

1 Language \mathcal{L}

In this section we define an imperative language \mathcal{L} with memory allocation and deallocation primitives, and for simplification we only use pointers as values.

The syntax of the language \mathcal{L} is as follows.

x, y, z, \dots (variables)	\in	Var
s (statements)	$::=$	$\mathbf{skip} \mid s_1; s_2 \mid *x \leftarrow y \mid \mathbf{free}(x)$ $\mid \mathbf{let } x = \mathbf{malloc}() \mathbf{ in } s \mid \mathbf{let } x = \mathbf{null} \mathbf{ in } s$ $\mid \mathbf{let } x = y \mathbf{ in } s \mid \mathbf{let } x = *y \mathbf{ in } s$ $\mid \mathbf{ifnull } (*x) \mathbf{ then } s_1 \mathbf{ else } s_2 \mid f(\vec{x})$ $\mid \mathbf{const}(*x)s \mid \mathbf{endconst}(*x)$
d (proc. defs.)	$::=$	$\{f \mapsto (x_1, \dots, x_n)s\}$
D (definitions)	$::=$	$\langle d_1 \cup \dots \cup d_n \rangle$
P (programs)	$::=$	$\langle D, s \rangle$

Notation \vec{x} is for a finite sequence $\{x_1, \dots, x_n\}$, where we assume that each element is distinct; $[\vec{x}'/\vec{x}]s$ is for a term obtained by replacing each free occurrence of \vec{x} in s with variables \vec{x}' .

The **Var** is a countably infinite set of *variables* and each variable is a pointer. The statement **skip** means "does nothing". The statement $s_1; s_2$ is a sequential execution of s_1 and s_2 . The statement $*x \leftarrow y$ updates the content of cell which is pointed to by x with the value y . The statement **free**(x) deallocates a memory cell which is pointed to by pointer x . The statement **let** $x = e$ **in** s evaluates the expression e , binds x to the result, and executes s . The expression **malloc**() allocates a new memory cell. The expression **null** evaluates to the null pointer. The expression $*y$ means dereferencing a memory cell pointed to by y . The statement **ifnull** ($*x$) **then** s_1 **else** s_2 executes s_1 if $*x$ is **null** and executes s_2 otherwise. The statement $f(\vec{x})$ expresses a procedure f with arguments \vec{x} . The statement **const**($*x$) s means ($*x$) is a constant in statement s ; the statement **endconst**($*x$) means from this point ($*x$) maybe not constant.

The d represents a procedure definition which maps a procedure name f to its procedure body $(\vec{x})s$; The D represents a set of procedure definitions $\langle d_1 \cup \dots \cup d_n \rangle$, and each definition is distinct; The pair $\langle D, s \rangle$ represents a program, where D is a set of definitions and s is a main statement; the E represents evaluation context.

1.1 Operational semantics

In this section we introduce operational semantics of language \mathcal{L} . We assume there is a countable infinite set \mathcal{H} of *heap addresses* ranged over by l .

We use a configuration $\langle H, R, s, n, C \rangle$ to express a run-time state. Each elements in the configuration is as follows.

- H , a *heap*, is a finite mapping from \mathcal{H} to $\mathcal{H} \cup \{\mathbf{null}\}$;

- R , an *environment*, is a finite mapping from **Var** to $\mathcal{H} \cup \{\mathbf{null}\}$;
- s is the statement that is being executed;
- n is a natural number that represents the number of memory cells available for allocation.
- C is a set of actions, which contains **const**(*x), **null**(*x) and $\neg\mathbf{null}(*x)$.

The operational semantics of the language \mathcal{L} is given by a labeled transition relation $\langle H, R, s, n, C \rangle \xrightarrow{\rho}_D \langle H', R', s', n', C' \rangle$. The label ρ is as follows.

$$\rho \text{ (label)} ::= \mathbf{malloc} \mid \mathbf{free} \mid \mathbf{null}(*x) \mid \neg\mathbf{null}(*x) \mid \tau$$

The ρ , an *action*, is **malloc**, **free**, or τ . The action **malloc** expresses an allocation of a memory cell; **free** expresses a deallocation of a memory cell; τ expresses the other actions. We often omit τ in $\xrightarrow{\tau}_D$. We use a metavariable σ for a finite sequence of actions $\rho_1 \dots \rho_n$. We write $\xrightarrow{\rho_1 \dots \rho_n}_D$ for $\xrightarrow{\rho_1}_D \xrightarrow{\rho_2}_D \dots \xrightarrow{\rho_n}_D$. We write $\xRightarrow{\rho}_D$ for $\xrightarrow{*}_D \xrightarrow{\rho}_D \xrightarrow{*}_D$. We write $\xRightarrow{\rho_1 \dots \rho_n}_D$ for $\xRightarrow{\rho_1}_D \dots \xRightarrow{\rho_n}_D$.

Notation the **Dom**(f) is a mapping from function name f to its domain; for a map f , the $f\{x \mapsto v\}$ and $f \setminus x$ are defined as follows:

$$\begin{aligned} f\{x \mapsto v\}(w) &= \begin{cases} v & \text{if } x = w \\ f(w) & \text{otherwise.} \end{cases} \\ (f \setminus x)(w) &= \begin{cases} \text{undefined} & \text{if } x = w \\ f(w) & \text{otherwise.} \end{cases} \end{aligned}$$

and $filter(C, *x)$ is defined by a pseudocode as follows:

$$\begin{aligned} filter(C, *x) &= \text{let } C' = C - \mathbf{const}(*x) \text{ in} \\ &\quad \text{if } \mathbf{const}(*x) \in C' \text{ then return } C' \\ &\quad \text{else return } C' \setminus \{\mathbf{null}(*x), \neg\mathbf{null}(*x)\} \end{aligned}$$

Figure 1 depicts the relation $\xrightarrow{\rho}_D$. Several important rules are listed as follows.

- SEM-CONSTSKIP: That a memory cell pointed to by x is no longer a constant is expressed by doing nothing.
- SEM-CONSTSEQ: That a memory cell pointed to by x should be a constant in a statement s is expressed by adding a statement **endconst**(*x) at the end of statement s .
- SEM-FREE: Deallocation of a memory cell pointed to by x is expressed by deleting the entry for $R(x)$ from the heap. This action increments the number of available cells (i.e., n) by one (i.e., $n + 1$).
- SEM-MALLOC and SEM-OUTOFMEM: Allocation of a memory cell is expressed by adding a fresh entry to the heap. This action is allowed only if the number of available cells is positive; if the number is zero, then the configuration leads to an error state **OutOfMemory**.

- SEM-ASSIGNEXN, SEM-FREEEXN, SEM-DEREFEXN and SEM-FREEEXN : These rules express an illegal access to memory. If such action is performed, then the configuration leads to exceptional state **MemEx**. This state **MemEx** is not seen as an erroneous state in the current paper, hence a well-typed program may lead to these states. The command **free**(x), if x is a null pointer, leads to **MemEx** in the current semantics, although it is equivalent to **skip** in the C language.
- SEM-CONSTEXN: expresses that if a constant $*x$ is changed in s it will raise **ConstEx** exception.

Our goal is to guarantee *total* memory-leak freedom and reject memory leaks. By our language \mathcal{L} , they are formally defined as follows:

Definition 1 (total memory-leak freedom). *A program $\langle D, s \rangle$ is totally memory-leak free if there is a natural number n such that it does not require more than n cells.*

Definition 2 (Memory leak). *A configuration $\langle H, R, s, n, C \rangle$ goes overflow if there is σ such that $\langle H, R, s, n, C \rangle \xrightarrow{\sigma} \text{OutOfMemory}$. A program $\langle D, s \rangle$ consumes at least n cells if $\langle \emptyset, \emptyset, s, n, \emptyset \rangle$ goes overflow.*

2 Type system

2.1 Types

The syntax of the types is as follows.

$$\begin{array}{ll}
P \text{ (behavioral types)} & ::= \mathbf{0} \mid P_1; P_2 \mid \mathbf{malloc} \mid \mathbf{free} \mid \alpha \mid \mu\alpha.P \\
& \quad \mid (x)P \mid (*x)(P_1, P_2) \mid \mathbf{const}(*x)P \mid \mathbf{endconst}(*x) \\
\Gamma \text{ (variable type environment)} & ::= \{x_1, x_2, \dots, x_n\} \\
\Psi \text{ (dependent function type)} & ::= (\vec{x})P \\
\Theta \text{ (function type environment)} & ::= \{f_1 : \Psi_1, \dots, f_n : \Psi_n\}
\end{array}$$

Behavioral types ranged over by P express the abstraction of behaviors of a program. The type $\mathbf{0}$ represents the do-nothing behavior; the type $P_1; P_2$ represents the sequential execution of P_1 and P_2 ; The type **malloc** represents an allocation of a memory cell exactly once; the type **free** represents a deallocation; the type $\mu\alpha.P$ represents the behavior of α defined by the recursive equation $\alpha = P$; the type $(*x)(P_1, P_2)$ represents that P_1 or P_2 is obtained dependent on $*x$; the type $P_1 + P_2$ represents the choice between P_1 and P_2 ; the α is a type variable; the type **const**($*x$) P represents that $*x$ is a constant in behavioral type P ; the type **endconst**($*x$) represents $*x$ no longer be a constant from this point.

A type environments for variables ranged over by Γ is a set of variables. Since our interest is the behavior of a program, not the types of values, a variable type environment does not carry information on the types of variables.

Dependent function types ranged over by Ψ represents the behavior of a function; \vec{x} is the formal arguments of the function.

Function types ranged over by Θ is a mapping from function names to dependent function types.

$$\begin{array}{c}
\frac{C' = \text{filter}(C, *x)}{\langle H, R, \text{endconst}(*x), n, C \rangle \rightarrow_D \langle H, R, \text{skip}, n, C' \rangle} \quad (\text{SEM-CONSTSKIP}) \\
\langle H, R, \text{const}(*x)s, n, C \rangle \rightarrow_D \langle H, R, s; \text{endconst}(*x), n, C \cup \{\text{const}(*x)\} \rangle \quad (\text{SEM-CONSTSEQ}) \\
\langle H, R, \text{skip}; s, n, C \rangle \rightarrow_D \langle H, R, s, n, C \rangle \quad (\text{SEM-SKIP}) \\
\frac{\langle H, R, s_1, n, C \rangle \xrightarrow{\rho}_D \langle H', R', s'_1, n', C' \rangle}{\langle H, R, s_1; s_2, n, C \rangle \xrightarrow{\rho}_D \langle H', R', s'_1; s_2, n', C' \rangle} \quad (\text{SEM-SEQ}) \\
\frac{x' \notin \text{Dom}(R)}{\langle H, R, \text{let } x = \text{null in } s, n, C \rangle \rightarrow_D \langle H, R \{x' \mapsto \text{null}\}, [x'/x]s, n, C \rangle} \quad (\text{SEM-LETNULL}) \\
\frac{x' \notin \text{Dom}(R)}{\langle H, R, \text{let } x = y \text{ in } s, n, C \rangle \rightarrow_D \langle H, R \{x' \mapsto R(y)\}, [x'/x]s, n, C \rangle} \quad (\text{SEM-LETEQ}) \\
\frac{H(R(x)) = \text{null}, \text{const}(*x) \notin C}{\langle H, R, \text{ifnull}(*x) \text{ then } s_1 \text{ else } s_2, n, C \rangle \xrightarrow{\text{null}(*x)}_D \langle H, R, s_1, n, C \rangle} \quad (\text{SEM-IFNULLT}) \\
\frac{H(R(x)) \neq \text{null}, \text{const}(*x) \notin C}{\langle H, R, \text{ifnull}(*x) \text{ then } s_1 \text{ else } s_2, n, C \rangle \xrightarrow{\neg \text{null}(*x)}_D \langle H, R, s_2, n, C \rangle} \quad (\text{SEM-IFNULLF}) \\
\frac{H(R(x)) = \text{null}, \text{const}(*x) \in C}{\langle H, R, \text{ifnull}(*x) \text{ then } s_1 \text{ else } s_2, n, C \rangle \xrightarrow{\text{null}(*x)}_D \langle H, R, s_1, n, C \cup \{\text{null}(*x)\} \rangle} \quad (\text{SEM-IFCONSTNULLT}) \\
\frac{H(R(x)) \neq \text{null}, \text{const}(*x) \in C}{\langle H, R, \text{ifnull}(*x) \text{ then } s_1 \text{ else } s_2, n, C \rangle \xrightarrow{\neg \text{null}(*x)}_D \langle H, R, s_2, n, C \cup \{\neg \text{null}(*x)\} \rangle} \quad (\text{SEM-IFCONSTNULLF}) \\
\frac{\forall z. R(x) = R(z) \Rightarrow \text{const}(*x) \notin C}{\langle H \{R(x) \mapsto v\}, R, *x \leftarrow y, n, C \rangle \rightarrow_D \langle H \{R(x) \mapsto R(y)\}, R, \text{skip}, n, C \rangle} \quad (\text{SEM-ASSIGN}) \\
\frac{x' \notin \text{Dom}(R) \quad R(y) \in \text{Dom}(H)}{\langle H, R, \text{let } x = *y \text{ in } s, n, C \rangle \rightarrow_D \langle H, R \{x' \mapsto H(R(y))\}, [x'/x]s, n, C \rangle} \quad (\text{SEM-LETDEREF}) \\
\frac{R(x) \neq \text{null} \text{ and } R(x) \in \text{Dom}(H)}{\langle H \{R(x) \mapsto v\}, R, \text{free}(x), n, C \rangle \xrightarrow{\text{free}}_D \langle H \setminus R(x), R, \text{skip}, n+1, C \rangle} \quad (\text{SEM-FREE}) \\
\frac{l \notin \text{Dom}(H) \quad n > 0 \quad x' \notin \text{Dom}(H) \cup \text{Dom}(R) \cup \text{fv}(C)}{\langle H, R, \text{let } x = \text{malloc}() \text{ in } s, n, C \rangle \xrightarrow{\text{malloc}}_D \langle H \{l \mapsto v\}, R \{x' \mapsto l\}, [x'/x]s, n-1, C \rangle} \quad (\text{SEM-MALLOC}) \\
\frac{D(f) = (\vec{y})s}{\langle H, R, f(\vec{x}), n, C \rangle \rightarrow_D \langle H, R, [\vec{x}/\vec{y}]s, n, C \rangle} \quad (\text{SEM-CALL}) \quad \frac{R(x) = \text{null} \text{ or } R(x) \notin \text{Dom}(H)}{\langle H, R, \text{free}(x), n, C \rangle \xrightarrow{\text{free}}_D \text{MemEx}} \quad (\text{SEM-FREEEXN}) \\
\frac{R(x) = \text{null} \text{ or } R(x) \notin \text{Dom}(H)}{\langle H, R, *x \leftarrow y, n, C \rangle \rightarrow_D \text{MemEx}} \quad (\text{SEM-ASSIGNEXN})_4 \quad \frac{R(y) = \text{null} \text{ or } R(y) \notin \text{Dom}(H)}{\langle H, R, \text{let } x = *y \text{ in } s, n, C \rangle \rightarrow_D \text{MemEx}} \quad (\text{SEM-DEREFEXN}) \\
\frac{\exists z. \text{const}(*z) \in C \text{ and } R(x) = R(z)}{\langle H \{R(x) \mapsto v\}, R, *x \leftarrow y, n, C \rangle \rightarrow_D \text{ConstEx}} \quad (\text{SEM-ASSIGNCONSTEXN}) \\
\langle H, R, \text{let } x = \text{malloc}() \text{ in } s, 0, C \rangle \xrightarrow{\text{malloc}}_D \text{OutOfMemory} \quad (\text{SEM-OUTOFMEM})
\end{array}$$

Figure 1: Operational semantics of \mathcal{L} .

Figure 2 depicts semantics of behavioral types with dependent types, and they are given by the labeled transition system. The relation $\langle P, C \rangle \xrightarrow{\rho} \langle P', C' \rangle$ means that P can make an action ρ , and P turns into P' after it makes action ρ ; C and C' record constant value environment before and after making action ρ respectively.

2.2 Typing rules

The type judgment for statements is of the form $\Theta; \Gamma \vdash s : P$, which represents that under the function type environment Θ and the variable type environment Γ , the abstracted behavioral type of statement s is P .

Before showing typing rules for statements in Figure 3, we need explain several important definitions. The first one is $OK_n(P, C)$, a predicate, where P represents the behavior of a program which consumes at most n memory cells under constant value environment C .

Definition 3 ($\#_\rho(\sigma)$). $\#_\rho(\sigma)$ is the number of ρ in the sequence σ .

Definition 4. $OK_n(P, C)$ holds if $\forall P'$ and σ . if $\langle P, C \rangle \xrightarrow{\sigma} \langle P', C' \rangle$, then $\#_m(\sigma) - \#_f(\sigma) \leq n$

Intuitively, $OK_n(P, C)$ represents at very running steps, the number of memory cells a program consumed will not exceed the number of memory cells the program requires.

Definition 5 (Subtyping). $C \vdash P_1 \leq P_2$ is the largest relation such that, for any P'_1, C' and ρ , if $\langle P_1, C \rangle \xrightarrow{\rho} \langle P'_1, C' \rangle$, then there exists P'_2 such that $\langle P_2, C \rangle \xRightarrow{\rho} \langle P'_2, C' \rangle$ and $C' \vdash P'_1 \leq P'_2$. We write $P_1 \leq P_2$ if $C \vdash P_1 \leq P_2$ for any C .

Figure 3 shows the typing rules. For example, the rule T-IFNULL represents the behavior of **ifnull** $(*x)$ **then** s_1 **else** s_2 is abstracted as $(*x)(P_1, P_2)$ where P_1 and P_2 are the behavior of s_1 and s_2 respectively; this conditional statement means that executing s_1 if $(*x)$ is a null pointer, otherwise s_2 . The typing rule T-PROGRAM represents a program requires at most n memory cells during running under the predication $OK_n(P, C)$, where P is behavioral type of statement s .

2.3 Type soundness

Theorem 2.1. If $\vdash \langle D, s \rangle : n$ for some n , then $\langle D, s \rangle$ is totally memory-leak free.

The proof is based on the following lemmas: preservation and lack of immediate overflow.

Definition 6. *consistency*(H, R, C): for all x . (1) if **null**($*x$) $\in C$, then **const**($*x$) $\in C$ and $H(R(x)) = \text{null}$ if $H(R(x))$ is defined (3) if $\neg \text{null}(*x) \in C$, then **const**($*x$) $\in C$ and $H(R(x)) \neq \text{null}$ if $H(R(x))$ is defined.

Definition 7. we write $\Theta \vdash \langle H, R, s, n, C \rangle : \langle P, C \rangle$, if there exists Γ such that $\Theta; \Gamma \vdash s : P$, $OK_n(P, C)$, *consistency*(H, R, C) and $\Gamma \subseteq \text{Dom}(R)$.

Lemma 2.2 (Preservation). suppose that $\Theta \vdash \langle H, R, s, n, C \rangle : \langle P, C \rangle$, if $\langle H, R, s, n, C \rangle \xrightarrow{\rho} \langle H', R', s', n', C' \rangle$ then $\exists P'$ and C' s.t. (1) $\Theta \vdash \langle H', R', s', n', C' \rangle : \langle P', C' \rangle$ and (2) $\langle P, C \rangle \xRightarrow{\rho} \langle P', C' \rangle$.

Lemma 2.3 (Lack of immediate overflow). If $\Theta \vdash \langle H, R, s, n, C \rangle : \langle P, C \rangle$, then $\langle H, R, s, n, C \rangle \not\xrightarrow{\text{malloc}} \text{OutOfMemory}$.

iiiiiii HEAD

$$\begin{array}{c}
\langle \mathbf{0}; P, C \rangle \rightarrow \langle P, C \rangle \quad (\text{TR-SKIP}) \\
\langle \mathbf{free}, C \rangle \xrightarrow{\mathbf{free}} \langle \mathbf{0}, C \rangle \quad (\text{TR-FREE}) \quad \langle \mu\alpha.P, C \rangle \rightarrow \langle [\mu\alpha.P/\alpha]P, C \rangle \quad (\text{TR-REC}) \\
\langle P_1 + P_2, C \rangle \rightarrow \langle P_1, C \rangle \quad (\text{TR-CHOICEL}) \quad \langle P_1 + P_2, C \rangle \rightarrow \langle P_2, C \rangle \quad (\text{TR-CHOICER}) \\
\frac{\langle P_1, C \rangle \xrightarrow{\rho} \langle P'_1, C' \rangle}{\langle P_1; P_2, C \rangle \xrightarrow{\rho} \langle P'_1; P_2, C' \rangle} \quad (\text{TR-SEQ}) \\
\langle \mathbf{malloc}, C \rangle \xrightarrow{\mathbf{malloc}} \langle \mathbf{0}, C \rangle \quad (\text{TR-MALLOC}) \\
\frac{x' \text{ is fresh}}{\langle \langle x \rangle P, C \rangle \rightarrow \langle [x'/x]P, C \rangle} \quad (\text{TR-BIND}) \\
\langle \mathbf{const}(*x)P, C \rangle \rightarrow \langle P; \mathbf{endconst}(*x), C \cup \{\mathbf{const}(*x)\} \rangle \quad (\text{TR-CONST}) \\
\frac{C' = \text{filter}(C, *x)}{\langle \mathbf{endconst}(*x), C \rangle \rightarrow \langle \mathbf{0}, C' \rangle} \quad (\text{TR-ENDCONST}) \\
\frac{\mathbf{const}(*x) \notin C}{\langle \langle *x \rangle (P_1, P_2), C \rangle \xrightarrow{\mathbf{null}(*x)} \langle P_1, C \rangle} \quad (\text{TR-NOTCONST1}) \quad \frac{\mathbf{const}(*x) \notin C}{\langle \langle *x \rangle (P_1, P_2), C \rangle \xrightarrow{\neg \mathbf{null}(*x)} \langle P_2, C \rangle} \quad (\text{TR-NOTCONST2}) \\
\frac{\mathbf{null}(*x) \in C \quad \mathbf{const}(*x) \in C}{\langle \langle *x \rangle (P_1, P_2), C \rangle \rightarrow \langle P_1, C \rangle} \quad (\text{TR-NULLIN}) \quad \frac{\neg \mathbf{null}(*x) \in C \quad \mathbf{const}(*x) \in C}{\langle \langle *x \rangle (P_1, P_2), C \rangle \rightarrow \langle P_2, C \rangle} \quad (\text{TR-NNULLIN}) \\
\frac{\mathbf{null}(*x), \neg \mathbf{null}(*x) \notin C \quad \mathbf{const}(*x) \in C}{\langle \langle *x \rangle (P_1, P_2), C \rangle \xrightarrow{\mathbf{null}(*x)} \langle P_1, C \cup \mathbf{null}(*x) \rangle} \quad (\text{TR-NNULLNOTIN1}) \\
\frac{\mathbf{null}(*x), \neg \mathbf{null}(*x) \notin C \quad \mathbf{const}(*x) \in C}{\langle \langle *x \rangle (P_1, P_2), C \rangle \xrightarrow{\neg \mathbf{null}(*x)} \langle P_2, C \cup \neg \mathbf{null}(*x) \rangle} \quad (\text{TR-NNULLNOTIN2})
\end{array}$$

Figure 2: semantics of behavioral types with dependent types.

$$\begin{array}{c}
\Theta; \Gamma \vdash \mathbf{skip} : \mathbf{0} \quad (\text{T-SKIP}) \qquad \frac{\Theta; \Gamma \vdash s_1 : P_1 \quad \Theta; \Gamma \vdash s_2 : P_2}{\Theta; \Gamma \vdash s_1; s_2 : P_1; P_2} \quad (\text{T-SEQ}) \\
\Theta; \Gamma, x, y \vdash *x \leftarrow y : \mathbf{0} \quad (\text{T-ASSIGN}) \qquad \Theta; \Gamma, x \vdash \mathbf{free}(x) : \mathbf{free} \quad (\text{T-FREE}) \\
\frac{\Theta; \Gamma, x \vdash s : P}{\Theta; \Gamma \vdash \mathbf{let } x = \mathbf{malloc}() \mathbf{ in } s : \mathbf{malloc}; (x)P} \quad (\text{T-MALLOC}) \qquad \frac{\Theta; \Gamma, x, y \vdash s : P}{\Theta; \Gamma, y \vdash \mathbf{let } x = y \mathbf{ in } s : [y/x]P} \quad (\text{T-LETEQ}) \\
\frac{\Theta; \Gamma, x, y \vdash s : P}{\Theta; \Gamma, y \vdash \mathbf{let } x = *y \mathbf{ in } s : (x)P} \quad (\text{T-LETDEREF}) \qquad \frac{\Theta; \Gamma, x \vdash s : P}{\Theta; \Gamma \vdash \mathbf{let } x = \mathbf{null in } s : (x)P} \quad (\text{T-LETNULL}) \\
\Theta; \Gamma, x \vdash \mathbf{endconst}(*x) : \mathbf{endconst}(*x) \quad (\text{T-ENDCONST}) \\
\frac{\Theta; \Gamma, x \vdash s : P}{\Theta; \Gamma, x \vdash \mathbf{const}(*x)s : \mathbf{const}(*x)P} \quad (\text{T-CONST}) \\
\frac{\Theta; \Gamma, x \vdash s_1 : P_1 \quad \Theta; \Gamma, x \vdash s_2 : P_2}{\Theta; \Gamma, x \vdash \mathbf{ifnull}(*x) \mathbf{ then } s_1 \mathbf{ else } s_2 : (*x)(P_1, P_2)} \quad (\text{T-IFNULL}) \\
\Theta, f : (\vec{y})P; \Gamma, \vec{x} \vdash f(\vec{x}) : P[\vec{x}/\vec{y}] \quad (\text{T-CALL}) \\
\frac{\Theta; \Gamma \vdash s : P_1 \quad P_1 \leq P_2}{\Theta; \Gamma \vdash s : P_2} \quad (\text{T-SUB}) \\
\frac{\Theta(f) = (\vec{x})P \quad \mathbf{Dom}(D) = \mathbf{Dom}(\Theta) \quad \Theta; x_1, \dots, x_n \vdash s : P \text{ for each } f \mapsto (x_1, \dots, x_n)s \in D}{\vdash D : \Theta} \quad (\text{T-DEF}) \\
\frac{\vdash D : \Theta \quad \Theta; \emptyset \vdash s : P \quad OK_n(P, C)}{\vdash \langle D, s \rangle : n} \quad (\text{T-PROGRAM})
\end{array}$$

Figure 3: typing rules

3 Proof of Lemmas

Lemma 3.1. *If $OK_n(P, C)$ and $\langle P, C \rangle \xrightarrow{\rho} \langle P', C' \rangle$, then*

- $OK_{n-1}(P', C')$ if $\rho = \mathbf{malloc}$,
- $OK_{n+1}(P', C')$ if $\rho = \mathbf{free}$,
- $OK_n(P', C')$ if $\rho = \text{Otherwise}$

Proof. By induction on $\langle P, C \rangle \xrightarrow{\rho} \langle P', C' \rangle$.

- Case $P = \mathbf{0}; P'$ and $\langle \mathbf{0}; P', C \rangle \rightarrow \langle P', C \rangle$

We need to prove $OK_n(P', C)$. Assume that $OK_n(P', C)$ does not hold. Then, we have $\exists \sigma$ and Q s.t. $\langle P', C \rangle \xrightarrow{\sigma} \langle Q, C' \rangle$, $\#_m(\sigma) - \#_f(\sigma) > n$.

From the definition of that $OK_n(\mathbf{0}; P', C)$ holds, we have (1) if $\langle \mathbf{0}; P', C \rangle \rightarrow \langle P', C \rangle \xrightarrow{\sigma} \langle Q, C' \rangle$, then $\#_m(\sigma) - \#_f(\sigma) \leq n$, which are in contradiction to the assumption $\#_m(\sigma) - \#_f(\sigma) > n$. Therefore, $OK_n(P', C)$ holds.

- Case $P = \mathbf{malloc}$ and $\langle \mathbf{malloc}, C \rangle \xrightarrow{\mathbf{malloc}} \langle \mathbf{0}, C' \rangle$

we need to prove $OK_{n-1}(\mathbf{0}, C')$, which means we need to prove that for all σ and Q , if $\langle \mathbf{0}, C' \rangle \xrightarrow{\sigma} \langle Q, C' \rangle$ then $\#_m(\sigma) - \#_f(\sigma) \leq n - 1$. There is no σ and Q such that $\langle \mathbf{0}, C' \rangle \xrightarrow{\sigma} \langle Q, C' \rangle$. Therefore, $OK_{n-1}(\mathbf{0}, C')$ holds.

- Case $P = \mathbf{let } x = y \mathbf{ in } P'$ and $\langle \mathbf{let } x = y \mathbf{ in } P', C \rangle \rightarrow \langle [x'/x]P', C \rangle$

[TODO]

- Case $P = \mathbf{let } x = *y \mathbf{ in } P'$ and $\langle \mathbf{let } x = *y \mathbf{ in } P', C \rangle \rightarrow \langle [x'/x]P', F \rangle$

Similar to the above.

- Case $P = \mathbf{let } x = \mathbf{null} \mathbf{ in } P'$ and $\langle \mathbf{let } x = \mathbf{null} \mathbf{ in } P', C \rangle \rightarrow \langle [x'/x]P', C \rangle$

Similar to the above.

- Case $P = \mathbf{free}$ and $\langle \mathbf{free}, C \rangle \xrightarrow{\mathbf{free}} \langle \mathbf{0}, C \rangle$

We need to prove $OK_{n+1}(\mathbf{0}, C)$, which means we need to prove (1) $\forall \sigma$ and Q if $\langle \mathbf{0}, C \rangle \xrightarrow{\sigma} \langle Q, C' \rangle$, then $\#_m(\sigma) - \#_f(\sigma) \leq n + 1$. There is no Q and σ s.t. $\langle \mathbf{0}, C \rangle \xrightarrow{\sigma} \langle Q, C' \rangle$, so (1) holds. Therefore, $OK(\mathbf{0}, C)$ holds.

- Case $P = \mathbf{endconst}(*x)$ and $\frac{C' = \mathbf{filter}(C, *x)}{\langle \mathbf{endconst}(*x), C \rangle \rightarrow \langle \mathbf{0}, C' \rangle}$

We need to prove $OK_n(\mathbf{0}, C')$, which means we need to prove $\forall \sigma$ and Q if $\langle \mathbf{0}, C \rangle \xrightarrow{\sigma} \langle Q, C' \rangle$, then $\#_m(\sigma) - \#_f(\sigma) \leq n$ and (2) $OK(C')$ holds. There is no Q and σ s.t. $\langle \mathbf{0}, C \rangle \xrightarrow{\sigma} \langle Q, C' \rangle$. So $OK_n(\mathbf{0}, C')$ holds.

- Case $P = (*x)(P_1, P_2)$ and $\frac{\mathbf{const}(*x) \notin C}{\langle (*x)(P_1, P_2), C \rangle \xrightarrow{\mathbf{null}(*x)} \langle P_1, C \rangle}$

We need to prove $OK_n(P_1, C)$. Assume that $OK_n(P_1, C)$ does not hold. Then, we have $\exists \sigma$ and Q s.t. $\langle P_1, C \rangle \xrightarrow{\sigma} \langle Q, C' \rangle$ and $\#_m(\sigma) - \#_f(\sigma) > n$.

From the definition of that $OK_n((*) (P_1, P_2), C)$ holds, we have (1) if $\langle (*)(P_1, P_2), C \rangle \xrightarrow{\text{null}(*x)} \langle P_1, C \rangle \xrightarrow{\sigma} \langle Q, C' \rangle$ then $\#_m(\sigma) - \#_f(\sigma) \leq n$, which is in contradiction to the assumption $\#_m(\sigma) - \#_f(\sigma) > n$. Therefore, $OK_n(P_1, C)$ holds.

- Case $P = (*)(P_1, P_2)$ and $\frac{\text{const}(*x) \notin C}{\langle (*)(P_1, P_2), C \rangle \rightarrow \langle P_2, C \rangle}$

We need to prove $OK_n(P_2, C)$. Assume that $OK_n(P_2, C)$ does not hold. Then, we have $\exists \sigma$ and Q s.t. $\langle P_2, C \rangle \xrightarrow{\sigma} \langle Q, C' \rangle$ and $\#_m(\sigma) - \#_f(\sigma) > n$.

From the definition of that $OK_n((*) (P_1, P_2), C)$ holds, we have if $\langle (*)(P_1, P_2), C \rangle \xrightarrow{\neg \text{null}(*x)} \langle P_2, C \rangle \xrightarrow{\sigma} \langle Q, C' \rangle$, then $\#_m(\sigma) - \#_f(\sigma) \leq n$, which is in contradiction to the assumption. Therefore, $OK_n(P_2, C)$ holds.

- Case $P = (*)(P_1, P_2)$ and $\frac{\text{null}(*x) \in C}{\langle (*)(P_1, P_2), C \rangle \rightarrow \langle P_1, C \rangle} \frac{\text{const}(*x) \in C}{\langle (*)(P_1, P_2), C \rangle \rightarrow \langle P_2, C \rangle}$

We need to prove $OK_n(P_1, C)$. Assume that $OK_n(P_1, C)$ does not hold. Then, we have (1) $\exists \sigma$ and Q s.t. $\langle P_1, C \rangle \xrightarrow{\sigma} \langle Q, C' \rangle$ and $\#_m(\sigma) - \#_f(\sigma) > n$.

From the definition of that $OK_n((*) (P_1, P_2), C)$ holds, we have (1) if $\langle (*)(P_1, P_2), C \rangle \rightarrow \langle P_1, C \rangle \xrightarrow{\sigma} \langle Q, C' \rangle$, then $\#_m(\sigma) - \#_f(\sigma) \leq n$, which is in contradiction to the assumption. Therefore, $OK_n(P_1, C)$ holds.

- Case $P = (*)(P_1, P_2)$ and $\frac{\neg \text{null}(*x) \in C}{\langle (*)(P_1, P_2), C \rangle \rightarrow \langle P_2, C \rangle} \frac{\text{const}(*x) \in C}{\langle (*)(P_1, P_2), C \rangle \rightarrow \langle P_1, C \rangle}$

We need to prove $OK_n(P_2, C)$. Assume that $OK_n(P_2, C)$ does not hold. Then we have (1) $\exists \sigma$ and Q s.t. $\langle P_2, C \rangle \xrightarrow{\sigma} \langle Q, C' \rangle$ and $\#_m(\sigma) - \#_f(\sigma) > n$.

From the definition of that $OK_n((*) (P_1, P_2), C)$ holds, we have (1) if $\langle (*)(P_1, P_2), C \rangle \rightarrow \langle P_2, C \rangle \xrightarrow{\sigma} \langle Q, C' \rangle$, then $\#_m(\sigma) - \#_f(\sigma) \leq n$, which is in contradiction to the assumption. Therefore, $OK_n(P_2, C)$ holds.

- Case $P = (*)(P_1, P_2)$ and $\frac{\text{null}(*x), \neg \text{null}(*x) \notin C}{\langle (*)(P_1, P_2), C \rangle \xrightarrow{\text{null}(*x)} \langle P_1, C \cup \{\text{null}(*x)\} \rangle} \frac{\text{const}(*x) \in C}{\langle (*)(P_1, P_2), C \rangle \rightarrow \langle P_2, C \rangle}$

We need to prove $OK_n(P_1, C \cup \{\text{null}(*x)\})$. Assume that $OK_n(P_1, C \cup \{\text{null}(*x)\})$ does not hold. Then we have $\exists \sigma$ and Q s.t. $\langle P_1, C \cup \{\text{null}(*x)\} \rangle \xrightarrow{\sigma} \langle Q, C' \rangle$ and $\#_m(\sigma) - \#_f(\sigma) > n$.

From the definition of that $OK_n((*) (P_1, P_2), C)$ holds, we have if $\langle (*)(P_1, P_2), C \rangle \xrightarrow{\text{null}(*x)} \langle P_1, C \cup \{\text{null}(*x)\} \rangle \xrightarrow{\sigma} \langle Q, C' \rangle$, then $\#_m(\sigma) - \#_f(\sigma) \leq n$. Therefore, we get the contradiction and $OK_n(P_1, C \cup \{\text{null}(*x)\})$ holds.

- Case $P = (*)(P_1, P_2)$ and $\frac{\text{null}(*x), \neg \text{null}(*x) \notin C}{\langle (*)(P_1, P_2), C \rangle \xrightarrow{\neg \text{null}(*x)} \langle P_2, C \cup \{\neg \text{null}(*x)\} \rangle} \frac{\text{const}(*x) \in C}{\langle (*)(P_1, P_2), C \rangle \rightarrow \langle P_1, C \rangle}$

Similar to the above.

- Case $P = \text{const}(*x)P'$ and $\langle \text{const}(*x)P', C \rangle \rightarrow \langle P'; \text{endconst}(*x), C \cup \text{const}(*x) \rangle$

We need to prove $OK_n(P'; \text{endconst}(*x), C \cup \text{const}(*x))$. Assume that $OK_n(P'; \text{endconst}(*x), C \cup \text{const}(*x))$ does not hold. Then, we have $\exists \sigma$ and Q s.t. $\langle P'; \text{endconst}(*x), C \cup \text{const}(*x) \rangle \xrightarrow{\sigma} \langle Q, C' \rangle$ and $\#_m(\sigma) - \#_f(\sigma) > n$.

From the definition of that $OK_n(\text{const}(*x)P', C)$ holds, we have (1) if $\langle \text{const}(*x)P', C \rangle \rightarrow \langle P'; \text{endconst}(*x), C \cup \text{const}(*x) \rangle \xrightarrow{\sigma} \langle Q, C' \rangle$, then $\#_m(\sigma) - \#_f(\sigma) \leq n$, which is in contradiction to the assumption. Therefore, $OK_n(P'; \text{endconst}(*x), C \cup \text{const}(*x))$ holds.

- Case $P = \mu\alpha.P'$ and $\langle \mu\alpha.P', C \rangle \rightarrow \langle [\mu\alpha.P'/\alpha]P', C \rangle$

We need to prove $OK_n([\mu\alpha.P'/\alpha]P', C)$. Assume that $OK_n([\mu\alpha.P'/\alpha]P', C)$ does not hold. Then, we have (1) $\exists \sigma$ and Q s.t. $\langle [\mu\alpha.P'/\alpha]P', C \rangle \xrightarrow{\sigma} \langle Q, C' \rangle$ and $\sharp_m(\sigma) - \sharp_f(\sigma) > n$.

From the definition of that $OK_n(\mu\alpha.P', C)$ holds, we have (1) if $\langle \mu\alpha.P', C \rangle \rightarrow \langle [\mu\alpha.P'/\alpha]P', C \rangle \xrightarrow{\sigma} \langle Q, C' \rangle$, then $\sharp_m(\sigma) - \sharp_f(\sigma) \leq n$, which is a contradiction. Therefore, $OK([\mu\alpha.P'/\alpha]P', C)$ holds.

- Case $P = P_1; P_2$ and $\frac{\langle P_1, C \rangle \xRightarrow{\rho} \langle P'_1, C' \rangle}{\langle P_1; P_2, C \rangle \xRightarrow{\rho} \langle P'_1; P_2, C' \rangle}$

We need to prove $OK_{n'}(P'_1; P_2, C)$, where n' is determined by

$$n' = \begin{cases} n+1 & \rho = \mathbf{free} \\ n-1 & \rho = \mathbf{malloc} \\ n & \text{Otherwise.} \end{cases}$$

Assume that $OK_{n'}(P'_1; P_2, C')$ does not hold. Then, we have (1) $\exists \sigma, Q$ and C'' s.t. $\langle P'_1; P_2, C' \rangle \xrightarrow{\sigma} \langle Q, C'' \rangle$ and $\sharp_m(\sigma) - \sharp_f(\sigma) > n'$.

From the definition of that $OK_n(P_1; P_2, C)$ holds, we have (1) if $\langle P_1; P_2, C \rangle \xRightarrow{\rho} \langle P'_1; P_2, C' \rangle \xrightarrow{\sigma} \langle Q, C'' \rangle$, then $\sharp_m(\rho\sigma) - \sharp_f(\rho\sigma) \leq n$.

From (1), we get $n' + \sharp_m(\rho) - \sharp_f(\rho) < \sharp_m(\rho) + \sharp_m(\sigma) - \sharp_f(\rho) - \sharp_f(\sigma) \leq n$. For any ρ , the $n' + \sharp_m(\rho) - \sharp_f(\rho) = n$, therefore we get a contradiction. Therefore, $OK_{n'}(P'_1; P_2, C')$ holds.

□

Lemma 3.2. *If consistency(H, R, C) and $\langle H, R, s, n, C \rangle \xrightarrow{\rho} \langle H', R', s', n', C' \rangle$, then consistency(H', R', C').*

Proof. By induction on $\langle H, R, s, n, C \rangle \xrightarrow{\rho} \langle H', R', s', n', C' \rangle$

- Case: $\langle H, R, \mathbf{const}(*y)s, n, C \rangle \rightarrow \langle H, R, s; \mathbf{endconst}(*y), n', C \cup \mathbf{const}(*y) \rangle$.

We need to prove $\text{consistency}(H, R, C \cup \mathbf{const}(*y))$. We should prove for all x (1) if $\mathbf{null}(*x) \in C \cup \mathbf{const}(*y)$, then $\mathbf{const}(*x) \in C \cup \mathbf{const}(*y)$ and $H(R(x)) = \mathbf{null}$ if $H(R(x))$ is defined and (2) if $\neg \mathbf{null}(*x) \in C \cup \mathbf{const}(*y)$, then $\mathbf{const}(*x) \in C \cup \mathbf{const}(*y)$ and $H(R(x)) \neq \mathbf{null}$. By case analysis, let $\mathbf{const}(*y) \in C$, so $C \cup \mathbf{const}(*y)$ is C . Assuming $\mathbf{null}(*x) \in C$. By hypothesis, we have $\mathbf{const}(*x) \in C$ and $H(R(x)) = \mathbf{null}$ if $H(R(x))$ is defined. Therefore we got if $\mathbf{null}(*x) \in C \cup \mathbf{const}(*y)$, then $\mathbf{const}(*x) \in C \cup \mathbf{const}(*y)$ and $H(R(x)) = \mathbf{null}$ if $H(R(x))$ is defined. Let $\mathbf{const}(*y) \notin C$, and assume $\mathbf{null}(*x) \in C \cup \mathbf{const}(*y)$. By hypothesis, we have $\mathbf{const}(*x) \in C$ and $H(R(x)) = \mathbf{null}$ if $H(R(x))$ is defined. Because C is subset of $C \cup \mathbf{const}(*y)$, therefore we got if $\mathbf{null}(*x) \in C \cup \mathbf{const}(*y)$, then $\mathbf{const}(*x) \in C \cup \mathbf{const}(*y)$ and $H(R(x)) = \mathbf{null}$ if $H(R(x))$ is defined. So (1) holds. Similar to (1), (2) holds.

Therefore, $\text{consistency}(H, R, C \cup \mathbf{const}(*y))$ holds.

- Case: $\langle H, R, \mathbf{endconst}(*y), n, C \rangle \rightarrow \langle H, R, \mathbf{skip}, n, C' \rangle$ where $C' = \text{filter}(C, *y)$.

We need to prove $\text{consistency}(H, R, C')$ where $C' = \text{filter}(C, *y)$. From assumption $\text{consistency}(H, R, C)$, we have for all x (1) C does not contain both $\mathbf{null}(*x)$ and $\neg \mathbf{null}(*x)$. From definition of function $\text{filter}(C, *y)$, we know for all x C' does not contain both $\mathbf{null}(*x)$ and $\neg \mathbf{null}(*x)$.

(2) if $\mathbf{const}(*x) \in C$ and $\mathbf{null}(*x) \in C$, then $H(R(x)) = \mathbf{null}$. Assume that $\mathbf{const}(*x) \in C$ and $\mathbf{null}(*x) \in C$, then from the definition of $\mathbf{filter}(C, *y)$, we know $\mathbf{const}(*x) \in C'$ and $\mathbf{null}(*x) \in C'$, and H and R do not change, so $H(R(x)) = \mathbf{null}$. Therefore, for all x if $\mathbf{const}(*x) \in C'$ and $\mathbf{null}(*x) \in C'$, then $H(R(x)) = \mathbf{null}$. (3) similar to (2).

Therefore, $\mathbf{consistency}(H, R, C')$ holds.

- Case: $\langle H, R, \mathbf{free}(y), n, C \rangle \xrightarrow{\mathbf{free}} \langle H \setminus R(y), R, \mathbf{skip}, n + 1, C \rangle$.

We need to prove $\mathbf{consistency}(H \setminus R(y), R, C)$. From assumption $\mathbf{consistency}(H, R, C)$, we have for all x (1) C does not contain both $\mathbf{null}(*x)$ and $\neg \mathbf{null}(*x)$. (2) if $\mathbf{const}(*x) \in C$ and $\mathbf{null}(*x) \in C$, then $H(R(x)) = \mathbf{null}$. Assume that $\mathbf{const}(*x) \in C$ and $\mathbf{null}(*x) \in C$, and we know $(H \setminus R(y))(R(x)) = \mathbf{null}$. Therefore, for all x , if $\mathbf{const}(*x) \in C$ and $\mathbf{null}(*x) \in C$, then $(H \setminus R(y))(R(x)) = \mathbf{null}$. (3) similar to (2).

Therefore, $\mathbf{consistency}(H \setminus R(y), R, C)$ holds.

- Case: $\langle H, R, \mathbf{let } y = \mathbf{mallocin } s, n, C \rangle \xrightarrow{\mathbf{malloc}} \langle H \{l \mapsto v\}, R \{x' \mapsto l\}, [x'/y]s, n', C \rangle$.

We need to prove $\mathbf{consistency}(H \{l \mapsto v\}, R \{x' \mapsto l\}, C)$. From assumption $\mathbf{consistency}(H, R, C)$, we have for all x (1) C does not contain both $\mathbf{null}(*x)$ and $\neg \mathbf{null}(*x)$. (2) if $\mathbf{const}(*x) \in C$ and $\mathbf{null}(*x) \in C$, then $H(R(x)) = \mathbf{null}$. Assume that $\mathbf{const}(*x) \in C$ and $\mathbf{null}(*x) \in C$, then we have $H(R(x)) = \mathbf{null}$, and we know $H \{l \mapsto v\}(R \{x' \mapsto l\}(x)) = \mathbf{null}$. Therefore, we get for all x , if $\mathbf{const}(*x) \in C$ and $\mathbf{null}(*x) \in C$, then $H \{l \mapsto v\}(R \{x' \mapsto l\}(x)) = \mathbf{null}$. (3) similar to (2).

Therefore, $\mathbf{consistency}(H \{l \mapsto v\}, R \{x' \mapsto l\}, C)$ holds.

- Case: $\langle H, R, \mathbf{skip}; s, n, C \rangle \rightarrow \langle H, R, s, n', C \rangle$.

We need to prove $\mathbf{consistency}(H, R, C)$. Obviously, from assumption $\mathbf{consistency}(H, R, C)$.

- Case: $\langle H \{R(w) \mapsto v\}, R, *w \leftarrow y, n, C \rangle \rightarrow \langle H \{R(w) \mapsto R(y)\}, R, \mathbf{skip}, n, C \rangle$ where $\forall z. R(w) = R(z) \Rightarrow \mathbf{const}(*z) \notin C$

We need to prove $\mathbf{consistency}(H \{R(w) \mapsto R(y)\}, R, C)$. From assumption $\mathbf{consistency}(H, R, C)$, we have for all x (1) C does not contain both $\mathbf{null}(*x)$ and $\neg \mathbf{null}(*x)$ (2) if $\mathbf{const}(*x) \in C$ and $\mathbf{null}(*x) \in C$, then $H \{R(w) \mapsto v\}(R(x)) = \mathbf{null}$. Assume that $\mathbf{const}(*x) \in C$ and $\mathbf{null}(*x) \in C$, then we have $H \{R(w) \mapsto v\}(R(x)) = \mathbf{null}$, and we know $H \{R(w) \mapsto R(y)\}(R(x)) = \mathbf{null}$. Therefore, for all x , if $\mathbf{const}(*x) \in C$ and $\mathbf{null}(*x) \in C$, then $H \{R(w) \mapsto R(y)\}(R(x)) = \mathbf{null}$. (3) similar to (2).

Therefore, $\mathbf{consistency}(H \{R(w) \mapsto R(y)\}, R, C)$ holds.

- Case: $\langle H, R, \mathbf{let } z = \mathbf{yin } s, n, C \rangle \rightarrow \langle H, R \{z' \mapsto R(y)\}, [z'/z]s, n, C \rangle$

We need to prove $\mathbf{consistency}(H, R \{z' \mapsto R(y)\}, C)$. From assumption $\mathbf{consistency}(H, R, C)$, we have for all x (1) C does not contain both $\mathbf{null}(*x)$ and $\neg \mathbf{null}(*x)$ (2) if $\mathbf{const}(*x) \in C$ and $\mathbf{null}(*x) \in C$, then $H(R(x)) = \mathbf{null}$. Assume that $\mathbf{const}(*x) \in C$ and $\mathbf{null}(*x) \in C$, then we have $H(R(x)) = \mathbf{null}$, and we get $H(R \{z' \mapsto R(y)\}) = \mathbf{null}$. Therefore, for all x , if $\mathbf{const}(*x) \in C$ and $\mathbf{null}(*x) \in C$, then $H(R \{z' \mapsto R(y)\}) = \mathbf{null}$. (3) similar to (2).

Therefore, $\mathbf{consistency}(H, R \{z' \mapsto R(y)\}, C)$ holds.

- Case: $\langle H, R, \text{ifnull}(*y) \text{ then } s_1 \text{ else } s_2, n, C \rangle \xrightarrow{\text{null}(*y)} \langle H, R, s_1, n, C \rangle$ where $H(R(y)) = \text{null}$ and $\text{const}(*y) \notin C$

We need to prove $\text{consistency}(H, R, C)$. Obviously, $\text{consistency}(H, R, C)$ holds from assumption.

- Case: $\langle H, R, \text{ifnull}(*y) \text{ then } s_1 \text{ else } s_2, n, C \rangle \xrightarrow{\neg \text{null}(*y)} \langle H, R, s_2, n, C \rangle$ where $H(R(y)) \neq \text{null}$ and $\text{const}(*y) \notin C$

We need to prove $\text{consistency}(H, R, C)$. Obviously, $\text{consistency}(H, R, C)$ holds from assumption.

- Case: $\langle H, R, \text{ifnull}(*y) \text{ then } s_1 \text{ else } s_2, n, C \rangle \xrightarrow{\text{null}(*y)} \langle H, R, s_1, n, C \cup \text{null}(*y) \rangle$ where $H(R(y)) = \text{null}$ and $\text{const}(*y) \in C$

We need to prove $\text{consistency}(H, R, C \cup \text{null}(*y))$. From assumption $\text{consistency}(H, R, C)$, we have for all x (1) C does not contain both $\text{null}(*x)$ and $\neg \text{null}(*x)$. Assume that $\neg \text{null}(*y) \in C$, and because we know $\text{const}(*y) \in C$, then $H(R(y)) \neq \text{null}$. Then we get the contradiction $H(R(y))$ should be null. Therefore $\neg \text{null}(*y) \notin C$. Then we get for all x , $C \cup \text{null}(*y)$ does not contain both $\text{null}(*x)$ and $\neg \text{null}(*x)$. (2) if $\text{const}(*x) \in C$ and $\text{null}(*x) \in C$, then $H(R(x)) = \text{null}$. Assume that $\text{const}(*x) \in C$ and $\text{null}(*x) \in C$, then we have $H(R(x)) = \text{null}$, therefore $\text{const}(*x) \in C \cup \text{null}(*y)$ and $\text{null}(*x) \in C \cup \text{null}(*y)$. H and R do not change, so $H(R(x)) = \text{null}$. Therefore, we get for all x , if $\text{const}(*x) \in C \cup \text{null}(*y)$ and $\text{null}(*x) \in C \cup \text{null}(*y)$, then $H(R(x)) = \text{null}$. (3) similar to (2).

Therefore, $\text{consistency}(H, R, C \cup *y)$ holds.

□

Proof of Lemma 2.2: By induction on the derivation of $\langle H, R, s, n, C \rangle \xrightarrow{\rho} \langle H', R', s', n', C' \rangle$.

- Case: $\langle H, R, \text{const}(*x)s, n, C \rangle \rightarrow \langle H, R, s; \text{endconst}(*x), n, C \cup \{\text{const}(*x)\} \rangle$

From the assumption $\Theta; \Gamma \vdash \langle H, R, \text{const}(*x)s, n, C \rangle : \langle P, C \rangle$, we have $\Theta; \Gamma \vdash \text{const}(*x)s : P$, $OK_n(P, C)$ and $\text{consistency}(H, R, C)$. From the inversion of typing rules, we get $\Theta; \Gamma \vdash s : P''$ and $\text{const}(*x)P'' \leq P$ for some P'' . By subtyping, we have $P''; \text{endconst}(*x) \leq Q$ and $\langle P, C \rangle \Rightarrow \langle Q, C \cup \{\text{const}(*x)\} \rangle$ for some Q .

we need to find P' and C' s.t. $\Theta; \Gamma \vdash s; \text{endconst}(*x) : P'$, $OK_n(P', C')$, $\langle P, C \rangle \Rightarrow \langle P', C' \rangle$ and $\text{consistency}(H, R, C')$. Taking Q as P' and $C \cup \{\text{const}(*x)\}$ as C' . Therefore $\langle P, C \rangle \rightarrow \langle P', C' \rangle$ holds, and then $OK_n(P', C')$ and $\text{consistency}(H, R, C')$ hold from Lemma 4.1 and Lemma 4.2. From $\Theta; \Gamma \vdash s; \text{endconst}(*x) : P''; \text{endconst}(*x), P''; \text{endconst}(*x) \leq Q$ and T-SUB, $\Theta; \Gamma \vdash s; \text{endconst}(*x) : P'$ holds.

- Case: $\langle H, R, \text{endconst}(*x), n, C \rangle \rightarrow \langle H, R, \text{skip}, n, C' \rangle$ where $C' = \text{filter}(C, *x)$

From the assumption $\Theta; \Gamma \vdash \langle H, R, \text{endconst}(*x), n, C \rangle : \langle P, C \rangle$, we have $\Theta; \Gamma \vdash \text{endconst}(*x) : P$, $OK_n(P, C)$ and $\text{consistency}(H, R, C)$. From the inversion of typing rules, we get $\Theta; \Gamma \vdash \text{endconst}(*x) : \text{endconst}(*x)$ and $\text{endconst}(*x) \leq P$. By subtyping, we get $0 \leq Q$ and $\langle P, C \rangle \rightarrow \langle Q, C \rangle$ for some Q .

we need to find P' and C' s.t. $\Theta; \Gamma \vdash \text{skip} : P'$, $OK_n(P', C')$, $\langle P, C \rangle \Rightarrow \langle P', C' \rangle$ and $\text{consistency}(H, R, C')$. Taking Q as P' and C as C' , then $\langle P, C \rangle \rightarrow \langle P', C' \rangle$ holds, and then

$OK_n(P', C')$ and $consistency(H, R, C')$ hold from Lemma 4.1 and Lemma 4.2. From T-SKIP, T-SUB and $0 \leq Q$, then $\Theta; \Gamma \vdash \mathbf{skip} : P'$ holds.

- Case: $\langle H, R, \mathbf{free}(x), n, C \rangle \xrightarrow{\mathbf{free}} \langle H', R, \mathbf{skip}, n+1, C' \rangle$

From the assumption $\Theta; \Gamma \vdash \langle H, R, \mathbf{free}(x), n, C \rangle : \langle P, C \rangle$, we have $OK_n(P, C)$, $consistency(H, R, C)$ and $\Theta; \Gamma \vdash \mathbf{free}(x) : P$. From inversion of the typing rules, we have $\Theta; \Gamma \vdash \mathbf{free}(x) : \mathbf{free}$ and $\mathbf{free} \leq P$. By the subtyping, we have $\langle P, F \rangle \xrightarrow{\mathbf{free}} \langle Q, C' \rangle$ and $0 \leq Q$ for some Q .

We need to find P' and C' such that $\langle P, C \rangle \xrightarrow{\mathbf{free}} \langle P', C' \rangle$, $\Theta; \Gamma \vdash \mathbf{skip} : P'$, and $OK_{n+1}(P', C')$.

Take Q as P' and C as C' . Then, $\langle P, C \rangle \xrightarrow{\mathbf{free}} \langle P', C' \rangle$ holds, and $OK_{n+1}(P', C')$ holds from Lemma 4.1. We also have $\Theta; \Gamma \vdash \mathbf{skip} : P'$ from T-SKIP, $0 \leq Q$ and T-SUB.

- Case: $\langle H, R, \mathbf{let } x = \mathbf{malloc}() \text{ in } s, n, C \rangle \xrightarrow{\mathbf{malloc}} \langle H', R', [x'/x]s, n-1, C' \rangle$

From the assumption $\Theta; \Gamma \vdash \langle H, R, \mathbf{let } x = \mathbf{malloc}() \text{ in } s, n, C \rangle : \langle P, C \rangle$, we have $\Theta; \Gamma \vdash \mathbf{let } x = \mathbf{malloc}() \text{ in } s : P$, $OK_n(P, C)$ and $consistency(H, R, C)$. By the inversion of typing rules, we have $\Theta; \Gamma, x \vdash s : P''$ and $\mathbf{malloc}; (x)P'' \leq P$ for some P'' . By subtyping, we get $\langle P, C \rangle \xrightarrow{\mathbf{malloc}} \langle Q, F \rangle$ and $[x'/x]P'' \leq Q$ for some Q .

We need to find P' and C' such that $\Theta; \Gamma, x' \vdash [x'/x]s : P'$, $\langle P, C \rangle \xrightarrow{\mathbf{malloc}} \langle P', C' \rangle$, $consistency(H', R', C')$ and $OK_{n-1}(P', C')$. Take Q as P' and C as C' . Then $\langle P, C \rangle \xrightarrow{\mathbf{malloc}} \langle P', C' \rangle$ holds, and then $OK_{n-1}(P', C')$ and $consistency(H', R', C')$ hold by Lemma 4.1 and Lemma 4.2. From $\Theta; \Gamma, x \vdash s : P''$ and $\mathbf{malloc}; (x)P'' \leq P$, we have $\Theta; \Gamma, x'' \vdash [x''/x]s : [x''/x]P''$ and $\mathbf{malloc}; (x)P'' \leq P$, and then by the definition of subtyping we have $[x''/x]P'' \leq Q'$ for some Q' . Therefore, we get $\Theta; \Gamma, x'' \vdash [x''/x]s : Q'$. Take x'' as x' and Q' as P' , then $\Theta; \Gamma, x' \vdash [x'/x]s : P'$ holds.

- Case: $\langle H, R, \mathbf{skip}; s, n, C \rangle \rightarrow \langle H, R, s, n, C \rangle$

From the assumption $\Theta; \Gamma \vdash \langle H, R, \mathbf{skip}; s, n, C \rangle : \langle P, C \rangle$, we have $\Theta; \Gamma \vdash \mathbf{skip}; s : P$, $OK_n(P, C)$ and $consistency(H, R, C)$. From the inversion of the typing rules, we get $\Theta; \Gamma \vdash s : P''$ and $0; P'' \leq P$. From the definition of subtyping, we have $\langle P, C \rangle \Rightarrow \langle Q, C \rangle$ and $P'' \leq Q$ for some Q .

We need to find P' and C' such that $\Theta; \Gamma \vdash s : P'$ and $\langle P, C \rangle \rightarrow \langle P', C' \rangle$ and $OK_n(P', C')$. Take Q as P' and C as C' . Then $\langle P, C \rangle \Rightarrow \langle P', C' \rangle$ holds, and then $OK_n(P', C')$ and $consistency(H, R, C')$ hold. We also have $\Theta; \Gamma \vdash s : P'$ from T-SUB, $\Gamma \vdash s : P''$ and $P'' \leq Q$.

- Case: $\langle H, R, *x \leftarrow y, n, C \rangle \rightarrow \langle H', R, \mathbf{skip}, n, C' \rangle$

From the assumption $\Theta; \Gamma \vdash \langle H, R, *x \leftarrow y, n, C \rangle : \langle P, C \rangle$, we have $\Theta; \Gamma \vdash *x \leftarrow y : P$, $OK_n(P, C)$ and $consistency(H, R, C)$. From the inversion of typing rules, we have $0 \leq P$.

We need to find P' and C' such that $\Theta; \Gamma \vdash \mathbf{skip} : P'$, $\langle P, C \rangle \Rightarrow \langle P', C' \rangle$ and $OK_n(P', C')$. Take P as P' and C as C' . Then $\langle P, C \rangle \Rightarrow \langle P', C' \rangle$ holds, and then $OK_n(P', C')$ and $consistency(H', R, C')$ hold from Lemma 4.1 and Lemma 4.2. We also have $\Theta; \Gamma \vdash \mathbf{skip} : P'$ from T-SKIP, $0 \leq P$ and T-SUB.

- Case: $\langle H, R, \mathbf{let } x = y \text{ in } s, n, C \rangle \rightarrow \langle H, R', [x'/x]s, n, C' \rangle$

From the assumption $\Theta; \Gamma \vdash \langle H, R, \mathbf{let } x = y \text{ in } s, n, C \rangle : \langle P, C \rangle$, we have $\Theta; \Gamma, y \vdash \mathbf{let } x = y \text{ in } s : P$, $OK_n(P, C)$ and $consistency(H, R, C)$. From the inversion of typing rules, we

have $\Theta; \Gamma, x, y \vdash s : P''$ and $\text{let } x = y \text{ in } P'' \leq P$ for some P'' . By subtyping, we have $\langle P, C \rangle \rightarrow \langle Q, C \rangle$ and $[x'/x]P'' \leq Q$ for some Q .

We need to find P' and C' such that $\Theta; \Gamma, x', y \vdash [x'/x]s : P'$, $\langle P, C \rangle \rightarrow \langle P', C' \rangle$, $OK_n(P', C')$ and $\text{consistency}(H, R', C')$. Take Q as P' and C as C' . Then $\langle P, C \rangle \Rightarrow \langle P', C' \rangle$ and $OK_n(P', C')$ hold. From $\Theta; \Gamma, x, y \vdash s : P''$ and $\text{let } x = y \text{ in } P'' \leq P$, we have $\Theta; \Gamma, x'', y \vdash [x''/x]s : [x''/x]P''$ and $\text{let } x'' = y \text{ in } [x''/x]P'' \leq P$, and then by subtyping we have $[x''/x]P'' \leq Q'$ for some Q' . Therefore, we have $\Theta; \Gamma, x'', y \vdash [x''/x]s : Q'$. Take x'' as x' and Q' as P' , then $\Theta; \Gamma, x', y \vdash [x'/x]s : P'$ holds.

- Case: $\langle H, R, \text{let } x = \text{null in } s, n \rangle \rightarrow \langle H, R', [x'/x]s, n \rangle$

Similar to the above.

- Case: $\langle H, R, \text{let } x = *y \text{ in } s, n \rangle \rightarrow \langle H, R', [x'/x]s, n \rangle$

Similar to the above.

- Case: $\langle H, R, \text{ifnull } (*x) \text{ then } s_1 \text{ else } s_2, n, C \rangle \xrightarrow{\text{null}(*x)} \langle H, R, s_1, n, C \rangle$ if $H(R(x)) = \text{null}$ and $\text{const}(*x) \notin C$

From assumption $\Theta; \Gamma \vdash \langle H, R, \text{ifnull } (*x) \text{ then } s_1 \text{ else } s_2, n, C \rangle : \langle P, C \rangle$, we have $\Theta; \Gamma \vdash \text{ifnull } (*x) \text{ then } s_1 \text{ else } s_2 : P$, $OK_n(P, C)$ and $\text{consistency}(H, R, C)$. From the inversion of typing rules, we have $\Theta; \Gamma \vdash s_1 : P_1$, $\Theta; \Gamma \vdash s_2 : P_2$ and $(*x)(P_1, P_2) \leq P$. According to the rule Tr-NotConst1 and $\text{const}(*x) \notin C$, we have $\langle (*x)(P_1, P_2) \rangle \xrightarrow{\text{null}(*x)} \langle P_1, C \rangle$, and then by definition of subtyping, we get $\langle P, C \rangle \xrightarrow{\text{null}(*x)} \langle Q, C \rangle$ and $P_1 \leq Q$ for some Q .

We need to find P' and C' such that $\Theta; \Gamma \vdash s_1 : P'$, $\langle P, C \rangle \xrightarrow{\text{null}(*x)} \langle P', C' \rangle$ and $OK_n(P', C')$.

Take Q as P' and C as C' . Then $\langle P, C \rangle \xrightarrow{\text{null}(*x)} \langle P', C' \rangle$ and $OK_n(P', C')$ hold. We also have $\Theta; \Gamma \vdash s_1 : P'$ from T-SUB, $\Theta; \Gamma \vdash s_1 : P_1$ and $P_1 \leq Q$.

- Case: $\langle H, R, \text{ifnull } (*x) \text{ then } s_1 \text{ else } s_2, n, C \rangle \xrightarrow{\neg \text{null}(*x)} \langle H, R, s_1, n, C \rangle$ if $H(R(x)) \neq \text{null}$ and $\text{const}(*x) \notin C$

Similar to the above.

- Case: $\langle H, R, \text{ifnull } (*x) \text{ then } s_1 \text{ else } s_2, n, C \rangle \xrightarrow{\text{null}(*x)} \langle H, R, s_1, n, C' \rangle$ if $H(R(x)) = \text{null}$, $\text{const}(*x) \in C$ and $C' = C \cup \{\text{null}(*x)\}$

From assumption $\Theta; \Gamma \vdash \langle H, R, \text{ifnull } (*x) \text{ then } s_1 \text{ else } s_2, n, C \rangle : \langle P, C \rangle$, we have $\Theta; \Gamma \vdash \text{ifnull } (*x) \text{ then } s_1 \text{ else } s_2 : P$, $OK_n(P, C)$ and $\text{consistency}(H, R, C)$. From the inversion of typing rules, we have $\Theta; \Gamma \vdash s_1 : P_1$, $\Theta; \Gamma \vdash s_2 : P_2$ and $(*x)(P_1, P_2) \leq P$. According to rule Tr-NullNotIn1, $\text{const}(*x) \in C$ and $C' = C \cup \{\text{null}(*x)\}$, we have $\langle (*x)(P_1, P_2) \rangle \xrightarrow{\text{null}(*x)} \langle P_1, C \cup \{\text{null}(*x)\} \rangle$, and then by the definition of subtyping, we get $\langle P, C \rangle \xrightarrow{\text{null}(*x)} \langle Q, C \cup \{\text{null}(*x)\} \rangle$ and $P_1 \leq Q$ for some Q .

We need to find P' and C' such that $\Theta; \Gamma \vdash s_1 : P'$, $\langle P, C \rangle \xrightarrow{\text{null}(*x)} \langle P', C' \rangle$, $OK_n(P', C')$ and $\text{consistency}(H, R, C')$. Take Q as P' and $C \cup \{\text{null}(*x)\}$ as C' . Then $\langle P, C \rangle \xrightarrow{\text{null}(*x)} \langle P', C' \rangle$ holds, and then $OK_n(P', C')$ and $\text{consistency}(H, R, C')$ hold by Lemma 4.1 and Lemma 4.2. We also have $\Theta; \Gamma \vdash s_1 : P'$ from T-SUB, $\Theta; \Gamma \vdash s_1 : P_1$ and $P_1 \leq Q$.

- Case: $\langle H, R, \text{ifnull}(*x) \text{ then } s_1 \text{ else } s_2, n, C \rangle \xrightarrow{\neg\text{null}(*x)} \langle H, R, s_2, n, C' \rangle$ if $H(R(x)) \neq \text{null}$, $\text{const}(*x) \in C$ and $C' = C \cup \{\neg\text{null}(*x)\}$

Similar to the above proof.

- Case: $\langle H, R, s_1; s_2, n, C \rangle \rightarrow \langle H', R', s'_1; s_2, n', C' \rangle$

From the assumption $\Theta; \Gamma \vdash \langle H, R, s_1; s_2, n, C \rangle : \langle P, C \rangle$, we have $\Theta; \Gamma \vdash s_1; s_2 : P$, $OK_n(P, C)$ and $\text{consistency}(H, R, C)$. By inversion of typing rules, we have $\Theta; \Gamma \vdash s_1 : P_1$, $\Theta; \Gamma \vdash s_2 : P_2$ and $P_1; P_2 \leq P$ for some P_1 and P_2 .

By IH on $\langle H, R, s_1, n, C \rangle$ with derivation $\langle H, R, s_1, n, C \rangle \xrightarrow{\rho} \langle H', R', s'_1, n', C' \rangle$, we have $\exists P'_1, C'_1$ s.t. $\Theta; \Gamma \vdash \langle H', R', s'_1, n', C' \rangle : \langle P'_1, C'_1 \rangle$ and $\langle P_1, C \rangle \xrightarrow{\rho} \langle P'_1, C'_1 \rangle$.

By subtyping we have $\langle P, C \rangle \xrightarrow{\rho} \langle Q, C'_1 \rangle$ and $P'_1; P_2 \leq Q$ for some Q .

We need to find P' and C' s.t. $\langle P, C \rangle \xrightarrow{\rho} \langle P', C' \rangle$, $OK_n(P', C')$ and $\Theta; \Gamma \vdash s'_1; s_2 : P'$. Take Q as P' and C'_1 as C' , $\langle P, C \rangle \xrightarrow{\rho} \langle P', C' \rangle$ and $OK_n(P', C')$ hold. By T-Sub, $\Theta; \Gamma \vdash s'_1; s_2 : P'_1; P_2$ and $P'_1; P_2 \leq Q$, we have $\Theta; \Gamma \vdash s'_1; s_2 : P'$ holds.

□

We write $\langle H, R, s, n, C \rangle \xrightarrow{\rho}$ if there is a transition $\xrightarrow{\rho}$ from $\langle H, R, s, n, C \rangle$.

Lemma 3.3. *If $\Theta \vdash \langle H, R, s, n, C \rangle : \langle P, C \rangle$ and $\langle H, R, s, n, C \rangle \xRightarrow{\rho}$ and $\rho \in \{\text{malloc}, \text{free}, \text{null}(*x), \neg\text{null}(*x)\}$, then there exists P' and C' such that $\langle P, C \rangle \xRightarrow{\rho} \langle P', C' \rangle$.*

Proof. Induction on the derivation of $\Theta; \Gamma \vdash \langle H, R, s, n, C \rangle : \langle P, C \rangle$.

□

Proof of Lemma 2.3:

By contradiction. Assume $\langle H, R, s, n, C \rangle \xrightarrow{\rho} \text{OutOfMemory}$. Then, n is 0 and $\rho = \text{malloc}$ from SEM-OUTOFMEM. From the assumption we have $\Theta; \Gamma \vdash s : P$ and $OK_0(P, C)$. From Lemma 4.3, there exists P' and C' such that $\langle P, C \rangle \xRightarrow{\text{malloc}} \langle P', C' \rangle$. However, this contradicts $OK_0(P, C)$.

□

Proof of Theorem 2.1:

We have $\Theta; \emptyset \vdash s : P, \vdash D : \Theta$, $OK_n(P, C)$ and $\text{consistency}(H, R, C)$.

Suppose that there exists σ such that $\langle \emptyset, \emptyset, s, n, C \rangle \xrightarrow{\sigma} \langle H', R', s', n', C' \rangle \xrightarrow{\rho} \text{OutOfMemory}$. Then, $n' = 0$ and $\rho = \text{malloc}$. From Lemma 2.2, there exists P' and C' such that $\Theta; \Gamma \vdash s' : P'$, $\langle P, C \rangle \xRightarrow{\sigma} \langle P', C' \rangle$, and $OK_0(P', C')$; hence $\langle H', R', s', 0 \rangle \xrightarrow{\text{malloc}}$. However, this contradicts Lemma 2.3.

□

=====

4 Proof of Lemmas

Lemma 4.1. *If $OK_n(P, C)$ and $\langle P, C \rangle \xrightarrow{\rho} \langle P', C' \rangle$, then*

- $OK_{n-1}(P', C')$ if $\rho = \text{malloc}$,
- $OK_{n+1}(P', C')$ if $\rho = \text{free}$,

- $OK_n(P', C')$ if $\rho = \text{Otherwise}$

Proof. By induction on $\langle P, C \rangle \xrightarrow{\rho} \langle P', C' \rangle$.

- Case $P = \mathbf{0}; P'$ and $\langle \mathbf{0}; P', C \rangle \rightarrow \langle P', C \rangle$

We need to prove $OK_n(P', C)$. Assume that $OK_n(P', C)$ does not hold. Then, we have $\exists \sigma$ and Q s.t. $\langle P', C \rangle \xrightarrow{\sigma} \langle Q, C' \rangle$, $\sharp_m(\sigma) - \sharp_f(\sigma) > n$.

From the definition of that $OK_n(\mathbf{0}; P', C)$ holds, we have (1) if $\langle \mathbf{0}; P', C \rangle \rightarrow \langle P', C \rangle \xrightarrow{\sigma} \langle Q, C' \rangle$, then $\sharp_m(\sigma) - \sharp_f(\sigma) \leq n$, which are in contradiction to the assumption $\sharp_m(\sigma) - \sharp_f(\sigma) > n$. Therefore, $OK_n(P', C)$ holds.

- Case $P = \mathbf{malloc}$ and $\langle \mathbf{malloc}, C \rangle \xrightarrow{\mathbf{malloc}} \langle \mathbf{0}, C \rangle$

we need to prove $OK_{n-1}(\mathbf{0}, C)$, which means we need to prove that for all σ and Q , if $\langle \mathbf{0}, C \rangle \xrightarrow{\sigma} \langle Q, C' \rangle$ then $\sharp_m(\sigma) - \sharp_f(\sigma) \leq n - 1$. There is no σ and Q such that $\langle \mathbf{0}, C \rangle \xrightarrow{\sigma} \langle Q, C' \rangle$. Therefore, $OK_{n-1}(\mathbf{0}, C)$ holds.

- Case $P = \mathbf{let } x = y \mathbf{ in } P'$ and $\langle \mathbf{let } x = y \mathbf{ in } P', C \rangle \rightarrow \langle [x'/x]P', C \rangle$

[TODO]

- Case $P = \mathbf{let } x = *y \mathbf{ in } P'$ and $\langle \mathbf{let } x = *y \mathbf{ in } P', C \rangle \rightarrow \langle [x'/x]P', F \rangle$

Similar to the above.

- Case $P = \mathbf{let } x = \mathbf{null} \mathbf{ in } P'$ and $\langle \mathbf{let } x = \mathbf{null} \mathbf{ in } P', C \rangle \rightarrow \langle [x'/x]P', C \rangle$

Similar to the above.

- Case $P = \mathbf{free}$ and $\langle \mathbf{free}, C \rangle \xrightarrow{\mathbf{free}} \langle \mathbf{0}, C \rangle$

We need to prove $OK_{n+1}(\mathbf{0}, C)$, which means we need to prove (1) $\forall \sigma$ and Q if $\langle \mathbf{0}, C \rangle \xrightarrow{\sigma} \langle Q, C' \rangle$, then $\sharp_m(\sigma) - \sharp_f(\sigma) \leq n + 1$. There is no Q and σ s.t. $\langle \mathbf{0}, C \rangle \xrightarrow{\sigma} \langle Q, C' \rangle$, so (1) holds. Therefore, $OK(\mathbf{0}, C)$ holds.

- Case $P = \mathbf{endconst}(*x)$ and $\frac{C' = \mathbf{filter}(C, *x)}{\langle \mathbf{endconst}(*x), C \rangle \rightarrow \langle \mathbf{0}, C' \rangle}$

We need to prove $OK_n(\mathbf{0}, C')$, which means we need to prove $\forall \sigma$ and Q if $\langle \mathbf{0}, C \rangle \xrightarrow{\sigma} \langle Q, C' \rangle$, then $\sharp_m(\sigma) - \sharp_f(\sigma) \leq n$ and (2) $OK(C')$ holds. There is no Q and σ s.t. $\langle \mathbf{0}, C \rangle \xrightarrow{\sigma} \langle Q, C' \rangle$. So $OK_n(\mathbf{0}, C')$ holds.

- Case $P = (*x)(P_1, P_2)$ and $\frac{\mathbf{const}(*x) \notin C}{\langle (*x)(P_1, P_2), C \rangle \xrightarrow{\mathbf{null}(*x)} \langle P_1, C \rangle}$

We need to prove $OK_n(P_1, C)$. Assume that $OK_n(P_1, C)$ does not hold. Then, we have $\exists \sigma$ and Q s.t. $\langle P_1, C \rangle \xrightarrow{\sigma} \langle Q, C' \rangle$ and $\sharp_m(\sigma) - \sharp_f(\sigma) > n$.

From the definition of that $OK_n((*x)(P_1, P_2), C)$ holds, we have (1) if $\langle (*x)(P_1, P_2), C \rangle \xrightarrow{\mathbf{null}(*x)} \langle P_1, C \rangle \xrightarrow{\sigma} \langle Q, C' \rangle$ then $\sharp_m(\sigma) - \sharp_f(\sigma) \leq n$, which is in contradiction to the assumption $\sharp_m(\sigma) - \sharp_f(\sigma) > n$. Therefore, $OK_n(P_1, C)$ holds.

- Case $P = (*x)(P_1, P_2)$ and $\frac{\mathbf{const}(*x) \notin C}{\langle (*x)(P_1, P_2), C \rangle \rightarrow \langle P_2, C \rangle}$

We need to prove $OK_n(P_2, C)$. Assume that $OK_n(P_2, C)$ does not hold. Then, we have $\exists \sigma$ and Q s.t. $\langle P_2, C \rangle \xrightarrow{\sigma} \langle Q, C' \rangle$ and $\sharp_m(\sigma) - \sharp_f(\sigma) > n$.

From the definition of that $OK_n((*x)(P_1, P_2), C)$ holds, we have if $\langle (*x)(P_1, P_2), C \rangle \xrightarrow{\neg \mathbf{null}(*x)} \langle P_2, C \rangle \xrightarrow{\sigma} \langle Q, C' \rangle$, then $\sharp_m(\sigma) - \sharp_f(\sigma) \leq n$, which is in contradiction to the assumption. Therefore, $OK_n(P_2, C)$ holds.

- Case $P = (*x)(P_1, P_2)$ and $\frac{\mathbf{null}(*x) \in C \quad \mathbf{const}(*x) \in C}{\langle (*x)(P_1, P_2), C \rangle \rightarrow \langle P_1, C \rangle}$

We need to prove $OK_n(P_1, C)$. Assume that $OK_n(P_1, C)$ does not hold. Then, we have (1) $\exists \sigma$ and Q s.t. $\langle P_1, C \rangle \xrightarrow{\sigma} \langle Q, C' \rangle$ and $\sharp_m(\sigma) - \sharp_f(\sigma) > n$.

From the definition of that $OK_n((*x)(P_1, P_2), C)$ holds, we have (1) if $\langle (*x)(P_1, P_2), C \rangle \rightarrow \langle P_1, C \rangle \xrightarrow{\sigma} \langle Q, C' \rangle$, then $\sharp_m(\sigma) - \sharp_f(\sigma) \leq n$, which is in contradiction to the assumption. Therefore, $OK_n(P_1, C)$ holds.

- Case $P = (*x)(P_1, P_2)$ and $\frac{\neg \mathbf{null}(*x) \in C \quad \mathbf{const}(*x) \in C}{\langle (*x)(P_1, P_2), C \rangle \rightarrow \langle P_2, C \rangle}$

We need to prove $OK_n(P_2, C)$. Assume that $OK_n(P_2, C)$ does not hold. Then we have (1) $\exists \sigma$ and Q s.t. $\langle P_2, C \rangle \xrightarrow{\sigma} \langle Q, C' \rangle$ and $\sharp_m(\sigma) - \sharp_f(\sigma) > n$.

From the definition of that $OK_n((*x)(P_1, P_2), C)$ holds, we have (1) if $\langle (*x)(P_1, P_2), C \rangle \rightarrow \langle P_2, C \rangle \xrightarrow{\sigma} \langle Q, C' \rangle$, then $\sharp_m(\sigma) - \sharp_f(\sigma) \leq n$, which is in contradiction to the assumption. Therefore, $OK_n(P_2, C)$ holds.

- Case $P = (*x)(P_1, P_2)$ and $\frac{\mathbf{null}(*x), \neg \mathbf{null}(*x) \notin C \quad \mathbf{const}(*x) \in C}{\langle (*x)(P_1, P_2), C \rangle \xrightarrow{\mathbf{null}(*x)} \langle P_1, C \cup \{\mathbf{null}(*x)\} \rangle}$

We need to prove $OK_n(P_1, C \cup \{\mathbf{null}(*x)\})$. Assume that $OK_n(P_1, C \cup \{\mathbf{null}(*x)\})$ does not hold. Then we have $\exists \sigma$ and Q s.t. $\langle P_1, C \cup \{\mathbf{null}(*x)\} \rangle \xrightarrow{\sigma} \langle Q, C' \rangle$ and $\sharp_m(\sigma) - \sharp_f(\sigma) > n$.

From the definition of that $OK_n((*x)(P_1, P_2), C)$ holds, we have if $\langle (*x)(P_1, P_2), C \rangle \xrightarrow{\mathbf{null}(*x)} \langle P_1, C \cup \{\mathbf{null}(*x)\} \rangle \xrightarrow{\sigma} \langle Q, C' \rangle$, then $\sharp_m(\sigma) - \sharp_f(\sigma) \leq n$. Therefore, we get the contradiction and $OK_n(P_1, C \cup \{\mathbf{null}(*x)\})$ holds.

- Case $P = (*x)(P_1, P_2)$ and $\frac{\mathbf{null}(*x), \neg \mathbf{null}(*x) \notin C \quad \mathbf{const}(*x) \in C}{\langle (*x)(P_1, P_2), C \rangle \xrightarrow{\neg \mathbf{null}(*x)} \langle P_2, C \cup \{\neg \mathbf{null}(*x)\} \rangle}$

Similar to the above.

- Case $P = \mathbf{const}(*x)P'$ and $\langle \mathbf{const}(*x)P', C \rangle \rightarrow \langle P'; \mathbf{endconst}(*x), C \cup \mathbf{const}(*x) \rangle$

We need to prove $OK_n(P'; \mathbf{endconst}(*x), C \cup \mathbf{const}(*x))$. Assume that $OK_n(P'; \mathbf{endconst}(*x), C \cup \mathbf{const}(*x))$ does not hold. Then, we have $\exists \sigma$ and Q s.t. $\langle P'; \mathbf{endconst}(*x), C \cup \mathbf{const}(*x) \rangle \xrightarrow{\sigma} \langle Q, C' \rangle$ and $\sharp_m(\sigma) - \sharp_f(\sigma) > n$.

From the definition of that $OK_n(\mathbf{const}(*x)P', C)$ holds, we have (1) if $\langle \mathbf{const}(*x)P', C \rangle \rightarrow \langle P'; \mathbf{endconst}(*x), C \cup \mathbf{const}(*x) \rangle \xrightarrow{\sigma} \langle Q, C' \rangle$, then $\sharp_m(\sigma) - \sharp_f(\sigma) \leq n$, which is in contradiction to the assumption. Therefore, $OK_n(P'; \mathbf{endconst}(*x), C \cup \mathbf{const}(*x))$ holds.

- Case $P = \mu\alpha.P'$ and $\langle \mu\alpha.P', C \rangle \rightarrow \langle [\mu\alpha.P'/\alpha]P', C \rangle$

We need to prove $OK_n([\mu\alpha.P'/\alpha]P', C)$. Assume that $OK_n([\mu\alpha.P'/\alpha]P', C)$ does not hold. Then, we have (1) $\exists \sigma$ and Q s.t. $\langle [\mu\alpha.P'/\alpha]P', C \rangle \xrightarrow{\sigma} \langle Q, C' \rangle$ and $\sharp_m(\sigma) - \sharp_f(\sigma) > n$.

From the definition of that $OK_n(\mu\alpha.P', C)$ holds, we have (1) if $\langle \mu\alpha.P', C \rangle \rightarrow \langle [\mu\alpha.P'/\alpha]P', C \rangle \xrightarrow{\sigma} \langle Q, C' \rangle$, then $\sharp_m(\sigma) - \sharp_f(\sigma) \leq n$, which is a contradiction. Therefore, $OK([\mu\alpha.P'/\alpha]P', C)$ holds.

- Case $P = P_1; P_2$ and $\frac{\langle P_1, C \rangle \xRightarrow{\rho} \langle P'_1, C' \rangle}{\langle P_1; P_2, C \rangle \xRightarrow{\rho} \langle P'_1; P_2, C' \rangle}$

We need to prove $OK_{n'}(P'_1; P_2, C)$, where n' is determined by

$$n' = \begin{cases} n+1 & \rho = \mathbf{free} \\ n-1 & \rho = \mathbf{malloc} \\ n & \text{Otherwise.} \end{cases}$$

Assume that $OK_{n'}(P'_1; P_2, C')$ does not hold. Then, we have (1) $\exists \sigma, Q$ and C'' s.t. $\langle P'_1; P_2, C' \rangle \xrightarrow{\sigma} \langle Q, C'' \rangle$ and $\sharp_m(\sigma) - \sharp_f(\sigma) > n'$.

From the definition of that $OK_n(P_1; P_2, C)$ holds, we have (1) if $\langle P_1; P_2, C \rangle \xRightarrow{\rho} \langle P'_1; P_2, C' \rangle \xrightarrow{\sigma} \langle Q, C'' \rangle$, then $\sharp_m(\rho\sigma) - \sharp_f(\rho\sigma) \leq n$.

From (1), we get $n' + \sharp_m(\rho) - \sharp_f(\rho) < \sharp_m(\rho) + \sharp_m(\sigma) - \sharp_f(\rho) - \sharp_f(\sigma) \leq n$. For any ρ , the $n' + \sharp_m(\rho) - \sharp_f(\rho) = n$, therefore we get a contradiction. Therefore, $OK_{n'}(P'_1; P_2, C')$ holds.

□

Lemma 4.2. *If consistency(H, R, C) and $\langle H, R, s, n, C \rangle \xrightarrow{\rho} \langle H', R', s', n', C' \rangle$, then consistency(H', R', C').*

Proof. By induction on $\langle H, R, s, n, C \rangle \xrightarrow{\rho} \langle H', R', s', n', C' \rangle$

- Case: $\langle H, R, \mathbf{const}(*y)s, n, C \rangle \rightarrow \langle H, R, s; \mathbf{endconst}(*y), n', C \cup \mathbf{const}(*y) \rangle$.

We need to prove consistency($H, R, C \cup \mathbf{const}(*y)$). We should prove for all x (1) if $\mathbf{null}(*x) \in C \cup \mathbf{const}(*y)$, then $\mathbf{const}(*x) \in C \cup \mathbf{const}(*y)$ and $H(R(x)) = \mathbf{null}$ if $H(R(x))$ is defined and (2) if $\neg \mathbf{null}(*x) \in C \cup \mathbf{const}(*y)$, then $\mathbf{const}(*x) \in C \cup \mathbf{const}(*y)$ and $H(R(x)) \neq \mathbf{null}$. By case analysis, let $\mathbf{const}(*y) \in C$, so $C \cup \mathbf{const}(*y)$ is C . Assuming $\mathbf{null}(*x) \in C$. By hypothesis, we have $\mathbf{const}(*x) \in C$ and $H(R(x)) = \mathbf{null}$ if $H(R(x))$ is defined. Therefore we got if $\mathbf{null}(*x) \in C \cup \mathbf{const}(*y)$, then $\mathbf{const}(*x) \in C \cup \mathbf{const}(*y)$ and $H(R(x)) = \mathbf{null}$ if $H(R(x))$ is defined. Let $\mathbf{const}(*y) \notin C$, and assume $\mathbf{null}(*x) \in C \cup \mathbf{const}(*y)$. By hypothesis, we have $\mathbf{const}(*x) \in C$ and $H(R(x)) = \mathbf{null}$ if $H(R(x))$ is defined. Because C is subset of $C \cup \mathbf{const}(*y)$, therefore we got if $\mathbf{null}(*x) \in C \cup \mathbf{const}(*y)$, then $\mathbf{const}(*x) \in C \cup \mathbf{const}(*y)$ and $H(R(x)) = \mathbf{null}$ if $H(R(x))$ is defined. So (1) holds. Similar to (1), (2) holds.

Therefore, consistency($H, R, C \cup \mathbf{const}(*y)$) holds.

- Case: $\langle H, R, \mathbf{endconst}(*y), n, C \rangle \rightarrow \langle H, R, \mathbf{skip}, n, C' \rangle$ where $C' = \mathbf{filter}(C, *y)$.

We need to prove consistency(H, R, C' where $C' = \mathbf{filter}(C, *y)$). From assumption consistency(H, R, C), we have for all x (1) C does not contain both $\mathbf{null}(*x)$ and $\neg \mathbf{null}(*x)$. From definition of function $\mathbf{filter}(C, *y)$, we know for all x C' does not contain both $\mathbf{null}(*x)$ and $\neg \mathbf{null}(*x)$.

(2) if $\mathbf{const}(*x) \in C$ and $\mathbf{null}(*x) \in C$, then $H(R(x)) = \mathbf{null}$. Assume that $\mathbf{const}(*x) \in C$ and $\mathbf{null}(*x) \in C$, then from the definition of $\mathbf{filter}(C, *y)$, we know $\mathbf{const}(*x) \in C'$ and $\mathbf{null}(*x) \in C'$, and H and R do not change, so $H(R(x)) = \mathbf{null}$. Therefore, for all x if $\mathbf{const}(*x) \in C'$ and $\mathbf{null}(*x) \in C'$, then $H(R(x)) = \mathbf{null}$. (3) similar to (2).

Therefore, $\mathbf{consistency}(H, R, C')$ holds.

- Case: $\langle H, R, \mathbf{free}(y), n, C \rangle \xrightarrow{\mathbf{free}} \langle H \setminus R(y), R, \mathbf{skip}, n + 1, C \rangle$.

We need to prove $\mathbf{consistency}(H \setminus R(y), R, C)$. From assumption $\mathbf{consistency}(H, R, C)$, we have for all x (1) C does not contain both $\mathbf{null}(*x)$ and $\neg \mathbf{null}(*x)$. (2) if $\mathbf{const}(*x) \in C$ and $\mathbf{null}(*x) \in C$, then $H(R(x)) = \mathbf{null}$. Assume that $\mathbf{const}(*x) \in C$ and $\mathbf{null}(*x) \in C$, and we know $(H \setminus R(y))(R(x)) = \mathbf{null}$. Therefore, for all x , if $\mathbf{const}(*x) \in C$ and $\mathbf{null}(*x) \in C$, then $(H \setminus R(y))(R(x)) = \mathbf{null}$. (3) similar to (2).

Therefore, $\mathbf{consistency}(H \setminus R(y), R, C)$ holds.

- Case: $\langle H, R, \mathbf{let } y = \mathbf{mallocin } s, n, C \rangle \xrightarrow{\mathbf{malloc}} \langle H \{l \mapsto v\}, R \{x' \mapsto l\}, [x'/y]s, n', C \rangle$.

We need to prove $\mathbf{consistency}(H \{l \mapsto v\}, R \{x' \mapsto l\}, C)$. From assumption $\mathbf{consistency}(H, R, C)$, we have for all x (1) C does not contain both $\mathbf{null}(*x)$ and $\neg \mathbf{null}(*x)$. (2) if $\mathbf{const}(*x) \in C$ and $\mathbf{null}(*x) \in C$, then $H(R(x)) = \mathbf{null}$. Assume that $\mathbf{const}(*x) \in C$ and $\mathbf{null}(*x) \in C$, then we have $H(R(x)) = \mathbf{null}$, and we know $H \{l \mapsto v\}(R \{x' \mapsto l\}(x)) = \mathbf{null}$. Therefore, we get for all x , if $\mathbf{const}(*x) \in C$ and $\mathbf{null}(*x) \in C$, then $H \{l \mapsto v\}(R \{x' \mapsto l\}(x)) = \mathbf{null}$. (3) similar to (2).

Therefore, $\mathbf{consistency}(H \{l \mapsto v\}, R \{x' \mapsto l\}, C)$ holds.

- Case: $\langle H, R, \mathbf{skip}; s, n, C \rangle \rightarrow \langle H, R, s, n', C \rangle$.

We need to prove $\mathbf{consistency}(H, R, C)$. Obviously, from assumption $\mathbf{consistency}(H, R, C)$.

- Case: $\langle H \{R(w) \mapsto v\}, R, *w \leftarrow y, n, C \rangle \rightarrow \langle H \{R(w) \mapsto R(y)\}, R, \mathbf{skip}, n, C \rangle$ where $\forall z. R(w) = R(z) \Rightarrow \mathbf{const}(*z) \notin C$

We need to prove $\mathbf{consistency}(H \{R(w) \mapsto R(y)\}, R, C)$. From assumption $\mathbf{consistency}(H, R, C)$, we have for all x (1) C does not contain both $\mathbf{null}(*x)$ and $\neg \mathbf{null}(*x)$ (2) if $\mathbf{const}(*x) \in C$ and $\mathbf{null}(*x) \in C$, then $H \{R(w) \mapsto v\}(R(x)) = \mathbf{null}$. Assume that $\mathbf{const}(*x) \in C$ and $\mathbf{null}(*x) \in C$, then we have $H \{R(w) \mapsto v\}(R(x)) = \mathbf{null}$, and we know $H \{R(w) \mapsto R(y)\}(R(x)) = \mathbf{null}$. Therefore, for all x , if $\mathbf{const}(*x) \in C$ and $\mathbf{null}(*x) \in C$, then $H \{R(w) \mapsto R(y)\}(R(x)) = \mathbf{null}$. (3) similar to (2).

Therefore, $\mathbf{consistency}(H \{R(w) \mapsto R(y)\}, R, C)$ holds.

- Case: $\langle H, R, \mathbf{let } z = \mathbf{yin } s, n, C \rangle \rightarrow \langle H, R \{z' \mapsto R(y)\}, [z'/z]s, n, C \rangle$

We need to prove $\mathbf{consistency}(H, R \{z' \mapsto R(y)\}, C)$. From assumption $\mathbf{consistency}(H, R, C)$, we have for all x (1) C does not contain both $\mathbf{null}(*x)$ and $\neg \mathbf{null}(*x)$ (2) if $\mathbf{const}(*x) \in C$ and $\mathbf{null}(*x) \in C$, then $H(R(x)) = \mathbf{null}$. Assume that $\mathbf{const}(*x) \in C$ and $\mathbf{null}(*x) \in C$, then we have $H(R(x)) = \mathbf{null}$, and we get $H(R \{z' \mapsto R(y)\}) = \mathbf{null}$. Therefore, for all x , if $\mathbf{const}(*x) \in C$ and $\mathbf{null}(*x) \in C$, then $H(R \{z' \mapsto R(y)\}) = \mathbf{null}$. (3) similar to (2).

Therefore, $\mathbf{consistency}(H, R \{z' \mapsto R(y)\}, C)$ holds.

- Case: $\langle H, R, \text{ifnull}(*y) \text{ then } s_1 \text{ else } s_2, n, C \rangle \xrightarrow{\text{null}(*y)} \langle H, R, s_1, n, C \rangle$ where $H(R(y)) = \text{null}$ and $\text{const}(*y) \notin C$

We need to prove $\text{consistency}(H, R, C)$. Obviously, $\text{consistency}(H, R, C)$ holds from assumption.

- Case: $\langle H, R, \text{ifnull}(*y) \text{ then } s_1 \text{ else } s_2, n, C \rangle \xrightarrow{\neg \text{null}(*y)} \langle H, R, s_2, n, C \rangle$ where $H(R(y)) \neq \text{null}$ and $\text{const}(*y) \notin C$

We need to prove $\text{consistency}(H, R, C)$. Obviously, $\text{consistency}(H, R, C)$ holds from assumption.

- Case: $\langle H, R, \text{ifnull}(*y) \text{ then } s_1 \text{ else } s_2, n, C \rangle \xrightarrow{\text{null}(*y)} \langle H, R, s_1, n, C \cup \text{null}(*y) \rangle$ where $H(R(y)) = \text{null}$ and $\text{const}(*y) \in C$

We need to prove $\text{consistency}(H, R, C \cup \text{null}(*y))$. From assumption $\text{consistency}(H, R, C)$, we have for all x (1) C does not contain both $\text{null}(*x)$ and $\neg \text{null}(*x)$. Assume that $\neg \text{null}(*y) \in C$, and because we know $\text{const}(*y) \in C$, then $H(R(y)) \neq \text{null}$. Then we get the contradiction $H(R(y))$ should be null. Therefore $\neg \text{null}(*y) \notin C$. Then we get for all x , $C \cup \text{null}(*y)$ does not contain both $\text{null}(*x)$ and $\neg \text{null}(*x)$. (2) if $\text{const}(*x) \in C$ and $\text{null}(*x) \in C$, then $H(R(x)) = \text{null}$. Assume that $\text{const}(*x) \in C$ and $\text{null}(*x) \in C$, then we have $H(R(x)) = \text{null}$, therefore $\text{const}(*x) \in C \cup \text{null}(*y)$ and $\text{null}(*x) \in C \cup \text{null}(*y)$. H and R do not change, so $H(R(x)) = \text{null}$. Therefore, we get for all x , if $\text{const}(*x) \in C \cup \text{null}(*y)$ and $\text{null}(*x) \in C \cup \text{null}(*y)$, then $H(R(x)) = \text{null}$. (3) similar to (2).

Therefore, $\text{consistency}(H, R, C \cup *y)$ holds.

□

Proof of Lemma 2.2: By induction on the derivation of $\langle H, R, s, n, C \rangle \xrightarrow{\rho} \langle H', R', s', n', C' \rangle$.

- Case: $\langle H, R, \text{const}(*x)s, n, C \rangle \rightarrow \langle H, R, s; \text{endconst}(*x), n, C \cup \{\text{const}(*x)\} \rangle$

From the assumption $\Theta; \Gamma \vdash \langle H, R, \text{const}(*x)s, n, C \rangle : \langle P, C \rangle$, we have $\Theta; \Gamma \vdash \text{const}(*x)s : P$, $OK_n(P, C)$ and $\text{consistency}(H, R, C)$. From the inversion of typing rules, we get $\Theta; \Gamma \vdash s : P''$ and $\text{const}(*x)P'' \leq P$ for some P'' . By subtyping, we have $P''; \text{endconst}(*x) \leq Q$ and $\langle P, C \rangle \Rightarrow \langle Q, C \cup \{\text{const}(*x)\} \rangle$ for some Q .

we need to find P' and C' s.t. $\Theta; \Gamma \vdash s; \text{endconst}(*x) : P'$, $OK_n(P', C')$, $\langle P, C \rangle \Rightarrow \langle P', C' \rangle$ and $\text{consistency}(H, R, C')$. Taking Q as P' and $C \cup \{\text{const}(*x)\}$ as C' . Therefore $\langle P, C \rangle \rightarrow \langle P', C' \rangle$ holds, and then $OK_n(P', C')$ and $\text{consistency}(H, R, C')$ hold from Lemma 4.1 and Lemma 4.2. From $\Theta; \Gamma \vdash s; \text{endconst}(*x) : P''; \text{endconst}(*x), P''; \text{endconst}(*x) \leq Q$ and T-SUB, $\Theta; \Gamma \vdash s; \text{endconst}(*x) : P'$ holds.

- Case: $\langle H, R, \text{endconst}(*x), n, C \rangle \rightarrow \langle H, R, \text{skip}, n, C' \rangle$ where $C' = \text{filter}(C, *x)$

From the assumption $\Theta; \Gamma \vdash \langle H, R, \text{endconst}(*x), n, C \rangle : \langle P, C \rangle$, we have $\Theta; \Gamma \vdash \text{endconst}(*x) : P$, $OK_n(P, C)$ and $\text{consistency}(H, R, C)$. From the inversion of typing rules, we get $\Theta; \Gamma \vdash \text{endconst}(*x) : \text{endconst}(*x)$ and $\text{endconst}(*x) \leq P$. By subtyping, we get $0 \leq Q$ and $\langle P, C \rangle \rightarrow \langle Q, C \rangle$ for some Q .

we need to find P' and C' s.t. $\Theta; \Gamma \vdash \text{skip} : P'$, $OK_n(P', C')$, $\langle P, C \rangle \Rightarrow \langle P', C' \rangle$ and $\text{consistency}(H, R, C')$. Taking Q as P' and C as C' , then $\langle P, C \rangle \rightarrow \langle P', C' \rangle$ holds, and then

$OK_n(P', C')$ and $consistency(H, R, C')$ hold from Lemma 4.1 and Lemma 4.2. From T-SKIP, T-SUB and $0 \leq Q$, then $\Theta; \Gamma \vdash \mathbf{skip} : P'$ holds.

- Case: $\langle H, R, \mathbf{free}(x), n, C \rangle \xrightarrow{\mathbf{free}} \langle H', R, \mathbf{skip}, n+1, C' \rangle$

From the assumption $\Theta; \Gamma \vdash \langle H, R, \mathbf{free}(x), n, C \rangle : \langle P, C \rangle$, we have $OK_n(P, C)$, $consistency(H, R, C)$ and $\Theta; \Gamma \vdash \mathbf{free}(x) : P$. From inversion of the typing rules, we have $\Theta; \Gamma \vdash \mathbf{free}(x) : \mathbf{free}$ and $\mathbf{free} \leq P$. By the subtyping, we have $\langle P, F \rangle \xrightarrow{\mathbf{free}} \langle Q, C' \rangle$ and $0 \leq Q$ for some Q .

We need to find P' and C' such that $\langle P, C \rangle \xrightarrow{\mathbf{free}} \langle P', C' \rangle$, $\Theta; \Gamma \vdash \mathbf{skip} : P'$, and $OK_{n+1}(P', C')$.

Take Q as P' and C as C' . Then, $\langle P, C \rangle \xrightarrow{\mathbf{free}} \langle P', C' \rangle$ holds, and $OK_{n+1}(P', C')$ holds from Lemma 4.1. We also have $\Theta; \Gamma \vdash \mathbf{skip} : P'$ from T-SKIP, $0 \leq Q$ and T-SUB.

- Case: $\langle H, R, \mathbf{let } x = \mathbf{malloc}() \text{ in } s, n, C \rangle \xrightarrow{\mathbf{malloc}} \langle H', R', [x'/x]s, n-1, C' \rangle$

From the assumption $\Theta; \Gamma \vdash \langle H, R, \mathbf{let } x = \mathbf{malloc}() \text{ in } s, n, C \rangle : \langle P, C \rangle$, we have $\Theta; \Gamma \vdash \mathbf{let } x = \mathbf{malloc}() \text{ in } s : P$, $OK_n(P, C)$ and $consistency(H, R, C)$. By the inversion of typing rules, we have $\Theta; \Gamma, x \vdash s : P''$ and $\mathbf{malloc}; (x)P'' \leq P$ for some P'' . By subtyping, we get $\langle P, C \rangle \xrightarrow{\mathbf{malloc}} \langle Q, F \rangle$ and $[x'/x]P'' \leq Q$ for some Q .

We need to find P' and C' such that $\Theta; \Gamma, x' \vdash [x'/x]s : P'$, $\langle P, C \rangle \xrightarrow{\mathbf{malloc}} \langle P', C' \rangle$, $consistency(H', R', C')$ and $OK_{n-1}(P', C')$. Take Q as P' and C as C' . Then $\langle P, C \rangle \xrightarrow{\mathbf{malloc}} \langle P', C' \rangle$ holds, and then $OK_{n-1}(P', C')$ and $consistency(H', R', C')$ hold by Lemma 4.1 and Lemma 4.2. From $\Theta; \Gamma, x \vdash s : P''$ and $\mathbf{malloc}; (x)P'' \leq P$, we have $\Theta; \Gamma, x'' \vdash [x''/x]s : [x''/x]P''$ and $\mathbf{malloc}; (x)P'' \leq P$, and then by the definition of subtyping we have $[x''/x]P'' \leq Q'$ for some Q' . Therefore, we get $\Theta; \Gamma, x'' \vdash [x''/x]s : Q'$. Take x'' as x' and Q' as P' , then $\Theta; \Gamma, x' \vdash [x'/x]s : P'$ holds.

- Case: $\langle H, R, \mathbf{skip}; s, n, C \rangle \rightarrow \langle H, R, s, n, C \rangle$

From the assumption $\Theta; \Gamma \vdash \langle H, R, \mathbf{skip}; s, n, C \rangle : \langle P, C \rangle$, we have $\Theta; \Gamma \vdash \mathbf{skip}; s : P$, $OK_n(P, C)$ and $consistency(H, R, C)$. From the inversion of the typing rules, we get $\Theta; \Gamma \vdash s : P''$ and $0; P'' \leq P$. From the definition of subtyping, we have $\langle P, C \rangle \Rightarrow \langle Q, C \rangle$ and $P'' \leq Q$ for some Q .

We need to find P' and C' such that $\Theta; \Gamma \vdash s : P'$ and $\langle P, C \rangle \rightarrow \langle P', C' \rangle$ and $OK_n(P', C')$. Take Q as P' and C as C' . Then $\langle P, C \rangle \Rightarrow \langle P', C' \rangle$ holds, and then $OK_n(P', C')$ and $consistency(H, R, C')$ hold. We also have $\Theta; \Gamma \vdash s : P'$ from T-SUB, $\Gamma \vdash s : P''$ and $P'' \leq Q$.

- Case: $\langle H, R, *x \leftarrow y, n, C \rangle \rightarrow \langle H', R, \mathbf{skip}, n, C' \rangle$

From the assumption $\Theta; \Gamma \vdash \langle H, R, *x \leftarrow y, n, C \rangle : \langle P, C \rangle$, we have $\Theta; \Gamma \vdash *x \leftarrow y : P$, $OK_n(P, C)$ and $consistency(H, R, C)$. From the inversion of typing rules, we have $0 \leq P$.

We need to find P' and C' such that $\Theta; \Gamma \vdash \mathbf{skip} : P'$, $\langle P, C \rangle \Rightarrow \langle P', C' \rangle$ and $OK_n(P', C')$. Take P as P' and C as C' . Then $\langle P, C \rangle \Rightarrow \langle P', C' \rangle$ holds, and then $OK_n(P', C')$ and $consistency(H', R, C')$ hold from Lemma 4.1 and Lemma 4.2. We also have $\Theta; \Gamma \vdash \mathbf{skip} : P'$ from T-SKIP, $0 \leq P$ and T-SUB.

- Case: $\langle H, R, \mathbf{let } x = y \text{ in } s, n, C \rangle \rightarrow \langle H, R', [x'/x]s, n, C' \rangle$

From the assumption $\Theta; \Gamma \vdash \langle H, R, \mathbf{let } x = y \text{ in } s, n, C \rangle : \langle P, C \rangle$, we have $\Theta; \Gamma, y \vdash \mathbf{let } x = y \text{ in } s : P$, $OK_n(P, C)$ and $consistency(H, R, C)$. From the inversion of typing rules, we

have $\Theta; \Gamma, x, y \vdash s : P''$ and $\text{let } x = y \text{ in } P'' \leq P$ for some P'' . By subtyping, we have $\langle P, C \rangle \rightarrow \langle Q, C \rangle$ and $[x'/x]P'' \leq Q$ for some Q .

We need to find P' and C' such that $\Theta; \Gamma, x', y \vdash [x'/x]s : P'$, $\langle P, C \rangle \rightarrow \langle P', C' \rangle$, $OK_n(P', C')$ and $\text{consistency}(H, R', C')$. Take Q as P' and C as C' . Then $\langle P, C \rangle \Rightarrow \langle P', C' \rangle$ and $OK_n(P', C')$ hold. From $\Theta; \Gamma, x, y \vdash s : P''$ and $\text{let } x = y \text{ in } P'' \leq P$, we have $\Theta; \Gamma, x'', y \vdash [x''/x]s : [x''/x]P''$ and $\text{let } x'' = y \text{ in } [x''/x]P'' \leq P$, and then by subtyping we have $[x''/x]P'' \leq Q'$ for some Q' . Therefore, we have $\Theta; \Gamma, x'', y \vdash [x''/x]s : Q'$. Take x'' as x' and Q' as P' , then $\Theta; \Gamma, x', y \vdash [x'/x]s : P'$ holds.

- Case: $\langle H, R, \text{let } x = \text{null in } s, n \rangle \rightarrow \langle H, R', [x'/x]s, n \rangle$

Similar to the above.

- Case: $\langle H, R, \text{let } x = *y \text{ in } s, n \rangle \rightarrow \langle H, R', [x'/x]s, n \rangle$

Similar to the above.

- Case: $\langle H, R, \text{ifnull } (*x) \text{ then } s_1 \text{ else } s_2, n, C \rangle \xrightarrow{\text{null}(*x)} \langle H, R, s_1, n, C \rangle$ if $H(R(x)) = \text{null}$ and $\text{const}(*x) \notin C$

From assumption $\Theta; \Gamma \vdash \langle H, R, \text{ifnull } (*x) \text{ then } s_1 \text{ else } s_2, n, C \rangle : \langle P, C \rangle$, we have $\Theta; \Gamma \vdash \text{ifnull } (*x) \text{ then } s_1 \text{ else } s_2 : P$, $OK_n(P, C)$ and $\text{consistency}(H, R, C)$. From the inversion of typing rules, we have $\Theta; \Gamma \vdash s_1 : P_1$, $\Theta; \Gamma \vdash s_2 : P_2$ and $(*x)(P_1, P_2) \leq P$. According to the rule Tr-NotConst1 and $\text{const}(*x) \notin C$, we have $\langle (*x)(P_1, P_2) \rangle \xrightarrow{\text{null}(*x)} \langle P_1, C \rangle$, and then by definition of subtyping, we get $\langle P, C \rangle \xrightarrow{\text{null}(*x)} \langle Q, C \rangle$ and $P_1 \leq Q$ for some Q .

We need to find P' and C' such that $\Theta; \Gamma \vdash s_1 : P'$, $\langle P, C \rangle \xrightarrow{\text{null}(*x)} \langle P', C' \rangle$ and $OK_n(P', C')$.

Take Q as P' and C as C' . Then $\langle P, C \rangle \xrightarrow{\text{null}(*x)} \langle P', C' \rangle$ and $OK_n(P', C')$ hold. We also have $\Theta; \Gamma \vdash s_1 : P'$ from T-SUB, $\Theta; \Gamma \vdash s_1 : P_1$ and $P_1 \leq Q$.

- Case: $\langle H, R, \text{ifnull } (*x) \text{ then } s_1 \text{ else } s_2, n, C \rangle \xrightarrow{\neg \text{null}(*x)} \langle H, R, s_1, n, C \rangle$ if $H(R(x)) \neq \text{null}$ and $\text{const}(*x) \notin C$

Similar to the above.

- Case: $\langle H, R, \text{ifnull } (*x) \text{ then } s_1 \text{ else } s_2, n, C \rangle \xrightarrow{\text{null}(*x)} \langle H, R, s_1, n, C' \rangle$ if $H(R(x)) = \text{null}$, $\text{const}(*x) \in C$ and $C' = C \cup \{\text{null}(*x)\}$

From assumption $\Theta; \Gamma \vdash \langle H, R, \text{ifnull } (*x) \text{ then } s_1 \text{ else } s_2, n, C \rangle : \langle P, C \rangle$, we have $\Theta; \Gamma \vdash \text{ifnull } (*x) \text{ then } s_1 \text{ else } s_2 : P$, $OK_n(P, C)$ and $\text{consistency}(H, R, C)$. From the inversion of typing rules, we have $\Theta; \Gamma \vdash s_1 : P_1$, $\Theta; \Gamma \vdash s_2 : P_2$ and $(*x)(P_1, P_2) \leq P$. According to rule Tr-NullNotIn1, $\text{const}(*x) \in C$ and $C' = C \cup \{\text{null}(*x)\}$, we have $\langle (*x)(P_1, P_2) \rangle \xrightarrow{\text{null}(*x)} \langle P_1, C \cup \{\text{null}(*x)\} \rangle$, and then by the definition of subtyping, we get $\langle P, C \rangle \xrightarrow{\text{null}(*x)} \langle Q, C \cup \{\text{null}(*x)\} \rangle$ and $P_1 \leq Q$ for some Q .

We need to find P' and C' such that $\Theta; \Gamma \vdash s_1 : P'$, $\langle P, C \rangle \xrightarrow{\text{null}(*x)} \langle P', C' \rangle$, $OK_n(P', C')$ and $\text{consistency}(H, R, C')$. Take Q as P' and $C \cup \{\text{null}(*x)\}$ as C' . Then $\langle P, C \rangle \xrightarrow{\text{null}(*x)} \langle P', C' \rangle$ holds, and then $OK_n(P', C')$ and $\text{consistency}(H, R, C')$ hold by Lemma 4.1 and Lemma 4.2. We also have $\Theta; \Gamma \vdash s_1 : P'$ from T-SUB, $\Theta; \Gamma \vdash s_1 : P_1$ and $P_1 \leq Q$.

- Case: $\langle H, R, \text{ifnull}(*x) \text{ then } s_1 \text{ else } s_2, n, C \rangle \xrightarrow{\neg \text{null}(*x)} \langle H, R, s_2, n, C' \rangle$ if $H(R(x)) \neq \text{null}$, $\text{const}(*x) \in C$ and $C' = C \cup \{\neg \text{null}(*x)\}$

Similar to the above proof.

- Case: $\langle H, R, s_1; s_2, n, C \rangle \rightarrow \langle H', R', s'_1; s'_2, n', C' \rangle$

From the assumption $\Theta; \Gamma \vdash \langle H, R, s_1; s_2, n, C \rangle : \langle P, C \rangle$, we have $\Theta; \Gamma \vdash s_1; s_2 : P$, $OK_n(P, C)$ and $\text{consistency}(H, R, C)$. By inversion of typing rules, we have $\Theta; \Gamma \vdash s_1 : P_1$, $\Theta; \Gamma \vdash s_2 : P_2$ and $P_1; P_2 \leq P$ for some P_1 and P_2 .

By IH on $\langle H, R, s_1, n, C \rangle$ with derivation $\langle H, R, s_1, n, C \rangle \xrightarrow{\rho} \langle H', R', s'_1, n', C' \rangle$, we have $\exists P'_1, C'_1$ s.t. $\Theta; \Gamma \vdash \langle H', R', s'_1, n', C' \rangle : \langle P'_1, C'_1 \rangle$ and $\langle P_1, C \rangle \xrightarrow{\rho} \langle P'_1, C'_1 \rangle$.

By subtyping we have $\langle P, C \rangle \xrightarrow{\rho} \langle Q, C'_1 \rangle$ and $P'_1; P_2 \leq Q$ for some Q .

We need to find P' and C' s.t. $\langle P, C \rangle \xrightarrow{\rho} \langle P', C' \rangle$, $OK_n(P', C')$ and $\Theta; \Gamma \vdash s'_1; s'_2 : P'$. Take Q as P' and C'_1 as C' , $\langle P, C \rangle \xrightarrow{\rho} \langle P', C' \rangle$ and $OK_n(P', C')$ hold. By T-Sub, $\Theta; \Gamma \vdash s'_1; s'_2 : P'_1; P_2$ and $P'_1; P_2 \leq Q$, we have $\Theta; \Gamma \vdash s'_1; s'_2 : P'$ holds.

□

We write $\langle H, R, s, n, C \rangle \xrightarrow{\rho}$ if there is a transition $\xrightarrow{\rho}$ from $\langle H, R, s, n, C \rangle$.

Lemma 4.3. *If $\Theta \vdash \langle H, R, s, n, C \rangle : \langle P, C \rangle$ and $\langle H, R, s, n, C \rangle \xrightarrow{\rho}$ and $\rho \in \{\text{malloc}, \text{free}, \text{null}(*x), \neg \text{null}(*x)\}$, then there exists P' and C' such that $\langle P, C \rangle \xrightarrow{\rho} \langle P', C' \rangle$.*

Proof. Induction on the derivation of $\Theta; \Gamma \vdash \langle H, R, s, n, C \rangle : \langle P, C \rangle$.

□

Proof of Lemma 2.3:

By contradiction. Assume $\langle H, R, s, n, C \rangle \xrightarrow{\rho} \text{OutOfMemory}$. Then, n is 0 and $\rho = \text{malloc}$ from SEM-OUTOFMEM. From the assumption we have $\Theta; \Gamma \vdash s : P$ and $OK_0(P, C)$. From Lemma 4.3, there exists P' and C' such that $\langle P, C \rangle \xrightarrow{\text{malloc}} \langle P', C' \rangle$. However, this contradicts $OK_0(P, C)$.

□

Proof of Theorem 2.1:

We have $\Theta; \emptyset \vdash s : P, \vdash D : \Theta$, $OK_n(P, C)$ and $\text{consistency}(H, R, C)$.

Suppose that there exists σ such that $\langle \emptyset, \emptyset, s, n, C \rangle \xrightarrow{\sigma} \langle H', R', s', n', C' \rangle \xrightarrow{\rho} \text{OutOfMemory}$. Then, $n' = 0$ and $\rho = \text{malloc}$. From Lemma 2.2, there exists P' and C' such that $\Theta; \Gamma' \vdash s' : P'$, $\langle P, C \rangle \xrightarrow{\sigma} \langle P', C' \rangle$, and $OK_0(P', C')$; hence $\langle H', R', s', 0 \rangle \xrightarrow{\text{malloc}}$. However, this contradicts Lemma 2.3.

□ `lllllll cbf9f98628ddf24a08c69bd0bc0f8a8c90557969`