# 1 Aliasing (Old)

The **alias status** of a set of identifiers $\{x_\alpha\}$ is exactly one of the following: aliases, non-aliases, undetermined-aliases.

- The $x_\alpha$ are aliases if each $x_\alpha$ refers to the same memory in the heap.
- The $x_\alpha$ are non-aliases if each $x_\alpha$ refers to distinct memory in the heap.
- The $x_\alpha$ are undermined-aliases if they may be aliases or non-aliases.

An **alias class** is a pair $[S, I]$ where $S$ is an alias status and $I$ is a set of identifiers where the identifiers of $I$ have alias status $S$. $\mathsf{identifiers}(\{[S_\alpha, I_\alpha]\}) := \bigcup I_\alpha$ is the set of identifiers of a set of alias classes. It is possible to keep track of undermined-aliases classes. However, for the sake of efficiency, some give identifiers are considered undermined-aliases if no subset of them are asserted as aliases nor non-aliases by any alias class.

A set of alias classes $\{[S_\alpha, I_\alpha]\}$ is **overlapping** if and only if

$$\bigcup I_\alpha \neq \varnothing.$$

A set of alias statuses $\{S_\alpha\}$ is **compatible** if and only if

$$\forall S \in \{S_\alpha\} : \forall \alpha : S_\alpha = S$$

A set of alias classes $A$ is **compatible** if and only if

$$\forall \{[S_\alpha, I_\alpha]\} \subset A : \{[S_\alpha, I_\alpha]\} \text{ is overlapping} \implies \{S_\alpha\} \text{ is compatible}$$

This is to say that a set of compabile alias classes must not assert that a pair of identifiers are both aliases and non-aliases — every overlapping set of alias classes is compatible.

Given two compatible sets of alias classes $A, A'$, the compatibility of $A \cup A'$ can be considered, written $A \uplus A'$. Deciding $A \uplus A'$ reduces to computing $\mathsf{simplify}(A \cup A')$ which either preserves compatibility or raises an exception, where

$$\begin{aligned}
\mathsf{simplify}(\{[S_\alpha, I_\alpha]\}) := & \left\{ \left[S, \bigcup I_{\alpha_i}\right] \;\middle|\; \forall \alpha : \{\alpha_i\} = \mathsf{LOS}(\alpha) \wedge ((\forall i : S = S_{\alpha_i}) \vee (\mathsf{raise\ exception})) \right\}, \\
\mathsf{LOS}(\alpha) := & \{\alpha_i\}, \text{ the largest subset of } \{\alpha\} \\
& \text{such that } \alpha \in \{\alpha_i\} \text{ and } \{I_{\alpha_i}\} \text{ is overlapping.}
\end{aligned}$$

For each set of overlapping alias classes, simplify either combines then or throws an exception.

### 1.1 Deciding Alias Class Compatibility

A set of alias classes $A$ is **compatible** with a formula $\phi$ if and only if $A$ is compatible with the aliasing assertions yielded by $\phi$, written $A \uplus \phi$. The following algorithm decides $A \uplus \phi$.

$$A \uplus \phi \iff \textsf{match } \phi \textsf{ with} \left| \begin{array}{lll} e & \mapsto & A \uplus \textsf{asserted}(e) \\ \texttt{acc}(e.f) & \mapsto & A \uplus \{[\textsf{non-aliases}, \{e\} \cup \textsf{identifiers}(A)]\} \\ \phi_1 \circledast \phi_2 & \mapsto & A \uplus \textsf{asserted}(\phi_1) \uplus \textsf{asserted}(\phi_2) \\ \phi_1 \wedge \phi_2 & \mapsto & (A \uplus \phi_1) \wedge (A \uplus \phi_2) \\ \texttt{if } e \texttt{ then } \phi_1 \texttt{ else } \phi_2 & \mapsto & \textsf{if } A \uplus \textsf{asserted}(e) \\ & & \ \textsf{then } A \uplus \textsf{asserted}(e) \uplus \textsf{asserted}(\phi_1) \\ & & \ \textsf{else } A \uplus \neg\ \textsf{asserted}(e) \uplus \textsf{asserted}(\phi_2) \\ \phi & \mapsto & A \uplus \textsf{asserted}(\phi) \end{array} \right.$$

where

$$\begin{aligned} \neg A &:= \{[\neg S_\alpha, I_\alpha] \ \mid \ [S_\alpha, I_\alpha] \in A\}\,, \\ \neg\ \textsf{aliases} &:= \textsf{non-aliases}, \\ \neg\ \textsf{non-aliases} &:= \textsf{aliases}. \end{aligned}$$

The following algorithm collects the set of alias classes asserted by a given formula $\phi$.

$$\textsf{asserted}(\phi) := \textsf{match } \phi \textsf{ with} \left| \begin{array}{lll} x = y & \mapsto & \{[\textsf{aliases}, \{x, y\}]\} \\ x \odot y & \mapsto & \{[\textsf{non-aliases}, \{e\}]\} \\ e & \mapsto & \varnothing \\ \texttt{unfolding } \alpha(\overline{e}) \texttt{ in } \phi & \mapsto & A \uplus \phi \end{array} \right.$$

The following algorithm collects the set of identifiers in a given set of alias classes $A$.

$$\textsf{identifiers}(\{[S_\alpha, I_\alpha]\}) := \bigcup I_\alpha$$