Pseudo-code for Algorithm B

ANONYMOUS AUTHOR(S)

ACM Reference Format:
Anonymous Author(s). 2020. Pseudo-code for Algorithm B. In *Proceedings of ACM SIGPLAN Conference on Functional Programming (ICFP'20).* ACM, New York, NY, USA, 3 pages.

Fig. 1 presents, `unify`, the main algorithm for unification. `unify` takes in a set of components $\mathcal{G}$, a set of variable closure constraints γ, and a substitution σ. `unify` returns a set of constraints and a substitution if it succeeds. `unify` may reports unsolvability in two cases: (1) a cycle is detected if `select` does not find an component; (2) $\texttt{dec}_0$ and $\texttt{dec}_1$ does not solve a component.

As in Section 7.2, a component is divided into two parts: one contains the two-variable equations and the other contains the rest. The two-variable equations are not immediately solvable, and thus are processed after the others. Fig. 2 presents $\texttt{dec}_0$ that solves the equations that each have a non-variable term. Fig. 3 presents $\texttt{dec}_1$ that solves the equations that each contain two variables. $\texttt{dec}_0$ and $\texttt{dec}_1$ maintain a queue, q, of the solved variables. A variable may be added to q at most once. `unify` first invokes $\texttt{dec}_0$ and then $\texttt{dec}_1$. When $\texttt{dec}_1$ is invoked, A variable x is removed from q. Then equations in g that each have an x on a side is removed from g. Since x is solved, these equations now each have a non-variable on one side and they are now immediately solvable.

$\texttt{unify}(\mathcal{G};\gamma;\sigma)$:
if $\mathcal{G} = \emptyset$ **then**
    return $\gamma;\sigma$
else
    $g;\mathcal{G} \leftarrow \texttt{select}(\mathcal{G})$
    ;; g_0 contains the equations in g that each have a non-variable term
    ;; g_1 contains the equations in g that each have two variables on both sides
    if $g_0 = []$ **then**
        $\gamma \leftarrow g_1 \uplus \gamma$
        return $\texttt{unify}(\mathcal{G},\gamma,\sigma)$
    else
        $\mathcal{G};\sigma;q \leftarrow \texttt{dec}_0(g_0,\mathcal{G},\sigma,[]))$
        $\mathcal{G};\sigma \quad \leftarrow \texttt{dec}_1(g_1,\mathcal{G},\sigma,q))$
    end if
    return $\texttt{unify}(\mathcal{G},\gamma,\sigma)$
end if

Fig. 1. The main unification algorithm

$\texttt{dec}_0(g_0,\mathcal{G},\sigma,q)$:
if $g_0 = []$ **then**
    return $\mathcal{G};\sigma;q$
else
    $\langle\phi_1\,;t_1\rangle \overset{\alpha?}{=} \langle\phi_2\,;t_2\rangle \quad\leftarrow\quad \texttt{first}(g_0)$
    $g_0 \quad\leftarrow\quad \texttt{rest}(g_0)$
    $\mathcal{G};\sigma;q \quad\leftarrow\quad \texttt{solve}(\langle\phi_1\,;\sigma(t_1)\rangle \overset{\alpha?}{=} \langle\phi_2\,;\sigma(t_2)\rangle;\mathcal{G};\sigma;q)$
    return $\texttt{dec}_0(g_0,\mathcal{G},\sigma,q)$
end if

Fig. 2. Algorithm for a part of a component

$\texttt{dec}_1(g_1,\mathcal{G},\sigma,q)$:
if $q = []$ **then**
    return $\mathcal{G};\sigma$
else
    $x \leftarrow \texttt{first}(q)$
    $q \leftarrow \texttt{rest}(q)$
    forall $\langle\phi_1\,;t_1\rangle \overset{\alpha?}{=} \langle\phi_2\,;t_2\rangle$ **in** $\texttt{ref}(g_1,x)$
        $\mathcal{G};\sigma;q \leftarrow \texttt{solve}(\langle\phi_1\,;\sigma(t_1)\rangle \overset{\alpha?}{=} \langle\phi_2\,;\sigma(t_2)\rangle;\mathcal{G};\sigma;q)$
    return $\texttt{dec}_1(g_1,\mathcal{G},\sigma,q)$
end if

Fig. 3. Algorithm for a part of component

$$
\begin{array}{lll}
\text{(c.1)} & \texttt{solve}(\langle \phi_1 \,; a_1 \rangle \overset{\alpha?}{=} \langle \phi_2 \,; a_2 \rangle ; \mathcal{G} ; \sigma ; q) & \longrightarrow \quad \mathcal{G} ; \sigma ; q \\
 & & \text{where} \quad \langle \phi_1 \,; a_1 \rangle \overset{\alpha}{\approx} \langle \phi_2 \,; a_2 \rangle \\
\text{(c.2)} & \texttt{solve}(\langle \phi_1 \,; \lambda a_1 .\, t_1 \rangle \overset{\alpha?}{=} \langle \phi_2 \,; \lambda a_2 .\, t_2 \rangle ; \mathcal{G} ; \sigma ; q) & \longrightarrow \quad \mathcal{G}' ; \sigma ; q \\
 & & \text{where} \quad \mathcal{G}' \text{ is } \{\langle \phi_1, a_1 \,; t_1 \rangle \overset{\alpha?}{=} \langle \phi_2, a_2 \,; t_2 \rangle\} \\
 & & \qquad\qquad \uplus \mathcal{G} \\
\text{(c.3)} & \texttt{solve}(\langle \phi_1 \,; f(t_1, t_2) \rangle \overset{\alpha?}{=} \langle \phi_2 \,; f(t_1', t_2') \rangle ; \mathcal{G} ; \sigma ; q) & \longrightarrow \quad \mathcal{G}' ; \sigma ; q \\
 & & \text{where} \quad \mathcal{G}' \text{ is } \{\langle \phi_1 \,; t_1 \rangle \overset{\alpha?}{=} \langle \phi_2 \,; t_1' \rangle\} \\
 & & \qquad\qquad \uplus \{\langle \phi_1 \,; t_2 \rangle \overset{\alpha?}{=} \langle \phi_2 \,; t_2' \rangle\} \\
 & & \qquad\qquad \uplus \mathcal{G} \\
\text{(d.1)} & \texttt{solve}(\langle \phi_1 \,; x_1 \rangle \overset{\alpha?}{=} \langle \phi_2 \,; a_2 \rangle ; \mathcal{G} ; \sigma ; q) & \longrightarrow \quad \mathcal{G} ; \sigma' ; q' \\
 & & \text{where} \quad \sigma' \text{ is } \{x_1 := a_1\} \uplus \sigma \\
 & & \qquad\quad\;\; q' \text{ is } x_1 \uplus q \\
 & & \qquad\quad\;\; \langle \phi_1 \,; a_1 \rangle \overset{\alpha}{\approx} \langle \phi_2 \,; a_2 \rangle \\
\text{(d.2)} & \texttt{solve}(\langle \phi_1 \,; x_1 \rangle \overset{\alpha?}{=} \langle \phi_2 \,; \lambda a_2 .\, t_2 \rangle ; \mathcal{G}' ; \sigma ; q) & \longrightarrow \quad \mathcal{G}' ; \sigma' ; q' \\
 & & \text{where} \quad \mathcal{G}' \text{ is } \{\langle \phi_1, a_1 \,; x_1' \rangle \overset{\alpha?}{=} \langle \phi_2, a_2 \,; t_2 \rangle\} \\
 & & \qquad\qquad \uplus \mathcal{G} \\
 & & \qquad\quad\;\; \sigma' \text{ is } \{x_1 := \lambda a_1 .\, x_1'\} \uplus \sigma \\
 & & \qquad\quad\;\; q' \text{ is } x_1 \uplus q \\
 & & \qquad\quad\;\; a_1 \text{ and } x_1' \text{ are fresh} \\
\text{(d.3)} & \texttt{solve}(\langle \phi_1 \,; x_1 \rangle \overset{\alpha?}{=} \langle \phi_2 \,; f(t_1, t_2) \rangle ; \mathcal{G} ; \sigma ; q) & \longrightarrow \quad \mathcal{G}' ; \sigma' ; q' \\
 & & \text{where} \quad \mathcal{G}' \text{ is } \{\langle \phi_1 \,; x_1' \rangle \overset{\alpha?}{=} \langle \phi_2 \,; t_1 \rangle\} \\
 & & \qquad\qquad \{\langle \phi_1 \,; x_2' \rangle \overset{\alpha?}{=} \langle \phi_2 \,; t_2 \rangle\} \\
 & & \qquad\qquad \uplus \mathcal{G} \\
 & & \qquad\quad\;\; \sigma' \text{ is } \{x_1 := f(x_1', x_2')\} \uplus \sigma \\
 & & \qquad\quad\;\; q' \text{ is } x_1 \uplus q \\
 & & \qquad\quad\;\; x_1' \text{ and } x_2' \text{ are fresh}
\end{array}
$$

Fig. 4. Algorithm for a single step