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

Anonymous author(s)

I. CHECKING SEMANTICS OF INFUSE

To introduce full semantics of INFUSE, we first define some necessary functions and operators:

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,

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

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

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

- Type(l) = l.linkType.
- Assignments(l) = l.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.,

- Concatenate(l_1, l_2) = (Type(l_1), Assignments(l_1) $\cup$ Assignments(l_2)).
- $S_1 \otimes S_2$ = {Concatenate(l_1, l_2) | $l_1 \in S_1 \wedge l_2 \in S_2$}, 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. 4 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 $\mathsf{eval}_{\mathsf{entire}}$ and $\mathsf{eval}_{\mathsf{partial}}$ functions (shown in Fig. 2) to calculate truth values of subformula f concerning different elements.

$$\tau_{\mathsf{entire}}[\exists v \in C(f)]_\alpha = \mathsf{F} \vee \tau_{\mathsf{entire}}[f]_{\mathsf{bind}((v,x_1),\alpha)} \vee \cdots \vee \tau_{\mathsf{entire}}[f]_{\mathsf{bind}((v,x_n),\alpha)} | x_i \in C$$

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

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

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

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

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

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

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

Fig. 3. INFUSE's entire truth value evaluation semantics for and, or, and implies formulas.

2) and, or, and implies formulas, i.e., (f_1) and/or/implies (f_2): Fig. 3 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.

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

Fig. 4. INFUSE's partial truth value evaluation semantics for the existential formula.

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

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

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

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 $bfunc(v_1, \cdots, v_n)$: 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.

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

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

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

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

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

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

B. Link Generation

In the following, we list INFUSE's link generation semantics for the rest of 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. 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 $\mathsf{gen}_{\mathsf{entire}}$ and $\mathsf{gen}_{\mathsf{partial}}$ functions (shown in Fig. 9) to generate links of subformula f concerning different elements.

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

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

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

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

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

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

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

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

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 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.

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

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

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

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

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

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

3) not and $bfunc$ formulas, i.e, not (f) and $bfunc(v_1, \cdots, v_n)$: 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 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

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

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

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

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

subformula f decides the reusability of its previous links. The $bfunc$ formula still generates an empty link.

$$\mathcal{L}_{\textsf{entire}}[\textsf{not}\ (f)]_\alpha = \textsf{FlipSet}(\mathcal{L}_{\textsf{entire}}[f]_\alpha).$$

$$\mathcal{L}_{\textsf{entire}}[bfunc(\gamma_1, \cdots, \gamma_n)]_\alpha = \emptyset.$$

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

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.*

$\mathcal{L}_{\textsf{partial}}[\textsf{not}\ (f)]_\alpha =$
$(1)\mathcal{L}_0[\textsf{not}\ (f)]_\alpha,$ if $\textsf{Affected}(f) = \textsf{F}.$
$(2)\textsf{FlipSet}(\mathcal{L}_{\textsf{partial}}[f]_\alpha),$ if $\textsf{Affected}(f) = \textsf{T}.$

$$\mathcal{L}_{\textsf{partial}}[bfunc(\gamma_1, \cdots, \gamma_n)]_\alpha = \emptyset.$$

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

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 $T = \{chg_1, \cdots, chg_n\}$. 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:

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

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

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

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

$$inc_x \in (\bigcup_{i=1}^{n-1} \textsf{chk}(P_{i-1}, \{chg_i\}, s) - \textsf{chk}(P_0, T, s)) \tag{3}$$

Suppose inc_x is first exposed by chg_j ($1 \leq j < n$), i.e., $inc_x \in \textsf{chk}(P_{j-1}, \{chg_j\}, s)$ and $inc_x \notin \textsf{chk}(P_{j-2}, \{chg_{j-1}\}, s)$. Due to our definition of E/H/I-changes, chg_j is an E-change. Moreover, since $inc_x \notin \textsf{chk}(P_{n-1}, \{chg_n\}, s)$, which is equal to $\textsf{chk}(P_0, T, s)$, 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., $inc_x \notin \textsf{chk}(P_{k-1}, \{chg_k\}, s)$. By definition, chg_k must be an H-change. Therefore, we can derive that:

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

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

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 here. □

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 select four formulas (i.e., universal, and, not, $bfunc$ formulas) to form the *kernel* and other three (i.e., existential, or, and implies) can be expressed via the four kernel ones. We here prove INFUSE's checking correctness of truth value evaluation semantics for the four kernel 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 $(e_1, \cdots, e_m)$ after applying a context change chg. The truth value τ of the universal formula is defined as the conjunction of truth values $(t_1, \cdots, t_m)$ 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 τ. (b) if chg added the element e_m into C, $t_1, \cdots, t_{m-1}$ would remain unchanged, thus, τ would be the conjunction of its last value and t_m asscociated with e_m. (c) if chg deleted the element e_{m+1} from C, $t_1, \cdots, t_m$ would remain unchanged, thus, τ 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 the truth value evaluation semantics of INFUSE for universal formula to prove its correctness. Firstly, the correctness of 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 $(y_1, \cdots, y_a)$ in C, therefore, the last truth value (τ_0) is reusable according to case (b) in PCC. The correctness of new truth values $(t_1, \cdots, t_a)$ 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 $(t_1, \cdots, t_{a+u})$ 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 $(t_1, \cdots, t_{a+u})$ associated with new elements or updated elements are guaranteed by $\text{eval}_{\text{entire}}$. Truth values $(t_{a+u+1}, \cdots, t_n)$ 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 $\text{eval}_{\text{partial}}$.

and formula. The correctness of entire semantics for and formula is trivial since it evaluates the truth value based on the logic of the formula. As for 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.

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 has subformula nor references a context.

Therefore, the correctness of truth value evaluation semantics for the kernel formulas are proved, i.e., INFUSE can achieve the same truth values as existing checking techniques for the kernel formulas. Since other three formulas can be expressed by the kernel formulas, INFUSE's truth value evaluation semantics for other three formulas are also correct. Moreover, The correctness of link generation semantics can be proved similarly, incurring INFUSE can achieve the same links as existing checking techniques. In conclusion, INFUSE can achieve the same inconsistency checking results as existing checking techniques. □