BFF: Foundational and Automated Verification of Bitfield-Manipulating Programs (Appendix)

FENGMIN ZHU, MPI-SWS, Germany
MICHAEL SAMMLER, MPI-SWS, Germany
RODOLPHE LEPIGRE, MPI-SWS, Germany
DEREK DREYER, MPI-SWS, Germany
DEEPAK GARG, MPI-SWS, Germany

A EXTRACTION & CLEARING OPERATIONS ON TERMS

In this section, we present the definitions of two operations on terms—extraction and clearing—that are not shown in the main text, together with their meta-properties.

Extraction. The definition of the extraction operation $t_1 \searrow t_2$ follows the same structure as the merge operation shown in the main text. This operation is defined only when t_1 and t_2 are sorted to the same signature and t_2 is a mask.

$$
\begin{aligned}
\mathsf{nil} \searrow t &= \mathsf{nil} && (1)\\
t \searrow \mathsf{nil} &= \mathsf{nil} && (2)\\
(r_1 \mapsto v_1 :: t_1) \searrow (r_2 \mapsto v_2 :: t_2) &= \textbf{if } r_1 = r_2 \textbf{ then } r_1 \mapsto (v_1 \searrow^{\mathrm{v}} v_2) :: (t_1 \searrow t_2) && (3)\\
& \textbf{else if } r_1 \prec r_2 \textbf{ then } t_1 \searrow (r_2 \mapsto v_2 :: t_2)\\
& \textbf{else} (r_1 \mapsto v_1 :: t_1) \searrow t_2\\
v_1 \searrow^{\mathrm{v}} \mathsf{mask} &= v_1 && (4)\\
\mathsf{nested}(t_r) \searrow^{\mathrm{v}} \mathsf{nested}(m_r) &= \mathsf{nested}(t_r \searrow m_r) && (5)\\
\mathsf{mask} \searrow^{\mathrm{v}} \mathsf{nested}(m_r) &= \mathsf{nested}(m_r) && (6)
\end{aligned}
$$

An auxiliary function $v_1 \searrow^{\mathrm{v}} v_2$ handles the extraction operation over data values (in 3): If v_2 is mask, v_1 needs be extracted and thus is returned (4). Otherwise, v_2 must be a nested mask $\mathsf{nested}(m_r)$. If v_1 is also nested, then the operation applies to the nested terms recursively (5). Otherwise, v_1 must be mask. In this case, the algorithm needs to extract the values of the ranges specified in m_r, which is just m_r itself (6).

The following theorem states that on well-sorted terms that satisfy the preconditions of $\searrow$, the operation preserves sorts, and that, *semantically*, $\searrow$ simulates the bitwise & operation.

Theorem A.1. *Suppose* $\vdash t_1 : \sigma$ *and* $\vdash t_2 : \sigma$. *If* $\mathsf{is\_mask}(t_2)$, *then:*

(1) $\vdash t_1 \searrow t_2 : \sigma$;
(2) $[\![t_1 \searrow t_2]\!] = [\![t_1]\!] \,\&\, [\![t_2]\!]$, *where & is the bitwise AND operator on Coq integers.*

Clearing. The definition of the clearing operation $t_1 \searrow t_2$ also follows the same structure as the merge operation shown in the main text. Note the duality to extraction since clearing is the opposite—instead of extracting bitfields, it unsets those values. This operation is defined only when t_1 and t_2 are sorted to the same signature and t_2 is a mask.

Authors' addresses: Fengmin Zhu, MPI-SWS, Saarland Informatics Campus, Germany, paulzhu@mpi-sws.org; Michael Sammler, MPI-SWS, Saarland Informatics Campus, Germany, msammler@mpi-sws.org; Rodolphe Lepigre, MPI-SWS, Saarland Informatics Campus, Germany, lepigre@mpi-sws.org; Derek Dreyer, MPI-SWS, Saarland Informatics Campus, Germany, dreyer@mpi-sws.org; Deepak Garg, MPI-SWS, Saarland Informatics Campus, Germany, dg@mpi-sws.org.

$$
\begin{aligned}
\mathsf{nil} \searrow_{\sim} t &= \mathsf{nil} && (7)\\
t \searrow_{\sim} \mathsf{nil} &= t && (8)\\
(r_1 \mapsto v_1 :: t_1) \searrow_{\sim} (r_2 \mapsto v_2 :: t_2) &= \textbf{if } r_1 = r_2 \textbf{ then } r_1 \mapsto (v_1 \searrow_{\sim}^{\mathrm{v}} v_2) :: (t_1 \searrow_{\sim} t_2) && (9)\\
&\quad\ \textbf{else if } r_1 \prec r_2 \textbf{ then } r_1 \mapsto v_1 :: (t_1 \searrow_{\sim} (r_2 \mapsto v_2 :: t_2))\\
&\quad\ \textbf{else} (r_1 \mapsto v_1 :: t_1) \searrow_{\sim} t_2\\
v_1 \searrow_{\sim}^{\mathrm{v}} \mathsf{mask} &= \mathsf{data}(0) && (10)\\
\mathsf{nested}(t_r) \searrow_{\sim}^{\mathrm{v}} \mathsf{nested}(m_r) &= \mathsf{nested}(t_r \searrow_{\sim} m_r) && (11)\\
\mathsf{mask} \searrow_{\sim}^{\mathrm{v}} \mathsf{nested}(m_r) &= \mathsf{nested}(\mathsf{comp}(m_r)) && (12)
\end{aligned}
$$

An auxiliary function $v_1 \searrow_{\sim}^{\mathrm{v}} v_2$ handles the clearing operation over data values (in 9): If v_2 is mask, v_1 needs be cleared (or unset) and thus the algorithm returns $\mathsf{data}(0)$ (10). If both are nested terms, then the operation applies to the nested terms recursively (11). Finally, if v_1 is mask and v_2 is a nested mask m_r, then the algorithm needs to clear all the ranges mentioned by m_r in v_1. Since v_1 is a mask, this means the result will be the negation of m_r (denoted by $\mathsf{comp}(m_r)$) wrapped by nested (12).

The following theorem states that on well-sorted terms that satisfy the preconditions of $\searrow_{\sim}$, the operation preserves sorts, and that, *semantically*, $\searrow_{\sim}$ simulates the bitwise `&~` operation.

Theorem A.2. *Suppose* $\vdash t_1 : \sigma$ *and* $\vdash t_2 : \sigma$. *If* $\mathsf{is\_mask}(t_2)$, *then:*

(1) $\vdash t_1 \searrow_{\sim} t_2 : \sigma$;

(2) $[\![t_1 \searrow_{\sim} t_2]\!] = [\![t_1]\!] \mathbin{\&} {\sim}[\![t_2]\!]$, *where* $\sim$ *is the bitwise NOT operator on Coq integers.*