

# Documentation of MRs for Testing Constraint Checking Implementations

June 8, 2023

## 1 Symbol Conventions

For ease of narrative, in the following sections, we use  $D_0$  and  $S_0$  to denote data under checking and a constraint for checking in source test input,  $Rt_0$  and  $Rl_0$  to denote truth value and link set in the source test output. Similarly, we use  $D_1$ ,  $S_1$ ,  $Rt_1$  and  $Rl_1$  to denote corresponding variables in the follow-up test.

## 2 MR-Data

There are two different MRs in this part, and they only transform the data under checking and keep the constraint unchanged.

### 2.1 Link

Before introducing the MR-Data, let's talk more about links.

Links explain how a constraint is violated or satisfied. A link is composed of two parts: *type* and *bindings*. Type of a link is either *vio* or *sat*, indicating that this link is a violated link or a satisfied link respectively. Bindings of a link is a set of variable assignments, which can help explain how a constraint is violated or satisfied. For example,  $l_1 :< vio, \{x = 1, y = 2\} >$  and  $l_2 :< vio, \{x = 2, y = 2\} >$  are two different links. The type of  $l_1$  is *vio* and the bindings of  $l_1$  is  $\{x = 1, y = 2\}$ .

Then, we will define "cover" relation between two link sets. Let  $l_1$  and  $l_2$  be two links. We use  $type(l_1)$  to denote the type of  $l_1$  and  $bindings(l_1)$  to denote the bindings of  $l_1$ . If  $l_1$  and  $l_2$  have the same type and all the bindings of  $l_1$  are also contained by  $l_2$ , we say  $l_2$  "covers"  $l_1$  and denote it as  $l_1 \preceq l_2$ . This is the definition of the cover relation between two links:

$$l_1 \preceq l_2 \Leftrightarrow type(l_1) == type(l_2) \wedge bindings(l_1) \subseteq bindings(l_2) \quad (1)$$

Further more, let  $ls_1 = \{l_{11}, l_{12}, \dots, l_{1n}\}$  and  $ls_2 = \{l_{21}, l_{22}, \dots, l_{2m}\}$  be two link sets. If for every link  $l_i$  in set  $ls_1$ , there exists an link  $l_j$  in set  $ls_2$  that satisfies  $l_i \preceq l_j$ , then we say  $ls_2$  "covers"  $ls_1$  and denote it as  $ls_1 \preceq ls_2$ . This is the definition of the cover relation between two link sets:

$$ls_1 \preceq ls_2 \Leftrightarrow \forall l_i \in ls_1 (\exists l_j \in ls_2 (l_i \preceq l_j)) \quad (2)$$

It is easy to see that  $\preceq$  is partial order relation and the inclusion relation of link sets ( $\subseteq$ ) is a special case of the cover relation ( $\preceq$ ), i.e.  $ls_1 \subseteq ls_2 \Rightarrow ls_1 \preceq ls_2$ .

Besides that, we use  $var(l)$  and  $var(ls)$  to denote the set of the variables appeared in a link or a link set. And we use  $val(ls, x)$  to denote all the value assigned to the variable  $x$ .

To illustrate these concepts more clearly, let's look at the example in Fig. 1. In the example,  $ls_1 \preceq ls_2$ , because the only link  $l_1$  in  $ls_1$  is also in  $ls_2$ , i.e.  $ls_1 \subseteq ls_2$ . On the other hand,  $l_1 \preceq l_3$  and  $l_2 \preceq l_4$ , so  $ls_2 \preceq ls_3$ .

For more details of the rules for calculating links, please refer to the existing work [8].

Let:

$$\begin{aligned}
l_1 &= \langle vio, \{x = 1\} \rangle \\
l_2 &= \langle vio, \{y = 4\} \rangle \\
l_3 &= \langle vio, \{x = 1, y = 3\} \rangle \\
l_4 &= \langle vio, \{x = 2, y = 4\} \rangle \\
ls_1 &= \{l_1\} \\
ls_2 &= \{l_1, l_2\} \\
ls_3 &= \{l_3, l_4\}
\end{aligned}$$

then:

$$\begin{aligned}
ls_1 &\leqslant ls_2 \leqslant ls_3 \\
var(ls_3) &= \{x, y\} \\
val(ls_3, x) &= \{1, 2\}
\end{aligned}$$

Figure 1: An example of cover relation among link sets

## 2.2 Impact of data changes

Then, we talk about the impact of data changes.

According to the existing work [7], impact of data changes can be divided into two categories: inconsistency-incurring and inconsistency-hidden(also named as inc+/inc-). The inconsistency-incurring changes leads to more possibilities for incurring violation and inconsistency-hidden changes has the opposite effect. These two kinds of impacts of data changes depend on the constraint and can be derived from the constraint in a static way.

For example, constraint  $S : \forall x \in X(not(\exists y \in Y(same(X, Y))))$  has the inconsistency-incurring changes set  $\{\langle +, X \rangle, \langle +, Y \rangle\}$ , which means if we add any element to X or Y, constraint  $s$  would tend to generate more violations.

For more details of the derivation rules for two impact of data changes, please refer to the existing work [7].

## 2.3 MR-Data 1

Now we can elaborate on our MR-Data 1 as follow:

**MR-Data 1:** If  $Rt_0 == F$ , we apply any change with an inconsistency-incurring impact on  $S_0$  to  $D_0$  to obtain  $D_1$ . Then, in the follow-up execution,  $Rt_1$  should still be  $F$  and  $Rl_1$  should satisfy the condition:  $Rl_0 \preceq Rl_1$ .

MR-Data 1 indicates that if  $Rt_0$  is  $F$ , an inconsistency-incurring change applied to  $D_0$  can only lead to more violations of the whole constraint and the original inconsistencies still exists and would not be removed in the follow-up execution. That's why  $Rt_1$  is still  $F$  and  $Rl_1$  covers the  $Rl_0$ .

Fig. 2 has shown an example of MR-Data 1. In the example, the source output  $Rt_0$  is  $F$ . We apply an inconsistency-incurring impact change  $\langle +, C_1 \rangle$  to  $D_0$  to obtain  $D_1$  for follow-up execution. Then,  $Rt_1$  is still  $F$  which is the same as the  $Rt_0$ , and  $Rl_1$  covers the  $Rl_0$ , i.e.  $Rl_0 \preceq Rl_1$ . That is, the output of the follow-up execution satisfies MR-Data 1.

## 2.4 MR-Data 2

MR-Data 2 is the other side of MR-Data 1:

**MR-Data 2:** If  $Rt_0 == T$ , we apply any change with an inconsistency-hidden impact on  $S_0$  to  $D_0$  to obtain  $D_1$ . Then, in the follow-up execution,  $Rt_1$  should still be  $T$  and  $Rl_1$  should satisfy

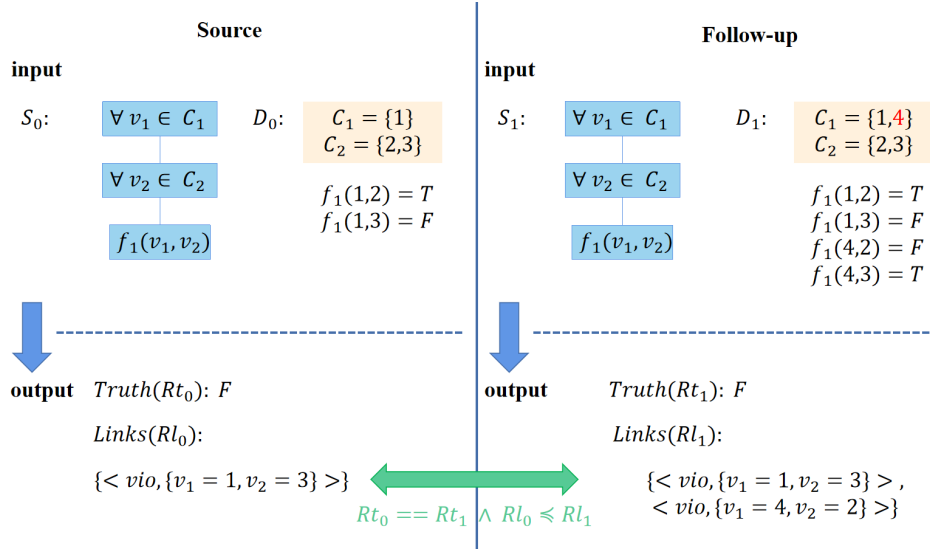


Figure 2: An example of MR-Data 1

the condition:  $Rl_0 \preceq Rl_1$ .

Similar to MR-Data1, MR-Data 2 indicates that if  $Rt_0$  is  $T$ , an inconsistency-hidden change applied to  $D_0$  can only lead to fewer violations, in other words, more satisfaction. That's why  $Rt_1$  is still  $T$  and  $Rl_1$  covers the  $Rl_0$ .

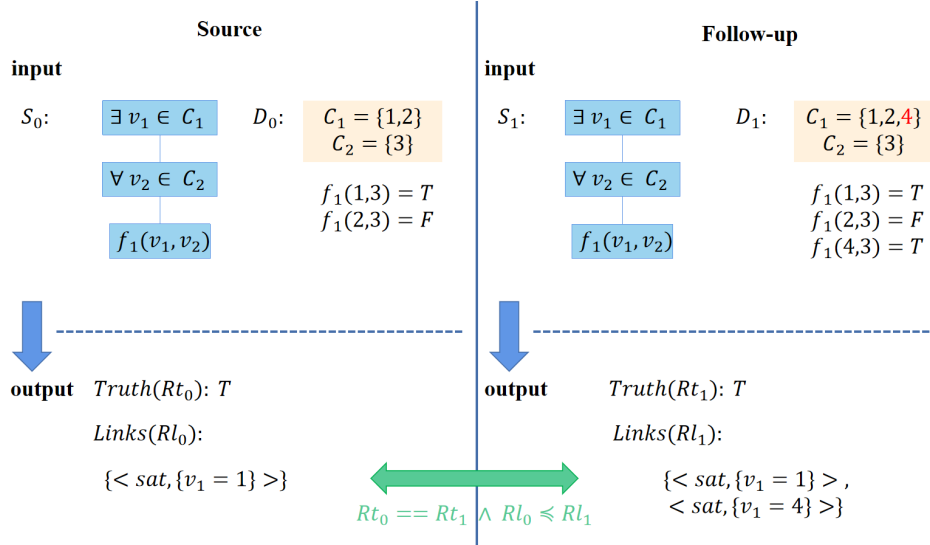


Figure 3: An example of MR-Data 2

Fig. 3 has shown an example of MR-Data 2. In the example, the source output  $Rt_0$  is  $T$ . We apply an inconsistency-hidden impact change  $\langle +, C_1 \rangle$  to  $D_0$  to obtain  $D_1$  for follow-up execution. Then,  $Rt_1$  is still  $T$  which is the same as the  $Rt_0$ , and  $Rl_1$  covers the  $Rl_0$ , i.e.  $Rl_0 \preceq Rl_1$ . That is, the output of the follow-up execution satisfies MR-Data 2.

### 3 MR-Cons

There are nine different MRs in this part, and they only transform the constraint for checking and keep the data unchanged.

#### 3.1 MR-Cons 1

**MR-Cons 1:** If the root of the constraint  $S_0$  is  $\forall v \in C$ , we transform it to  $\exists v \in C$  to obtain  $S_1$ . Then, in the follow-up execution,  $Rt_1$  and  $Rl_1$  should satisfy different conditions according to the following 3 cases:

- Case 1: If  $Rt_0$  is  $T$ ,  $Rt_1$  should be  $T$  and  $val(Rl_1, v) == C$ .
- Case 2: If  $Rt_0$  is  $F$  and  $val(Rt_0, v) == C$ ,  $Rt_1$  should be  $F$  and  $Rt_1$  should be empty.
- Case 3: If  $Rt_0$  is  $F$  and  $val(Rt_0, v) \subset C$ ,  $Rt_1$  should be  $T$  and  $val(Rt_1, v) == C - val(Rt_0, v)$ .

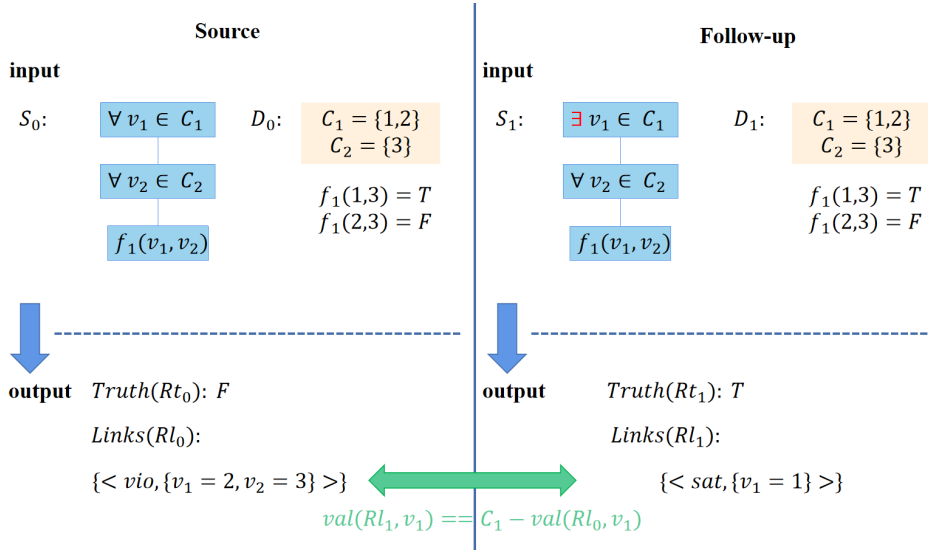


Figure 4: An example of MR-Cons 1

MR-Cons 1 exploits symmetrical relationship between  $\forall$  and  $\exists$ . The root node is calculated with many branches that take truth value  $T$  or  $F$  depending on different values of the variable  $v$ . The branches with the truth value  $F$  will become violated links for  $\forall$ , and the branches with the truth value  $T$  will become satisfied links for  $\exists$ . The values of the variable  $v$  in the two cases are the entire set of the matched data pattern  $C$ .

Fig. 4 has shown an example of MR-Cons 1. In the example, the root of the source constraint  $S_0$  is  $\forall$ . We transform it to  $\exists$  to obtain  $S_1$  for follow-up execution. Since  $Rt_0$  is  $F$  and  $val(Rt_0, v_1) = \{2\} \subset C_1 = \{1, 2\}$ , this is the case 3 of MR-Cons 1. Therefore,  $Rt_1$  should be  $T$  and  $val(Rt_1, v_1)$  should be  $C_1 - val(Rt_0, v_1) = \{1\}$ . And that's exactly what happened in the follow-up output, which indicates that it satisfies MR-Cons 1.

#### 3.2 MR-Cons 2

MR-Cons 2 is the other side of MR-Cons 1 and can be elaborated on as follow:

**MR-Cons 2:** If the root of the constraint  $S_0$  is  $\exists v \in C$ , we transform it to  $\forall v \in C$  to obtain  $S_1$ . Then, in the follow-up execution,  $Rt_1$  and  $Rl_1$  should satisfy different conditions according to the following 3 cases:

- Case 1: If  $Rt_0$  is  $F$ ,  $Rt_1$  should be  $F$  and  $val(Rl_1, v) == C$ .
- Case 2: If  $Rt_0$  is  $T$  and  $val(Rt_0, v) == C$ ,  $Rt_1$  should be  $T$  and  $Rt_1$  should be empty.
- Case 3: If  $Rt_0$  is  $T$  and  $val(Rt_0, v) \subset C$ ,  $Rt_1$  should be  $F$  and  $val(Rt_1, v) == C - val(Rt_0, v)$ .

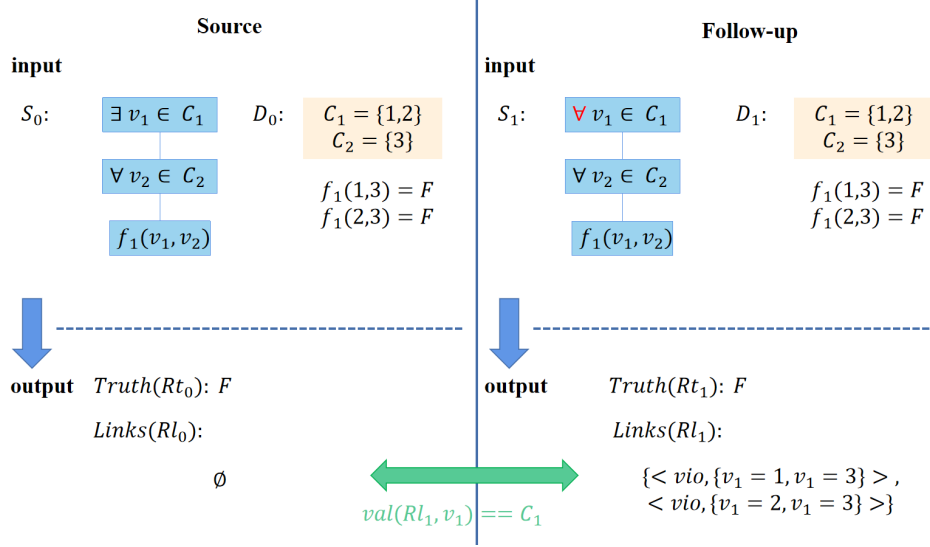


Figure 5: An example of MR-Cons 2

Fig. 5 has shown an example of MR-Cons 2. In the example, the root of the source constraint  $S_0$  is  $\exists$ . We transform it to  $\forall$  to obtain  $S_1$  for follow-up execution. Since  $Rt_0$  is  $F$ , this is the case 1 of MR-Cons 2. Therefore,  $Rt_1$  should be  $F$  and  $val(Rt_1, v_1)$  should be the same  $C_1 = \{1, 2\}$ . And that's exactly what happened in the follow-up output, which indicates that it satisfies MR-Cons 2.

### 3.3 MR-Cons 3

Before we introduce more MRs, let's talk more about the links of binary quantifier.

Let  $S : f_l \text{ op } f_r$  be a constraint, where  $op$  is a binary quantifier (e.g. *and*, *or*, *implies*) and  $f_l/f_r$  is the left/right sub-constraint of  $op$ . According to the rules of link generation [8], the links of  $S$  (denoted as  $ls$ ) is one of the following 5 cases:

- (1).  $ls$  is empty.
- (2).  $ls$  only contains links from left sub-constraint (denoted as  $ls_{f_l}$ ).
- (3).  $ls$  only contains links from right sub-constraint (denoted as  $ls_{f_r}$ ).
- (4).  $ls$  is the union of  $ls_{f_l}$  and  $ls_{f_r}$ .
- (5).  $ls$  is the cartesian product of  $ls_{f_l}$  and  $ls_{f_r}$ .

For the convenience of description, we use the  $flag(ls, S)$  to denote these cases. For example, if  $flag(ls, S) == 2$ , it means links composition of constraint  $S$  is case (2) above, i.e.  $ls$  only contains links from its left sub-constraint.

Now we can explain our MR-Cons 3 as follows:

**MR-Cons 3:** If the root of the constraint  $S_0$  is *and*, which can be in form of  $f_1$  and  $f_2$ , we transform it to  $f_1$  or  $f_2$  to obtain  $S_1$ . Then, in the follow-up execution,  $Rt_1$  and  $Rl_1$  should satisfy different conditions according to the following 8 cases:

- Case 1: If  $Rt_0$  is  $T$  and  $flag(Rl_0, S_0) == 1$ ,  $Rt_1$  should be  $T$  and  $flag(Rl_1, S_1) == 1$ .
- Case 2: If  $Rt_0$  is  $T$  and  $flag(Rl_0, S_0) == 2$ ,  $Rt_1$  should be  $T$  and  $flag(Rl_1, S_1) == 2$ .
- Case 3: If  $Rt_0$  is  $T$  and  $flag(Rl_0, S_0) == 3$ ,  $Rt_1$  should be  $T$  and  $flag(Rl_1, S_1) == 3$ .
- Case 4: If  $Rt_0$  is  $T$  and  $flag(Rl_0, S_0) == 5$ ,  $Rt_1$  should be  $T$  and  $flag(Rl_1, S_1) == 4$ .
- Case 5: If  $Rt_0$  is  $F$  and  $flag(Rl_0, S_0) == 4$ ,  $Rt_1$  should be  $F$  and  $flag(Rl_1, S_1) == 5$ .
- Case 6: If  $Rt_0$  is  $F$  and  $flag(Rl_0, S_0) == 2$ , either  $Rt_1 == F \wedge flag(Rl_1, S_1) == 2$ , or  $Rt_1 == T \wedge (flag(Rl_1, S_1) == 1 \vee flag(Rl_1, S_1) == 3)$ .
- Case 7: If  $Rt_0$  is  $F$  and  $flag(Rl_0, S_0) == 3$ , either  $Rt_1 == F \wedge flag(Rl_1, S_1) == 3$ , or  $Rt_1 == T \wedge (flag(Rl_1, S_1) == 1 \vee flag(Rl_1, S_1) == 2)$ .
- Case 8: If  $Rt_0$  is  $F$  and  $flag(Rl_0, S_0) == 1$ , either  $Rt_1 == F \wedge flag(Rl_1, S_1) == 1$ , or  $Rt_1 == T \wedge (flag(Rl_1, S_1) == 1 \vee flag(Rl_1, S_1) == 2 \vee flag(Rl_1, S_1) == 3)$ .

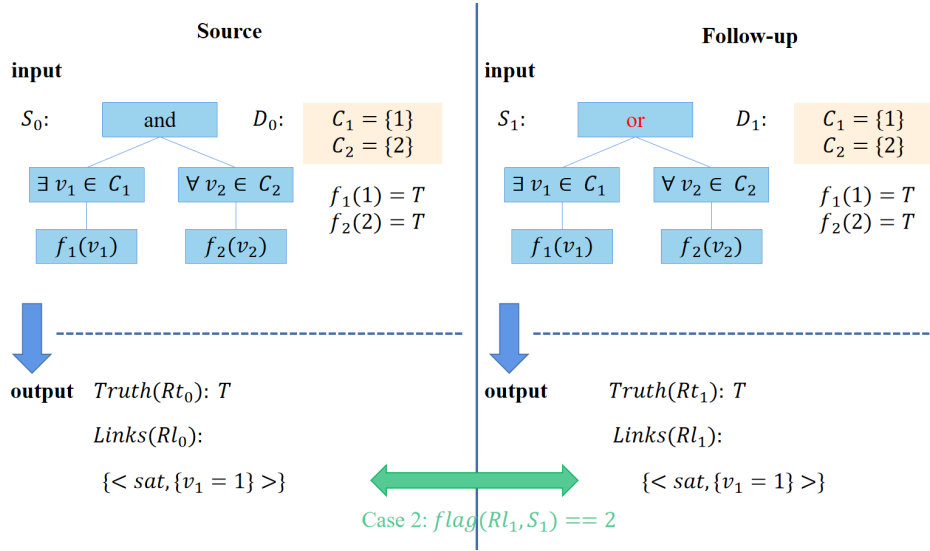


Figure 6: An example of MR-Cons 3

Fig. 6 has shown an example of MR-Cons 3. In the example, the root of the source constraint  $S_0$  is *and*. We transform it to *or* to obtain  $S_1$  for follow-up execution. Since  $Rt_0$  is  $T$  and  $Rl_0$  only contains the variables from the left sub-constraint, i.e.  $flag(Rl_0, S_0) == 2$ , this is the case 2 of MR-Cons 3. Therefore,  $Rt_1$  should be  $F$  and  $Rl_1$  should only contain the variables from the left sub-constraint, i.e.  $flag(Rl_1, S_1) == 2$ . And that's exactly what happened in the follow-up output, which indicates that it satisfies MR-Cons 3.

### 3.4 MR-Cons 4

**MR-Cons 4:** If the root of the constraint  $S_0$  is *or*, which can be in form of  $f_1$  or  $f_2$ , we transform it to  $f_1$  and  $f_2$  to obtain  $S_1$ . Then, in the follow-up execution,  $Rt_1$  and  $Rl_1$  should satisfy different conditions according to the following 8 cases:

- Case 1: If  $Rt_0$  is  $F$  and  $flag(Rl_0, S_0) == 1$ ,  $Rt_1$  should be  $F$  and  $flag(Rl_1, S_1) == 1$ .
- Case 2: If  $Rt_0$  is  $F$  and  $flag(Rl_0, S_0) == 2$ ,  $Rt_1$  should be  $F$  and  $flag(Rl_1, S_1) == 2$ .
- Case 3: If  $Rt_0$  is  $F$  and  $flag(Rl_0, S_0) == 3$ ,  $Rt_1$  should be  $F$  and  $flag(Rl_1, S_1) == 3$ .

- Case 4: If  $Rt_0$  is  $F$  and  $flag(Rl_0, S_0) == 5$ ,  $Rt_1$  should be  $F$  and  $flag(Rl_1, S_1) == 4$ .
- Case 5: If  $Rt_0$  is  $T$  and  $flag(Rl_0, S_0) == 4$ ,  $Rt_1$  should be  $T$  and  $flag(Rl_1, S_1) == 5$ .
- Case 6: If  $Rt_0$  is  $T$  and  $flag(Rl_0, S_0) == 2$ , either  $Rt_1 == T \wedge flag(Rl_1, S_1) == 2$ , or  $Rt_1 == F \wedge (flag(Rl_1, S_1) == 1 \vee flag(Rl_1, S_1) == 3)$ .
- Case 7: If  $Rt_0$  is  $T$  and  $flag(Rl_0, S_0) == 3$ , either  $Rt_1 == T \wedge flag(Rl_1, S_1) == 3$ , or  $Rt_1 == F \wedge (flag(Rl_1, S_1) == 1 \vee flag(Rl_1, S_1) == 2)$ .
- Case 8: If  $Rt_0$  is  $T$  and  $flag(Rl_0, S_0) == 1$ , either  $Rt_1 == T \wedge flag(Rl_1, S_1) == 1$ , or  $Rt_1 == F \wedge (flag(Rl_1, S_1) == 1 \vee flag(Rl_1, S_1) == 2 \vee flag(Rl_1, S_1) == 3)$ .

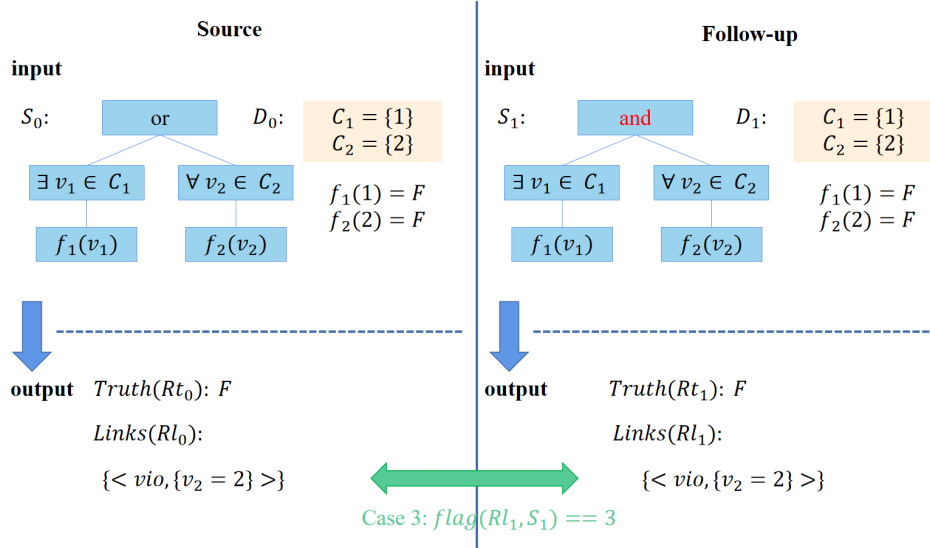


Figure 7: An example of MR-Cons 4

Fig. 7 has shown an example of MR-Cons 4. In the example, the root of the source constraint  $S_0$  is *or*. We transform it to *and* to obtain  $S_1$  for follow-up execution. Since  $Rt_0$  is  $F$  and  $Rl_0$  only contains the variables from the right sub-constraint, i.e.  $flag(Rl_0, S_0) == 3$ , this is the case 3 of MR-Cons 4. Therefore,  $Rt_1$  should be  $F$  and  $Rl_1$  should only contain the variables from the right sub-constraint, i.e.  $flag(Rl_1, S_1) == 3$ . And that's exactly what happened in the follow-up output, which indicates that it satisfies MR-Cons 4.

### 3.5 MR-Cons 5

**MR-Cons 5:** If the root of the constraint  $S_0$  is *or*, which can be in form of  $f_1$  or  $f_2$ , we transform it to  $f_1$  *implies*  $f_2$  to obtain  $S_1$ . Then, in the follow-up execution,  $Rt_1$  and  $Rl_1$  should satisfy different conditions according to the following 8 cases:

- Case 1: If  $Rt_0$  is  $F$  and  $flag(Rl_0, S_0) == 1$ ,  $Rt_1$  should be  $T$  and  $flag(Rl_1, S_1) == 1$ .
- Case 2: If  $Rt_0$  is  $F$  and  $flag(Rl_0, S_0) == 2$ ,  $Rt_1$  should be  $T$  and  $flag(Rl_1, S_1) == 2$ .
- Case 3: If  $Rt_0$  is  $F$  and  $flag(Rl_0, S_0) == 3$ ,  $Rt_1$  should be  $T$  and  $flag(Rl_1, S_1) == 1$ .
- Case 4: If  $Rt_0$  is  $F$  and  $flag(Rl_0, S_0) == 5$ ,  $Rt_1$  should be  $T$  and  $flag(Rl_1, S_1) == 2$ .
- Case 5: If  $Rt_0$  is  $T$  and  $flag(Rl_0, S_0) == 4$ ,  $Rt_1$  should be  $T$  and  $flag(Rl_1, S_1) == 3$ .

- Case 6: If  $Rt_0$  is  $T$  and  $flag(Rl_0, S_0) == 3$ ,  $Rt_1$  should be  $T$  and  $flag(Rl_1, S_1) == 3 \vee flag(Rl_1, S_1) == 4$ .
- Case 7: If  $Rt_0$  is  $T$  and  $flag(Rl_0, S_0) == 2$ , either  $Rt_1 == T \wedge flag(Rl_1, S_1) == 1$ , or  $Rt_1 == F \wedge (flag(Rl_1, S_1) == 2 \vee flag(Rl_1, S_1) == 5)$ .
- Case 8: If  $Rt_0$  is  $T$  and  $flag(Rl_0, S_0) == 1$ , either  $Rt_1 == T \wedge (flag(Rl_1, S_1) == 1 \vee flag(Rl_1, S_1) == 2)$ , or  $Rt_1 == F \wedge (flag(Rl_1, S_1) == 1 \vee flag(Rl_1, S_1) == 3)$ .

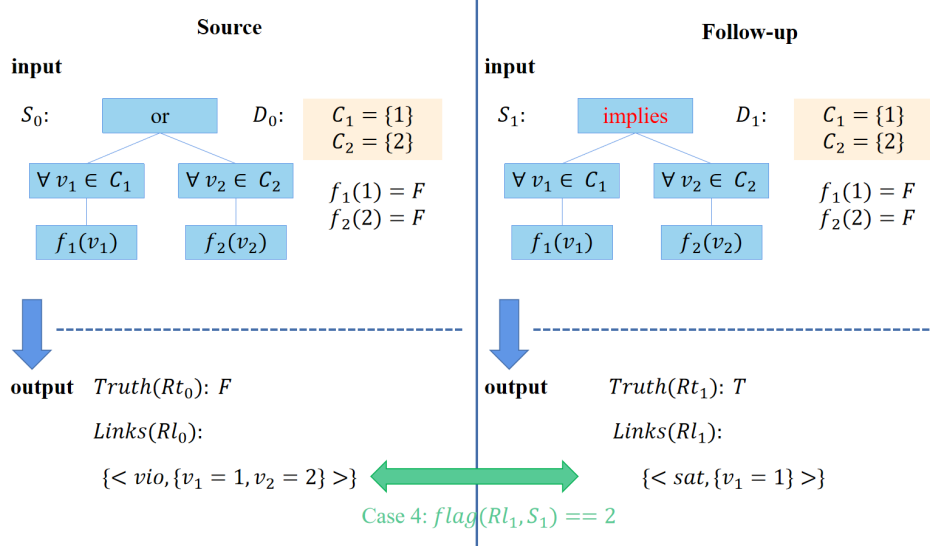


Figure 8: An example of MR-Cons 5

Fig. 8 has shown an example of MR-Cons 5. In the example, the root of the source constraint  $S_0$  is *or*. We transform it to *implies* to obtain  $S_1$  for follow-up execution. Since  $Rt_0$  is  $F$  and  $Rl_0$  is the cartesian product of the links of left sub-constraint and the links of right sub-constraint, i.e.  $flag(Rl_0, S_0) == 5$ , this is the case 4 of MR-Cons 5. Therefore,  $Rt_1$  should be  $T$  and  $Rl_1$  should only contain the variables from the left sub-constraint, i.e.  $flag(Rl_1, S_1) == 2$ . And that's exactly what happened in the follow-up output, which indicates that it satisfies MR-Cons 5.

### 3.6 MR-Cons 6

**MR-Cons 6:** If the root of the constraint  $S_0$  is *implies*, which can be in form of  $f_1$  *implies*  $f_2$ , we transform it to  $f_1$  or  $f_2$  to obtain  $S_1$ . Then, in the follow-up execution,  $Rt_1$  and  $Rl_1$  should satisfy different conditions according to the following 8 cases:

- Case 1: If  $Rt_0$  is  $F$  and  $flag(Rl_0, S_0) == 1$ ,  $Rt_1$  should be  $T$  and  $flag(Rl_1, S_1) == 1$ .
- Case 2: If  $Rt_0$  is  $F$  and  $flag(Rl_0, S_0) == 2$ ,  $Rt_1$  should be  $T$  and  $flag(Rl_1, S_1) == 2$ .
- Case 3: If  $Rt_0$  is  $F$  and  $flag(Rl_0, S_0) == 3$ ,  $Rt_1$  should be  $T$  and  $flag(Rl_1, S_1) == 1$ .
- Case 4: If  $Rt_0$  is  $F$  and  $flag(Rl_0, S_0) == 5$ ,  $Rt_1$  should be  $T$  and  $flag(Rl_1, S_1) == 2$ .
- Case 5: If  $Rt_0$  is  $T$  and  $flag(Rl_0, S_0) == 4$ ,  $Rt_1$  should be  $T$  and  $flag(Rl_1, S_1) == 3$ .
- Case 6: If  $Rt_0$  is  $T$  and  $flag(Rl_0, S_0) == 3$ ,  $Rt_1$  should be  $T$  and  $flag(Rl_1, S_1) == 3 \vee flag(Rl_1, S_1) == 4$ .
- Case 7: If  $Rt_0$  is  $T$  and  $flag(Rl_0, S_0) == 2$ , either  $Rt_1 == T \wedge flag(Rl_1, S_1) == 1$ , or  $Rt_1 == F \wedge (flag(Rl_1, S_1) == 2 \vee flag(Rl_1, S_1) == 5)$ .



- Case 8: If  $Rt_0$  is  $T$  and  $flag(Rl_0, S_0) == 1$ , either  $Rt_1 == T \wedge (flag(Rl_1, S_1) == 1 \vee flag(Rl_1, S_1) == 2)$ , or  $Rt_1 == F \wedge (flag(Rl_1, S_1) == 1 \vee flag(Rl_1, S_1) == 3)$ .

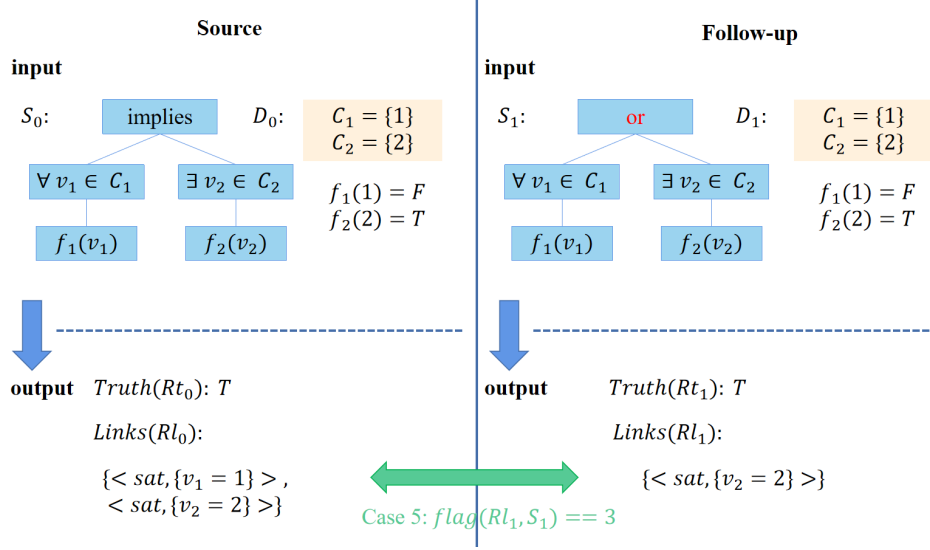


Figure 9: An example of MR-Cons 6

Fig. 9 has shown an example of MR-Cons 6. In the example, the root of the source constraint  $S_0$  is *implies*. We transform it to *or* to obtain  $S_1$  for follow-up execution. Since  $Rt_0$  is  $T$  and  $Rl_0$  is the union of the links of left sub-constraint and the links of right sub-constraint, i.e.  $flag(Rl_0, S_0) == 4$ , this is the case 5 of MR-Cons 6. Therefore,  $Rt_1$  should be  $T$  and  $Rl_1$  should only contain the variables from the right sub-constraint, i.e.  $flag(Rl_1, S_1) == 3$ . And that's exactly what happened in the follow-up output, which indicates that it satisfies MR-Cons 6.

### 3.7 MR-Cons 7

**MR-Cons 7:** If the root of the constraint  $S_0$  is *and*, which can be in form of  $f_1$  and  $f_2$ , we transform it to  $f_1$  *implies*  $f_2$  to obtain  $S_1$ . Then, in the follow-up execution,  $Rt_1$  and  $Rl_1$  should satisfy different conditions according to the following 8 cases:

- Case 1: If  $Rt_0$  is  $T$  and  $flag(Rl_0, S_0) == 1$ ,  $Rt_1$  should be  $T$  and  $flag(Rl_1, S_1) == 1$ .
- Case 2: If  $Rt_0$  is  $T$  and  $flag(Rl_0, S_0) == 2$ ,  $Rt_1$  should be  $T$  and  $flag(Rl_1, S_1) == 1$ .
- Case 3: If  $Rt_0$  is  $T$  and  $flag(Rl_0, S_0) == 3$ ,  $Rt_1$  should be  $T$  and  $flag(Rl_1, S_1) == 3$ .
- Case 4: If  $Rt_0$  is  $T$  and  $flag(Rl_0, S_0) == 5$ ,  $Rt_1$  should be  $T$  and  $flag(Rl_1, S_1) == 3$ .
- Case 5: If  $Rt_0$  is  $F$  and  $flag(Rl_0, S_0) == 4$ ,  $Rt_1$  should be  $T$  and  $flag(Rl_1, S_1) == 2$ .
- Case 6: If  $Rt_0$  is  $F$  and  $flag(Rl_0, S_0) == 2$ ,  $Rt_1$  should be  $T$  and  $flag(Rl_1, S_1) == 2 \vee flag(Rl_1, S_1) == 4$ .
- Case 7: If  $Rt_0$  is  $F$  and  $flag(Rl_0, S_0) == 3$ , either  $Rt_1 == T \wedge flag(Rl_1, S_1) == 1$ , or  $Rt_1 == F \wedge (flag(Rl_1, S_1) == 3 \vee flag(Rl_1, S_1) == 5)$ .
- Case 8: If  $Rt_0$  is  $F$  and  $flag(Rl_0, S_0) == 1$ , either  $Rt_1 == T \wedge (flag(Rl_1, S_1) == 1 \vee flag(Rl_1, S_1) == 3)$ , or  $Rt_1 == F \wedge (flag(Rl_1, S_1) == 1 \vee flag(Rl_1, S_1) == 2)$ .

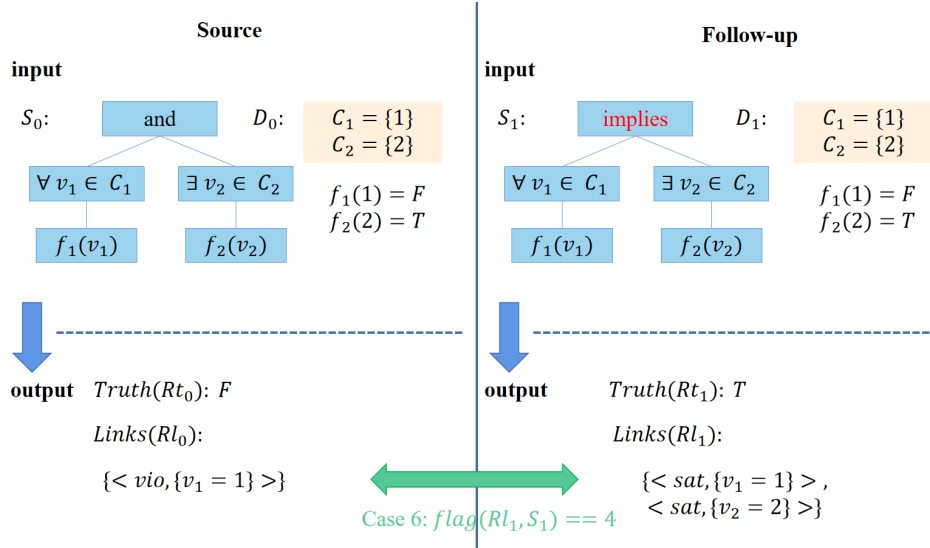


Figure 10: An example of MR-Cons 7

Fig. 10 has shown an example of MR-Cons 7. In the example, the root of the source constraint  $S_0$  is *and*. We transform it to *implies* to obtain  $S_1$  for follow-up execution. Since  $Rt_0$  is  $F$  and  $Rl_0$  only contains the variables from the left sub-constraint, i.e.  $flag(Rl_0, S_0) == 2$ , this is the case 6 of MR-Cons 7. Therefore,  $Rt_1$  should be  $T$  and  $flag(Rl_1, S_1)$  should be either 2 or 4. And in the follow-up output,  $Rl_1$  is the union of the links of left-constraint and the links of right-constraint, i.e.  $flag(Rl_1) == 4$ , which indicates that it satisfies MR-Cons 7.

### 3.8 MR-Cons 8

**MR-Cons 8:** If the root of the constraint  $S_0$  is *implies*, which can be in form of  $f_1$  *implies*  $f_2$ , we transform it to  $f_1$  *and*  $f_2$  to obtain  $S_1$ . Then, in the follow-up execution,  $Rt_1$  and  $Rl_1$  should satisfy different conditions according to the following 8 cases:

- Case 1: If  $Rt_0$  is  $F$  and  $flag(Rl_0, S_0) == 1$ ,  $Rt_1$  should be  $F$  and  $flag(Rl_1, S_1) == 1$ .
- Case 2: If  $Rt_0$  is  $F$  and  $flag(Rl_0, S_0) == 2$ ,  $Rt_1$  should be  $F$  and  $flag(Rl_1, S_1) == 1$ .
- Case 3: If  $Rt_0$  is  $F$  and  $flag(Rl_0, S_0) == 3$ ,  $Rt_1$  should be  $F$  and  $flag(Rl_1, S_1) == 3$ .
- Case 4: If  $Rt_0$  is  $F$  and  $flag(Rl_0, S_0) == 5$ ,  $Rt_1$  should be  $F$  and  $flag(Rl_1, S_1) == 3$ .
- Case 5: If  $Rt_0$  is  $T$  and  $flag(Rl_0, S_0) == 4$ ,  $Rt_1$  should be  $F$  and  $flag(Rl_1, S_1) == 2$ .
- Case 6: If  $Rt_0$  is  $T$  and  $flag(Rl_0, S_0) == 2$ ,  $Rt_1$  should be  $F$  and  $flag(Rl_1, S_1) == 2 \vee flag(Rl_1, S_1) == 4$ .
- Case 7: If  $Rt_0$  is  $T$  and  $flag(Rl_0, S_0) == 3$ , either  $Rt_1 == F \wedge flag(Rl_1, S_1) == 1$ , or  $Rt_1 == T \wedge (flag(Rl_1, S_1) == 3 \vee flag(Rl_1, S_1) == 5)$ .
- Case 8: If  $Rt_0$  is  $T$  and  $flag(Rl_0, S_0) == 1$ , either  $Rt_1 == F \wedge (flag(Rl_1, S_1) == 1 \vee flag(Rl_1, S_1) == 3)$ , or  $Rt_1 == T \wedge (flag(Rl_1, S_1) == 1 \vee flag(Rl_1, S_1) == 2)$ .

Fig. 11 has shown an example of MR-Cons 8. In the example, the root of the source constraint  $S_0$  is *implies*. We transform it to *and* to obtain  $S_1$  for follow-up execution. Since  $Rt_0$  is  $T$  and  $Rl_0$  only contains the variables from the right sub-constraint, i.e.  $flag(Rl_0, S_0) == 3$ , this is the case 7 of MR-Cons 8. In the follow-up output,  $Rt_1$  is  $F$  and  $Rl_1$  is empty, i.e.  $flag(Rl_1, S_1) == 1$ , which satisfies the assertion condition of case 7, and therefore, satisfies MR-Cons 8.

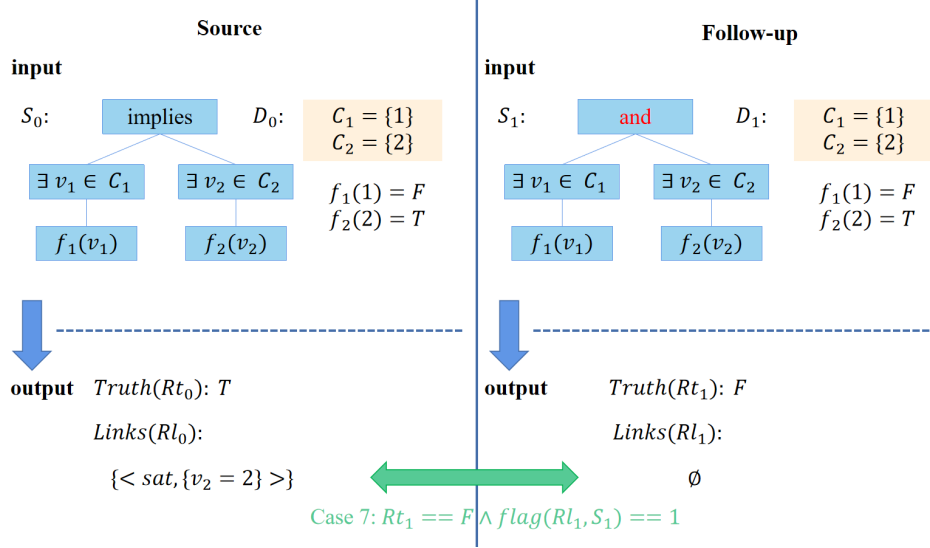


Figure 11: An example of MR-Cons 8

### 3.9 MR-Cons 9

**MR-Cons 9:** If the root of the constraint  $S_0$  is *implies*, which can be in form of  $f_1$  *implies*  $f_2$ , we transform it to  $f_2$  *implies*  $f_1$  to obtain  $S_1$ . Then, in the follow-up execution,  $Rt_1$  and  $Rl_1$  should satisfy different conditions according to the following 8 cases:

- Case 1: If  $Rt_0$  is  $F$  and  $flag(Rl_0, S_0) == 1$ ,  $Rt_1$  should be  $T$  and  $flag(Rl_1, S_1) == 1$ .
- Case 2: If  $Rt_0$  is  $F$  and  $flag(Rl_0, S_0) == 2$ ,  $Rt_1$  should be  $T$  and  $flag(Rl_1, S_1) == 2$ .
- Case 3: If  $Rt_0$  is  $F$  and  $flag(Rl_0, S_0) == 3$ ,  $Rt_1$  should be  $T$  and  $flag(Rl_1, S_1) == 3$ .
- Case 4: If  $Rt_0$  is  $F$  and  $flag(Rl_0, S_0) == 5$ ,  $Rt_1$  should be  $T$  and  $flag(Rl_1, S_1) == 4$ .
- Case 5: If  $Rt_0$  is  $T$  and  $flag(Rl_0, S_0) == 4$ ,  $Rt_1$  should be  $F$  and  $flag(Rl_1, S_1) == 5$ .
- Case 6: If  $Rt_0$  is  $T$  and  $flag(Rl_0, S_0) == 2$ , either  $Rt_1 == F \wedge flag(Rl_1, S_1) == 2$ , or  $Rt_1 == T \wedge (flag(Rl_1, S_1) == 1 \vee flag(Rl_1, S_1) == 3)$ .
- Case 7: If  $Rt_0$  is  $T$  and  $flag(Rl_0, S_0) == 3$ , either  $Rt_1 == F \wedge flag(Rl_1, S_1) == 3$ , or  $Rt_1 == T \wedge (flag(Rl_1, S_1) == 1 \vee flag(Rl_1, S_1) == 2)$ .
- Case 8: If  $Rt_0$  is  $T$  and  $flag(Rl_0, S_0) == 1$ , either  $Rt_1 == F \wedge flag(Rl_1, S_1) == 1$ , or  $Rt_1 == T \wedge (flag(Rl_1, S_1) == 1 \vee flag(Rl_1, S_1) == 2 \vee flag(Rl_1, S_1) == 3)$ .

Fig. 12 has shown an example of MR-Cons 9. In the example, the root of the source constraint  $S_0$  is *implies*. We switch its two sub-formulas to obtain  $S_1$  for follow-up execution. Since  $Rt_0$  is  $T$  and  $Rl_0$  is empty, i.e.  $flag(Rl_0, S_0) == 1$ , this is the case 8 of MR-Cons 9. In the follow-up output,  $Rt_1$  is  $T$  and  $Rl_1$  only contains the variables from the left sub-constraint, i.e.  $flag(Rl_1, S_1) == 2$ , which satisfies the assertion condition of case 8, and therefore, satisfies MR-Cons 9.

## 4 MR-All

There are two different MRs in this part, and they transform the data under checking and the constraint simultaneously.

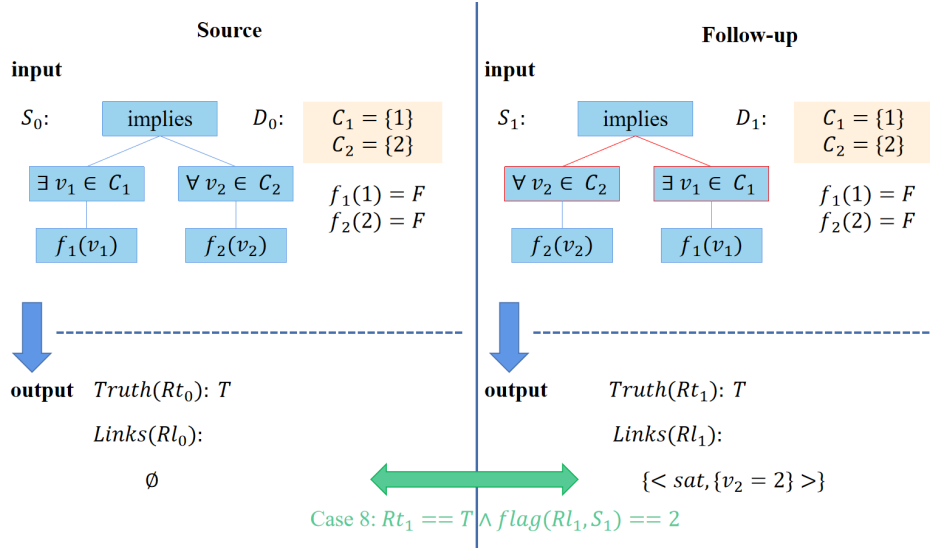


Figure 12: An example of MR-Cons 9

#### 4.1 Positive quantifier and negative quantifier

Let's start with an in-depth analysis of the quantifier (e.g.  $\forall$ ,  $\exists$ , *and*, *or*) in the constraint. A constraint's quantifiers can be divided into two parts: *positive* and *negative*, according to their impact to the whole constraint. A positive quantifier supports the root's violation or satisfaction while a negative quantifier supports it in the opposite way. It means if a positive quantifier is  $F$  under given data, the whole constraint tends to be  $F$  while if a positive quantifier is  $T$ , the whole constraint tends to be  $T$ . In brief, a positive quantifier with  $T$  or a negative quantifier with  $F$  contributes to the whole constraint taking the truth value of  $T$ ; a positive quantifier with  $F$  or a negative quantifier with  $T$  contributes to the whole constraint taking the truth value of  $F$ .

Let us consider the example in the Fig. 13. In the figure, quantifiers colored with green stands for positive quantifiers and quantifiers colored with red stands for negative quantifiers. Quantifier *not* in constraint  $S$  is positive, which means if it is  $T$ , it tends to make  $S$  be  $T$ , while if it is  $F$ , it tends to make  $S$  be  $F$ .

In definitions, only the quantifier that is transitively through *implies* from the left or *not* from the root can reverse its supporting direction initialized to be positive, which has been clearly illustrated in Fig. 13.

#### 4.2 Constraint transformation

Since we want to transform both the data part and constraint part at the same time, we need to control the transformation impact. We control to insert a new formula which naturally requires new data being added into the data part. We propose two kinds of transformations: *AND-Transformation* and *OR-Transformation*. AND-Transformation means linking a quantifier with a newly added *and*, which links to a newly added constraint as another branch. Similarly, OR-Transformation means linking quantifier with a newly added *or*. Note that the choice of the linking node and the newly added constraint can be arbitrary which means there are a lot of instances of these two kinds of transformation even for the same constraint. Fig 13 has clearly shown an example of two kinds of transformations.

#### 4.3 MR-All 1

Now, we can elaborate on our MR-All 1 using concepts above:

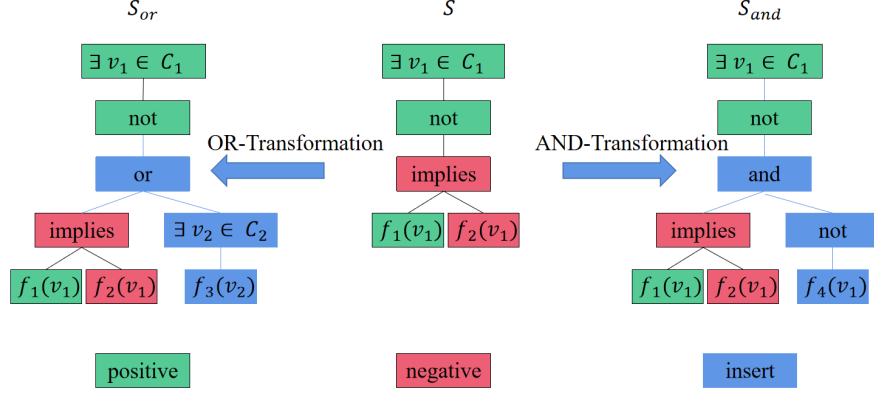


Figure 13: An example of constraint transformation

**MR-All 1:** If  $Rt_0 == F$ , we apply AND-Transformation to any positive quantifier or apply OR-Transformation to any negative quantifier in  $S_0$  to obtain  $D_1$  and  $S_1$ . Then, in the follow-up execution,  $Rt_1$  should still be  $F$  and  $Rl_1$  should satisfy the condition:  $Rl_0 \preceq Rl_1$ .

MR-All 1 indicates that AND-Transformation to any positive quantifier or OR-Transformation when  $Rt_0$  is a kind of inconsistency-incurring change for input, which would lead to more violations and make the  $Rt_0$  still be  $F$  and  $Rl_1$  cover  $Rl_0$ .

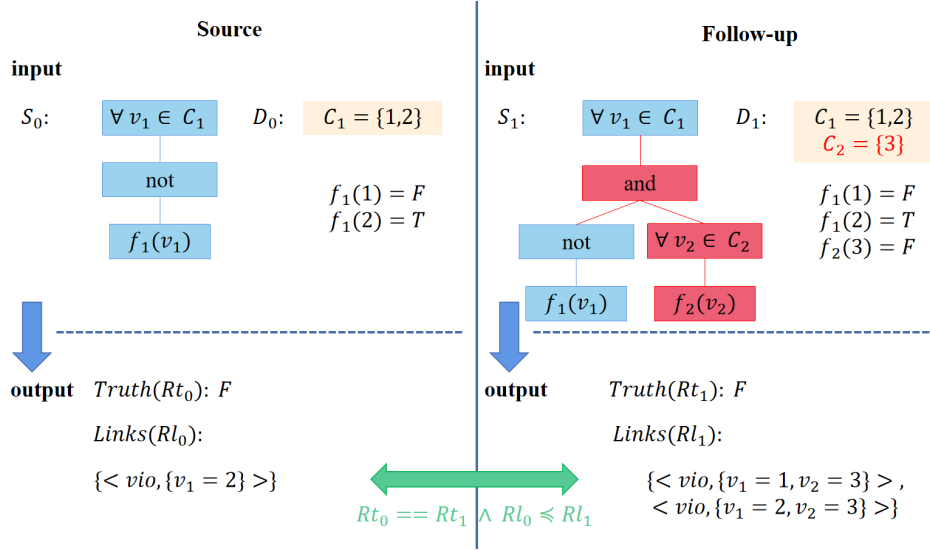


Figure 14: An example of MR-All 1

Fig. 14 has shown an example of MR-All 1. In the example, the source output  $Rt_0$  is  $F$ . We apply AND-Transformation by inserting a newly *and* node above a positive quantifier *not* for follow-up execution. Then,  $Rt_1$  is still  $F$  which is the same as the  $Rt_0$ , and  $Rl_1$  covers the  $Rl_0$ , i.e.  $Rl_0 \preceq Rl_1$ . That is, the output of the follow-up execution satisfies MR-All 1.

#### 4.4 MR-All 2

MR-All 2 is the other side of MR-All 1:

**MR-All 2:** If  $Rt_0 == T$ , we apply OR-Transformation to any positive quantifier or apply AND-Transformation to any negative quantifier in  $S_0$  to obtain  $D_1$  and  $S_1$ . Then, in the follow-up execution,  $Rt_1$  should still be  $T$  and  $Rl_1$  should satisfy the condition:  $Rl_0 \preceq Rl_1$ .

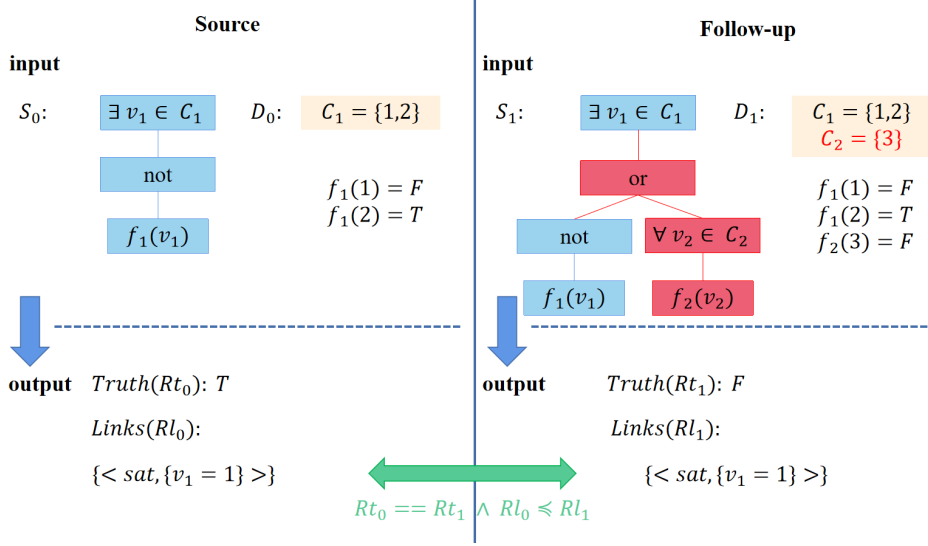


Figure 15: An example of MR-All 2

Fig. 15 has shown an example of MR-All 2. In the example, the source output  $Rt_0$  is  $T$ . We apply OR-Transformation by inserting a newly *and* node above a positive quantifier *not* for follow-up execution. Then,  $Rt_1$  is still  $T$  which is the same as the  $Rt_0$ , and  $Rl_1$  covers the  $Rl_0$ , i.e.  $Rl_0 \preceq Rl_1$ . That is, the output of the follow-up execution satisfies MR-All 2.

## 5 MR-CCT

If more internal states can be observed, e.g., CCT status like substantial parts accessible in existing work [10], we can additional design MRs with respect to measure the impact for such detailed CCT statuses under different transformations between source and follow-up executions.

### 5.1 CCT and SCCT

Normally, a constraint for checking can be expanded to a *Consistency Computation Tree* (CCT) based on both its static constraint structure and the current data under checking. SCCT [10] is substantial parts in CCT, which is the minimum set of nodes contributing to the link generation. Fig. 16 has shown an example of CCT and SCCT. In the figure, the nodes colored with purple are SCCT nodes and the links of this constraint is entirely up to them. By comparing the SCCT in the source output and the follow-up output through different transformations, we can design the MR-CCT.

In the next section, we will use  $SCCT_0$  and  $SCCT_1$  to denote the set of SCCT nodes in the source output and follow-up output separately.

### 5.2 MR-CCT 1

MR-CCT 1 uses the transformation similar to MR-Data 1 and can be elaborated on as follow:

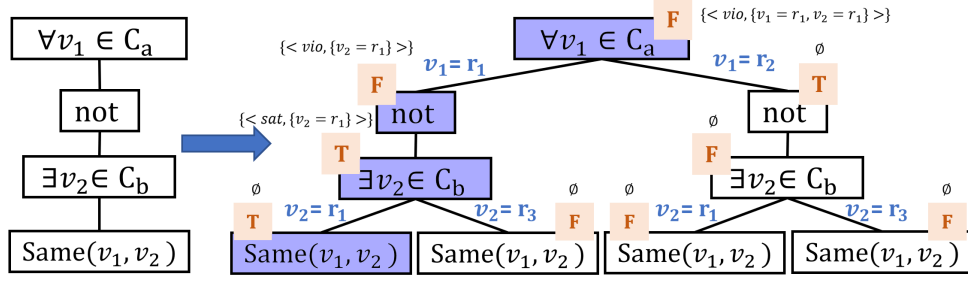


Figure 16: An example of CCT and SCCT

**MR-CCT 1:** If  $Rt_0 == F$ , we apply any change with an inconsistency-incurring impact on  $S_0$  to  $D_0$  to obtain  $D_1$ . Then, in the follow-up execution,  $SCCT_1$  should contain  $SCCT_0$ , i.e.  $SCCT_0 \subseteq SCCT_1$ .

MR-CCT 1 indicates that inconsistency-incurring changes applied to data will cause SCCT to become larger and cover the original SCCT. In fact,  $SCCT_0 \subseteq SCCT_1$  is another expression of  $l_0 \leq l_1$ .

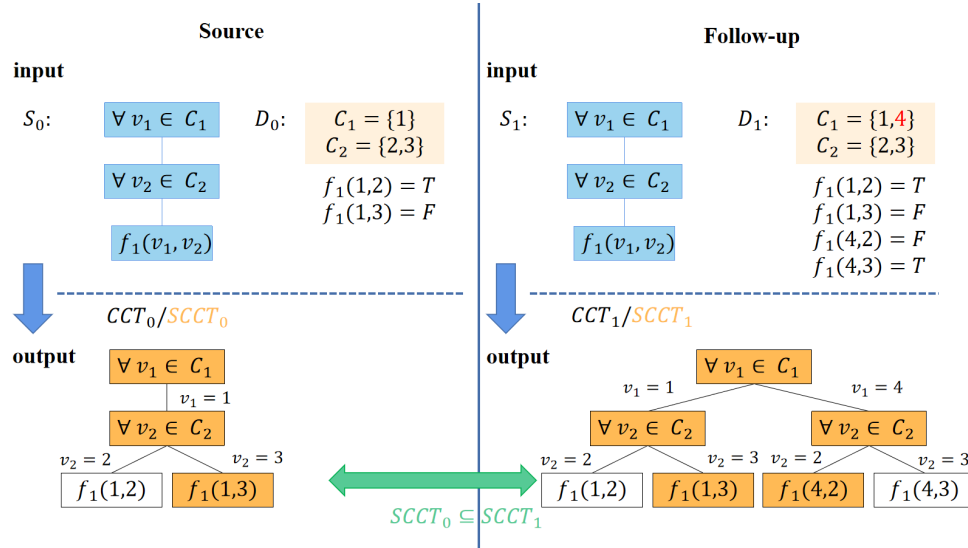


Figure 17: An example of MR-CCT 1

Fig. 17 has shown an example of MR-CCT 1. Both  $SCCT_0$  and  $SCCT_1$  have been colored with orange in the figure. In the example, the source output  $Rt_0$  is  $T$ . We apply an inconsistency-incurring impact change  $\langle +, C_1 \rangle$  to  $D_0$  to obtain  $D_1$  for follow-up execution. Then, in the follow-up output,  $SCCT_1$  contains all the nodes in the  $SCCT_0$ , i.e.  $SCCT_0 \subseteq SCCT_1$ , which can be vividly seen in the figure. That is, the output of the follow-up execution satisfies MR-CCT 1.

### 5.3 MR-CCT 2

Similarly, if we use the transformation mentioned in MR-All 1, then we can derive the MR-CCT 2:

**MR-CCT 2:** If  $Rt_0 == F$ , we apply AND-Transformation to any positive quantifier or apply OR-Transformation to any negative quantifier in  $S_0$  to obtain  $D_1$  and  $S_1$ . Then, in the follow-up execution, if the newly added nodes in  $SCCT_1$  are ignored,  $SCCT_1$  should contain  $SCCT_0$ , i.e.  $SCCT_0 \subseteq SCCT_1$ .

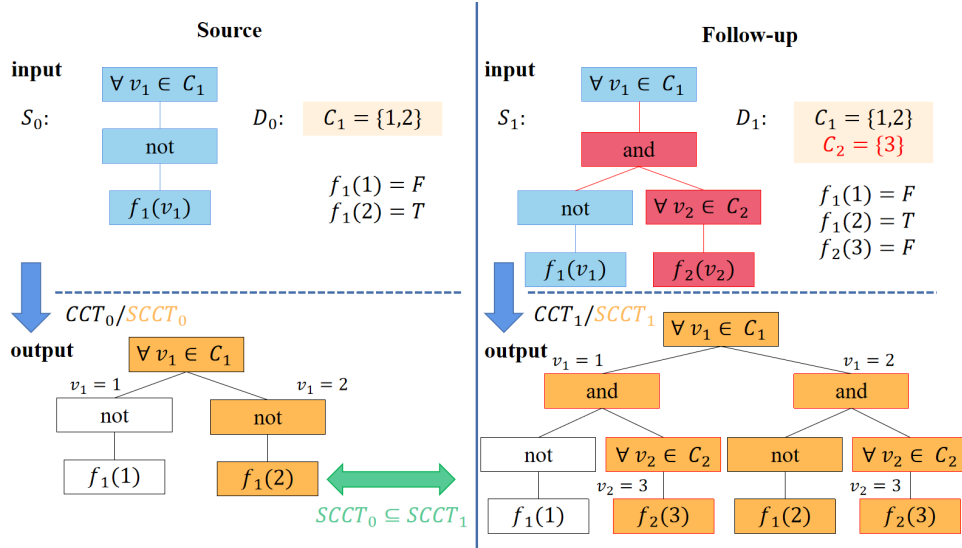


Figure 18: An example of MR-CCT 2

Fig. 18 has shown an example of MR-CCT 2. Both  $SCCT_0$  and  $SCCT_1$  have been colored with orange in the figure. In the example, the source output  $Rt_0$  is  $F$ . We apply AND-Transformation by inserting a newly *and* node above a positive quantifier *not* for follow-up execution. Then, in the follow-up output,  $SCCT_1$  contains all the nodes in the  $SCCT_0$ , i.e.  $SCCT_0 \subseteq SCCT_1$ , which can be vividly seen in the figure. That is, the output of the follow-up execution satisfies MR-CCT 2.