Representing Program Logics in Evaluation Logic

Eugenio Moggi*
DISI, Univ. of Genova, v.le Benedetto XV 3, 16132 Genova, ITALY
moggi@disi.unige.it

Abstract

We consider the *representation* of three program logics in EL_T (Evaluation Logic): $VTLoE$ (Variable Typed Logic of Effects), modal μ-calculus, Hoare's Logic. The most interesting result is the definitional extension of EL_T with logical operators inspired by $VTLoE$. Unlike their original counterparts, these logical operators make sense for a wide range of programming languages. In fact, they are defined independently from the interpretation of computational types, and their logical properties are derivable from the axioms of EL_T. Also for the modal μ-calculus it is possible to give a treatment in terms of EL_T far more general than that in terms of Labelled Transition Systems. We have considered also a representation of Hoare logic into EL_T, but we have not achieved results at the same level of generality reached for $VTLoE$.

Introduction

Most program logics are tied up to specific programming languages, look rather ad hoc, and standard logical axioms may be unsound. This state of affairs is not very satisfactory, since it is time-consuming (even for logicians) to get acquainted with new logics and develop good proof strategies. On the other hand, Evaluation Logic EL_T (see [Mog94b, Mog94a]) is a straightforward extension of Higher Order Logic HOL, where the dependencies from a programming language are confined to computational types and auxiliary operations on them. In the same way as one need Peano's axioms to prove properties of the natural numbers in HOL, one must first axiomatize the properties of program constructs (i.e. the auxiliary operations) in EL_T, before one can prove interesting properties of programs written in a given programming language. One may expect that EL_T, because of its generality, is unlikely to express interesting properties of programs (or it may do so in a clumsy way, as it happens in LCF for non-functional languages).

Our results show that this is not so. Indeed, we consider two paradigmatic program logics, Variable Typed Logic of Effects $VTLoE$ (see [HMSTar]) and the modal μ-calculus (see [Eme90]), and define two simple translations of these program logics into EL_T. Moreover, we show that the logical axioms for these program logics can be validated in EL_T from *almost* no assumptions. This is in sharp contrast with the usual way they are validated, i.e. by appealing to a satisfaction relation defined in terms of an operational semantics (for an ML-like language in the case of $VTLoE$, and for a CCS-like language in the case of the modal μ-calculus).

The most interesting result is the definitional extension of EL_T with logical operators inspired by $VTLoE$. Unlike their original counterparts, these logical operators make sense for a wide range of programming languages. In fact, they are defined independently from the interpretation of computational types, and their logical properties are derivable from the axioms of EL_T. Also for the modal μ-calculus it is possible to give a treatment in terms of EL_T far more general than that in terms of Labelled Transition Systems. We have considered also a representation of Hoare logic into EL_T, but we have not achieved results at the same level of generality reached for $VTLoE$.

*This work is supported by ESPRIT BRA 6811 (Categorical Logic In Computer Science II), EC SCIENCE twinning ERBSC1*CT920795 (Progr. Lang. Semantics and Program Logics) and HCM Scientific Cooperation Network CHRX-CT93-0046 (Lambda Calcul Typé)

In summary, EL_T seems quite good for expressing *modalities* present in various program logics (with the exception of temporal logics), and even at validating their *logical* properties. Of course, one cannot expect to validate in *pure* EL_T properties that are programming language dependent. In this case one can only hope to validate them in some suitable EL_T-theory, which axiomatize the relevant assumptions about the programming language (e.g. see the section on Hoare logic).

The paper is organized as follows. Section 1 reviews the syntax and axioms of EL_T. Section 2 recall the original semantics of $VTLoE$, introduces the representation of $VTLoE$ into EL_T, and investigates its main properties. Section 3 investigates briefly the representations of Minimal Modal Logic and Hoare Logic into EL_T. For each of the representations considered in the paper, we discuss at the end of the relevant section what are the main open issues.

1 Syntax and axioms of EL_T

Evaluation logic EL_T is a conservative extension of (dependently) typed predicate calculus obtained by adding *computational types*. Its equational calculus is the metalanguage for computational monads ML_T. We adopt the setting of [Mog94a], where the *evaluation modalities* of [Pit91] turn out to be definable. The syntactic categories of EL_T are (dependent) types, terms and formulas.

- $\Gamma \vdash \tau$ type means "τ is a type in context Γ". Types are closed under the rule

  $$(T)\ \frac{\Gamma \vdash \tau \text{ type}}{\Gamma \vdash T\tau \text{ type}}$$

  $T\tau$ is called a **computational type**, and terms of type $T\tau$ should be thought of as programs which return values of type τ. Contexts are built from the empty context $\emptyset$ using the rule

  $$(\text{add})\ \frac{\Gamma \vdash \tau \text{ type}}{\Gamma, x\colon\tau \vdash}\ x \notin \text{DV}(\Gamma) \qquad \text{where DV}(\Gamma) \text{ is the set of variables declared in } \Gamma.$$

- $\Gamma \vdash e\colon\tau$ means "e is a term of type τ in context Γ". Terms are closed under the rules

  $$(\text{lift})\ \frac{\Gamma \vdash e\colon\tau}{\Gamma \vdash [e]\colon T\tau} \qquad (\text{let})\ \frac{\Gamma \vdash e_1\colon T\tau_1 \qquad \Gamma, x\colon\tau_1 \vdash e_2\colon T\tau_2}{\Gamma \vdash (\text{let}\, x{\Leftarrow}e_1 \,\text{in}\, e_2)\colon T\tau_2}\ x \notin \text{FV}(\tau_2)$$

  Intuitively the program $[e]$ simply returns the value e, while $(\text{let}\, x{\Leftarrow}e_1 \,\text{in}\, e_2)$ first evaluates e_1 and binds the result to x, then evaluates e_2.

- $\Gamma \vdash \phi$ prop means "ϕ is a formula in context Γ". Formulas are closed under the rules

  $$(\text{necessity})\ \frac{\Gamma \vdash e\colon T\tau \qquad \Gamma, x\colon\tau \vdash \phi \text{ prop}}{\Gamma \vdash [x{\Leftarrow}e]\phi \text{ prop}}$$

  $$(\text{possibility})\ \frac{\Gamma \vdash e\colon T\tau \qquad \Gamma, x\colon\tau \vdash \phi \text{ prop}}{\Gamma \vdash \langle x{\Leftarrow}e\rangle\phi \text{ prop}}$$

  $$(\text{evaluation})\ \frac{\Gamma \vdash e\colon T\tau \qquad \Gamma \vdash v\colon\tau}{\Gamma \vdash e{\Downarrow}v \text{ prop}}$$

  Intuitively the formula $[x{\Leftarrow}e]\phi$ means that every possible result of program e satisfies ϕ, $\langle x{\Leftarrow}e\rangle\phi$ means that some possible result of program e satisfies ϕ, and $e{\Downarrow}v$ means that v is one possible result of program e.

- $\Gamma \vdash \Phi \Longrightarrow \phi$ means "Φ entails ϕ", where Φ is a finite set of formulas.

Notation 1.1 We may use the following derived notation:

- "$\text{let}\, \overline{x}{\Leftarrow}\overline{e} \,\text{in}\, e$" for "$\text{let}\, x_1{\Leftarrow}e_1 \,\text{in}\, (\ldots(\text{let}\, x_n{\Leftarrow}e_n \,\text{in}\, e)\ldots)$"
- "$\langle \overline{x} \Leftarrow \overline{e}, e\rangle$" for "$\text{let}\, \overline{x}{\Leftarrow}\overline{e} \,\text{in}\, (\text{let}\, x{\Leftarrow}e \,\text{in}\, [\langle \overline{x}, x\rangle])$"
- "$e_1; e_2$" for "$\text{let}\, x{\Leftarrow}e_1 \,\text{in}\, e_2$", where $x \notin \text{FV}(e_2)$.

A model of EL_T consists of a quasi-topos $\mathcal{C}$, e.g. the category **Set** of sets, with a fibered monad T over $cod\colon\mathcal{C}^{\rightarrow} \rightarrow \mathcal{C}$. However, models play only a marginal role for the purposes of the paper.

Valid rules. In these models the logical rules for extensional Intuitionistic HOL and the equational rules below are sound, where $[e/x]\_$ is **substitution** of e for x in $\_$ (with suitable renaming of bound variables in $\_$):

- (let.ξ) $\dfrac{\Gamma, x{:}\,\tau_1 \vdash e_1 =_{T\tau_2} e_2}{\Gamma, c{:}\,T\tau_1 \vdash (\mathrm{let}\ x{\Leftarrow}c\ \mathrm{in}\ e_1) =_{T\tau_2} (\mathrm{let}\ x{\Leftarrow}c\ \mathrm{in}\ e_2)}$ $x \notin \mathrm{FV}(\tau_2)$
- (ass) $\Gamma \vdash \mathrm{let}\ x_2{\Leftarrow}(\mathrm{let}\ x_1{\Leftarrow}e_1\ \mathrm{in}\ e_2)\ \mathrm{in}\ e_3 =_{T\tau_3} \mathrm{let}\ x_1{\Leftarrow}c_1\ \mathrm{in}\ (\mathrm{let}\ x_2{\Leftarrow}e_2\ \mathrm{in}\ e_3)$ $x_2 \notin \mathrm{FV}(e_3)$
- (T.β) $\Gamma \vdash \mathrm{let}\ x{\Leftarrow}[e_1]\ \mathrm{in}\ e_2 =_{T\tau_2} [e_1/x]e_2$
- (T.η) $\Gamma \vdash \mathrm{let}\ x{\Leftarrow}e\ \mathrm{in}\ [x] =_{T\tau} e$

Definability results and derived rules. Modalities and evaluation predicate are definable using subset types and higher order quantifiers:

- $\Gamma, c{:}\,T\tau \vdash ([x{\Leftarrow}c]\phi) \stackrel{\Delta}{\equiv} \exists c'{:}\,T(\{x{:}\,\tau|\phi\}).c =_{T\tau} \mathrm{let}\ x'{\Leftarrow}c'\ \mathrm{in}\ [i(x')]$
  where $\Gamma, x'{:}\,\{x{:}\,\tau|\phi\} \vdash i(x'){:}\,\tau$ is the inclusion of $\{x{:}\,\tau|\phi\}$ into τ
- $\Gamma, c{:}\,T\tau, x{:}\,\tau \vdash (c{\Downarrow}x) \stackrel{\Delta}{\equiv} \forall X{:}\,\Omega^\tau.([x{\Leftarrow}c]X(x)) \supset X(x)$
  where Ω is the type of truth values
- $\Gamma, c{:}\,T\tau \vdash (\langle x{\Leftarrow}c\rangle\phi) \stackrel{\Delta}{\equiv} \forall w{:}\,\Omega.([x{\Leftarrow}c](\phi \supset w)) \supset w.$

The following rules for necessity are derivables from the equational rules for computational types (in extensional Intuitionistic HOL with subset types):

- ($\Box$-$\top^*$) $\Gamma, c{:}\,T\tau \vdash [x{\Leftarrow}c]\top$
- ($\Box$-$\Longrightarrow$) $\dfrac{\Gamma, x{:}\,\tau \vdash \Phi, \phi \Longrightarrow \psi}{\Gamma, c{:}\,T\tau \vdash \Phi, [x{\Leftarrow}c]\phi \Longrightarrow [x{\Leftarrow}c]\psi}$ $x \notin \mathrm{FV}(\Phi)$
- ($\Box$-T) $\dfrac{\Gamma, x{:}\,\tau_1 \vdash e{:}\,\tau_2 \quad \Gamma, y{:}\,\tau_2 \vdash \phi\ \mathrm{prop}}{\Gamma, c{:}\,T\tau_1 \vdash [x{\Leftarrow}c]([e/y]\phi) \Longrightarrow [y{\Leftarrow}(\mathrm{let}\ x{\Leftarrow}c\ \mathrm{in}\ [e])]\phi}$
- ($\Box$-$=$) $\Gamma, c{:}\,T\tau_1 \vdash [x{\Leftarrow}c](e_1 =_{\tau_2} e_2) \Longrightarrow (\mathrm{let}\ x{\Leftarrow}c\ \mathrm{in}\ [e_1]) =_{T\tau_2} (\mathrm{let}\ x{\Leftarrow}c\ \mathrm{in}\ [e_2])$ $x \notin \mathrm{FV}(\tau_2)$
- ($\Box$-η) $\Gamma, x{:}\,\tau \vdash \phi \Longrightarrow [x{\Leftarrow}[x]]\phi$

Additional rules. In some cases one wants to consider additional axioms for necessity, which hold only under further assumptions about T (see [Mog94b]). Here is a sample of these axioms:

- if T preserves subobjects, then
  ($\Box$-μ) $\Gamma, c{:}\,T^2\tau \vdash [y{\Leftarrow}c]([x{\Leftarrow}y]\phi) \Longrightarrow [x{\Leftarrow}(\mathrm{let}\ y{\Leftarrow}c\ \mathrm{in}\ y)]\phi$
- if T preserves finite intersections (of subobjects), then
  ($\Box$-$\wedge^*$) $\Gamma, c{:}\,T\tau \vdash [x{\Leftarrow}c](\phi_1 \wedge \phi_2) \Longleftrightarrow ([x{\Leftarrow}c]\phi_1) \wedge ([x{\Leftarrow}c]\phi_2)$
- ($\Box$-$\supset^*$) $\Gamma, c{:}\,T\tau \vdash [x{\Leftarrow}c](\phi_1 \supset \phi_2) \Longleftrightarrow \phi_1 \supset ([x{\Leftarrow}c]\phi_2)$ $x \notin \mathrm{FV}(\phi_1)$
- if T preserves inverse images (of subobjects), then
  ($\Box$-T^*) $\dfrac{\Gamma, x{:}\,\tau_1 \vdash e{:}\,\tau_2 \quad \Gamma, y{:}\,\tau_2 \vdash \phi\ \mathrm{prop}}{\Gamma, c{:}\,T\tau_1 \vdash [x{\Leftarrow}c]([e/y]\phi) \Longleftrightarrow [y{\Leftarrow}(\mathrm{let}\ x{\Leftarrow}c\ \mathrm{in}\ [e])]\phi}$
- if the unit η of T is monic, then
  ($\Box$-η^*) $\Gamma, x{:}\,\tau \vdash \phi \Longleftrightarrow [x{\Leftarrow}[x]]\phi$

When we need to assume some of these additional axioms, it will be explicitly said.

2 Representing $VTLoE$ into EL_T

2.1 A brief summary of $VTLoE$

We recall the features of $VTLoE$ relevant for the purposes of this paper, a complete account can be found in [MT92, HMSTar] (we have not considered $VTLoE$ classes). Talcott and Mason (see [Tal93]) are currently investigating *localized* semantics of $VTLoE$, which differ in the interpretation of universal quantification. Our translation of $VTLoE$ in EL_T can validate axioms that are not true in the original semantics of $VTLoE$, but are valid in the localized semantics.

2.1.1 Syntax and operational semantics of λ_{mk}

λ_{mk} is an untyped variant of Standard ML with references. To keep the presentation as simple as possible, we consider a variant of λ_{mk} without atoms and pairs.

x	$\in \mathsf{X}$	$:= \ldots$	variables
f_n	$\in \mathsf{F}_n$	$:= \ldots$	n-ary primitive operations
v	$\in \mathsf{V}$	$:= x \mid \lambda x.e$	values
e	$\in \mathsf{E}$	$:= v \mid e_0 e_1 \mid f_n(e_1, \ldots, e_n)$	expressions

The primitive *memory* operations are: cell creation and initialization $mk(x)$, dereferencing $get(x)$ and assignment $set(x, y)$ (written also $x{:=}\, y$). One may consider other primitive operations such as: test whether a value is a cell $cell(x)$ and test equality of cells $eq(x, y)$. In the sequel we need only the primitive memory operations. Moreover, we may use some *standard notation* from call-by-value λ-calculus, e.g. we write "let$\{x = e_1\}e_2$" for "$(\lambda x.e_2)e_1$".

Notation 2.1 For defining the operational semantics and the satisfaction relation [HMSTar] introduces memory contexts $\mu \in \mathsf{M}$, which provide a syntactic representation for *states*, and value substitutions $\sigma \in \mathsf{S}$, which correspond to environments.

- A **value substitution** $\sigma \in \mathsf{S}$ is a function from a finite subset of X to V.

  We write $\mathrm{DV}(\sigma)$ for the domain of σ (the declared variables) and $\mathrm{FV}(\sigma)$ for the set of variables free in some element of the codomain of σ. We write $[v/x]e$ for the expression obtained by substituting x with v in e, and e^σ for the expression obtained by applying the value substitution σ in parallel.

- A **memory context** $\mu \in \mathsf{M}$ is a sequence $\{x_1{:=}\, v_1, \ldots, x_n{:=}\, v_n\}$ s.t. the variables x_i are distinct and $\mathrm{FV}(v_i) \subseteq \{x_1, \ldots, x_n\}$ for each $i \leq n$.

  We write $\mathrm{DV}(\mu)$ for $\{x_1, \ldots, x_n\}$ and $\mu(x_i)$ for v_i. Intuitively, each $x \in \mathrm{DV}(\mu)$ corresponds to a location, whose stored value is $\mu(x)$. When $\mathrm{FV}(v) \subseteq \mathrm{DV}(\mu)$, we write $\mu\{x{:=}\, v\}$ for the memory context μ' s.t. $\mu'(x) = v$ and $\mu'(x') \simeq \mu(x')$ ($\simeq$ is Kleene's equality) for any $x' \neq x$.

- A **description** is a pair $\mu; e$, where $\mu \in \mathsf{M}$ and $e \in \mathsf{E}$ s.t. $\mathrm{FV}(e) \subseteq \mathrm{DV}(\mu)$. A **value description** $\mu; v$ is a description in which v is a value.

We give a big step operational semantics $\Rightarrow$ for λ_{mk}, i.e. a (functional) relation between descriptions and value descriptions, which is *equivalent* to the small step operational semantics $\mapsto$ in [HMSTar].

Definition 2.2 (Operational semantics) *$\Rightarrow$ is the smallest relation closed under the rules:*

(val) $\mu_0; v \Rightarrow \mu_0; v$

(app) $$\frac{\mu_0; e_0 \Rightarrow \mu_1; (\lambda x.e) \quad \mu_1; e_1 \Rightarrow \mu_2; v_1 \quad \mu_2; [v_1/x]e \Rightarrow \mu_3; v}{\mu_2; (e_0 e_1) \Rightarrow \mu_3; v}$$

(mk) $$\frac{\mu_0; e \Rightarrow \mu_1; v}{\mu_0; mk(e) \Rightarrow \mu_1\{x{:=}\, v\}; x}\ x \notin \mathrm{DV}(\mu)$$

(get) $$\frac{\mu_0; e \Rightarrow \mu_1; x}{\mu_0; get(e) \Rightarrow \mu_1; \mu_1(x)}$$

$$(set)\ \frac{\mu_0; e_0 \Rightarrow \mu_1; x \quad \mu_1; e_1 \Rightarrow \mu_2; v}{\mu_0; set(e_0, e_1) \Rightarrow \mu_2\{x := v\}; v}$$

We write $\mu; e \Downarrow$, when $\mu; e \Rightarrow \mu'; v$ for some value description $\mu'; v$. **Operational equivalence** *is the congruence $\cong$ over E s.t. $e_0 \cong e_1 \overset{\Delta}{\iff} \emptyset; C[e_0] \Downarrow$ iff $\emptyset; C[e_1] \Downarrow$ for every closing context $C[\_]$.*

2.1.2 Syntax and satisfaction relation of $VTLoE$

$VTLoE$ is basically First Order Logic on top of λ_{mk} extended with a modality similar to the necessity modality of Dynamic Logic.

$$\phi \in \mathsf{W} ::= e_1 \cong e_2 \mid \{x \Leftarrow e\}\phi \mid \phi_1 \supset \phi_2 \mid \forall x.\phi \quad \text{contextual assertions}$$

Remark 2.3 The original definition of contextual assertion uses allows formulas of the form $U[\![\phi]\!]$, where U is a *univalent context*. However, one may restrict without loosing expressiveness to univalent contexts of the form let$\{x = e\}_$, in this case we write "$\{x \Leftarrow e\}\phi$" for "$U[\![\phi]\!]$".

Other logical connectives and quantifies are defined as in classical logic using $\supset$, $\forall$ and $\bot$, where $\bot$ is some unsatisfiable assertion, e.g. $(\lambda x, y.x) \cong (\lambda x, y.y)$.

$VTLoE$ differs from Dynamic Logic for two aspects: the underlying programming language is λ_{mk} (instead of a simple while-language), it is based on predicate (rather than propositional) logic.

Satisfaction $\models$ is a relation on the set of triples (μ, σ, ϕ) s.t. $\mathrm{FV}(\sigma) \subseteq \mathrm{DV}(\mu)$ and $\mathrm{FV}(\phi) \subseteq \mathrm{DV}(\sigma)$. We write $\mu \models \phi[\sigma]$ for $(\mu, \sigma, \phi) \in \models$.

Definition 2.4 (Satisfaction relation) *$\mu \models \phi[\sigma]$ is defined (using the operational semantics) by structural induction on ϕ:*

- $\mu \models (e_1 \cong e_2)[\sigma] \overset{\Delta}{\iff} \mu; v(e_0^\sigma \Downarrow)$ *iff* $\mu; v(e_1^\sigma \Downarrow)$ *for every* $v \in \mathsf{V}$ *s.t.* $\mathrm{FV}(v) \subseteq \mathrm{DV}(\mu)$
- $\mu \models (\{x \Leftarrow e\}\phi)[\sigma] \overset{\Delta}{\iff} \mu; e \Rightarrow \mu'; v$ *and* $\mu' \models \phi[\sigma\{v/x\}]$ *for some* $\mu' \in \mathsf{M}$ *and* $v \in \mathsf{V}$
- $\mu \models (\phi_1 \supset \phi_2)[\sigma] \overset{\Delta}{\iff} \mu \models \phi_1[\sigma]$ *implies* $\mu \models \phi_2[\sigma]$
- $\mu \models (\forall x.\phi)[\sigma] \overset{\Delta}{\iff} \mu \models \phi[\sigma\{v/x\}]$ *for every* $v \in \mathsf{V}$ *s.t.* $\mathrm{FV}(v) \subseteq \mathrm{DV}(\mu)$

We say that ϕ is **valid** *($\models \phi$) $\overset{\Delta}{\iff} \mu \models \phi[\sigma]$ for every $\mu \in \mathsf{M}$ and $\sigma \in \mathsf{S}$ s.t. $\mathrm{FV}(\sigma) \subseteq \mathrm{DV}(\mu)$ and $\mathrm{FV}(\phi) \subseteq \mathrm{DV}(\sigma)$.*

Remark 2.5 The definition of the satisfaction relation differs from [HMSTar] only in irrelevant details, due to the use of a big step operational semantics. [HMSTar] proves a basic (but not trivial) consistency result between validity and operational equivalent, i.e. $\models e_0 \cong e_1$ iff $e_0 \cong e_1$

The logical rules validated by the above semantics are:

- (R) $\vdash e_0 \cong e_1 \Longrightarrow R[e_0] \cong R[e_1]$ R reduction context

  in first approximation a reduction context is a context of the form let$\{x = _\}e$
- (ucx.eq) $\vdash e_0 \cong e_1, \{x = e_0\}\phi \Longrightarrow \{x = e_1\}\phi$

  this axiom is valid only in the localized semantics of [Tal93]
- (ca.i) $\dfrac{\vdash \phi}{\vdash U[\![\phi]\!]}$ U univalent context

  univalent contexts could be identified with contexts of the form let$\{x = e\}_$
- (ca.ii) $\vdash U[\![e_0 \cong e_1]\!] \Longrightarrow U[e_0] \cong U[e_1]$
- (ca.iii) $\vdash U_0[\![U_1[\![\phi]\!]]\!] \Longrightarrow U_0[U_1][\![\phi]\!]$
- (con.triv) $\vdash U[\![\bot]\!] \Longrightarrow U[\![\phi]\!]$

- (con.not) $\vdash U[\![\neg\phi]\!] \iff (U[\![\bot]\!] \vee \neg U[\![\phi]\!])$
- (con.imp) $\vdash U[\![\phi_1 \supset \phi_2]\!] \iff (U[\![\phi_1]\!] \supset U[\![\phi_2]\!])$
- (con.$\forall$) $\vdash U[\![\forall x.\phi]\!] \Longrightarrow \forall x.U[\![\phi]\!]$ $x \notin \mathrm{FV}(U)$
- ($\Longrightarrow$.$\forall$) $\dfrac{\vdash \Phi \Longrightarrow \phi}{\vdash \Phi \Longrightarrow \forall x.\phi}$ $x \notin \mathrm{FV}(\Phi)$
- ($\Longrightarrow$.{_}) $\dfrac{\vdash \Phi \Longrightarrow \{x = e\}\phi_0 \quad \vdash \phi_0 \Longrightarrow \phi_1}{\vdash \Phi \Longrightarrow \{x = e\}\phi_1}$

2.2 Translation of λ_{mk} in $ML_T(\Sigma)$

According to the monadic approach (see [Mog91]) the semantics of a programming language should factor through the metalanguage $ML_T(\Sigma)$ for computational monads over a suitable signature. In the case of λ_{mk} we use the following signature Σ (used also for translating $VTLoE$ into $EL_T(\Sigma)$):

- primitive types

  L locations, V values

- primitive functions

  $ref: V \to TL$ cell creation and initialization,
  $lkp: L \to TV$ dereferencing,
  $upd: L, V \to T1$ assignment,
  $in: (L + (TV)^V) \to V$, $out: V \to (L + (TV)^V)$
  describing the isomorphism $V \cong L + (TV)^V$;

  auxiliary constant

  $\bot_V : TV$ diverging computation.

We define a translation $(\_)^*$ mapping an expression e of λ_{mk} with n free variables into a term e^* of $ML_T(\Sigma)$ with the same free variables and type TV (when the free variables have type V).

Definition 2.6 (Translation) *The term e^* is defined by structural induction on e:*

$$
\begin{array}{rcl}
x^* & \triangleq & [x] \\
(\lambda x.e)^* & \triangleq & [in_2(\lambda x{:}V.e^*)] \\
(e_0 e_1)^* & \triangleq & \mathrm{let}\ x_0, x_1 \Leftarrow e_0^*, e_1^*\ \mathrm{in}\ \mathit{case}\ out(x_0)\ \mathit{of} \\
 & & in_1(l) \Rightarrow \bot_V \mid in_2(f) \Rightarrow f x_1 \\
mk(e)^* & \triangleq & \mathrm{let}\ x \Leftarrow e^*\ \mathrm{in}\ \mathrm{let}\ l \Leftarrow ref(x)\ \mathrm{in}\ [in_1(l)] \\
get(e)^* & \triangleq & \mathrm{let}\ x \Leftarrow e^*\ \mathrm{in}\ \mathit{case}\ out(x)\ \mathit{of} \\
 & & in_1(l) \Rightarrow lkp(l) \mid in_2(f) \Rightarrow \bot_V \\
set(e_0, e_1)^* & \triangleq & \mathrm{let}\ x_0, x_1 \Leftarrow e_0^*, e_1^*\ \mathrm{in}\ \mathit{case}\ out(x_0)\ \mathit{of} \\
 & & in_1(l) \Rightarrow upd(l, x_1); [x_1] \mid in_2(f) \Rightarrow \bot_V
\end{array}
$$

where $in_1: L \to V$ and $in_2: (TV)^V \to V$ are the restrictions of in to each of the two components of $L + (TV)^V$. $\bot_V$ is used to indicate non-termination because of type error.

Proposition 2.7 (Computational Adequacy) *There is a model of $ML_T(\Sigma)$, which gives a computationally adequate model of λ_{mk} via $(\_)^*$.*

Proof The simplest way to give a denotational model of λ_{mk} and prove computational adequacy for it is by translating λ_{mk} into FPC[1] (see [FP93]), and by showing that the operational semantics

[1] FPC is basically the pure functional part of Standard ML.

of λ_{mk} can be reduced to that of FPC. Therefore, any computational adequate model of FPC, e.g. the interpretation in the category of cpos will induce (via the translation) a computationally adequate model of λ_{mk}. For instance, in the induced model in the category of cpos the interpretation of primitive types (L and V) and computational types TX is given by the least solution to the following domain equations:

$$\begin{aligned} L &= N \cong 1 + N \text{ the set of natural numbers} \\ V &= L + (V \to TV) \\ S &= \Sigma m{:}\, N.V^m \cong 1 + (S \times V) \\ TX &= S \to (X \times S)_\perp \end{aligned}$$

The interpretation of $ref{:}\, V \to TL$ is given by $ref(v) \stackrel{\Delta}{\equiv} \lambda\langle m, s\rangle{:}\, S.\langle m, \langle m+1, s[m{:}= v]\rangle\rangle$, while that of $lkp{:}\, L \to TV$ and $upd{:}\, L, V \to T1$ is a variation of that for the monad of side-effects. ∎

2.3 Translation of $VTLoE$ in $EL_T(\Sigma)$

The translation $(\_)^*$ establishes a correspondence between expression of λ_{mk} with n free variables and function from V^n to TV in $ML_T(\Sigma)$. This correspondence can be extended to other syntactic categories of $VTLoE$ as follows:

- a memory context μ *creating* m locations is mapped to a computation μ^* of type $T(L^m)$
- a value substitution σ associating n variables to value expressions *involving* m locations is mapped to a function σ^* from L^m to V^n
- a contextual assertion ϕ with n free variables is mapped to a predicate ϕ^* over $T(V^n)$.

In this way the operationally defined satisfaction relation $\mu \models \phi[\sigma]$ can be expressed in $EL_T(\Sigma)$ by the closed formula $\phi^*(\text{let } \bar{l} \Leftarrow \mu^* \text{ in } [\sigma^*(\bar{l})])$. The above correspondence suggests a definitional extension of EL_T, which *introduces* the logical operators of $VTLoE$.

Definition 2.8 *We define the following (derived) predicate constructors:*

- $\_ \cong \_{:}\, (TY)^X, (TY)^X \to \Omega^{TX}$

$$(\cong)\ \frac{x{:}\, X \vdash e_i{:}\, TY \quad (i = 1, 2)}{c{:}\, TX \vdash (x{:}\, X.e_1 \cong e_2)(c) \stackrel{\Delta}{\Longleftrightarrow} \langle x \Leftarrow c, e_1\rangle =_{T(X\times Y)} \langle x \Leftarrow c, e_2\rangle}$$

  where $\langle x \Leftarrow c, e\rangle$ *stands for* $\text{let } x \Leftarrow c \text{ in } (\text{let } y \Leftarrow e \text{ in } [\langle x, y\rangle])$

- $\{\_\}\_{:}\, (TY)^X, \Omega^{T(X\times Y)} \to \Omega^{TX}$

$$(\{\ \})\ \frac{x{:}\, X \vdash e{:}\, TY \quad c'{:}\, T(X\times Y) \vdash \phi(c')}{c{:}\, TX \vdash (\{x{:}\, X.e\}\phi)(c) \stackrel{\Delta}{\Longleftrightarrow} \phi(\langle x \Leftarrow c, e\rangle)}$$

- $\_ \odot^* \_{:}\, \Omega^{TX}, \Omega^{TX} \to \Omega^{TX}$, *where* $\odot$ *is a binary logical connective*

$$(\odot^*)\ \frac{c{:}\, TX \vdash \phi_i(c) \quad (i = 1, 2)}{c{:}\, TX \vdash (\phi_1 \odot^* \phi_2)(c) \stackrel{\Delta}{\Longleftrightarrow} \phi_1(c) \odot \phi_2(c)}$$

- $\forall^*_Y{:}\, \Omega^{T(X\times Y)} \to \Omega^{TX}$

$$(\forall^*_Y)\ \frac{c'{:}\, T(X\times Y) \vdash \phi(c')}{c{:}\, TX \vdash (\forall^*_Y \phi)(c) \stackrel{\Delta}{\Longleftrightarrow} (\forall c'{:}\, T(X\times Y).(\text{let } x, y \Leftarrow c' \text{ in } [x]) =_{TX} c \supset \phi(c'))}$$

We extend the translation $(\_)^*$ of λ_{mk} to the contextual assertions of $VTLoE$.

Definition 2.9 (Translation) *The predicate $\phi^*_{\bar{x}}$, where $\bar{x}$ is a list of variables including those in* $\mathrm{FV}(\phi)$*, is defined (via the derived operators) by induction on the structure of ϕ:*

$$\begin{array}{|rcl|}\hline (e_1 \cong e_2)^*_{\bar{x}} & \stackrel{\Delta}{=} & (\bar{x}{:}\,V.e_1^* \cong e_2^*) \\ (\{x \Leftarrow e\}\phi)^*_{\bar{x}} & \stackrel{\Delta}{=} & \{\lambda\bar{x}{:}\,V.e^*\}\phi^*_{\bar{x},x} \\ (\phi \supset \psi)^*_{\bar{x}} & \stackrel{\Delta}{=} & (\phi^*_{\bar{x}}) \supset^* (\psi^*_{\bar{x}}) \\ (\forall x.\phi)^*_{\bar{x}} & \stackrel{\Delta}{=} & \forall^*_V(\phi^*_{\bar{x},x}) \\ \hline \end{array}$$

Remark 2.10 In $VTLoE$ falsity is defined in terms of $\cong$. In EL_T one could proceed similarly by defining $\bot^*(c) \stackrel{\Delta}{\Longleftrightarrow} (x{:}\,X.e_0 \cong e_1)(c)$ for suitable expressions e_i, but there are other plausible definitions: $[x{\Leftarrow}c]\bot$ and $\bot$. These formulas are related by the following entailments:

$$c{:}\,TX \vdash \bot \Longrightarrow [x{\Leftarrow}c]\bot \Longrightarrow (x{:}\,X.e_1 \cong e_2)(c)$$

$\bot^*(c) \stackrel{\Delta}{\Longleftrightarrow} \bot$ is the most natural choice, and it is consistent with the definition of $\odot^*$. In $VTLoE$ other useful logical operators, e.g. the S4-like modality $\boxed{\cdot}$, are definable from those above. Similarly, in EL_T $\boxed{\cdot}^*{:}\,\Omega^{TX} \to \Omega^{TX}$ can be defined as $(\boxed{\cdot}^*\phi)(c) \stackrel{\Delta}{\Longleftrightarrow} \forall f{:}\,(T1)^X.\phi(\langle x \Leftarrow c, f(x)\rangle)$.

Remark 2.11 The correspondence between memory contexts μ creating m locations and closed terms c of type $T(L^m)$ is a bit loose, i.e. there are terms which do not correspond to a memory context. For instance, c may return an m-uple of locations with repetitions, or it may diverge. On this basis, it seems more accurate to map contextual assertions into formulas over a subset of TX, by *relativizing* the translation w.r.t. a predicate $good_X{:}\,\Omega^{TX}$ s.t. $good_{V^n}(\mathrm{let}\ \bar{l}{\Leftarrow}\mu^*\ \mathrm{in}\ [\sigma^*(\bar{l})])$ for every $\mu \in \mathsf{M}$ and $\sigma \in \mathsf{S}$ with $\mathrm{FV}(\sigma) \subseteq \mathrm{DV}(\mu)$, e.g.

$$c{:}\,\{c{:}\,TX | good_X(c)\} \vdash (\{x{:}\,X.e\}\phi)(c) \stackrel{\Delta}{\Longleftrightarrow} good_{X\times Y}(\langle x \Leftarrow c, e\rangle) \supset \phi(\langle x \Leftarrow c, e\rangle)$$

For deterministic languages, a plausible choice for $good(c)$ is $\langle x{\Leftarrow}c\rangle\top$, i.e. "$c$ terminates".

It seems difficult to relate the operationally defined validity to provability in $EL_T(\Sigma)$. However, one can show that ϕ^* derivable implies $\models \phi$, when ϕ is of the form $(e_1 \cong e_2)$. This follows from an internal consistency property for $\cong$ and the existence of computationally adequate models.

Proposition 2.12 (Internal Consistency) *The following birule is derivable*

$$(=\text{-}\cong)\ \frac{x{:}\,X \vdash e_1 =_{TY} e_2}{\overline{\overline{c{:}\,TX \vdash (x{:}\,X.e_1 \cong e_2)(c)}}}$$

Proposition 2.13 (Computational Adequacy) *There is a model of $EL_T(\Sigma)$, which gives a computationally adequate model of λ_{mk} via $(\_)^*$.*

Proof The model of $ML_T(\Sigma)$ given by Proposition 2.7 cannot be extended to EL_T, as the category of cpos is not a quasi-topos. But we can take a quasi-topos $\mathcal{C}$ which *contains* the category cpos as a full sub-biCCC, keep the interpretation of Σ as in cpos, and extend the interpretation of computational types using the *fibered* monad $TX = S \to (X \times S)_\bot$. A possible choice for $\mathcal{C}$ is the quasi-topos of ω-sets (see [Mog94a]), since the category of effectively given Scott domains and computable maps is a full sub-biCCC of it. ∎

Proposition 2.14 *Given a model M of $EL_T(\Sigma)$, which gives a computationally adequate model of λ_{mk} via $(\_)^*$, then $(\bar{x}{:}\,V.e_1^* \cong e_2^*)$ true in M implies $\models e_1 \cong e_2$ (in $VTLoE$).*

Proof We have the following sequence of entailments:

- $c{:}\,T(V^n) \vdash (\bar{x}{:}\,V.e_1^* \cong e_2^*)(c)$ valid in M implies, by Proposition 2.12
- $\bar{x}{:}\,V \vdash e_1^* = e_2^*$ valid in M implies, by M computationally adequate
- e_1 and e_2 operationally equivalent implies, by a result in [HMSTar]
- $\models e_1 \cong e_2$ in $VTLoE$.

∎

2.4 Provable properties of $VTLoE$ logical operators

In this section we give the most interesting properties of $VTLoE$ logical operators, which are provable in EL_T, and discuss possible mismatches between truth in the proposed semantics of $VTLoE$ and provability in EL_T. For each logical rule of $VTLoE$, we say whether its translation is derivable in EL_T, and give a corresponding derivable rule in EL_T.

Notation 2.15 We use the following shorthand for EL_T sequents: when ϕ_i $(i = 1, \ldots, n)$ and ϕ are closed terms of type Ω^{TX}, then we write $\vdash \phi_1, \ldots, \phi_n \Longrightarrow \phi$ for $c{:}TX \vdash \phi_1(c), \ldots, \phi_n(c) \Longrightarrow \phi(c)$.

- (R) $\dfrac{x{:}X \vdash e_i{:}TY \quad x{:}X, y{:}Y \vdash e{:}TZ}{\vdash (x{:}X.e_0 \cong e_1) \Longrightarrow (x{:}X.(\text{let } y{\Leftarrow}e_0 \text{ in } e) \cong (\text{let } y{\Leftarrow}e_1 \text{ in } e))}$

- (ucx.eq) $\dfrac{x{:}X \vdash e_i{:}TY}{\vdash (x{:}X.e_0 \cong e_1), (\{x{:}X.e_0\}\phi) \Longrightarrow (\{x{:}X.e_1\}\phi)} \ \phi{:}\Omega^{T(X\times Y)}$

- (ca.i) $\dfrac{x{:}X \vdash e{:}TY \quad \vdash \emptyset \Longrightarrow \phi}{\vdash \emptyset \Longrightarrow \{x{:}X.e\}\phi} \ \phi{:}\Omega^{T(X\times Y)}$

- (ca.ii) $\dfrac{x{:}X \vdash e{:}TY \quad x{:}X, y{:}Y \vdash e_i{:}TZ}{\vdash \{x{:}X.e\}(x, y{:}X{\times}Y.e_0 \cong e_1) \Longrightarrow (x{:}X.(\text{let } y{\Leftarrow}e \text{ in } e_0) \cong (\text{let } y{\Leftarrow}e \text{ in } e_1))}$

- (ca.iii) $\dfrac{x{:}X \vdash e_0{:}TY \quad x{:}X, y{:}Y \vdash e_1{:}TZ}{\vdash \{x{:}X.e_0\}\{x, y{:}X{\times}Y.e_1\}\phi \Longrightarrow \{x{:}X.\langle y \Leftarrow e_0, e_1\rangle\}\phi} \ \phi{:}\Omega^{T(X\times Y\times Z)}$

- (con.triv) $\dfrac{x{:}X \vdash e{:}TY}{\vdash \{x{:}X.e\}\bot^* \Longrightarrow \{x{:}X.e\}\phi} \ \phi{:}\Omega^{T(X\times Y)}$

  the rule (triv) is derivable when $\bot^*(c) = \bot$, but it is not derivable for the other definitions of $\bot^*$ considered in Remark 2.10, unless the translation is relativized using a predicate *good* s.t. $c{:}TX \vdash \mathit{good}(c), \bot^*(c) \Longrightarrow \bot$

- (con.not) $\dfrac{x{:}X \vdash e{:}TY}{\vdash \{x{:}X.e\}(\neg^*\phi) \Longleftrightarrow (\{x{:}X.e\}\bot^*) \vee^* (\{x{:}X.e\}\phi)} \ \phi{:}\Omega^{T(X\times Y)}$

  (con.not) is not derivable in EL_T, since it relies on classical logic

- (con.imp) $\dfrac{x{:}X \vdash e{:}TY}{\vdash \{x{:}X.e\}(\phi_1 \supset^* \phi_2) \Longleftrightarrow (\{x{:}X.e\}\phi_1) \supset^* (\{x{:}X.e\}\phi_2)} \ \phi_1, \phi_2{:}\Omega^{T(X\times Y)}$

- (con.$\forall$) $\dfrac{x{:}X \vdash e{:}TY}{\vdash \{x{:}X.e\}(\forall^*_Z\phi) \Longrightarrow \forall^*_Z(\{x, z{:}X{\times}Z.e\}\phi)} \ \phi{:}\Omega^{T(X\times Z\times Y)}$

- ($\Longrightarrow$.$\forall$) $\dfrac{\vdash (c'{:}T(X{\times}Y).\Phi(\text{let } x, y{\Leftarrow}c' \text{ in } [x])) \Longrightarrow \phi}{\vdash \Phi \Longrightarrow \forall^*_Y\phi} \ \Phi{:}\Omega^{TX},\ \phi{:}\Omega^{T(X\times Y)}$

  this is the translation of the $VTLoE$ rule ($\Longrightarrow$.$\forall$) $\dfrac{\vdash \Phi \Longrightarrow \phi}{\vdash \Phi \Longrightarrow \forall x.\phi} \ x \notin \mathrm{FV}(\Phi)$

- ($\Longrightarrow$.{_}) $\dfrac{x{:}X \vdash e{:}TY \quad \vdash \Phi \Longrightarrow \{x{:}X.e\}\phi_0 \quad \vdash \phi_0 \Longrightarrow \phi_1}{\vdash \Phi \Longrightarrow \{x{:}X.e\}\phi_1} \ \Phi{:}\Omega^{TX},\ \phi_i{:}\Omega^{T(X\times Y)}$

The rules above (unless stated otherwise) are derivable using only the three equational rules of EL_T and standard reasoning in first order logic. We give in details the more interesting derivations.

Proposition 2.16 *The rules (R) and (ucx.eq) are derivable in EL_T.*

Proof For (R) we must derive $\langle x \Leftarrow c, (\text{let}\, y{\Leftarrow}e_0\, \text{in}\, e)\rangle = \langle x \Leftarrow c, (\text{let}\, y{\Leftarrow}e_1\, \text{in}\, e)\rangle$ from $\langle x \Leftarrow c, e_0\rangle = \langle x \Leftarrow c, e_1\rangle$. This is immediate by $\langle x \Leftarrow c, (\text{let}\, y{\Leftarrow}e_i\, \text{in}\, e)\rangle = \text{let}\, x, y{\Leftarrow}\langle x \Leftarrow c, e_i\rangle\, \text{in let}\, z{\Leftarrow}e\, \text{in}\, [\langle x, z\rangle]$, which follows from the equational rules of EL_T.

For (ucx.eq) we must derive $\phi(\langle x \Leftarrow c, e_1\rangle)$ from $\langle x \Leftarrow c, e_0\rangle = \langle x \Leftarrow c, e_1\rangle$ and $\phi(\langle x \Leftarrow c, e_0\rangle)$. This is immediate by congruence. ∎

Also the rules (ca) are easy consequences of the equational rules of EL_T.

Proposition 2.17 *The rules (con.∀) and (⟹.∀) are derivable in* EL_T*.*

Proof For (con.∀) we must derive $\forall c_1{:}T(X{\times}Z).(\text{let}\, x, z{\Leftarrow}c_1\, \text{in}\, [x]) = c \supset \phi(\langle x, z \Leftarrow c_1, e\rangle)$ from $\forall c_2{:}T(X{\times}Z{\times}Y).(\text{let}\, x, z, y{\Leftarrow}c_2\, \text{in}\, [\langle x, y\rangle]) = \langle x \Leftarrow c, e\rangle \supset \phi(c_2)$. Take any $c_1{:}T(X{\times}Z)$ s.t. $(\text{let}\, x, z{\Leftarrow}c_1\, \text{in}\, [x]) = c$, and let $c_2 \stackrel{\Delta}{=} \langle x, z \Leftarrow c_1, e\rangle{:}T(X{\times}Z{\times}Y)$, then we must derive $\phi(c_2)$. For this it suffices to show that $(\text{let}\, x, z, y{\Leftarrow}c_2\, \text{in}\, [\langle x, y\rangle]) = \langle x \Leftarrow c, e\rangle$:

- $\text{let}\, x, z, y{\Leftarrow}c_2\, \text{in}\, [\langle x, y\rangle] =$ by definition of c_2
- $\text{let}\, x, z, y{\Leftarrow}\langle x, z \Leftarrow c_1, e\rangle\, \text{in}\, [\langle x, y\rangle] =$ by definition of $\langle x \Leftarrow \_, \_\rangle$
- $\text{let}\, x, z, y{\Leftarrow}(\text{let}\, x, z{\Leftarrow}c_1\, \text{in let}\, y{\Leftarrow}e\, \text{in}\, [\langle x, z, y\rangle])\, \text{in}\, [\langle x, y\rangle] =$ by the equational rules of EL_T
- $\text{let}\, x, z{\Leftarrow}c_1\, \text{in let}\, y{\Leftarrow}e\, \text{in}\, [\langle x, y\rangle] =$ by the equational rules of EL_T, since $z \notin \text{FV}(e)$
- $\text{let}\, x{\Leftarrow}(\text{let}\, x, z{\Leftarrow}c_1\, \text{in}\, [x])\, \text{in let}\, y{\Leftarrow}e\, \text{in}\, [\langle x, y\rangle] =$ by the assumption on c_1
- $\text{let}\, x{\Leftarrow}c\, \text{in let}\, y{\Leftarrow}e\, \text{in}\, [\langle x, y\rangle] =$ by definition of $\langle x \Leftarrow \_, \_\rangle$
- $\langle x \Leftarrow c, e\rangle$.

For (⟹.∀) we must derive $\forall c'{:}T(X{\times}Y).(\text{let}\, x, y{\Leftarrow}c'\, \text{in}\, [x]) = c \supset \phi(c')$ from $\Phi(c)$ for any $c{:}TX$, under the assumption that $c'{:}T(X{\times}Y) \vdash \Phi(\text{let}\, x, y{\Leftarrow}c'\, \text{in}\, [x]) \Longrightarrow \phi(c')$. Take any $c'{:}T(X{\times}Y)$ s.t. $(\text{let}\, x, y{\Leftarrow}c'\, \text{in}\, [x]) = c$, then we must derive $\phi(c')$.

- $\Phi(\text{let}\, x, y{\Leftarrow}c'\, \text{in}\, [x])$ by assumption on c' and the hypothesis $\Phi(c)$
- $\phi(c')$ by cut with the assumption $c'{:}T(X{\times}Y) \vdash \Phi(\text{let}\, x, y{\Leftarrow}c'\, \text{in}\, [x]) \Longrightarrow \phi(c')$.

∎

2.5 Open issues

The problem regarding $VTLoE$ is how to construct *good models*, i.e. models capable of validating (most of) the non-logical axioms. We have not been able to adapt models based on functor categories for Algol-like languages (see[Ole85, OT92, OT93]) or for dynamic creation of names (see [Mog89, PS93]). What follows briefly explains the source of the difficulties.

A suitable functor category for modeling Algol-like languages and languages with dynamic creation of names is $\mathbf{Cpo}^{\mathcal{I}}$, where $\mathcal{I}$ the the category of finite cardinals and injective maps, while **Cpo** is the category of posets with sups of ω-chains and monotonic maps preserving these sups. $\mathbf{Cpo}^{\mathcal{I}}$ shares with **Cpo** all categorical properties which makes it suitable for denotational semantics: limits, colimits, exponentials, **Cpo**-enrichment, lifting. Moreover, in $\mathbf{Cpo}^{\mathcal{I}}$ one can interpret the type L of locations as the inclusion functor of $\mathcal{I}$ into **Cpo** (which represents a substantial improvement over the simple-minded interpretation of L as the natural number object N). Given an O-monad T over **Cpo** (i.e. a monad in the enriched sense) and an object $V \in \mathbf{Cpo}$ (of storable values) one can define an O-monad T_V (over $\mathbf{Cpo}^{\mathcal{I}}$) for computations with local variables of type V:

- $T_V X m \stackrel{\Delta}{=} V^m {\Rightarrow} T(Xm{\times}V^m)$
- $T_V X(f{:}m \hookrightarrow m+n)(c{:}T_V Xm)([s_m, s_n]{:}V^{m+n}) \stackrel{\Delta}{=} \text{let}_T\, \langle x, s'_m\rangle{\Leftarrow}c(s_m)\, \text{in}\, [\langle Xfx, [s'_m, s_n]\rangle]_T$, where $[s_m, s_n]{:}V^{m+n}$ is obtained by *merging* $s_m{:}V^m$ and $s_n{:}V^n$. For simplicity, we have taken f to be the inclusion of m into $m+n$.

Given an object $V \in \mathbf{Cpo}^{\mathcal{I}}$ one could define, by analogy with the above definition, a more complex O-monad T'_V for computations which dynamically create variables of type V.

Therefore, to model λ_{mk} one would like to find a $V \in \mathbf{Cpo}^{\mathcal{I}}$ s.t. $V \cong V \Rightarrow T'_V V$, i.e. one would like to define V and T'_V by mutal recursion. However, this cannot be done by the standard technique of embedding-projection, because U embedded in V does not imply that $T'_U X$ is embedded in $T'_V X$. The problem shows up already for the simpler monad T_V. In fact, an embedding $e: U \to V$ in **Cpo** induces an obvious embedding $e_{X,m}: T_U Xm \to T_V Xm$ (in **Cpo**), but $e_{X,m}$ is **not natural** in m (though the corresponding projection $e^R_{X,m}$ is).

3 Representing other program logics

3.1 The modal μ-calculus

The modal μ-calculus and Minimal Modal Logic MML (see [Stiar]) are representative of various modal logics of interest for expressing and proving properties of dynamic systems. For simplicity, we consider only MML, since the modal μ-calculus extends MML with logical operators definable in HOL (and therefore in EL_T). MML is an *endogenous* logic (unlike $VTLoE$), therefore the syntax of assertions does not refer to a programming language. However, the general pattern of translation into EL_T is similar to that used for $VTLoE$, i.e. assertions are mapped into predicates over computational types.

In MML assertions are built from a set of primitive assertions P using modalities depending on a fixed set A of *actions*.

$$\phi \in \mathsf{W} ::= P \mid \phi_0 \wedge \phi_1 \mid \neg\phi \mid [a]\phi \quad \text{assertions}$$

The interpretation of assertions depends on a labeled transition system $(S, \to)$ s.t. $\to \subseteq S \times A \times S$.

Definition 3.1 (Standard Interpretation) *Given a LTS $(S, \to)$ (and an interpretation of primitive assertions), assertions are interpreted by subsets of S, according to the following inductive definition:*

$$\begin{array}{|rcl|}\hline [\![\phi_0 \wedge \phi_1]\!] & \stackrel{\Delta}{=} & [\![\phi_0]\!] \cap [\![\phi_1]\!] \\ [\![\neg\phi]\!] & \stackrel{\Delta}{=} & \{s: S \mid s \notin [\![\phi]\!]\} \\ [\![[a]\phi]\!] & \stackrel{\Delta}{=} & \{s: S \mid \forall s': S. s \stackrel{a}{\to} s' \supset s' \in [\![\phi]\!]\} \\ \hline \end{array}$$

The logical rules validated by the above interpretations are (besides the usual propositional rules):

- $(\Box.\top) \vdash [a]\top$

  $(\Box.\wedge) \vdash [a](\phi_0 \wedge \phi_1) \Longleftrightarrow ([a]\phi_0) \wedge ([e]\phi_1)$

- $(\Box. \Longrightarrow) \dfrac{\vdash \phi_0 \Longrightarrow \phi_1}{\vdash [a]\phi_0 \Longrightarrow [a]\phi_1}$

3.1.1 Translation of MML in $EL_T(\Sigma)$

By analogy with $VTLoE$, we translate assertions of MML into predicates over a computational type, e.g. $T0$. However, in this case the signature Σ for EL_T is not suggested by a programming language, because MML is endogenous:

- primitive types

  A actions

- primitive logical operators

  $\Box_X: \Omega^X, \Omega^{A \times TX} \to \Omega^{TX}$ box.

Definition 3.2 (Translation) *The translation ϕ^* is defined by structural induction:*

$$\begin{array}{|rcl|}\hline (\phi_0 \wedge \phi_1)^*(c) & \stackrel{\Delta}{=} & \phi_0^*(c) \wedge \phi_1^*(c) \\ (\neg\phi)^*(c) & \stackrel{\Delta}{=} & \neg\phi^*(c) \\ ([a]\phi)^*(c) & \stackrel{\Delta}{=} & \Box_0(\top, \lambda a'\colon A, c'\colon T0.a = a' \supset \phi^*(c'))(c) \\ \hline \end{array}$$

The set-theoretic model corresponding to the standard interpretation is given by taking the monad $TX = \nu X'.\mathcal{P}_{fin}(X + (A\times X'))$, so $T0$ is the set of finitely branching synchronization trees (up to strong bisimulation), while $\Box_X(\phi,\psi)(c)$ is $\forall u\colon X + (A\times TX).u \in c \supset$ case u of $\phi \mid \psi$. Given a finitely branching LTS $(S, \rightarrow)$, it is easy to show that for any state $s \in S$ and assertion ϕ of MML one has $s \in [\![\phi]\!]$ iff the synchronization tree $t(s) \in T0$, obtained by unfolding the LTS starting from s, satisfies ϕ^*. In fact, the set-theoretic model is an instance of a general model construction. Given a monad T (s.t. any functor $T(X + (A\times\_))$ has a final co-algebra):

- take the monad $T' = FT$, where F is the monad transformer $FTX = \nu X'.T(X + (A\times X'))$,
- then (using necessity for T) define $\Box$ for T' as $\Box_X(\phi,\psi)(c) \stackrel{\Delta}{\Longleftrightarrow} [u{\Leftarrow}c]_T$case u of $\phi \mid \psi$.

In this model based on T' all rules of MML are valid, provided necessity for T satisfies ($\Box$-$\wedge^*$).

Remark 3.3 LTSs are not so adequate to model concurrent languages like CCS with value passing or the π-calculus. On the other hand, the translation of MML into EL_T can be easily modified to cope with modal logics suitable for these languages. In particular, the general construction of a model of $EL_T(\Sigma)$ can be easily adapted, by replacing the monad transformer F with one of the form $F_H TX = \nu X'.T(X + HX')$ (for some suitable H). For instance, in the case of value passing one should take $HY = (A\times V\times Y) + (A\times Y^V)$, where A is the set of channels and V is the set of values transmitted along channels.

3.1.2 Open issues

Another important class of program logics, related to modal logics, is the family of temporal logics (see [Stiar]). systems. The main difference between temporal and modal logics is that in the formers an assertion is interpreted by a set of *runs* (i.e. complete paths in the LTS), while in the latters it is interpreted by a set of states. We have not been able to represent temporal logics into EL_T, since the notion of run does not have any obvious counterpart.

3.2 Hoare logic

Hoare logic HL (for while-languages) is the best known and one of the simplest *exogenous* program logics. However, when one tries to extend HL to more complex programming languages, its semantics and its rules may become rather subtle. Since we want to investigate the general pattern of the translation(s) from HL into EL_T, we consider a simplified version without while.

In HL there are two syntactic categories, programs e and assertions ϕ:

$$\begin{array}{lll} \phi \in \mathsf{W} ::= P \mid \neg\phi \mid \phi_1 \wedge \phi_2 & & \text{assertions} \\ e \in \mathsf{E} ::= \alpha \mid e_0; e_1 \mid if(\phi, e_0, e_1) & & \text{programs} \end{array}$$

and on top of them one has entailments $\phi_0 \Longrightarrow \phi_1$ and Hoare triples $\{\phi_0\}e\{\phi_1\}$.

Definition 3.4 (Standard Interpretation) *Given a set S of* states *(and an interpretation of primitive programs α and assertions P), a program e is interpreted by a binary relation $[\![e]\!]$ on S and an assertion ϕ is interpreted by a (decidable) subset $[\![\phi]\!]$ of S:*

$$\begin{array}{|rcl|}\hline [\![\phi_0 \wedge \phi_1]\!] & \stackrel{\Delta}{=} & [\![\phi_0]\!] \cap [\![\phi_1]\!] \\ [\![\neg\phi]\!] & \stackrel{\Delta}{=} & \{s\colon S \mid s \notin [\![\phi]\!]\} \\ \hline [\![e_0; e_1]\!] & \stackrel{\Delta}{=} & [\![e_0]\!]; [\![e_1]\!] \text{ relational composition} \\ [\![if(\phi, e_0, e_1)]\!] & \stackrel{\Delta}{=} & ([\![\phi]\!]\times S) \cap [\![e_0]\!]) \cup ([\![\neg\phi]\!]\times S) \cap [\![e_1]\!]) \\ \hline \end{array}$$

while the top level judgements are interpreted by truth values:

$$\begin{array}{|rcl|}\hline \phi_0 \Longrightarrow \phi_1 & \stackrel{\Delta}{\Longleftrightarrow} & [\![\phi_0]\!] \subseteq [\![\phi_1]\!] \\ \{\phi_0\}e\{\phi_1\} & \stackrel{\Delta}{\Longleftrightarrow} & \forall s_0, s_1 \colon S.(s_0 \in [\![\phi_0]\!] \land s_0[\![e]\!]s_1) \supset s_1 \in [\![\phi_1]\!] \\ \hline \end{array}$$

The above interpretation validates the following logical rules:

- $(\Longrightarrow)$ $\dfrac{\vdash \phi_0' \Longrightarrow \phi_0 \quad \vdash \{\phi_0\}e\{\phi_1\} \quad \vdash \phi_1 \Longrightarrow \phi_1'}{\vdash \{\phi_0'\}e\{\phi_1'\}}$

- $(\land)$ $\dfrac{\vdash \{\phi_0\}e\{\phi_1\} \quad \vdash \{\phi_0\}e\{\phi_2\}}{\vdash \{\phi_0\}(e)\{\phi_1 \land \phi_2\}}$

- $(;)$ $\dfrac{\vdash \{\phi_0\}e_0\{\phi_1\} \quad \vdash \{\phi_1\}e_1\{\phi_2\}}{\vdash \{\phi_0\}(e_0; e_1)\{\phi_2\}}$

- (if) $\dfrac{\vdash \{\phi_0 \land \phi\}e_0\{\phi_1\} \quad \vdash \{\phi_0 \land \neg\phi\}e_1\{\phi_1\}}{\vdash \{\phi_0\}if(\phi, e_0, e_1)\{\phi_1\}}$

3.2.1 Translation of HL in $EL_T(\Sigma)$

We introduce a suitable signature Σ for EL_T, so that the above interpretation of HL factors through a translation $(\_)^*$ into $EL_T(\Sigma)$. Moreover, the above rules for HL can be derived from simple axioms for the operations in Σ, which may hold in models different from the *intended* one.

- primitive types

  S states

- primitive functions

  $lkp \colon TS$ lookup,
  $upd \colon S \to T1$ update

- axioms

  (upd.1) $s \colon S \vdash upd(s); lkp = upd(s); [s] \colon TS$

  (upd.2) $s, s' \colon S \vdash upd(s); upd(s') = upd(s') \colon TS$

  (lkp.1) $\vdash \text{let } s \Leftarrow lkp \text{ in } upd(s) = [*] \colon T1$

  (lkp.2) $\Gamma \vdash lkp; e = e \colon T\tau$

The model corresponding to the standard interpretation uses the monad $TX = S \to \mathcal{P}((X \times S))$ in **Set**. Also in this case, one can define a general model construction. Given a monad T:

- take the monad $T' = FT$, where F is the monad transformer $FTX = T(X \times S)^S$,

- then (using let and lift for T) define $lkp \stackrel{\Delta}{=} \lambda s \colon S.[\langle s, s\rangle]_T$ and $upd(s) \stackrel{\Delta}{=} \lambda s' \colon S.[\langle *, s\rangle]_T$.

In this model based on T' the equational axioms for lkp and upd are easily shown to be valid.

Definition 3.5 (Translation) *The translation $(\_)^*$ from HL to $EL_T(\Sigma)$ maps (in the obvious way) assertions into* **decidable** *formulas over S, i.e. $\phi^*(s) \colon 2 = 1 + 1 \subseteq \Omega$, programs into terms of type $T1$ and the rest into judgements:*

$$\begin{array}{|rcl|}\hline (e_0; e_1)^* & \stackrel{\Delta}{=} & e_0^*; e_1^* \\ if(\phi, e_0, e_1)^* & \stackrel{\Delta}{=} & \text{let } s \Leftarrow lkp \text{ in } (\text{case } \phi^*(s) \text{ of } e_0^* \mid e_1^*) \\ \hline (\phi_1 \Longrightarrow \phi_2)^* & \stackrel{\Delta}{=} & s \colon S \vdash \phi_1^*(s) \Longrightarrow \phi_2^*(s) \\ \hline (\{\phi_1\}e\{\phi_2\})^* & \stackrel{\Delta}{=} & s \colon S \vdash \phi_1^*(s) \Longrightarrow [s' \Leftarrow upd(s); e; lkp]\phi_2^*(s') \\ \hline \end{array}$$

Remark 3.6 Usually one associate to a primitive program α a function $\alpha' : S \to S$. In this case, the translation α^* is given by (let $s \Leftarrow lkp \,\mathrm{in}\, upd(\alpha' s)$).

Proposition 3.7 *The (translation of) HL rules are derivable in EL_T from the axioms for Σ.*

Proof For each rule we say which axioms for Σ (and additional properties of necessity) are needed:

- $(\Longrightarrow)$
- $(\wedge)$ from ($\Box$-$\wedge$*)
- (;) from ($\Box$-μ) and (lkp.1)
- (if) from (upd.1)

∎

3.2.2 Other translations of HL in $EL_T(\Sigma)$

The above translation is the one which mimics more directly the standard interpretation of Hoare triples. However, one could think of other translations, for instance inspired by the one given for $VTLoE$. Here is a sample of possible ways of representing an Hoare triple $\{\phi_1\}e\{\phi_2\}$ as a formula of $EL_T(\Sigma)$, where ϕ_i are predicates over S and e is a term of type $T1$:

1 $\forall s\colon S.\phi_1(s) \supset [s' \Leftarrow upd(s); e; lkp]\phi_2(s')$

1' $\forall s\colon S.[s' \Leftarrow upd(s); e; lkp](\phi_1(s) \supset \phi_2(s'))$

2 $\forall c\colon T1.([s \Leftarrow c; lkp]\phi_1(s)) \supset [s' \Leftarrow c; e; lkp]\phi_2(s')$

2' $\forall c\colon T1.[s, s' \Leftarrow (c; lkp), (e; lkp)](\phi_1(s) \supset \phi_2(s'))$

The above translations agree, when T is a simple state monad, e.g. $TX = (X \times S)_\perp^S$. In general, the following implications hold (provided certain additional axioms for EL_T are satisfied):

- $1 \Longrightarrow 1'$ provided ($\Box$-$\supset$*)
- $1' \Longrightarrow 1$ provided ($\Box$-$\wedge$*)
- $1 \Longrightarrow 2$ provided (lkp.1) and ($\Box$-μ)
- $2 \Longrightarrow 1$ provided (upd.1)
- $1' \Longrightarrow 2'$ provided (lkp.1) and ($\Box$-μ)
- $2' \Longrightarrow 1'$ provided (upd.1)
- $2' \Longrightarrow 2$ provided ($\Box$-$\wedge$*)

3.2.3 Open issues

It is not clear which of the above representations for Hoare triples scales up best w.r.t. changes to T. In any case, when necessity fails to satisfy the additional axioms, some of the rules for HL cannot be validate in $EL_T(\Sigma)$, no matter which of the translations one uses.

References

[Eme90] E.A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*. North Holland, 1990.

[FP93] M.P. Fiore and G.D. Plotkin. An axiomatisation of computationally adequate domain theoretic models of FPC. Draft, November, 1993.

[HMSTar] F. Honsell, I.A. Mason, S.F. Smith, and C. Talcott. A variable typed logic of effects. *Information and Computation*, to appear.

[Mog89] E. Moggi. An abstract view of programming languages. Technical Report ECS-LFCS-90-113, Edinburgh Univ., Dept. of Comp. Sci., 1989. Lecture Notes for course CS 359, Stanford Univ.

[Mog91] E. Moggi. Notions of computation and monads. *Information and Computation*, 93(1), 1991.

[Mog94a] E. Moggi. A general semantics for evaluation logic. In *9th LICS Conf.* IEEE, 1994.

[Mog94b] E. Moggi. A semantics for evaluation logic. *Fundamenta Informaticae*, 22(1/2), 1994.

[MT92] I. Mason and C. Talcott. References, local variables and operational reasoning. In *7th LICS Conf.* IEEE, 1992.

[Ole85] F.J. Oles. Type algebras, functor categories and block structure. In M. Nivat and J.C. Reynolds, editors, *Algebraic Methods in Semantics*, 1985.

[OT92] P.W. O'Hearn and R.D. Tennent. Semantics of local variables. In *Applications of Categories in Computer Science*, number 177 in L.M.S. Lecture Notes Series. Cambridge University Press, 1992.

[OT93] P.W. O'Hearn and R.D. Tennent. Relational parametricity and local variables. In *20th POPL*. ACM, 1993.

[Pit91] A.M. Pitts. Evaluation logic. In G. Birtwistle, editor, *IVth Higher Order Workshop, Banff 1990*, volume 283 of *Workshops in Computing*. Springer Verlag, 1991.

[PS93] A.M. Pitts and I.D.B. Stark. On the observable properties of higher order functions that dynamically create local names (preliminary report). In *Workshop on State in Programming Languages, Copenhagen*, 1993. Yale Univ., Comp. Sci. Tech. Report.

[Stiar] C. Stirling. Modal and temporal logics. In Samson Abramsky, Dov M. Gabbay, and Tom S. E. Maibaum, editors, *Handbook of Logic in Computer Science, Volume III*, chapter 3.10. Oxford University Press, to appear.

[Tal93] C. Talcott. Localized semantics for VTLoE. Working Notes, 21 October, 1993.