Capturing Term Algebra Computations in Matching Logic

Dorel Lucanu[1]

[1]Alexandru Ioan Cuza University of Iași

LAP, Dubrovnik, September 26, 2022

Plan

Matching Logic - a Foundation for K Framework[1]

A minimal logic where

- definition of programming languages and
- behavioral properties of their programs

can uniformly specified.

[1] https://kframework.org/

Example: Language Definition

A language definition is just a ML theory:

$$\langle \texttt{if}\,(e)\,s_1\,\texttt{else}\,s_2 \leadsto \kappa\rangle\,\langle\sigma\rangle \wedge \neg e{:}\,\mathit{Value} \to \bullet\langle e \leadsto \texttt{if}\,(\_)\,s_1\,\texttt{else}\,s_2 \leadsto \kappa\rangle\,\langle\sigma\rangle$$
$$\langle v {\leadsto} \texttt{if}\,(\_)\,s_1\,\texttt{else}\,s_2 \leadsto \kappa\rangle\,\langle\sigma\rangle \wedge v{:}\,\mathit{Value} \to \bullet\langle \texttt{if}\,(v)\,s_1\,\texttt{else}\,s_2 \leadsto \kappa\rangle\,\langle\sigma\rangle$$

$$\begin{aligned}\langle \texttt{if}\,(b)\,s_1\,\texttt{else}\,s_2 \leadsto \kappa\rangle\,\langle\sigma\rangle \wedge b{:}\,\mathit{Bool} \to {} & \bullet\langle s_1 \leadsto \kappa\rangle\,\langle\sigma\rangle \wedge b = \texttt{true} \\ & \vee \\ & \bullet\langle s_2 \leadsto \kappa\rangle\,\langle\sigma\rangle \wedge b = \texttt{false}\end{aligned}$$

$$\langle \texttt{choose}\,x\,\texttt{from}\,e \leadsto \kappa\rangle\,\langle\sigma\rangle \wedge \neg e{:}\,\mathit{Value} \to \bullet\,\langle e \leadsto \texttt{choose}\,x\,\texttt{from}\,\_ \leadsto \kappa\rangle\,\langle\sigma\rangle$$
$$\langle v \leadsto \texttt{choose}\,x\,\texttt{from}\,\_ \leadsto \kappa\rangle\,\langle\sigma\rangle \wedge v{:}\,\mathit{Value} \to \bullet\,\langle \texttt{choose}\,x\,\texttt{from}\,v \leadsto \kappa\rangle\,\langle\sigma\rangle$$

$$\langle \texttt{choose}\,x\,\texttt{from}\,v \leadsto \kappa\rangle\,\langle\sigma\rangle \wedge v{:}\,\mathit{Value} \to \bullet\exists x_0.\,\langle\kappa\rangle\,\langle\sigma[x \mapsto x_0]\rangle \wedge x_0 \in v$$

where

$\bullet\varphi'$ is matched by all the configurations for that there exists a next configuration in φ' (strong next)

Example: Program Properties

Program properties are also ML patterns:

$$\langle \texttt{while (x > 0) x = x-1;} \rangle \langle \mathrm{x} \mapsto \$x \rangle \wedge \$x > 3 \rightarrow \Box_w \langle \cdot \rangle \langle \mathrm{x} \mapsto 0 \rangle$$

$$\langle \texttt{while (true) \{\}} \rangle \langle \mathrm{x} \mapsto \$x \rangle \rightarrow \Box_w \langle \cdot \rangle \langle \mathrm{x} \mapsto 0 \rangle$$

where

$\circ\varphi' \equiv \neg\bullet\neg\varphi'$ (all-paths/weak next)

$\Box_w\psi \equiv \nu X.\, \psi \vee (\circ X \wedge \bullet\top)$ (weak always finally)

The Addressed Challenge

- the program properties are checked using symbolic execution (SE)
- SE uses various algorithms: matching, unification, anti-unifiction, etc.
- these computations are ML term patterns transformers
- questions:
  - what is the relationship between the initial pattern and the transformed one?
  - how this relation is proved internally within ML?

Plan

An ML Itinerary

- An alternative to Hoare/Floyd Logic (Roșu, Ellison, Schulte, AMAST 2010)
- (Many-sorted) Matching Logic (Roșu, LMCS 2017)
- Matching mu-Logic (Chen, Roșu, LICS 2019)
- Applicative Matching Logic (Chen & Roșu, TR 2019; Chen & Lucanu & Roșu: Matching Logic explained. JLAMP 2021.)

Syntax

Patterns:

$\varphi ::= x$	elementary variable $(x \in EV)$
$\mid X$	set variabile $(X \in SV)$
$\mid \sigma$	symbol $(\sigma \in \Sigma)$
$\mid \varphi_1\ \varphi_2$	application
$\mid \bot$	bottom
$\mid \varphi_1 \to \varphi_2$	implication
$\mid \exists x.\varphi$	existential binder
$\mid \mu X.\varphi$ *if* φ *is positive in* X	least fixpoint binder

Semantics Intuitively

M a set

ρ : elementary-variables $\cup$ set-variables $\to M \cup \mathcal{P}(M)$

x	singleton subset $\{\rho(x)\}$
$\mid X$	subset $\rho(X) \subseteq M$
$\mid \sigma$	subset $\sigma_M \subseteq M$
$\mid \varphi_1\ \varphi_2$	$\_\bullet\_ : M \times M \to \mathcal{P}(M)$
$\mid \bot$	$\emptyset$
$\mid \varphi_1 \to \varphi_2$	$M \setminus (\lvert\varphi_1\rvert_{M,\rho} \setminus \lvert\varphi_2\rvert_{M,\rho})$
$\mid \exists x.\varphi$	$\bigcup_{a\in M} \lvert\varphi\rvert_{M,\rho[a/x]}$
$\mid \mu X.\varphi$	$\mathbf{lfp}(A \mapsto \lvert\varphi\rvert_{M,\rho[A/X]})$

$M \models \varphi$ iff $\lvert\varphi\rvert_{M,\rho} = M$ fora all ρ

Derived Patterns

$$\top \equiv \neg\bot \qquad \varphi_1 \vee \varphi_2 \equiv \neg\varphi_1 \to \varphi_2$$

$$\neg\varphi \equiv \varphi \to \bot \qquad \varphi_1 \wedge \varphi_2 \equiv \neg(\neg\varphi_1 \vee \neg\varphi_2)$$

$$\forall x.\varphi \equiv \neg\exists x.\neg\varphi \qquad \varphi_1 \leftrightarrow \varphi_2 \equiv (\varphi_1 \to \varphi_2) \wedge (\varphi_2 \to \varphi_1)$$

$$\nu X.\,\varphi \equiv \neg\mu X.\,\neg\varphi[\neg X/X] \qquad \varphi \text{ positive in } X$$

Definedness Theory

theory DEF

Symbols: def

Notations: $\lceil \varphi \rceil \equiv \mathrm{def}\ \varphi$

Axioms: (Definedness) $\forall x.\, \lceil x \rceil$

Notations:

$\lfloor \varphi \rfloor \equiv \neg \lceil \neg \varphi \rceil$ // totality

$\varphi_1 = \varphi_2 \equiv \lfloor \varphi_1 \leftrightarrow \varphi_2 \rfloor$ // equality

$\varphi_1 \subseteq \varphi_2 \equiv \lfloor \varphi_1 \rightarrow \varphi_2 \rfloor$ // set inclusion

$x \in \varphi \equiv x \subseteq \varphi$ // membership

endtheory

Plan

Term Algebra

- Sorts: S
- Operation symbols: F
  each operation symbol $f \in F$ has a type $s_1 \ldots s_n \to s$ (we equivalently write $f \in F_{s_1 \ldots s_n, s}$)
  if $n = 0$, then f is a constant
- (algebraic) signature: $\Sigma = (S, F)$, $f \in F_{s_1 \ldots s_n, s}$
- variables: $X = (X_s)_{s \in S}$; we write $x : s$ for $x \in X_s$
- terms: $T_\Sigma(X) = (T_\Sigma(X)_s)_{s \in S}$
  - $X_s \subseteq T_\Sigma(X)_s$
  - if $t_i \in T_\Sigma(X)_{s_i}$, $i = 1, \ldots, n$, and $f \in F_{s_1 \ldots s_n, s}$, then $f(t_1, \ldots, t_n) \in T_\Sigma(X)_s$
    (for $n = 0$, we get $F_{[], s} \subseteq T_\Sigma(X)_s$)
- $T_\Sigma(X)$ is the Σ-algebra freely generated by X
- $T_\sigma = T_\Sigma(\emptyset)$ is the initial algebra in the category of Σ-algebras

Sorts

theory SORT **Imports:** DEF

Symbols: $inh, Sort\,,\quad s \in S$

Notations:

$\top_s \equiv inh\ \ s$ // inhabitants of sort s

$s_1 \leq s_2 \equiv \top_{s_1} \subseteq \top_{s_2}$ // subsort relation

$\neg_s\varphi \equiv (\neg\varphi) \wedge \top_s$ // negation within sort s

$\forall x{:}s.\,\varphi \equiv \forall x.\, x \in \top_s \rightarrow \varphi$ // $\forall$ within sort s

$\exists x{:}s.\,\varphi \equiv \exists x.\, x \in \top_s \wedge \varphi$ // $\exists$ within sort s

$\mu X{:}s.\,\varphi \equiv \mu X.\, X \subseteq \top_s \wedge \varphi$ // μ within sort s

$\nu X{:}s.\,\varphi \equiv \nu X.\, X \subseteq \top_s \wedge \varphi$ // ν within sort s

$\varphi{:}s \equiv \exists z{:}s.\,\varphi = z$ // “typing”

Axioms: $\forall x.\, x \in \top_{Sort} \leftrightarrow \lceil \top_x \rceil$

$\exists x.\, x = Sort$

$Sort \in \top_{Sort}$

$s \in \top_{Sort}$

endtheory

Function Symbols F[2]

theory ALGEBRA(S, F)

Symbols: $s \in S, f \in F, \mathrm{SigOps}, \mathrm{SigArgs}$

Notations:

$$f : s_1 \otimes \cdots \otimes s_n \ominus s \equiv \forall x_1 : s_1 \ldots \forall x_n : s_n . \exists y : s . f\, x_1 \ldots x_n = y$$

Axioms:

(Sort)	$s : Sort$ **for** $s \in S$
(Function)	$f : s_1 \otimes \cdots \otimes s_n \ominus s$ **for** $f \in F_{s_1 \ldots s_n, s}$
(Signature Ops)	$\top_{\mathrm{SigOps}} = \bigvee_{f \in F} f$
(Signature Args)	$\top_{\mathrm{SigArgs}} = \bigvee_{f \in F_{s_1 \ldots s_n, s}} \top_{s_1 \otimes \cdots \otimes s_n}$

endtheory

[2]Product, sum, and function sorts are defined in Chen & Lucanu & Roșu: Matching Logic Explained. JLAMP 2021.

No-Confusion and No-Junk Properties

theory NOCONFUSION(S, F) Imports: ALGEBRA(S, F)

Axioms:

(Distinct Function) $f \neq f'$ `for distinct operation symbols` $f, f' \in F$

(No Confusion) $\forall f, f' : \text{SigOps}.\, \forall args, args' : \text{SigArgs}.$
$$(f\ args) = (f'\ args') \rightarrow f = f' \wedge args = args'$$

endtheory

theory NOJUNK(S, F) Imports: ALGEBRA(S, F)

Notations:

$D_s \equiv (\text{proj}\ i\ \ D)$ `if` s `is` s_i `for some` $s_i \in S_{\text{nonvoid}}$

$D_s \equiv \bot$ `if` $s \in S_{\text{nonvoid}}$

$f\ D_w \equiv f\ D_{s'_1} \ldots D_{s'_m}$ `where` $w = s'_1 \ldots s'_m$

$$F\,D \equiv \left\langle \bigvee_{f \in F_{w,s_1}, w \in S*} f\ D_w\,, \ldots, \bigvee_{f \in F_{w,s_n}, w \in S*} f\ D_w \right\rangle$$

Axioms:

(No Junk Void) $\top_s = \bot$ `for each` $s \in S_{\text{void}}$

(No Junk Non-Void) $\langle \top_{s_1}, \ldots, \top_{s_n} \rangle = \mu D.\, F\, D$

endtheory

Term Algebra[3]

```
theory TERMALGEBRA(S, F)
  Imports: NOCONFUSION(S, F)
  Imports: NOJUNK(S, F)
endtheory
```

Theorem

If $M \vDash \mathrm{TERMALGEBRA}(F)$, then $\alpha(M)$ is isomorphic to the term algebra T_F.

where $\alpha(M)$ is the algebra A with

- A_s the interpretation of the pattern $\top_s$,
- $A_f : A_{s_1} \times \cdots \times A_{s_n} \to A_s$ (enforced by the axiom $f : s_1 \otimes \cdots \otimes s_n \ominus s$)

[3] Chen & Lucanu & Roșu: Initial algebra semantics in matching logic. TR UIUC 2020.

Plan

Computations in Term Algebra (Partial List)

- matching: given t_1 and t_2, compute σ s.t. $t_2 = t_1\,\sigma$ (if any);
- unification: given t_1 and t_2, compute (the most general) σ s.t. $t_2\sigma = t_1\,\sigma$ (if any);
- antiunification: given t_1 and t_2, compute (the least general) t and σ_1, σ_2 s.t. $t\,\sigma_1 = t_1$ and $t\,\sigma_2 = t_2$;
- rewriting: $t_1 \Rightarrow_R t_2$ iff there is a rule $\ell \Rightarrow r \in R$, a context u, and a matching substitution σ s.t. $t_1 = u[\ell\,\sigma]$ and $t_2 = u[r\,\sigma]$;
- narrowing: $t_1 \Rightarrow_R t_2$ iff there is a rule $\ell \Rightarrow r \in R$, a context u, and a unifier σ s.t. $t_1 = u[\ell\,\sigma]$ and $t_2 = u[r\,\sigma]$;
- ...

Questions:

- Which of these can be captured in ML?
- If yes, can the corresponding algorithm be instrumented to produce proof objects for the relationships between their inputs and outputs?

Example: Capturing the Unification I

Term Algebra	ML
term t	t
ground instances of t	$\exists var(t).\, t$
substitution σ (σ not circular[4])	$\phi^\sigma \equiv \bigwedge_{x \mapsto u \in \sigma} x = u$
$t\,\sigma$	$\exists var(t) \setminus var(t\,\sigma).\, t \wedge \phi^\sigma$
$\sigma \leq \eta$ w.r.t. t	$\exists var(t, \eta).\, t \wedge \phi^\eta \to \exists var(t, \sigma).\, t \wedge \phi^\sigma$
σ unifier of t_1 and t_2 ($t_1\sigma = t_2\sigma$)	$\exists var(\sigma) \setminus var(t_1, t_2).\, t_1 \wedge t_2 \wedge \phi^\sigma$
most general unifier $mgu(t_1, t_2)$	$t_1 \wedge t_2 \quad (= (t_i \wedge (t_1 = t_2)))$

[4] $\forall x.\, \forall i \geq 1.\, x \notin var(x\,\sigma^i)$

Example: Capturing the Unification II[5]

Unification Algorithm

Delete:	$P \cup \{t \doteq t\} \Rightarrow P$
Decomposition:	$P \cup \{(f\ t_1\ \ldots\ t_n) \doteq (f\ t'_1\ \ldots\ t'_n)\} \Rightarrow P \cup \{t_1 \doteq t'_1, \ldots, t_n \doteq t'_n\}$
Orient:	$P \cup \{(f\ t_1\ \ldots\ t_n) \doteq x\} \Rightarrow P \cup \{x \doteq (f\ t_1\ \ldots\ t_n)\}$
Elimination:	$P \cup \{x \doteq t\} \Rightarrow P\{x \mapsto t\} \cup \{x \doteq t\}$ if $x \notin var(t), x \in var(P)$
Symbol clash:	$P \cup \{(f\ t_1\ \ldots\ t_n) \doteq (g\ t'_1\ \ldots\ t'_n)\} \Rightarrow \dashv$
Occurs check:	$P \cup \{x \doteq (f\ t_1\ \ldots\ t_n)\} \Rightarrow \dashv$, if $x \in var(((f\ t_1\ \ldots\ t_n)))$

Capturing Unification Algorithm in ML

Unification Algorithm	ML
$P = \{u_1 \doteq v_1, \ldots, u_n \doteq v_n\}$	$\phi^P = \bigwedge_i u_i = v_i$
execution step $P \Rightarrow P'$	$\phi^P \leftrightarrow \phi^{P'}$ proof object for $\mathrm{TERMALGEBRA}(S, F) \vdash \phi^P \leftrightarrow \phi^{P'}$
successful execution $P \Rightarrow \sigma$	$\phi^P \leftrightarrow \phi^\sigma$ proof object for $\mathrm{TERMALGEBRA}(S, F) \vdash \phi^P \leftrightarrow \phi^\sigma$

[5] Arusoaie, Lucanu: Unification in Matching Logic. FM 2019.

Plan

Concluding remarks

- Matching Logic (ML) faithfully captures the term algebra (up to an isomorphism).
- Many meta-level concepts in term algebra can be internally represented within ML.
- Certain algorithms in the term algebra can be seen as transformers of ML patterns logically related (e.g., equivalent patterns).
- These algorithms can be instrumented to produce proof objects for the logical relation between patterns.
- This talk presented the particular case of the syntactical unification.

Questions?

Thanks!