

# INFUSE: Towards Efficient Context Consistency by Incremental-Concurrent Check Fusion (Appendix)

Anonymous author(s)

## I. CHECKING SEMANTICS OF INFUSE

We first give a comprehensive explanation to link for better understanding. Each link explains how a specific consistency constraints is satisfied or violated via listing the certain related elements, where the link's `linkType` refers to the satisfaction or violation and the link's variable assignments contains mappings between variables and elements. Both satisfaction and violation links are useful since `not` and `implies` formulas invert them to each other for any consistency constraint.

We then define some necessary functions and operators to introduce full semantics of INFUSE:

1) *Affected function*: We define the *Affected* function to indicate whether a formula itself or its subformula is affected by the context changes in a constraint checking task. Given a formula from a consistency constraint, the *Affected* function returns **T** (means **True**) if and only if the formula itself or its subformula references a context involved in the *ASet*, *DSet* or *USet* associated with this constraint; otherwise, **F** (means **False**). Formally,

- $\text{Affected}(\forall/\exists v \in C(f)) = \text{T}$ , if  $\text{ASet} \neq \emptyset$  or  $\text{DSet} \neq \emptyset$  or  $\text{USet} \neq \emptyset$  or  $\text{Affected}(f) = \text{T}$ ; otherwise, **F**.
- $\text{Affected}((f_1) \text{ and/or/implies } (f_2)) = \text{T}$ , if  $\text{Affected}(f_1) = \text{T}$  or  $\text{Affected}(f_2) = \text{T}$ ; otherwise, **F**.
- $\text{Affected}(\text{not } (f)) = \text{T}$ , if  $\text{Affected}(f) = \text{T}$ ; otherwise, **F**.
- $\text{Affected}(bfunc(v_1, v_2, \dots, v_n)) = \text{F}$ .

2) *Flip and FlipSet functions*: We define the *Flip* function to reverse a link's `linkType` without changing the link's variable assignments, and the *FlipSet* function conducts *Flip* function for each link in a Link set. Formally,

- $\text{Flip}(\text{violated}, \text{variable assignments}) = (\text{satisfied}, \text{variable assignments})$ .
- $\text{Flip}(\text{satisfied}, \text{variable assignments}) = (\text{violated}, \text{variable assignments})$ .
- $\text{FlipSet}(S) = \{\text{Flip}(l) \mid l \in S\}$ .

3) *Type and Assignments functions*: We define the *Type* and *Assignments* functions to retrieve a link's specific link-Type and variable assignments respectively, i.e.,

- $\text{Type}(l) = l.\text{linkType}$ .
- $\text{Assignments}(l) = l.\text{variable assignments}$ .

4) *Concatenate function and  $\otimes$  operator*: We define *Concatenate* function to combine two links with the same `linkType` and produce a new one consisted of this `linkType` and the union of all concerned variable assignments in links. The  $\otimes$  operator concatenates two link sets by conducting the

*Concatenate* function to the link pairs combined with every link in set  $S_1$  and every link in set  $S_2$ , i.e.,

- $\text{Concatenate}(l_1, l_2) = (\text{Type}(l_1), \text{Assignments}(l_1) \cup \text{Assignments}(l_2))$ .
- $S_1 \otimes S_2 = \{\text{Concatenate}(l_1, l_2) \mid l_1 \in S_1 \wedge l_2 \in S_2\}$ , if  $S_1 \neq \emptyset$  and  $S_2 \neq \emptyset$ ; otherwise,  $S_1 \cup S_2$ .

### A. Truth Value Evaluation

In the following, we list INFUSE's truth value evaluation semantics for the remaining six formula types in constraint language (those for the universal formula have been introduced in the paper body), i.e., existential, and, or, implies, not, and *bfunc* formulas.

1) *Existential formula, i.e.,  $\exists v \in C(f)$* : Fig. 3 gives INFUSE's partial truth value evaluation semantics for the existential formula (also five cases simplified). Similar to that for the universal formula discussed in our paper body, it also invokes  $\text{eval}_{\text{entire}}$  and  $\text{eval}_{\text{partial}}$  functions (shown in Fig. 2) to calculate truth values of subformula  $f$  concerning different elements.

$$\begin{aligned} \tau_{\text{entire}}[\exists v \in C(f)]_{\alpha} = \\ \text{F} \vee \tau_{\text{entire}}[f]_{\text{bind}((v, x_1), \alpha)} \vee \dots \vee \tau_{\text{entire}}[f]_{\text{bind}((v, x_n), \alpha)} \mid x_i \in C \end{aligned}$$

Fig. 1. INFUSE's entire truth value evaluation semantics for the existential formula.

$$\begin{aligned} \text{eval}_{\text{entire}}(\tau[f]_{\text{bind}((v, x_i), \alpha)} \mid x_i \in \text{Set}) = \\ (1) \tau_{\text{entire}}[f]_{\text{bind}((v, x_1), \alpha)} \parallel \dots \parallel \tau_{\text{entire}}[f]_{\text{bind}((v, x_s), \alpha)}, \\ \text{if } \exists v \in C(f) \text{ is a concurrent point;} \\ (2) \tau_{\text{entire}}[f]_{\text{bind}((v, x_1), \alpha)} ; \dots ; \tau_{\text{entire}}[f]_{\text{bind}((v, x_s), \alpha)}, \\ \text{otherwise.} \end{aligned}$$

$$\begin{aligned} \text{eval}_{\text{partial}}(\tau[f]_{\text{bind}((v, x_i), \alpha)} \mid x_i \in \text{Set}) = \\ (1) \tau_{\text{partial}}[f]_{\text{bind}((v, x_1), \alpha)} \parallel \dots \parallel \tau_{\text{partial}}[f]_{\text{bind}((v, x_s), \alpha)}, \\ \text{if } \exists v \in C(f) \text{ is a concurrent point;} \\ (2) \tau_{\text{partial}}[f]_{\text{bind}((v, x_1), \alpha)} ; \dots ; \tau_{\text{partial}}[f]_{\text{bind}((v, x_s), \alpha)}, \\ \text{otherwise.} \end{aligned}$$

Fig. 2. Semantics of the *eval* functions (partial and entire checking).

2) *and, or, and implies formulas, i.e.,  $(f_1) \text{ and/or/implies } (f_2)$* : Fig. 4 gives INFUSE's entire truth value evaluation semantics for the tree formulas. Moreover, since *and*, *or*, and *implies* formulas reference none context, we only need to consider the *Affected* function on their subformulas  $f_1$  and  $f_2$ . Incremental evaluation would be applied on the affected subformulas as shown in Fig. 5.

<sup>1</sup>Last updated on April 8th, 2022.

$$\begin{aligned}
\tau_{\text{partial}}[\exists v \in C(f)]_{\alpha} = & \\
(1) & \tau_0[\exists v \in C(f)]_{\alpha}, \text{ if } \text{Affected}(f) = \text{F} \text{ and } (ASet = \emptyset \text{ and } DSet = \emptyset \text{ and } USet = \emptyset). \\
(2) & \tau_0[\exists v \in C(f)]_{\alpha} \vee t_1 \vee \dots \vee t_a, \text{ where } (t_1, \dots, t_a) = \text{eval}_{\text{entire}}(\tau[f]_{\text{bind}((v, y_j), \alpha)} \mid y_j \in ASet), \\
& \text{if } \text{Affected}(f) = \text{F} \text{ and } (ASet \neq \emptyset \text{ and } DSet = \emptyset \text{ and } USet = \emptyset). \\
(3) & \text{F} \vee \tau_0[f]_{\text{bind}((v, x_1), \alpha)} \vee \dots \vee \tau_0[f]_{\text{bind}((v, x_{n-a-u}), \alpha)} \vee t_1 \vee \dots \vee t_{a+u} \mid x_i \in C - (ASet \cup USet), \\
& \text{where } (t_1, \dots, t_{a+u}) = \text{eval}_{\text{entire}}(\tau[f]_{\text{bind}((v, y_j), \alpha)} \mid y_j \in ASet \cup USet), \\
& \text{if } \text{Affected}(f) = \text{F} \text{ and } (DSet \neq \emptyset \text{ or } USet \neq \emptyset). \\
(4) & \text{F} \vee t_1 \vee \dots \vee t_n, \text{ where } (t_1, \dots, t_n) = \text{eval}_{\text{partial}}(\tau[f]_{\text{bind}((v, x_i), \alpha)} \mid x_i \in C), \\
& \text{if } \text{Affected}(f) = \text{T} \text{ and } (ASet = \emptyset \text{ and } DSet = \emptyset \text{ and } USet = \emptyset). \\
(5) & \text{F} \vee t_1 \vee \dots \vee t_n, \text{ where } (t_1, \dots, t_{a+u}) = \text{eval}_{\text{entire}}(\tau[f]_{\text{bind}((v, y_j), \alpha)} \mid y_j \in ASet \cup USet) \\
& \text{and } (t_{a+u+1}, \dots, t_n) = \text{eval}_{\text{partial}}(\tau[f]_{\text{bind}((v, x_i), \alpha)} \mid x_i \in C - (ASet \cup USet)), \\
& \text{if } \text{Affected}(f) = \text{T} \text{ and } (ASet \neq \emptyset \text{ or } DSet \neq \emptyset \text{ or } USet \neq \emptyset).
\end{aligned}$$

Fig. 3. INFUSE's partial truth value evaluation semantics for the existential formula.

$$\tau_{\text{entire}}[(f_1) \text{ and } (f_2)]_{\alpha} = \tau_{\text{entire}}[f_1]_{\alpha} \wedge \tau_{\text{entire}}[f_2]_{\alpha}$$

$$\tau_{\text{entire}}[(f_1) \text{ or } (f_2)]_{\alpha} = \tau_{\text{entire}}[f_1]_{\alpha} \vee \tau_{\text{entire}}[f_2]_{\alpha}$$

$$\tau_{\text{entire}}[(f_1) \text{ implies } (f_2)]_{\alpha} = \neg \tau_{\text{entire}}[f_1]_{\alpha} \vee \tau_{\text{entire}}[f_2]_{\alpha}$$

Fig. 4. INFUSE's entire truth value evaluation semantics for and, or, and implies formulas.

$$\begin{aligned}
\tau_{\text{partial}}[(f_1) \text{ and } (f_2)]_{\alpha} = & \\
(1) & \tau_0[(f_1) \text{ and } (f_2)]_{\alpha}, \text{ if } \text{Affected}(f_1) = \text{Affected}(f_2) = \text{F}. \\
(2) & \tau_0[f_1]_{\alpha} \wedge \tau_{\text{partial}}[f_2]_{\alpha}, \text{ if } \text{Affected}(f_1) = \text{F}, \text{Affected}(f_2) = \text{T}. \\
(3) & \tau_{\text{partial}}[f_1]_{\alpha} \wedge \tau_0[f_2]_{\alpha}, \text{ if } \text{Affected}(f_1) = \text{T}, \text{Affected}(f_2) = \text{F}. \\
(4) & \tau_{\text{partial}}[f_1]_{\alpha} \wedge \tau_{\text{partial}}[f_2]_{\alpha}, \text{ if } \text{Affected}(f_1) = \text{Affected}(f_2) = \text{T}.
\end{aligned}$$

$$\begin{aligned}
\tau_{\text{partial}}[(f_1) \text{ or } (f_2)]_{\alpha} = & \\
(1) & \tau_0[(f_1) \text{ or } (f_2)]_{\alpha}, \text{ if } \text{Affected}(f_1) = \text{Affected}(f_2) = \text{F}. \\
(2) & \tau_0[f_1]_{\alpha} \vee \tau_{\text{partial}}[f_2]_{\alpha}, \text{ if } \text{Affected}(f_1) = \text{F}, \text{Affected}(f_2) = \text{T}. \\
(3) & \tau_{\text{partial}}[f_1]_{\alpha} \vee \tau_0[f_2]_{\alpha}, \text{ if } \text{Affected}(f_1) = \text{T}, \text{Affected}(f_2) = \text{F}. \\
(4) & \tau_{\text{partial}}[f_1]_{\alpha} \vee \tau_{\text{partial}}[f_2]_{\alpha}, \text{ if } \text{Affected}(f_1) = \text{Affected}(f_2) = \text{T}.
\end{aligned}$$

$$\begin{aligned}
\tau_{\text{partial}}[(f_1) \text{ implies } (f_2)]_{\alpha} = & \\
(1) & \tau_0[(f_1) \text{ implies } (f_2)]_{\alpha}, \text{ if } \text{Affected}(f_1) = \text{Affected}(f_2) = \text{F}. \\
(2) & \neg \tau_0[f_1]_{\alpha} \vee \tau_{\text{partial}}[f_2]_{\alpha}, \text{ if } \text{Affected}(f_1) = \text{F}, \text{Affected}(f_2) = \text{T}. \\
(3) & \neg \tau_{\text{partial}}[f_1]_{\alpha} \vee \tau_0[f_2]_{\alpha}, \text{ if } \text{Affected}(f_1) = \text{T}, \text{Affected}(f_2) = \text{F}. \\
(4) & \neg \tau_{\text{partial}}[f_1]_{\alpha} \vee \tau_{\text{partial}}[f_2]_{\alpha}, \text{ if } \text{Affected}(f_1) = \text{Affected}(f_2) = \text{T}.
\end{aligned}$$

Fig. 5. INFUSE's partial truth value evaluation semantics for and, or, and implies formulas.

3) *not* and *bfunc* formulas, i.e., *not* (*f*) and *bfunc*( $v_1, \dots, v_n$ ): Fig. 6 gives INFUSE's entire truth value evaluation semantics for the two formulas. Fig. 7 gives INFUSE's partial truth value evaluation semantics for the *not* formula and the *bfunc* formula. For the *not* formula, the reusability of its last truth value depends on the *Affected* function on its subformula *f*. For the *bfunc* formula, its last truth value is always reusable in partial semantics.

$$\tau_{\text{entire}}[(\text{not } (f))]_{\alpha} = \neg \tau_{\text{entire}}[f]_{\alpha}$$

$$\tau_{\text{entire}}[bfunc(v_1, \dots, v_n)]_{\alpha} = bfunc(v_1, \dots, v_n).$$

Fig. 6. INFUSE's entire truth value evaluation semantics for *not* and *bfunc* formulas.

$$\begin{aligned}
\tau_{\text{partial}}[(\text{not } (f))]_{\alpha} = & \\
(1) & \tau_0[(\text{not } (f))]_{\alpha}, \text{ if } \text{Affected}(f) = \text{F}. \\
(2) & \neg \tau_{\text{partial}}[f]_{\alpha}, \text{ if } \text{Affected}(f) = \text{T}.
\end{aligned}$$

$$\tau_{\text{partial}}[bfunc(v_1, \dots, v_n)]_{\alpha} = \tau_0[bfunc(v_1, \dots, v_n)]_{\alpha}.$$

Fig. 7. INFUSE's partial truth value evaluation semantics for *not* and *bfunc* formulas.

$$\begin{aligned}
\mathcal{L}_{\text{entire}}[\exists v \in C(f)]_{\alpha} = & \\
\{l \mid l \in \{(\text{satisfied}, \{(v, x_i)\})\} \otimes \mathcal{L}_{\text{entire}}[f]_{\text{bind}((v, x_i), \alpha)} \} & \\
\mid x_i \in C \wedge \tau[f]_{\text{bind}((v, x_i), \alpha)} = \text{T}. &
\end{aligned}$$

Fig. 8. INFUSE's entire link generation semantics for the existential formula.

$$\begin{aligned}
\text{gen}_{\text{entire}}(\mathcal{L}[f]_{\text{bind}((v, x_i), \alpha)} \mid x_i \in \text{Set} \wedge \tau[f]_{\text{bind}((v, x_i), \alpha)} = \text{T}) & \\
(1) \mathcal{L}_{\text{entire}}[f]_{\text{bind}((v, x_1), \alpha)} \parallel \dots \parallel \mathcal{L}_{\text{entire}}[f]_{\text{entire}((v, x_s), \alpha)}, & \\
\text{if } \exists v \in C(f) \text{ is a concurrent point.} & \\
(2) \mathcal{L}_{\text{entire}}[f]_{\text{bind}((v, x_1), \alpha)} ; \dots ; \mathcal{L}_{\text{entire}}[f]_{\text{bind}((v, x_s), \alpha)}, & \\
\text{otherwise.} &
\end{aligned}$$

$$\begin{aligned}
\text{gen}_{\text{partial}}(\mathcal{L}[f]_{\text{bind}((v, x_i), \alpha)} \mid x_i \in \text{Set} \wedge \tau[f]_{\text{bind}((v, x_i), \alpha)} = \text{T}) & \\
(1) \mathcal{L}_{\text{partial}}[f]_{\text{bind}((v, x_1), \alpha)} \parallel \dots \parallel \mathcal{L}_{\text{partial}}[f]_{\text{bind}((v, x_s), \alpha)}, & \\
\text{if } \exists v \in C(f) \text{ is a concurrent point.} & \\
(2) \mathcal{L}_{\text{partial}}[f]_{\text{bind}((v, x_1), \alpha)} ; \dots ; \mathcal{L}_{\text{partial}}[f]_{\text{bind}((v, x_s), \alpha)}, & \\
\text{otherwise.} &
\end{aligned}$$

Fig. 9. Semantics of the *gen* functions (partial and entire checking).

$$\begin{aligned}
\mathcal{L}_{\text{partial}}[\exists v \in C(f)]_{\alpha} = & \\
(1) & \mathcal{L}_0[\exists v \in C(f)]_{\alpha}, \text{ if } \text{Affected}(f) = \text{F} \text{ and } (A\text{Set} = \emptyset \text{ and } D\text{Set} = \emptyset \text{ and } U\text{Set} = \emptyset). \\
(2) & \mathcal{L}_0[\exists v \in C(f)]_{\alpha} \cup (\{(\text{satisfied}, \{v, y_1\})\} \otimes l_1) \cup \dots \cup (\{(\text{satisfied}, \{v, y_{a'}\})\} \otimes l_{a'}), \\
& \text{where } (l_1, \dots, l_{a'}) = \text{gen}_{\text{entire}}(\mathcal{L}[f]_{\text{bind}((v, y_j), \alpha)} \mid y_j \in A\text{Set} \wedge \tau[f]_{\text{bind}((v, y_j), \alpha)} = \text{T}), \\
& \text{if } \text{Affected}(f) = \text{F} \text{ and } (A\text{Set} \neq \emptyset \text{ and } D\text{Set} = \emptyset \text{ and } U\text{Set} = \emptyset). \\
(3) & (\{(\text{satisfied}, \{v, y_1\})\} \otimes l_1) \cup \dots \cup (\{(\text{satisfied}, \{v, y_{a'+u'}\})\} \otimes l_{a'+u'}) \cup \\
& \{l \mid l \in \{(\text{satisfied}, \{v, x_i\})\} \otimes \mathcal{L}_0[f]_{\text{bind}((v, x_i), \alpha)} \mid x_i \in C - (A\text{Set} \cup U\text{Set}) \wedge \tau[f]_{\text{bind}((v, x_i), \alpha)} = \text{T}, \\
& \text{where } (l_1, \dots, l_{a'+u'}) = \text{gen}_{\text{entire}}(\mathcal{L}[f]_{\text{bind}((v, y_j), \alpha)} \mid y_j \in A\text{Set} \cup U\text{Set} \wedge \tau[f]_{\text{bind}((v, y_j), \alpha)} = \text{T}), \\
& \text{if } \text{Affected}(f) = \text{F} \text{ and } (D\text{Set} \neq \emptyset \text{ or } U\text{Set} \neq \emptyset). \\
(4) & \emptyset \cup (\{(\text{satisfied}, \{v, x_1\})\} \otimes l_1) \cup \dots \cup (\{(\text{satisfied}, \{v, x_{n'}\})\} \otimes l_{n'}), \\
& \text{where } (l_1, \dots, l_{n'}) = \text{gen}_{\text{partial}}(\mathcal{L}[f]_{\text{bind}((v, x_i), \alpha)} \mid x_i \in C \wedge \tau[f]_{\text{bind}((v, x_i), \alpha)} = \text{T}), \\
& \text{if } \text{Affected}(f) = \text{T} \text{ and } (A\text{Set} = \emptyset \text{ and } D\text{Set} = \emptyset \text{ and } U\text{Set} = \emptyset). \\
(5) & \emptyset \cup (\{(\text{satisfied}, \{v, y_1\})\} \otimes l_1) \cup \dots \cup (\{(\text{satisfied}, \{v, y_{n'}\})\} \otimes l_{n'}), \\
& \text{where } (l_1, \dots, l_{a'+u'}) = \text{gen}_{\text{entire}}(\mathcal{L}[f]_{\text{bind}((v, y_j), \alpha)} \mid y_j \in A\text{Set} \cup U\text{Set} \wedge \tau[f]_{\text{bind}((v, y_j), \alpha)} = \text{T}) \\
& \text{and } (l_{a'+u'+1}, \dots, l_{n'}) = \text{gen}_{\text{partial}}(\mathcal{L}[f]_{\text{bind}((v, x_i), \alpha)} \mid x_i \in C - (A\text{Set} \cup U\text{Set}) \wedge \tau[f]_{\text{bind}((v, x_i), \alpha)} = \text{T}), \\
& \text{if } \text{Affected}(f) = \text{T} \text{ and } (A\text{Set} \neq \emptyset \text{ or } D\text{Set} \neq \emptyset \text{ or } U\text{Set} \neq \emptyset).
\end{aligned}$$

Fig. 10. INFUSE's partial link generation semantics for the existential formula.

### B. Link Generation

In the following, we list INFUSE's link generation semantics for the rest of formula types in constraint language similarly (those for the universal formula have been introduced in the paper body).

1) *Existential formula, i.e.,  $\exists v \in C(f)$* : Fig. 10 gives INFUSE's partial link generation semantics for the existential formula (five cases simplified). Similar to that for the universal formula, it also invokes  $\text{gen}_{\text{entire}}$  and  $\text{gen}_{\text{partial}}$  functions (shown in Fig. 9) to generate links of subformula  $f$  concerning different elements.

$$\begin{aligned}
\mathcal{L}_{\text{entire}}[(f_1) \text{ and } (f_2)]_{\alpha} = & \\
(1) & \mathcal{L}_{\text{entire}}[f_1]_{\alpha} \otimes \mathcal{L}_{\text{entire}}[f_2]_{\alpha}, \text{ if } \tau[f_1]_{\alpha} = \tau[f_2]_{\alpha} = \text{T}. \\
(2) & \mathcal{L}_{\text{entire}}[f_1]_{\alpha} \cup \mathcal{L}_{\text{entire}}[f_2]_{\alpha}, \text{ if } \tau[f_1]_{\alpha} = \tau[f_2]_{\alpha} = \text{F}. \\
(3) & \mathcal{L}_{\text{entire}}[f_2]_{\alpha}, \text{ if } \tau[f_1]_{\alpha} = \text{T}, \tau[f_2]_{\alpha} = \text{F}. \\
(4) & \mathcal{L}_{\text{entire}}[f_1]_{\alpha}, \text{ if } \tau[f_1]_{\alpha} = \text{F}, \tau[f_2]_{\alpha} = \text{T}.
\end{aligned}$$

$$\begin{aligned}
\mathcal{L}_{\text{entire}}[(f_1) \text{ or } (f_2)]_{\alpha} = & \\
(1) & \mathcal{L}_{\text{entire}}[f_1]_{\alpha} \cup \mathcal{L}_{\text{entire}}[f_2]_{\alpha}, \text{ if } \tau[f_1]_{\alpha} = \tau[f_2]_{\alpha} = \text{T}. \\
(2) & \mathcal{L}_{\text{entire}}[f_1]_{\alpha} \otimes \mathcal{L}_{\text{entire}}[f_2]_{\alpha}, \text{ if } \tau[f_1]_{\alpha} = \tau[f_2]_{\alpha} = \text{F}. \\
(3) & \mathcal{L}_{\text{entire}}[f_1]_{\alpha}, \text{ if } \tau[f_1]_{\alpha} = \text{T}, \tau[f_2]_{\alpha} = \text{F}. \\
(4) & \mathcal{L}_{\text{entire}}[f_2]_{\alpha}, \text{ if } \tau[f_1]_{\alpha} = \text{F}, \tau[f_2]_{\alpha} = \text{T}.
\end{aligned}$$

$$\begin{aligned}
\mathcal{L}_{\text{entire}}[(f_1) \text{ implies } (f_2)]_{\alpha} = & \\
(1) & \text{FlipSet}(\mathcal{L}_{\text{entire}}[f_1]_{\alpha}) \otimes \mathcal{L}_{\text{entire}}[f_2]_{\alpha}, \text{ if } \tau[f_1]_{\alpha} = \text{T}, \tau[f_2]_{\alpha} = \text{F}. \\
(2) & \text{FlipSet}(\mathcal{L}_{\text{entire}}[f_1]_{\alpha}) \cup \mathcal{L}_{\text{entire}}[f_2]_{\alpha}, \text{ if } \tau[f_1]_{\alpha} = \text{F}, \tau[f_2]_{\alpha} = \text{T}. \\
(3) & \mathcal{L}_{\text{entire}}[f_2]_{\alpha}, \text{ if } \tau[f_1]_{\alpha} = \tau[f_2]_{\alpha} = \text{T}. \\
(4) & \text{FlipSet}(\mathcal{L}_{\text{entire}}[f_1]_{\alpha}), \text{ if } \tau[f_1]_{\alpha} = \tau[f_2]_{\alpha} = \text{F}.
\end{aligned}$$

Fig. 11. INFUSE's entire link generation semantics for and, or, and implies formulas.

2) *and, or, and implies formulas, i.e.,  $(f_1)$  and/or/implies  $(f_2)$* : For ease of understanding, we take the and formula as an example to explain principles for its link generation.

- if both  $f_1$  and  $f_2$  are evaluated to true, then they together decide the satisfaction of the formula, thus, the  $\otimes$  operator

$$\begin{aligned}
\mathcal{L}_{\text{partial}}[(f_1) \text{ and } (f_2)]_{\alpha} = & \\
(1) & \mathcal{L}_0[(f_1) \text{ and } (f_2)]_{\alpha}, \text{ if } \text{Affected}(f_1) = \text{Affected}(f_2) = \text{F}. \\
(2) & \text{a. } \mathcal{L}_{\text{partial}}[f_1]_{\alpha} \otimes \mathcal{L}_0[f_2]_{\alpha}, \text{ if } \tau[f_1]_{\alpha} = \tau[f_2]_{\alpha} = \text{T}. \\
& \text{b. } \mathcal{L}_{\text{partial}}[f_1]_{\alpha} \cup \mathcal{L}_0[f_2]_{\alpha}, \text{ if } \tau[f_1]_{\alpha} = \tau[f_2]_{\alpha} = \text{F}. \\
& \text{c. } \mathcal{L}_0[f_2]_{\alpha}, \text{ if } \tau[f_1]_{\alpha} = \text{T}, \tau[f_2]_{\alpha} = \text{F}. \\
& \text{d. } \mathcal{L}_{\text{partial}}[f_1]_{\alpha}, \text{ if } \tau[f_1]_{\alpha} = \text{F}, \tau[f_2]_{\alpha} = \text{T}. \\
& \text{if } \text{Affected}(f_1) = \text{T}, \text{Affected}(f_2) = \text{F}. \\
(3) & \text{a. } \mathcal{L}_0[f_1]_{\alpha} \otimes \mathcal{L}_{\text{partial}}[f_2]_{\alpha}, \text{ if } \tau[f_1]_{\alpha} = \tau[f_2]_{\alpha} = \text{T}. \\
& \text{b. } \mathcal{L}_0[f_1]_{\alpha} \cup \mathcal{L}_{\text{partial}}[f_2]_{\alpha}, \text{ if } \tau[f_1]_{\alpha} = \tau[f_2]_{\alpha} = \text{F}. \\
& \text{c. } \mathcal{L}_{\text{partial}}[f_2]_{\alpha}, \text{ if } \tau[f_1]_{\alpha} = \text{T}, \tau[f_2]_{\alpha} = \text{F}. \\
& \text{d. } \mathcal{L}_0[f_1]_{\alpha}, \text{ if } \tau[f_1]_{\alpha} = \text{F}, \tau[f_2]_{\alpha} = \text{T}. \\
& \text{if } \text{Affected}(f_1) = \text{F}, \text{Affected}(f_2) = \text{T}. \\
(4) & \text{a. } \mathcal{L}_{\text{partial}}[f_1]_{\alpha} \otimes \mathcal{L}_{\text{partial}}[f_2]_{\alpha}, \text{ if } \tau[f_1]_{\alpha} = \tau[f_2]_{\alpha} = \text{T}. \\
& \text{b. } \mathcal{L}_{\text{partial}}[f_1]_{\alpha} \cup \mathcal{L}_{\text{partial}}[f_2]_{\alpha}, \text{ if } \tau[f_1]_{\alpha} = \tau[f_2]_{\alpha} = \text{F}. \\
& \text{c. } \mathcal{L}_{\text{partial}}[f_2]_{\alpha}, \text{ if } \tau[f_1]_{\alpha} = \text{T}, \tau[f_2]_{\alpha} = \text{F}. \\
& \text{d. } \mathcal{L}_{\text{partial}}[f_1]_{\alpha}, \text{ if } \tau[f_1]_{\alpha} = \text{F}, \tau[f_2]_{\alpha} = \text{T}. \\
& \text{if } \text{Affected}(f_1) = \text{Affected}(f_2) = \text{T}.
\end{aligned}$$

Fig. 12. INFUSE's partial link generation semantics for the and formula

is conducted to generate links that explain how the and formula is satisfied.

- if both  $f_1$  and  $f_2$  are evaluated to false, then either of them decides the violation of the formula, thus, the union of links from  $f_1$  and  $f_2$  explains how the and formula is violated.
- if one subformula is evaluated to true and the other is evaluated to false, then the latter decides the violation of the formula, thus, links coming from the latter can explain how the and formula is violated.

Principles for or and implies formulas are similar, which incur INFUSE's entire link generation semantics for the three formulas as shown in Fig. 11.

Similar to INFUSE's truth value evaluation semantics for the three formulas, INFUSE conducts incremental generation according to the **Affected** function on subformulas  $f_1$  and  $f_2$ .

$$\begin{aligned}
\mathcal{L}_{\text{partial}}[(f_1) \text{ or } (f_2)]_{\alpha} = & \\
(1) \mathcal{L}_0[(f_1) \text{ or } (f_2)]_{\alpha}, & \text{ if } \text{Affected}(f_1) = \text{Affected}(f_2) = \text{F}. \\
(2) \text{a. } \mathcal{L}_{\text{partial}}[f_1]_{\alpha} \cup \mathcal{L}_0[f_2]_{\alpha}, & \text{ if } \tau[f_1]_{\alpha} = \tau[f_2]_{\alpha} = \text{T}. \\
\text{b. } \mathcal{L}_{\text{partial}}[f_1]_{\alpha} \otimes \mathcal{L}_0[f_2]_{\alpha}, & \text{ if } \tau[f_1]_{\alpha} = \tau[f_2]_{\alpha} = \text{F}. \\
\text{c. } \mathcal{L}_{\text{partial}}[f_1]_{\alpha}, & \text{ if } \tau[f_1]_{\alpha} = \text{T}, \tau[f_2]_{\alpha} = \text{F}. \\
\text{d. } \mathcal{L}_0[f_2]_{\alpha}, & \text{ if } \tau[f_1]_{\alpha} = \text{F}, \tau[f_2]_{\alpha} = \text{T}. \\
\text{if } \text{Affected}(f_1) = \text{T}, \text{Affected}(f_2) = \text{F}. & \\
(3) \text{a. } \mathcal{L}_0[f_1]_{\alpha} \cup \mathcal{L}_{\text{partial}}[f_2]_{\alpha}, & \text{ if } \tau[f_1]_{\alpha} = \tau[f_2]_{\alpha} = \text{T}. \\
\text{b. } \mathcal{L}_0[f_1]_{\alpha} \otimes \mathcal{L}_{\text{partial}}[f_2]_{\alpha}, & \text{ if } \tau[f_1]_{\alpha} = \tau[f_2]_{\alpha} = \text{F}. \\
\text{c. } \mathcal{L}_0[f_1]_{\alpha}, & \text{ if } \tau[f_1]_{\alpha} = \text{T}, \tau[f_2]_{\alpha} = \text{F}. \\
\text{d. } \mathcal{L}_{\text{partial}}[f_2]_{\alpha}, & \text{ if } \tau[f_1]_{\alpha} = \text{F}, \tau[f_2]_{\alpha} = \text{T}. \\
\text{if } \text{Affected}(f_1) = \text{F}, \text{Affected}(f_2) = \text{T}. & \\
(4) \text{a. } \mathcal{L}_{\text{partial}}[f_1]_{\alpha} \cup \mathcal{L}_{\text{partial}}[f_2]_{\alpha}, & \text{ if } \tau[f_1]_{\alpha} = \tau[f_2]_{\alpha} = \text{T}. \\
\text{b. } \mathcal{L}_{\text{partial}}[f_1]_{\alpha} \otimes \mathcal{L}_{\text{partial}}[f_2]_{\alpha}, & \text{ if } \tau[f_1]_{\alpha} = \tau[f_2]_{\alpha} = \text{F}. \\
\text{c. } \mathcal{L}_{\text{partial}}[f_1]_{\alpha}, & \text{ if } \tau[f_1]_{\alpha} = \text{T}, \tau[f_2]_{\alpha} = \text{F}. \\
\text{d. } \mathcal{L}_{\text{partial}}[f_2]_{\alpha}, & \text{ if } \tau[f_1]_{\alpha} = \text{F}, \tau[f_2]_{\alpha} = \text{T}. \\
\text{if } \text{Affected}(f_1) = \text{Affected}(f_2) = \text{T}. &
\end{aligned}$$

Fig. 13. INFUSE's partial link generation semantics for the or formula

$$\begin{aligned}
\mathcal{L}_{\text{partial}}[(f_1) \text{ implies } (f_2)]_{\alpha} = & \\
(1) \mathcal{L}_0[(f_1) \text{ implies } (f_2)]_{\alpha}, & \text{ if } \text{affected}(f_1) = \text{affected}(f_2) = \text{F}. \\
(2) \text{a. } \text{FlipSet}(\mathcal{L}_{\text{partial}}[f_1]_{\alpha}) \otimes \mathcal{L}_0[f_2]_{\alpha}, & \text{ if } \tau[f_1]_{\alpha} = \text{T}, \tau[f_2]_{\alpha} = \text{F}. \\
\text{b. } \text{FlipSet}(\mathcal{L}_{\text{partial}}[f_1]_{\alpha}) \cup \mathcal{L}_0[f_2]_{\alpha}, & \text{ if } \tau[f_1]_{\alpha} = \text{F}, \tau[f_2]_{\alpha} = \text{T}. \\
\text{c. } \mathcal{L}_0[f_2]_{\alpha}, & \text{ if } \tau[f_1]_{\alpha} = \tau[f_2]_{\alpha} = \text{T}. \\
\text{d. } \text{FlipSet}(\mathcal{L}_{\text{partial}}[f_1]_{\alpha}), & \text{ if } \tau[f_1]_{\alpha} = \tau[f_2]_{\alpha} = \text{F}. \\
\text{if } \text{Affected}(f_1) = \text{T}, \text{Affected}(f_2) = \text{F}. & \\
(3) \text{a. } \text{FlipSet}(\mathcal{L}_0[f_1]_{\alpha}) \otimes \mathcal{L}_{\text{partial}}[f_2]_{\alpha}, & \text{ if } \tau[f_1]_{\alpha} = \text{T}, \tau[f_2]_{\alpha} = \text{F}. \\
\text{b. } \text{FlipSet}(\mathcal{L}_0[f_1]_{\alpha}) \cup \mathcal{L}_{\text{partial}}[f_2]_{\alpha}, & \text{ if } \tau[f_1]_{\alpha} = \text{F}, \tau[f_2]_{\alpha} = \text{T}. \\
\text{c. } \mathcal{L}_{\text{partial}}[f_2]_{\alpha}, & \text{ if } \tau[f_1]_{\alpha} = \tau[f_2]_{\alpha} = \text{T}. \\
\text{d. } \text{FlipSet}(\mathcal{L}_0[f_1]_{\alpha}), & \text{ if } \tau[f_1]_{\alpha} = \tau[f_2]_{\alpha} = \text{F}. \\
\text{if } \text{Affected}(f_1) = \text{F}, \text{Affected}(f_2) = \text{T}. & \\
(4) \text{a. } \text{FlipSet}(\mathcal{L}_{\text{partial}}[f_1]_{\alpha}) \otimes \mathcal{L}_{\text{partial}}[f_2]_{\alpha}, & \\
\text{if } \tau[f_1]_{\alpha} = \text{T}, \tau[f_2]_{\alpha} = \text{F}. & \\
\text{b. } \text{FlipSet}(\mathcal{L}_{\text{partial}}[f_1]_{\alpha}) \cup \mathcal{L}_{\text{partial}}[f_2]_{\alpha}, & \\
\text{if } \tau[f_1]_{\alpha} = \text{F}, \tau[f_2]_{\alpha} = \text{T}. & \\
\text{c. } \mathcal{L}_{\text{partial}}[f_2]_{\alpha}, & \text{ if } \tau[f_1]_{\alpha} = \tau[f_2]_{\alpha} = \text{T}. \\
\text{d. } \text{FlipSet}(\mathcal{L}_{\text{partial}}[f_1]_{\alpha}), & \text{ if } \tau[f_1]_{\alpha} = \tau[f_2]_{\alpha} = \text{F}. \\
\text{if } \text{Affected}(f_1) = \text{Affected}(f_2) = \text{T}. &
\end{aligned}$$

Fig. 14. INFUSE's partial link generation semantics for the implies formula

INFUSE's partial link generation semantics for and, or, and implies formulas are given in Fig. 12, Fig. 13, and Fig. 14 respectively.

$$\begin{aligned}
\mathcal{L}_{\text{entire}}[\text{not } (f)]_{\alpha} &= \text{FlipSet}(\mathcal{L}_{\text{entire}}[f]_{\alpha}). \\
\mathcal{L}_{\text{entire}}[bfunc(\gamma_1, \dots, \gamma_n)]_{\alpha} &= \emptyset.
\end{aligned}$$

Fig. 15. INFUSE's entire link generation semantics for not and bfunc formulas

3) *not* and *bfunc* formulas, i.e., *not* (f) and *bfunc*( $v_1, \dots, v_n$ ): Fig. 15 gives INFUSE's entire link generation semantics for not and bfunc formulas. For not formula, it inverts the linkType of links coming from its

$$\begin{aligned}
\mathcal{L}_{\text{partial}}[\text{not } (f)]_{\alpha} &= \\
(1) \mathcal{L}_0[\text{not } (f)]_{\alpha}, & \text{ if } \text{Affected}(f) = \text{F}. \\
(2) \text{FlipSet}(\mathcal{L}_{\text{partial}}[f]_{\alpha}), & \text{ if } \text{Affected}(f) = \text{T}. \\
\mathcal{L}_{\text{partial}}[bfunc(\gamma_1, \dots, \gamma_n)]_{\alpha} &= \emptyset.
\end{aligned}$$

Fig. 16. INFUSE's partial link generation semantics for not and bfunc formulas

subformula *f*. For *bfunc* formula, it always generates an empty link since the links that contain variables in the *bfunc* formula would be generated at where these variables are defined (i.e., universal and existential formulas). Fig. 16 gives INFUSE's partial link generation semantics for the two formulas. For the not formula, the Affected function on its subformula *f* decides the reusability of its previous links. The *bfunc* formula still generates an empty link.

## II. THEOREMS AND PROOFS

**Theorem 1 (WHAT-Correctness).** *Given any consistency constraint and associated context pool, INFUSE produces the same result for its arranged valid context changes, no matter it checks these changes as a whole or individually.*

*Proof.* Let the concerned constraint be *s* with the associated context pool  $P_0$ . INFUSE's arranged valid context changes compose a constraint checking task  $T = (chg_1, \dots, chg_n)$ .  $P_i$  represents the context pool right after applying context change  $chg_i$ . As discussed in Section II, in order to prove this WHAT-Correctness theorem, we actually aim to prove:

$$\text{chk}(P_0, T, s) = \bigcup_{i=1}^n \text{chk}(P_{i-1}, (chg_i), s) \quad (1)$$

Since  $\text{chk}(P_0, T, s)$  and  $\text{chk}(P_{n-1}, (chg_n), s)$  both represent the inconsistency result when checking contexts in the final context pool  $P_n$  against *s*, we can easily know that  $\text{chk}(P_0, T, s) = \text{chk}(P_{n-1}, (chg_n), s)$ . Therefore, to get Equation (1), we only need to prove:

$$\bigcup_{i=1}^{n-1} \text{chk}(P_{i-1}, (chg_i), s) \subseteq \text{chk}(P_0, T, s) \quad (2)$$

We use reduction to absurdity by assuming that Equation (2) does not hold. That is, there is an  $inc_x$  satisfying:

$$inc_x \in \left( \bigcup_{i=1}^{n-1} \text{chk}(P_{i-1}, (chg_i), s) - \text{chk}(P_0, T, s) \right) \quad (3)$$

Suppose  $inc_x$  is first exposed by  $chg_j$  ( $1 \leq j < n$ ), i.e.,  $inc_x \in \text{chk}(P_{j-1}, (chg_j), s)$  and  $inc_x \notin \text{chk}(P_{j-2}, (chg_{j-1}), s)$ . Due to our definition of E/H/I-changes,  $chg_j$  is an E-change. Moreover, since  $inc_x \notin \text{chk}(P_{n-1}, (chg_n), s)$ , which is equal to  $\text{chk}(P_0, T, s)$ , it should be hidden no later than  $chg_n$  is applied and checked. Suppose  $inc_x$  is actually hidden by  $chg_k$  ( $j < k \leq n$ ), i.e.,

$inc_x \notin \text{chk}(P_{k-1}, (chg_k), s)$ . By definition,  $chg_k$  must be an H-change. Therefore, we can derive that:

$$inc_x \in \text{chk}(P_{j-1}, (chg_j), s), \quad (4)$$

$$inc_x \notin \text{chk}(P_{k-1}, (chg_k), s). \quad (5)$$

This actually denotes that  $inc_x$  was first exposed by an E-change  $chg_j$ , and then hidden by a H-change  $chg_k$ , which clearly violates the nonexistence of an ordered E-change and H-change in any constraint checking task according to the validity criterion (Definition 4). Therefore, this leads to a contradiction to our assumption, so Equation (2) holds and thus Equation (1) can be easily proved as such. This completes our proof.  $\square$

**Theorem 2 (HOW-Correctness).** *Given any consistency constraint and associated context pool, INFUSE produces the same result by its check fusion semantics, as existing constraint checking techniques do.*

*Proof.* Since the semantic structures of true value evaluation and link generation are highly consistent, we only give our proof when it comes to the truth value semantics. We here prove INFUSE's checking correctness of truth value evaluation semantics for all seven formulas in detail.

**Universal formula.** We would rely on the checking correctness of ECC, Con-C, and PCC, thus, we explain their truth value evaluation semantics for universal formula briefly here.

Let the universal formula be  $\forall v \in C(f)$  and  $C$  contains  $m$  elements  $(e_1, \dots, e_m)$  after applying a context change  $chg$ . The truth value  $\tau$  of the universal formula is defined as the conjunction of truth values  $(t_1, \dots, t_m)$  of subformula  $f$  for all elements in  $C$ . ECC evaluates each  $t_i$  in a sequential manner while Con-C evaluates each  $t_i$  concurrently. PCC considers the effect of  $chg$ , which can be split into four cases: (a) if  $chg$  did not affect the formula at all, each  $t_i$  would remain unchanged, as well as  $\tau$ . (b) if  $chg$  added the element  $e_m$  into  $C$ ,  $t_1, \dots, t_{m-1}$  would remain unchanged, thus,  $\tau$  would be the conjunction of its last value and  $t_m$  associated with  $e_m$ . (c) if  $chg$  deleted the element  $e_{m+1}$  from  $C$ ,  $t_1, \dots, t_m$  would remain unchanged, thus,  $\tau$  would be the conjunction of them. (d) if  $chg$  affected another context related to  $f$ , then all  $t_i$  would need to be reevaluated partially in a similar manner.

We now analyze INFUSE's truth value evaluation semantics for universal formula to prove its correctness. Firstly, the correctness of the entire semantics as shown in Fig. 7 in the main content is similarly guaranteed by the correctness of ECC's semantics due to their similarity. Secondly, Con-C's correctness confirms that evaluating truth values concurrently for independent elements can get the same results as evaluating serially, which guarantees the correctness of  $\text{eval}_{\text{entire}}$  and  $\text{eval}_{\text{partial}}$ . Therefore, we only specifically analyze the correctness concerning cases of the partial semantics in Fig. 6 in the main content:

- Case (1) is exactly the same as case (a) in PCC since it only focuses on whether the whole formula is affected.

- Case (2) extends the idea of case (b) in PCC to multiple context changes. These context changes only added elements  $(y_1, \dots, y_a)$  in  $C$ , therefore, the last truth value  $(\tau_0)$  is reusable according to case (b) in PCC. The correctness of new truth values  $(t_1, \dots, t_a)$  associated with new elements are guaranteed by  $\text{eval}_{\text{entire}}$ .
- Case (3) fuses the idea of case (b) and case (c) in PCC and extends to multiple context changes. Truth values associated with elements that were not deleted or updated by forthcoming context changes are reusable according to case (c) in PCC. The correctness of new truth values  $(t_1, \dots, t_{a+u})$  associated with new or updated elements are also guaranteed by  $\text{eval}_{\text{entire}}$ .
- Case (4) is exactly the same as case (d) in PCC, since it only focuses on whether subformula  $f$  is affected when  $C$  is not affected.
- Case (5) fuses the idea of case (b), case (c), and case (d) in PCC and extends to multiple context changes. The correctness of truth values  $(t_1, \dots, t_{a+u})$  associated with new elements or updated elements are guaranteed by  $\text{eval}_{\text{entire}}$ . Truth values  $(t_{a+u+1}, \dots, t_n)$  associated with elements that were not deleted or updated should be reevaluated partially since subformula  $f$  is affected according to case (d) in PCC. their correctness are guaranteed by  $\text{eval}_{\text{partial}}$ .

**Existential formula.** Since INFUSE's truth value evaluation semantics for the existential formula is quite similar to that for the universal formula, the correctness of INFUSE's truth value evaluation semantics for the existential formula can be proved follow the same procedure, i.e., the correctness of the entire semantics can be guaranteed by ECC's correctness, Con-C's correctness supports the  $\text{eval}_{\text{entire}}$  and  $\text{eval}_{\text{partial}}$  functions, and the partial semantics can be analyzed similarly.

**and, or, and implies formulas.** The correctness of the entire semantics for and formula is trivial since it evaluates the truth value based on the logic of the formula. As for the partial semantics, every and formula has two subformulas, each of which could be affected by INFUSE's arranged valid context changes. Therefore, INFUSE partitions all situation into four cases as shown in Fig. 5. Besides, or and implies formulas can be proved in the same way.

**not formula.** The entire semantics for not formula is straightforward. The partial semantics contain two cases since the subformula of not formula is either affected or not affected.

**bfunc formula.** *bfunc* formula returns its result as we expect in the entire semantics and its last truth value is always reusable since it neither owns any subformula nor references any context.

Therefore, the correctness of truth value evaluation semantics for all seven formulas are proved, i.e., INFUSE can achieve the same truth values as existing checking techniques. Moreover, The correctness of link generation semantics can be proved similarly, incurring INFUSE can achieve the same links as existing checking techniques. As a summary, INFUSE can achieve the same inconsistency checking results as existing checking techniques. This completes our proof.  $\square$