CLAIR: The Combined Language and Abstract Interpretation Resource

Roberto Bagnara
Department of Mathematics
University of Parma
Italy
bagnara@cs.unipr.it

Version 1.0

Abstract

The CLAIR system has been developed in order to study and experiment with various aspects of programming languages. This paper provides formal presentations of the supported languages, gives a brief description of the implementation, and explains, by means of examples, how the system can be used.

Contents

Chapter 1

Introduction

CLAIR is a system that has been developed in order to study and experiment with various aspects of programming languages. In particular:

1. lexical analysis;
2. syntactic analysis and generation of the abstract syntax tree (parsing);
3. static checking of type correctness;
4. operational semantics expressed by means of transition systems;
5. interpretation;
6. static analysis;
7. compilation.

CLAIR supports two languages: a simple functional language (SFL) and an imperative language (SIL) that recalls Pascal to some extent. Both languages adopt the static scoping rule.

The CLAIR approach is based on structured operational semantics *à la Plotkin* for the formal description [1] and on the Prolog language for the implementation. One of the advantages of this combined approach is that it is relatively easy to extend the system so as to support other language features.

Chapters 2 and 3 present an operational semantics of the SFL and SIL languages, respectively. These chapters, aiming at a formal and unambiguous description of the languages, are intentionally non-discursive.

Chapter 4 briefly illustrates the concrete syntax recognized by CLAIR's parser, also establishing its relation with the abstract syntax of Chapters 2 and 3 and with its representation adopted in the Prolog system.

Chapter 5 contains a terse description of the implementation.

Chapter 6 shows a typical working session with CLAIR.

Chapter 2

The SFL Language

2.1 Abstract Syntax

2.1.1 Base Sets and Corresponding Syntactic Meta-Variables

Basic types: $t \in \text{Types} \stackrel{\text{def}}{=} \{\text{int}, \text{bool}\}$;

Integers: $m, n \in \text{Int} \stackrel{\text{def}}{=} \{\ldots, -2, -1, 0, 1, 2, \ldots\}$;

Booleans: $b \in T \stackrel{\text{def}}{=} \{\text{tt}, \text{ff}\}$;

Variables: $x, y \in \text{Var} \stackrel{\text{def}}{=} \{x_1, x_2, \ldots\}$;

Binary operators: $\text{bop} \in \text{Bop} \stackrel{\text{def}}{=} \{+, -, *, \text{div}, \text{mod}, ==, <>, <=, >=, <, >, \text{and}, \text{or}\}$.

2.1.2 Derived Sets:

Expressible types: $\text{et} \in \text{ETypes}$

$$\text{et} ::= t \tag{2.1}$$

Denotable types: $\text{dt} \in \text{DTypes}$

$$\text{dt} ::= \text{et} | \text{form} \to \text{et} \tag{2.2}$$

Constants: $\text{con} \in \text{Con}$

$$\text{con} ::= m | b \tag{2.3}$$

Declarations: $d, d_0, d_1 \in \text{Def}$

$$d ::= \textbf{nil} \mid x : t = e | d_0; d_1 | d_0 \textbf{ parallel } d_1 | d_0 \textbf{ private } d_1 | \textbf{ function } f(\text{form}) : \text{et} = e | \textbf{ rec } d \tag{2.4}$$

Expressions: $e, e_0, e_1 \in \text{Exp}$

$$e ::= \text{con} | x | e_0 \text{bop} e_1 | \textbf{ not } e | -e | \textbf{ if } e \textbf{ then } e_0 \textbf{ else } e_1 | \textbf{ let } d \textbf{ in } e | \textbf{ input } \text{et} | f(\text{ae}) \tag{2.5}$$

Actual expressions: ae ∈ ActExp

$$\text{ae} ::= .|e, \text{ae} \tag{2.6}$$

Formal parameters: form ∈ Formals

$$\text{form} ::= .|x : t, \text{form}| \ \textbf{name}\ x : t, \text{form}\ \textbf{function}\ f(\text{form}) \to \text{et}, \text{form} \tag{2.7}$$

2.2 Static Semantics

Let $V \subseteq_{\text{f}} \text{Var}$ and $\text{TEnv}(V) \stackrel{\text{def}}{=} V \to \text{DTypes}$. Then

$$\alpha : V \stackrel{\text{def}}{\Longleftrightarrow} \alpha \in \text{TEnv}(V) \tag{2.8}$$

and

$$\text{TEnv} \stackrel{\text{def}}{=} \bigcup_{V \subseteq \text{Var}} \text{TEnv}(V). \tag{2.9}$$

For each $V \subseteq_{\text{f}} \text{Var}$ and each $\alpha : V$ we will define the following predicates:

$$\alpha \vdash \text{con} : t, \qquad \text{where con} \in \text{Con and } t \in \text{Types}; \tag{2.10}$$
$$\alpha \vdash e : \text{et}, \qquad \text{where } e \in \text{Exp and et} \in \text{ETypes}; \tag{2.11}$$
$$\alpha \vdash d, \qquad \text{where } d \in \text{Def}; \tag{2.12}$$
$$\vdash d : \beta, \qquad \text{where } d \in \text{Def and } \beta \in \text{TEnv}; \tag{2.13}$$
$$\alpha, \text{form} \vdash \text{ae}, \qquad \text{where form} \in \text{Formals and ae} \in \text{ActExp}; \tag{2.14}$$

We will also define the auxiliary predicates

$$\text{form} : \beta, \text{with form} \in \text{Formals}; \tag{2.15}$$
$$\mathit{match}(\text{form}_1, \text{form}_2), \text{with form}_1, \text{form}_2 \in \text{Formals}. \tag{2.16}$$

and the usual polymorphic functions FV and DV.

2.2.1 Constants

$$\text{FV}(m) \stackrel{\text{def}}{=} \varnothing, \qquad \text{if } \alpha \vdash m : \text{int}; \tag{2.17}$$
$$\text{FV}(b) \stackrel{\text{def}}{=} \varnothing, \qquad \text{if } \alpha \vdash b : \text{bool}. \tag{2.18}$$

2.2.2 Expressions

Constants:

$$\text{FV}(\text{con}) \stackrel{\text{def}}{=} \text{FV}(\text{con}), \qquad \text{(sic!)}; \tag{2.19}$$
$$\frac{\alpha \vdash \text{con} : t}{\alpha \vdash \text{con} : t} \qquad \text{(sic!)}. \tag{2.20}$$

Variables:

$$\mathrm{FV}(x) \stackrel{\mathrm{def}}{=} \{x\}, \tag{2.21}$$
$$\alpha \vdash x : \alpha(x). \tag{2.22}$$

Boolean Negation:

$$\mathrm{FV}(\textbf{not}\ e) \stackrel{\mathrm{def}}{=} \mathrm{FV}(e), \tag{2.23}$$
$$\frac{\alpha \vdash e : \mathrm{bool}}{\alpha \vdash \textbf{not}\ e : \mathrm{bool}} \tag{2.24}$$

Unary Minus:

$$\mathrm{FV}(-e) \stackrel{\mathrm{def}}{=} \mathrm{FV}(e), \tag{2.25}$$
$$\frac{\alpha \vdash e : \mathrm{int}}{\alpha \vdash -e : \mathrm{int}}. \tag{2.26}$$

Expression Block:

$$\mathrm{FV}(\textbf{let}\ d\ \textbf{in}\ e) \stackrel{\mathrm{def}}{=} \mathrm{FV}(e) \setminus \mathrm{DV}(d), \tag{2.27}$$
$$\frac{\alpha \vdash d, \vdash d : \beta, \alpha[\beta] \vdash e : \mathrm{et}}{\alpha \vdash \textbf{let}\ d\ \textbf{in}\ e : \mathrm{et}}\ \text{if } \beta : W. \tag{2.28}$$

Conditional Expression:

$$\mathrm{FV}(\textbf{if}\ e\ \textbf{then}\ e_1\ \textbf{else}\ e_2) \stackrel{\mathrm{def}}{=} \mathrm{FV}(e) \cup \mathrm{FV}(e_1) \cup \mathrm{FV}(e_2) \tag{2.29}$$
$$\frac{\alpha \vdash e : \mathrm{bool}, \alpha \vdash e_1 : \mathrm{et}, \alpha \vdash e_2 : \mathrm{et}}{\alpha \vdash \textbf{if}\ e\ \textbf{then}\ e_1\ \textbf{else}\ e_2 : \mathrm{et}} \tag{2.30}$$

Function Call:

$$\mathrm{FV}(f(\mathrm{ae})) \stackrel{\mathrm{def}}{=} \{f\} \cup \mathrm{FV}(\mathrm{ae}), \tag{2.31}$$
$$\frac{\alpha, \mathrm{form} \vdash \mathrm{ae}}{\alpha \vdash f(\mathrm{ae}) : \mathrm{et}}\ \text{if } \alpha(f) = \mathrm{form} \to \mathrm{et}. \tag{2.32}$$

Input:

$$\mathrm{FV}(\textbf{input}\ \mathrm{et}) \stackrel{\mathrm{def}}{=} \varnothing, \tag{2.33}$$
$$\alpha \vdash \textbf{input}\ \mathrm{et} : \mathrm{et}. \tag{2.34}$$

Binary Operators: If $\mathrm{bop} \in \{+, -, *, div, mod\}$,

$$t_1\ \mathrm{bop}\ t_2 = \mathrm{int} \iff t_1 = \mathrm{int}, t_2 = \mathrm{int}; \tag{2.35}$$

if $\mathrm{bop} \in \{==, <>, <=, >=, <, >\}$

$$t_1\ \mathrm{bop}\ t_2 = \mathrm{bool} \iff t_1 = \mathrm{int}, t_2 = \mathrm{int}; \tag{2.36}$$

if bop $\in \{and, or\}$

$$t_1 \text{ bop } t_2 = \text{bool} \iff t_1 = \text{bool}, t_2 = \text{bool}. \tag{2.37}$$

With this definition we define

$$\text{FV}(e_1 \text{ bop } e_2) \stackrel{\text{def}}{=} \text{FV}(e_1) \cup \text{FV}(e_2), \tag{2.38}$$

$$\frac{\alpha \vdash e_1 : \text{et}_1, \alpha \vdash e_2 : \text{et}_2}{\alpha \vdash e_1 \text{ bop } e_2 : \text{et}} \text{ if et} = \text{et}_1 \text{tbopet}_2. \tag{2.39}$$

Actual Expressions

$$\text{FV}(.) \stackrel{\text{def}}{=} \varnothing, \tag{2.40}$$

$$\text{FV}(e, \text{ae}) \stackrel{\text{def}}{=} \text{FV}(e) \cup \text{FV}(\text{ae}). \tag{2.41}$$

2.2.3 Declarations

Nil:

$$\text{FV}(\textbf{nil}) \stackrel{\text{def}}{=} \varnothing, \tag{2.42}$$

$$\text{DV}(\textbf{nil}) \stackrel{\text{def}}{=} \varnothing, \tag{2.43}$$

$$\alpha \vdash \textbf{nil}, \tag{2.44}$$

$$\vdash \textbf{nil}: \varnothing. \tag{2.45}$$

Simple Declaration:

$$\text{FV}(x : \text{et} = e) \stackrel{\text{def}}{=} \text{FV}(e), \tag{2.46}$$

$$\text{DV}(x : \text{et} = e) \stackrel{\text{def}}{=} \{x\}, \tag{2.47}$$

$$\tag{2.48}$$

$$\frac{\alpha \vdash e : \text{et}}{\alpha \vdash x : \text{et} = e} \tag{2.49}$$

$$\vdash x : \text{et} = e : \{x = \text{et}\}. \tag{2.50}$$

Sequential Composition:

$$\text{FV}(d_1; d_2) \stackrel{\text{def}}{=} \text{FV}(d_1) \cup (\text{FV}(d_2) \setminus \text{DV}(d_1)), \tag{2.51}$$

$$\text{DV}(d_1; d_2) \stackrel{\text{def}}{=} \text{DV}(d_1) \cup \text{DV}(d_2), \tag{2.52}$$

$$\frac{\alpha \vdash d_1, \vdash d_1 : \beta, \alpha[\beta] \vdash d_2}{\alpha \vdash d_1; d_2} \text{ if } \beta : W, \tag{2.53}$$

$$\frac{\vdash d_1 : \alpha, \vdash d_2 : \beta}{\vdash d_1; d_2 : \alpha[\beta]} \tag{2.54}$$

Parallel Composition:

$$\mathrm{FV}(d_1 \textbf{ parallel } d_2) \stackrel{\mathrm{def}}{=} \mathrm{FV}(d_1) \cup \mathrm{FV}(d_2), \tag{2.55}$$

$$\mathrm{DV}(d_1 \textbf{ parallel } d_2) \stackrel{\mathrm{def}}{=} \mathrm{DV}(d_1) \cup \mathrm{DV}(d_2), \tag{2.56}$$

$$\frac{\alpha \vdash d_1, \alpha \vdash d_2}{\alpha \vdash d_1 \textbf{ parallel } d_2} \text{ if } \mathrm{DV}(d_1) \cap \mathrm{DV}(d_2) = \varnothing, \tag{2.57}$$

$$\frac{\vdash d_1 : \alpha, \vdash d_2 : \beta}{\vdash d_1 \textbf{ parallel } d_2 : \alpha \cup \beta} \tag{2.58}$$

Private Composition:

$$\mathrm{FV}(d_1 \textbf{ private } d_2) \stackrel{\mathrm{def}}{=} \mathrm{FV}(d_1) \cup (\mathrm{FV}(d_2) \setminus \mathrm{DV}(d_1)), \tag{2.59}$$

$$\mathrm{DV}(d_1 \textbf{ private } d_2) \stackrel{\mathrm{def}}{=} \mathrm{DV}(d_1) \cup \mathrm{DV}(d_2), \tag{2.60}$$

$$\frac{\alpha \vdash d_1, \vdash d_1 : \beta, \alpha[\beta] \vdash d_2}{\alpha \vdash d_1 \textbf{ private } d_2} \text{ if } \beta : W, \tag{2.61}$$

$$\frac{\vdash d_1 : \alpha, \vdash d_2 : \beta}{\vdash d_1 \textbf{ private } d_2 : \beta} \tag{2.62}$$

Function Declaration:

$$\mathrm{FV}(\textbf{function } f(\mathrm{form}) : \mathrm{et} = e) \stackrel{\mathrm{def}}{=} \mathrm{FV}(e) \setminus \mathrm{DV}(\mathrm{form}), \tag{2.63}$$

$$\mathrm{DV}(\textbf{function } f(\mathrm{form}) : \mathrm{et} = e) \stackrel{\mathrm{def}}{=} \{f\} \tag{2.64}$$

$$\frac{\mathrm{form} : \beta, \alpha[\beta] \vdash e : \mathrm{et}}{\alpha \vdash \textbf{function } f(\mathrm{form}) : \mathrm{et} = e} \text{ if } \beta : W, \tag{2.65}$$

$$\vdash \textbf{function } f(\mathrm{form}) : \mathrm{et} = e : \{f = \mathrm{form} \to \mathrm{et}\}. \tag{2.66}$$

Recursive Declaration:

$$\mathrm{FV}(\textbf{rec } d) \stackrel{\mathrm{def}}{=} \mathrm{FV}(d) \setminus \mathrm{DV}(d), \tag{2.67}$$

$$\mathrm{DV}(\textbf{rec } d) \stackrel{\mathrm{def}}{=} \mathrm{DV}(d), \tag{2.68}$$

$$\frac{\vdash d : \beta, \alpha[\beta \P W] \vdash d}{\alpha \vdash \textbf{rec } d} \text{ if } W = \mathrm{FV}(d) \cap \mathrm{DV}(d), \tag{2.69}$$

$$\frac{\vdash d : \beta}{\vdash \textbf{rec } d : \beta} \tag{2.70}$$

Formal Parameters

Defined Variables:

$$\mathrm{DV}(.) \stackrel{\mathrm{def}}{=} \varnothing, \tag{2.71}$$

$$\mathrm{DV}(x : \mathrm{et}, \mathrm{form}) \stackrel{\mathrm{def}}{=} \{x\} \cup \mathrm{DV}(\mathrm{form}), \tag{2.72}$$

$$\mathrm{DV}(\textbf{name}\ x : \mathrm{et}, \mathrm{form}) \stackrel{\mathrm{def}}{=} \{x\} \cup \mathrm{DV}(\mathrm{form}), \tag{2.73}$$

$$\mathrm{DV}(\textbf{function}\ f(\mathrm{form}_1) : \mathrm{et}, \mathrm{form}) \stackrel{\mathrm{def}}{=} \{f\} \cup \mathrm{DV}(\mathrm{form}). \tag{2.74}$$

Predicate $\mathrm{form} : \beta$**:**

$$. : \varnothing, \tag{2.75}$$

$$\frac{\mathrm{form} : \beta}{(x : \mathrm{et}, \mathrm{form}) : \{x = \mathrm{et}\} \cup \beta} \text{ if } x \notin \mathrm{DV}(\mathrm{form}), \tag{2.76}$$

$$\frac{\mathrm{form} : \beta}{(\textbf{name}\ x : \mathrm{et}, \mathrm{form}) : \{x = \mathrm{et}\} \cup \beta} \text{ if } x \notin \mathrm{DV}(\mathrm{form}), \tag{2.77}$$

$$\frac{\mathrm{form} : \beta}{(\textbf{function}\ f(\mathrm{form}_1) : \mathrm{et}, \mathrm{form}) : \{f = \mathrm{form}_1 \to \mathrm{et}\} \cup \beta} \text{ if } f \notin \mathrm{DV}(\mathrm{form}), \tag{2.78}$$

Predicate $\mathrm{match}(\mathrm{form}, \mathrm{form})$**:** True if the lists of formals coincide as far as the number, order, type and mechanisms of the formals contained.

$$\mathrm{match}(.,.), \tag{2.79}$$

$$\frac{\mathrm{match}(\mathrm{form}_1, \mathrm{form}_2)}{\mathrm{match}((x : \mathrm{et}, \mathrm{form}_1), (y : \mathrm{et}, \mathrm{form}_2))} \tag{2.80}$$

$$\frac{\mathrm{match}(\mathrm{form}_1, \mathrm{form}_2)}{\mathrm{match}((\textbf{name}\ x : \mathrm{et}, \mathrm{form}_1), (\textbf{name}\ y : \mathrm{et}, \mathrm{form}_2))} \tag{2.81}$$

$$\frac{\mathrm{match}(f_1, f_2), \mathrm{match}(f_3, f_4)}{\mathrm{match}((\textbf{function}\ f(f_3) : \mathrm{et}, f_1), (\textbf{function}\ g(f_4) : \mathrm{et}, f_2))} \tag{2.82}$$

Predicate $\alpha, \mathrm{form} \vdash \mathrm{ae}$**:** This predicate defines the legality of an actual expression with respect to a formal parameter and a type environment.

$$\alpha, . \vdash . \tag{2.83}$$

Call by value:

$$\frac{\alpha, \mathrm{form} \vdash \mathrm{ae}, \alpha \vdash e : \mathrm{et}}{\alpha, (x : \mathrm{et}, \mathrm{form}) \vdash e, \mathrm{ae}} \tag{2.84}$$

Call by name:

$$\frac{\alpha, \mathrm{form} \vdash \mathrm{ae}, \alpha \vdash e : \mathrm{et}}{\alpha, (\textbf{name}\ x : \mathrm{et}, \mathrm{form}) \vdash e, \mathrm{ae}} \tag{2.85}$$

Functional parameters:

$$\frac{\mathrm{match}(\mathrm{form}_1, \mathrm{form}_2), (\alpha, \mathrm{form}) \vdash \mathrm{ae}}{\alpha, (\textbf{function}\ f(\mathrm{form}_1) : \mathrm{et}, \mathrm{form}) \vdash g, \mathrm{ae}} \text{ if } \alpha(g) = \mathrm{form}_2 \to \mathrm{et}. \tag{2.86}$$

2.3 Dynamic Semantics

We define the following sets, with the corresponding meta-variables:

$$\mathrm{cl} \in \mathrm{NExp} \stackrel{\mathrm{def}}{=} \left\{\, e \in \mathrm{Exp} \;\middle|\; \mathrm{FV}(e) = \varnothing \,\right\}, \tag{2.87}$$

$$\mathrm{Abstracts} \stackrel{\mathrm{def}}{=} \{\lambda \mathrm{form}.e : \mathrm{et}\}. \tag{2.88}$$

and then

$$\mathrm{dval} \in \mathrm{DVal} \stackrel{\mathrm{def}}{=} \mathrm{Con} + \mathrm{NExp} + \mathrm{Abstracts}. \tag{2.89}$$

Moreover, for each $V \subseteq_{\mathrm{f}} \mathrm{Var}$,

$$\mathrm{Env}(V) \stackrel{\mathrm{def}}{=} V \to \mathrm{DVal}, \tag{2.90}$$

$$\rho \in \mathrm{Env} \stackrel{\mathrm{def}}{=} \bigcup_{V \subseteq_{\mathrm{f}} \mathrm{Var}} \mathrm{Env}(V). \tag{2.91}$$

We add the following productions to the abstract syntax of SFL:

$$\mathrm{ae} ::= \lambda \mathrm{form}.e : \mathrm{et}, \mathrm{ae}, \tag{2.92}$$

$$d ::= \mathrm{form} = \mathrm{ae}, \tag{2.93}$$

$$d ::= \rho. \tag{2.94}$$

We also define:

$$\mathrm{FV}(\lambda \mathrm{form}.e : \mathrm{et}, \mathrm{ae}) \stackrel{\mathrm{def}}{=} (\mathrm{FV}(e) \setminus \mathrm{DV}(\mathrm{form})) \cup \mathrm{FV}(\mathrm{ae}), \tag{2.95}$$

$$\mathrm{DV}(\mathrm{form} = \mathrm{ae}) \stackrel{\mathrm{def}}{=} \mathrm{DV}(\mathrm{form}), \mathrm{FV}(\mathrm{form} = \mathrm{ae}) = \mathrm{FV}(\mathrm{ae}), \tag{2.96}$$

and

$$\frac{\alpha, \mathrm{form} \vdash \mathrm{ae}}{\alpha \vdash \mathrm{form} = \mathrm{ae}} \tag{2.97}$$

$$\frac{\mathrm{form} : \beta}{\vdash \mathrm{form} = \mathrm{ae} : \beta} \tag{2.98}$$

Finally, we define

$$\mathrm{DV}(\rho) \stackrel{\mathrm{def}}{=} V, \tag{2.99}$$

and, if $\rho : V$,

$$\mathrm{FV}(\rho) \stackrel{\mathrm{def}}{=} \bigcup_{x \in V} \mathrm{FV}(\rho(x)), \tag{2.100}$$

where FV(dval) is defined as follows:

$$\mathrm{FV}(\mathrm{con}) \stackrel{\mathrm{def}}{=} \varnothing, \tag{2.101}$$

$$\mathrm{FV}(\mathrm{cl}) \stackrel{\mathrm{def}}{=} \varnothing, \tag{2.102}$$

$$\mathrm{FV}(\lambda \mathrm{form}.e : \mathrm{et}) \stackrel{\mathrm{def}}{=} \mathrm{FV}(e) \setminus \mathrm{DV}(\mathrm{form}). \tag{2.103}$$

We are now in position to define the predicates $\vdash \mathrm{dval} : \mathrm{dt}$ and $\alpha \vdash \mathrm{dval}$.

Constants:

$$\vdash m : \mathrm{int}, \qquad \alpha \vdash m, \tag{2.104}$$

$$\vdash b : \mathrm{bool}, \qquad \alpha \vdash b. \tag{2.105}$$

Name Expressions:

$$\frac{\varnothing \vdash \mathrm{cl} : \mathrm{et}}{\vdash \mathrm{cl} : \mathrm{et}} \tag{2.106}$$

$$\alpha \vdash \mathrm{cl} \tag{2.107}$$

Abstracts:

$$\vdash \lambda \mathrm{form}.e : \mathrm{et} : \mathrm{form} \to \mathrm{et}, \tag{2.108}$$

$$\frac{\mathrm{form} : \beta, \alpha[\beta] \vdash e : \mathrm{et}}{\alpha \vdash \lambda \mathrm{form}.e : \mathrm{et}} \text{ if } \beta : W. \tag{2.109}$$

Now we can define the rules for environments. Let $\rho : V$. Then

$$\frac{\forall x \in V \vdash \rho(x) : \beta(x)}{\vdash \rho : \beta} \tag{2.110}$$

$$\frac{\forall x \in V \alpha \vdash \rho(x)}{\alpha \vdash \rho} \tag{2.111}$$

Let Acon the set defined by

$$\mathrm{acon} \in \mathrm{Acon} \tag{2.112}$$

where

$$\mathrm{acon} ::= .|\mathrm{con}, \mathrm{acon}|\lambda \mathrm{form}.e : \mathrm{et}, \mathrm{acon}|\mathrm{cl}, \mathrm{acon} \tag{2.113}$$

We can now define the configuration and relations of the transition system. We will define, for each $\alpha \in \mathrm{TEnv}$:

$$\Gamma(\alpha, E) = \{\, e \in \mathrm{Exp} \mid \exists \mathrm{et} \in \mathrm{ETypes} : \alpha \vdash e : \mathrm{et} \,\} \tag{2.114}$$

$$T(\alpha, E) = \mathrm{Con} \tag{2.115}$$

$$\rho \vdash e \to e' \tag{2.116}$$

$$\Gamma(\alpha, \mathrm{Ae}) = \{\, \mathrm{ae} \in \mathrm{ActExp} \mid \exists \mathrm{form} \in \mathrm{Formals} : \alpha, \mathrm{form} \vdash \mathrm{ae} \,\} \tag{2.117}$$
$$T(\alpha, \mathrm{Ae}) = \mathrm{Acon} \tag{2.118}$$

$$\rho, \mathrm{form} \vdash \mathrm{ae} \to \mathrm{ae}' \tag{2.119}$$

$$\Gamma(\alpha, D) = \{\, d \in \mathrm{Def} \mid \alpha \vdash d \,\} \tag{2.120}$$
$$T(\alpha, D) = \{\rho \in \mathrm{Env} \mid \alpha \vdash \rho\} \tag{2.121}$$

$$\rho \vdash d \to d'. \tag{2.122}$$

We will also define the predicate

$$\mathrm{acon} \vdash \mathrm{form} \to \rho. \tag{2.123}$$

2.3.1 Declarations

Nil:

$$\rho \vdash \mathbf{nil} \to \varnothing \tag{2.124}$$

Simple Definition:

$$\frac{\rho \vdash e \to e'}{\rho \vdash x : \mathrm{et} = e \to x : \mathrm{et} = e'} \tag{2.125}$$
$$\rho \vdash x : \mathrm{et} = \mathrm{con} \to \{x = \mathrm{con}\} \tag{2.126}$$

Sequential Composition:

$$\frac{\rho \vdash d_1 \to d_1'}{\rho \vdash d_1; d_2 \to d_1'; d_2} \tag{2.127}$$
$$\frac{\rho[\rho_0] \vdash d_2 \to d_2' \qquad \rho_0 : \alpha_0}{\rho \vdash \rho_0; d_2 \to \rho_0; d_2'} \tag{2.128}$$
$$\rho \vdash \rho_0; \rho_1 \to \rho_0[\rho_1] \tag{2.129}$$

Private Composition:

$$\frac{\rho \vdash d_1 \to d_1'}{\rho \vdash d_1 \ \mathbf{private}\ d_2 \to d_1' \ \mathbf{private}\ d_2} \tag{2.130}$$
$$\frac{\rho[\rho_0] \vdash d_2 \to d_2' \qquad \rho_0 : \alpha_0}{\rho \vdash \rho_0 \ \mathbf{private}\ d_2 \to \rho_0 \ \mathbf{private}\ d_2'} \tag{2.131}$$
$$\rho \vdash \rho_0 \ \mathbf{private}\ \rho_1 \to \rho_1 \tag{2.132}$$

Parallel Composition:

$$\frac{\rho \vdash d_1 \to d_1'}{\rho \vdash d_1 \textbf{ parallel } d_2 \to d_1' \textbf{ parallel } d_2} \tag{2.133}$$

$$\frac{\rho \vdash d_2 \to d_2' \qquad \rho_0 : \alpha_0}{\rho \vdash \rho_0 \textbf{ parallel } d_2 \to \rho_0 \textbf{ parallel } d_2'} \tag{2.134}$$

$$\rho \vdash \rho_0 \textbf{ parallel } \rho_1 \to \rho_0 \cup \rho_1 \tag{2.135}$$

Function Declaration:

$$\rho \vdash \textbf{function } f(\text{form}) : \text{et} = e \to \{\, f = \lambda \text{form}.\ \textbf{let } \rho \P W \textbf{ in } e : \text{et} \,\} \tag{2.136}$$

where $W = \text{FV}(e) \setminus \text{DV}(\text{form})$.

Recursive Declaration:

$$\frac{\rho \setminus R \vdash d \to d'}{\rho \vdash \textbf{rec } d \to \textbf{rec } d'} \text{ where } R \stackrel{\text{def}}{=} \text{DV}(d) \cap \text{FV}(d), \tag{2.137}$$

$$\rho \vdash \textbf{rec } \rho_0 \to \rho_1 \tag{2.138}$$

where

$$\begin{aligned} \rho_1 \stackrel{\text{def}}{=} & \{\, x = \text{con} \mid x = \text{con} \in \rho_0 \,\} \\ & \cup \{\, x = \text{cl} \mid x = \text{cl} \in \rho_0 \,\} \\ & \cup \left\{ f(\text{form}) : \text{et} = \textbf{letrec } \rho_0 \setminus F \textbf{ in } e \;\middle|\; \begin{array}{l} f(\text{form}) : \text{et} = e \in \rho_0 \\ F = \text{DV}(\text{form}) \end{array} \right\} \end{aligned}$$

Parameter Passing ($\text{form} = \text{ae}$**):**

$$\frac{\rho, \text{form} \vdash \text{ae} \to \text{ae}'}{\rho \vdash \text{form} = \text{ae} \to \text{form} = \text{ae}'} \tag{2.139}$$

$$\frac{\text{acon} \vdash \text{form} \to \rho_0}{\rho \vdash \text{form} = \text{acon} \to \rho_0} \tag{2.140}$$

Predicate $\rho, \text{form} \vdash \text{ae} \to \text{ae}'$**:**

$$., . \vdash . \to . \tag{2.141}$$

Call by value:

$$\frac{\rho \vdash e \to e'}{\rho, (x : \text{et}, \text{form}) \vdash e, \text{ae} \to e', \text{ae}}, \tag{2.142}$$

$$\frac{\rho, \text{form} \vdash \text{ae} \to \text{ae}'}{\rho, (x : \text{et}, \text{form}) \vdash \text{con}, \text{ae} \to \text{con}, \text{ae}'} \tag{2.143}$$

Call by name:

$$\frac{V \stackrel{\text{def}}{=} \mathrm{FV}(e) \neq \varnothing}{\rho, (\textbf{name}\ x : \mathrm{et}, \mathrm{form}) \vdash e, \mathrm{ae} \rightarrow \textbf{let}\ \rho\P V\ \textbf{in}\ e, \mathrm{ae}}, \tag{2.144}$$

$$\frac{\rho, \mathrm{form} \vdash \mathrm{ae} \rightarrow \mathrm{ae}'}{\rho, (\textbf{name}\ x : \mathrm{et}, \mathrm{form}) \vdash \mathrm{cl}, \mathrm{ae} \rightarrow \mathrm{cl}, \mathrm{ae}'}. \tag{2.145}$$

Functional parameter:

$$\frac{\rho(g) = \lambda \mathrm{form}_2.e : \mathrm{et}}{\rho, (\textbf{function}\ f(\mathrm{form}_1) : \mathrm{et}, \mathrm{form}) \vdash g, \mathrm{ae} \rightarrow \lambda \mathrm{form}_2.e : \mathrm{et}, \mathrm{ae}}, \tag{2.146}$$

$$\frac{\rho, \mathrm{form} \vdash \mathrm{ae} \rightarrow \mathrm{ae}'}{\rho, (\textbf{function}\ f(\mathrm{form}_1) : \mathrm{et}, \mathrm{form}) \vdash \lambda \mathrm{form}_2.e : \mathrm{et}, \mathrm{ae} \rightarrow \lambda \mathrm{form}_2.e : \mathrm{et}, \mathrm{ae}'} \tag{2.147}$$

Predicate $\mathrm{acon} \vdash \mathrm{form} \rightarrow \rho$**:**

$$. \vdash . \rightarrow \varnothing \tag{2.148}$$

$$\frac{\mathrm{acon} \vdash \mathrm{form} \rightarrow \rho}{\mathrm{con}, \mathrm{acon} \vdash x : \mathrm{et}, \mathrm{form} \rightarrow \{x = \mathrm{con}\} \cup \rho} \tag{2.149}$$

$$\frac{\mathrm{acon} \vdash \mathrm{form} \rightarrow \rho}{\mathrm{cl}, \mathrm{acon} \vdash \textbf{name}\ x : \mathrm{et}, \mathrm{form} \rightarrow \{x = \mathrm{cl}\} \cup \rho} \tag{2.150}$$

$$\frac{\mathrm{acon} \vdash \mathrm{form} \rightarrow \rho}{\lambda \mathrm{form}_1.e : \mathrm{et}, \mathrm{acon} \vdash \textbf{function}\ f(\mathrm{form}_2), \mathrm{form} \rightarrow \{f = \lambda \mathrm{form}_1.e : \mathrm{et}\} \cup \rho} \tag{2.151}$$

2.3.2 Expressions

Not:

$$\frac{\rho \vdash e \rightarrow e'}{\rho \vdash \textbf{not}\ e \rightarrow \textbf{not}\ e'} \tag{2.152}$$

$$\rho \vdash \textbf{not}\ \mathrm{tt} \rightarrow ff \tag{2.153}$$

$$\rho \vdash \textbf{not}\ ff \rightarrow \mathrm{tt} \tag{2.154}$$

Minus:

$$\frac{\rho \vdash e \rightarrow e'}{\rho \vdash -e \rightarrow -e'} \tag{2.155}$$

$$\rho \vdash -m \rightarrow n \qquad \text{where } n = -m. \tag{2.156}$$

Identifier:

$$\rho \vdash x \rightarrow \rho(x) \tag{2.157}$$

Binary operators:

$$\frac{\rho \vdash e_0 \to e_0'}{\rho \vdash e_0 \text{ bop } e_1 \to e_0' \text{ bop } e_1} \tag{2.158}$$

$$\frac{\rho \vdash e_1 \to e_1'}{\rho \vdash \text{con bop } e_1 \to \text{con bop } e_1'} \tag{2.159}$$

$$\rho \vdash \text{con}_1 \text{ bop con}_2 \to \text{con}, \qquad \text{where con} = \text{con}_1 \textbf{ bop } \text{con}_2. \tag{2.160}$$

Conditional expression:

$$\frac{\rho \vdash e \to e'}{\rho \vdash \textbf{if } e \textbf{ then } e_1 \textbf{ else } e_2 \to \textbf{if } e' \textbf{ then } e_1 \textbf{ else } e_2} \tag{2.161}$$

$$\rho \vdash \textbf{if tt then } e_1 \textbf{ else } e_2 \to e_1 \tag{2.162}$$

$$\rho \vdash \textbf{if } ff \textbf{ then } e_1 \textbf{ else } e_2 \to e_2 \tag{2.163}$$

Block:

$$\frac{\rho \vdash d \to d'}{\rho \vdash \textbf{let } d \textbf{ in } e \to \textbf{let } d' \textbf{ in } e} \tag{2.164}$$

$$\frac{\rho[\rho_0] \vdash e \to e' \qquad \rho_0 : \alpha_0}{\rho \vdash \textbf{let } \rho_0 \textbf{ in } e \to \textbf{let } \rho_0 \textbf{ in } e'} \tag{2.165}$$

$$\rho \vdash \textbf{let } \rho_0 \textbf{ in con} \to \text{con} \tag{2.166}$$

Function Call:

$$\rho \vdash f(\text{ae}) \to \textbf{let form} = \text{ae } \textbf{in } e \qquad \text{if } \{\rho(f) = \lambda\text{form}.e : \text{et}\}. \tag{2.167}$$

Input: If the user digits a representation of m,

$$\rho \vdash \textbf{input int} \to m; \tag{2.168}$$

if the user digits a representation of b,

$$\rho \vdash \textbf{input bool} \to b. \tag{2.169}$$

Chapter 3

The SIL Language

3.1 Abstract Syntax

3.1.1 Base Sets and Corresponding Syntactic Meta-Variables

Basic types: $t \in \text{Types} \stackrel{\text{def}}{=} \{\text{int}, \text{bool}\}$;

Integers: $m, n \in \text{Int} \stackrel{\text{def}}{=} \{\ldots, -2, -1, 0, 1, 2, \ldots\}$;

Booleans: $b \in T = \{\text{tt}, \text{ff}\}$;

Variables: $x, y \in \text{Var} \stackrel{\text{def}}{=} \{x_1, x_2, \ldots\}$;

Binary operators: $\text{bop} \in \text{Bop} \stackrel{\text{def}}{=} \{+, -, *, \text{div}, \text{mod}, ==, <>, <=, >=, <, >, \text{and}, \text{or}\}$.

3.1.2 Derived Sets:

General types: gt $\in$ GTypes

$$\text{gt} ::= t | (\text{lim}, t)\ \textbf{array} \tag{3.1}$$

where lim $\in$ Lim is defined by

$$\text{lim} ::= m..n | m..n, \text{lim} \tag{3.2}$$

Expressible types: et $\in$ ETypes

$$\text{et} ::= t \tag{3.3}$$

Type of lists of expressions: elt $\in$ ELTypes

$$\text{elt} ::= .|\text{et}, \text{elt} \tag{3.4}$$

Denotable types: dt $\in$ DTypes

$$\text{dt} ::= \text{gt}|\text{form} \rightarrow \text{et}|\text{form}\ \textbf{proc}\ |\text{et}\ \textbf{loc} \tag{3.5}$$

Constants: con $\in$ Con

$$\text{con} ::= m|b \tag{3.6}$$

Declarations: $d, d_0, d_1 \in$ Def

$$\begin{aligned} d ::= & \textbf{nil} \mid \textbf{const}\ x : t = e \mid \textbf{var}\ x : \text{gt} = e \\ & \mid d_0; d_1 \mid d_0\ \textbf{parallel}\ d_1 \mid d_0\ \textbf{private}\ d_1 \\ & \quad \mid \textbf{procedure}\ p(\text{form})c \mid \textbf{function}\ f(\text{form}) : \text{et} = e \mid \textbf{rec}\ d \end{aligned} \tag{3.7}$$

Expressions: $e, e_0, e_1 \in$ Exp

$$\begin{aligned} e ::= & \text{con} \mid x \mid x[el] \mid e_0\ \text{bop}\ e_1 \mid \textbf{not}\ e \mid -e \mid \textbf{if}\ e\ \textbf{then}\ e_0\ \textbf{else}\ e_1 \\ & \mid \textbf{input}\ \text{et} \mid f(\text{ae}) \mid \textbf{let}\ d\ \textbf{in}\ e \mid \textbf{expr}\ d; c\ \textbf{result}\ e \end{aligned} \tag{3.8}$$

Commands: $c, c_1, c_2 \in$ Com

$$\begin{aligned} c ::= & \textbf{nop} \mid x := e \mid x[el] := e \mid c_1; c_2 \mid \textbf{if}\ e\ \textbf{then}\ c_1\ \textbf{else}\ c_2 \\ & \mid \textbf{while}\ e\ \textbf{do}\ c \mid d; c \mid \textbf{print}\ el \mid p(\text{ae}) \end{aligned} \tag{3.9}$$

Lists of expressionsi: el $\in$ ExpList

$$\text{el} ::= .|e, \text{el} \tag{3.10}$$

Actual expressions: ae $\in$ ActExp

$$\text{ae} ::= .|e, \text{ae} \tag{3.11}$$

Formal parameters: form $\in$ Formals

$$\begin{aligned} \text{form} ::= & .|x : t, \text{form} \mid \textbf{name}\ x : t, \text{form} \mid \textbf{ref}\ x : t, \text{form} \\ & \mid \textbf{copy}\ x : t, \text{form} \mid \textbf{const}\ x : t, \text{form} \\ & \quad \mid \textbf{function}\ f(\text{form}) \to \text{et}, \text{form} \mid \textbf{procedure}\ p(\text{form}), \text{form} \end{aligned} \tag{3.12}$$

3.2 Static Semantics

Let $V \subseteq_{\text{f}} \text{Var}$ and $\text{TEnv}(V) \stackrel{\text{def}}{=} V \to \text{DTypes}$. Then

$$\alpha : V \stackrel{\text{def}}{\Longleftrightarrow} \alpha \in \text{TEnv}(V) \tag{3.13}$$

and

$$\text{TEnv} \stackrel{\text{def}}{=} \bigcup_{V \subseteq_{\text{f}} \text{Var}} \text{TEnv}(V). \tag{3.14}$$

For each $V \subseteq_{\mathrm{f}} \mathrm{Var}$ and each $\alpha : V$, we will define the following predicates:

$$\alpha \vdash \mathrm{con} : t, \qquad \text{where con} \in \mathrm{Con} \text{ and } t \in \mathrm{Types}; \tag{3.15}$$
$$\alpha \vdash e : \mathrm{et}, \qquad \text{where } e \in \mathrm{Exp} \text{ and et} \in \mathrm{ETypes}; \tag{3.16}$$
$$\alpha \vdash \mathrm{el} : \mathrm{elt}, \qquad \text{where el} \in \mathrm{ExpList} \text{ and elt} \in \mathrm{ELTypes}; \tag{3.17}$$
$$\alpha \vdash d, \qquad \text{where } d \in \mathrm{Def}; \tag{3.18}$$
$$\vdash d : \beta, \qquad \text{where } d \in \mathrm{Def} \text{ and } \beta \in \mathrm{TEnv}; \tag{3.19}$$
$$\alpha, \mathrm{form} \vdash \mathrm{ae}, \qquad \text{where form} \in \mathrm{Formals} \text{ and ae} \in \mathrm{ActExp}. \tag{3.20}$$

We will also define the usual polymorphic functions FV and DV and the auxiliary predicates

$$\mathrm{form} : \beta, \quad \text{with form} \in \mathrm{Formals}; \tag{3.21}$$
$$\mathrm{match}(\mathrm{form}_1, \mathrm{form}_2), \quad \text{with } \mathrm{form}_1, \mathrm{form}_2 \in \mathrm{Formals}; \tag{3.22}$$
$$\vdash \mathrm{lim}, \quad \text{with lim} \in \mathrm{Lim}; \tag{3.23}$$
$$\mathrm{ok}(\mathrm{lim}, \mathrm{elt}), \quad \text{with lim} \in \mathrm{Lim} \text{ and elt} \in \mathrm{ELTypes}. \tag{3.24}$$

3.2.1 Constants

$$\mathrm{FV}(m) \stackrel{\mathrm{def}}{=} \varnothing, \qquad \text{if } \alpha \vdash m : \mathrm{int}; \tag{3.25}$$
$$\mathrm{FV}(b) \stackrel{\mathrm{def}}{=} \varnothing, \qquad \text{if } \alpha \vdash b : \mathrm{bool}. \tag{3.26}$$

3.2.2 Expressions

Constants:

$$\mathrm{FV}(\mathrm{con}) \stackrel{\mathrm{def}}{=} \mathrm{FV}(\mathrm{con}), \qquad \text{(sic!)}; \tag{3.27}$$
$$\frac{\alpha \vdash \mathrm{con} : t}{\alpha \vdash \mathrm{con} : t} \qquad \text{(sic!)}. \tag{3.28}$$

Identifiers:

$$\mathrm{FV}(x) \stackrel{\mathrm{def}}{=} \{x\}, \tag{3.29}$$
$$\alpha \vdash x : \mathrm{et}, \tag{3.30}$$

if $\alpha(x) = \mathrm{et}$ or $\alpha(x) = \mathrm{et}$ **loc**.

Array references:

$$\mathrm{FV}(x[\mathrm{el}]) \stackrel{\mathrm{def}}{=} \{x\} \cup \mathrm{FV}(\mathrm{el}), \tag{3.31}$$
$$\frac{\alpha \vdash \mathrm{el} : \mathrm{elt} \qquad \alpha(x) = (\mathrm{lim}, \mathrm{et})\ \mathbf{array} \qquad \mathrm{ok}(\mathrm{lim}, \mathrm{elt})}{\alpha \vdash x[\mathrm{el}] : \mathrm{et}} \tag{3.32}$$

Predicate $\mathrm{ok}(\mathrm{lim}, \mathrm{elt})$**:** True if and only if elt has as many elements (all of type int) as lim.

$$\mathrm{ok}(.,.) \tag{3.33}$$
$$\frac{\mathrm{ok}(\mathrm{lim}, \mathrm{elt})}{\mathrm{ok}((n..m, \mathrm{lim}) : (\mathrm{int}, \mathrm{elt}))} \tag{3.34}$$

Boolean Negation:

$$\mathrm{FV}(\textbf{not}\ e) \stackrel{\text{def}}{=} \mathrm{FV}(e), \tag{3.35}$$

$$\frac{\alpha \vdash e : \mathrm{bool}}{\alpha \vdash \textbf{not}\ e : \mathrm{bool}} \tag{3.36}$$

Unary Minus:

$$\mathrm{FV}(-e) \stackrel{\text{def}}{=} \mathrm{FV}(e), \tag{3.37}$$

$$\frac{\alpha \vdash e : \mathrm{int}}{\alpha \vdash -e : \mathrm{int}} \tag{3.38}$$

Expression Block:

$$\mathrm{FV}(\textbf{let}\ d\ \textbf{in}\ e) \stackrel{\text{def}}{=} \mathrm{FV}(e) \setminus \mathrm{DV}(d), \tag{3.39}$$

$$\frac{\alpha \vdash d \qquad \vdash d : \beta \qquad \alpha[\beta] \vdash e : \mathrm{et}}{\alpha \vdash \textbf{let}\ d\ \textbf{in}\ e : \mathrm{et}}\ \text{if } \beta : W. \tag{3.40}$$

Conditional Expression:

$$\mathrm{FV}(\textbf{if}\ e\ \textbf{then}\ e_1\ \textbf{else}\ e_2) \stackrel{\text{def}}{=} \mathrm{FV}(e) \cup \mathrm{FV}(e_1) \cup \mathrm{FV}(e_2) \tag{3.41}$$

$$\frac{\alpha \vdash e : \mathrm{bool}, \alpha \vdash e_1 : \mathrm{et}, \alpha \vdash e_2 : \mathrm{et}}{\alpha \vdash \textbf{if}\ e\ \textbf{then}\ e_1\ \textbf{else}\ e_2 : \mathrm{et}} \tag{3.42}$$

Function Call:

$$\mathrm{FV}(f(\mathrm{ae})) \stackrel{\text{def}}{=} \{f\} \cup \mathrm{FV}(\mathrm{ae}), \tag{3.43}$$

$$\frac{\alpha, \mathrm{form} \vdash \mathrm{ae}}{\alpha \vdash f(\mathrm{ae}) : \mathrm{et}}\ \text{if } \alpha(f) = \mathrm{form} \to \mathrm{et}. \tag{3.44}$$

Input:

$$\mathrm{FV}(\textbf{input}\ \mathrm{et}) \stackrel{\text{def}}{=} \varnothing, \tag{3.45}$$

$$\alpha \vdash \textbf{input}\ \mathrm{et} : \mathrm{et}. \tag{3.46}$$

Expressions with Side-Effects:

$$\mathrm{FV}(\textbf{expr}\ d; c\ \textbf{result}\ e) \stackrel{\text{def}}{=} \big(\mathrm{FV}(c) \cup \mathrm{FV}(e)\big) \setminus \mathrm{DV}(d), \tag{3.47}$$

$$\frac{\alpha \vdash d, \vdash d : \beta, \alpha[\beta] \vdash c, \alpha[\beta] \vdash e : et}{\alpha \vdash \textbf{expr}\ d; c\ \textbf{result}\ e : \mathrm{et}}\ \text{if } \beta : W. \tag{3.48}$$

Binary Operators: If $\mathrm{bop} \in \{+, -, *, div, mod\}$,

$$t_1\ \mathrm{bop}\ t_2 = \mathrm{int} \iff t_1 = \mathrm{int}, t_2 = \mathrm{int}; \tag{3.49}$$

if bop $\in \{==, <>, <=, >=, <, >\}$

$$t_1 \text{ bop } t_2 = \text{bool} \iff t_1 = \text{int}, t_2 = \text{int}; \tag{3.50}$$

if bop $\in \{and, or\}$

$$t_1 \text{ bop } t_2 = \text{bool} \iff t_1 = \text{bool}, t_2 = \text{bool}. \tag{3.51}$$

With this definition we define

$$\text{FV}(e_1 \text{ bop } e_2) \stackrel{\text{def}}{=} \text{FV}(e_1) \cup \text{FV}(e_2), \tag{3.52}$$

$$\frac{\alpha \vdash e_1 : \text{et}_1, \alpha \vdash e_2 : \text{et}_2}{\alpha \vdash e_1 \text{ bop } e_2 : \text{et}} \text{ if et} = \text{et}_1 \text{ bop et}_2. \tag{3.53}$$

Actual Expressions

$$\text{FV}(.) \stackrel{\text{def}}{=} \varnothing, \tag{3.54}$$

$$\text{FV}(e, \text{ae}) \stackrel{\text{def}}{=} \text{FV}(e) \cup \text{FV}(\text{ae}). \tag{3.55}$$

Lists of Expressions

$$\text{FV}(.) \stackrel{\text{def}}{=} \varnothing, \tag{3.56}$$

$$\text{FV}(e, \text{el}) \stackrel{\text{def}}{=} \text{FV}(e) \cup \text{FV}(\text{el}). \tag{3.57}$$

$$\alpha \vdash . : ., \tag{3.58}$$

$$\frac{\alpha \vdash e : \text{et}, \alpha \vdash \text{el} : \text{elt}}{\alpha \vdash e, \text{el} : \text{et}, \text{elt}} \tag{3.59}$$

3.2.3 Declarations

Nil:

$$\text{FV}(\textbf{nil}) \stackrel{\text{def}}{=} \varnothing, \tag{3.60}$$

$$\text{DV}(\textbf{nil}) \stackrel{\text{def}}{=} \varnothing, \tag{3.61}$$

$$\alpha \vdash \textbf{nil}, \tag{3.62}$$

$$\vdash \textbf{nil}: \varnothing. \tag{3.63}$$

Constant Declaration:

$$\text{FV}(\textbf{const } x : \text{et} = e) \stackrel{\text{def}}{=} \text{FV}(e), \tag{3.64}$$

$$\text{DV}(\textbf{const } x : \text{et} = e) \stackrel{\text{def}}{=} \{x\}, \tag{3.65}$$

$$\frac{\alpha \vdash e : \text{et}}{\alpha \vdash \textbf{const } x : \text{et} = e} \tag{3.66}$$

$$\vdash \textbf{const } x : \text{et} = e : \{x = \text{et}\}. \tag{3.67}$$

Simple Variable Declaration:

$$\mathrm{FV}(\textbf{var}\ x : \mathrm{et} = e) \stackrel{\mathrm{def}}{=} \mathrm{FV}(e), \tag{3.68}$$

$$\mathrm{DV}(\textbf{var}\ x : \mathrm{et} = e) \stackrel{\mathrm{def}}{=} \{x\}, \tag{3.69}$$

$$\frac{\alpha \vdash e : \mathrm{et}}{\alpha \vdash \textbf{var}\ x : \mathrm{et} = e} \tag{3.70}$$

$$\vdash \textbf{var}\ x : \mathrm{et} = e : \{x = \mathrm{et}\ \textbf{loc}\}. \tag{3.71}$$

Array Declaration:

$$\mathrm{FV}(\textbf{var}\ x : (\mathrm{lim}, t)\ \textbf{array}= e) \stackrel{\mathrm{def}}{=} \mathrm{FV}(e), \tag{3.72}$$

$$\mathrm{DV}(\textbf{var}\ x : (\mathrm{lim}, t)\ \textbf{array}= e) \stackrel{\mathrm{def}}{=} \{x\}, \tag{3.73}$$

$$\frac{\alpha \vdash e : \mathrm{et} \qquad \vdash \mathrm{lim}}{\alpha \vdash \textbf{var}\ x : (\mathrm{lim}, \mathrm{et})\ \textbf{array}= e} \tag{3.74}$$

$$\vdash \textbf{var}\ x : (\mathrm{lim}, t)\ \textbf{array}= e : \{x = (\mathrm{lim}, \mathrm{et})\ \textbf{array}\}. \tag{3.75}$$

Predicate $\vdash \mathrm{lim}$**:** True if and only if lim is legal.

$$\vdash ., \tag{3.76}$$

$$\frac{\vdash \mathrm{lim}}{\vdash n..m, \mathrm{lim}}\ \text{if } n \leq m. \tag{3.77}$$

Function Declaration:

$$\mathrm{FV}(\textbf{function}\ f(\mathrm{form}) : \mathrm{et} = e) \stackrel{\mathrm{def}}{=} \mathrm{FV}(e) \setminus \mathrm{DV}(\mathrm{form}), \tag{3.78}$$

$$\mathrm{DV}(\textbf{function}\ f(\mathrm{form}) : \mathrm{et} = e) \stackrel{\mathrm{def}}{=} \{f\} \tag{3.79}$$

$$\frac{\mathrm{form} : \beta, \alpha[\beta] \vdash e : \mathrm{et}}{\alpha \vdash \textbf{function}\ f(\mathrm{form}) : \mathrm{et} = e}\ \text{if } \beta : W, \tag{3.80}$$

$$\vdash \textbf{function}\ f(\mathrm{form}) : \mathrm{et} = e : \{f = \mathrm{form} \to \mathrm{et}\}. \tag{3.81}$$

Procedure Declaration:

$$\mathrm{FV}(\textbf{procedure}\ p(\mathrm{form})c) \stackrel{\mathrm{def}}{=} \mathrm{FV}(c) \setminus \mathrm{DV}(\mathrm{form}), \tag{3.82}$$

$$\mathrm{DV}(\textbf{procedure}\ p(\mathrm{form})c) \stackrel{\mathrm{def}}{=} \{p\} \tag{3.83}$$

$$\frac{\mathrm{form} : \beta, \alpha[\beta] \vdash c}{\alpha \vdash \textbf{procedure}\ p(\mathrm{form})c}\ \text{if } \beta : W, \tag{3.84}$$

$$\vdash \textbf{procedure}\ p(\mathrm{form})c : \{p = \mathrm{form}\ \textbf{proc}\}. \tag{3.85}$$

Sequential Composition:

$$\mathrm{FV}(d_1; d_2) \stackrel{\mathrm{def}}{=} \mathrm{FV}(d_1) \cup (\mathrm{FV}(d_2) \setminus \mathrm{DV}(d_1)), \tag{3.86}$$

$$\mathrm{DV}(d_1; d_2) \stackrel{\mathrm{def}}{=} \mathrm{DV}(d_1) \cup \mathrm{DV}(d_2), \tag{3.87}$$

$$\frac{\alpha \vdash d_1, \vdash d_1 : \beta, \alpha[\beta] \vdash d_2}{\alpha \vdash d_1; d_2} \text{ if } \beta : W, \tag{3.88}$$

$$\frac{\vdash d_1 : \alpha, \vdash d_2 : \beta}{\vdash d_1; d_2 : \alpha[\beta]} \tag{3.89}$$

Parallel Composition:

$$\mathrm{FV}(d_1 \textbf{ parallel } d_2) \stackrel{\mathrm{def}}{=} \mathrm{FV}(d_1) \cup \mathrm{FV}(d_2), \tag{3.90}$$

$$\mathrm{DV}(d_1 \textbf{ parallel } d_2) \stackrel{\mathrm{def}}{=} \mathrm{DV}(d_1) \cup \mathrm{DV}(d_2), \tag{3.91}$$

$$\frac{\alpha \vdash d_1, \alpha \vdash d_2}{\alpha \vdash d_1 \textbf{ parallel } d_2} \text{ if } \mathrm{DV}(d_1) \cap \mathrm{DV}(d_2) = \varnothing, \tag{3.92}$$

$$\frac{\vdash d_1 : \alpha, \vdash d_2 : \beta}{\vdash d_1 \textbf{ parallel } d_2 : \alpha \cup \beta} \tag{3.93}$$

Private Composition:

$$\mathrm{FV}(d_1 \textbf{ private } d_2) \stackrel{\mathrm{def}}{=} \mathrm{FV}(d_1) \cup (\mathrm{FV}(d_2) \setminus \mathrm{DV}(d_1)), \tag{3.94}$$

$$\mathrm{DV}(d_1 \textbf{ private } d_2) \stackrel{\mathrm{def}}{=} \mathrm{DV}(d_1) \cup \mathrm{DV}(d_2), \tag{3.95}$$

$$\frac{\alpha \vdash d_1, \vdash d_1 : \beta, \alpha[\beta] \vdash d_2}{\alpha \vdash d_1 \textbf{ private } d_2} \text{ if } \beta : W, \tag{3.96}$$

$$\frac{\vdash d_1 : \alpha, \vdash d_2 : \beta}{\vdash d_1 \textbf{ private } d_2 : \beta} \tag{3.97}$$

Recursive Declaration:

$$\mathrm{FV}(\textbf{rec } d) \stackrel{\mathrm{def}}{=} \mathrm{FV}(d) \setminus \mathrm{DV}(d), \tag{3.98}$$

$$\mathrm{DV}(\textbf{rec } d) \stackrel{\mathrm{def}}{=} \mathrm{DV}(d), \tag{3.99}$$

$$\frac{\vdash d : \beta, \alpha[\beta \P W] \vdash d}{\alpha \vdash \textbf{rec } d} \text{ if } W = \mathrm{FV}(d) \cap \mathrm{DV}(d), \tag{3.100}$$

$$\frac{\vdash d : \beta}{\vdash \textbf{rec } d : \beta} \tag{3.101}$$

Formal Parameters:

Defined Variables:

$$\mathrm{DV}(.) \stackrel{\text{def}}{=} \varnothing, \tag{3.102}$$

$$\mathrm{DV}(x : \mathrm{et}, \mathrm{form}) \stackrel{\text{def}}{=} \{x\} \cup \mathrm{DV}(\mathrm{form}), \tag{3.103}$$

$$\mathrm{DV}(\textbf{name}\ x : \mathrm{et}, \mathrm{form}) \stackrel{\text{def}}{=} \{x\} \cup \mathrm{DV}(\mathrm{form}), \tag{3.104}$$

$$\mathrm{DV}(\textbf{ref}\ x : \mathrm{et}, \mathrm{form}) \stackrel{\text{def}}{=} \{x\} \cup \mathrm{DV}(\mathrm{form}), \tag{3.105}$$

$$\mathrm{DV}(\textbf{const}\ x : \mathrm{et}, \mathrm{form}) \stackrel{\text{def}}{=} \{x\} \cup \mathrm{DV}(\mathrm{form}), \tag{3.106}$$

$$\mathrm{DV}(\textbf{copy}\colon \mathrm{et}, \mathrm{form}) \stackrel{\text{def}}{=} \{x\} \cup \mathrm{DV}(\mathrm{form}), \tag{3.107}$$

$$\mathrm{DV}\big(\textbf{function}\ f(\mathrm{form}_1) : \mathrm{et}, \mathrm{form}\big) \stackrel{\text{def}}{=} \{f\} \cup \mathrm{DV}(\mathrm{form}), \tag{3.108}$$

$$\mathrm{DV}\big(\textbf{procedure}\ p(\mathrm{form}_1), \mathrm{form}\big) \stackrel{\text{def}}{=} \{p\} \cup \mathrm{DV}(\mathrm{form}). \tag{3.109}$$

Predicate $\mathrm{form} : \beta$**:**

$$. : \varnothing, \tag{3.110}$$

$$\frac{\mathrm{form} : \beta}{(x : \mathrm{et}, \mathrm{form}) : \{x = \mathrm{et}\} \cup \beta}\ \text{if } x \notin \mathrm{DV}(\mathrm{form}), \tag{3.111}$$

$$\frac{\mathrm{form} : \beta}{(\textbf{name}\ x : \mathrm{et}, \mathrm{form}) : \{x = \mathrm{et}\} \cup \beta}\ \text{if } x \notin \mathrm{DV}(\mathrm{form}), \tag{3.112}$$

$$\frac{\mathrm{form} : \beta}{(\textbf{ref}\ x : \mathrm{et}, \mathrm{form}) : \{x = \mathrm{et}\} \cup \beta}\ \text{if } x \notin \mathrm{DV}(\mathrm{form}), \tag{3.113}$$

$$\frac{\mathrm{form} : \beta}{(\textbf{const}\ x : \mathrm{et}, \mathrm{form}) : \{x = \mathrm{et}\} \cup \beta}\ \text{if } x \notin \mathrm{DV}(\mathrm{form}), \tag{3.114}$$

$$\frac{\mathrm{form} : \beta}{(\textbf{copy}\ x : \mathrm{et}, \mathrm{form}) : \{x = \mathrm{et}\} \cup \beta}\ \text{if } x \notin \mathrm{DV}(\mathrm{form}), \tag{3.115}$$

$$\frac{\mathrm{form} : \beta}{(\textbf{function}\ f(\mathrm{form}_1) : \mathrm{et}, \mathrm{form}) : \{f = \mathrm{form}_1 \to \mathrm{et}\} \cup \beta}\ \text{if } f \notin \mathrm{DV}(\mathrm{form}), \tag{3.116}$$

$$\frac{\mathrm{form} : \beta}{(\textbf{procedure}\ p(\mathrm{form}_1), \mathrm{form}) : \{p = \mathrm{form}_1\ \textbf{proc}\} \cup \beta}\ \text{if } p \notin \mathrm{DV}(\mathrm{form}). \tag{3.117}$$

Predicate $\alpha, \mathrm{form} \vdash \mathrm{ae}$**:** This predicate defines legality of an actual expression with respect to an actual expression and a type environment.

$$\alpha, . \vdash . \tag{3.118}$$

Call by value:

$$\frac{\alpha, \mathrm{form} \vdash \mathrm{ae}, \alpha \vdash e : \mathrm{et}}{\alpha, (x : \mathrm{et}, \mathrm{form}) \vdash e, \mathrm{ae}} \tag{3.119}$$

Call by name:

$$\frac{\alpha, \text{form} \vdash \text{ae}, \alpha \vdash e : \text{et}}{\alpha, (\textbf{name}\ x : \text{et}, \text{form}) \vdash e, \text{ae}} \tag{3.120}$$

Call by reference:

$$\frac{\alpha, \text{form} \vdash \text{ae}}{\alpha, (\textbf{ref}\ x : \text{et}, \text{form}) \vdash y, \text{ae}} \text{ if } \alpha(y) = \text{et}\ \textbf{loc}, \tag{3.121}$$

$$\frac{\alpha, \text{form} \vdash \text{ae}, \alpha \vdash y[\text{el}] : \text{et}}{\alpha, (\textbf{ref}\ x : \text{et}, \text{form}) \vdash y[\text{el}], \text{ae}} \text{ if } \alpha(y) = (\text{lim}, \text{et})\ \textbf{array}. \tag{3.122}$$

Call by constant:

$$\frac{\alpha, \text{form} \vdash \text{ae}, \alpha \vdash e : \text{et}}{\alpha, (\textbf{const}\ x : \text{et}, \text{form}) \vdash e, \text{ae}} \tag{3.123}$$

Call by value-result:

$$\frac{\alpha, \text{form} \vdash \text{ae}}{\alpha, (\textbf{copy}\ x : \text{et}, \text{form}) \vdash y, \text{ae}} \text{ if } \alpha(y) = \text{et}\ \textbf{loc}, \tag{3.124}$$

$$\frac{\alpha, \text{form} \vdash \text{ae}, \alpha \vdash y[\text{el}] : \text{et}}{\alpha, (\textbf{copy}\ x : \text{et}, \text{form}) \vdash y[\text{el}], \text{ae}} \text{ if } \alpha(y) = (\text{lim}, \text{et})\ \textbf{array}. \tag{3.125}$$

Functional parameters:

$$\frac{\text{match}(\text{form}_1, \text{form}_2), (\alpha, \text{form}) \vdash \text{ae}}{\alpha, (\textbf{function}\ f(\text{form}_1) : \text{et}, \text{form}) \vdash g, \text{ae}} \text{ if } \alpha(g) = \text{form}_2 \to \text{et}. \tag{3.126}$$

Procedural parameters:

$$\frac{\text{match}(\text{form}_1, \text{form}_2), (\alpha, \text{form}) \vdash \text{ae}}{\alpha, (\textbf{procedure}\ p(\text{form}_1), \text{form}) \vdash q, \text{ae}} \text{ if } \alpha(q) = \text{form}_2\ \textbf{proc}. \tag{3.127}$$

Predicate match(form, form)**:** True if the lists of formals coincide as far as the number, order, type and mechanisms of the formals contained.

$$\text{match}(.,.), \tag{3.128}$$

$$\frac{\text{match}(\text{form}_1, \text{form}_2)}{\text{match}((x : \text{et}, \text{form}_1), (y : \text{et}, \text{form}_2))} \tag{3.129}$$

$$\frac{\text{match}(\text{form}_1, \text{form}_2)}{\text{match}((\textbf{name}\ x : \text{et}, \text{form}_1), (\textbf{name}\ y : \text{et}, \text{form}_2))} \tag{3.130}$$

$$\frac{\text{match}(\text{form}_1, \text{form}_2)}{\text{match}((\textbf{ref}\ x : \text{et}, \text{form}_1), (\textbf{ref}\ y : \text{et}, \text{form}_2))} \tag{3.131}$$

$$\frac{\text{match}(\text{form}_1, \text{form}_2)}{\text{match}((\textbf{const}\ x : \text{et}, \text{form}_1), (\textbf{const}\ y : \text{et}, \text{form}_2))} \tag{3.132}$$

$$\frac{\text{match}(\text{form}_1, \text{form}_2)}{\text{match}((\textbf{copy}\ x : \text{et}, \text{form}_1), (\textbf{copy}\ y : \text{et}, \text{form}_2))} \tag{3.133}$$

$$\frac{\text{match}(f_1, f_2), \text{match}(f_3, f_4)}{\text{match}((\textbf{function}\ f(f_3) : \text{et}, f_1), (\textbf{function}\ g(f_4) : \text{et}, f_2))} \tag{3.134}$$

$$\frac{\text{match}(f_1, f_2), \text{match}(f_3, f_4)}{\text{match}((\textbf{procedure}\ p(f_3), f_1), (\textbf{procedure}\ q(f_4), f_2))} \tag{3.135}$$

3.2.4 Commands

Nop:

$$\text{FV}(\textbf{nop}) \stackrel{\text{def}}{=} \varnothing, \tag{3.136}$$

$$\alpha \vdash \textbf{nop}\ . \tag{3.137}$$

Assignment:

$$\text{FV}(x := e) \stackrel{\text{def}}{=} \{x\} \cup \text{FV}(e), \tag{3.138}$$

$$\text{FV}(x[\text{el}] := e) = \text{FV}(x[\text{el}]) \cup \text{FV}(e), \tag{3.139}$$

$$\frac{\alpha \vdash e : \text{et}}{\alpha \vdash x := e} \text{ if } \alpha(x) = \text{et}\ \textbf{loc}, \tag{3.140}$$

$$\frac{\alpha \vdash e : \text{et}, \alpha \vdash x[\text{el}] : \text{et}}{\alpha \vdash x[\text{el}] := e} \text{ if } \alpha(x) = (\text{lim}, \text{et})\ \textbf{array}. \tag{3.141}$$

Sequential composition:

$$\text{FV}(c_1; c_2) \stackrel{\text{def}}{=} \text{FV}(c_1) \cup \text{FV}(c_2), \tag{3.142}$$

$$\frac{\alpha \vdash c_1, \alpha \vdash c_2}{\alpha \vdash c_1; c_2} \tag{3.143}$$

If:

$$\mathrm{FV}(\textbf{if } e \textbf{ then } c_1 \textbf{ else } c_2) \stackrel{\text{def}}{=} \mathrm{FV}(e) \cup \mathrm{FV}(c_1) \cup \mathrm{FV}(c_2), \tag{3.144}$$

$$\frac{\alpha \vdash e : \mathrm{bool}, \alpha \vdash c_1, \alpha \vdash c_2}{\alpha \vdash \textbf{if } e \textbf{ then } c_1 \textbf{ else } c_2} \tag{3.145}$$

While:

$$\mathrm{FV}(\textbf{while } e \textbf{ do } c) \stackrel{\text{def}}{=} \mathrm{FV}(e) \cup \mathrm{FV}(c), \tag{3.146}$$

$$\frac{\alpha \vdash e : \mathrm{bool}, \alpha \vdash c}{\alpha \vdash \textbf{while } e \textbf{ do } c} \tag{3.147}$$

Block:

$$\mathrm{FV}(d; c) \stackrel{\text{def}}{=} \mathrm{FV}(c) \setminus \mathrm{DV}(d) \tag{3.148}$$

$$\frac{\alpha \vdash d, \vdash d : \beta, \alpha[\beta] \vdash c}{\alpha \vdash d; c} \text{ if } \beta : W. \tag{3.149}$$

Procedure call:

$$\mathrm{FV}(p(\mathrm{ae})) = \{p\} \cup \mathrm{FV}(\mathrm{ae}), \tag{3.150}$$

$$\frac{\alpha, \mathrm{form} \vdash \mathrm{ae}}{\alpha \vdash p(\mathrm{ae})} \text{ if } \alpha(p) = \mathrm{form}\ \textbf{proc}. \tag{3.151}$$

Print:

$$\mathrm{FV}(\textbf{print } \mathrm{el}) \stackrel{\text{def}}{=} \mathrm{FV}(\mathrm{el}), \tag{3.152}$$

$$\frac{\alpha \vdash \mathrm{el} : \mathrm{elt}}{\alpha \vdash \textbf{print } \mathrm{el}} \tag{3.153}$$

3.3 Dynamic Semantics

The set of locations Loc is given by the disjoint union

$$l \in \mathrm{Loc} \stackrel{\text{def}}{=} \mathrm{Loc}(\mathrm{int}) + \mathrm{Loc}(\mathrm{bool}) \tag{3.154}$$

where $\mathrm{Loc}(\mathrm{int}) \stackrel{\text{def}}{=} \mathrm{Loc}(\mathrm{bool}) \stackrel{\text{def}}{=} \mathbb{N}$. The function $\mathrm{NewLoc}\colon \mathrm{Types} \times \wp(\mathrm{Loc}) \to \mathrm{Loc}$ is defined, for each $L \subseteq_{\mathrm{f}} \mathrm{Loc}$, by

$$\mathrm{NewLoc}(t, L) \stackrel{\text{def}}{=} (\max L) + 1 \in \mathrm{Loc}(t). \tag{3.155}$$

This gives significant advantages in the treatment of arrays since, by providing consecutive memory cells, it allows laying out arrays by columns (or by rows), thereby simplifying the translation of multi-indexes into an offset with respect to the array's base address. To this purpose, we will refer to the function

$$\mathrm{offset}\colon \mathrm{Lim} \times \mathrm{ConL} \to \mathrm{Loc} \tag{3.156}$$

We can now define the set of stores

$$\text{Stores} \stackrel{\text{def}}{=} \left\{ \sigma \colon L \to \text{Con} \;\middle|\; \begin{array}{l} L \subseteq_{\text{f}} \text{Loc} \\ \forall l \in \text{Loc(int)} \cap L : \sigma(l) \in \text{Int} \\ \forall l \in \text{Loc(bool)} \cap L : \sigma(l) \in T \end{array} \right\} \tag{3.157}$$

and

$$\text{Abstracts} \stackrel{\text{def}}{=} \{\lambda\text{form}.e : \text{et}\} \cup \{\lambda\text{form}.c\}, \tag{3.158}$$

$$c \in \text{Copy} \stackrel{\text{def}}{=} \text{Loc} \times \text{Loc}, \tag{3.159}$$

$$a \in \text{Array} \stackrel{\text{def}}{=} \text{Lim} \times \text{Loc}, \tag{3.160}$$

$$\text{cl} \in \text{NExp} \stackrel{\text{def}}{=} \left\{ e \in \text{Exp} \;\middle|\; \text{FV}(e) = \varnothing \right\}, \tag{3.161}$$

and thus

$$\text{dval} \in \text{DVal} \stackrel{\text{def}}{=} \text{Con} + \text{Loc} + \text{Copy} + \text{Array} + \text{NExp} + \text{Abstracts}. \tag{3.162}$$

For each $V \subseteq_{\text{f}} \text{Var}$,

$$\text{Env}(V) \stackrel{\text{def}}{=} V \to (\text{DTypes} \times \text{DVal}), \tag{3.163}$$

$$\rho \in \text{Env} \stackrel{\text{def}}{=} \bigcup_{V \subseteq_{\text{f}} \text{Var}} \text{Env}(V). \tag{3.164}$$

Finally, if $\rho \in \text{Env}(V)$, then, for each $x \in V$,

$$\rho_t(x) \stackrel{\text{def}}{=} \pi_1\big(\rho(x)\big), \tag{3.165}$$

$$\rho_v(x) \stackrel{\text{def}}{=} \pi_2\big(\rho(x)\big), \tag{3.166}$$

$$\tag{3.167}$$

where π_1 and π_2 denote the projections over the first and second component, respectively.

We add the following productions to the abstract syntax of SIL:

$$\text{ae} ::= \lambda\text{form}.e : \text{et}, \text{ae}, \tag{3.168}$$

$$\text{ae} ::= \lambda\text{form}.c, \text{ae}, \tag{3.169}$$

$$\text{ae} ::= l, \text{ae}, \tag{3.170}$$

$$d ::= \text{form} = \text{ae}, \tag{3.171}$$

$$d ::= \rho, \tag{3.172}$$

$$c ::= \textbf{halt}\ . \tag{3.173}$$

Consequently, we define:

$$\text{FV}(\lambda\text{form}.e : \text{et}, \text{ae}) \stackrel{\text{def}}{=} (\text{FV}(e) \setminus \text{DV}(\text{form})) \cup \text{FV}(\text{ae}), \tag{3.174}$$

$$\text{FV}(\lambda\text{form}.c, \text{ae}) \stackrel{\text{def}}{=} (\text{FV}(c) \setminus \text{DV}(\text{form})) \cup \text{FV}(\text{ae}), \tag{3.175}$$

$$\text{FV}(l, \text{ae}) \stackrel{\text{def}}{=} \text{FV}(\text{ae}), \tag{3.176}$$

$$\text{DV}(\text{form} = \text{ae}) \stackrel{\text{def}}{=} \text{DV}(\text{form}), \tag{3.177}$$

$$\text{FV}(\text{form} = \text{ae}) \stackrel{\text{def}}{=} \text{FV}(\text{ae}), \tag{3.178}$$

and

$$\frac{\alpha, \mathrm{form} \vdash \mathrm{ae}}{\alpha \vdash \mathrm{form} = \mathrm{ae}} \tag{3.179}$$

$$\frac{\mathrm{form} : \beta}{\vdash \mathrm{form} = \mathrm{ae} : \beta} \tag{3.180}$$

If $\rho : V$, we define

$$\mathrm{DV}(\rho) \stackrel{\mathrm{def}}{=} V, \tag{3.181}$$

and

$$\mathrm{FV}(\rho) \stackrel{\mathrm{def}}{=} \bigcup_{x \in V} \mathrm{FV}\big(\rho_v(x)\big), \tag{3.182}$$

where FV(dval) is defined as follows:

$$\mathrm{FV}(\mathrm{con}) \stackrel{\mathrm{def}}{=} \varnothing, \tag{3.183}$$

$$\mathrm{FV}(l) \stackrel{\mathrm{def}}{=} \varnothing, \tag{3.184}$$

$$\mathrm{FV}(\mathrm{cl}) \stackrel{\mathrm{def}}{=} \varnothing, \tag{3.185}$$

$$\mathrm{FV}\big((l_1, l_2)\big) \stackrel{\mathrm{def}}{=} \varnothing, \tag{3.186}$$

$$\mathrm{FV}\big((\mathrm{lim}, l)\big) \stackrel{\mathrm{def}}{=} \varnothing, \tag{3.187}$$

$$\mathrm{FV}(\lambda \mathrm{form}.e : \mathrm{et}) \stackrel{\mathrm{def}}{=} \mathrm{FV}(e) \setminus \mathrm{DV}(\mathrm{form}), \tag{3.188}$$

$$\mathrm{FV}(\lambda \mathrm{form}.c) \stackrel{\mathrm{def}}{=} \mathrm{FV}(c) \setminus \mathrm{DV}(\mathrm{form}). \tag{3.189}$$

Finally, we define

$$\mathrm{FV}(\mathbf{halt}) \stackrel{\mathrm{def}}{=} \varnothing. \tag{3.190}$$

We are now in position to define the predicates $\vdash \mathrm{dval} : \mathrm{dt}$ and $\alpha \vdash \mathrm{dval}$.

Constants:

$$\vdash m : \mathrm{int}, \qquad \alpha \vdash m, \tag{3.191}$$

$$\vdash b : \mathrm{bool}, \qquad \alpha \vdash b. \tag{3.192}$$

Locations:

$$\vdash l : t\ \mathbf{loc}, \qquad \text{if } l \in \mathrm{Loc}(t); \tag{3.193}$$

$$\alpha \vdash l \tag{3.194}$$

Abstracts:

$$\vdash \lambda\text{form}.e : \text{et} : \text{form} \to \text{et} \tag{3.195}$$

$$\frac{\text{form} : \beta, \alpha[\beta] \vdash e : \text{et}}{\alpha \vdash \lambda\text{form}.e : \text{et}} \text{ if } \beta : W, \tag{3.196}$$

$$\vdash \lambda\text{form}.c : \text{form } \mathbf{proc} \tag{3.197}$$

$$\frac{\text{form} : \beta, \alpha[\beta] \vdash c}{\alpha \vdash \lambda\text{form}.c} \text{ if } \beta : W. \tag{3.198}$$

Value-result parameters:

$$\vdash (l_1, l_2) : t \ \mathbf{loc}, \qquad \text{if } l_1, l_2 \in \text{Loc}(t); \tag{3.199}$$

$$\alpha \vdash (l_1, l_2). \tag{3.200}$$

Arrays:

$$\vdash (\text{lim}, l) : (\text{lim}, t) \ \mathbf{array}, \tag{3.201}$$

if, for each multi-index I respecting lim, we have $l + \text{offset}(\text{lim}, I) \in \text{Loc}(t)$;

$$\alpha \vdash (\text{lim}, l). \tag{3.202}$$

Name expressions:

$$\frac{\varnothing \vdash \text{cl} : \text{et}}{\vdash \text{cl} : \text{et}} \tag{3.203}$$

$$\alpha \vdash \text{cl}. \tag{3.204}$$

We can now define the rules for environments, for each $\rho : V$:

$$\frac{\forall x \in V : \vdash \rho_v(x) : \beta(x)}{\vdash \rho : \beta} \tag{3.205}$$

$$\frac{\forall x \in V : \alpha \vdash \rho_v(x)}{\alpha \vdash \rho} \tag{3.206}$$

Let the sets Acon and ConL be defined as follows:

$$\text{acon} \in \text{Acon}, \tag{3.207}$$

where

$$\text{acon} ::= .|\text{con}, \text{acon}|\lambda\text{form}.e : \text{et}, \text{acon}|\lambda\text{form}.c, \text{acon}|l, \text{acon}|\text{cl}, \text{acon}; \tag{3.208}$$

$$\text{conl} \in \text{ConL}, \tag{3.209}$$

where

$$\text{conl} ::= .|\text{con}, \text{conl}. \tag{3.210}$$

The configurations and relations of the transition system are as follows, for each $\alpha \in \mathrm{TEnv}$ and implicitly assuming $\sigma \in \mathrm{Stores}$:

$$\Gamma(\alpha, E) \stackrel{\text{def}}{=} \{\, \langle e, \sigma\rangle \mid e \in \mathrm{Exp}, \exists \mathrm{et} \in \mathrm{ETypes} \,.\, \alpha \vdash e : \mathrm{et} \,\}, \tag{3.211}$$

$$T(\alpha, E) \stackrel{\text{def}}{=} \{\langle \mathrm{con}, \sigma\rangle\}, \tag{3.212}$$

$$\rho \vdash \langle e, \sigma\rangle \to \langle e', \sigma'\rangle, \tag{3.213}$$

$$\Gamma(\alpha, \mathrm{Ae}) \stackrel{\text{def}}{=} \{\, \langle \mathrm{ae}, \sigma\rangle \mid \mathrm{ae} \in \mathrm{ActExp}, \exists \mathrm{form} \in \mathrm{Formals} \,.\, \alpha, \mathrm{form} \vdash \mathrm{ae} \,\}, \tag{3.214}$$

$$T(\alpha, \mathrm{Ae}) \stackrel{\text{def}}{=} \{\langle \mathit{acon}, \sigma\rangle\}, \tag{3.215}$$

$$\rho, \mathrm{form} \vdash \langle \mathrm{ae}, \sigma\rangle \to \langle \mathrm{ae}', \sigma'\rangle, \tag{3.216}$$

$$\Gamma(\alpha, \mathrm{El}) \stackrel{\text{def}}{=} \{\, \langle \mathrm{el}, \sigma\rangle \mid \mathrm{el} \in \mathrm{ExpList}, \exists \mathrm{elt} \in \mathrm{ELTypes} \,.\, \alpha \vdash \mathrm{el} : \mathrm{elt} \,\}, \tag{3.217}$$

$$T(\alpha, \mathrm{El}) \stackrel{\text{def}}{=} \{\, \langle \mathrm{conl}, \sigma\rangle\}, \tag{3.218}$$

$$\rho \vdash \langle \mathrm{el}, \sigma\rangle \to \langle \mathrm{el}', \sigma'\rangle, \tag{3.219}$$

$$\Gamma(\alpha, D) \stackrel{\text{def}}{=} \{\, \langle d, \sigma\rangle \mid d \in \mathrm{Def}, \alpha \vdash d \,\}, \tag{3.220}$$

$$T(\alpha, D) \stackrel{\text{def}}{=} \{\, \langle \rho, \sigma\rangle \mid \rho \in \mathrm{Env}, \alpha \vdash \rho \,\}, \tag{3.221}$$

$$\rho \vdash \langle d, \sigma\rangle \to \langle d', \sigma'\rangle, \tag{3.222}$$

$$\Gamma(\alpha, C) \stackrel{\text{def}}{=} \{\, \langle c, \sigma\rangle \mid c \in \mathrm{Com}, \alpha \vdash c \,\}, \tag{3.223}$$

$$T(\alpha, C) \stackrel{\text{def}}{=} \{\langle \mathbf{halt}, \sigma\rangle\}, \tag{3.224}$$

$$\rho \vdash \langle c, \sigma\rangle \to \langle c', \sigma'\rangle. \tag{3.225}$$

We will also define the predicate

$$\mathrm{acon} \vdash \mathrm{form} \to \rho, \sigma. \tag{3.226}$$

3.3.1 Declarations

Nil:

$$\frac{}{\rho \vdash \langle \mathit{nil}, \sigma\rangle \to \langle \varnothing, \sigma\rangle.} \tag{3.227}$$

Constant:

$$\frac{\rho \vdash \langle e, \sigma\rangle \to \langle e', \sigma'\rangle,}{\rho \vdash \langle \textbf{const } x : \text{et} = e, \sigma\rangle \to \langle \textbf{const } x : \text{et} = e', \sigma'\rangle}, \tag{3.228}$$

$$\frac{}{\rho \vdash \langle \textbf{const } x : \text{et} = \text{con}, \sigma\rangle \to \langle \{x = \text{con}\}, \sigma\rangle}. \tag{3.229}$$

Variable:

$$\frac{\rho \vdash \langle e, \sigma\rangle \to \langle e', \sigma'\rangle}{\rho \vdash \langle \textbf{var } x : \text{et} = e, \sigma\rangle \to \langle \textbf{var } x : \text{et} = e', \sigma'\rangle}, \tag{3.230}$$

$$\frac{\sigma : L \qquad l = \text{NewLoc}(\text{et}, L)}{\rho \vdash \langle \textbf{var } x : \text{et} = \text{con}, \sigma\rangle \to \langle \{x = l\}, \sigma[l = \text{con}]\rangle}, \tag{3.231}$$

$$\frac{\rho \vdash \langle e, \sigma\rangle \to \langle e', \sigma'\rangle}{\rho \vdash \langle \textbf{var } x : (\text{lim}, \text{et}) \textbf{ array}= e, \sigma\rangle \to \langle \textbf{var } x : (\text{lim}, \text{et}) \textbf{ array}= e', \sigma'\rangle}, \tag{3.232}$$

$$\frac{}{\rho \vdash \langle \textbf{var } x : (\text{lim}, \text{et}) \textbf{ array}= \text{con}, \sigma\rangle \to \langle \{x = (\text{lim}, \text{base})\}, \sigma \cup \sigma_0\rangle}, \tag{3.233}$$

where $\sigma_0 \stackrel{\text{def}}{=} \{\text{base} = \text{con}, \text{base}{+}1 = \text{con}, \ldots, \text{base}{+}k{-}1 = \text{con}\}$, k is the number of elements of the array, and, finally, $\sigma : L$ and $\sigma_0 : L_0$ implies $L \cap L_0 = \varnothing$.

Sequential composition:

$$\frac{\rho \vdash \langle d_0, \sigma\rangle \to \langle d_0', \sigma'\rangle}{\rho \vdash \langle d_0; d_1, \sigma\rangle \to \langle d_0'; d_1, \sigma'\rangle}, \tag{3.234}$$

$$\frac{\rho[\rho_0] \vdash \langle d_1, \sigma\rangle \to \langle d_1', \sigma'\rangle \qquad \rho_0 : \alpha_0}{\rho \vdash \langle \rho_0; d_1, \sigma\rangle \to \langle \rho_0; d_1', \sigma'\rangle}, \tag{3.235}$$

$$\frac{}{\rho \vdash \langle \rho_0; \rho_1, \sigma\rangle \to \langle \rho_0[\rho_1], \sigma\rangle}. \tag{3.236}$$

Private composition:

$$\frac{\rho \vdash \langle d_0, \sigma\rangle \to \langle d_0', \sigma'\rangle}{\rho \vdash \langle d_0 \textbf{ private } d_1, \sigma\rangle \to \langle d_0' \textbf{ private } d_1, \sigma'\rangle}, \tag{3.237}$$

$$\frac{\rho[\rho_0] \vdash \langle d_1, \sigma\rangle \to \langle d_1', \sigma'\rangle \qquad \rho_0 : \alpha_0}{\rho \vdash \langle \rho_0 \textbf{ private } d_1, \sigma\rangle \to \langle \rho_0 \textbf{ private } d_1', \sigma'\rangle}, \tag{3.238}$$

$$\frac{}{\rho \vdash \langle \rho_0 \textbf{ private } \rho_1, \sigma\rangle \to \langle \rho_1, \sigma\rangle}. \tag{3.239}$$

Parallel composition:

$$\frac{\rho \vdash \langle d_0, \sigma\rangle \to \langle d_0', \sigma'\rangle}{\rho \vdash \langle d_0 \textbf{ parallel } d_1, \sigma\rangle \to \langle d_0' \textbf{ parallel } d_1, \sigma'\rangle}, \tag{3.240}$$

$$\frac{\rho \vdash \langle d_1, \sigma\rangle \to \langle d_1', \sigma'\rangle}{\rho \vdash \langle \rho_0 \textbf{ parallel } d_1, \sigma\rangle \to \langle \rho_0 \textbf{ parallel } d_1', \sigma'\rangle}, \tag{3.241}$$

$$\frac{}{\rho \vdash \langle \rho_0 \textbf{ parallel } \rho_1, \sigma\rangle \to \langle \rho_0 \cup \rho_1, \sigma\rangle}. \tag{3.242}$$

Procedure declaration:

$$\frac{I = \mathrm{FV}(c) \setminus \mathrm{DV}(\mathrm{form})}{\rho \vdash \langle \textbf{procedure } p(\mathrm{form})c, \sigma\rangle \to \langle \{p = \lambda \mathrm{form}.\rho\P I; c\}, \sigma\rangle}. \tag{3.243}$$

Function declaration:

$$\frac{I = \mathrm{FV}(e) \setminus \mathrm{DV}(\mathrm{form})}{\rho \vdash \langle \textbf{function } f(\mathrm{form}) : \mathrm{et} = e, \sigma\rangle \to \langle \{f = \lambda \mathrm{form}. \textbf{ let } \rho\P I \textbf{ in } e : \mathrm{et}\}, \sigma\rangle}. \tag{3.244}$$

Recursive declaration: [1]

$$\frac{\rho \setminus \big(\mathrm{DV}(d) \cap \mathrm{FV}(d)\big) \vdash \langle d, \sigma\rangle \to \langle d', \sigma'\rangle}{\rho \vdash \langle \textbf{rec } d, \sigma\rangle \to \langle \textbf{rec } d', \sigma'\rangle}, \tag{3.245}$$

$$\begin{aligned}
\rho_1 = {} & \big\{\, x = (t, \mathrm{con}) \;\big|\; x = (t, \mathrm{con}) \in \rho_0 \,\big\} \cup \big\{\, x = (t, \mathrm{cl}) \;\big|\; x = (t, \mathrm{cl}) \in \rho_0 \,\big\} \\
& \cup \big\{\, x = (t, l) \;\big|\; x = (t, l) \in \rho_0 \,\big\} \cup \big\{\, x = (t, (l_1, l_2)) \;\big|\; x = (t, (l_1, l_2)) \in \rho_0 \,\big\} \\
& \cup \big\{\, x = (t, (\lim, \mathrm{base})) \;\big|\; x = (t, (\lim, \mathrm{base})) \in \rho_0 \,\big\} \\
& \cup \Big\{\, f = (t, \textbf{letrec } \rho \setminus \mathrm{DV}(\mathrm{form}) \textbf{ in } e) \;\Big|\; \big(f = (t, \lambda \mathrm{form}.e : \mathrm{et})\big) \in \rho_0 \,\Big\} \\
& \cup \Big\{\, p = (t, \textbf{rec } \rho \setminus \mathrm{DV}(\mathrm{form}); c) \;\Big|\; \big(p = (t, \lambda \mathrm{form}.c)\big) \in \rho_0 \,\Big\}
\end{aligned}$$

$$\frac{}{\rho \vdash \langle \textbf{rec } \rho_0, \sigma\rangle \to \langle \rho_1, \sigma\rangle} \tag{3.246}$$

Parameter passing:

$$\frac{\rho, \mathrm{form} \vdash \langle \mathrm{ae}, \sigma\rangle \to \langle \mathrm{ae}', \sigma'\rangle}{\rho \vdash \langle \mathrm{form} = \mathrm{ae}, \sigma\rangle \to \langle \mathrm{form} = \mathrm{ae}', \sigma'\rangle}, \tag{3.247}$$

$$\frac{\mathrm{acon} \vdash \mathrm{form} \to \rho_0, \sigma_0}{\rho \vdash \langle \mathrm{form} = \mathrm{acon}, \sigma\rangle \to \langle \rho_0, \sigma \cup \sigma_0\rangle}. \tag{3.248}$$

Predicate $\rho, \mathrm{form} \vdash \langle \mathrm{ae}, \sigma\rangle \to \langle \mathrm{ae}', \sigma'\rangle$:

$$\frac{}{., . \vdash \langle ., \sigma\rangle \to \langle ., \sigma\rangle}. \tag{3.249}$$

[1] **FIXME:** This rule must be checked very carefully.

Call by value:

$$\frac{\rho \vdash \langle e, \sigma\rangle \to \langle e', \sigma'\rangle}{\rho, (x : \mathrm{et}, \mathrm{form}) \vdash \langle (e, \mathrm{ae}), \sigma\rangle \to \langle (e', \mathrm{ae}), \sigma'\rangle}, \tag{3.250}$$

$$\frac{\rho, \mathrm{form} \vdash \langle \mathrm{ae}, \sigma\rangle \to \langle \mathrm{ae}', \sigma'\rangle}{\rho, (x : \mathrm{et}, \mathrm{form}) \vdash \langle (\mathrm{con}, \mathrm{ae}), \sigma\rangle \to \langle (\mathrm{con}, \mathrm{ae}'), \sigma'\rangle}. \tag{3.251}$$

Call by name:

$$\frac{V \stackrel{\mathrm{def}}{=} \mathrm{FV}(e) \neq \varnothing}{\rho, (\mathbf{name}\ x : \mathrm{et}, \mathrm{form}) \vdash \langle (e, \mathrm{ae}), \sigma\rangle \to \langle (\mathbf{let}\ \rho\P V\ \mathbf{in}\ e, \mathrm{ae}), \sigma\rangle}, \tag{3.252}$$

$$\frac{\rho, \mathrm{form} \vdash \langle \mathrm{ae}, \sigma\rangle \to \langle \mathrm{ae}', \sigma'\rangle}{\rho, (\mathbf{name}\ x : \mathrm{et}, \mathrm{form}) \vdash \langle (\mathrm{cl}, \mathrm{ae}), \sigma\rangle \to \langle (\mathrm{cl}, \mathrm{ae}'), \sigma'\rangle}. \tag{3.253}$$

Call by reference:

$$\frac{\rho, \mathrm{form} \vdash \langle \mathrm{ae}, \sigma\rangle \to \langle \mathrm{ae}', \sigma'\rangle}{\rho, (\mathbf{ref}\ x : \mathrm{et}, \mathrm{form}) \vdash \langle (l, \mathrm{ae}), \sigma\rangle \to \langle (l, \mathrm{ae}'), \sigma'\rangle}, \tag{3.254}$$

$$\frac{\rho_v(y) = l}{\rho, (\mathbf{ref}\ x : \mathrm{et}, \mathrm{form}) \vdash \langle (y, \mathrm{ae}), \sigma\rangle \to \langle (l, \mathrm{ae}), \sigma\rangle}, \tag{3.255}$$

$$\frac{\rho \vdash \langle \mathrm{el}, \sigma\rangle \to \langle \mathrm{el}', \sigma'\rangle}{\rho, (\mathbf{ref}\ x : \mathrm{et}, \mathrm{form}) \vdash \langle (y[\mathrm{el}], \mathrm{ae}), \sigma\rangle \to \langle (y[\mathrm{el}'], \mathrm{ae}), \sigma'\rangle}, \tag{3.256}$$

$$\frac{\rho_v(y) = (\mathrm{lim}, \mathrm{base}) \qquad l = \mathrm{base} + \mathrm{offset}(\mathrm{lim}, \mathrm{conl})}{\rho, (\mathbf{ref}\ x : \mathrm{et}, \mathrm{form}) \vdash \langle (y[\mathrm{conl}], \mathrm{ae}), \sigma\rangle \to \langle (l, \mathrm{ae}), \sigma\rangle} \tag{3.257}$$

Call by constant:

$$\frac{\rho \vdash \langle e, \sigma\rangle \to \langle e', \sigma'\rangle}{\rho, (\mathbf{const}\ x : \mathrm{et}, \mathrm{form}) \vdash \langle (e, \mathrm{ae}), \sigma\rangle \to \langle (e', \mathrm{ae}), \sigma'\rangle}, \tag{3.258}$$

$$\frac{\rho, \mathrm{form} \vdash \langle \mathrm{ae}, \sigma\rangle \to \langle \mathrm{ae}', \sigma'\rangle}{\rho, (\mathbf{const}\ x : \mathrm{et}, \mathrm{form}) \vdash \langle (\mathrm{con}, \mathrm{ae}), \sigma\rangle \to \langle (\mathrm{con}, \mathrm{ae}'), \sigma'\rangle}. \tag{3.259}$$

Call by value-result:

$$\frac{\rho, \mathrm{form} \vdash \langle \mathrm{ae}, \sigma\rangle \to \langle \mathrm{ae}', \sigma'\rangle}{\rho, (\mathbf{copy}\ x : \mathrm{et}, \mathrm{form}) \vdash \langle (l, \mathrm{ae}), \sigma\rangle \to \langle (l, \mathrm{ae}'), \sigma'\rangle}, \tag{3.260}$$

$$\frac{\rho_v(y) = l}{\rho, (\mathbf{copy}\ x : \mathrm{et}, \mathrm{form}) \vdash \langle (y, \mathrm{ae}), \sigma\rangle \to \langle (l, \mathrm{ae}), \sigma\rangle}, \tag{3.261}$$

$$\frac{\rho \vdash \langle \mathrm{el}, \sigma\rangle \to \langle \mathrm{el}', \sigma'\rangle}{\rho, (\mathbf{copy}\ x : \mathrm{et}, \mathrm{form}) \vdash \langle (y[\mathrm{el}], \mathrm{ae}), \sigma\rangle \to \langle (y[\mathrm{el}'], \mathrm{ae}), \sigma'\rangle}, \tag{3.262}$$

$$\frac{\rho_v(y) = (\mathrm{lim}, \mathrm{base}) \qquad l = \mathrm{base} + \mathrm{offset}(\mathrm{lim}, \mathrm{conl})}{\rho, (\mathbf{copy}\ x : \mathrm{et}, \mathrm{form}) \vdash \langle (y[\mathrm{conl}], \mathrm{ae}), \sigma\rangle \to \langle (l, \mathrm{ae}), \sigma\rangle}. \tag{3.263}$$

Functional parameter:

$$\frac{\rho_v(g) = \lambda \mathrm{form}_2.e : \mathrm{et}}{\rho, (\textbf{function}\ f(\mathrm{form}_1) : \mathrm{et}, \mathrm{form}) \vdash \langle (g, \mathrm{ae}), \sigma\rangle \to \langle (\lambda \mathrm{form}_2.e : \mathrm{et}, \mathrm{ae}), \sigma\rangle}, \tag{3.264}$$

$$\frac{\rho, \mathrm{form} \vdash \langle \mathrm{ae}, \sigma\rangle \to \langle \mathrm{ae}', \sigma'\rangle}{\rho, (\textbf{function}\ f(\mathrm{form}_1) : \mathrm{et}, \mathrm{form}) \vdash \langle (\lambda \mathrm{form}_2.e : \mathrm{et}, \mathrm{ae}), \sigma\rangle \to \langle (\lambda \mathrm{form}_2.e : \mathrm{et}, \mathrm{ae}'), \sigma'\rangle}. \tag{3.265}$$

Procedural parameter:

$$\frac{\rho_v(q) = \lambda \mathrm{form}_2.c}{\rho, (\textbf{procedure}\ p(\mathrm{form}_1), \mathrm{form}) \vdash \langle (q, \mathrm{ae}), \sigma\rangle \to \langle (\lambda \mathrm{form}_2.c, \mathrm{ae}), \sigma\rangle}, \tag{3.266}$$

$$\frac{\rho, \mathrm{form} \vdash \langle \mathrm{ae}, \sigma\rangle \to \langle \mathrm{ae}', \sigma'\rangle}{\rho, (\textbf{procedure}\ p(\mathrm{form}_1), \mathrm{form}) \vdash \langle (\lambda \mathrm{form}_2.c, \mathrm{ae}), \sigma\rangle \to \langle (\lambda \mathrm{form}_2.c, \mathrm{ae}'), \sigma'\rangle}. \tag{3.267}$$

Predicate $\mathrm{acon} \vdash \mathrm{form} \to \rho, \sigma$**:**

$$\frac{}{. \vdash . \to \varnothing, \varnothing}. \tag{3.268}$$

Call by value:

$$\frac{\mathrm{acon} \vdash \mathrm{form} \to \rho, \sigma \qquad \sigma : L \qquad l = \mathrm{NewLoc}(\mathrm{et}, L)}{\mathrm{con}, \mathrm{acon} \vdash x : \mathrm{et}, \mathrm{form} \to \{x = l\} \cup \rho, \sigma \cup \{l = \mathrm{con}\}}. \tag{3.269}$$

Call by name:

$$\frac{\mathrm{acon} \vdash \mathrm{form} \to \rho, \sigma}{\mathrm{cl}, \mathrm{acon} \vdash \textbf{name}\ x : \mathrm{et}, \mathrm{form} \to \{x = \mathrm{cl}\} \cup \rho, \sigma}. \tag{3.270}$$

Call by reference:

$$\frac{\mathrm{acon} \vdash \mathrm{form} \to \rho, \sigma}{l, \mathrm{acon} \vdash \textbf{ref}\ x : \mathrm{et}, \mathrm{form} \to \{x = l\} \cup \rho, \sigma}. \tag{3.271}$$

Call by constant:

$$\frac{\mathrm{acon} \vdash \mathrm{form} \to \rho, \sigma}{\mathrm{con}, \mathrm{acon} \vdash \textbf{const}\ x : \mathrm{et}, \mathrm{form} \to \{x = \mathrm{con}\} \cup \rho, \sigma}. \tag{3.272}$$

Call by value-result:

$$\frac{\mathrm{acon} \vdash \mathrm{form} \to \rho, \sigma \qquad \sigma : L \qquad l' = \mathrm{NewLoc}(\mathrm{et}, L)}{l, \mathrm{acon} \vdash \textbf{copy}\ x : \mathrm{et}, \mathrm{form} \to \{x = (l, l')\} \cup \rho, \sigma \cup \{l' = \sigma(l)\}}. \tag{3.273}$$

Functional parameter:

$$\frac{\text{acon} \vdash \text{form} \to \rho, \sigma}{\lambda\text{form}_1.e : \text{et}, \text{acon} \vdash \textbf{function}\ f(\text{form}_2), \text{form} \to \{f = \lambda\text{form}_1.e : \text{et}\} \cup \rho, \sigma}. \tag{3.274}$$

Procedural parameter:

$$\frac{\text{acon} \vdash \text{form} \to \rho, \sigma}{\lambda\text{form}_1.c, \text{acon} \vdash \textbf{procedure}\ p(\text{form}_2), \text{form} \to \{p = \lambda\text{form}_1.c\} \cup \rho, \sigma}. \tag{3.275}$$

3.3.2 Expressions

Identifier:

Constant:

$$\frac{\rho_v(x) = \text{con}}{\rho \vdash \langle x, \sigma\rangle \to \langle \text{con}, \sigma\rangle}. \tag{3.276}$$

Variable:

$$\frac{\rho_v(x) = l \qquad \sigma(l) = \text{con}}{\rho \vdash \langle x, \sigma\rangle \to \langle \text{con}, \sigma\rangle}. \tag{3.277}$$

Value-result parameter:

$$\frac{\rho_v(x) = (l_1, l_2) \qquad \sigma(l_2) = \text{con}}{\rho \vdash \langle x, \sigma\rangle \to \langle \text{con}, \sigma\rangle}. \tag{3.278}$$

Name:

$$\frac{\rho_v(x) = \text{cl}}{\rho \vdash \langle x, \sigma\rangle \to \langle \text{cl}, \sigma\rangle}. \tag{3.279}$$

Not:

$$\frac{\rho \vdash \langle e, \sigma\rangle \to \langle e', \sigma'\rangle}{\rho \vdash \langle \textbf{not}\ e, \sigma\rangle \to \langle \textbf{not}\ e', \sigma'\rangle}, \tag{3.280}$$

$$\frac{}{\rho \vdash \langle \textbf{not}\ \text{tt}, \sigma\rangle \to \langle \text{ff}, \sigma\rangle}, \tag{3.281}$$

$$\frac{}{\rho \vdash \langle \textbf{not}\ \text{ff}, \sigma\rangle \to \langle \text{tt}, \sigma\rangle}. \tag{3.282}$$

Minus.

$$\frac{\rho \vdash \langle e, \sigma\rangle \to \langle e', \sigma'\rangle}{\rho \vdash \langle -e, \sigma\rangle \to \langle -e', \sigma'\rangle}, \tag{3.283}$$

$$\frac{n = -m}{\rho \vdash \langle -m, \sigma\rangle \to \langle n, \sigma\rangle}. \tag{3.284}$$

Array reference:

$$\frac{\rho \vdash \langle \mathrm{el}, \sigma\rangle \to \langle \mathrm{el}', \sigma'\rangle}{\rho \vdash \langle x[\mathrm{el}], \sigma\rangle \to \langle x[\mathrm{el}'], \sigma'\rangle}, \tag{3.285}$$

$$\frac{\rho_v(x) = (\mathrm{lim}, base) \qquad \sigma(base + \mathrm{offset}(\mathrm{lim}, \mathrm{conl})) = \mathrm{con}}{\rho \vdash \langle x[\mathrm{conl}], \sigma\rangle \to \langle \mathrm{con}, \sigma\rangle}. \tag{3.286}$$

Binary operators:

$$\frac{\rho \vdash \langle e_0, \sigma\rangle \to \langle e_0', \sigma'\rangle}{\rho \vdash \langle e_0 \ \mathrm{bop}\ e_1, \sigma\rangle \to \langle e_0' \ \mathrm{bop}\ e_1, \sigma'\rangle}, \tag{3.287}$$

$$\frac{\rho \vdash \langle e_1, \sigma\rangle \to \langle e_1', \sigma'\rangle}{\rho \vdash \langle \mathrm{con} \ \mathrm{bop}\ e_1, \sigma\rangle \to \langle \mathrm{con} \ \mathrm{bop}\ e_1', \sigma'\rangle}, \tag{3.288}$$

$$\frac{\mathrm{con} = \mathrm{con}_1 \ \mathrm{bop}\ \mathrm{con}_2}{\rho \vdash \langle \mathrm{con}_1 \ \mathrm{bop}\ \mathrm{con}_2, \sigma\rangle \to \langle \mathrm{con}, \sigma\rangle}. \tag{3.289}$$

Conditional expression:

$$\frac{\rho \vdash \langle e, \sigma\rangle \to \langle e', \sigma'\rangle}{\rho \vdash \langle \textbf{if}\ e\ \textbf{then}\ e_1\ \textbf{else}\ e_2, \sigma\rangle \to \langle \textbf{if}\ e'\ \textbf{then}\ e_1\ \textbf{else}\ e_2, \sigma'\rangle}, \tag{3.290}$$

$$\frac{}{\rho \vdash \langle \textbf{if}\ \mathrm{tt}\ \textbf{then}\ e_1\ \textbf{else}\ e_2, \sigma\rangle \to \langle e_1, \sigma\rangle}, \tag{3.291}$$

$$\frac{}{\rho \vdash \langle \textbf{if}\ \mathrm{ff}\ \textbf{then}\ e_1\ \textbf{else}\ e_2, \sigma\rangle \to \langle e_2, \sigma\rangle}. \tag{3.292}$$

Input.

$$\frac{\text{the user digits } \mathrm{repr}(m)}{\rho \vdash \langle \textbf{input}\ \mathrm{int}, \sigma\rangle \to \langle m, \sigma\rangle}, \tag{3.293}$$

$$\frac{\text{the user digits } \mathrm{repr}(b)}{\rho \vdash \langle \textbf{input}\ \mathrm{bool}, \sigma\rangle \to \langle b, \sigma\rangle}. \tag{3.294}$$

Expressions with side-effects:

$$\frac{\rho \vdash \langle d, \sigma\rangle \to \langle d', \sigma'\rangle}{\rho \vdash \langle \textbf{expr } d; c \textbf{ result } e, \sigma\rangle \to \langle \textbf{expr } d'; c \textbf{ result } e, \sigma'\rangle}, \tag{3.295}$$

$$\frac{\rho[\rho_0] \vdash \langle c, \sigma\rangle \to \langle c', \sigma'\rangle}{\rho \vdash \langle \textbf{expr } \rho_0; c \textbf{ result } e, \sigma\rangle \to \langle \textbf{expr } \rho_0; c' \textbf{ result } e, \sigma'\rangle}, \tag{3.296}$$

$$\frac{\rho[\rho_0] \vdash \langle e, \sigma\rangle \to \langle e', \sigma'\rangle}{\rho \vdash \langle \textbf{expr } \rho_0; \textbf{haltresult } e, \sigma\rangle \to \langle \textbf{expr } \rho_0; \textbf{haltresult } e', \sigma'\rangle}, \tag{3.297}$$

$$\frac{}{\rho \vdash \langle \textbf{expr } \rho_0; \textbf{haltresult } m, \sigma\rangle \to \langle m, \sigma\rangle}. \tag{3.298}$$

Function call:

$$\frac{\rho_v(f) = \lambda \text{form}.e : \text{et}}{\rho \vdash \langle f(\text{ae}), \sigma\rangle \to \langle \textbf{let } \text{form} = \text{ae } \textbf{in } e, \sigma\rangle}. \tag{3.299}$$

Block:

$$\frac{\rho \vdash \langle d, \sigma\rangle \to \langle d', \sigma'\rangle}{\rho \vdash \langle \textbf{let } d \textbf{ in } e, \sigma\rangle \to \langle \textbf{let } d' \textbf{ in } e, \sigma'\rangle}, \tag{3.300}$$

$$\frac{\rho[\rho_0] \vdash \langle e, \sigma\rangle \to \langle e', \sigma'\rangle \qquad \rho_0 : \alpha_0}{\rho \vdash \langle \textbf{let } \rho_0 \textbf{ in } e, \sigma\rangle \to \langle \textbf{let } \rho_0 \textbf{ in } e', \sigma'\rangle}, \tag{3.301}$$

(the following realizes the final phase of parameter passing by value-result)

$$\frac{\forall (l_i', l_i'') \in \text{cod}(\rho_0) : \sigma' = \sigma\big[l_1' = \sigma(l_1''), ..., l_k' = \sigma(l_k'')\big]}{\rho \vdash \langle \textbf{let } \rho_0 \textbf{ in } m, \sigma\rangle \to \langle m, \sigma'\rangle}. \tag{3.302}$$

3.3.3 Lists of Expressions

$$\frac{}{\rho \vdash \langle ., \sigma\rangle \to \langle ., \sigma\rangle}, \tag{3.303}$$

$$\frac{\rho \vdash \langle e, \sigma\rangle \to \langle e', \sigma'\rangle}{\rho \vdash \langle (e, \text{el}), \sigma\rangle \to \langle (e', \text{el}), \sigma'\rangle}, \tag{3.304}$$

$$\frac{\rho \vdash \langle \text{el}, \sigma\rangle \to \langle \text{el}', \sigma'\rangle}{\rho \vdash \langle (\text{con}, \text{el}), \sigma\rangle \to \langle (\text{con}, \text{el}'), \sigma'\rangle}. \tag{3.305}$$

3.3.4 Commands

Nop:

$$\frac{}{\rho \vdash \langle \textbf{nop}, \sigma\rangle \to \langle \textbf{halt}, \sigma\rangle}. \tag{3.306}$$

Assignment:

$$\frac{\rho \vdash \langle e, \sigma\rangle \to \langle e', \sigma'\rangle}{\rho \vdash \langle x := e, \sigma\rangle \to \langle x := e', \sigma'\rangle}, \tag{3.307}$$

$$\frac{\rho_v(x) = l}{\rho \vdash \langle x := \mathrm{con}, \sigma\rangle \to \langle \mathbf{halt}, \sigma[l = \mathrm{con}]\rangle}, \tag{3.308}$$

$$\frac{\rho \vdash \langle \mathrm{el}, \sigma\rangle \to \langle \mathrm{el}', \sigma'\rangle}{\rho \vdash \langle x[\mathrm{el}] := e, \sigma\rangle \to \langle x[\mathrm{el}'] := e, \sigma'\rangle}, \tag{3.309}$$

$$\frac{\rho \vdash \langle e, \sigma\rangle \to \langle e', \sigma'\rangle}{\rho \vdash \langle x[\mathrm{conl}] := e, \sigma\rangle \to \langle x[\mathrm{conl}] := e', \sigma'\rangle}, \tag{3.310}$$

$$\frac{\rho_v(x) = (\mathrm{lim}, \mathrm{base}) \qquad l = \mathrm{base} + \mathrm{offset}(\mathrm{lim}, \mathrm{conl})}{\rho \vdash \langle x[\mathrm{conl}] := \mathrm{con}, \sigma\rangle \to \langle \mathbf{halt}, \sigma[l = \mathrm{con}]\rangle}. \tag{3.311}$$

Command sequence:

$$\frac{\rho \vdash \langle c_0, \sigma\rangle \to \langle c_0', \sigma'\rangle}{\rho \vdash \langle c_0; c_1, \sigma\rangle \to \langle c_0'; c_1, \sigma'\rangle}, \tag{3.312}$$

$$\frac{}{\rho \vdash \langle \mathbf{halt}; c_1, \sigma\rangle \to \langle c_1, \sigma\rangle}. \tag{3.313}$$

Conditional command:

$$\frac{\rho \vdash \langle e, \sigma\rangle \to \langle e', \sigma'\rangle}{\rho \vdash \langle \mathbf{if}\ e\ \mathbf{then}\ c_1\ \mathbf{else}\ c_2, \sigma\rangle \to \langle \mathbf{if}\ e'\ \mathbf{then}\ c_1\ \mathbf{else}\ c_2, \sigma'\rangle}, \tag{3.314}$$

$$\frac{}{\rho \vdash \langle \mathbf{if}\ \mathrm{tt}\ \mathbf{then}\ c_1\ \mathbf{else}\ c_2, \sigma\rangle \to \langle c_1, \sigma\rangle}, \tag{3.315}$$

$$\frac{}{\rho \vdash \langle \mathbf{if}\ \mathrm{ff}\ \mathbf{then}\ c_1\ \mathbf{else}\ c_2, \sigma\rangle \to \langle c_2, \sigma\rangle}. \tag{3.316}$$

While command:

$$\frac{}{\rho \vdash \langle \mathbf{while}\ e\ \mathbf{do}\ c, \sigma\rangle \to \langle \mathbf{if}\ e\ \mathbf{then}\ c; \mathbf{while}\ e\ \mathbf{do}\ c\ \mathbf{elsehalt}, \sigma\rangle}. \tag{3.317}$$

Block:

$$\frac{\rho \vdash \langle d, \sigma\rangle \to \langle d', \sigma'\rangle}{\rho \vdash \langle d; c, \sigma\rangle \to \langle d'; c, \sigma'\rangle}, \tag{3.318}$$

$$\frac{\rho[\rho_0] \vdash \langle c, \sigma\rangle \to \langle c', \sigma'\rangle \qquad \rho_0 : \alpha_0}{\rho \vdash \langle \rho_0; c, \sigma\rangle \to \langle \rho_0; c', \sigma'\rangle}, \tag{3.319}$$

(the following realizes the final phase of parameter passing by value-result)

$$\frac{\forall (l_i', l_i'') \in \mathrm{cod}(\rho_0) : \sigma' = \sigma\big[l_1' = \sigma(l_1''), ..., l_k' = \sigma(l_k'')\big]}{\rho \vdash \langle \rho_0; \mathbf{halt}, \sigma\rangle \to \langle \mathbf{halt}, \sigma'\rangle}. \tag{3.320}$$

Print:

$$\frac{\rho \vdash \langle \mathrm{el}, \sigma \rangle \to \langle \mathrm{el}', \sigma' \rangle}{\rho \vdash \langle \textbf{print } \mathrm{el}, \sigma \rangle \to \langle \textbf{print } \mathrm{el}', \sigma' \rangle}, \tag{3.321}$$

$$\frac{}{\rho \vdash \langle \textbf{print } \mathrm{conl}, \sigma \rangle \to \langle \textbf{halt}, \sigma \rangle}. \tag{3.322}$$

Procedure call:

$$\frac{\rho_v(p) = \lambda \mathrm{form}.c}{\rho \vdash \langle p(\mathrm{ae}), \sigma \rangle \to \langle \mathrm{form} = \mathrm{ae}; c, \sigma \rangle}. \tag{3.323}$$

Chapter 4

Concrete Syntax

In this chapter we describe the concrete syntax of the SFL and SIL languages.

4.1 Identifiers

An identifier is constituted by any sequence of letters, digits and the underscore character '_' whose first character is a letter or underscore.

Some identifiers are reserved as keywords of the language and, consequently, cannot be used to denote variables, constants, functions and so forth. The following identifiers are reserved both in SFL and SIL:

```
integer     boolean     input       name
rec         and         or          mod
div         if          then        else
true        false       not         function
```

The following identifiers are reserved only in SFL:

```
let         in
```

The following identifiers are reserved only in SIL:

```
const       var         array       of
begin       end         expr        result
ref         copy        while       do
for         to          downto      procedure
print
```

4.2 Operators

The priorities and the type of associativity of the arithmetic, logic and composition operators are summarized in Table 4.1.

4.3 The Grammar of SFL

This grammar, as well as the one for SIL given in the next section, serves the only purpose of showing the constructs that are recognized by CLAIR's parser.

Operator(s)	Priority	Associativity
`==` `<>` `<` `<=` `>=` `>`	0	left
`+` `-` `or`	1	left
`*` `div` `mod` `and`	2	left
`+` `-` (unary) `not`	3	right
`;`	0	left
`|`	1	left
`->`	2	left

Table 4.1: Priority and associativity of the operators.

Of course, the true grammar, which is implemented in the `parse` module, is very different, since it cannot admit left-recursion (the parser being *recursive–descent*), since it must impose the operators' priority and so on.

⟨*Progr*⟩ ::= ⟨*Expr*⟩

⟨*Type*⟩ ::= '`integer`'
| '`boolean`'

⟨*Num*⟩ ::= ⟨*numeral*⟩

⟨*Const*⟩ ::= '`true`'
| '`false`'
| ⟨*Num*⟩

⟨*Decl*⟩ ::= ⟨*Id*⟩ '`:`' ⟨*Type*⟩ '`=`' ⟨*Expr*⟩
| '`rec`' ⟨*Decl*⟩
| '`function`' ⟨*Id*⟩ '`(`' ⟨*FormList*⟩ '`)`' '`:`' ⟨*Type*⟩ '`=`' ⟨*Expr*⟩
| '`(`' ⟨*Decl*⟩ '`)`'
| ⟨*Decl*⟩ '`;`' ⟨*Decl*⟩
| ⟨*Decl*⟩ '`|`' ⟨*Decl*⟩
| ⟨*Decl*⟩ '`->`' ⟨*Decl*⟩

⟨*FormList*⟩ ::= .
| ⟨*Form*⟩ '`,`' ⟨*FormList*⟩

⟨*Form*⟩ ::= ⟨*Id*⟩ '`:`' ⟨*Type*⟩
| '`name`' ⟨*Id*⟩ '`:`' ⟨*Type*⟩
| '`function`' ⟨*Id*⟩ '`(`' ⟨*FormList*⟩ '`)`' '`:`' ⟨*Type*⟩

⟨*Expr*⟩ ::= ⟨*Const*⟩
| ⟨*Id*⟩
| ⟨*Expr*⟩ '`==`' ⟨*Expr*⟩
| ⟨*Expr*⟩ '`<>`' ⟨*Expr*⟩
| ⟨*Expr*⟩ '`<` ' ⟨*Expr*⟩
| ⟨*Expr*⟩ '`<=`' ⟨*Expr*⟩
| ⟨*Expr*⟩ '`>=`' ⟨*Expr*⟩
| ⟨*Expr*⟩ '`> `' ⟨*Expr*⟩
| ⟨*Expr*⟩ '`+ `' ⟨*Expr*⟩

| ⟨*Expr*⟩ '`-` ' ⟨*Expr*⟩
| ⟨*Expr*⟩ '`or`' ⟨*Expr*⟩
| ⟨*Expr*⟩ '`*` ' ⟨*Expr*⟩
| ⟨*Expr*⟩ '`div`' ⟨*Expr*⟩
| ⟨*Expr*⟩ '`mod`' ⟨*Expr*⟩
| ⟨*Expr*⟩ '`and`' ⟨*Expr*⟩
| '`not`' ⟨*Expr*⟩
| '`+`' ⟨*Expr*⟩
| '`-`' ⟨*Expr*⟩
| '`if`' ⟨*Expr*⟩ '`then`' ⟨*Expr*⟩ '`else`' ⟨*Expr*⟩
| '`let`' ⟨*Decl*⟩ '`in`' ⟨*Expr*⟩
| '`input`' ⟨*Type*⟩
| ⟨*Id*⟩ '`(`' ⟨*ExprList*⟩ '`)`'
| '`(`' ⟨*Expr*⟩ '`)`'

⟨*ExprList*⟩ ::= .
| ⟨*Expr*⟩ '`,`' ⟨*ExprList*⟩

4.4 The Grammar of SIL

⟨*Progr*⟩ ::= '`program`' ⟨*Id*⟩ ⟨*Comm*⟩

⟨*Type*⟩ ::= '`integer`'
| '`boolean`'

⟨*GType*⟩ ::= ⟨*Type*⟩
| '`array`' '`[`' ⟨*Limits*⟩ '`]`' '`of`' ⟨*Type*⟩

⟨*Limits*⟩ ::= .
| ⟨*Num*⟩ '`..`' ⟨*Num*⟩ '`,`' ⟨*Limits*⟩

⟨*Num*⟩ ::= ⟨*numerale*⟩

⟨*Const*⟩ ::= '`true`'
| '`false`'
| ⟨*Num*⟩

⟨*Decl*⟩ ::= '`const`' ⟨*Id*⟩ '`:`' ⟨*Type*⟩ '`=`' ⟨*Expr*⟩
| '`var`' ⟨*Id*⟩ '`:`' ⟨*GType*⟩
| '`var`' ⟨*Id*⟩ '`:`' ⟨*GType*⟩ '`=`' ⟨*Expr*⟩
| '`rec`' ⟨*Decl*⟩
| '`function`' ⟨*Id*⟩ '`(`' ⟨*FormList*⟩ '`)`' '`:`' ⟨*Type*⟩ '`=`' ⟨*Expr*⟩
| '`procedure`' ⟨*Id*⟩ '`(`' ⟨*FormList*⟩ '`)`' ⟨*CompStat*⟩
| '`(`' ⟨*Decl*⟩ '`)`'
| ⟨*Decl*⟩ '`;`' ⟨*Decl*⟩
| ⟨*Decl*⟩ '`|`' ⟨*Decl*⟩
| ⟨*Decl*⟩ '`->`' ⟨*Decl*⟩

⟨*FormList*⟩ ::= .
| ⟨*Form*⟩ '`,`' ⟨*FormList*⟩

⟨*Form*⟩ ::= ⟨*Id*⟩ ‘`:`’ ⟨*Type*⟩
| ‘`name`’ ⟨*Id*⟩ ‘`:`’ ⟨*Type*⟩
| ‘`ref`’ ⟨*Id*⟩ ‘`:`’ ⟨*Type*⟩
| ‘`const`’ ⟨*Id*⟩ ‘`:`’ ⟨*Type*⟩
| ‘`copy`’ ⟨*Id*⟩ ‘`:`’ ⟨*Type*⟩
| ‘`function`’ ⟨*Id*⟩ ‘`(`’ ⟨*FormList*⟩ ‘`)`’ ‘`:`’ ⟨*Type*⟩
| ‘`procedure`’ ⟨*Id*⟩ ‘`(`’ ⟨*FormList*⟩ ‘`)`’

⟨*Expr*⟩ ::= ⟨*Const*⟩
| ⟨*Id*⟩
| ⟨*Id*⟩ ‘`[`’ ⟨*ExprList*⟩ ‘`]`’
| ⟨*Expr*⟩ ‘`==`’ ⟨*Expr*⟩
| ⟨*Expr*⟩ ‘`<>`’ ⟨*Expr*⟩
| ⟨*Expr*⟩ ‘`< `’ ⟨*Expr*⟩
| ⟨*Expr*⟩ ‘`<=`’ ⟨*Expr*⟩
| ⟨*Expr*⟩ ‘`>=`’ ⟨*Expr*⟩
| ⟨*Expr*⟩ ‘`> `’ ⟨*Expr*⟩
| ⟨*Expr*⟩ ‘`+ `’ ⟨*Expr*⟩
| ⟨*Expr*⟩ ‘`- `’ ⟨*Expr*⟩
| ⟨*Expr*⟩ ‘`or`’ ⟨*Expr*⟩
| ⟨*Expr*⟩ ‘`* `’ ⟨*Expr*⟩
| ⟨*Expr*⟩ ‘`div`’ ⟨*Expr*⟩
| ⟨*Expr*⟩ ‘`mod`’ ⟨*Expr*⟩
| ⟨*Expr*⟩ ‘`and`’ ⟨*Expr*⟩
| ‘`not`’ ⟨*Expr*⟩
| ‘`+`’ ⟨*Expr*⟩
| ‘`-`’ ⟨*Expr*⟩
| ‘`if`’ ⟨*Expr*⟩ ‘`then`’ ⟨*Expr*⟩ ‘`else`’ ⟨*Expr*⟩
| ‘`input`’ ⟨*Type*⟩
| ⟨*Id*⟩ ‘`(`’ ⟨*ExprList*⟩ ‘`)`’
| ‘`expr`’ ⟨*Decl*⟩ ‘`;`’ ⟨*Comm*⟩ ‘`result`’ ⟨*Expr*⟩
| ‘`expr`’ ⟨*Comm*⟩ ‘`result`’ ⟨*Expr*⟩
| ‘`(`’ ⟨*Expr*⟩ ‘`)`’

⟨*ExprList*⟩ ::= .
| ⟨*Expr*⟩ ‘`,`’ ⟨*ExprList*⟩

⟨*CompStat*⟩ ::= ‘`begin`’ ⟨*Decl*⟩ ‘`;`’ ⟨*Comm*⟩ ‘`end`’
| ‘`begin`’ ⟨*Comm*⟩ ‘`end`’

⟨*Comm*⟩ ::= ⟨*CompStat*⟩
| ‘`if`’ ⟨*Expr*⟩ ‘`then`’ ⟨*Comm*⟩ ‘`else`’ ⟨*Comm*⟩
| ‘`if`’ ⟨*Expr*⟩ ‘`then`’ ⟨*Comm*⟩
| ⟨*Id*⟩ ‘`:=`’ ⟨*Expr*⟩
| ⟨*Id*⟩ ‘`[`’ ⟨*ExprList*⟩ ‘`]`’ ‘`:=`’ ⟨*Expr*⟩
| ‘`while`’ ⟨*Expr*⟩ ‘`do`’ ⟨*Comm*⟩
| ⟨*Id*⟩ ‘`(`’ ⟨*ExprList*⟩ ‘`)`’
| ‘`print`’ ‘`(`’ ⟨*Format*⟩ ‘`)`’
| ‘`print`’ ‘`(`’ ⟨*Format*⟩ ‘`,`’ ⟨*Exprlist*⟩ ‘`)`’
| ‘`for`’ ⟨*Id*⟩ ‘`:=`’ ⟨*Expr*⟩ ‘`to`’ ⟨*Expr*⟩ ‘`do`’ ⟨*Comm*⟩

| ‘`for`’ ⟨*Id*⟩ ‘`:=`’ ⟨*Expr*⟩ ‘`downto`’ ⟨*Expr*⟩ ‘`do`’ ⟨*Comm*⟩
| ⟨*Comm*⟩ ‘`;`’ ⟨*Comm*⟩

Chapter 5

Implementation

5.1 Short Guide to the Sources

CLAIR source is contained in several files. The sources are commented to some extent and can be consulted for all the implementation details of the system. Here we give a brief description of each source file:

statsem.pl Contains the predicates that implement all the checks and functions pertaining to static semantics.

fdinsem.pl Contains the predicated implementing the interpreter of SFL.

idinsem.pl Contains the predicated implementing the interpreter of SIL.

tokenize.pl Contains CLAIR's tokenizer.

parse.pl Contains CLAIR's parser, written using DCGs (Definite Clause Grammars). The parse accepts a list of tokens generated by the tokenizer and, if the syntax is correct, produces an abstract tree for the program in the form of a Prolog term. See Section 5.2 on the following page for information about the encoding.

common.pl Contains predicates that are used both for the static semantics and for the interpretation of CLAIR's languages.

misc.pl Contains some utility predicates used here and there.

runtime.pl Contains the run-tim system support for SFL and SLI.

trace.pl Contains predicates implementing the "pretty printing" of the program abstract tree. That tree is generated by the parser and subsequently modified by the dynamic semantics predicates.

errors.pl Contains the predicates for handling the errors detected by the static semantics checks.

commands.pl Contains the predicates implementing the interface with the user and also the main predicate of CLAIR.

sysdep.pl Contiene the non-ISO, *system dependent* predicated of CLAIR. All the other files should be written in pure ISO Prolog.

5.2 Syntax and Representation

In this section we review all the constructs of SFL and SIL. For each construct we will give the following information:

1. concrete syntax (preceeded by the word **Concrete**);
2. abstract syntax (preceeded by the word **Abstract**);
3. representation of the abstract syntax as a Prolog term (preceeded by the word **Representation**).

5.2.1 Type and GType

Simple Types

Concrete syntax:

⟨*Type*⟩ ::= '`integer`'
| '`boolean`'

Abstract syntax:

int|bool

Representation:

$$\text{repr}(\text{int}) \stackrel{\text{def}}{=} \texttt{int}$$
$$\text{repr}(\text{bool}) \stackrel{\text{def}}{=} \texttt{bool}$$

General Types (SIL only)

Concrete syntax:

⟨*GType*⟩ ::= ⟨*Type*⟩
| '`array`' '`[`' ⟨*Limits*⟩ '`]`' '`of`' ⟨*Type*⟩

⟨*Limits*⟩ ::= .
| ⟨*Num*⟩ '`..`' ⟨*Num*⟩ '`,`' ⟨*Limits*⟩

Abstract syntax:

$$t|(\text{lim}, t)\ \textbf{array} \tag{5.1}$$
$$\text{lim} ::= .|n..m, \text{lim} \tag{5.2}$$

Representation:

$$\text{repr}\big((\text{lim}, t)\ \textbf{array}\big) \stackrel{\text{def}}{=} \texttt{array}\big(\text{repr}(\text{lim}), \text{repr}(t)\big),$$
$$\text{repr}(.) \stackrel{\text{def}}{=} \texttt{[]},$$
$$\text{repr}(n..m, \text{lim}) \stackrel{\text{def}}{=} \texttt{[b}\big(\text{repr}(n), \text{repr}(m)\big)\texttt{|}\ \text{repr}(\text{lim})\texttt{]}.$$

5.2.2 Const, Num

Integer Numbers

Concrete syntax:

$\langle Num \rangle$ ::= $\langle numeral \rangle$

Abstract syntax:

m

Representation:

repr(m) $\stackrel{\text{def}}{=}$ the Prolog term representing the integer m.

Integer and Boolean Constants

Concrete syntax:

$\langle Const \rangle$ ::= '`true`'
| '`false`'
| $\langle Num \rangle$

Abstract syntax:

tt|ff|m

Representation:

repr(tt) $\stackrel{\text{def}}{=}$ `tt`,

repr(ff) $\stackrel{\text{def}}{=}$ `ff`,

repr(m) $\stackrel{\text{def}}{=}$ repr(m), (sic!).

5.2.3 Declarations

Simple Declaration (SFL only)

Concrete syntax:

$\langle Decl \rangle$ ::= $\langle Id \rangle$ '`:`' $\langle Type \rangle$ '`=`' $\langle Expr \rangle$

Abstract syntax:

$x : t = e$

Representation:

repr($x : t = e$) $\stackrel{\text{def}}{=}$ `def`$\big($repr(x), repr(t), repr(e)$\big)$.

Constant Declaration (SIL only)

Concrete syntax:

⟨*Decl*⟩ ::= '`const`' ⟨*Id*⟩ '`:`' ⟨*Type*⟩ '`=`' ⟨*Expr*⟩

Abstract syntax:

const $x : t = e$

Representation:

$$\text{repr}(\textbf{const}\ x : t = e) \stackrel{\text{def}}{=} \texttt{con}\big(\text{repr}(x), \text{repr}(t), \text{repr}(e)\big).$$

Variable Declaration (SIL only)

Concrete syntax:

⟨*Decl*⟩ ::= '`var`' ⟨*Id*⟩ '`:`' ⟨*GType*⟩ ['`=`' ⟨*Expr*⟩]

Abstract syntax:

var $x : t = e$

Representation:

$$\text{repr}(\textbf{var}\ x : t = e) \stackrel{\text{def}}{=} \texttt{var}\big(\text{repr}(x), \text{repr}(t), \text{repr}(e)\big).$$

Sequential Composition

Concrete syntax:

⟨*Decl*⟩ ::= ⟨*Decl*⟩ '`;`' ⟨*Decl*⟩

Abstract syntax:

$d_1; d_2$

Representation:

$$\text{repr}(d_1; d_2) \stackrel{\text{def}}{=} \texttt{seq}\big(\text{repr}(d_1), \text{repr}(d_2)\big).$$

Parallel Composition

Concrete syntax:

⟨*Decl*⟩ ::= ⟨*Decl*⟩ '`|`' ⟨*Decl*⟩

Abstract syntax:

d_1 **parallel** d_2

Representation:

$$\text{repr}(d_1\ \textbf{parallel}\ d_2) \stackrel{\text{def}}{=} \texttt{parallel}\big(\text{repr}(d_1), \text{repr}(d_2)\big).$$

Private Composition

Concrete syntax:

$\langle Decl\rangle$ ::= $\langle Decl\rangle$ '`->`' $\langle Decl\rangle$

Abstract syntax:

d_1 **private** d_2

Representation:

$$\mathrm{repr}(d_1 \textbf{ private } d_2) \stackrel{\mathrm{def}}{=} \texttt{private}\big(\mathrm{repr}(d_1), \mathrm{repr}(d_2)\big).$$

Recursive Declaration

Concrete syntax:

$\langle Decl\rangle$ ::= '`rec`' $\langle Decl\rangle$

Abstract syntax:

rec d

Representation:

$$\mathrm{repr}(\textbf{rec } d) \stackrel{\mathrm{def}}{=} \texttt{rec}\big(\mathrm{repr}(d)\big).$$

Function Declaration

Concrete syntax:

$\langle Decl\rangle$ ::= '`function`' $\langle Id\rangle$ '`(`' $\langle FormList\rangle$ '`)`' '`:`' $\langle Type\rangle$ '`=`' $\langle Expr\rangle$

Abstract syntax:

function $f(\mathrm{form}) : \mathrm{et} = e$

Representation:

$$\mathrm{repr}\big(\textbf{function } f(\mathrm{form}) : \mathrm{et} = e\big) \stackrel{\mathrm{def}}{=} \texttt{fundef}\big(\mathrm{repr}(f), \mathrm{repr}(\mathrm{form}), \mathrm{repr}(et), \mathrm{repr}(e)\big).$$

Procedure Declaration (SIL only)

Concrete syntax:

$\langle Decl\rangle$ ::= '`procedure`' $\langle Id\rangle$ '`(`' $\langle FormList\rangle$ '`)`' $\langle CompStat\rangle$

Abstract syntax:

procedure $p(\mathrm{form})c$

Representation:

$$\mathrm{repr}\big(\textbf{procedure } p(\mathrm{form})c\big) \stackrel{\mathrm{def}}{=} \texttt{procdef}\big(\mathrm{repr}(p), \mathrm{repr}(\mathrm{form}), \mathrm{repr}(c)\big).$$

5.2.4 Formal Parameters

Concrete syntax:

⟨*Form*⟩ ::= ⟨*Id*⟩ ‘`:`’ ⟨*Type*⟩
| ⟨*Id*⟩ ‘`:`’ ⟨*Type*⟩
| ‘`name`’ ⟨*Id*⟩ ‘`:`’ ⟨*Type*⟩
| ‘`ref`’ ⟨*Id*⟩ ‘`:`’ ⟨*Type*⟩
| ‘`const`’ ⟨*Id*⟩ ‘`:`’ ⟨*Type*⟩
| ‘`copy`’ ⟨*Id*⟩ ‘`:`’ ⟨*Type*⟩
| ‘`function`’ ⟨*Id*⟩ ‘`(`’ ⟨*FormList*⟩ ‘`)`’ ‘`:`’ ⟨*Type*⟩
| ‘`procedure`’ ⟨*Id*⟩ ‘`(`’ ⟨*FormList*⟩ ‘`)`’

⟨*FormList*⟩ ::= .
| ⟨*Form*⟩ ‘`,`’ ⟨*FormList*⟩

Abstract syntax:

parspec ::= $x : t$| **name** $x : t$| **ref** $x : t$| **copy** $x : t$| **const** $x : t$
| **function** f(form) → et| **procedure** p(form),
form ::= .|parspec, form

Representation:

$$\begin{aligned}
\mathrm{repr}(x : t) &\stackrel{\text{def}}{=} \texttt{fval}\big(\mathrm{repr}(x), \mathrm{repr}(t)\big), && \text{(SFL only)}\\
\mathrm{repr}(x : t) &\stackrel{\text{def}}{=} \texttt{ival}\big(\mathrm{repr}(x), \mathrm{repr}(t)\big), && \text{(SIL only)}\\
\mathrm{repr}(\textbf{name}\ x : t) &\stackrel{\text{def}}{=} \texttt{name}\big(\mathrm{repr}(x), \mathrm{repr}(t)\big),\\
\mathrm{repr}(\textbf{ref}\ x : t) &\stackrel{\text{def}}{=} \texttt{ref}\big(\mathrm{repr}(x), \mathrm{repr}(t)\big),\\
\mathrm{repr}(\textbf{const}\ x : t) &\stackrel{\text{def}}{=} \texttt{const}\big(\mathrm{repr}(x), \mathrm{repr}(t)\big),\\
\mathrm{repr}(\textbf{copy}\ x : t) &\stackrel{\text{def}}{=} \texttt{copy}\big(\mathrm{repr}(x), \mathrm{repr}(t)\big),\\
\mathrm{repr}\big(\textbf{function}\ f(\mathrm{form}) \to \mathrm{et}\big) &\stackrel{\text{def}}{=} \texttt{funcpar}\big(\mathrm{repr}(f), \mathrm{repr}(\mathrm{form}), \mathrm{repr}(t)\big),\\
\mathrm{repr}\big(\ \textbf{procedure}\ p(\mathrm{form})\big) &\stackrel{\text{def}}{=} \texttt{procpar}\big(\mathrm{repr}(p), \mathrm{repr}(\mathrm{form})\big),\\
\mathrm{repr}(.) &\stackrel{\text{def}}{=} \texttt{[]},\\
\mathrm{repr}(\mathrm{parspec}, \mathrm{form}) &\stackrel{\text{def}}{=} \texttt{[}\,\mathrm{repr}(\mathrm{parspec})\,\texttt{|}\,\mathrm{repr}(\mathrm{form})\texttt{]}.
\end{aligned}$$

5.2.5 Expressions

Identifiers

Concrete syntax:

⟨*Expr*⟩ ::= ⟨*Id*⟩

Abstract syntax:

x

Representation:

$$\mathrm{repr}(x) \stackrel{\text{def}}{=} \texttt{i}\big(\mathrm{repr}(x)\big).$$

Array Reference (SIL only)

Concrete syntax:

⟨*Expr*⟩ ::= ⟨*Id*⟩ '`[`' ⟨*ExprList*⟩ '`]`'

Abstract syntax:

x[el]

Representation:

$\mathrm{repr}\big(x[\mathrm{el}]\big) \stackrel{\mathrm{def}}{=} \texttt{aref}\big(\mathrm{repr}(x), \mathrm{repr}(\mathrm{el})\big).$

Constants

Concrete syntax:

⟨*Expr*⟩ ::= ⟨*Const*⟩

Abstract syntax:

con

Representation:

$\mathrm{repr}(\mathrm{con}) \stackrel{\mathrm{def}}{=} \mathrm{repr}(\mathrm{con}).$

Binary Operators

Concrete syntax:

⟨*Expr*⟩ ::= ⟨*Expr*⟩ '`==`' ⟨*Expr*⟩
| ⟨*Expr*⟩ '`<>`' ⟨*Expr*⟩
| ⟨*Expr*⟩ '`< `' ⟨*Expr*⟩
| ⟨*Expr*⟩ '`<=`' ⟨*Expr*⟩
| ⟨*Expr*⟩ '`>=`' ⟨*Expr*⟩
| ⟨*Expr*⟩ '`> `' ⟨*Expr*⟩
| ⟨*Expr*⟩ '`+ `' ⟨*Expr*⟩
| ⟨*Expr*⟩ '`- `' ⟨*Expr*⟩
| ⟨*Expr*⟩ '`* `' ⟨*Expr*⟩
| ⟨*Expr*⟩ '`div`' ⟨*Expr*⟩
| ⟨*Expr*⟩ '`mod`' ⟨*Expr*⟩
| ⟨*Expr*⟩ '`and`' ⟨*Expr*⟩
| ⟨*Expr*⟩ '`or`' ⟨*Expr*⟩ @end example

Abstract syntax:

e_1 bop e_2

Representation:

$$\begin{aligned}
\mathrm{repr}(e_1 == e_2) &\stackrel{\mathrm{def}}{=} \texttt{eq}\big(\mathrm{repr}(e_1), \mathrm{repr}(e_2)\big),\\
\mathrm{repr}(e_1 <> e_2) &\stackrel{\mathrm{def}}{=} \texttt{ne}\big(\mathrm{repr}(e_1), \mathrm{repr}(e_2)\big),\\
\mathrm{repr}(e_1 < e_2) &\stackrel{\mathrm{def}}{=} \texttt{lt}\big(\mathrm{repr}(e_1), \mathrm{repr}(e_2)\big),\\
\mathrm{repr}(e_1 <= e_2) &\stackrel{\mathrm{def}}{=} \texttt{le}\big(\mathrm{repr}(e_1), \mathrm{repr}(e_2)\big),\\
\mathrm{repr}(e_1 >= e_2) &\stackrel{\mathrm{def}}{=} \texttt{ge}\big(\mathrm{repr}(e_1), \mathrm{repr}(e_2)\big),\\
\mathrm{repr}(e_1 > e_2) &\stackrel{\mathrm{def}}{=} \texttt{gt}\big(\mathrm{repr}(e_1), \mathrm{repr}(e_2)\big),\\
\mathrm{repr}(e_1 + e_2) &\stackrel{\mathrm{def}}{=} \texttt{plus}\big(\mathrm{repr}(e_1), \mathrm{repr}(e_2)\big),\\
\mathrm{repr}(e_1 - e_2) &\stackrel{\mathrm{def}}{=} \texttt{minus}\big(\mathrm{repr}(e_1), \mathrm{repr}(e_2)\big),\\
\mathrm{repr}(e_1 * e_2) &\stackrel{\mathrm{def}}{=} \texttt{prod}\big(\mathrm{repr}(e_1), \mathrm{repr}(e_2)\big),\\
\mathrm{repr}(e_1 \,\mathrm{div}\, e_2) &\stackrel{\mathrm{def}}{=} \texttt{div}\big(\mathrm{repr}(e_1), \mathrm{repr}(e_2)\big),\\
\mathrm{repr}(e_1 \,\mathrm{mod}\, e_2) &\stackrel{\mathrm{def}}{=} \texttt{mod}\big(\mathrm{repr}(e_1), \mathrm{repr}(e_2)\big),\\
\mathrm{repr}(e_1 \,\mathrm{and}\, e_2) &\stackrel{\mathrm{def}}{=} \texttt{and}\big(\mathrm{repr}(e_1), \mathrm{repr}(e_2)\big),\\
\mathrm{repr}(e_1 \,\mathrm{or}\, e_2) &\stackrel{\mathrm{def}}{=} \texttt{or}\big(\mathrm{repr}(e_1), \mathrm{repr}(e_2)\big).
\end{aligned}$$

Not

Concrete syntax:

⟨*Expr*⟩ ::= '`not`' ⟨*Expr*⟩

Abstract syntax:

not e

Representation:

$$\mathrm{repr}(\textbf{not}\ e) \stackrel{\mathrm{def}}{=} \texttt{not}\big(\mathrm{repr}(e)\big).$$

Minus

Concrete syntax:

⟨*Expr*⟩ ::= '`-`' ⟨*Expr*⟩

Abstract syntax:

$-e$

Representation:

$$\mathrm{repr}(-e) \stackrel{\mathrm{def}}{=} \texttt{uminus}\big(\mathrm{repr}(e)\big).$$

Conditional Expression

Concrete syntax:

$\langle \textit{Expr} \rangle$ ::= '`if`' $\langle \textit{Expr} \rangle$ '`then`' $\langle \textit{Expr} \rangle$ '`else`' $\langle \textit{Expr} \rangle$

Abstract syntax:

if e **then** e_0 **else** e_1

Representation:

$$\text{repr}(\textbf{if } e \textbf{ then } e_0 \textbf{ else } e_1) \stackrel{\text{def}}{=} \texttt{if}\big(\text{repr}(e), \text{repr}(e_1), \text{repr}(e_2)\big).$$

Expression Block (SFL only)

Concrete syntax:

$\langle \textit{Expr} \rangle$ ::= '`let`' $\langle \textit{Decl} \rangle$ '`in`' $\langle \textit{Expr} \rangle$

Abstract syntax:

let d **in** e

Representation:

$$\text{repr}(\textbf{let } d \textbf{ in } e) \stackrel{\text{def}}{=} \texttt{let}\big(\text{repr}(d), \text{repr}(e)\big).$$

Input

Concrete syntax:

$\langle \textit{Expr} \rangle$ ::= '`input`' $\langle \textit{Type} \rangle$

Abstract syntax:

input et

Representation:

$$\text{repr}(\textbf{input } \text{et}) \stackrel{\text{def}}{=} \texttt{input}\big(\text{repr}(\text{et})\big).$$

Function Call

Concrete syntax:

$\langle \textit{Expr} \rangle$ ::= $\langle \textit{Id} \rangle$ '`(`' $\langle \textit{ExprList} \rangle$ '`)`'

Abstract syntax:

$f(\text{ae})$

Representation:

$$\text{repr}\big(f(\text{ae})\big) \stackrel{\text{def}}{=} \texttt{funcall}\big(\text{repr}(\text{ae})\big).$$

Expressions with Side-Effects (SIL only)

Concrete syntax:

⟨*Expr*⟩ ::= '`expr`' ⟨*Decl*⟩ '`;`' ⟨*Comm*⟩ '`result`' ⟨*Expr*⟩
| ⟨*Expr*⟩ ::= '`expr`' ⟨*Comm*⟩ '`result`' ⟨*Expr*⟩

Abstract syntax:

expr $d; c$ **result** e| **exprnil**; c **result** e

Representation:

$\text{repr}(\textbf{expr}\ d; c\ \textbf{result}\ e) \stackrel{\text{def}}{=} \texttt{side}\big(\text{repr}(d), \text{repr}(c), \text{repr}(e)\big).$

5.2.6 Lists of Expressions

Concrete syntax:

⟨*ExprList*⟩ ::= .
| ⟨*Expr*⟩ '`,`' ⟨*ExprList*⟩

Abstract syntax:

ae ::= .|e, ae

Representation:

$\text{repr}(.) \stackrel{\text{def}}{=} \texttt{[]},$

$\text{repr}(e, \text{ae}) \stackrel{\text{def}}{=} \texttt{[}\,\text{repr}(e)\,\texttt{|}\,\text{repr}(\text{ae})\texttt{]}.$

5.2.7 Commands (SIL only)

Compound Statement

Concrete syntax:

⟨*CompStat*⟩ ::= '`begin`' ⟨*Decl*⟩ '`;`' ⟨*Comm*⟩ '`end`'

Abstract syntax:

$d; c$

Representation:

$\text{repr}(d; c) \stackrel{\text{def}}{=} \texttt{block}\big(\text{repr}(d), \text{repr}(c)\big).$

Sequence of Commands

Concrete syntax:

⟨*Comm*⟩ ::= ⟨*Comm*⟩ '`;`' ⟨*Comm*⟩

Abstract syntax:

$c_1; c_2$

Representation:

$\text{repr}(c_1; c_2) \stackrel{\text{def}}{=} \texttt{seq}\big(\text{repr}(c_1), \text{repr}(c_2)\big).$

Conditional Command

Concrete syntax:

$\langle \mathit{Comm} \rangle$ ::= '`if`' $\langle \mathit{Expr} \rangle$ '`then`' $\langle \mathit{Comm} \rangle$ '`else`' $\langle \mathit{Comm} \rangle$

Abstract syntax:

if e **then** c_1 **else** c_2

Representation:

$$\mathrm{repr}(\textbf{if } e \textbf{ then } c_1 \textbf{ else } c_2) \stackrel{\mathrm{def}}{=} \texttt{if}\big(\mathrm{repr}(e), \mathrm{repr}(c_1), \mathrm{repr}(c_2)\big).$$

Assignment

Concrete syntax:

$\langle \mathit{Comm} \rangle$ ::= $\langle \mathit{Id} \rangle$ '`:=`' $\langle \mathit{Expr} \rangle$
| $\langle \mathit{Id} \rangle$ '`[`' $\langle \mathit{ExprList} \rangle$ '`]`' '`:=`' $\langle \mathit{Expr} \rangle$

Abstract syntax:

$x := e | x[\mathrm{el}] := e$

Representation:

$$\mathrm{repr}(x := e) \stackrel{\mathrm{def}}{=} \texttt{ass}\big(\mathrm{repr}(x), \mathrm{repr}(e)\big),$$
$$\mathrm{repr}(x[\mathrm{el}] := e) \stackrel{\mathrm{def}}{=} \texttt{ass}\Big(\texttt{aref}\big(\mathrm{repr}(x), \mathrm{repr}(\mathrm{el})\big), \mathrm{repr}(e)\Big).$$

While Command

Concrete syntax:

$\langle \mathit{Comm} \rangle$ ::= '`while`' $\langle \mathit{Expr} \rangle$ '`do`' $\langle \mathit{Comm} \rangle$

Abstract syntax:

while e **do** c

Representation:

$$\mathrm{repr}(\textbf{while } e \textbf{ do } c) \stackrel{\mathrm{def}}{=} \texttt{while}\big(\mathrm{repr}(e), \mathrm{repr}(c)\big).$$

Procedure Call

Concrete syntax:

$\langle \mathit{Comm} \rangle$ ::= $\langle \mathit{Id} \rangle$ '`(`' $\langle \mathit{ExprList} \rangle$ '`)`'

Abstract syntax:

$p(\mathrm{ae})$

Representation:

$$\mathrm{repr}\big(p(\mathrm{ae})\big) \stackrel{\mathrm{def}}{=} \texttt{proccall}\big(\mathrm{repr}(p), \mathrm{repr}(\mathrm{ae})\big).$$

Output Command

Concrete syntax:

⟨*Comm*⟩ ::= '`print`' '`(`' ⟨*Format*⟩ '`,`' ⟨*Exprlist*⟩] '`)`'
| '`print`' '`(`' ⟨*Format*⟩ '`)`'

Abstract syntax:

print *el*

Representation:[1]

repr(**print** *el*) ≃ `print`(`<Format>`, repr(el)).

Comando for

Concrete syntax:

⟨*Comm*⟩ ::= '`for`' ⟨*Id*⟩ '`:=`' ⟨*Expr*⟩ '`to`' ⟨*Expr*⟩ '`do`' ⟨*Comm*⟩
| '`for`' ⟨*Id*⟩ '`:=`' ⟨*Expr*⟩ '`downto`' ⟨*Expr*⟩ '`do`' ⟨*Comm*⟩

The `for` command has no corresponding abstract syntax, since the parser module immediately expands it into a `while` cycle as follows:

```
for X := E1 to (downto) E2 do C
```

is translated to

```
X := E1;
begin
  const $limit : integer = E2;
  while X <= (>=) $limit do begin
    begin
      const X : integer = X;
      C
    end;
    X := X + (-) 1
  end
end
```

Notice that `C` (the command) is executed in an environment where `X` (the `for` index) is a constant, and this not modifiable. The `for` command of SIL is this syntactically and semantically identical to the `for` command of Pascal.

5.2.8 Program (SIL only)

Concrete syntax:

⟨*Progr*⟩ ::= '`program`' ⟨*Id*⟩ ⟨*Comm*⟩

Every SIL program is preceded by the `program` keyword followed by an identifier. This construct has no correspondence in the abstract syntax.

[1] Kludge: yes, this is a representation of the concrete syntax and not of the abstract one.

5.3 Correspondence Between Theory and Implementation

Table 5.1 relates the most important predicates used in the definition of the static and dynamic semantics with the Prolog predicates the implement them and the source files that contain the corresponding implementation.

Predicate	Prolog	Source file
$\alpha \vdash con : t$	`essem/3`	`statsem.pl`
$\alpha \vdash e : \text{et}$	`essem/3`	`statsem.pl`
$\alpha \vdash d$	`wfd/3`	`statsem.pl`
$\vdash d : \beta$	`dssem/2`	`statsem.pl`
$\alpha \vdash c$	`wfc/2`	`statsem.pl`
$\alpha \vdash \text{el} : \text{elt}$	`elssem/3`	`statsem.pl`
$\alpha, \text{form} \vdash \text{ae}$	`aeMatchForm/3`	`statsem.pl`
$\text{form} : \beta$	`fEnv/2`	`statsem.pl`
$\text{match}(\text{form}, \text{form})$	`formMatchForm/2`	`statsem.pl`
$\rho \vdash e \rightarrow e'$	`e1t/3`	`fdinsem.pl`
$\rho \vdash d \rightarrow d'$	`d1t/3`	`fdinsem.pl`
$\text{acon} \vdash \text{form} \rightarrow \rho$	`makeEnv/3`	`runtime.pl`
$\rho, \text{form} \vdash \text{ae} \rightarrow \text{ae}'$	`ae1t/4`	`fdinsem.pl`
$\rho \vdash \langle e, \sigma \rangle \rightarrow \langle e', \sigma' \rangle$	`e1t/5`	`idinsem.pl`
$\rho \vdash \langle d, \sigma \rangle \rightarrow \langle d', \sigma' \rangle$	`d1t/5`	`idinsem.pl`
$\rho \vdash \langle c, \sigma \rangle \rightarrow \langle c', \sigma' \rangle$	`c1t/5`	`idinsem.pl`
$\text{acon} \vdash \text{form} \rightarrow \rho, \sigma$	`makeEnvStore/4`	`runtime.pl`
$\rho, \text{form} \vdash \langle \text{ae}, \sigma \rangle \rightarrow \langle \text{ae}', \sigma' \rangle$	`ae1t/6`	`idinsem.pl`

Table 5.1: Correspondence between the predicates used in the theory and their implementation.

Chapter 6

A Session with CLAIR

The following is the transcription of what could be a typical work session with CLAIR. Of course, this is not the same thing as providing proper documentation for users of the program. On the other hand, the environment privided by CLAIR is rather self-explanatory: there is an '`help`' command, the error messages should be understandable and so forth. Notive that the static semantics checks give a quite detailed explanation of the semantic errors they may find.

```
This is CLAIR, the Combined Language and Abstract Interpretation Resource
Copyright (C) 2002-2005 Roberto Bagnara <bagnara@cs.unipr.it>
Type 'help' for advice, 'quit' to exit.
Default settings follows.

The language currently recognized is: imperative;
    program file currently loaded is: *no file loaded*;
                     program name is: *no program loaded*;
                and has been checked: no;
                      tracing is set: off;
              tracing configurations: yes;
                       tracing store: yes;
              tracing output goes to: user.

CLAIR> help

CLAIR provides you with the following commands:

help      - the command you have just issued;
quit      - leave CLAIR, back to the system;
info      - display the current interpreter status;
list  <f> - show file <fname> on the screen;
load  <f> - load program from file <fname>;
edit  <f> - edit file <f>;
run       - run the currently loaded program, if any;
check     - check static semantics of the currently
            loaded program, if any;
trace ... - set trace status, possible formats are:
```

```
  trace on/off            - turn tracing on/off
  trace config/store/both - what to trace
  trace file <f>          - where to trace
lang <l>  - set the language type, possible values for <l>
            are: ‘imperative’ (default), ‘functional’;
show      - pretty print abstract syntax of program;
shell <c> - if <c> is omitted, open a nested shell session,
            shell command <c> is executed otherwise;
dir       - shows the current directory contents;
prolog    - leave CLAIR, back to Prolog;
CLAIR> load ../tests/nprime
Loading program file ‘../tests/nprime’
Tokenising... done.
Parsing... syntax ok!
CLAIR> check
Static semantics ok.
CLAIR> run
enter n to find the n-th prime> 3
the 3th prime is 5
enter n to find the n-th prime> 12
the 12th prime is 37
enter n to find the n-th prime> -1
0.1 seconds to run.
CLAIR> l
Ambiguous command ‘l’: choose one of [list, load, language]
CLAIR> language functional
CLAIR> lo ../tests/apply
Loading program file ‘../tests/apply’
Tokenising... done.
Parsing... syntax ok!
CLAIR> info
The language currently recognized is: functional;
    program file currently loaded is: ../tests/apply;
                      program name is: *no name*;
                and has been checked: no;
                       tracing is set: off;
              tracing configurations: yes;
                        tracing store: yes;
              tracing output goes to: user.
CLAIR> check
Undeclared identifier: ‘nowaty’
Type mismatch: a value of type ‘function(integer) -> integer’
               cannot be passed, being formal ‘limit’
               of type ‘integer’
Static semantics check failed.
CLAIR> edit ../tests/apply

        [edit session to correct errors]

CLAIR> load apply
```

```
Cannot find ‘apply’
CLAIR> load ../tests/apply
Loading program file ‘../tests/apply’
Tokenising... done.
Parsing... *bleach*, syntax error!
CLAIR> lang
Illegal command: ‘language’ requires 1 argument
CLAIR> lang fun
CLAIR> load ../tests/apply
Loading program file ‘../tests/apply’
Tokenising... done.
Parsing... syntax ok!
CLAIR> check
Static semantics ok, expression type is int.
CLAIR> trace on
CLAIR> trace file apply.trace
CLAIR> info
The language currently recognized is: functional;
    program file currently loaded is: ../tests/apply;
                      program name is: *no name*;
                and has been checked: yes;
                       tracing is set: on;
               tracing configurations: yes;
                        tracing store: yes;
               tracing output goes to: apply.trace.
CLAIR> run
7
0.24 seconds to run.
CLAIR> quit
Bye!
```

Bibliography

[1] G. D. Plotkin. A structural approach to operational semantics. Technical Report DAIMI FN-19, Aarhus University, Department of Computer Science, 1981. 3

Appendix A

GNU GENERAL PUBLIC LICENSE

Version 2, June 1991

Copyright © 1989, 1991 Free Software Foundation, Inc.
59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software—to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

   Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

   You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

(a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

(b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

(c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

(a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

(b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

(c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at

all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

   Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

Appendix: How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
<one line to give the program's name and a brief idea of what it does.>
Copyright (C) <yyyy> <name of author>

This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
```

```
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.
```

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

```
Gnomovision version 69, Copyright (C) <yyyy> <name of author>
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type ‘show w’.
This is free software, and you are welcome to redistribute it
under certain conditions; type ‘show c’ for details.
```

The hypothetical commands ‘show w’ and ‘show c’ should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than ‘show w’ and ‘show c’; they could even be mouse-clicks or menu items–whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a “copyright disclaimer” for the program, if necessary. Here is a sample; alter the names:

```
Yoyodyne, Inc., hereby disclaims all copyright interest in the program
‘Gnomovision’ (which makes passes at compilers) written by James Hacker.

<signature of Ty Coon>, 1 April 1989
Ty Coon, President of Vice
```

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

Appendix B

GNU Free Documentation License

Version 1.2, November 2002
Copyright © 2000,2001,2002 Free Software Foundation, Inc. 59 Temple Place, Suite 330, Boston, MA 02111-1307, USA
Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document *free* in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein.

The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies

in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided

that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements."

COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections

with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warrany Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright (C) *year your name*. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

> with the Invariant Sections being *list their titles*, with the Front-Cover Texts being *list*, and with the Back-Cover Texts being *list*.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.