4.4 Formulating SMT Statements

So far we have described the inference rules and the subtyping rule. We have yet to describe an algorithm that can derive a valid type for a set of given subtyping rules.

Definition 4.1: Liquid Type Variable

We say $\mathcal{K} := \{\kappa_i \mid i \in \mathbb{N}\}$ is the set of all *liquid type variables*.

Note that κ is a special character.

Definition 4.2: Template

We say $\hat{T}$ is a *template* $:\Leftrightarrow$

$$\begin{aligned} & \hat{T} \text{ is of form } \{\nu : Int \mid [k]_S\} \\ & \text{where } k \in \mathcal{K} \text{ and } S : \mathcal{V} \nrightarrow \mathcal{Q} \\ \vee\ & \hat{T} \text{ is of form } a : \{\nu : Int \mid [k]_S\} \to \hat{T} \\ & \text{where } k \in \mathcal{K}, \hat{T} \text{ is a template and } S : \mathcal{V} \nrightarrow \mathcal{Q}. \end{aligned}$$

We define $\mathcal{T}^? := \{\hat{T} \mid \hat{T} \text{ is a template}\}$

A template will be used for a liquid type with unknown refinement. Note that the inference rule for function applications introduces a refinement substitution S. For templates this substitution is not defined and needs to be delayed until the corresponding liquid type has been derived. We will point out whenever the substitution $[k]_S$ will be applied.

Definition 4.3: Type Variable Context

Let $K := \{[k]_S | k \in \mathcal{K} \wedge S : \mathcal{V} \nrightarrow \mathcal{Q}\}$.

—

We say $\Theta : \mathcal{V} \nrightarrow (\mathcal{Q} \cup K)$ is a type variable context.

Our algorithm will resolve a set of suptyping conditions:

Definition 4.4: Subtyping Condition

We say c is a *Subtyping Condition* $:\Leftrightarrow$

c is of form $\hat{T}_1 <:_{\Theta,\Lambda} \hat{T}_2$

where $\hat{T}_1, \hat{T}_2$ are a liquid types or templates, Θ is a type variable context and $\Lambda \subset \mathcal{Q}$.

We define $\mathcal{C} := \{c \mid c \text{ is a subtyping condition}\}$

We will also need a function to obtain the set of all liquid type variables of a template or subtyping condition.

Definition 4.5: Vars

$$\begin{aligned}
\text{Vars} : (\mathcal{T} \cup \mathcal{T}^?) &\to \mathcal{P}(\mathcal{K}) \\
\text{Vars}(\{\nu \in Int | r\}) &= \{\} \\
\text{Vars}(\{\nu \in Int | \kappa_i\}) &= \{\kappa_i\} \\
\text{Vars}(a : \{\nu \in Int | \kappa_i\} \to \hat{T}) &= \{\kappa_i\} \cup \text{Vars}(\hat{T})
\end{aligned}$$

$$\begin{aligned}
\text{Vars} : \mathcal{C} &\to \mathcal{P}(\mathcal{K}) \\
\text{Vars}(\hat{T}_1 <:_{\Theta,\Lambda} \hat{T}_2) &= \text{Vars}(\hat{T}_1) \cup \text{Vars}(\hat{T}_2) \\
&\quad \cup \{k | (_, q) \in \Theta \wedge q = [k]_S \text{ for } k \in \mathcal{K} \text{ and } S : \mathcal{V} \nrightarrow \mathcal{Q}\}
\end{aligned}$$

The main idea of the algorithm is to first generate a set of predicates and then exclude elements until all subtyping conditions are valid for the remaining predicates. By conjunction over all remaining predicates we result in a valid refinement.

We therefore need a function, depending on a set of variable Q, that will generate a set of predicates. Note that the resulting set should be finite and a subset of $\mathcal{Q}$. If the generated set is too small, then our resulting subtyping conditions might be too weak.

$$\begin{aligned}
\mathit{Init} : \mathcal{P}(\mathcal{V}) &\to \mathcal{P}(\mathcal{Q}) \\
\mathit{Init}(V) ::= &\{0 < \nu\} \\
&\cup \{a < \nu \mid a \in V\} \\
&\cup \{\nu < 0\} \\
&\cup \{\nu < a \mid a \in V\} \\
&\cup \{\nu = a \mid a \in V\} \\
&\cup \{\nu = 0\} \\
&\cup \{a < \nu \vee \nu = a \mid a \in V\} \\
&\cup \{\nu < a \vee \nu = a \mid a \in V\} \\
&\cup \{0 < \nu \vee \nu = 0\} \\
&\cup \{\nu < 0 \vee \nu = 0\} \\
&\cup \{\neg(\nu = a) \mid a \in V\} \\
&\cup \{\neg(\nu = 0)\}
\end{aligned}$$

We can always extend the realm of predicates if the resulting refinements are too weak.

4.4.1 The Inference Algorithm

$\text{Infer} : \mathcal{P}(\mathcal{C}) \to (\mathcal{K} \twoheadrightarrow \mathcal{Q})$

$\text{Infer}(C) =$

$$\begin{aligned}
&\text{Let } V := \bigcup_{\hat{T}_1 <:_{\Theta,\Lambda} \hat{T}_2 \in C} \{a \mid (a, _) \in \Theta\} \\
&\quad Q_0 := \mathit{Init}(V), \\
&\quad A_0 := \{(\kappa, Q_0) \mid \kappa \in \bigcup_{c \in C} \text{Var}(c)\}, \\
&\quad A := \text{Solve}(\bigcup_{c \in C} \text{Split}(c), A_0) \\
&\text{in } \{(\kappa, \bigwedge Q) \mid (\kappa, Q) \in A\}
\end{aligned}$$

where $V \subseteq \mathcal{V}, Q_0, Q \subseteq \mathcal{Q}, A_0, A \in \mathcal{K} \twoheadrightarrow \mathcal{Q}, \Theta$ be a type variable context and $\Lambda \subseteq \mathcal{Q}$.

We first split the subtyping conditions for functions into subtyping conditions for simpler templates:

$$\begin{aligned}
\mathcal{C}^- := \{\ & \{\nu : \mathit{Int} | q_1\} <:_{\Theta,\Lambda} \{\nu : \mathit{Int} | q_2\} \\
& \mid (q_1 \in \mathcal{Q} \vee q_1 = [k_1]_{S_1} \text{ for } k_1 \in \mathcal{K}, S_1 \in \mathcal{V} \twoheadrightarrow \mathcal{Q}) \\
& \wedge (q_2 \in \mathcal{Q} \vee q_2 = [k_2]_{S_2} \text{ for } k_2 \in \mathcal{K}, S_2 \in \mathcal{V} \twoheadrightarrow \mathcal{Q})\}.
\end{aligned}$$

With this we can now define the Split function.

$$\begin{aligned}
&\text{Split} : \mathcal{C} \twoheadrightarrow \mathcal{P}(\mathcal{C}^-) \\
&\text{Split}(a : \{\nu : \mathit{Int} | q_1\} \to \hat{T}_2 <:_{\Theta,\Lambda} a : \{\nu : \mathit{Int} | q_3\} \to \hat{T}_4) = \\
&\qquad \text{Split}(\hat{T}_3 <:_{\Theta,\Lambda} \hat{T}_1) \cup \text{Split}(\hat{T}_2 <:_{\Theta \cup \{(a,q_3)\},\Lambda} \hat{T}_4\}) \\
&\text{Split}(\{\nu : \mathit{Int} | q_1\} <:_{\Theta,\Lambda} \{\nu : \mathit{Int} | q_2\}) = \\
&\qquad \{\{\nu : \mathit{Int} | q_1\} <:_{\Theta,\Lambda} \{\nu : \mathit{Int} | q_2\}\}
\end{aligned}$$

Note that Split will result in an error, if the subtyping condition is not one of the two cases above.

We resolve the obtained subtyping conditions by repeatably checking if a subtyping condition is not valid and removing all predicates that contradict it. By removing the predicate we weaken the resulting refinement.

$\mathrm{Solve} : \mathcal{P}(\mathcal{C}^-) \times (\mathcal{K} \nrightarrow \mathcal{Q}) \to (\mathcal{K} \nrightarrow \mathcal{Q})$

$\mathrm{Solve}(C, A) =$

Let $S := \{(k, \bigwedge Q) \mid (k, Q) \in A\}$.

If there exists $(\{v : Int \mid q_1\} <:_{\Theta,\Lambda} \{v : Int \mid [k_2]_{S_2}\}) \in C$ such that

$\neg(\forall i_1 \in \mathrm{value}_\Gamma(\{\nu : Int | r'_1\}). \ldots \forall i_n \in \mathrm{value}_\Gamma(\{\nu : Int | r'_n\}).$
$[\![r_1 \wedge p]\!]_{\{(b_1,i_1),\ldots,(b_n,i_n)\}} \Rightarrow [\![r_2]\!]_{\{(b_1,i_1),\ldots,(b_n,i_n)\}})$

for $r_2 := \bigwedge [S(\kappa_2)]_{S_2}$, $p := \bigwedge \Lambda$,

$$r_1 := \begin{cases} \bigwedge [S(q_1)]_{S_1} & \text{if } q_1 \in \mathcal{K} \\ q_1 & \text{if } q_1 \in \mathcal{Q} \end{cases},$$

$$\begin{aligned} \Theta' := \{\ & (a, r) \\ & \mid r \text{ has the form } q \wedge (a, q) \in \Theta \wedge q \in \mathcal{Q} \\ & \vee\ r \text{ has the form } [[k]_S]_{S_0} \wedge (a, q) \in \Theta \\ & \qquad \wedge\ q \text{ has the form } [k]_{S_0} \wedge k \in \mathcal{K} \wedge S_0 \in \mathcal{V} \nrightarrow \mathcal{Q}\} \end{aligned}$$

$\{(b_1, r'_1), \ldots, (b_n, r'_n)\} = \Theta'$

then $\mathrm{Solve}(C, \mathrm{Weaken}(c, A))$ else A

where $k, k_2 \in \mathcal{K}, S : \mathcal{K} \nrightarrow \mathcal{Q}, Q, \Lambda \subseteq \mathcal{Q}, S_1, S_2 : \mathcal{V} \nrightarrow \mathcal{Q}, q_1 \in \mathcal{K} \cup \mathcal{Q}$,
Θ be a type variable context, $r_1, p, r_2 \in \mathcal{Q}, a \in \mathcal{V}, \Theta' : \mathcal{V} \nrightarrow \mathcal{Q}, r \in \mathcal{Q}, n \in \mathbb{N}, b_i \in \mathcal{V}$,
$r_i \in \mathcal{Q}$ for $i \in \mathbb{N}_0^n$ and $[t]_A$ denotes the substitution for the term t with a substitution A.

Note that we can use a SMT solver to validate

$$\begin{aligned} &\neg(\forall i_1 \in \mathrm{value}_\Gamma(\{\nu : Int | r'_1\}). \ldots \forall i_n \in \mathrm{value}_\Gamma(\{\nu : Int | r'_n\}). \\ &\quad [\![r_1 \wedge p]\!]_{\{(b_1,i_1),\ldots,(b_n,i_n)\}} \Rightarrow [\![r_2]\!]_{\{(b_1,i_1),\ldots,(b_n,i_n)\}}) \end{aligned}$$

by deciding the satisfiablity of

$$(\bigwedge_{j=0}^{n} r'_j) \wedge r_1 \wedge p \wedge \neg r_2$$

with free variables $b_i \in \mathbb{N}$ for $i \in \mathbb{N}_1^n$.

$\text{Weaken} : \mathcal{C}^- \times (\mathcal{K} \rightarrowtail \mathcal{Q}) \rightarrowtail (\mathcal{K} \rightarrowtail \mathcal{Q})$
$\text{Weaken}(\{\nu : Int|x\} <:_{\Theta,\Lambda} \{\nu : Int|[k_2]_{S_2}\}, A) =$
Let $S := \{(k, \bigwedge Q) \mid (k, Q) \in A\}$,

$$r_1 := \begin{cases} \bigwedge [S(k_1)]_{S_1} & \text{if } x \text{ has the form } [k_1]_{S_1} \\ x & \text{if } x \in \mathcal{Q} \end{cases},$$

$p := \bigwedge \{[q]_S \mid q \in \Lambda\}$,
$\Theta' := \{\ (a, r)$
$\quad \mid r \text{ has the form } q \wedge (a, q) \in \Theta \wedge q \in \mathcal{Q}$
$\quad \vee\ r \text{ has the form } [[k]_S]_{S_0} \wedge (a, q) \in \Theta$
$\quad\quad \wedge\ q \text{ has the form } [k]_{S_0} \wedge k \in \mathcal{K} \wedge S_0 \in \mathcal{V} \rightarrowtail \mathcal{Q}\}$
$\{(b_1, r'_1), \ldots, (b_n, r'_n)\} = \Theta'$
$Q_2 := \{\ q$
$\quad \mid q \in A(k_2) \wedge \text{wellFormed}(q, \{(b_1, \{\nu : Int|r'_1\}), \ldots, (b_n, \{\nu : Int|r'_n\})\})$
$\quad \wedge (\forall i_1 \in \text{value}_\Gamma(\{\nu : Int|r'_1\}) \ldots \forall i_n \in \text{value}_\Gamma(\{\nu : Int|r'_n\}).$
$\quad\quad [\![r_1 \wedge p]\!]_{\{(b_1,i_1),\ldots,(b_n,i_n)\}} \Rightarrow [\![[q]_{S_2}]\!]_{\{(b_1,i_1),\ldots,(b_n,i_n)\}})\}$
in $\{(k, Q) \mid (k, Q) \in A \wedge k \neq k_2\} \cup \{(k_2, Q_2)\}$
where $k, k_1 \in \mathcal{K}, Q, Q_2 \subseteq \mathcal{Q}, S : \mathcal{K} \rightarrowtail \mathcal{Q}, r_1 \in \mathcal{Q}, p \in \mathcal{Q}, S_1, S_2 : \mathcal{V} \rightarrowtail \mathcal{Q}, \Theta' : \mathcal{V} \rightarrowtail \mathcal{T}$, $a \in \mathcal{V}, T' \in \mathcal{T} \cup \mathcal{T}^? n \in \mathbb{N}, b_i \in \mathcal{V}, T_i \in \mathcal{T}$ for $i \in \mathbb{N}_0^n$ and $[t]_A$ denotes the substitution for the term t with a substitution A.

Note that we can use a SMT solver to validate

$$\forall i_1 \in \text{value}_\Gamma(\{\nu : Int|r'_1\}) \ldots \forall i_n \in \text{value}_\Gamma(\{\nu : Int|r'_n\}).$$
$$[\![r_1 \wedge p]\!]_{\{(b_1,i_1),\ldots,(b_n,i_n)\}} \Rightarrow [\![[q]_{S_2}]\!]_{\{(b_1,i_1),\ldots,(b_n,i_n)\}}$$

To do so, we first need to compute $r_2 := [q]_{S_2}$, with that we can now use a SMT solver to decide the satisfiablity of

$$\neg((\bigwedge_{j=0}^{n} r'_j) \wedge r_1 \wedge p) \vee r_2$$

with free variables $b_i \in \mathbb{N}$ for $i \in \mathbb{N}_1^n$.

Example 4.1

Assume that we have given the following suptyping conditions:

$$\Theta := \{(a, \{Int|\kappa_1\}), (b, \{Int|\kappa_2\})\}$$
$$C_0 := \{\{\nu : Int|\nu = b\} <:_{\Theta,\{a<b\}} \{\nu : Int|\kappa_3\},$$
$$\{\nu : Int|\nu = a\} <:_{\Theta,\{\neg(a<b)\}} \{\nu : Int|\kappa_3\},$$
$$a : \{\nu : Int|\kappa_1\} \rightarrow b : \{\nu : Int|\kappa_2\} \rightarrow \{\nu : Int|\kappa_3\}$$
$$<:_{\{\},\{\}} a : \{\nu : Int|\, True\} \rightarrow b : \{\nu : Int|\, True\} \rightarrow \{\nu : Int|\kappa_4\}$$

Then $V := \{a, b\}$ and $A_0 := \{(\kappa_1, \mathit{Init}(V)), (\kappa_2, \mathit{Init}(V)), (\kappa_3, \mathit{Init}(V)), (\kappa_4, \mathit{Init}(V))\}$.

Splitting the conditions

We will only consider the last subtyping condition of C_0, all other conditions do not need to be split.

$$
\begin{aligned}
&\mathrm{Split}(a : \{\nu : \mathit{Int} | \kappa_1\} \to b : \{\nu : \mathit{Int} | \kappa_2\} \to \{\nu : \mathit{Int} | \kappa_3\} \\
&\quad <:_{\{\},\{\}} a : \{\nu : \mathit{Int} | \mathit{True}\} \to b : \{\nu : \mathit{Int} | \mathit{True}\} \to \{\nu : \mathit{Int} | \kappa_4\}) \\
&= \mathrm{Split}(a : \{\nu : \mathit{Int} | \kappa_1\} <:_{\{\},\{\}} a : \{\nu : \mathit{Int} | \mathit{True}\}) \\
&\quad \cup \mathrm{Split}(b : \{\nu : \mathit{Int} | \kappa_2\} \to \{\nu : \mathit{Int} | \kappa_3\} \\
&\qquad <:_{\{(a,\{\nu:\mathit{Int}|\mathit{True}\})\},\{\}} b : \{\nu : \mathit{Int} | \mathit{True}\} \to \{\nu : \mathit{Int} | \kappa_4\}) \\
&= \{a : \{\nu : \mathit{Int} | \mathit{True}\} <:_{\{\},\{\}} a : \{\nu : \mathit{Int} | \kappa_1\}\} \\
&\quad \cup \mathrm{Split}(b : \{\nu : \mathit{Int} | \mathit{True}\} <:_{\{(a,\{\nu:\mathit{Int}|\mathit{True}\})\},\{\}} b : \{\nu : \mathit{Int} | \kappa_2\}) \\
&\quad \cup \mathrm{Split}(\{\nu : \mathit{Int} | \kappa_3\} <:_{\Theta,\{\}} \{\nu : \mathit{Int} | \kappa_4\}) \\
&= \{\{\nu : \mathit{Int} | \mathit{True}\} <:_{\{\},\{\}} \{\nu : \mathit{Int} | \kappa_1\}, \\
&\quad \{\nu : \mathit{Int} | \mathit{True}\} <:_{\{(a,\{\nu:\mathit{Int}|\mathit{True}\})\},\{\}} \{\nu : \mathit{Int} | \kappa_2\}, \\
&\quad \{\nu : \mathit{Int} | \kappa_3\} <:_{\{\Theta\},\{\}} \{\nu : \mathit{Int} | \kappa_4\}\}
\end{aligned}
$$

So in conclusion we have the following set of subtypings conditions:

$$
\begin{aligned}
C := \{&\{\nu : \mathit{Int} | \nu = b\} <:_{\Theta,\{a<b\}} \{\nu : \mathit{Int} | \kappa_3\}, \\
&\{\nu : \mathit{Int} | \nu = a\} <:_{\Theta,\{\neg(a<b)\}} \{\nu : \mathit{Int} | \kappa_3\}, \\
&\{\nu : \mathit{Int} | \mathit{True}\} <:_{\{\},\{\}} \{\nu : \mathit{Int} | \kappa_1\}, \\
&\{\nu : \mathit{Int} | \mathit{True}\} <:_{\{(a,\{\nu:\mathit{Int}|\mathit{True}\})\},\{\}} \{\nu : \mathit{Int} | \kappa_2\}, \\
&\{\nu : \mathit{Int} | \kappa_3\} <:_{\Theta,\{\}} \{\nu : \mathit{Int} | \kappa_4\}\}
\end{aligned}
$$

We therefore now will go through each condition $c \in C$ and check its validity.

Iteration 1, Case $c = \{\nu : \mathit{Int} | \nu = b\} <:_{\Theta,\{a<b\}} \{\nu : \mathit{Int} | \kappa_3\}$:

We define $S := \{(\kappa_1, \bigwedge \mathit{Init}(V)), (\kappa_2, \bigwedge \mathit{Init}(V)), (\kappa_3, \bigwedge \mathit{Init}(V)), (\kappa_4, \bigwedge \mathit{Init}(V))\}$.

$\mathit{Init}(V)$ contains $\nu = 0$ and $\neg\nu = 0$, so we know that $\bigwedge \mathit{Init}(V)$ can be simplified to *False*.

We now check if

$$\begin{aligned}
&\forall a \in \text{values}_{\{\}}(\{\nu : \mathit{Int} | \mathit{False}\}).\\
&\forall b \in \text{values}_{\{\}}(\{\nu : \mathit{Int} | \mathit{False}\}).\\
&\forall \nu \in \text{values}_{\{\}}(\{\nu : \mathit{Int} | \mathit{True}\}).\\
&\nu = b \wedge a < b\\
&\models \forall a \in \text{values}_{\{\}}(\{\nu : \mathit{Int} | \mathit{False}\}).\\
&\quad \forall b \in \text{values}_{\{\}}(\{\nu : \mathit{Int} | \mathit{False}\}).\\
&\quad \forall \nu \in \text{values}_{\{\}}(\{\nu : \mathit{Int} | \mathit{True}\}).\\
&\quad \mathit{False}
\end{aligned}$$

is not valid.

We know that $\text{values}_{\{\}}(\{\nu : \mathit{False}\}) = \{\}$, and therefore this can be simplified to $\mathit{True} \models \mathit{True}$, which is valid.

Iteration 1, Case $c = \{\nu : \mathit{Int} | \nu = a\} <:_{\Theta,\{\neg(a<b)\}} \{\nu : \mathit{Int} | \kappa_3\}$:

The argument is analogously to the previous case.

Iteration 1, Case $c = \{\nu : \mathit{Int} | \mathit{True}\} <:_{\{\},\{\}} \{\nu : \mathit{Int} | \kappa_1\}$:

We now check if

$$\forall \nu \in \text{values}_{\{\}}(\{\nu : \mathit{Int} | \mathit{True}\}).\mathit{True} \models \forall \nu \in \text{values}_{\{\}}(\{\nu : \mathit{Int} | \mathit{True}\}).\mathit{False}$$

is valid. This time we can ignore the quantifiers and thus it simplifies to $\mathit{True} \models \mathit{False}$, which is not valid.

We therefore will now weaken A_0. To do so we compute all $q \in A_0(\kappa_1)$ such that wellFormed(q) and

$$\forall \nu \in \text{values}_{\{\}}(\{\nu : \mathit{Int} | \mathit{True}\}).[\![\mathit{True}]\!]_{\{\}} \models \forall \nu \in \text{values}_{\{\}}(\{\nu : \mathit{Int} | \mathit{True}\}).[\![q]\!]_{\{\}}.$$

There are only two values for q that are well formed: *True* and *False*.

The resulting set is $Q_2 := \{\mathit{True}\}$ and thus we replace A_0 with

$$A := \{(\kappa_1, \{\mathit{True}\}), (\kappa_2, \mathit{Init}(V)), (\kappa_3, \mathit{Init}(V)), (\kappa_4, \mathit{Init}(V))\}$$

Iteration 1, Case $c = \{\nu : \mathit{Int} | \mathit{True}\} <:_{\{(a,\{\nu:\mathit{Int}|\mathit{True}\})\},\{\}} \{\nu : \mathit{Int} | \kappa_2\}$:

The argument is analogously to the previous case, resulting in the updated value for A:

$$A = \{(\kappa_1, \{\mathit{True}\}), (\kappa_2, \{\mathit{True}\}), (\kappa_3, \mathit{Init}(V)), (\kappa_4, \mathit{Init}(V))\}$$

Iteration 1, Case $c = \{\nu : \mathit{Int} | \kappa_3\} <:_{\Theta,\{\}} \{\nu : \mathit{Int} | \kappa_4\}\}$:

The suptyping condition is valid, analogously to the first case of this iteration.

Iteration 2, Case $c = \{\nu : Int | \nu = b\} <:_{\Theta,\{a<b\}} \{\nu : Int | \kappa_3\}$:

We check the validity of

$$\begin{aligned}
&\forall a \in \text{values}_{\{\}}(\{\nu : Int | True\}).\\
&\forall b \in \text{values}_{\{\}}(\{\nu : Int | True\}).\\
&\forall \nu \in \text{values}_{\{\}}(\{\nu : Int | True\}).\\
&\nu = b \wedge a < b\\
&\models \forall a \in \text{values}_{\{\}}(\{\nu : Int | True\}).\\
&\quad \forall b \in \text{values}_{\{\}}(\{\nu : Int | True\}).\\
&\quad \forall \nu \in \text{values}_{\{\}}(\{\nu : Int | True\}).\\
&\quad False.
\end{aligned}$$

It is easy to see, that it is not valid.

Thus we now compute all $q \in A(\kappa_3)$ such that wellFormed(q) and

$$\begin{aligned}
&\forall a \in \text{values}_{\{\}}(\{\nu : Int | True\}).\\
&\forall b \in \text{values}_{\{\}}(\{\nu : Int | True\}).\\
&\forall \nu \in \text{values}_{\{\}}(\{\nu : Int | True\}).\\
&\nu = b \wedge a < b\\
&\models \forall a \in \text{values}_{\{\}}(\{\nu : Int | True\}).\\
&\quad \forall b \in \text{values}_{\{\}}(\{\nu : Int | True\}).\\
&\quad \forall \nu \in \text{values}_{\{\}}(\{\nu : Int | True\}).\\
&\quad q.
\end{aligned}$$

is valid. The resulting set Q_2 is the following.

$$\begin{aligned}
Q_2 := \{&a < \nu, \nu = b, \neg(\nu = a), \nu < b \vee \nu = b, b < \nu \vee \nu = b, \nu < a \vee \nu = a,\\
&a < \nu \vee \nu = a\}
\end{aligned}$$

Therefore we update A:

$$\begin{aligned}
A = \{&(\kappa_1, \{True\}), (\kappa_2, \{True\}),\\
&(\kappa_3, \{a < \nu, \nu = b, \neg(\nu = a), \nu < b \vee \nu = b, b < \nu \vee \nu = b, \nu < a \vee \nu = a,\\
&\quad a < \nu \vee \nu = a\}),\\
&(\kappa_4, Init(V))\}
\end{aligned}$$

Iteration 2, Case $c = \{\nu : Int | \nu = a\} <:_{\Theta,\{\neg(a<b)\}} \{\nu : Int | \kappa_3\}$:

We check the validity of

$$\begin{aligned}
&\forall a \in \text{values}_{\{\}}(\{\nu : Int | True\}).\\
&\forall b \in \text{values}_{\{\}}(\{\nu : Int | True\}).\\
&\forall \nu \in \text{values}_{\{\}}(\{\nu : Int | True\}).\\
&\nu = a \wedge \neg(a < b)\\
&\models \forall a \in \text{values}_{\{\}}(\{\nu : Int | True\}).\\
&\quad \forall b \in \text{values}_{\{\}}(\{\nu : Int | True\}).\\
&\quad \forall \nu \in \text{values}_{\{\}}(\{\nu : Int | True\}).\\
&\quad a < \nu \wedge \nu = b \wedge \neg(\nu = a) \wedge \nu < b \vee \nu = b\\
&\qquad \wedge b < \nu \vee \nu = b \wedge \nu < a \vee \nu = a \wedge a < \nu \vee \nu = a.
\end{aligned}$$

It is not valid, because $\nu = a \wedge \neg(a < b) \models \nu = b$ is not valid.

Thus we compute all $q \in A(\kappa_3)$ such that wellFromed(q) and

$$\begin{aligned}
&\forall a \in \text{values}_{\{\}}(\{\nu : Int | True\}).\\
&\forall b \in \text{values}_{\{\}}(\{\nu : Int | True\}).\\
&\forall \nu \in \text{values}_{\{\}}(\{\nu : Int | True\}).\\
&\nu = a \wedge \neg(a < b)\\
&\models \forall a \in \text{values}_{\{\}}(\{\nu : Int | True\}).\\
&\quad \forall b \in \text{values}_{\{\}}(\{\nu : Int | True\}).\\
&\quad \forall \nu \in \text{values}_{\{\}}(\{\nu : Int | True\}).\\
&\quad q.
\end{aligned}$$

is valid. The resulting set Q_2 is the following.

$$Q_2 := \{\nu < b \vee \nu = b, \nu < a \vee \nu = a\}$$

Thus we update A:

$$A = \{(\kappa_1, \{True\}), (\kappa_2, \{True\}), \quad (\kappa_3, \{\nu < b \vee \nu = b, \nu < a \vee \nu = a\}), (\kappa_4, Init(V))\}$$

Iteration 2, Case $\{\nu : Int | True\} <:_{\{\},\{\}} \{\nu : Int | \kappa_1\}$:

Nothing has changed since the last iteration, therefore this case can be skipped.

Iteration 2, Case $\{\nu : Int | True\} <:_{\{(a,\{\nu:Int|True\})\},\{\}} \{\nu : Int | \kappa_2\}$:

The argument is analogously to the previous case, therefore this case can be skipped.

Iteration 2, Case : $\{\nu : Int | \kappa_3\} <:_{\Theta,\{\}} \{\nu : Int | \kappa_4\}$:

We check the validity of

$$\begin{aligned}
&\forall a \in \text{values}_{\{\}}(\{\nu : Int | True\}).\\
&\forall b \in \text{values}_{\{\}}(\{\nu : Int | True\}).\\
&\forall \nu \in \text{values}_{\{\}}(\{\nu : Int | True\}).\\
&\{\nu < b \vee \nu = b \wedge \nu < a \vee \nu = a\}\\
&\vDash \forall a \in \text{values}_{\{\}}(\{\nu : Int | True\}).\\
&\quad \forall b \in \text{values}_{\{\}}(\{\nu : Int | True\}).\\
&\quad \forall \nu \in \text{values}_{\{\}}(\{\nu : Int | True\}).\\
&\quad False.
\end{aligned}$$

We see that this is not valid, therefore we derive the new set Q_2. Note that $A(\kappa_3) \subseteq Init(V)$ and therefore $Q_2 = A(\kappa_3)$.

We update the corresponding entry in A:

$$\begin{aligned}
A = \{&(\kappa_1, \{True\}), (\kappa_2, \{True\}),\\
&(\kappa_3, \{\nu < b \vee \nu = b, \nu < a \vee \nu = a\}),\\
&(\kappa_4, \{\nu < b \vee \nu = b, \nu < a \vee \nu = a\})\}
\end{aligned}$$

Iteration 3:

In this iteration all subtyping conditions are valid, thus the algorithm stops. The resulting set of substitutions is therefore the following

$$\begin{aligned}
&\{(\kappa_1, True), (\kappa_2, True),\\
&(\kappa_3, (\nu < b \vee \nu = b) \wedge (\nu < a \vee \nu = a)),\\
&(\kappa_4, (\nu < b \vee \nu = b) \wedge (\nu < a \vee \nu = a))\}
\end{aligned}$$

4.4.2 Correctness

The algorithm that we described can fail if the subtyping conditions are not well-formed.

Definition 4.6: Wellformed Subtyping Condition

We say a subtyping condition c is wellformed if the following holds

$$\begin{aligned}
&c \text{ has the form } \{\nu : Int | [k_1]_S\} <:_{\Theta,\Lambda} \{\nu : Int | [k_2]_S\}\\
\vee\ &c \text{ has the form } \{\nu : Int | r\} <:_{\Theta,\Lambda} \{\nu : Int | [k_2]_S\}\\
\vee\ &c \text{ has the form } \{\nu : Int | [k_1]_S\} \to \hat{T}_1 <:_{\Theta,\Lambda} \{\nu : Int | r\} \to \hat{T}_2\\
&\quad \text{such that } \hat{T}_1 <:_{\Theta,\Lambda} \hat{T}_2 \text{ is wellformed.}
\end{aligned}$$

where $r \in \mathcal{Q}$,$k_1, k_2 \in \mathcal{K}$, $S : \mathcal{V} \nrightarrow \mathcal{Q}$,$\Theta$ is a type variable context, $\Lambda \subset \mathcal{Q}$ and $T_1, T_2 \in (\mathcal{T} \cup \mathcal{T}^?)$

Theorem 4.1

Let C be a set of wellformed conditions, $S := \text{Infer}(C)$ and $V := \bigcup_{\hat{T}_1 <:_{\Theta,\Lambda} \hat{T}_2 \in C} \{a \mid (a, \_) \in \Theta\}$

—

For all subtyping condition $(\hat{T}_1 <:_{\Theta,\Lambda} \hat{T}_2) \in C$,

$$
\begin{aligned}
&[\hat{T}_1]_S \in \mathcal{T} \wedge [\hat{T}_2]_S \in \mathcal{T} \\
\wedge &[\hat{T}_1]_S <:_{\Theta,\Lambda} [\hat{T}_2]_S \\
\wedge &\forall S' \in (\mathcal{V} \to \mathcal{Q}).(\forall a.\exists Q \in \mathit{Init}(V).S'(a) = \bigwedge Q) \\
&\quad \wedge [\hat{T}_1]_{S'} \in \mathcal{T} \wedge [\hat{T}_2]_{S'} \in \mathcal{T} \wedge ([\hat{T}_1]_{S'} <:_{\Theta,\Lambda} [\hat{T}_2]_{S'} \Rightarrow (\forall a.S(a) \Rightarrow S'(a)))
\end{aligned}
$$

The first condition states that $[\hat{T}_1]_S$ and $[\hat{T}_2]_S$ are not templates. The second condition states that $[\hat{T}_1]_S$ is a subtype of $[\hat{T}_2]_S$. The third condition states that S is the sharpest solution.

Theorem 4.2

Let c is a wellformed condition and $C := \text{Split}(c)$.

—

Then $C \subseteq \mathcal{C}^- \wedge \forall c \in C.c$ is a wellformed condition.

Theorem 4.3

Let $C \subseteq \mathcal{C}^-$ be a set of well-formed conditions, $A_1, A_2 : \mathcal{K} \nrightarrow \mathcal{Q}$. Let $A_2 = \text{Solve}(C, A_1)$ and $S = \{(\kappa, \bigwedge Q) \mid (\kappa, Q) \in A\}$

—

For all subtyping condition $(\hat{T}_1 <:_{\Theta,\Lambda} \hat{T}_2) \in C$,

$$
\begin{aligned}
&[\hat{T}_1]_S \in \mathcal{T} \wedge [\hat{T}_2]_S \in \mathcal{T} \\
\wedge &[\hat{T}_1]_S <:_{\Theta,\Lambda} [\hat{T}_2]_S \\
\wedge &\forall S' \in (\mathcal{V} \to \mathcal{Q}).(\forall a.\exists Q \in \mathit{Init}(V).S'(a) = \bigwedge Q) \\
&\quad \wedge [\hat{T}_1]_{S'} \in \mathcal{T} \wedge [\hat{T}_2]_{S'} \in \mathcal{T} \wedge ([\hat{T}_1]_{S'} <:_{\Theta,\Lambda} [\hat{T}_2]_{S'} \Rightarrow (\forall a.S(a) \Rightarrow S'(a)))
\end{aligned}
$$

The post-condition is the same as for Infer.