Algorithm 1: Parameter Checking/Member Accessing

Procedure $check(\gamma, \alpha_{func}, \overline{\alpha_{arg}}, \alpha_{call})$
  $\tau_f^*, \overline{\tau_a^*}, C \leftarrow \gamma(\alpha_{func}), \gamma(\overline{\alpha_{arg}}), \varnothing$;
  for $\tau_f \in \tau_f^*$ **do**
    L$\leftarrow$ () ; // A N-length sequence, N is the number of parameters, L_i is the set of compatible arguments for the i-th parameter
    if $|\tau_f^p| \neq |\overline{\tau_a^*}|$ **then**
      continue ; // Skip those function whose parameter number is not equal to the argument number
    end
    for $\tau_p, \tau_a^* \in \tau_f^p, \overline{\tau_a^*}$ **do**
      $c \leftarrow \varnothing$; // A set of compatible arguments
      for $\tau_a \in \tau_a^*$ **do**
        if $\tau_a <: \tau_p$; // A subtyping judgment. The full algorithmic judgment rules are in subtype.py
        then
          $c \leftarrow c \cup \{\tau_a\}$; // Record the compatible arguments
        end
      end
      $L \leftarrow L@[c]$; // Record L_i
    end
    $V \leftarrow \bigtimes_{i=0}^{|L|} L_i$; // A cartesian product of all sets in L. V is the set of compatible argument combinations
    if $V \neq \varnothing$ **then**
      $C[\tau_f^r \rightarrow C(\tau_f^r) \cup (V, \tau_f)]$; // Record all compatible argument combinations, the function, and the return type; V, τ_f, τ_f^r corresponds to the type of $\bar{x}$, f and $f(\bar{x})$
    end
  end
  $\mathcal{L}(\bigwedge_{\tau_r \in dom(C)} \alpha_c = \tau_r \Leftrightarrow \bigvee_{f \in C(\tau_r)} \bigvee_{\bar{v} \in f_0} (\bar{\alpha_a} = \bar{v} \wedge \alpha_f = f_1))$;
  $\mathcal{L}(\bigvee_{\tau_r \in dom(C)} \alpha_c = \tau_r)$;
  return $dom(C)$
end
Procedure $access(\gamma, \alpha_e, \alpha_m, l)$
  $\tau_e^*, C \leftarrow \gamma(\alpha_e), \varnothing$;
  for $\tau_e \in \tau_e^*$ **do**
    for $cls \in CT(\tau_e^x).mro$ **do**
      ; // τ_e is a instance, τ_e^x is the name of the class of the instance, $CT(x_e^x)$ is the class; We inspect all super class of the class, in method resolution order
      if $l \in fields(cls)$ **then**
        ; // Once we found the first super class that contain the accessing field, we get the type of that field, denoted as τ_m
        $\tau_m \leftarrow ftype(cls, l)$;
        $C[\tau_m \rightarrow C(\tau_m) \cup \{\tau_e\}]$; // Record all field type and object type; τ_e, τ_m corresponds to the type of e and $e.l$
        break;
      end
    end
  end
  $\mathcal{L}(\bigwedge_{\tau_m \in dom(C)} \alpha_m = \tau_m \Leftrightarrow \bigvee_{\tau_e \in C(\tau_m)} (\alpha_e = \tau_e))$;
  $\mathcal{L}(\bigvee_{\tau_m \in dom(C)} \alpha_m = \tau_m)$;
  return $dom(C)$
end