph  $H$  with  $H \models c$ .

```

1 while  $G \not\models c$  do
2    $P \leftarrow \{q : C_{k_{\max}+2} \hookrightarrow H \mid q \not\models \text{cut}_0(\text{sub}_{k_{\max}+2}(c))\};$ 
3   Choose  $p \in P$  uniformly at random ;
4   Choose  $r \in \{0, 1\}$  uniformly at random;
5   if  $r = 0$  and  $\mathcal{R}$  contains a repairing sequence for  $C_{k_{\max}+2}$  then
6     Apply the repairing sequence for  $C_{k_{\max}+2}$  at match  $p$  and let  $H$  be the
      derived graph ;
7      $M \leftarrow \{q : C_j \hookrightarrow H \mid j \text{ odd and } \neg \exists q' : C_j \hookrightarrow G(\text{tr} \circ q' = q)\};$ 
8   else
9     Apply the repairing sequence for  $C_{k_{\max}+3}$  at match  $p$  and let  $H$  be the
      derived graph ;
10     $M \leftarrow \{q : C_j \hookrightarrow H \mid j \text{ odd and } \exists q' : C_{j+1} \hookrightarrow G(q =
      q' \circ a_j \wedge \text{tr} \circ q' \text{ is not total})\};$ 
11  end
12  while  $H \not\models_{k_{\max}(c,G)} c$  do
13    Choose  $p : C_j \hookrightarrow H \in M$  uniformly at random ;
14    Choose  $r \in \{0, 1\}$  uniformly at random ;
15    if  $r = 0$  and  $\mathcal{R}$  contains a repairing sequence for  $C_j$  then
16      Apply the repairing sequence for  $C_j$  at match  $p$  and let  $H'$  be the
        derived graph ;
17       $M' \leftarrow \{q : C_i \hookrightarrow H' \mid i \text{ odd and } \neg \exists q' : C_i \hookrightarrow H(\text{tr} \circ q' = q)\} ;$ 
18    else
19      Apply the repairing sequence for  $C_{j+1}$  at match  $p$  and let  $H'$  be the
        derived graph ;
20       $M' \leftarrow \{q : C_i \hookrightarrow H' \mid i \text{ odd and } \exists q' : C_{i+1} \hookrightarrow G(q =
        q' \circ a_j \wedge \text{tr} \circ q' \text{ is not total})\};$ 
21    end
22     $M \leftarrow (M \setminus \{p\}) \cup M' ;$ 
23     $H \leftarrow H';$ 
24  end
25   $G \leftarrow H;$ 
26 end
27 return  $G;$ 

```

---

Since  $c$  is circular conflict free, there must exist graphs  $C_i$ , such that  $C_i$  has no conflict with any other graph  $C_{i'}$  and  $C_{k_{\max}+2}$  has a (transitive) conflict with  $C_i$ . Therefore, the application of repairing sequences at occurrences of these graphs will not lead to the insertion or removal of any universally or existentially bound graph, respectively. Since  $c$  is finite, the number of graphs  $C_i$  such that  $C_{k_{\max}+2}$  has a (transitive) conflict with  $C_i$  is finite. Since  $|M'|$  is also finite, after a finite number of repairing sequence applications,  $M'$  only contains occurrences of graphs that do not have any conflicts. After a repairing sequence has been applied at all those occurrences,  $M'$  is empty and  $H \models_{k_{\max}(c,G)} c$ , since all occurrence  $p$  of  $C_j$ , that either have been inserted or an occurrence  $q$  of  $C_{j+1}$  with  $p = a_j \circ q$  has been removed, satisfy  $\exists(C_{j+1}, \text{true})$ . Additionally holds that  $nvc_{k_{\max}+1}(H) < nvc_{k_{\max}+1}(G)$ .

Therefore, after a finite number of iterations, the satisfaction at layer has been increased by at least 1. It follows that after a finite number of iterations  $G \models c$ . Then, Algorithm 1 terminates and returns  $G$ .  $\square$

**Example 6.4.** Consider constraint  $c = \forall(C_2^2, \exists(C_2^1, \text{true}))$  composed of the graphs shown in Figure 5. This constraint is circular conflict free and a repairing set for  $c$  is given in Figure 20. There does exist a repairing sequence for  $C_2^2$  via the rule `remove` and a repairing sequence for  $C_2^1$  via the rule `insert`. Using the rule set `{remove, insert}`, Algorithm 1 could return one of the graphs  $G_1, G_2$  or  $G_3$  given in Figure 20, depending on the repairing sequences that have been used.

## 6.5 Rule-based Graph Repair for multiple Constraints

Now, we will introduce our rule-based repair approach for a set of constraints in UANF.

**Definition 6.8 (satisfaction of constraint sets).** Let a set of constraints  $\mathcal{C}$  be given. A graph  $G$  satisfies  $\mathcal{C}$ , denoted by  $G \models \mathcal{C}$ , if  $G \models \bigwedge_{c \in \mathcal{C}} c$ . The set  $\mathcal{C}$  is called satisfiable if a graph  $G$  with  $G \models \mathcal{C}$  exists.

To guarantee that a set of constraints can be repaired by a set of rules, we need to extend the notion of repairing sets such that a set of rules is called a *repairing set* for a set of constraints if it is a repairing set for each constraint in the constraint set.

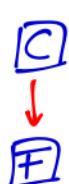
**Definition 6.9 (repairing set for a set of constraints).** Let a set of constraints  $\mathcal{C}$  and a set of rules  $\mathcal{R}$  be given. Then,  $\mathcal{R}$  is called a repairing set for  $\mathcal{C}$  if  $\mathcal{R}$  is a repairing set for all constraints  $c \in \mathcal{C}$ .

We also extend the notion of conflicts to *conflicts between constraints*. Intuitively, a constraint  $c$  has a conflict with another constraint  $c'$  if one of its graphs has a conflict with a graph of  $c'$ .

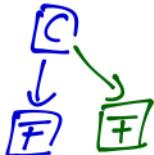
**Definition 6.10 (conflict between constraints).** Let constraints  $c, c'$  in UANF and a set of rules  $\mathcal{R}$  be given. Then,  $c$  has a conflict with  $c'$  if a repairing sequence

$$C_k = G_0 \xrightarrow{\rho_1, m_1} \dots \xrightarrow{\rho_n, m_n} G_n$$

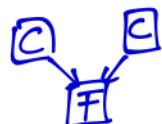
remove :



insert :



$G$  :



$G_1$  :



$G_2$  :



$G_3$  :

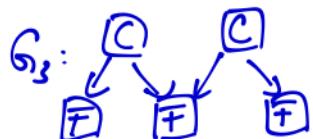


Figure 20: Possible outputs of the repairing process for  $G$  and  $\forall(C_2^2, \exists(C_2^1, \text{true}))$  using the rule set  $\{\text{remove}, \text{insert}\}$ .

---

**Algorithm 2:** Repair for a circular constraint-conflict free set of constraints

---

**Data:** A graph  $G$ , circular constraint-conflict free set of constraints  $\mathcal{C}$  and a repairing set  $\mathcal{R}$  for  $\mathcal{C}$ .

**Result:** A graph  $H$  with  $H \models \bigwedge_{c \in \mathcal{C}} c$ .

```
1  $(c_1, \dots, c_n) \leftarrow$  topological ordering of  $\mathcal{C}$  ;
2 for  $i \leftarrow 1$  to  $n$  do
3   | Repair  $c_i$  in  $G$  with Algorithm 1, let  $H$  be the retuned graph ;
4   |  $G \leftarrow H$  ;
5 end
6 return  $G$ ;
```

---

for a graph  $C_k$  of  $c$  exists such that the concurrent rule of this sequence is not basic consistency maintaining rule w.r.t.  $\forall(C_j, \text{false})$  or  $\exists(C_j, \text{true})$  for any universally or existentially bound graph  $C_j$  of  $c'$ .

The following Lemma is a useful statement for the correctness proof of our repair approach. It states that the application of a repairing sequence of a constraint  $c$  cannot destroy the satisfaction of  $c'$  if  $c$  has no conflict with  $c'$ .

**Lemma 6.11.** *Let two constraints  $c$  and  $c'$  in UANF, such that  $c$  has no conflict with  $c'$  w.r.t. to a set of rules  $\rho$ , be given. Then the concurrent rule  $\rho$  of each repairing sequence for  $c$  is a  $c'$ -preserving rule.*

*Proof.* Assume that  $\rho$  is not a  $c'$ -preserving rule. Then, there does exist a transformation  $t : G \Rightarrow_{\rho, m} H$  such that  $G \models c'$  and  $H \not\models c'$ . Therefore, either an universally bound graph of  $c'$  has been inserted or an existentially bound graph of  $c'$  has been removed. Since  $\rho$  is a basic maintaining rule w.r.t.  $\forall(C_j, \text{false})$  for all universally bound graphs  $C_j$  of  $c'$  and a basic consistency maintaining rule w.r.t.  $\exists(C_j, \text{true})$  for all existentially bound graphs  $C_j$  of  $c'$ , this is a contradiction.  $\square$

The *conflicts graph* for a set of constraints and *circular conflicts* for it are defined in a similar manner as conflict graphs and circular conflicts for one constraint. A set of constraints is called *circular conflict free* if each of its constraints is circular conflict free and there does not exist a sequence  $c = c_0, \dots, c_n = c$  such that  $c_i$  has a conflict with  $c_{i+1}$  for all  $0 \leq i < n$ . In other words, the conflict graph of this set is acyclic.

**Definition 6.12 (conflict graphs, circular conflicts).** *Let a set of constraints  $\mathcal{C}$  in UANF be given. The conflict graph of  $\mathcal{C}$  is constructed in the following way. For each constraint  $c \in \mathcal{C}$  there does exist a node. If a conflict between  $c$  and  $c'$  exists, there does exist an edge  $e$  with  $\text{src}(e) = c$  and  $\text{tar}(e) = c'$ .*

*A constraint  $c$  has a transitive conflict with  $c'$  if the conflict graph of  $\mathcal{C}$  contains a path from  $c$  to  $c'$ . A constraint  $c$  has a circular conflict if  $c$  has a transitive conflict with itself. A set of constraints  $\mathcal{C}$  is called circular conflict free if each constraint in  $\mathcal{C}$  is circular conflict free and does not contain any circular conflicts.*

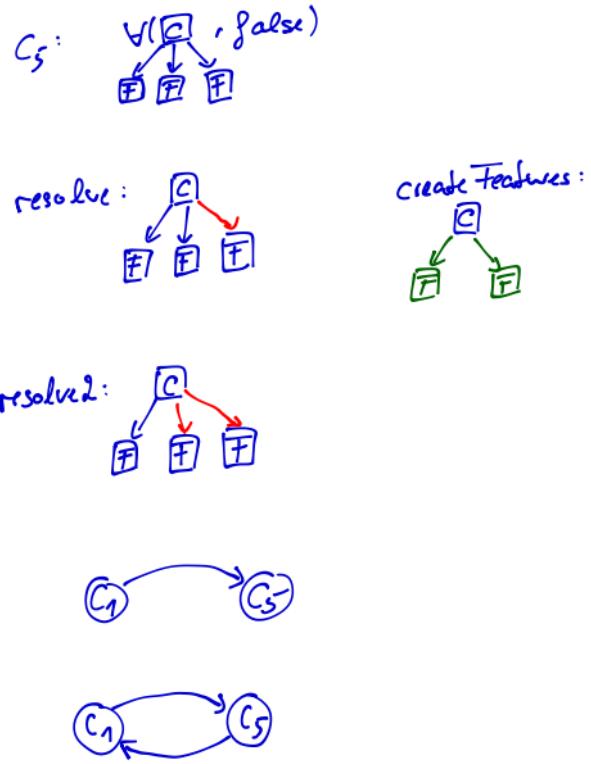


Figure 21: Constraints  $c_5$  and conflicts graphs of the constraint set  $\{c_1, c_5\}$  with the rule sets  $\{\text{resolve}, \text{createFeatures}\}$  and  $\{\text{resolve2}, \text{createFeatures}\}$ .

**Example 6.5.** Consider the rules `resolve`, `resolve2`, `createFeatures` and constraints  $c_1$  and  $c_5$  given in Figures 21 and 5. The constraint set  $\mathcal{C} = \{c_1, c_5\}$  is a multiplicity stating that “Each node of type Class is connected to exactly two nodes of type Feature”. With the rule set  $\mathcal{R}_1 = \{\text{resolve}, \text{createFeatures}\}$ , there is only one conflict in  $\mathcal{C}$ ;  $c_1$  has a conflict with  $c_5$  since an application of `createFeatures` could lead to an insertion of the universally bound graph of  $c_5$ . With the rule set  $\mathcal{R}_2 = \{\text{resolve2}, \text{createFeatures}\}$  there are two conflicts. Again, there is a conflict of  $c_1$  with  $c_5$ , but also a conflict of  $c_5$  with  $c_1$  since an application of `resolve` can destroy an occurrence of the existentially bound graph of  $c_1$ .

Therefore, our approach can repair with the rule set  $\mathcal{R}_1$  but not with  $\mathcal{R}_2$  because in this case,  $\mathcal{C}$  is not circular conflict free.

Our repair process makes use of the fact that the conflict graph of a circular conflict free set of constraints in UANF is acyclic. In particular, our approach uses the *topological ordering* of this conflict graph.

**Definition 6.13 (topological ordering of a graph).** Let a graph  $G$  be given. A sequence  $(v_1, \dots, v_n)$  of nodes of  $G$  is called a topological ordering of  $G$  no edge  $e \in E_G$

with  $\text{src}(e) = v_i$ ,  $\text{tar}(e) = v_j$  and  $i \geq j$  exists. The topological ordering of a circular conflict free set of constraints  $\mathcal{C}$  is the topological ordering of its conflicts graph.

It is a well known fact that each directed acyclic graph has a topological ordering and therefore each conflict graph of a circular conflict free set of constraints also has a topological ordering.

The repair process is given in Algorithm 2 and proceeds in the following way. First, the topological ordering of the constraint set is determined (line 1) Then, Algorithm 1 is used to repair each constraint of  $\mathcal{C}$  in order of the topological ordering (line 2 –4). Through this, it is ensured that the satisfaction of a constraint that has already been repaired will not be destroyed by the repair of another constraint.

**Theorem 6.5.** *Let a graph  $G$ , a satisfiable, circular conflict free set of constraints in UANF,  $\mathcal{C}$ , and a set of rules  $\mathcal{R}$  be given. Then, Algorithm 2 terminates and returns a graph  $H$  with  $H \models \mathcal{C}$ .*

*Proof.* Since  $\mathcal{C}$  is finite and each  $c \in \mathcal{C}$  is circular conflict free, Algorithm 1 terminates for each  $c \in \mathcal{C}$ . Therefore, Algorithm 2 will also terminate.

It remains to show that the returned graph satisfies  $\mathcal{C}$ . Let  $(c_1, \dots, c_n)$  be a topological ordering of  $\mathcal{C}$ . Then no constraint  $c_j$  with  $j \neq 1$  has a conflict with  $c_1$  and with Lemma 6.11 follows that the concurrent rule of each repairing sequence for each  $c_i$  with  $2 \leq i \leq n$  is a  $c_1$ -preserving rule. In general, the concurrent rule of each repairing sequence for  $c_j$  is a  $c_i$ -preserving rule if  $i < j$ . Note that in Algorithm 2 each repairing sequence can be replaced by its concurrent rule. After one iteration,  $G \models c_1$ . Assume that after  $m$  iterations it holds that  $G \models c_i$  for all  $1 \leq i \leq m$ . In iteration  $m + 1$ ,  $c_{m+1}$  will be repaired by Algorithm 2. Since each concurrent rule of each repairing sequence of  $c_{m+1}$  is a  $c_i$ -preserving rule for all  $1 \leq i \leq m$ , the application of repairing sequence can be replaced by its concurrent rule and Algorithm 2 only applies repairing sequences, it follows that  $H \models c_i$  for all  $1 \leq i \leq m + 1$ . Therefore, after  $n$  iterations,  $H \models c_i$  for all  $1 \leq i \leq n$ . It follows immediately that the returned graph  $G$  satisfies  $\mathcal{C}$ .  $\square$

## 7 Conclusion

## References

- [1] H. Ehrig, K. Ehrig, U. Prange, and G. Taentzer. [Fundamentals of algebraic graph transformation](#). *Monographs in theoretical computer science. An EATCS series*. Springer, 2006.
- [2] H. Ehrig, A. Habel, and L. Lambers. Parallelism and concurrency theorems for rules with nested application conditions. *Electronic Communications of the EASST*, 26, 2010.
- [3] A. Habel and K.-H. Pennemann. Nested constraints and application conditions for high-level structures. In *Formal methods in software and systems modeling*, pages 293–308. Springer, 2005.
- [4] A. Habel and K.-H. Pennemann. [Correctness of high-level transformation systems relative to nested conditions](#). *Mathematical Structures in Computer Science*, 19(2):245–296, 2009.
- [5] J. Kosiol, D. Strüber, G. Taentzer, and S. Zschaler. [Sustaining and improving graduated graph consistency: A static analysis of graph transformations](#). *Science of Computer Programming*, 214:102729, 2022.
- [6] D. Plump. Confluence of graph transformation revisited. In *Processes, Terms and Cycles: Steps on the Road to Infinity*, pages 280–308. Springer, 2005.
- [7] C. Sandmann and A. Habel. [Rule-based graph repair](#). *arXiv preprint arXiv:1912.09610*, 2019.