Denotational Semantics of first-order Object Calculus

Thomas Sixuan Lou

March 1, 2017

1 Introduction

2 A Pure Calculus with Constraints

In this section, we will define $\mathbf{O}_c$, an pure object-based calculus with constraints introduced through a special binder $\iota(o).e$.

The calculus is parametrized over an algebra and an theory on that algebra. Intuitively, the algebra describes the meanings of those primitive syntactic expressions. An theory defines a equivalence relation on that algebra, which the solver will solve constraints with respect to.

Example 2.1 (Modular arithmetic)**.** The arithmetic algebra $AE = \langle \mathbb{N}, 0, 1, +, \times \rangle$ is an algebra over with carriers of natural numbers $\mathbb{N}$, characteristic elements $0, 1$ and the following binary operations:

$$+ : \mathbb{N} \to \mathbb{N} \to \mathbb{N}$$
$$\times : \mathbb{N} \to \mathbb{N} \to \mathbb{N}$$

satisfying

$$0 + x \doteq x \doteq x + 0 \quad x + (y + z) \doteq (x + y) + z$$
$$1 \times x \doteq x \doteq x \times 1 \quad x \times (y \times z) \doteq (x \times y) \times z$$

$\mathbb{N}_m$ is an theory on AE with the following equivalence relation:

$$a \sim b \text{ iff } a \equiv b \pmod{m}$$

Example 2.2 (Boolean algebra)**.** A distributive lattice $\mathcal{L}$ on set L is the algebra $\langle L, \wedge, \vee \rangle$ with operations $\wedge, \vee : L \to L \to L$ satisfying

$$\begin{aligned} x \wedge x &\doteq x & x \vee x &\doteq x \\ x \wedge y &\doteq y \wedge x & x \vee (x \wedge y) &\doteq x \\ x \vee y &\doteq y \vee x & x \wedge (x \vee y) &\doteq x \\ x \wedge (y \wedge z) &\doteq (x \wedge y) \wedge z & x \wedge (y \vee z) &\doteq (x \wedge y) \vee (x \wedge z) \\ x \vee (y \vee z) &\doteq (x \vee y) \vee z & x \vee (y \wedge z) &\doteq (x \vee y) \wedge (x \vee z) \end{aligned}$$

A boolean algebra $\mathcal{B} = \langle B, 0, 1, \wedge, \vee, ' \rangle$ is an extension to $\mathcal{L}$ with characteristic elements $0, 1$ and an complement operator $' : B \to B$, satisfying

$$\begin{aligned} x \wedge 0 &\doteq 0 & x \wedge 1 &\doteq x \\ x \vee 1 &\doteq 1 & x \vee 0 &\doteq x \\ x' \wedge x &\doteq 0 & x' \vee x &\doteq 1 \end{aligned}$$

$\mathbf{Su}(X)$ (subsets of X) is a theory on $\mathcal{B}$ with the following equivalence relation:

$$x \sim y \text{ iff } \forall z. z \in x \leftrightarrow z \in y$$

$c ::= \iota(x).e$	constraint binder	$a, b ::=$	terms
$e ::=$	constraint terms	$t(a_1, \ldots, a_n)$	primitive terms $t \in \mathcal{L}$
$a_1 \sim a_2$	relation	l, r	labels
$e_1 \wedge e_2$	conjunction	x	variable
$e_1 \vee e_2$	disjunction	$a.l_i \quad a.r_i$	method/constraint invocation
$\neg e_1$	negation	$[l_i = \varsigma(x).b_i, r_j = c_j]_{\substack{i=1..n \\ j=1..m}}$	object
$\exists x.e_1$	decl of local variable	$a.l_i \Leftarrow \varsigma(x).b \quad a.r_i \Leftarrow c$	method/constraint update
$\mathcal{E}[-]$	context		

Figure 1: Syntax of $\mathbf{O}_c$ under theory T on algebra A

In this paper, the constraint solver is abstracted as a blackbox that is able to find a single best solution from a constraint whenever possible. The solver is also parametrized over an algebra and a theory so that it's general enough to solve equivalence constraints of arbitrary interpretations.

For example, an solver $\mathfrak{S}$ over $\mathbb{N}/_7$ is expected to be able to obtain a solution from constraints on the relation $\pmod 7$: given constraint $c = a \sim 0 \wedge b \sim a$

$$c \overset{\mathfrak{S}}{\rightsquigarrow} \{a \mapsto 7, b \mapsto 0\}$$

2.1 Syntax Formalization

We now formally introduce the calculus. The syntax of the calculus is constructed from a language $\mathcal{L}$, and various special syntactical constructs. There are two syntactical categories, the object fragment (a) and the constraint fragment (c, e). They are separated so that rules involving constraint solving are easier to write.

The complete syntax definition is given in Figure 1. Each object is declared as a collection of $\langle$field name, method$\rangle$ and $\langle$constraint name, constraint$\rangle$ pairs. Each method (resp. constraint) is given as a binder "$\varsigma(x).a$" (resp. "$\iota(x).e$"), where x is bound to which object it's invoked from.

We define free variables of $\mathbf{O}_c$ terms and substitutions in Figure 2 and Figure 3 respectively.

$$\begin{aligned}
FV(t(a_1, \ldots, a_n)) &= \bigcup_{i \in [n]} FV(a_i) \\
FV(x) &= \{x\} \\
FV(l) &= \{l\} \\
FV(a.l_i) &= FV(a) \\
FV(a.r_i) &= FV(a) \\
FV([l_i = \varsigma(x).b_i, r_j = c_j]_{\substack{i \in [n] \\ j \in [m]}}) &= \bigcup_{i \in [n]} FV(\varsigma(x).b_i) \cup \bigcup_{j \in [m]} FV(c_j) \\
FV(a.l_i \Leftarrow \varsigma(x).b) &= FV(a) \cup FV(\varsigma(x).b)
\end{aligned}$$

$$\begin{aligned}
FV(a.r_i \Leftarrow c) &= FV(a) \cup FV(c) \\
FV(\varsigma(x).b) &= FV(b) \backslash \{x\} \\
FV(\iota(x).e) &= FV(e) \backslash \{x\} \\
FV(a_1 \sim a_2) &= FV(a_1) \cup FV(a_2) \\
FV(e_1 \wedge e_2) &= FV(e_1) \cup FV(e_2) \\
FV(e_1 \vee e_2) &= FV(e_1) \cup FV(e_2) \\
FV(\neg e) &= FV(e) \\
FV(\exists x.e) &= FV(e) \backslash \{x\}
\end{aligned}$$

Figure 2: Free variables in $\mathbf{O}_c$

Since our calculus was parametrized over arbitrary algebras and theories, we define the notion of evaluation that evaluate a term involving primitives (i.e. with terms in $\mathcal{L}$) down to its general form by substituting

$$
\begin{aligned}
[a/y]t(a_1,\ldots,a_n) &= t([a/y]a_1,\ldots,[a/y]a_n)\\
[a/y]x &= x\\
[a/y]y &= a\\
[a/y]b.l_i &= ([a/y]b).l_i\\
[a/y]b.r_i &= ([a/y]b).r_i\\
[a/y]\varsigma(x).b &= \varsigma(z)[a/y][z/x]b\\
[a/y]\iota(x).b &= \iota(z)[a/y][z/x]b\\
[a/y][l_i = \varsigma(x).b_i, r_j = c_j] &= [l_i = [a/y]\varsigma(x).b_i, r_j = [a/y]c_j]
\end{aligned}
$$

$$
\begin{aligned}
[a_1/y]a.l_i \Leftarrow \varsigma(x).b &= ([a_1/y]a).l_i \Leftarrow [a_1/x]\varsigma(x).b\\
[a_1/y]a.r_i \Leftarrow c &= ([a_1/y]a).r_i \Leftarrow [a_1/x]c\\
[a/y]a_1 \sim a_2 &= [a/y]a_1 \sim [a/y]a_2\\
[a/y]e_1 \wedge e_2 &= [a/y]e_1 \wedge [a/y]e_2\\
[a/y]e_1 \vee e_2 &= [a/y]e_1 \vee [a/y]e_2\\
[a/y]\neg e &= \neg([a/y]e)\\
[a/y]\exists x.e &= \exists z.([a/y][z/x]e)
\end{aligned}
$$

Figure 3: Substitution in $\mathbf{O}_c$

$$
\begin{aligned}
\mathfrak{e}(f(a_1,\ldots,a_2)) &= f_{\mathscr{I}}(\mathfrak{e}(a_1),\ldots,\mathfrak{e}(a_n))\\
\mathfrak{e}(x) &= x\\
\mathfrak{e}(a.l_i) &= \mathfrak{e}(a).l_i\\
\mathfrak{e}(a.r_i) &= \mathfrak{e}(a).r_i\\
\mathfrak{e}([l_i = \varsigma(x).b_i, r_i = c_i]) &= [l_i = \varsigma(x).\mathfrak{e}(b_i), r_i = \mathfrak{e}(c_i)]\\
\mathfrak{e}(a.l_i \Leftarrow \varsigma(x).b) &= \mathfrak{e}(a).l_i \Leftarrow \varsigma(x).\mathfrak{e}(b)\\
\mathfrak{e}(a.r_i \Leftarrow c) &= \mathfrak{e}(a).l_i \Leftarrow \mathfrak{e}(c)
\end{aligned}
$$

$$
\begin{aligned}
\mathfrak{e}(\iota(x).e) &= \iota(x).\mathfrak{e}(e)\\
\mathfrak{e}(a_1 \sim a_2) &= \iota(a_1) \sim \iota(a_2)\\
\mathfrak{e}(e_1 \wedge e_2) &= \mathfrak{e}(e_1) \wedge \mathfrak{e}(e_2)\\
\mathfrak{e}(e_1 \vee e_2) &= \mathfrak{e}(e_1) \vee \mathfrak{e}(e_2)\\
\mathfrak{e}(\neg e) &= \neg\mathfrak{e}(e)\\
\mathfrak{e}(\exists x.e) &= \exists x.\mathfrak{e}(e)
\end{aligned}
$$

Figure 4: Evaluation

primitives with terms and formulas in the algebra.

In evaluating terms using the equational facts from the algebra, we have avoided adding rules of evaluating arbitrary terms into our calculus. Thus, we only need to focus the core object and constraint solving fragment in our calculus.

Definition 2.3 (Evaluation)**.** Let $\mathcal{L}$ be a language, let $\mathcal{A} = \langle A, \{f_n\}_n\rangle$ be an algebra and $\mathscr{I}$ be an interpretation of the language to the algebra. Let $f_{\mathscr{I}}$ denote the function f in the language under interpretation $\mathscr{I}$ (it will be a function in $\mathcal{A}$). A *evaluation* of term a is defined inductively in Figure 4. The evaluated term is called a *generalized term.*

2.2 Semantics of Constraints

Let $\mathfrak{S}_{\mathcal{A},\mathcal{T}}$ be a solver parametrized by an algebra $\mathcal{A}$ and a theory $\mathcal{T}$. It interacts with our calculus by the following satisfaction judgment $o(c) \models_{\mathfrak{S}_{\mathcal{A},\mathcal{T}}} S$, where o is an object, S is a set of objects and c is a constraint. The judgment means: S is the set of objects that satisfies constraint c when evaluated from an particular object o.

This satisfaction judgment gives the semantics of equations (more generally, $\sim$-relations) in our calculus. If we leave out the constraint c and the initial object o is the judgment, it induces a semantic map on constraints:

$$[\![c]\!] : \mathit{Object} \to \mathcal{P}(\mathit{Object})$$

The link between the semantic map and the judgment is established by:

$$o(c) \models_{\mathfrak{S}_{\mathcal{A},\mathcal{T}}} [\![c]\!](o)$$

$$\frac{[o/x]a_1 \rightsquigarrow v_1 \quad [o/x]a_2 \to v_2 \quad v_1 \sim v_2}{o(\iota(x).a_1 \sim a_2) \models_{\mathfrak{S}_{\mathcal{A},\mathcal{T}}} \{o\}} \text{ SAT-EQ}$$

$$\frac{[o/x]a_1 \in A \quad [o/x]a_2 \in A \quad [o/x]a_1 \rightsquigarrow v_1 \quad [o/x]a_2 \rightsquigarrow v_2 \quad v_1 \not\sim v_2}{o(\iota(x).a_1 \sim a_2) \models_{\mathfrak{S}_{\mathcal{A},\mathcal{T}}} \varnothing} \text{ SAT-EMPTY}$$

$$\frac{l = [o/x]a_1 \quad [o/x]a_2 \rightsquigarrow v \quad l \notin FV([o/x]a_2) \quad o.l \Leftarrow \varsigma(\rho(x))v \to o'}{o(\iota(x).a_1 \sim a_2) \models_{\mathfrak{S}_{\mathcal{A},\mathcal{T}}} \{o'\}} \text{ SAT-UPDATE}_1$$

$$\frac{l = [o/x]a_2 \quad l \notin FV(\hat{a}_1) \quad \hat{a}_1 \rightsquigarrow v \quad [o/x]a_1 = \hat{a}_1 \text{ is not a label or a variable} \quad o.l \Leftarrow \varsigma(\rho(x))v \rightsquigarrow o'}{o(\iota(x).a_1 \sim a_2) \models_{\mathfrak{S}_{\mathcal{A},\mathcal{T}}} \{o'\}} \text{ SAT-UPDATE}_2$$

Figure 5: Satisfaction Judgment

Thus, the judgment is an externalization of that semantic map, hiding away the computational contents in interpreting constraints.

We may sometimes want to construct a new construct c by combining several constraints $\{c_i\}_{i\in I}$, aside from creating a new object with all of c_i inserted, we may also internalize it using first-order connectives we've defined earlier. Thus, it inspires us to define the following auxiliary functions:

$$(\iota(x).e_1) \otimes (\iota(x).e_2) \triangleq \iota(x).e_1 \wedge e_2$$
$$(\iota(x).e_1) \oplus (\iota(x).e_2) \triangleq \iota(x).e_1 \vee e_2$$

2.2.1 Semantics of First-order constraints

Figure 6: Satisfaction Judgment on First-order Constraints

2.3 Reduction

Definition 2.4 (Reduction)**.**

Theorem 2.5 (monotonicity of objects)**.** *For any object $o = [l_i = \varsigma(x).b_i, r_j = c_j]_{i\in[n],j\in[m]}$, given any constraint c, indices $p, q \in [m]$, if $o(c_p) \models_{\mathfrak{S}_{\mathcal{A},\mathcal{T}}} S$ and $o.r_q \Leftarrow c \rightarrowtail o'$, then $o' \in S$.*

2.4 Operational Semantics

$$\frac{}{[l_i = \varsigma(x).b_i, r_j = c_j].l_k \rightarrowtail [[l_i = \varsigma(x).b_i, r_i = c_i]_{\substack{i\in[n]\\ j\in[m]}}/x]b_k}\ \text{RED-M-INVOKE}$$

$$\frac{[l_i = \varsigma(x).b_i, r_j = c_j](c_k) \models_{\mathfrak{S}_{\mathcal{A},\mathcal{T}}} S \quad o' \in S}{[l_i = \varsigma(x).b_i, r_j = c_j].c_k \rightarrowtail o'}\ \text{RED-C-INVOKE}$$

$$\frac{[l_k = \varsigma(x).b, l_{i\neq k} = \varsigma(x).b_i, r_j = c_j](c_1 \otimes \cdots \otimes c_m) \models_{\mathfrak{S}_{\mathcal{A},\mathcal{T}}} S \quad o' \in S}{[l_i = \varsigma(x).b_i, r_j = c_j].l_k \Leftarrow \varsigma(x).b \rightarrowtail o'}\ \text{RED-M-UPDATE}$$

$$\frac{[l_i = \varsigma(x).b_i, r_{j\neq k} = c_j, r_k = c](c) \models_{\mathfrak{S}_{\mathcal{A},\mathcal{T}}} S \quad o' \in S}{[l_i = \varsigma(x).b_i, r_j = c_j].c_k \Leftarrow c \rightarrowtail o'}\ \text{RED-C-UPDATE}$$

$$\frac{a \rightarrowtail a'}{\mathcal{E}[a] \rightsquigarrow \mathcal{E}[a']}\ \text{RED-CONGR}$$

Figure 7: Reduction relation in $\mathbf{O}_c$