Online Appendix to: Partial Constraint Checking for Context Consistency in Pervasive Computing

CHANG XU and S. C. CHEUNG
The Hong Kong University of Science and Technology
W. K. CHAN
City University of Hong Kong
and
CHUNYANG YE
The Hong Kong University of Science and Technology
and Chinese Academy of Sciences

APPENDIX A. THEOREM 1 AND THE PROOF

We present and prove Theorem 1 in this appendix.

THEOREM 1. *The following gives truth value evaluation semantics for existential,* or*, and* implies *formulas in* PCC.

1. $\mathcal{T}\,[\exists\gamma \text{ in } S\,(f)]_\alpha =$
   (1) $\mathcal{T}_0[\exists\gamma \text{ in } S\,(f)]_\alpha$,
   if S has no change and Affected$(f) = \bot$;
   (2) $\mathcal{T}_0[\exists\gamma \text{ in } S\,(f)]_\alpha \vee \mathcal{T}\,[f]_{\mathsf{bind}((\gamma,x),\alpha)} |\ \{x\} = S - S_0$,
   if S has a context addition change;
   (3) $\bot \vee \mathcal{T}_0[f]_{\mathsf{bind}((\gamma,x_n),\alpha)} \vee \cdots \vee \mathcal{T}_0[f]_{\mathsf{bind}((\gamma,x_n),\alpha)} |\ x_i \in S$,
   if S has a context deletion change;
   (4) $\bot \vee \mathcal{T}\,[f]_{\mathsf{bind}((\gamma,x_1),\alpha)} \vee \cdots \vee \mathcal{T}\,[f]_{\mathsf{bind}((\gamma,x_n),\alpha)} | x_i \in S$,
   if Affected$(f) = \top$.
2. $\mathcal{T}\,[(f_1) \text{ or } (f_2)\,]_\alpha =$
   (1) $\mathcal{T}_0[(f_1) \text{ or } (f_2)\,]_\alpha$, if Affected$(f_1) =$ Affected$(f_2) = \bot$;
   (2) $\mathcal{T}\,[f_1]_\alpha \vee \mathcal{T}_0[f_2]_\alpha$, if Affected$(f_1) = \top$;
   (3) $\mathcal{T}_0[f_1]_\alpha \vee \mathcal{T}\,[f_2]_\alpha$, if Affected$(f_2) = \top$.
3. $\mathcal{T}\,[(f_1) \text{ implies } (f_2)\,]_\alpha =$
   (1) $\mathcal{T}_0[(f_1) \text{ implies } (f_2)\,]_\alpha$, if Affected$(f_1) =$ Affected$(f_2) = \bot$;
   (2) $\mathcal{T}\,[f_1]_\alpha \to \mathcal{T}_0[f_2]_\alpha$, if Affected$(f_1) = \top$;
   (3) $\mathcal{T}_0[f_1]_\alpha \to \mathcal{T}\,[f_2]_\alpha$, if Affected$(f_2) = \top$.

DOI 10.1145/1656250.1656253 http://doi.acm.org/10.1145/1656250.1656253

The truth value evaluation semantics for the existential formula are similar to those for the universal formula in PCC. The semantics follow the same four cases that identify different degrees for reusing the last evaluation results. The truth value evaluation semantics for or and implies formulas are straightforward, similar to those for the and formula in PCC. We divide all situations into the same three cases, where a context change affects at most one of the two subformulas according to our checking strategy.

In the following, we give the proof of Theorem 1 by focusing on the existential formula. The semantics of or and implies formulas can be proved similarly.

PROOF OF THEOREM 1 (PART 1).

1. $\mathcal{T}\,[\exists\gamma \text{ in } S\ (f)]_\alpha$

$=\quad \mathcal{T}\,[\mathsf{not}(\forall\gamma \text{ in } S\ (\mathsf{not}(f)))\,]_\alpha$

$=\quad (1)\ \mathcal{T}_0[\mathsf{not}(\forall\gamma \text{ in } S\ (\mathsf{not}(f)))\,]_\alpha, \qquad \text{if } \mathsf{Affected}(\forall\gamma \text{ in } S\ (\mathsf{not}(f))) = \bot;$
$\quad\ (2)\ \neg\,\mathcal{T}\,[\forall\gamma \text{ in } S\ (\mathsf{not}(f))\,]_\alpha, \qquad \text{if } \mathsf{Affected}(\forall\gamma \text{ in } S\ (\mathsf{not}(f))) = \top.$

$=\quad (1)\ \mathcal{T}_0[\mathsf{not}(\forall\gamma \text{ in } S\ (\mathsf{not}(f)))\,]_\alpha,$
$\qquad\quad$ if S has no change and $\mathsf{Affected}(f) = \bot$;
$\qquad (2a)\ \neg\,(\mathcal{T}_0[\forall\gamma \text{ in } S\ (\mathsf{not}(f))\,]_\alpha \wedge \mathcal{T}\,[\mathsf{not}(f)]_{\mathsf{bind}((\gamma,x),\alpha)}) \mid \{x\} = S - S_0,$
$\qquad\quad$ if S has a context addition change;
$\qquad (2b)\ \neg\,(\top \wedge \mathcal{T}_0[\mathsf{not}(f)]_{\mathsf{bind}((\gamma,x_1),\alpha)} \wedge \cdots \wedge \mathcal{T}_0\,[\mathsf{not}(f)]_{\mathsf{bind}((\gamma,x_n),\alpha)}) \mid x_i \in S,$
$\qquad\quad$ if S has a context deletion change;
$\qquad (2c)\ \neg\,(\top \wedge \mathcal{T}\,[\mathsf{not}(f)]_{\mathsf{bind}((\gamma,x_1),\alpha)} \wedge \cdots \wedge \mathcal{T}\,[\mathsf{not}(f)]_{\mathsf{bind}((\gamma,x_n),\alpha)}) \mid x_i \in S,$
$\qquad\quad$ if $\mathsf{Affected}(f) = \top$.

$=\quad (1)\ \mathcal{T}_0[\exists\gamma \text{ in } S\ (f)]_\alpha,$
$\qquad\quad$ if S has no change and $\mathsf{Affected}(f) = \bot$;
$\qquad (2a)\ \neg\,(\mathcal{T}_0[\forall\gamma \text{ in } S\ (\mathsf{not}(f))\,]_\alpha \wedge \neg\,\mathcal{T}\,[f]_{\mathsf{bind}((\gamma,x),\alpha)}) \mid \{x\} = S - S_0,$
$\qquad\quad$ if S has a context addition change;
$\qquad (2b)\ \neg\,(\top \wedge \neg\,\mathcal{T}_0[f]_{\mathsf{bind}((\gamma,x_1),\alpha)} \wedge \cdots \wedge \neg\,\mathcal{T}_0[f]_{\mathsf{bind}((\gamma,x_n),\alpha)}) \mid x_i \in S,$
$\qquad\quad$ if S has a context deletion change;
$\qquad (2c)\ \neg\,(\top \wedge \neg\,\mathcal{T}\,[f]_{\mathsf{bind}((\gamma,x_1),\alpha)} \wedge \cdots \wedge \neg\,\mathcal{T}[f]_{\mathsf{bind}((\gamma,x_n),\alpha)}) \mid x_i \in S,$
$\qquad\quad$ if $\mathsf{Affected}(f) = \top$.

$=\quad (1)\ \mathcal{T}_0[\exists\gamma \text{ in } S\ (f)]_\alpha,$
$\qquad\quad$ if S has no change and $\mathsf{Affected}(f) = \bot$;
$\qquad (2a)\ \mathcal{T}_0[\mathsf{not}(\forall\gamma \text{ in } S\ (\mathsf{not}(f)))\,]_\alpha \vee \mathcal{T}\,[f]_{\mathsf{bind}((\gamma,x),\alpha)} \mid \{x\} = S - S_0,$
$\qquad\quad$ if S has a context addition change;
$\qquad (2b)\ \bot \vee \mathcal{T}_0[f]_{\mathsf{bind}((\gamma,x_1),\alpha)} \vee \cdots \vee \mathcal{T}_0[f]_{\mathsf{bind}((\gamma,x_n),\alpha)} \mid x_i \in S,$
$\qquad\quad$ if S has a context deletion change;
$\qquad (2c)\ \bot \vee \mathcal{T}\,[f]_{\mathsf{bind}((\gamma,x_1),\alpha)} \vee \cdots \vee \mathcal{T}\,[f]_{\mathsf{bind}((\gamma,x_n),\alpha)} \mid x_i \in S,$
$\qquad\quad$ if $\mathsf{Affected}(f) = \top$

$=\quad (1)\ \mathcal{T}_0[\exists\gamma \text{ in } S\ (f)]_\alpha,$
$\qquad\quad$ if S has no change and $\mathsf{Affected}(f) = \bot$;
$\qquad (2a)\ \mathcal{T}_0[\exists\gamma \text{ in } S\ (f)]_\alpha \vee \mathcal{T}\,[f]_{\mathsf{bind}((\gamma,x),\alpha)} \mid \{x\} = S - S_0,$
$\qquad\quad$ if S has a context addition change;

(2b) $\bot \vee \mathcal{T}_0[f]_{\mathsf{bind}((\gamma,x_1),\alpha)} \vee \cdots \vee \mathcal{T}_0[f]_{\mathsf{bind}((\gamma,x_n),\alpha)} | x_i \in S$,
if S has a context deletion change;

(2c) $\bot \vee \mathcal{T}[f]_{\mathsf{bind}((\gamma,x_1),\alpha)} \vee \cdots \vee \mathcal{T}[f]_{\mathsf{bind}((\gamma,x_n),\alpha)} | x_i \in S$,
if $\mathsf{Affected}(f) = \top$. □

APPENDIX B. THEOREM 2 AND 3, AND THEIR PROOFS

We present and prove Theorems 2 and 3 in this appendix.

THEOREM 2. *The following gives link generation semantics for existential,* or, *and* implies *formulas in ECC.*

1. $\mathcal{L}[\exists\gamma \text{ in } S\ (f)]_\alpha =$
   $\{l \,|\, l \in \{(\mathsf{satisfied}, \{(\gamma, x)\})\} \otimes \mathcal{L}[f]_{\mathsf{bind}((\gamma,x),\alpha)} \wedge x \in S \wedge \mathcal{T}[f]_{\mathsf{bind}((\gamma,x),\alpha)} = \top\}$
2. $\mathcal{L}[(f_1) \text{ or } (f_2)\ ]_\alpha =$
   (1) $\mathcal{L}[f_1]_\alpha \cup \mathcal{L}[f_2]_\alpha$, if $\mathcal{T}[f_1]_\alpha = \mathcal{T}[f_2]_\alpha = \top$;
   (2) $\mathcal{L}[f_1]_\alpha \otimes \mathcal{L}[f_2]_\alpha$, if $\mathcal{T}[f_1]_\alpha = \mathcal{T}[f_2]_\alpha = \bot$;
   (3) $\mathcal{L}[f_1]_\alpha$, if $\mathcal{T}[f_1]_\alpha = \top$ and $\mathcal{T}[f_2]_\alpha = \bot$;
   (4) $\mathcal{L}[f_2]_\alpha$, if $\mathcal{T}[f_1]_\alpha = \bot$ and $\mathcal{T}[f_2]_\alpha = \top$
3. $\mathcal{L}[(f_1) \text{ implies } (f_2)\ ]_\alpha =$
   (1) $\mathsf{FlipSet}(\mathcal{L}[f_1]_\alpha) \otimes \mathcal{L}[f_2]_\alpha$, if $\mathcal{T}[f_1]_\alpha = \top$ and $\mathcal{T}[f_2]_\alpha = \bot$;
   (2) $\mathsf{FlipSet}(\mathcal{L}[f_1]_\alpha) \cup \mathcal{L}[f_2]_\alpha$, if $\mathcal{T}[f_1]_\alpha = \bot$ and $\mathcal{T}[f_2]_\alpha = \top$;
   (3) $\mathcal{L}[f_2]_\alpha$, if $\mathcal{T}[f_1]_\alpha = \mathcal{T}[f_2]_\alpha = \top$;
   (4) $\mathsf{FlipSet}(\mathcal{L}[f_1]_\alpha)$, if $\mathcal{T}[f_1]_\alpha = \mathcal{T}[f_2]_\alpha = \bot$.

In Theorem 2, the link generation semantics for the existential formula are similar to those for the universal formula, but they generate satisfaction links instead. For or and implies formulas, their link generation semantics can be interpreted in the same way as for the and formula. We thus omit the details. In the following, we give the proof of Theorem 2 by focusing on the existential formula. The semantics of or and implies formulas can be proved similarly.

PROOF OF THEOREM 2 (PART 1).

1. $\mathcal{L}[\exists\gamma \text{ in } S\ (f)]_\alpha$
   $= \mathcal{L}[\text{not } (\forall\gamma \text{ in } S \text{ (not } (f)))\ ]_\alpha$
   $= \{\mathsf{Flip}(l) \,|\, l \in \mathrm{L}[\forall\gamma \text{ in } S \text{ (not}(f))]_\alpha\}$
   $= \{\mathsf{Flip}(l) \,|\, l \in \{l_0 | l_0 \in \{(\mathsf{violated}, \{(\gamma, x)\})\} \otimes \mathcal{L}[\text{not } (f)]_{\mathsf{bind}((\gamma,x),\alpha)} \wedge x \in S \wedge \mathcal{T}[\text{not } (f)]_{\mathsf{bind}((\gamma,x),\alpha)} = \bot\}\}$
   $= \{\mathsf{Flip}(l) \,|\, l \in \{l_0 | l_0 \in \{(\mathsf{violated}, \{(\gamma, x)\})\} \otimes \{\mathsf{Flip}(l_1) \,|\, l_1 \in \mathcal{L}[(f)]_{\mathsf{bind}((\gamma,x),\alpha)}\} \wedge x \in S \wedge \neg\, \mathcal{T}[f]_{\mathsf{bind}((\gamma,x),\alpha)} = \bot\}\}$
   $= \{l \,|\, l \in \{(\mathsf{satisfied}, \{(\gamma, x)\})\} \otimes \mathcal{L}[f]_{\mathsf{bind}((\gamma,x),\alpha)} \wedge x \in S \wedge \mathcal{T}[f]_{\mathsf{bind}((\gamma,x),\alpha)} = \top\}$. □

THEOREM 3. *The following gives link generation semantics for existential,* or, *and* implies *formulas in PCC.*

1. $\mathcal{L}[\exists\gamma \text{ in } S\ (f)]_\alpha =$
   (1) $\mathcal{L}_0[\exists\gamma \text{ in } S\ (f)]_\alpha$,
   if S has no change and $\mathsf{Affected}(f) = \bot$;

(2) $\mathcal{L}_0[\exists\gamma \text{ in } S\ (f)]_\alpha \cup \{l \mid l \in \{(\text{satisfied}, \{(\gamma, x)\})\} \otimes \mathcal{L}[f]_{\text{bind}((\gamma,x),\alpha)} \wedge \{x\} = S - S_0 \wedge \mathcal{T}\ [f]_{\text{bind}((\gamma,x),\alpha)} = \top\}$,
if S has a context addition change;
(3) $\{l \mid l \in \{(\text{satisfied}, \{(\gamma, x)\})\} \otimes \mathcal{L}_0[f]_{\text{bind}((\gamma,x),\alpha)} \wedge x \in S \wedge \mathcal{T}\ [f]_{\text{bind}((\gamma,x),\alpha)} = \top\}$,
if S has a context deletion change;
(4) $\{l \mid l \in \{(\text{satisfied}, \{(\gamma, x)\})\} \otimes \mathcal{L}[f]_{\text{bind}((\gamma,x),\alpha)} \wedge x \in S \wedge \mathcal{T}\ [f]_{\text{bind}((\gamma,x),\alpha)} = \top\}$,
if $\text{Affected}(f) = \top$.

2. $\mathcal{L}[(f_1) \text{ or } (f_2)\ ]_\alpha =$
(1) $\mathcal{L}_0[(f_1) \text{ or } (f_2)\ ]_\alpha$,
if $\text{Affected}(f_1) = \text{Affected}(f_2) = \bot$;
(2) a. $\mathcal{L}[f_1]_\alpha \cup \mathcal{L}_0[f_2]_\alpha$, if $\mathcal{T}\ [f_1]_\alpha = \mathcal{T}\ [f_2]_\alpha = \top$;
b. $\mathcal{L}[f_1]_\alpha \otimes \mathcal{L}_0[f_2]_\alpha$, if $\mathcal{T}\ [f_1]_\alpha = \mathcal{T}\ [f_2]_\alpha = \bot$;
c. $\mathcal{L}[f_1]_\alpha$, if $\mathcal{T}\ [f_1]_\alpha = \top$ and $\mathcal{T}\ [f_2]_\alpha = \bot$;
d. $\mathcal{L}_0[f_2]_\alpha$, if $\mathcal{T}\ [f_1]_\alpha = \bot$ and $\mathcal{T}\ [f_2]_\alpha = \top$,
if $\text{Affected}(f_1) = \top$;
(3) a. $\mathcal{L}_0[f_1]_\alpha \cup \mathcal{L}[f_2]_\alpha$, if $\mathcal{T}\ [f_1]_\alpha = \mathcal{T}\ [f_2]_\alpha = \top$;
b. $\mathcal{L}_0[f_1]_\alpha \otimes \mathcal{L}[f_2]_\alpha$, if $\mathcal{T}\ [f_1]_\alpha = \mathcal{T}\ [f_2]_\alpha = \bot$;
c. $\mathcal{L}_0[f_1]_\alpha$, if $\mathcal{T}\ [f_1]_\alpha = \top$ and $\mathcal{T}\ [f_2]_\alpha = \bot$;
d. $\mathcal{L}[f_2]_\alpha$, if $\mathcal{T}\ [f_1]_\alpha = \bot$ and $\mathcal{T}\ [f_2]_\alpha = \top$,
if $\text{Affected}(f_2) = \top$.

3. $\mathcal{L}[(f_1) \text{ implies } (f_2)\ ]_\alpha =$
(1) $\mathcal{L}_0[(f_1) \text{ implies } (f_2)\ ]_\alpha$,
if $\text{Affected}(f_1) = \text{Affected}(f_2) = \bot$;
(2) a. $\text{FlipSet}(\mathcal{L}[f_1]_\alpha) \otimes \mathcal{L}_0[f_2]_\alpha$, if $\mathcal{T}\ [f_1]_\alpha = \top$ and $\mathcal{T}\ [f_2]_\alpha = \bot$;
b. $\text{FlipSet}(\mathcal{L}[f_1]_\alpha) \cup \mathcal{L}_0[f_2]_\alpha$, if $\mathcal{T}\ [f_1]_\alpha = \bot$ and $\mathcal{T}\ [f_2]_\alpha = \top$;
c. $\mathcal{L}_0[f_2]_\alpha$, if $\mathcal{T}\ [f_1]_\alpha = \mathcal{T}\ [f_2]_\alpha = \top$;
d. $\text{FlipSet}(\mathcal{L}[f_1]_\alpha)$, if $\mathcal{T}\ [f_1]_\alpha = \mathcal{T}\ [f_2]_\alpha = \bot$,
if $\text{Affected}(f_1) = \top$;
(3) a. $\text{FlipSet}(\mathcal{L}_0[f_1]_\alpha) \otimes \mathcal{L}[f_2]_\alpha$, if $\mathcal{T}\ [f_1]_\alpha = \top$ and $\mathcal{T}\ [f_2]_\alpha = \bot$;
b. $\text{FlipSet}(\mathcal{L}_0[f_1]_\alpha) \cup \mathcal{L}[f_2]_\alpha$, if $\mathcal{T}\ [f_1]_\alpha = \bot$ and $\mathcal{T}\ [f_2]_\alpha = \top$;
c. $\mathcal{L}[f_2]_\alpha$, if $\mathcal{T}\ [f_1]_\alpha = \mathcal{T}\ [f_2]_\alpha = \top$;
d. $\text{FlipSet}(\mathcal{L}_0[f_1]_\alpha)$, if $\mathcal{T}\ [f_1]_\alpha = \mathcal{T}\ [f_2]_\alpha = \bot$,
if $\text{Affected}(f_2) = \top$. □

In the following, we give the proof of Theorem 3 by focusing on the existential formula. The semantics of or and implies formulas can be proved similarly.

PROOF OF THEOREM 3 (PART 1).

1. $\mathcal{L}[\exists\gamma \text{ in } S\ (f)]_\alpha$
$= \mathcal{L}[\text{not } (\forall\gamma \text{ in } S \text{ (not } (f))\ ]_\alpha$
$=$ (1) $\mathcal{L}_0[\text{not } (\forall\gamma \text{ in } S \text{ (not } (f))\ ]_\alpha$,
if $\text{Affected}(\forall\gamma \text{ in } S \text{ (not } (f))) = \bot$;
(2) $\{\text{Flip}(l) \mid l \in \text{L}[\forall\gamma \text{ in } S \text{ (not } (f))\ ]_\alpha\}$,
if $\text{Affected}(\forall\gamma \text{ in } S \text{ (not } (f))) = \top$.
$=$ (1) $\mathcal{L}_0[\text{not } (\forall\gamma \text{ in } S \text{ (not } (f))\ ]_\alpha$,
if S has no change and $\text{Affected}(f) = \bot$;

(2a) $\{\mathsf{Flip}(l)|l \in \mathcal{L}_0[\forall\gamma \text{ in } S \text{ (not } (f))]_\alpha \cup \{l_0|l_0 \in \{(\text{violated}, \{(\gamma, x)\})\} \otimes \mathcal{L}[\text{not } (f)]_{\mathsf{bind}((\gamma,x),\alpha)} \wedge \{x\} = S - S_0 \wedge \mathcal{T}[\text{not } (f)]_{\mathsf{bind}((\gamma,x),\alpha)} = \bot\}\}$,
if S has a context addition change;

(2b) $\{\mathsf{Flip}(l)|l \in \{l_0|l_0 \in \{(\text{violated}, \{(\gamma, x)\})\} \otimes \mathcal{L}_0[\text{not } (f)]_{\mathsf{bind}((\gamma,x),\alpha)} \wedge x \in S \wedge \mathcal{T}[\text{not } (f)]_{\mathsf{bind}((\gamma,x),\alpha)} = \bot\}\}$,
if S has a context deletion change;

(2c) $\{\mathsf{Flip}(l)|l \in \{l_0|l_0 \in \{(\text{violated}, \{(\gamma, x)\})\} \otimes \mathcal{L}[\text{not } (f)]_{\mathsf{bind}((\gamma,x),\alpha)} \wedge x \in S \wedge \mathcal{T}[\text{not } (f)]_{\mathsf{bind}((\gamma,x),\alpha)} = \bot\}\}$,
if $\mathsf{Affected}(f) = \top$.

$=$ (1) $\mathcal{L}_0[\exists\gamma \text{ in } S\ (f)]_\alpha$,
if S has no change and $\mathsf{Affected}(f) = \bot$;

(2a) $\mathcal{L}_0[\text{not } (\forall\gamma \text{ in } S \text{ (not } (f)))]_\alpha \cup \{l|l \in \{(\text{satisfied}, \{(\gamma, x)\})\} \otimes \mathcal{L}[f]_{\mathsf{bind}((\gamma,x),\alpha)} \wedge \{x\} = S - S_0 \wedge \neg\mathcal{T}[f]_{\mathsf{bind}((\gamma,x),\alpha)} = \bot\}$,
if S has a context addition change;

(2b) $\{l|l \in \{(\text{satisfied}, \{(\gamma, x)\})\} \otimes \mathcal{L}_0[f]_{\mathsf{bind}((\gamma,x),\alpha)} \wedge x \in S \wedge \neg\mathrm{T}[f]_{\mathsf{bind}((\gamma,x),\alpha)} = \bot\}$,
if S has a context deletion change;

(2c) $\{l|l \in \{(\text{satisfied}, \{(\gamma, x)\})\} \otimes \mathcal{L}[f]_{\mathsf{bind}((\gamma,x),\alpha)} \wedge x \in S \wedge \neg\mathcal{T}[f]_{\mathsf{bind}((\gamma,x),\alpha)} = \bot\}$,
if $\mathsf{Affected}(f) = \top$.

$=$ (1) $\mathcal{L}_0[\exists\gamma \text{ in } S\ (f)]_\alpha$,
if S has no change and $\mathsf{Affected}(f) = \bot$;

(2a) $\mathcal{L}_0[\exists\gamma \text{ in } S\ (f)]_\alpha \cup \{l|l \in \{(\text{satisfied}, \{(\gamma, x)\})\} \otimes \mathcal{L}[f]_{\mathsf{bind}((\gamma,x),\alpha)} \wedge \{x\} = S - S_0 \wedge \mathcal{T}[f]_{\mathsf{bind}((\gamma,x),\alpha)} = \top\}$,
if S has a context addition change;

(2b) $\{l|l \in \{(\text{satisfied}, \{(\gamma, x)\})\} \otimes \mathcal{L}_0[f]_{\mathsf{bind}((\gamma,x),\alpha)} \wedge x \in S \wedge \mathcal{T}[f]_{\mathsf{bind}((\gamma,x),\alpha)} = \top\}$,
if S has a context deletion change;

(2c) $\{l|l \in \{(\text{satisfied}, \{(\gamma, x)\})\} \otimes \mathcal{L}[f]_{\mathsf{bind}((\gamma,x),\alpha)} \wedge x \in S \wedge \mathcal{T}[f]_{\mathsf{bind}((\gamma,x),\alpha)} = \top\}$,
if $\mathsf{Affected}(f) = \top$. □

APPENDIX C. SOUNDNESS THEOREMS AND THE PROOFS

We prove the two soundness theorems, Theorem 4 and Theorem 5, in this appendix.

THEOREM 4 (SOUNDNESS). *Given two context changes that occur concurrently, their handling order does not affect the final checking result of PCC for any constraint.*

PROOF OF THEOREM 4. We take the link generation as illustration of the proof (the truth value evaluation can be proved in the same way). We focus on the four kernel formula types (i.e., universal, and, not, and *bfunc* formulas), and the other three formula types (i.e., existential, or, and implies) can be proved by inference.

Let the two context changes that occur concurrently be $\mathsf{Chg}_1 = (\mathsf{Op}_1, \mathsf{S}_1, \mathsf{Ctx}_1)$ and $\mathsf{Chg}_2 = (\mathsf{Op}_2, \mathsf{S}_2, \mathsf{Ctx}_2)$, respectively. Symbols Op_1/Op_2 are operations, which

can be Add or Del. Add means that a context $\mathsf{Ctx}_1/\mathsf{Ctx}_2$ is added to a context set $\mathsf{S}_1/\mathsf{S}_2$, whereas Del means that a context $\mathsf{Ctx}_1/\mathsf{Ctx}_2$ is deleted from a context set $\mathsf{S}_1/\mathsf{S}_2$.

Since every context set is associated with a universal formula (we consider the four kernel formula types only), let the two universal formulas associated with S_1 and S_2 be F_1 and F_2, and the two variables defined in the two formulas F_1 and F_2 be γ_1 and γ_2, respectively.

We introduce three functions. Let function Sub return a universal formula's only subformula. Let functions SubFst and SubSec return an and formula's first subformula and second subformula, respectively.

To simplify the proof, we regard every context set used in a constraint unique. If a constraint uses a context set more than once, we give them different names. In the following, we partition all situations into different cases by considering context set $\mathsf{S}_1/\mathsf{S}_2$, context $\mathsf{Ctx}_1/\mathsf{Ctx}_2$, and operation $\mathsf{Op}_1/\mathsf{Op}_2$ together.

—*Case* (1) ($\mathsf{S}_1 = \mathsf{S}_2$). That is, the two context changes affect the same context set, and thus affect the same universal formula ($\mathsf{F}_1 = \mathsf{F}_2$). From $\mathsf{F}_1 = \mathsf{F}_2$, we get $\gamma_1 = \gamma_2$.

—*Case* (1.1) ($\mathsf{Ctx}_1 = \mathsf{Ctx}_2$). That is, the two context changes use the same context.

If we also have $\mathsf{Op}_1 = \mathsf{Op}_2$, then the two context changes are actually the same change ($\mathsf{Chg}_1 = \mathsf{Chg}_2$). Changing their handling order does not cause any difference. If $\mathsf{Op}_1 \neq \mathsf{Op}_2$, a context is added to and then deleted from the same context set at the same time. The combined effort is as if no context change was made. In fact, the situations in Case (1.1) may not occur in practice.

—*Case* (1.2) ($\mathsf{Ctx}_1 \neq \mathsf{Ctx}_2$). That is, the two context changes use different contexts.

—*Case* (1.2.1) ($\mathsf{Op}_1 = \mathsf{Op}_2$). If $\mathsf{Op}_1 = \mathsf{Op}_2 = \mathsf{Add}$, two contexts Ctx_1 and Ctx_2 are added to the same context set. No matter that Ctx_1 is added first or Ctx_2 is added first, the final checking result is the same. That is, $\mathcal{L}[\mathsf{F}_1]_\alpha = \mathcal{L}_0[\mathsf{F}_1]_\alpha \cup \{l \mid l \in \{(\mathsf{violated}, \{(\gamma_1, x)\})\} \otimes \mathcal{L}[\mathsf{Sub}(\mathsf{F}_1)]_{\mathsf{bind}((\gamma 1,x),\alpha)} \wedge \{x\} \in \{\mathsf{Ctx}_1, \mathsf{Ctx}_2\} \wedge \mathcal{T}\ [\mathsf{Sub}(\mathsf{F}_1)]_{\mathsf{bind}((\gamma 1,x),\alpha)} = \bot\}$.

If $\mathsf{Op}_1 = \mathsf{Op}_2 = \mathsf{Del}$, two contexts Ctx_1 and Ctx_2 are deleted from the same context set. No matter that Ctx_1 is deleted first or Ctx_2 is deleted first, the final checking result is also the same. That is, $\mathcal{L}[\mathsf{F}_1]_\alpha = \{l \mid l \in \{(\mathsf{violated}, \{(\gamma_1, x)\})\} \otimes \mathcal{L}_0[\mathsf{Sub}(\mathsf{F}_1)]_{\mathsf{bind}((\gamma 1,x),\alpha)} \wedge \{x\} \in \mathsf{S}_1 - \{\mathsf{Ctx}_1, \mathsf{Ctx}_2\} \wedge \mathcal{T}\ [\mathsf{Sub}(\mathsf{F}_1)]_{\mathsf{bind}((\gamma 1,x),\alpha)} = \bot\}$.

—*Case* (1.2.2) ($\mathsf{Op}_1 \neq \mathsf{Op}_2$). Without the loss of generality, we assume $\mathsf{Op}_1 = \mathsf{Add}$ and $\mathsf{Op}_2 = \mathsf{Del}$. This means that context Ctx_1 is added to and context Ctx_2 is deleted from the same context set.

No matter that Ctx_1 is added first or Ctx_2 is deleted first, the final checking result is the same. That is, $\mathcal{L}[\mathsf{F}_1]_\alpha = \{l \mid l \in \{(\mathsf{violated}, \{(\gamma_1, x)\})\} \otimes \mathcal{L}_0[\mathsf{Sub}(\mathsf{F}_1)]_{\mathsf{bind}((\gamma 1,x),\alpha)} \wedge \{x\} \in \mathsf{S}_1 - \{\mathsf{Ctx}_2\} \wedge \mathcal{T}\ [\mathsf{Sub}(\mathsf{F}_1)]_{\mathsf{bind}((\gamma 1,x),\alpha)} = \bot\} \cup \{l \mid l \in \{(\mathsf{violated}, \{(\gamma_1, x)\})\} \otimes \mathcal{L}[\mathsf{Sub}(\mathsf{F}_1)]_{\mathsf{bind}((\gamma 1,x),\alpha)} \wedge x = \mathsf{Ctx}_1 \wedge \mathcal{T}\ [\mathsf{Sub}(\mathsf{F}_1)]_{\mathsf{bind}((\gamma 1,x),\alpha)} = \bot\}$.

—*Case* (2) ($\mathsf{S}_1 \neq \mathsf{S}_2$). That is, the two context changes affect two different context sets ($\mathsf{S}_1 \neq \mathsf{S}_2$).

—*Case* (2.1) (F_1 *is an ancestor formula of* F_2). This means that F_2 is the subformula of F_1, or some subformula of the subformula of $\mathsf{F}_1, \ldots$, and so on. Therefore, F_1 must take the form of $\forall\gamma_1$ in S_1 $(\cdots \forall\gamma_2$ in S_2 $(\cdots)\cdots)$.

Suppose that context change Chg_1 is applied to F_1 first. Then, $\mathcal{L}[\mathsf{F}_1]'_\alpha = \mathcal{L}_0[\mathsf{F}_1]_\alpha \cup \{l \,|\, l \in \{(\text{violated}, \{(\gamma_1, x)\})\} \otimes \mathcal{L}[\mathsf{Sub}(\mathsf{F}_1)]_{\mathsf{bind}((\gamma 1,x),\alpha)} \wedge x = \mathsf{Ctx}_1 \wedge \mathcal{T}\ [\mathsf{Sub}(\mathsf{F}_1)]_{\mathsf{bind}((\gamma 1,x),\alpha)} = \bot\}$ if $\mathsf{Op}_1 = \mathsf{Add}$, or $\mathcal{L}[\mathsf{F}_1]'_\alpha = \{l \,|\, l \in \{(\text{violated}, \{(\gamma_1, x)\})\} \otimes \mathcal{L}_0[\mathsf{Sub}(\mathsf{F}_1)]_{\mathsf{bind}((\gamma 1,x),\alpha)} \wedge \{x\} \in \mathsf{S}_1 - \{\mathsf{Ctx}_1\} \wedge \mathcal{T}\ [\mathsf{Sub}(\mathsf{F}_1)]_{\mathsf{bind}((\gamma 1,x),\alpha)} = \bot\}$ if $\mathsf{Op}_1 = \mathsf{Del}$. Then, context change Chg_2 is applied to F_2. Since $\mathsf{Affected}(\mathsf{Sub}(\mathsf{F}_1)) = \top$, the final checking result is $\mathcal{L}[\mathsf{F}_1]_\alpha = \{l \,|\, l \in \{(\text{violated}, \{(\gamma_1, x)\})\} \otimes \mathcal{L}[\mathsf{Sub}(\mathsf{F}_1)]_{\mathsf{bind}((\gamma 1,x),\alpha)} \wedge \{x\} \in \mathsf{S}'_1 \wedge \mathcal{T}\ [\mathsf{Sub}(\mathsf{F}_1)]_{\mathsf{bind}((\gamma 1,x),\alpha)} = \bot\}$, where $\mathsf{S}'_1 = S_1 \cup \{\mathsf{Ctx}_1\}$ if $\mathsf{Op}_1 = \mathsf{Add}$ or $\mathsf{S}'_1 = \mathsf{S}_1 - \{\mathsf{Ctx}_1\}$ if $\mathsf{Op}_1 = \mathsf{Del}$.

On the other hand, suppose that context change Chg_2 is applied to F_2 first. Since $\mathsf{Affected}(\mathsf{Sub}(\mathsf{F}_1)) = \top$, $\mathcal{L}[\mathsf{F}_1]'_\alpha = \{l \,|\, l \in \{(\text{violated}, \{(\gamma_1, x)\})\} \otimes \mathcal{L}[\mathsf{Sub}(\mathsf{F}_1)]_{\mathsf{bind}((\gamma 1,x),\alpha)} \wedge \{x\} \in \mathsf{S}_1 \wedge \mathcal{T}\ [\mathsf{Sub}(\mathsf{F}_1)]_{\mathsf{bind}((\gamma 1,x),\alpha)} = \bot\}$. Then, context change chg_1 is applied to F_1. The final checking result is $\mathcal{L}[\mathsf{F}_1]_\alpha = \mathcal{L}[\mathsf{F}_1]_\alpha$' $\cup \{l \,|\, l \in \{(\text{violated}, \{(\gamma_1, x)\})\} \otimes \mathcal{L}[\mathsf{Sub}(\mathsf{F}_1)]_{\mathsf{bind}((\gamma 1,x),\alpha)} \wedge x = \mathsf{Ctx}_1 \wedge \mathcal{T}\ [\mathsf{Sub}(\mathsf{F}_1)]_{\mathsf{bind}((\gamma 1,x),\alpha)} = \bot\}$ if $\mathsf{Op}_1 = \mathsf{Add}$, or $\mathcal{L}[\mathsf{F}_1]_\alpha = \{l \,|\, l \in \{(\text{violated}, \{(\gamma_1, x)\})\} \otimes \mathcal{L}[\mathsf{Sub}(\mathsf{F}_1)]_{\mathsf{bind}((\gamma 1,x),\alpha)} \wedge \{x\} \in \mathsf{S}_1 - \{\mathsf{Ctx}_1\} \wedge \mathcal{T}\ [\mathsf{Sub}(\mathsf{F}_1)]_{\mathsf{bind}((\gamma 1,x),\alpha)} = \bot\}$ if $\mathsf{Op}_1 = \mathsf{Del}$. The two checking results can be merged into one representation: $\mathcal{L}[\mathsf{F}_1]_\alpha = \{l \,|\, l \in \{(\text{violated}, \{(\gamma_1, x)\})\} \otimes \mathcal{L}[\mathsf{Sub}(\mathsf{F}_1)]_{\mathsf{bind}((\gamma 1,x),\alpha)} \wedge \{x\} \in \mathsf{S}'_1 \wedge \mathcal{T}\ [\mathsf{Sub}(\mathsf{F}_1)]_{\mathsf{bind}((\gamma 1,x),\alpha)} = \bot\}$, where $\mathsf{S}'_1 = \mathsf{S}_1 \cup \{\mathsf{Ctx}_1\}$ if $\mathsf{Op}_1 = \mathsf{Add}$ or $\mathsf{S}'_1 = \mathsf{S}_1 - \{\mathsf{Ctx}_1\}$ if $\mathsf{Op}_1 = \mathsf{Del}$.

From the comparison, the two handling orders actually have the same the final checking result.

—*Case* (2.2) (F_2 *is an ancestor formula of* F_1). This case is similar to Case (2.1).

—*Case* (2.3) (*Neither Case* (2.1) *nor Case* (2.2)). Then F_1 and F_2 must share a common and formula by taking the form of $\mathsf{F}_3 = (\cdots \forall\ \gamma_1$ in $\mathsf{S}_1(\cdots)\cdots)$ and $(\cdots \forall\gamma_2$ in S_2 $(\cdots)\cdots)$ or having the two subformulas are switched.

From the link generation semantics for the and formula in PCC, $\mathcal{L}[\mathsf{SubFst}(\mathsf{F}_3)]_\alpha$ and $\mathcal{L}[\mathsf{SubSec}(\mathsf{F}_3)]_\alpha$ completely decide $\mathcal{L}[\mathsf{F}_3]_\alpha$. Since $\mathcal{L}[\mathsf{SubFst}(\mathsf{F}_3)]_\alpha$ and $\mathcal{L}[\mathsf{SubSec}(\mathsf{F}_3)]_\alpha$ have no direct relationship with each other, and only two context changes Chg_1 and Chg_2 affect them, respectively, the order for handling two changes Chg_1 and Chg_2 also affects only the order of updating $\mathcal{L}[\mathsf{SubFst}(\mathsf{F}_3)]_\alpha$ and $\mathcal{L}[\mathsf{SubSec}(\mathsf{F}_3)]_\alpha$, but does not affect the final checking result of $\mathcal{L}[\mathsf{F}_3]_\alpha$.

As a conclusion, the order for handling two context changes that occur concurrently does not affect the final checking result of PCC for any constraint. This completes the proof. □

To prove Theorem 5, we need the following lemma:

LEMMA 1. *An n-element list* $(e_{i1}, \cdots, e_{in})$ *can be transformed into another list of these elements* $(e_{j1}, \cdots, e_{jn})$ *by finite switches between adjacent elements.*

This lemma is easy to prove because a simple algorithm of continuously moving target elements to one direction by switching will do (e.g., move e_{j1} to the leftmost, then move e_{j2} to the second leftmost, and so on). We omit the proof.

With Lemma 1 and Theorem 4, we prove the following Theorem 5.

THEOREM 5 (SOUNDNESS). *Given any number of context changes that occur concurrently, their handling order does not affect the final checking result of PCC for any constraint.*

PROOF OF THEOREM 5. Let the number of context changes that occur concurrently be n ($n \geq 2$). We consider two lists of the n context changes (Chg_{i1}, ..., Chg_{in}) and (Chg_{j1}, ..., Chg_{jn}) that represent two different handling orders for these n context changes.

Lemma 1 says that (Chg_{i1}, ..., Chg_{in}) can be transformed into (Chg_{j1}, ..., Chg_{jn}) by finite switches of adjacent changes. These switches specify a finite path from (Chg_{i1}, ..., Chg_{in}) to (Chg_{j1}, ..., Chg_{jn}). Each list (except the first one) in the path differs from its prior one by two adjacent context changes. The two adjacent context changes are the same for the two lists but have reversed orders.

Theorem 4 says that handling two context changes in any order has the same final checking result for any constraint in PCC. Therefore, any two adjacent lists in the path have the same checking result. This is because the only difference between them is two order-reversed context changes. As a result, the first list (Chg_{i1}, ..., Chg_{in}) and the last list (Chg_{j1}, ..., Chg_{jn}) also have the same checking result.

Thus, the order for handling n context changes that occur concurrently does not affect the final checking result of PCC for any constraint. This completes the proof. □

APPENDIX D. EQUIVALENCE THEOREM AND THE PROOF

We prove the equivalence theorem, Theorem 6, in this appendix. We need the following lemma:

LEMMA 2. *Given one context change, PCC returns the same checking results as ECC does for any constraint.*

This lemma is easy to prove, as we have explained sufficiently when comparing the PCC and ECC checking semantics. Thus, we omit the proof details. However, when we consider multiple context changes that occur concurrently, whether PCC returns the same checking results as ECC does is not a trivial question. We use induction to prove Theorem 6.

THEOREM 6 (EQUIVALENCE). *Given any number of context changes that occur concurrently, PCC returns the same checking results as ECC does for any constraint.*

PROOF OF THEOREM 6. We use induction to prove the equivalence theorem. Let the number of context changes that occur concurrently be n.

(1) When $n = 1$, from Lemma 2, PCC returns the same checking results as ECC does.
(2) Suppose that, when $n = k$, PCC returns the same checking results as ECC does. Consider the case where $n = k + 1$. We denote the PCC's checking

results by P_{k+1} and ECC's checking results by E_{k+1} after handling $k + 1$ context changes. Our objective is to prove $P_{k+1} = E_{k+1}$.

We divide the $k + 1$ context changes into two groups G_A and G_B. Group G_A contains k context changes, whereas group G_B contains only one. The division can be arbitrary. We denote PCC's checking results by P_k and ECC's checking results by E_k after handling group G_A. From the assumption, we have $P_k = E_k$.

We introduce two notations $P_{k,1}$ and $E_{k,1}$. $P_{k,1}$ represents PCC's checking results when we first handle group G_A, which contains k context changes, and then handle group G_B, which contains only one context change. $E_{k,1}$ is similar but we use the ECC approach instead.

The difference between P_{k+1} and $P_{k,1}$ is the handling order for $k + 1$ changes. From Theorem 5, the handling order does not affect the final checking results. Thus we have $P_{k+1} = P_{k,1}$. We denote this relation by R1.

On the other hand, there is no difference between E_{k+1} and $E_{k,1}$ since ECC does not rely on any past checking results. Thus we have $E_{k+1} = E_{k,1}$. We denote this relation by R2.

From the assumption, $P_k = E_k$. Since $P_{k,1}$ and $E_{k,1}$ only differ from P_k and E_k by one context change from group G_B, $P_{k,1} = E_{k,1}$ holds according to Lemma 2. We denote this relation by R3.

Combining three relations R1, R2, and R3, we obtain $P_{k+1} = P_{k,1} = E_{k,1} = E_{k+1}$. Thus PCC returns the same checking results as ECC does when $n = k + 1$.

From (1) and (2), given any number of context changes that occur concurrently, PCC returns the same checking results as ECC does. This completes the proof. □