A General Approach to Define Binders using Matching Logic

Xiaohong Chen and Grigore Rosu
{xc3,grosu}@illinois.edu

University of Illinois at Urbana-Champaign
August 2020

The companion technical report (containing all proof details): http://hdl.handle.net/2142/106608
Motivation: K and Matching Logic

• The K formal language semantic framework ([http://kframework.org](http://kframework.org))
  • K is a language to define the formal semantics of any programming languages.
  • Language tools (parsers, interpreters, verifiers, etc.) are generated automatically by K.
  • K has been used to define the formal semantics of many real-world languages.
  • K allows users to define binders easily.

\[
\text{syntax } \textit{Exp} ::= \textit{Var} \\
\quad | \textit{Exp Exp} \\
\quad | \text{"lambda" Var "." Exp [binder]}
\]

• K definitions = Matching logic theories
Matching Logic is Expressive

• Many logical systems have been defined as matching logic theories.
  - FOL
  - Separation logic
  - Hoare logic
  - Temporal logics
  - Modal $\mu$-calculus
  - ...
• **new** This paper studies logical systems where binders play a major role.
  - $\lambda$-calculus
  - $\pi$-calculus
  - Type systems
  - ...

![Diagram of logical systems](image-url)
Main Contribution

1. We propose a simple variant of matching logic that is more suitable to capture binders (Sections 3-4).

2. We define $\lambda$-calculus as a matching logic theory $\Gamma^\lambda$ (Section 6).
   - **Key observation**: $\lambda x. e$ does two things: create the binding and build the term.
   - $[x: Var] e \equiv \text{intension } \exists x: Var. (x, e)$, which captures the graph of the function $x \mapsto e$ and thus captures the binding;
   - $\lambda x. e \equiv \text{lambda } [x: Var] e$, which builds the term.

3. We prove the correctness of $\Gamma^\lambda$ in terms of the following theorems:
   a. (Conservative Extension, pp. 20, Theorem 36).
      \[ \vdash_\lambda e_1 = e_2 \iff \Gamma^\lambda \vdash e_1 = e_2 \]
   b. (Deductive Completeness, pp. 20, Theorem 36).
      \[ \Gamma^\lambda \models e_1 = e_2 \iff \Gamma^\lambda \vdash e_1 = e_2 \]
   c. (Representative Completeness, pp. 22, Section 8.2.2).
      For any $\lambda$-theory $T$, there is a matching logic model $M_T \models \Gamma^\lambda$
      such that $T \vdash_\lambda e_1 = e_2 \iff M_T \models e_1 = e_2$.
   d. (Capturing All Models, pp. 19, Lemma 32).
      For any $\lambda$-calculus (concrete ccc) model $A$, there is a matching logic model $M_A \models \Gamma^\lambda$
      such that $A \models_\lambda e_1 = e_2 \iff M_T \models e_1 = e_2$.

4. We generalize it to other systems with binders such as System F, pure type systems, … (Section 9).
Overview of the Talk

• A high-level overview of matching logic: Syntax and semantics.

• An example: The encoding of $\lambda x.e$ in matching logic.

• Generalization to other binders (see Section 9).
Matching Logic

• A very simple and minimal logic, serving as the foundation of K: only 7 constructs

patterns \( \varphi ::= x \mid X \mid \sigma \mid \varphi_1 \varphi_2 \mid \bot \mid \varphi_1 \rightarrow \varphi_2 \mid \exists x. \varphi \)

- element variables (ranging over individual elements)
- set variables (ranging over sets)
- (set) symbols (built-in)
- application
- propositional constraints
- quantification

• The pattern matching semantics:

A pattern \( \varphi \) is interpreted as the set \( |\varphi| \) of elements that match it.

• A matching logic model \( M \) consists of:
  - a nonempty carrier set \( M \);
  - a binary application function \( a \cdot b : M \times M \rightarrow \mathcal{P}(M) \);
  - a symbol interpretation \( \sigma_M \subseteq M \) for every symbol \( \sigma \);
  - given a valuation \( \rho \) such that \( \rho(x) \in M \) and \( \rho(X) \subseteq M \), we define pattern interpretation \( |\varphi|_\rho \) as (see right)

\[
|x|_\rho = \{ \rho(x) \} \quad |X|_\rho = \rho(X) \quad |\sigma|_\rho = \sigma_M \\
|\bot|_\rho = \emptyset \\
|\varphi_1 \rightarrow \varphi_2|_\rho = M \setminus (|\varphi_1|_\rho \setminus |\varphi_2|_\rho) \\
|\varphi_1 \varphi_2|_\rho = \bigcup_{a_1 \in |\varphi_1|_\rho, a_2 \in |\varphi_2|_\rho} a_1 \cdot a_2 \\
|\exists x. \varphi|_\rho = \bigcup_{a \in M} |\varphi|_{\rho[a/x]}
\]
Matching Logic Theories

- We use a theory $\Gamma$ to axiomatically define the “target” systems/models.
- A theory has two components:
  - A set of symbols;
  - A set of patterns called axioms, which axiomatize/define the behaviors of the symbols;
  - We also introduce notations (syntactic sugar) so formulas/expressions of the other systems become well-formed patterns verbatim.
- $M$ is a model of $\Gamma$, if all axioms $\psi$ in $\Gamma$ hold in $M$, i.e., $|\psi|_{\rho} = M$ for all valuations $\rho$.

In Section 4, we define the matching logic theories of equality $\varphi_1 = \varphi_2$, membership $x \in \varphi$, sorts, functions $f: s_1 \times \cdots \times s_n \to s$, pairs $\langle \varphi_1, \varphi_2 \rangle$, power sets $2^s$ of sort $s$. Then, we use them to define the theories of $\lambda$-calculus, System F, etc.
Theory of $\lambda$-Calculus: $\Gamma^\lambda$

$\lambda$-calculus syntax: $e ::= x \mid e_1 e_2 \mid \lambda x. e$

$\alpha$-equivalent representations: 

- $\lambda x_1. e[x_1/x]$
- $\lambda x_2. e[x_2/x]$
- $\lambda x_3. e[x_3/x]$
- $\ldots$

Argument-value pairs: 

- $\langle x_1, e[x_1/x] \rangle$
- $\langle x_2, e[x_2/x] \rangle$
- $\langle x_3, e[x_3/x] \rangle$
- $\ldots$

The set of all pairs (graph): $\exists x: Var. \langle x, e \rangle$ The binding of $x$ in $e$ is created by the $\exists$-binder of matching logic.

The set of all pairs, intensionalized: 

- intension $\exists x: Var. \langle x, e \rangle$

Thus, the set $\exists x: Var. \langle x, e \rangle$ is treated as one element, avoiding pointwise intension (see Section 4.4).

We introduce notation $[x: Var] e \equiv$ intension $\exists x: Var. \langle x, e \rangle$

The matching logic encoding of $\lambda x. e$ is $\text{lambda} [x: Var] e$ where $\text{lambda}$ is a normal symbol/constructor.
Theory $\Gamma^\lambda$ and Its Correctness

$\lambda$-calculus $\rightarrow$ encoding $\rightarrow$ matching logic (within theory $\Gamma^\lambda$)

- variables: $x$ $\rightarrow$ $x$
- application: $e_1 e_2$ $\rightarrow$ $e_1 e_2$
- abstraction: $\lambda x. e$ $\rightarrow$ $\lambda x. e \equiv \text{lambda} [x: \text{Var}] e$
- beta-deduction: $(\lambda x. e)e' = e[e'/x]$ $\rightarrow$ $(\lambda x. e)e' = e[e'/x]$

equivalence: $\vdash_{\lambda} e_1 = e_2$ if and only if $\Gamma^\lambda \vdash e_1 = e_2$ if and only if $\Gamma^\lambda \models e_1 = e_2$

- lambda-calculus reasoning
- matching logic reasoning
- matching logic semantic validity
Conclusion

• We proposed a general approach to defining binders in matching logic, which is the minimal logical foundation of the K framework.

• We proposed a simple variant of matching logic (only 7 constructs);

• We studied untyped $\lambda$-calculus thoroughly and gave the encoding $\lambda x. e \equiv \text{lambda}[x:Var] e$. We proved the correctness.

• In the paper, we gave a systematic treatment of binders in many other systems such as System F, pure type systems, and $\pi$-calculus.

The companion technical report (containing all proof details): http://hdl.handle.net/2142/106608