Eiffel0: An Object-Oriented Language with Dynamic Frame Contracts

Bernd Schoeller
bernd.schoeller@inf.ethz.ch

December 12, 2006

ETH Zurich, Departement Informatik, Technical Report 542

1 Introduction

The purpose of this document is to define a very small language Eiffel0 as a subset of Eiffel[1]. The definition comes with a heap model and an operational semantics of the execution of Eiffel0 programs within the heap.

Eiffel0 is a subset of Eiffel. It main purpose is to explore reasoning about reference structures, aliasing and frames. Eiffel0 does not include:

- **Inheritance/Subtyping**: We are exploring dynamic structure of objects. As a first step, we do this investigation without dynamic binding.
- **Genericity**: As we do not support inheritance/subtyping, we can flatten out any class that defines a generic parameter to a class that uses the generic constraint (or ANY) as a replacement to the generic parameter.
- **Expanded types**: We model integers and booleans through the use of immutable classes and constant references to instances of these classes.
- **Exceptions**: Eiffel0 is used to reason about program correctness. The handling of exceptions is part of program robustness and not covered in Eiffel0.

Eiffel0 also removes some other constructs like *agents*, *once*, *loop variants and invariants*, *creation procedures*, *tuples*, *select statements*, *elseif branches*, *constants*, *indexing clauses*, *tags* and *comments*. Also, it does not introduce concepts of ECMA Eiffel like *attached types* or *conversion*.

As an addition to the Eiffel language, we introduce frame specifications by using *modify* and *use* expressions. The basic types integer and boolean are mapped to immutable references as there are no expanded types. An extra type of immutable sets of objects is introduced as well.

2 Grammar

ClassDefinition ::= **class** *ClassID*
    create
        RoutineID, . . .
    feature
        (*RoutineDefinition*|*AttributeDefinition*)*
    invariant
        Expression
    end

```
RoutineDefinition       ::= RoutineID [(LocalID:ClassID, . . . )] [:ClassID] is
                                  [ require
                                        Expression ]
                                  [ (use|modify)
                                        Expression ]
                                  [local
                                        LocalID:ClassID, . . . ]
                                  do
                                        Instruction*
                                  [ ensure
                                        Expression ]
                                  end

AttributeDefinition     ::= AttributeID:ClassID

Instruction             ::= CreateInstruction|LoopInstruction|IfInstruction|AssignmentInstruction|
                            CallInstruction

CreateInstruction       ::= create LocalID.RoutineID[(Expression,. . . )]

LoopInstruction         ::= from
                            until
                                  Expression
                            loop
                                  Instruction*
                            end

IfInstruction           ::= if
                                  Expression
                            then
                                  Instruction*
                            else
                                  Instruction*
                            end

AssignmentInstruction ::= ( AttributeId |LocalID ) : = Expression

CallInstruction         ::= [Expression.]RoutineID[(Expression,. . . )]

Expression              ::= QueryCall|AttributeRead|LocalID|ConstID|EqualityExpr |OldExpression

QueryCall               ::= [Expression.]RoutineID[(Expression,. . . )]

AttributeRead           ::= [Expression.]AttributeID

ExqualityExpression     ::= Expression = Expression

OldExpression           ::= old Expression
```

3 Memory Model

3.1 Heap

The memory model is based on a heap model suggested by Peter Müller[3]. The **Heap** is an abstract data type with the following operations:

$$\begin{array}{rcl}
\_.\_ & : & \mathit{Obj} \times \mathit{AttributeID} \to \mathit{Loc} \\
\_(\_) & : & \mathit{Heap} \times \mathit{Loc} \to \mathit{Obj} \\
\_\langle \_ := \_\rangle & : & \mathit{Heap} \times \mathit{Loc} \times \mathit{Obj} \to \mathit{Heap} \\
\mathsf{new} & : & \mathit{Heap} \times \mathit{ClassID} \to \mathit{Obj} \\
\_\langle \_\rangle & : & \mathit{Heap} \times \mathit{ClassID} \to \mathit{Heap} \\
\mathsf{alloc} & : & \mathit{Obj} \times \mathit{Heap} \to \text{Bool} \\
\mathsf{typeof} & : & \mathit{Obj} \to \mathit{ClassID}
\end{array}$$

Loc and **Obj** are disjoint sorts. The heap data type has the following properties:

$$L \neq K \vee f \neq g \Rightarrow H\langle L.f := X\rangle(K.g) = H(K.g) \tag{H1}$$
$$H\langle X.f := Y\rangle(X.f) = Y \tag{H2}$$
$$H\langle C\rangle(L) = H(L) \tag{H3}$$
$$\mathsf{alloc}(X, H\langle L := Y\rangle) \Leftrightarrow \mathsf{alloc}(X, H) \tag{H4}$$
$$\mathsf{alloc}(X, H\langle C\rangle) \Leftrightarrow \mathsf{alloc}(X, H) \vee X = \mathsf{new}(H, C) \tag{H5}$$
$$\mathsf{alloc}(H(L), H) \tag{H6}$$
$$\neg\mathsf{alloc}(\mathsf{new}(H, C), H) \tag{H7}$$
$$\mathsf{typeof}(\mathsf{new}(H, C)) = C \tag{H8}$$

For reasoning about object allocation, we have to talk about allocated and the non-allocated objects. We add the following two definitions:

$$\begin{array}{r}
\mathsf{Used}(h) \triangleq \{o \in \mathit{Obj} | \mathsf{alloc}(h, o)\} \\
\mathsf{Unused}(h) \triangleq \mathit{Obj} - \mathsf{Used}(h)
\end{array}$$

3.2 Memory safety

We assume that following a reference in on the heap from an allocated object will again yield an allocated object. Also, references stored in environments also lead to allocated objects:

$$\mathsf{alloc}(X, H) \Rightarrow \mathsf{alloc}(H(X.a), H) \tag{1}$$
$$\mathsf{alloc}(E(\_), H) \tag{2}$$

3.3 Local environment

Through the execution of each routine, we will keep track of arguments and local variables as part of the state. The is local environment is defined by the following abstract data type:

$$\begin{array}{rcl}
\_(\_) & : & \mathit{Env} \times \mathit{LocalID} \to \mathit{Obj} \\
\_\langle \_ := \_\rangle & : & \mathit{Env} \times \mathit{LocalID} \times \mathit{Obj} \to \mathit{Env}
\end{array}$$

The functions of the abstract data type have the following properties:

$$E\langle x := y\rangle(x) = y \tag{E2}$$
$$x \neq z \Rightarrow E\langle x := y\rangle(z) = E(z) \tag{E2}$$

3.4 State

The state s is a tuple (H, E) of a heap H and a local environment E. We will use the notation s_H if we want to refer to the heap of s and s_E if we want to refer to the environment of s.

4 Predefined objects and identifiers

There are a number of basic value types that are modeled by using immutable objects on the heap and constant identifiers (*ConstID*) to access these objects. This is done through the function const with the following signature:

$$\mathsf{const} : \mathit{ConstID} \rightarrow \mathit{Obj}$$

4.1 Boolean

A class `BOOLEAN` is defined that has two implicit instances *BooleanTrue* and *BooleanFalse*. Standard queries like **and**, **or** or **not** are defined in `BOOLEAN`. The constant identifiers **true** is used to access *BooleanTrue* and the constant identifier `false` is used to access *BooleanFalse*.

4.2 Integer

Similar to booleans, a class `INTEGER` is defined whose instances represent integer values. Regular mathematical operations are defined on `INTEGER`. The constant identifiers `0`,`1`,`-1`,`2`,`-2`,... are used to access these instances.

4.3 Void

`Void` is a *ConstID* that references the object representing nothingness. Any call const(`Void`).x is thus invalid and not correct.

4.4 Object Sets

As a last set of predefined objects, we define the class of all immutable, finite sets of objects. A class called `MML_SET` defines such objects. We assume to have standard operations defined on these sets like `extended`, `united`, etc. The empty set can be accessed through the *ConstID* "$\emptyset$".

4.5 Current

Every routine (command or query) has an implicit *LocalID* called `Current`. This implicit argument is bound to the target of the call.

4.6 Result

Any query definition has an implicit *LocalID* called `Result`. This variable is used to store the result value of a call.

5 Operational Semantics

5.1 Expressions

We will assume all expressions to be side-effect free (an assumption that is not enforced in Eiffel, but regarded as common practice through command/query separation [2]). As a result, we can define a function $Expr$ that is applied to an expression to compute its result. All expressions return **Obj**.

$$\mathsf{Expr}[\![c]\!](s) = \mathsf{const}(c) \tag{3}$$
$$\mathsf{Expr}[\![l]\!](s) = s_E(l) \tag{4}$$
$$\mathsf{Expr}[\![a]\!](s) = s_H(s_E(\texttt{Current}).a) \tag{5}$$
$$\mathsf{Expr}[\![E.a]\!](s) = s_H(\mathsf{Expr}[\![E]\!](s).a) \tag{6}$$
$$\mathsf{Expr}[\![E1 = E2]\!](s) = (\mathsf{Expr}[\![E1]\!](s) = \mathsf{Expr}[\![E2]\!](s)) \tag{7}$$
$$\frac{\langle \mathit{Body}_q, (s_H, \langle \texttt{Current} := \mathsf{Expr}[\![T]\!](s), \mathit{arg1} := \mathsf{Expr}[\![A1]\!](s), \dots \rangle)\rangle \to s'}{\mathsf{Expr}[\![T.q(A1, A2, \dots)]\!](s) = s'_E(\texttt{Result})} \tag{8}$$

(with $c \in \mathit{ConstID}$, $l, arg1, arg2 \in \mathit{LocalID}$ and $a \in \mathit{AttributeID}$. $E, E1, E2, T, A1, A2$ are expressions).

We demand (by rules 1 and 2) that the value returned by any expression is allocated if the call is valid and the target and all arguments are allocated as well. This leads to the — not directly obvious — necessity that the result computed by any query has to be an object that existed before the call to the query.

5.2 Ensure clauses

In ensure clauses, **old** references the pre-state of the computation. As a result, the ensure clause is not an expression on the state, but a relation between a pre-state s and a post-state s'.

$$\mathsf{OExpr}[\![c]\!](s, s') = \mathsf{const}(c) \tag{9}$$
$$\mathsf{OExpr}[\![l]\!](s, s') = s'_E(l) \tag{10}$$
$$\mathsf{OExpr}[\![a]\!](s, s') = s'_H(s'_E(\texttt{Current}).a) \tag{11}$$
$$\mathsf{OExpr}[\![E.a]\!](s, s') = s'_H(\mathsf{OExpr}[\![E]\!](s, s').a) \tag{12}$$
$$\mathsf{OExpr}[\![E1 = E2]\!](s, s') = \mathsf{OExpr}[\![E1]\!](s, s') = \mathsf{OExpr}[\![E2]\!](s, s') \tag{13}$$
$$\mathsf{OExpr}[\![\mathbf{old}\, E]\!](s, s') = \mathsf{Expr}[\![E]\!](s) \tag{14}$$
$$\frac{\langle \mathit{Body}_q, (s'_H, \langle \texttt{Current} := \mathsf{OExpr}[\![T]\!](s, s'), \mathit{arg1} := \mathsf{OExpr}[\![A1]\!](s, s'), \dots \rangle)\rangle \to s''}{\mathsf{OExpr}[\![T.q(A1, A2, \dots)]\!](s, s') = s''_E(\texttt{Result})} \tag{15}$$

5.3 Instructions

$$\langle , s\rangle \to s \tag{16}$$
$$\langle l := E, s\rangle \to (s_H, s_E\langle l := \mathsf{Expr}[\![E]\!](s)\rangle) \tag{17}$$
$$\langle a := E, s\rangle \to (s_H\langle \mathsf{Expr}[\![\texttt{Current}]\!](s).a := \mathsf{Expr}[\![E]\!](s)\rangle, s_E) \tag{18}$$
$$\frac{\langle S_1, s\rangle \to s', \langle S_1, s'\rangle \to s''}{\langle S_1\, S_2, s\rangle \to s''} \tag{19}$$
$$\frac{\langle S_1, s\rangle \to s'}{\langle \mathbf{if}\, b\, \mathbf{then}\, S_1\, \mathbf{else}\, S_2\, \mathbf{end}, s\rangle \to s'} \text{ iff } \mathsf{Expr}[\![b]\!](s) = \mathit{const}(\texttt{true}) \tag{20}$$
$$\frac{\langle S_2, s\rangle \to s''}{\langle \mathbf{if}\, b\, \mathbf{then}\, S_1\, \mathbf{else}\, S_2\, \mathbf{end}, s\rangle \to s''} \text{ iff} b(s) = \mathit{const}(\texttt{false}) \tag{21}$$
$$\frac{\langle S, s\rangle \to s', \langle \mathbf{from\, until}\, b\, \mathbf{loop}\, S\, \mathbf{end}, s'\rangle \to s''}{\langle \mathbf{from\, until}\, b\, \mathbf{loop}\, S\, \mathbf{end}, s\rangle \to s''} \text{ iff } \mathsf{Expr}[\![b]\!](s) = \mathit{const}(\texttt{true}) \tag{22}$$
$$\langle \mathbf{from\, until}\, b\, \mathbf{loop}\, S\, \mathbf{end}, s\rangle \to s \text{ iff } \mathsf{Expr}[\![b]\!](s) = \mathit{const}(\texttt{false}) \tag{23}$$
$$\frac{\langle \mathit{Body}_c, (s_H, \langle \texttt{Current} := \mathsf{Expr}[\![T]\!](s), \mathbf{arg1} := \mathsf{Expr}[\![A1]\!](s), \dots \rangle)\rangle \to s'}{\langle T.c(A1, A2, \dots), s\rangle \to (s'_H, s_E)} \tag{24}$$
$$\frac{\langle \mathit{Body}_c, (s_H\langle C\rangle, \langle \texttt{Current} := \mathsf{new}(s_H, C), \mathbf{arg1} := \mathsf{Expr}[\![a1]\!](s), \dots \rangle)\rangle \to s'}{\langle \mathbf{create}\, l.c(a1, a2, \dots), s\rangle \to (s'_H, s_E\langle l := \mathit{new}(s_H, C)\rangle)} \tag{25}$$

6 Syntactic Sugar

For `INTEGER`, `BOOLEAN` and `MML_SET`, there are the standard prefix, infix and postfix operators defined. These operators are mapped to feature calls. This enables us to write expressions in a natural way without increasing the complexity of the language. Here are some examples:

$$\begin{aligned}
a + b &\triangleq a.\texttt{infix\_plus}(b)\\
a - b &\triangleq a.\texttt{infix\_minus}(b)\\
a \cup b &\triangleq a.\texttt{infix\_united}(b)\\
-a &\triangleq a.\texttt{prefix\_inversed}\\
\neg a &\triangleq a.\texttt{prefix\_not}\\
a \vee b &\triangleq a.\texttt{infix\_or}\\
\ldots &\quad \ldots
\end{aligned}$$

If the **require** and **ensure** clause is not specified, we will assume the expression `true`. Statements that are included in the **from** clause are treated as if they were in front of the loop.

7 Correctness

7.1 Validity of calls

Any feature call (routine invocation or attribute read) $t.f$ has to make sure that there is indeed a feature f available in the object t. We will assume that the validity of calls can be certified by the use of static typing (although this is not the case for actual Eiffel, cause by unsoundness of its type system).

What remains is to assert that t is not `Void`, a value that is normally allowed for any type, though it does not provide any features. The use of the "non-null types" or the ECMA "attached mechanism" can help solving this problem. If such mechanisms are not available, then the analysis is more complicated by generating proof-obligations of the form $t \neq$ `Void` for any call $t.f$.

7.2 Correctness of Implementation

We regard the implementation of a routine to be correct, if, starting from any state s which yields const(`true`) for the evaluation of the precondition, will also yield true for the postcondition in the resulting state s'. That means, given a routine of class C with one of the following forms:

```
name (a_1,a_2,...) is          name (a_1,a_2,...):C is
   require                        require
      pre                            pre
   modify                         use
      mod                            use
   local                          local
      l_1,l_2,...                    l_1,l_2,...
   do                             do
      S                              S
   ensure                         ensure
      post                           post
   end                            end
```

and a given invariant *INV*, we know that the following formula holds

$$\frac{\forall s \in (\mathit{Heap} \times \mathit{Env}) : \\ \langle S, s \rangle \rightarrow s'}{\begin{array}{c}\mathsf{Expr}[\![\mathit{pre}]\!](s) = \mathsf{const}(\mathtt{true}) \wedge \mathsf{Expr}[\![\mathit{INV}]\!](s) = \mathsf{const}(\mathtt{true}) \Rightarrow \\ \mathsf{OExpr}[\![\mathit{post}]\!](s, s') = \mathsf{const}(\mathtt{true}) \wedge \mathsf{Expr}[\![\mathit{INV}]\!](s') = \mathsf{const}(\mathtt{true})\end{array}}$$

7.3 Observation of Frames

Any implementation of a command has to respect its **modify** frame. Respecting the modify frame means that only objects that are included in the modify frame may have their fields modified. Any implementation of a query has to respect its **use** frame. Respecting the use from means that during the execution of the application, only fields of objects that are included in the use frame are read.

These are the two requirements for the implementation when using frame contracts. If there is no frame contract, the set of all life objects is considered.

7.3.1 Use frames

Going through the grammar, we will define $\mathsf{UseSet}[\![E]\!]$ to be the set of objects whose fields are used by expression E, and $\mathsf{UseSet}[\![S]\!]$ to be the set objects whose fields are accessed through the execution of the statement S.

Expressions:

$$\mathsf{UseSet}[\![c]\!](s) = \emptyset \tag{26}$$

$$\mathsf{UseSet}[\![l]\!](s) = \emptyset \tag{27}$$

$$\mathsf{UseSet}[\![a]\!](s) = \{s_E(\mathtt{Current})\} \tag{28}$$

$$\mathsf{UseSet}[\![E.a]\!](s) = \{\mathsf{Expr}[\![E]\!](s)\} \cup \mathsf{UseSet}[\![E]\!](s) \tag{29}$$

$$\mathsf{UseSet}[\![E1 = E2]\!](s) = \mathsf{UseSet}[\![E1]\!](s) \cup \mathsf{UseSet}[\![E2]\!](s) \tag{30}$$

$$\begin{array}{l}\mathsf{UseSet}[\![T.q(A1, A2, \dots)]\!](s) = \\ \quad \mathsf{UseSet}[\![\mathit{Body}_q]\!](s_H,\ \langle \mathtt{Current} := \mathsf{Expr}[\![T]\!](s), \mathit{arg1} := \mathsf{Expr}[\![A1]\!](s), \dots \rangle) \cup \\ \quad \mathsf{UseSet}[\![T]\!](s) \cup \mathsf{UseSet}[\![A1]\!](s) \cup \mathsf{UseSet}[\![A2]\!](s) \cup \dots\end{array} \tag{31}$$

Instructions:

$$\mathsf{UseSet}[\![]\!](s) = \emptyset \tag{32}$$

$$\mathsf{UseSet}[\![l := E]\!](s) = \mathsf{UseSet}[\![E]\!](s) \tag{33}$$

$$\mathsf{UseSet}[\![a := E]\!](s) = \mathsf{UseSet}[\![E]\!](s) \tag{34}$$

$$\frac{\langle S1, s\rangle \to s'}{\mathsf{UseSet}[\![S_1\ S_2]\!](s) = \mathsf{UseSet}[\![S1]\!](s) \cup \mathsf{UseSet}[\![S2]\!](s')} \tag{35}$$

$$\begin{array}{r}\mathsf{UseSet}[\![\mathbf{if}\ b\ \mathbf{then}\ S_1\ \mathbf{else}\ S_2\ \mathbf{end}]\!](s) = \\ \mathsf{UseSet}[\![b]\!](s) \cup \mathsf{UseSet}[\![S_1]\!](s)\end{array} \text{ iff } \mathsf{Expr}[\![b]\!](s) = const(\mathtt{true}) \tag{36}$$

$$\begin{array}{r}\mathsf{UseSet}[\![\mathbf{if}\ b\ \mathbf{then}\ S_1\ \mathbf{else}\ S_2\ \mathbf{end}]\!](s) = \\ \mathsf{UseSet}[\![b]\!](s) \cup \mathsf{UseSet}[\![S_2]\!](s)\end{array} \text{ iff } \mathsf{Expr}[\![b]\!](s) = const(\mathtt{false}) \tag{37}$$

$$\frac{\langle S, s\rangle \to s'}{\mathsf{UseSet}[\![\mathbf{from\ until}\ b\ \mathbf{loop}\ S\ \mathbf{end}]\!](s) = \mathsf{UseSet}[\![b]\!](s) \cup} \text{ iff } \mathsf{Expr}[\![b]\!](s) = const(\mathtt{true}) \tag{38}$$
$$\mathsf{UseSet}[\![S]\!](s) \cup \mathsf{UseSet}[\![\mathbf{from\ until}\ b\ \mathbf{loop}\ S\ \mathbf{end}]\!](s')$$

$$\mathsf{UseSet}[\![\mathbf{from\ until}\ b\ \mathbf{loop}\ S\ \mathbf{end}]\!](s) = \mathsf{UseSet}[\![b]\!](s) \text{ iff } \mathsf{Expr}[\![b]\!](s) = const(\mathtt{false}) \tag{39}$$

$$\begin{array}{r}\mathsf{UseSet}[\![T.c(A1, A2, \dots)]\!](s) = \mathsf{UseSet}[\![T]\!](s) \cup \mathsf{UseSet}[\![A1]\!](s) \cup \mathsf{UseSet}[\![A2]\!](s) \cup \dots \cup \\ \mathsf{UseSet}[\![Body_c]\!](s_H, \langle \mathtt{Current} := \mathsf{Expr}[\![T]\!](s), \mathbf{arg1} := \mathsf{Expr}[\![A1]\!](s), \dots\rangle)\end{array} \tag{40}$$

$$\begin{array}{r}\mathsf{UseSet}[\![\mathbf{create}\ l.c(A1, A2, \dots)]\!](s) = \mathsf{UseSet}[\![A1]\!](s) \cup \mathsf{UseSet}[\![A2]\!](s) \cup \dots \cup \\ \mathsf{UseSet}[\![Body_c]\!](s_H\langle C\rangle, \langle \mathtt{Current} := \mathsf{new}(s_H, C), \mathbf{arg1} := \mathsf{Expr}[\![a1]\!](s), \dots\rangle)\end{array} \tag{41}$$

7.3.2 Correctness of use frames

We will assume that a query has a correct implementation with respect to its use frame, if the following property holds:

$$\begin{array}{c}\forall s \in (Heap \times Env) : \\ \dfrac{\langle S, s\rangle \to s'}{\begin{array}{r}\mathsf{Expr}[\![pre]\!](s) = \mathsf{const}(\mathtt{true}) \wedge \mathsf{Expr}[\![INV]\!](s) = \mathsf{const}(\mathtt{true}) \Rightarrow \\ (\mathsf{UseSet}[\![S]\!](s) \cap \mathsf{Used}(s_H)) \subseteq \mathsf{Expr}[\![use]\!](s)\end{array}}\end{array} \tag{UseOK}$$

7.3.3 Modify frames

We will define $\mathsf{ModifySet}[\![E]\!]$ to be the set of objects whose fields are modified through the execution of the statement S. In this definition, we will disregard expression, as we know that these are side-effect free.

$$\mathsf{ModifySet}[\![]\!](s) = \emptyset \tag{42}$$

$$\mathsf{ModifySet}[\![l := E]\!](s) = \emptyset \tag{43}$$

$$\mathsf{ModifySet}[\![a := E]\!](s) = \{s_E(\texttt{Current})\} \tag{44}$$

$$\frac{\langle S1, s\rangle \to s'}{\mathsf{ModifySet}[\![S_1\ S_2]\!](s) = \mathsf{ModifySet}[\![S1]\!](s) \cup \mathsf{ModifySet}[\![S2]\!](s')} \tag{45}$$

$$\begin{array}{r}\mathsf{ModifySet}[\![\mathbf{if}\ b\ \mathbf{then}\ S_1\ \mathbf{else}\ S_2\ \mathbf{end}]\!](s) = \\ \mathsf{ModifySet}[\![S_1]\!](s)\end{array} \text{ iff } \mathsf{Expr}[\![b]\!](s) = const(\texttt{true}) \tag{46}$$

$$\begin{array}{r}\mathsf{ModifySet}[\![\mathbf{if}\ b\ \mathbf{then}\ S_1\ \mathbf{else}\ S_2\ \mathbf{end}]\!](s) = \\ \mathsf{ModifySet}[\![S_2]\!](s)\end{array} \text{ iff } \mathsf{Expr}[\![b]\!](s) = const(\texttt{false}) \tag{47}$$

$$\frac{\langle S, s\rangle \to s'}{\begin{array}{l}\mathsf{ModifySet}[\![\mathbf{from\ until}\ b\ \mathbf{loop}\ S\ \mathbf{end}]\!](s) = \\ \mathsf{ModifySet}[\![S]\!](s) \cup \mathsf{ModifySet}[\![\mathbf{from\ until}\ b\ \mathbf{loop}\ S\ \mathbf{end}]\!](s')\end{array}} \text{ iff } \mathsf{Expr}[\![b]\!](s) = const(\texttt{true}) \tag{48}$$

$$\mathsf{ModifySet}[\![\mathbf{from\ until}\ b\ \mathbf{loop}\ S\ \mathbf{end}]\!](s) = \emptyset \text{ iff } \mathsf{Expr}[\![b]\!](s) = const(\texttt{false}) \tag{49}$$

$$\begin{array}{l}\mathsf{ModifySet}[\![T.c(A1, A2, \dots)]\!](s) = \\ \quad \mathsf{ModifySet}[\![Body_c]\!](s_H, \langle \texttt{Current} := \mathsf{Expr}[\![T]\!](s), \mathbf{arg1} := \mathsf{Expr}[\![A1]\!](s), \dots\rangle)\end{array} \tag{50}$$

$$\begin{array}{l}\mathsf{ModifySet}[\![\mathbf{create}\ l.c(A1, A2, \dots)]\!](s) = \\ \quad \mathsf{ModifySet}[\![Body_c]\!](s_H\langle C\rangle, \langle \texttt{Current} := \mathsf{new}(s_H, C), \mathbf{arg1} := \mathsf{Expr}[\![a1]\!](s), \dots\rangle)\end{array} \tag{51}$$

7.3.4 Correctness of modify frames

We will assume that a command has a correct implementation with respect to its use modify, if the following property holds:

$$\begin{array}{c}\forall s \in (Heap \times Env): \\ \dfrac{\langle S, s\rangle \to s'}{\begin{array}{l}\mathsf{Expr}[\![pre]\!](s) = \mathsf{const}(\texttt{true}) \wedge \mathsf{Expr}[\![INV]\!](s) = \mathsf{const}(\texttt{true}) \Rightarrow \\ \quad (\mathsf{ModifySet}[\![S]\!](s) \cap \mathsf{Used}(s_H)) \subseteq \mathsf{Expr}[\![mod]\!](s)\end{array}}\end{array} \tag{ModOK}$$

8 Proofs

In this section we will proof some interesting properties of the frames introduced in the operational semantics.

8.1 Fields outside of modify frame are immutable

Assuming a correct command c with the body $Body_c$ and the specified modification frame mod_c, we want to prove that all fields of objects outside of the frame keep their values.

$$\begin{array}{c}\forall s \in Heap \times Env, o \in Obj, a \in AttributeID, c \in ClassID: \\ \langle Body_c, s\rangle \to s' \wedge \mathsf{Expr}[\![pre]\!](s) = \mathsf{const}(\texttt{true}) \wedge \mathsf{Expr}[\![INV]\!](s) = \mathsf{const}(\texttt{true}) \\ \wedge alloc(o, s_H) \wedge o \notin \mathsf{Expr}[\![mod_c]\!](s) \Rightarrow s_H(o.a) = s'_H(o.a)\end{array}$$

Without loss of generality we will assume that s has been picked to satisfy the precondition and invariant. From the definition of correctness ModOK, we know that the following property is strictly stronger (dropping the quantifiers for brevity).

$$\langle S, s\rangle \to s' \wedge o \in \mathsf{Used}(s_H) - \mathsf{ModifySet}[\![S]\!](s) \Rightarrow s_H(o.a) = s'_H(o.a)$$

We prove by structurual induction over the derivation tree for any $\langle S, s\rangle$:

$$\langle , s\rangle \to s' \wedge o \in \mathsf{Used}(s_H) - \mathsf{ModifySet}[\![]\!](s) \Rightarrow s_H(o.a) = s'_H(o.a)$$

$$\mathit{(16)} \Rrightarrow s_H(o.a) = s_H(o.a) \ \Box$$

assignment to local variable:

$$\langle l := E, s\rangle \to s' \wedge o \in \mathsf{Used}(s_H) - \mathsf{ModifySet}[\![l := E]\!](s) \Rightarrow s_H(o.a) = s'_H(o.a)$$

$$\mathit{(17)} \Rrightarrow s_H(o.a) = s_H(o.a) \ \Box$$

assignment to attribute:

$$\langle a := E, s\rangle \to s' \wedge o \in \mathsf{Used}(s_H) - \mathsf{ModifySet}[\![a := E]\!](s) \Rightarrow s_H(o.a) = s'_H(o.a)$$

$$\mathit{(18)} \Rrightarrow o \in \mathsf{Used}(s_H) - \mathsf{ModifySet}[\![a := E]\!](s) \Rightarrow s_H(o.a) = s_H\langle s_E(\mathtt{Current}).a := \mathsf{Expr}[\![E]\!](s)\rangle(o.a)$$

$$\mathit{(44)} \Rrightarrow o \neq s_E(\mathit{Current}) \Rightarrow s_H(o.a) = s_H\langle s_E(\mathtt{Current}).a := \mathsf{Expr}[\![E]\!](s)\rangle(o.a)$$

$$\mathit{(H1)} \Rrightarrow y = y \ \Box$$

sequential composition:

$$\langle S_1\, S_2, s\rangle \to s' \wedge o \in \mathsf{Used}(s_H) - \mathsf{ModifySet}[\![S_1\, S_2]\!](s) \Rightarrow s_H(o.a) = s'_H(o.a)$$

$$\mathit{(19)} \Rrightarrow \langle S_1, s\rangle \to s'' \wedge \langle S_2, s''\rangle \to s' \wedge o \in \mathsf{Used}(s_H) - \mathsf{ModifySet}[\![S_1\, S_2]\!](s) \Rightarrow s_H(o.a) = s'_H(o.a)$$

$$\mathit{(45)} \Rrightarrow \langle S_1, s\rangle \to s'' \wedge \langle S_2, s''\rangle \to s' \wedge o \in \mathsf{Used}(s_H) - (\mathsf{ModifySet}[\![S_1]\!](s) \cup \mathsf{ModifySet}[\![S_2]\!](s))$$
$$\Rightarrow s_H(o.a) = s'_H(o.a)$$
$$\Rrightarrow \langle S_1, s\rangle \to s'' \wedge \langle S_2, s''\rangle \to s' \wedge o \in \mathsf{Used}(s_H) - \mathsf{ModifySet}[\![S_1]\!](s) \wedge$$
$$o \in \mathsf{Used}(s_H) - \mathsf{ModifySet}[\![S_2]\!](s) \Rightarrow s_H(o.a) = s'_H(o.a)$$

$$\mathit{(IH)} \Rrightarrow s_H(o.a) = s''_H(o.a) \wedge s''_H(o.a) = s'_H(o.a) \Rightarrow s_H(o.a) = s'_H(o.a) \ \Box$$

if-then-else (only the true case, false is similar):

$$\langle \mathbf{if}\, b\, \mathbf{then}\, S_1\, \mathbf{else}\, S_2\, \mathbf{end}, s\rangle \to s' \wedge \mathsf{Expr}[\![b]\!](s) = \mathsf{const}(\mathtt{true}) \wedge$$
$$o \in \mathsf{Used}(s_H) - \mathsf{ModifySet}[\![\mathbf{if}\, b\, \mathbf{then}\, S_1\, \mathbf{else}\, S_2\, \mathbf{end}]\!](s) \Rightarrow s_H(o.a) = s'_H(o.a)$$

$$\mathit{(20)} \Rrightarrow \langle S_1, s\rangle \to s' \wedge \mathsf{Expr}[\![b]\!](s) = \mathsf{const}(\mathtt{true}) \wedge$$
$$o \in \mathsf{Used}(s_H) - \mathsf{ModifySet}[\![\mathbf{if}\, b\, \mathbf{then}\, S_1\, \mathbf{else}\, S_2\, \mathbf{end}]\!](s) \Rightarrow s_H(o.a) = s'_H(o.a)$$

$$\mathit{(46)} \Rrightarrow \langle S_1, s\rangle \to s' \wedge o \in \mathsf{Used}(s_H) - \mathsf{ModifySet}[\![S_1]\!](s) \Rightarrow s_H(o.a) = s'_H(o.a)$$

$$\mathit{(IH)} \Rrightarrow s_H(o.a) = s'_H(o.a) \Rightarrow s_H(o.a) = s'_H(o.a) \ \Box$$

loop continuation:

$$\langle \mathbf{from}\, \mathbf{until}\, b\, \mathbf{loop}\, S\, \mathbf{end}, s\rangle \to s' \wedge \mathsf{Expr}[\![b]\!](s) = \mathsf{const}(\mathtt{false}) \wedge$$
$$o \in \mathsf{Used}(s_H) - \mathsf{ModifySet}[\![\mathbf{from}\, \mathbf{until}\, b\, \mathbf{loop}\, S\, \mathbf{end}]\!](s) \Rightarrow s_H(o.a) = s'_H(o.a)$$

$$\mathit{(22)} \Rrightarrow \langle S, s\rangle \to s'' \wedge \langle \mathbf{from}\, \mathbf{until}\, b\, \mathbf{loop}\, S\, \mathbf{end}, s''\rangle \to s' \wedge \mathsf{Expr}[\![b]\!](s) = \mathsf{const}(\mathtt{true}) \wedge$$
$$o \in \mathsf{Used}(s_H) - \mathsf{ModifySet}[\![\mathbf{from}\, \mathbf{until}\, b\, \mathbf{loop}\, S\, \mathbf{end}]\!](s) \Rightarrow s_H(o.a) = s'_H(o.a)$$

$$\mathit{(48)} \Rrightarrow \langle S, s\rangle \to s'' \wedge \langle \mathbf{from}\, \mathbf{until}\, b\, \mathbf{loop}\, S\, \mathbf{end}, s''\rangle \to s' \wedge \mathsf{Expr}[\![b]\!](s) = \mathsf{const}(\mathtt{true}) \wedge$$
$$o \in \mathsf{Used}(s_H) - (\mathsf{ModifySet}[\![S]\!](s) \cup \mathsf{ModifySet}[\![\mathbf{from}\, \mathbf{until}\, b\, \mathbf{loop}\, S\, \mathbf{end}]\!](s'')) \Rightarrow s_H(o.a) = s'_H(o.a)$$

$$\mathit{(IH)} \Rrightarrow \langle S, s\rangle \to s'' \wedge s''_H(o.a) = s'_H(o.a) \wedge \mathsf{Expr}[\![b]\!](s) = \mathsf{const}(\mathtt{true}) \wedge$$
$$o \in \mathsf{Used}(s_H) - \mathsf{ModifySet}[\![S]\!](s) \Rightarrow s_H(o.a) = s'_H(o.a)$$

$$\mathit{(IH)} \Rrightarrow s_H(o.a) = s''_H(o.a) \wedge s''_H(o.a) = s'_H(o.a) \Rightarrow s_H(o.a) = s'_H(o.a) \ \Box$$

loop termination:

$$\begin{aligned}
&\langle \mathbf{from}\,\mathbf{until}\,b\,\mathbf{loop}\,S\,\mathbf{end}, s\rangle \rightarrow s' \wedge \mathsf{Expr}[\![b]\!](s) = \mathsf{const}(\mathtt{false}) \wedge \\
&o \in \mathsf{Used}(s_H) - \mathsf{ModifySet}[\![\mathbf{from}\,\mathbf{until}\,b\,\mathbf{loop}\,S\,\mathbf{end}]\!](s) \Rightarrow s_H(o.a) = s'_H(o.a) \\
\mathit{(23)} \Rrightarrow\ &s_H(o.a) = s_H(o.a)\ \square
\end{aligned}$$

feature invocation:

$$\begin{aligned}
&\langle T.c(A1, A2, \dots), s\rangle \rightarrow s' \wedge o \in \mathsf{Used}(s_H) - \mathsf{ModifySet}[\![T.c(A1, A2, \dots)]\!](s) \Rightarrow s_H(o.a) = s'_H(o.a) \\
\mathit{(24)} \Rrightarrow\ &\langle \mathit{Body}_c, (s_H, \langle \mathtt{Current} := \mathsf{Expr}[\![T]\!](s), \mathbf{arg1} := \mathsf{Expr}[\![A1]\!](s), \dots\rangle \rightarrow (s'_H, s''_E) \wedge \\
&o \in \mathsf{Used}(s_H) - \mathsf{ModifySet}[\![T.c(A1, A2, \dots)]\!](s) \Rightarrow s_H(o.a) = s'_H(o.a) \\
\mathit{(50)} \Rrightarrow\ &\langle \mathit{Body}_c, (s_H, \langle \mathtt{Current} := \mathsf{Expr}[\![T]\!](s), \mathbf{arg1} := \mathsf{Expr}[\![A1]\!](s), \dots)\rangle \rightarrow (s'_H, s''_E) \wedge \\
&o \in \mathsf{Used}(s_H) - \mathsf{ModifySet}[\![\mathit{Body}_c]\!](s_H, \langle \mathtt{Current} := \mathsf{Expr}[\![T]\!](s), \mathbf{arg1} := \mathsf{Expr}[\![A1]\!](s), \dots) \\
&\ \Rightarrow s_H(o.a) = s'_H(o.a) \\
\mathit{(IH)} \Rrightarrow\ &s_H(o.a) = s'_H(o.a) \Rightarrow s_H(o.a) = s'_H(o.a)\ \square
\end{aligned}$$

object creation:

$$\begin{aligned}
&\langle \mathbf{create}\,l.c(A1, A2, \dots), s\rangle \rightarrow s' \wedge o \in \mathsf{Used}(s_H) - \mathsf{ModifySet}[\![\mathbf{create}\,l.c(A1, A2, \dots)]\!](s) \\
&\ \Rightarrow s_H(o.a) = s'_H(o.a) \\
\mathit{(25)} \Rrightarrow\ &\langle \mathit{Body}_c, (s_H\langle C\rangle, \langle \mathtt{Current} := \mathsf{new}(s_H, C), \mathbf{arg1} := \mathsf{Expr}[\![A1]\!](s), \dots)\rangle \rightarrow (s'_H, s''_E) \wedge \\
&o \in \mathsf{Used}(s_H) - \mathsf{ModifySet}[\![\mathbf{create}\,l.c(A1, A2, \dots)]\!](s) \Rightarrow s_H(o.a) = s'_H(o.a) \\
\mathit{(51)} \Rrightarrow\ &\langle \mathit{Body}_c, (s_H\langle C\rangle, \langle \mathtt{Current} := \mathsf{new}(s_H, C), \mathbf{arg1} := \mathsf{Expr}[\![A1]\!](s), \dots)\rangle \rightarrow (s'_H, s''_E) \wedge \\
&o \in \mathsf{Used}(s_H) - \mathsf{ModifySet}[\![\mathit{Body}_c]\!](s_H, \langle \mathtt{Current} := \mathsf{new}(s_H, C), \mathbf{arg1} := \mathsf{Expr}[\![A1]\!](s), \dots) \\
&\ \Rightarrow s_H(o.a) = s'_H(o.a) \\
\mathit{(IH)} \Rrightarrow\ &o \in \mathsf{Used}(s_H) \wedge (s_H\langle C\rangle(o.a) = s'_H(o.a)) \Rightarrow (s_H(o.a) = s'_H(o.a)) \\
\mathit{(H3)} \Rrightarrow\ &(s_H(o.a) = s'_H(o.a)) \Rightarrow (s_H(o.a) = s'_H(o.a))\ \square
\end{aligned}$$

8.2 Queries depend only on their use sets

We prove by induction that if a query yields a certain result, this result will not change if any field outside of its use set is changed (IH1):

$$\begin{aligned}
&\forall \mathit{res}, o, x : \mathit{Obj}, b : \mathit{AttributeID}, h : \mathit{Heap}, e : \mathit{Env} : \\
&\mathsf{Expr}[\![E]\!](h, e) = \mathit{res} \wedge o \notin \mathsf{UseSet}[\![E]\!](h, e) \Rightarrow \mathsf{Expr}[\![E]\!](h\langle o.b := x\rangle, e) = \mathit{res}
\end{aligned}$$

In parallel, we have to prove that the following property over all instructions S holds (IH2):

$$\begin{aligned}
&\forall \mathit{res}, o, x : \mathit{Obj}, b : \mathit{AttributeID}, h : \mathit{Heap}, e : \mathit{Env}, \exists y : \mathit{Obj} : \\
&\langle S, (h, e)\rangle \rightarrow (h', e') \wedge o \notin \mathsf{UseSet}[\![S]\!](h, e) \Rightarrow \langle S, (h\langle o.b := x\rangle, e)\rangle \rightarrow (h'\langle o.b := y\rangle, e')
\end{aligned}$$

constants

$$\begin{aligned}
&\mathsf{Expr}[\![c]\!](h, e) = \mathit{res} \wedge o \notin \mathsf{UseSet}[\![c]\!](h, e) \Rightarrow \mathsf{Expr}[\![c]\!](h\langle o.b := x\rangle, e) = \mathit{res} \\
\mathit{(3)} \Rrightarrow\ &\mathsf{const}(c) = \mathit{res} \wedge o \notin \mathsf{UseSet}[\![c]\!](h, e) \Rightarrow \mathsf{const}(c) = \mathit{res}\ \square
\end{aligned}$$

local variables

$$\begin{aligned}
& \mathsf{Expr}\llbracket l\rrbracket(h,e) = res \wedge o \notin \mathsf{UseSet}\llbracket l\rrbracket(h,e) \Rightarrow \mathsf{Expr}\llbracket l\rrbracket(h\langle o.b := x\rangle, e) = res \\
\textit{(4)} \; \Rrightarrow\; & e(l) = res \wedge o \notin \mathsf{UseSet}\llbracket c\rrbracket(h,e) \Rightarrow e(l) = res \;\Box
\end{aligned}$$

attribute read

$$\begin{aligned}
& \mathsf{Expr}\llbracket a\rrbracket(h,e) = res \wedge o \notin \mathsf{UseSet}\llbracket a\rrbracket(h,e) \Rightarrow \mathsf{Expr}\llbracket a\rrbracket(h\langle o.b := x\rangle, e) = res \\
\textit{(5)} \; \Rrightarrow\; & h(e(\texttt{Current}).a) = res \wedge o \notin \mathsf{UseSet}\llbracket a\rrbracket(h,e) \Rightarrow h\langle o.b := x\rangle(e(\texttt{Current}).a) = res \\
\textit{(28)} \; \Rrightarrow\; & h(e(\texttt{Current}).a) = res \wedge o \neq \texttt{Current} \Rightarrow h\langle o.b := x\rangle(e(\texttt{Current}).a) = res \\
\textit{(H1)} \; \Rrightarrow\; & h(e(\texttt{Current}).a) = res \Rightarrow h(e(\texttt{Current}).a) = res \;\Box
\end{aligned}$$

remote attribute read

$$\begin{aligned}
& \mathsf{Expr}\llbracket E.a\rrbracket(h,e) = res \wedge o \notin \mathsf{UseSet}\llbracket E.a\rrbracket(h,e) \Rightarrow \mathsf{Expr}\llbracket E.a\rrbracket(h\langle o.b := x\rangle, e) = res \\
\textit{(6)} \; \Rrightarrow\; & h(\mathsf{Expr}\llbracket E\rrbracket(h,e).a) = res \wedge o \notin \mathsf{UseSet}\llbracket E.a\rrbracket(h,e) \Rightarrow h\langle o.b := x\rangle(\mathsf{Expr}\llbracket E\rrbracket(h\langle o.b := x\rangle, e).a) = res \\
\textit{(29)} \; \Rrightarrow\; & h(\mathsf{Expr}\llbracket E\rrbracket(h,e).a) = res \wedge o \neq \mathsf{Expr}\llbracket E\rrbracket(h,e) \wedge o \notin \mathsf{UseSet}\llbracket E\rrbracket(h,e) \Rightarrow \\
& h\langle o.b := x\rangle(\mathsf{Expr}\llbracket E\rrbracket(h\langle o.b := x\rangle, e).a) = res \\
\textit{(IH1)} \; \Rrightarrow\; & h(\mathsf{Expr}\llbracket E\rrbracket(h,e).a) = res \wedge o \neq \mathsf{Expr}\llbracket E\rrbracket(h,e) \Rightarrow h\langle o.b := x\rangle(\mathsf{Expr}\llbracket E\rrbracket(h,e).a) = res \\
\textit{(H1)} \; \Rrightarrow\; & h(\mathsf{Expr}\llbracket E\rrbracket(h,e).a) = res \Rightarrow h(\mathsf{Expr}\llbracket E\rrbracket(h,e).a) = res \;\Box
\end{aligned}$$

reference comparison

$$\begin{aligned}
& \mathsf{Expr}\llbracket E1 = E2\rrbracket(h,e) = res \wedge o \notin \mathsf{UseSet}\llbracket E1 = E2\rrbracket(h,e) \Rightarrow \mathsf{Expr}\llbracket E1 = E2\rrbracket(h\langle o.b := x\rangle, e) = res \\
\textit{(7)} \; \Rrightarrow\; & (\mathsf{Expr}\llbracket E1\rrbracket(h,e) = \mathsf{Expr}\llbracket E2\rrbracket(h,e)) = res \wedge o \notin \mathsf{UseSet}\llbracket E1 = E2\rrbracket(h,e) \Rightarrow \\
& (\mathsf{Expr}\llbracket E1\rrbracket(h\langle o.b := x\rangle, e) = \mathsf{Expr}\llbracket E1\rrbracket(h\langle o.b := x\rangle, e)) = res \\
\textit{(30)} \; \Rrightarrow\; & (\mathsf{Expr}\llbracket E1\rrbracket(h,e) = \mathsf{Expr}\llbracket E2\rrbracket(h,e)) = res \wedge o \notin \mathsf{UseSet}\llbracket E1\rrbracket(h,e) \wedge o \notin E2(h,e) \Rightarrow \\
& (\mathsf{Expr}\llbracket E1\rrbracket(h\langle o.b := x\rangle, e) = \mathsf{Expr}\llbracket E1\rrbracket(h\langle o.b := x\rangle, e)) = res \\
\textit{(IH1)} \; \Rrightarrow\; & (\mathsf{Expr}\llbracket E1\rrbracket(h,e) = \mathsf{Expr}\llbracket E2\rrbracket(h,e)) \Rightarrow (\mathsf{Expr}\llbracket E1\rrbracket(h,e) = \mathsf{Expr}\llbracket E1\rrbracket(h,e)) = res \;\Box
\end{aligned}$$

query call

$$\begin{aligned}
& \mathsf{Expr}\llbracket T.q(A1, \dots)\rrbracket(h,e) = res \wedge o \notin \mathsf{UseSet}\llbracket T.q(A1, \dots)\rrbracket(h,e) \Rightarrow \\
& \mathsf{Expr}\llbracket T.q(A1, \dots)\rrbracket(h\langle o.b := x\rangle, e) = res \\
\textit{(8)} \; \Rrightarrow\; & \langle Body_q, (h, \langle \texttt{Current} := \mathsf{Expr}\llbracket T\rrbracket(h,e), arg1 := \mathsf{Expr}\llbracket A1\rrbracket(h,e)\rangle)\rangle \to (h', e') \wedge \\
& \langle Body_q, (h\langle o.b := x\rangle, \\
& \qquad \langle \texttt{Current} := \mathsf{Expr}\llbracket T\rrbracket(h\langle o.b := x\rangle, e), arg1 := \mathsf{Expr}\llbracket A1\rrbracket(h\langle o.b := x\rangle, e), \dots\rangle)\rangle \to (h'', e'') \wedge \\
& e'(\texttt{Result}) = res \wedge o \notin \mathsf{UseSet}\llbracket T.q(A1, \dots)\rrbracket(h,e) \Rightarrow e''(\texttt{Result}) = res \\
\textit{(31)} \; \Rrightarrow\; & \langle Body_q, (h, \langle \texttt{Current} := \mathsf{Expr}\llbracket T\rrbracket(h,e), arg1 := \mathsf{Expr}\llbracket A1\rrbracket(h,e)\rangle)\rangle \to (h', e') \wedge \\
& \langle Body_q, (h\langle o.b := x\rangle, \\
& \qquad \langle \texttt{Current} := \mathsf{Expr}\llbracket T\rrbracket(h\langle o.b := x\rangle, e), arg1 := \mathsf{Expr}\llbracket A1\rrbracket(h\langle o.b := x\rangle, e), \dots\rangle)\rangle \to (h'', e'') \wedge \\
& e'(\texttt{Result}) = res \wedge o \notin \mathsf{UseSet}\llbracket T\rrbracket(h,e) \wedge o \notin \mathsf{UseSet}\llbracket A1\rrbracket(h,e) \wedge \dots \wedge \\
& o \notin \mathsf{UseSet}\llbracket Body_q\rrbracket(h, \langle \texttt{Current} := \mathsf{Expr}\llbracket T\rrbracket(h,e), arg1 := \mathsf{Expr}\llbracket A1\rrbracket(h,e)\rangle) \Rightarrow e''(\texttt{Result}) = res \\
\textit{(IH1)} \; \Rrightarrow\; & \langle Body_q, (h, \langle \texttt{Current} := \mathsf{Expr}\llbracket T\rrbracket(h,e), arg1 := \mathsf{Expr}\llbracket A1\rrbracket(h,e)\rangle)\rangle \to (h', e') \wedge \\
& \langle Body_q, (h\langle o.b := x\rangle, \langle \texttt{Current} := \mathsf{Expr}\llbracket T\rrbracket(h,e), arg1 := \mathsf{Expr}\llbracket A1\rrbracket(h,e), \dots\rangle)\rangle \to (h'', e'') \wedge \\
& e'(\texttt{Result}) = res \wedge o \notin \mathsf{UseSet}\llbracket Body_q\rrbracket(h, \langle \texttt{Current} := \mathsf{Expr}\llbracket T\rrbracket(h,e), arg1 := \mathsf{Expr}\llbracket A1\rrbracket(h,e)\rangle) \Rightarrow \\
& e''(\texttt{Result}) = res \\
\textit{(IH2)} \; \Rrightarrow\; & e'(\texttt{Result}) = res \Rightarrow e'(\texttt{Result}) = res \;\Box
\end{aligned}$$

empty instruction

$$\begin{aligned}
&\langle , (h,e)\rangle \to (h',e') \land o \notin \mathsf{UseSet}\llbracket\rrbracket(h,e) \Rightarrow \langle , (h\langle o.b := x\rangle, e)\rangle \to (h'\langle o.b := y\rangle, e')\\
\mathit{(16)}\ \Rrightarrow\ &(h,e) = (h',e') \land o \notin \mathsf{UseSet}\llbracket\rrbracket(h,e) \Rightarrow (h\langle o.b := x\rangle, e) = (h'\langle o.b := y\rangle, e')\\
\Rrightarrow\ &(h\langle o.b := x\rangle, e) = (h\langle o.b := x\rangle, e)\ (\text{with}\, x = y)\ \Box
\end{aligned}$$

local assignment

$$\begin{aligned}
&\langle l := E, (h,e)\rangle \to (h',e') \land o \notin \mathsf{UseSet}\llbracket l := E\rrbracket(h,e) \Rightarrow \langle l := E, (h\langle o.b := x\rangle, e)\rangle \to (h'\langle o.b := y\rangle, e')\\
\Rrightarrow\ &\langle l := E, (h,e)\rangle \to (h',e') \land o \notin \mathsf{UseSet}\llbracket l := E\rrbracket(h,e) \Rightarrow \langle l := E, (h\langle o.b := x\rangle, e)\rangle \to (h'\langle o.b := x\rangle, e')\\
&(\text{with}\, x = y)\\
\mathit{(17)}\ \Rrightarrow\ &(h',e') = (h, e\langle l := \mathsf{Expr}\llbracket E\rrbracket(h,e)\rangle) \land o \notin \mathsf{UseSet}\llbracket l := E\rrbracket(h,e) \Rightarrow\\
&(h'\langle o.b := x\rangle, e') = (h\langle o.b := x\rangle, e\langle l := \mathsf{Expr}\llbracket E\rrbracket(h\langle o.b := x\rangle, e)\rangle)\\
\mathit{(33)}\ \Rrightarrow\ &(h',e') = (h, e\langle l := \mathsf{Expr}\llbracket E\rrbracket(h,e)\rangle) \land o \notin \mathsf{UseSet}\llbracket E\rrbracket(h,e) \Rightarrow\\
&(h'\langle o.b := x\rangle, e') = (h\langle o.b := x\rangle, e\langle l := \mathsf{Expr}\llbracket E\rrbracket(h\langle o.b := x\rangle, e)\rangle)\\
\mathit{(IH1)}\ \Rrightarrow\ &(h',e') = (h, e\langle l := \mathsf{Expr}\llbracket E\rrbracket(h,e)\rangle) \land o \notin \mathsf{UseSet}\llbracket E\rrbracket(h,e) \Rightarrow\\
&(h'\langle o.b := x\rangle, e') = (h\langle o.b := x\rangle, e\langle l := \mathsf{Expr}\llbracket E\rrbracket(h,e)\rangle)\\
\Rrightarrow\ &(h\langle o.b := x\rangle, e\langle l := \mathsf{Expr}\llbracket E\rrbracket(h,e)\rangle) = (h\langle o.b := x\rangle, e\langle l := \mathsf{Expr}\llbracket E\rrbracket(h,e)\rangle)\ \Box
\end{aligned}$$

attribute assignment

$$\begin{aligned}
&\langle a := E, (h,e)\rangle \to (h',e') \land o \notin \mathsf{UseSet}\llbracket a := E\rrbracket(h,e) \Rightarrow \langle a := E, (h\langle o.b := x\rangle, e)\rangle \to (h'\langle o.b := y\rangle, e')\\
\mathit{(18)}\ \Rrightarrow\ &(h',e') = (h\langle \mathsf{Expr}\llbracket\texttt{Current}\rrbracket(h,e).a := \mathsf{Expr}\llbracket E\rrbracket(h,e)\rangle, e) \land o \notin \mathsf{UseSet}\llbracket a := E\rrbracket(h,e) \Rightarrow\\
&(h'\langle o.b := y\rangle, e') = (h\langle o.b := x\rangle\langle \mathsf{Expr}\llbracket\texttt{Current}\rrbracket(h,e).a := \mathsf{Expr}\llbracket E\rrbracket(h,e)\rangle, e)\\
\mathit{(34)}\ \Rrightarrow\ &(h',e') = (h\langle \mathsf{Expr}\llbracket\texttt{Current}\rrbracket(h,e).a := \mathsf{Expr}\llbracket E\rrbracket(h,e)\rangle, e) \land o \notin \mathsf{UseSet}\llbracket E\rrbracket(h,e) \Rightarrow\\
&(h'\langle o.b := y\rangle, e') = (h\langle o.b := x\rangle\langle \mathsf{Expr}\llbracket\texttt{Current}\rrbracket(h\langle o.b := x\rangle, e).a := \mathsf{Expr}\llbracket E\rrbracket(h\langle o.b := x\rangle, e)\rangle, e)\\
\mathit{(IH1)}\ \Rrightarrow\ &(h',e') = (h\langle \mathsf{Expr}\llbracket\texttt{Current}\rrbracket(h,e).a := \mathsf{Expr}\llbracket E\rrbracket(h,e)\rangle, e) \Rightarrow\\
&(h'\langle o.b := y\rangle, e') = (h\langle o.b := x\rangle\langle \mathsf{Expr}\llbracket\texttt{Current}\rrbracket(ho.b := x, e).a := \mathsf{Expr}\llbracket E\rrbracket(h,e)\rangle, e)\\
\mathit{(3)}\ \Rrightarrow\ &(h',e') = (h\langle \mathsf{const}(\texttt{Current})(e).a := \mathsf{Expr}\llbracket E\rrbracket(h,e)\rangle, e) \Rightarrow\\
&(h'\langle o.b := y\rangle, e') = (h\langle o.b := x\rangle\langle \mathsf{const}(\texttt{Current})(e).a := \mathsf{Expr}\llbracket E\rrbracket(h,e)\rangle, e)\\
\Rrightarrow\ &(h\langle \mathsf{const}(\texttt{Current})(e).a := \mathsf{Expr}\llbracket E\rrbracket(h,e)\rangle\langle o.b := y\rangle, e) =\\
&(h\langle o.b := x\rangle\langle \mathsf{const}(\texttt{Current})(e).a := \mathsf{Expr}\llbracket E\rrbracket(h,e)\rangle, e)\\
\Rrightarrow\ &\mathit{Case} : o = \mathsf{const}(\texttt{Current})(e) \land a = b) : (\text{let}\, y = \mathsf{Expr}\llbracket E\rrbracket(h,e))\\
&(h\langle o.b := y\rangle, e) = (h\langle o.b := y\rangle, e)\ \Box\\
\Rrightarrow\ &\mathit{Case} : o \neq \mathsf{const}(\texttt{Current})(e) \lor a \neq b) : (\text{let}\, y = x)\\
&(h\langle \mathsf{const}(\texttt{Current})(e).a := \mathsf{Expr}\llbracket E\rrbracket(h,e)\rangle\langle o.b := x\rangle, e) =\\
&(h\langle o.b := x\rangle\langle \mathsf{const}(\texttt{Current})(e).a := \mathsf{Expr}\llbracket E\rrbracket(h,e)\rangle, e)\ \Box
\end{aligned}$$

sequential composition

$$
\begin{aligned}
& \langle S_1S_2, (h,e)\rangle \to (h',e') \wedge o \notin \mathsf{UseSet}[\![S_1S_2]\!](h,e) \Rightarrow \langle S_1S_2, (h\langle o.b := x\rangle, e)\rangle \to (h'\langle o.b := y\rangle, e') \\
\mathit{(19)} \; \Rrightarrow \; & \exists h^*, e^* : \langle S_1\rangle(h,e) \to (h^*,e^*) \wedge \langle S_2, (h^*,e^*)\rangle \to (h',e') \wedge o \notin \mathsf{UseSet}[\![S_1S_2]\!](h,e) \Rightarrow \\
& \exists h^{**}, e^{**} : \langle S_1, (h\langle o.b := x\rangle, e)\rangle \to (h^{**},e^{**}) \wedge \langle S_2, (h^{**},e^{**})\rangle \to (h'\langle o.b := y\rangle, e') \\
\mathit{(35)} \; \Rrightarrow \; & \exists h^*, e^* : \langle S_1\rangle(h,e) \to (h^*,e^*) \wedge \langle S_2, (h^*,e^*)\rangle \to (h',e') \wedge \\
& o \notin \mathsf{UseSet}[\![S_1]\!](h,e) \wedge o \notin \mathsf{UseSet}[\![S_2]\!](h^*,e^*) \Rightarrow \\
& \exists h^{**}, e^{**} : \langle S_1, (h\langle o.b := x\rangle, e)\rangle \to (h^{**},e^{**}) \wedge \langle S_2, (h^{**},e^{**})\rangle \to (h'\langle o.b := y\rangle, e') \\
\mathit{(IH2)} \; \Rrightarrow \; & \exists h^*, e^*, z : \langle S_1, (h\langle o.b := x\rangle, e)\rangle \to (h^*\langle o.b := z\rangle, e^*) \wedge \langle S_2, (h^*\langle o.b := z\rangle, e^*)\rangle \to (h'\langle o.b := z\rangle, e') \Rightarrow \\
& \exists h^{**}, e^{**} : \langle S_1, (h\langle o.b := x\rangle, e)\rangle \to (h^{**},e^{*}) \wedge \langle S_2, (h^{**},e^{*})\rangle \to (h'\langle o.b := y\rangle, e') \; \square
\end{aligned}
$$

if-then-else (then branch only, else branch similiar proof)

$$
\begin{aligned}
& \langle \mathbf{if}\, b\, \mathbf{then}\, S_1\, \mathbf{else}\, S_2\, \mathbf{end}, (h,e)\rangle \to (h',e') \wedge \\
& o \notin \mathsf{UseSet}[\![\mathbf{if}\, b\, \mathbf{then}\, S_1\, \mathbf{else}\, S_2\, \mathbf{end}]\!](h,e) \wedge \\
& \mathsf{Expr}[\![b]\!](h,e) = \mathsf{const}(\mathtt{true}) \Rightarrow \\
& \langle \mathbf{if}\, b\, \mathbf{then}\, S_1\, \mathbf{else}\, S_2\, \mathbf{end}, (h\langle o.b := x\rangle, e)\rangle \to (h'\langle o.b := y\rangle, e') \\
\mathit{(36)} \; \Rrightarrow \; & \langle \mathbf{if}\, b\, \mathbf{then}\, S_1\, \mathbf{else}\, S_2\, \mathbf{end}, (h,e)\rangle \to (h',e') \wedge \\
& o \notin \mathsf{UseSet}[\![b]\!](h,e) \wedge o \notin \mathsf{UseSet}[\![S_1]\!](h,e) \wedge \\
& \mathsf{Expr}[\![b]\!](h,e) = \mathsf{const}(\mathtt{true}) \Rightarrow \\
& \langle \mathbf{if}\, b\, \mathbf{then}\, S_1\, \mathbf{else}\, S_2\, \mathbf{end}, (h\langle o.b := x\rangle, e)\rangle \to (h'\langle o.b := y\rangle, e') \\
\mathit{(IH1)} \; \Rrightarrow \; & \langle \mathbf{if}\, b\, \mathbf{then}\, S_1\, \mathbf{else}\, S_2\, \mathbf{end}, (h,e)\rangle \to (h',e') \wedge \\
& o \notin \mathsf{UseSet}[\![b]\!](h,e) \wedge o \notin \mathsf{UseSet}[\![S_1]\!](h,e) \wedge \\
& \mathsf{Expr}[\![b]\!](h,e) = \mathsf{const}(\mathtt{true}) \wedge \mathsf{Expr}[\![b]\!](h\langle o.b := x\rangle, e) = \mathsf{const}(\mathtt{true}) \Rightarrow \\
& \langle \mathbf{if}\, b\, \mathbf{then}\, S_1\, \mathbf{else}\, S_2\, \mathbf{end}, (h\langle o.b := x\rangle, e)\rangle \to (h'\langle o.b := y\rangle, e') \\
\mathit{(20)} \; \Rrightarrow \; & \langle S_1, (h,e)\rangle \to (h',e') \wedge \\
& o \notin \mathsf{UseSet}[\![b]\!](h,e) \wedge o \notin \mathsf{UseSet}[\![S_1]\!](h,e) \Rightarrow \\
& \langle S_1, (h\langle o.b := x\rangle, e)\rangle \to (h'\langle o.b := y\rangle, e') \\
\mathit{(IH2)} \; \Rrightarrow \; & \langle S_1, (h\langle o.b := x\rangle, e)\rangle \to (h'\langle o.b := y\rangle, e') \Rightarrow \langle S_1, (h\langle o.b := x\rangle, e)\rangle \to (h'\langle o.b := y\rangle, e') \; \square
\end{aligned}
$$

loop continuation

$$
\begin{array}{rl}
& \langle \textbf{from until}\, b\, \textbf{loop}\, S\, \textbf{end}, (h, e)\rangle \to (h', e') \wedge \\
& o \notin \mathsf{UseSet}\llbracket \textbf{from loop}\, b\, \textbf{loop}\, S\, \textbf{end}\rrbracket(h, e) \wedge \\
& \mathsf{Expr}\llbracket b\rrbracket(h, e) = \mathsf{const}(\texttt{true}) \Rightarrow \\
& \langle \textbf{from until}\, b\, \textbf{loop}\, S\, \textbf{end}, (h\langle o.b := x\rangle, e)\rangle \to (h'\langle o.b := y\rangle, e') \\
\textit{(38)}\ \Rrightarrow & \langle \textbf{from until}\, b\, \textbf{loop}\, S\, \textbf{end}, (h, e)\rangle \to (h', e') \wedge \\
& o \notin \mathsf{UseSet}\llbracket b\rrbracket(h, e) \wedge o \notin \mathsf{UseSet}\llbracket S\rrbracket(h, e) \wedge o \notin \mathsf{UseSet}\llbracket \textbf{from loop}\, b\, \textbf{loop}\, S\, \textbf{end}\rrbracket(h', e') \wedge \\
& \mathsf{Expr}\llbracket b\rrbracket(h, e) = \mathsf{const}(\texttt{true}) \Rightarrow \\
& \langle \textbf{from until}\, b\, \textbf{loop}\, S\, \textbf{end}, (h\langle o.b := x\rangle, e)\rangle \to (h'\langle o.b := y\rangle, e') \\
\textit{(IH1)}\ \Rrightarrow & \langle \textbf{from until}\, b\, \textbf{loop}\, S\, \textbf{end}, (h, e)\rangle \to (h', e') \wedge \\
& o \notin \mathsf{UseSet}\llbracket b\rrbracket(h, e) \wedge o \notin \mathsf{UseSet}\llbracket S\rrbracket(h, e) \wedge o \notin \mathsf{UseSet}\llbracket \textbf{from loop}\, b\, \textbf{loop}\, S\, \textbf{end}\rrbracket(h', e') \wedge \\
& \mathsf{Expr}\llbracket b\rrbracket(h, e) = \mathsf{const}(\texttt{true}) \wedge \mathsf{Expr}\llbracket b\rrbracket(h\langle o.b := x\rangle, e) = \mathsf{const}(\texttt{true}) \Rightarrow \\
& \langle \textbf{from until}\, b\, \textbf{loop}\, S\, \textbf{end}, (h\langle o.b := x\rangle, e)\rangle \to (h'\langle o.b := y\rangle, e') \\
\textit{(22)}\ \Rrightarrow & \exists h^*, e^* : \langle S\rangle(h, e) \to (h^*, e^*) \wedge \langle \textbf{from until}\, b\, \textbf{loop}\, S\, \textbf{end}, (h^*, e^*)\rangle \to (h', e') \wedge \\
& o \notin \mathsf{UseSet}\llbracket S\rrbracket(h, e) \wedge o \notin \mathsf{UseSet}\llbracket \textbf{from loop}\, b\, \textbf{loop}\, S\, \textbf{end}\rrbracket(h', e') \Rightarrow \\
& \exists h^{**}, e^{**} : \langle S\rangle(h\langle o.b := x\rangle, e)) \to (h^{**}, e^{**}) \wedge \langle \textbf{from until}\, b\, \textbf{loop}\, S\, \textbf{end}, (h^{**}, e^{**})\rangle \to (h'\langle o.b := y\rangle, e') \\
\textit{(IH2)}\ \Rrightarrow & \exists h^*, e^*, z : \langle S\rangle(h\langle o.b := x\rangle, e)) \to (h^*\langle o.a := z\rangle, e^*) \wedge \\
& \langle \textbf{from until}\, b\, \textbf{loop}\, S\, \textbf{end}, (h^*\langle o.a := z\rangle, e^*)\rangle \to (h'\langle o.b := y\rangle, e') \Rightarrow \\
& \exists h^{**}, e^{**} : \langle S\rangle(h\langle o.b := x\rangle, e)) \to (h^{**}, e^{**}) \wedge \langle \textbf{from until}\, b\, \textbf{loop}\, S\, \textbf{end}, (h^{**}, e^{**})\rangle \to (h'\langle o.b := y\rangle, e') \ \Box
\end{array}
$$

feature invocation

$$
\begin{array}{rl}
& \langle T.c(A1, \dots)\rangle(h, e) \to (h', e) \wedge o \notin \mathsf{UseSet}\llbracket T.c(A1, \dots)\rrbracket(h, e) \Rightarrow \\
& \langle T.c(A1, \dots)\rangle(h\langle o.b := y\rangle, e) \to (h'\langle o.b := y\rangle, e) \\
\textit{(24)}\ \Rrightarrow & \langle Body_c, (h, \langle \texttt{Current} := \mathsf{Expr}\llbracket T\rrbracket(h, e), arg1 := \mathsf{Expr}\llbracket A1\rrbracket(h, e), \dots\rangle)\rangle \to (h', e') \wedge \\
& o \notin \mathsf{UseSet}\llbracket T.c(A1, \dots)\rrbracket(h, e) \Rightarrow \\
& \langle Body_c, (h\langle o.b := x\rangle, \langle \texttt{Current} := \mathsf{Expr}\llbracket T\rrbracket(h\langle o.b := y\rangle, e), \\
& arg1 := \mathsf{Expr}\llbracket A1\rrbracket(h\langle o.b := y\rangle, e), \dots\rangle)\rangle \to (h'\langle o.b := y\rangle, e') \\
\textit{(40)}\ \Rrightarrow & \langle Body_c, (h, \langle \texttt{Current} := \mathsf{Expr}\llbracket T\rrbracket(h, e), arg1 := \mathsf{Expr}\llbracket A1\rrbracket(h, e), \dots\rangle)\rangle \to (h', e') \wedge \\
& o \notin \mathsf{UseSet}\llbracket T\rrbracket(h, e) \wedge o \notin \mathsf{UseSet}\llbracket A1\rrbracket(h, e) \wedge \\
& o \notin \mathsf{UseSet}\llbracket Body_c\rrbracket(h, \langle \texttt{Current} := \mathsf{Expr}\llbracket T\rrbracket(h, e), arg1 := \mathsf{Expr}\llbracket A1\rrbracket(h, e), \dots\rangle) \Rightarrow \\
& \langle Body_c, (h\langle o.b := x\rangle, \langle \texttt{Current} := \mathsf{Expr}\llbracket T\rrbracket(h\langle o.b := y\rangle, e), \\
& arg1 := \mathsf{Expr}\llbracket A1\rrbracket(h\langle o.b := y\rangle, e), \dots\rangle)\rangle \to (h'\langle o.b := y\rangle, e') \\
\textit{(IH1)}\ \Rrightarrow & \langle Body_c, (h, \langle \texttt{Current} := \mathsf{Expr}\llbracket T\rrbracket(h, e), arg1 := \mathsf{Expr}\llbracket A1\rrbracket(h, e), \dots\rangle)\rangle \to (h', e') \wedge \\
& o \notin \mathsf{UseSet}\llbracket Body_c\rrbracket(h, \langle \texttt{Current} := \mathsf{Expr}\llbracket T\rrbracket(h, e), arg1 := \mathsf{Expr}\llbracket A1\rrbracket(h, e), \dots\rangle) \Rightarrow \\
& \langle Body_c, (h\langle o.b := x\rangle, \langle \texttt{Current} := \mathsf{Expr}\llbracket T\rrbracket(h, e), arg1 := \mathsf{Expr}\llbracket A1\rrbracket(h, e), \dots\rangle)\rangle \to (h'\langle o.b := y\rangle, e') \\
\textit{(IH2)}\ \Rrightarrow & \langle Body_c, (h\langle o.b := x\rangle, \langle \texttt{Current} := \mathsf{Expr}\llbracket T\rrbracket(h, e), arg1 := \mathsf{Expr}\llbracket A1\rrbracket(h, e), \dots\rangle)\rangle \to (h'\langle o.b := y\rangle, e') \Rightarrow \\
& \langle Body_c, (h\langle o.b := x\rangle, \langle \texttt{Current} := \mathsf{Expr}\llbracket T\rrbracket(h, e), arg1 := \mathsf{Expr}\llbracket A1\rrbracket(h, e), \dots\rangle)\rangle \to (h'\langle o.b := y\rangle, e') \ \Box
\end{array}
$$

object creation

$$\begin{aligned}
&\langle \mathbf{create}\, l.c(A1,\dots)\rangle(h,e) \to (h', e\langle l := \mathsf{new}(h,C)\rangle)\wedge \\
&o \notin \mathsf{UseSet}[\![\mathbf{create}\, l.c(A1,\dots)]\!](h,e) \Rightarrow \\
&\langle \mathbf{create}\, l.c(A1,\dots)\rangle(h\langle o.b := y\rangle, e) \to (h'\langle o.b := y\rangle, e\langle l := \mathsf{new}(h\langle o.b := y\rangle, C)\rangle) \\
\mathit{(40)}\ \Rrightarrow\ &\langle \mathbf{create}\, l.c(A1,\dots)\rangle(h,e) \to (h', e\langle l := \mathsf{new}(h,C)\rangle)\wedge \\
&o \notin \mathsf{UseSet}[\![Body_c]\!](h, \langle \mathtt{Current} := \mathsf{Expr}[\![T]\!](h,e), arg1 := \mathsf{Expr}[\![A1]\!](h,e),\dots\rangle)\wedge \\
&o \notin \mathsf{UseSet}[\![T]\!](h,e) \wedge o \notin \mathsf{UseSet}[\![A1]\!](h,e) \wedge \dots \Rightarrow \\
&\langle \mathbf{create}\, l.c(A1,\dots)\rangle(h\langle o.b := y\rangle, e) \to (h'\langle o.b := y\rangle, e\langle l := \mathsf{new}(h\langle o.b := y\rangle, C)\rangle) \\
\mathit{(24)}\ \Rrightarrow\ &\langle Body_c, (h, \langle \mathtt{Current} := \mathsf{Expr}[\![T]\!](h,e), arg1 := \mathsf{Expr}[\![A1]\!](h,e),\dots\rangle)\rangle \to (h', e')\wedge \\
&o \notin \mathsf{UseSet}[\![Body_c]\!](h, \langle \mathtt{Current} := \mathsf{Expr}[\![T]\!](h,e), arg1 := \mathsf{Expr}[\![A1]\!](h,e),\dots\rangle)\wedge \\
&o \notin \mathsf{UseSet}[\![T]\!](h,e) \wedge o \notin \mathsf{UseSet}[\![A1]\!](h,e) \wedge \dots \Rightarrow \\
&\langle Body_c, (h\langle o.b := x\rangle, \langle \mathtt{Current} := \mathsf{Expr}[\![T]\!](h\langle o.b := x\rangle, e), arg1 := \mathsf{Expr}[\![A1]\!](h\langle o.b := x\rangle, e),\dots\rangle)\rangle \to \\
&(h'\langle o.b := y\rangle, e') \\
\mathit{(IH1)}\ \Rrightarrow\ &\langle Body_c, (h, \langle \mathtt{Current} := \mathsf{Expr}[\![T]\!](h,e), arg1 := \mathsf{Expr}[\![A1]\!](h,e),\dots\rangle)\rangle \to (h', e')\wedge \\
&o \notin \mathsf{UseSet}[\![Body_c]\!](h, \langle \mathtt{Current} := \mathsf{Expr}[\![T]\!](h,e), arg1 := \mathsf{Expr}[\![A1]\!](h,e),\dots\rangle) \Rightarrow \\
&\langle Body_c, (h\langle o.b := x\rangle, \langle \mathtt{Current} := \mathsf{Expr}[\![T]\!](h,e), arg1 := \mathsf{Expr}[\![A1]\!](h,e),\dots\rangle)\rangle \to (h'\langle o.b := y\rangle, e') \\
\mathit{(IH2)}\ \Rrightarrow\ &\langle Body_c, (h\langle o.b := x\rangle, \langle \mathtt{Current} := \mathsf{Expr}[\![T]\!](h,e), arg1 := \mathsf{Expr}[\![A1]\!](h,e),\dots\rangle)\rangle \to (h'\langle o.b := y\rangle, e') \Rightarrow \\
&\langle Body_c, (h\langle o.b := x\rangle, \langle \mathtt{Current} := \mathsf{Expr}[\![T]\!](h,e), arg1 := \mathsf{Expr}[\![A1]\!](h,e),\dots\rangle)\rangle \to (h'\langle o.b := y\rangle, e')\ \Box
\end{aligned}$$

Through repeated application of the proved theorem, we have shown that we can change any number of objects outside of the use frame and still preserve the return value of the expressions:

$$\mathsf{Expr}[\![E]\!](h,e) = res \wedge (\forall o \in \mathsf{UseSet}[\![E]\!](s), a : h(o.a) = h'(o.a)) \Rightarrow \mathsf{Expr}[\![E]\!](h',e)$$

8.3 Frame separation property

The following axiomatic rule describes that we can preserve a property on a query q over the call of a routine c if we know that the frame of the query and the frame of the call are separate:

$$\frac{(q\,\mathbf{use}, qf) \qquad (c\,\mathbf{modify}\,cf) \qquad \{Q\}c\{R\}}{\{Q \wedge P(q) \wedge qf \cap cf = \emptyset\}c\{R \wedge P(q)\}}$$

$q\,\mathbf{use}\,qf$ means that the query q has a use from defined through the expression qf. $c\,\mathbf{modify}\,cf$ means that c has a modification frame defined as c. As both informations are available staticly, we can assume their correctness properties ModOK and UseOK as defined in section 7.3.

We translate the Hoare-triple into its form in the operational semantics and prove it (assuming that s satisfies the precondition of c and q as well as the invariant):

$$
\begin{aligned}
& \langle c, s\rangle \rightarrow s' \wedge P(\mathsf{Expr}\llbracket q \rrbracket(s)) \wedge \mathsf{Expr}\llbracket qf \rrbracket(s) \cap \mathsf{Expr}\llbracket cf \rrbracket(s) = \emptyset \Rightarrow P(\mathsf{Expr}\llbracket q \rrbracket(s')) \\
(\mathit{ModOK}) \Rrightarrow\; & \langle c, s\rangle \rightarrow s' \wedge P(\mathsf{Expr}\llbracket q \rrbracket(s)) \wedge \mathsf{Expr}\llbracket qf \rrbracket(s) \cap (\mathsf{ModifySet}\llbracket c \rrbracket(s) \cap \mathsf{Used}(s_H)) = \emptyset \Rightarrow \\
& P(\mathsf{Expr}\llbracket q \rrbracket(s')) \\
(\mathit{UseOK}) \Rrightarrow\; & \langle c, s\rangle \rightarrow s' \wedge P(\mathsf{Expr}\llbracket q \rrbracket(s)) \wedge (\mathsf{UseSet}\llbracket q \rrbracket(s) \cap \mathsf{ModifySet}\llbracket c \rrbracket(s) \cap \mathsf{Used}(s_H)) = \emptyset \Rightarrow \\
& P(\mathsf{Expr}\llbracket q \rrbracket(s')) \\
(\mathit{Sec.8.1}) \Rrightarrow\; & \langle c, s\rangle \rightarrow s' \wedge P(\mathsf{Expr}\llbracket q \rrbracket(s)) \wedge (\mathsf{UseSet}\llbracket q \rrbracket(s) \cap \mathsf{ModifySet}\llbracket c \rrbracket(s) \cap \mathsf{Used}(s_H)) = \emptyset \wedge \\
& (\forall o \in \mathsf{Used}(s_H) - \mathsf{ModifySet}\llbracket c \rrbracket(s), a : s_H(o.a) = s'_H(o.a)) \Rightarrow P(\mathsf{Expr}\llbracket q \rrbracket(s')) \\
(\mathit{24}) \Rrightarrow\; & \langle c, (h, e)\rangle \rightarrow (h', e) \wedge P(\mathsf{Expr}\llbracket q \rrbracket(h, e)) \wedge \\
& (\mathsf{UseSet}\llbracket q \rrbracket(h, e) \cap \mathsf{ModifySet}\llbracket c \rrbracket(h, e) \cap \mathsf{Used}(h)) = \emptyset \wedge \\
& (\forall o \in \mathsf{Used}(h) - \mathsf{ModifySet}\llbracket c \rrbracket(h, e), a : h(o.a) = h(o.a)) \Rightarrow P(\mathsf{Expr}\llbracket q \rrbracket(h', e)) \\
(\mathit{Sec.8.2}) \Rrightarrow\; & \langle c, (h, e)\rangle \rightarrow (h', e) \wedge P(\mathsf{Expr}\llbracket q \rrbracket(h, e)) \wedge \\
& (\mathsf{UseSet}\llbracket q \rrbracket(h, e) \cap \mathsf{ModifySet}\llbracket c \rrbracket(h, e) \cap \mathsf{Used}(h)) = \emptyset \wedge \\
& \mathsf{Expr}\llbracket q \rrbracket(h, e) = \mathsf{Expr}\llbracket q \rrbracket(h', e) \\
& (\forall o \in \mathsf{Used}(h) - \mathsf{ModifySet}\llbracket c \rrbracket(h, e), a : h(o.a) = h(o.a)) \Rightarrow P(\mathsf{Expr}\llbracket q \rrbracket(h', e)) \;\square
\end{aligned}
$$

8.4 Summary

We have defined a significant subset of Eiffel, called Eiffel0. This subset of Eiffel allows the implementation of standard algorithms and is powerful enough to expose problems of aliasing and frame. Then we have defined an operational semantics for this language as well as correctness conditions.

A special feature of Eiffel0 is the inclusion of frame specifications into the contracts of features. Frame specifications are expressions that evaluate to sets of objects. These frame sets describe which objects a command modifies, as well as which object a query uses to compute its result. We have defined what it means for an implementation to satisfy its frame contract.

With these definitions of correctness, we have show two important properties of frames: that a correct implementation of a command will not change any objects outside of its modify frame and that the result of a query will not change if the objects in its use frame are left unchanged.

Last but not least, we have created a Hoare-style rule that allows reasoning over feature invocations. We show the soundness of this rule.

References

[1] Bertrand Meyer. *Eiffel the Language*. Prentice Hall, 1992. Eiffel the Language.

[2] Bertrand Meyer. *Object-Oriented Software Construction*. Prentice Hall, 1997.

[3] Peter Müller. *Modular Specification and Verification of Object-Oriented Programs*. PhD thesis, Fern-Universität Hagen, 2001.