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

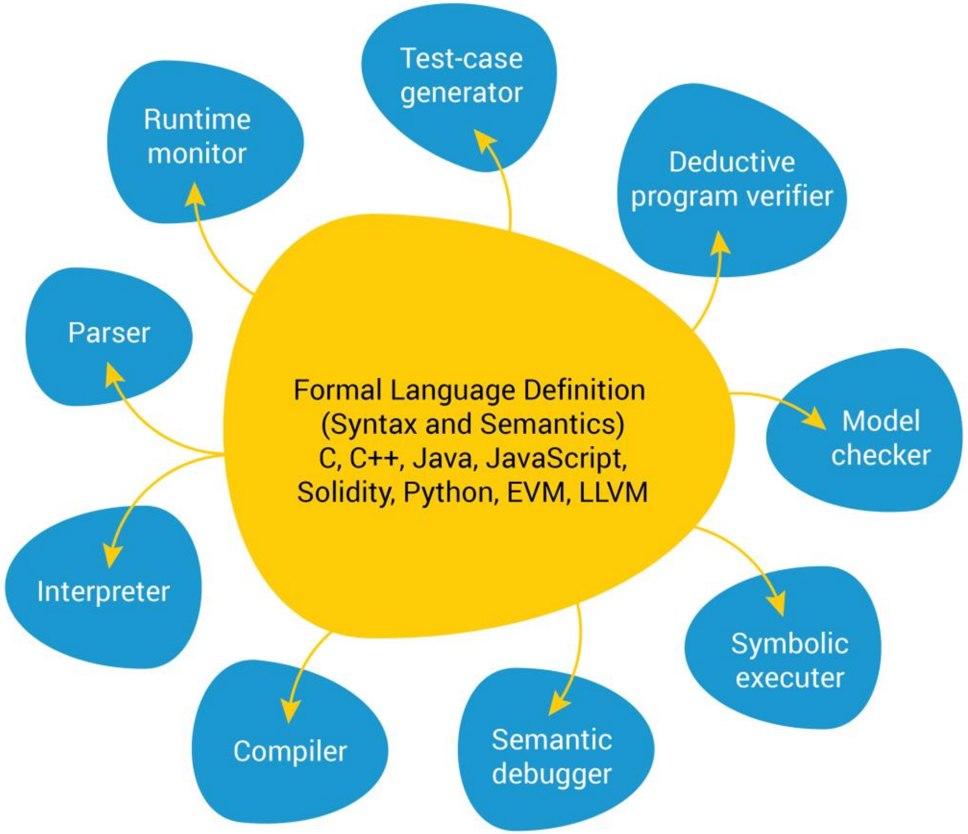
The conference paper: http://fsl.cs.illinois.edu/FSL/papers/2020/chen-rosu-2020-icfp/chen-rosu-2020-icfp-public.pdf
The companion technical report (containing all proof details): http://hdl.handle.net/2142/106608

Motivation: K and Matching Logic

- The K formal language semantic framework (http://kframework.org)
  - 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.

```
syntax Exp ::= Var
             | Exp Exp
             | "lambda" Var "." Exp [binder]
```

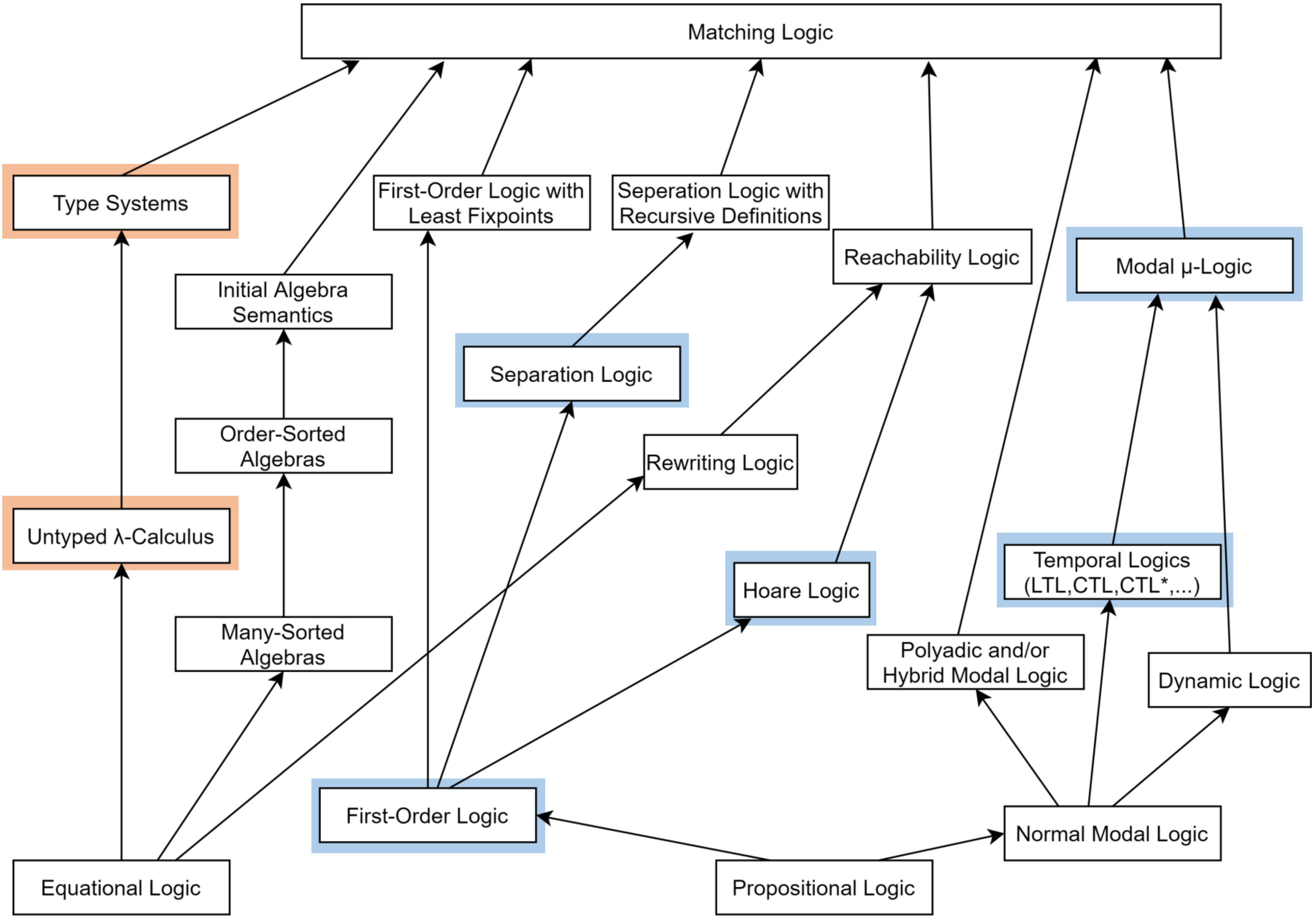
  - K definitions = Matching logic theories



K Framework

Matching Logic is Expressive

- Many logical systems have been defined as matching logic theories.
  - FOL
  - Separation logic
  - Hoare logic
  - Temporal logics
  - Modal μ-calculus
  - …
- **new** This paper studies logical systems where binders play a major role.
  - λ-calculus
  - π-calculus
  - Type systems
  - …



Main Contribution

1. We propose a simple variant of matching logic that is more suitable to capture binders (Sections 3-4).
2. We define λ-calculus as a matching logic theory Γ^λ (Section 6).
   - **Key observation**: $\lambda x. e$ does two things: create the binding and build the term.
   - $[x{:}Var]\ e \equiv \text{intension}\ \exists x{:}Var. \langle x, e\rangle$,
     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 Γ^λ 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).
      $\Gamma^\lambda \vDash e_1 = e_2$ iff $\Gamma^\lambda \vdash e_1 = e_2$
   c. (Representative Completeness, pp. 22, Section 8.2.2).
      For any λ-theory T, there is a matching logic model $M_T \vDash \Gamma^\lambda$
      such that $T \vdash_\lambda e_1 = e_2$ iff $M_T \vDash e_1 = e_2$.
   d. (Capturing All Models, pp. 19, Lemma 32).
      For any λ-calculus (concrete ccc) model A, there is a matching logic model $M_A \vDash \Gamma^\lambda$
      such that $A \vDash_\lambda e_1 = e_2$ iff $M_T \vDash e_1 = e_2$.
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 $\varphi ::= x \mid X \mid \sigma \mid \varphi_1\, \varphi_2 \mid \bot \mid \varphi_1 \to \varphi_2 \mid \exists x.\, \varphi$

- x: element variables (ranging over individual elements)
- X: set variables (ranging over sets)
- σ: (set) symbols
- $\varphi_1\, \varphi_2$: (built-in) application
- $\bot$, $\varphi_1 \to \varphi_2$: propositional constraints
- $\exists x.\, \varphi$: quantification

- The pattern matching semantics:

A pattern φ 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 $\_ \cdot \_ : M \times M \to \mathcal{P}(M)$;
  - a symbol interpretation $\sigma_M \subseteq M$ for every symbol σ;
  - given a valuation ρ such that $\rho(x) \in M$ and $\rho(X) \subseteq M$, we define pattern interpretation $|\varphi|_\rho$ as (see right)

pattern interpretation

$|x|_\rho = \{\rho(x)\} \quad |X|_\rho = \rho(X) \quad |\sigma|_\rho = \sigma_M$

$|\bot|_\rho = \emptyset$

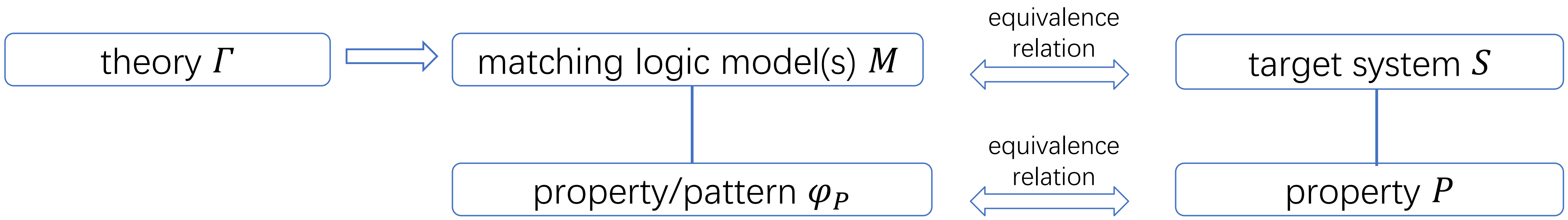
$|\varphi_1 \to \varphi_2|_\rho = M \setminus (|\varphi_1|_\rho \setminus |\varphi_2|_\rho)$

$$|\varphi_1\, \varphi_2|_\rho = \bigcup_{a_1 \in |\varphi_1|_\rho,\, a_2 \in |\varphi_2|_\rho} a_1 \cdot a_2$$

$$|\exists x.\, \varphi|_\rho = \bigcup_{a \in M} |\varphi|_{\rho[a/x]}$$

Matching Logic Theories

- We use a theory Γ 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 Γ, if all axioms ψ in Γ hold in M, i.e., $|\psi|_\rho = M$ for all valuations ρ.



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 λ-calculus, System F, etc.

Theory of λ-Calculus: Γ^{λ}

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

α-equivalent representations: $\lambda x_1.\, e[x_1/x]$, $\lambda x_2.\, e[x_2/x]$, $\lambda x_3.\, e[x_3/x]$, $\cdots$

argument-value pairs: $\langle x_1, e[x_1/x]\rangle$, $\langle x_2, e[x_2/x]\rangle$, $\langle x_3, e[x_3/x]\rangle$, $\cdots$

$$|\varphi_1\, \varphi_2|_\rho = \bigcup_{a_1 \in |\varphi_1|_\rho,\, a_2 \in |\varphi_2|_\rho} a_1 \cdot a_2$$

$$|\exists x.\, \varphi|_\rho = \bigcup_{a \in M} |\varphi|_{\rho[a/x]}$$

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: $\mathbf{intension}\ \exists x{:}\, Var.\, \langle x, e\rangle$

we introduce notation $[x{:}\, Var]\ e \equiv \mathbf{intension}\ \exists x{:}\, Var.\, \langle x, e\rangle$

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

The matching logic encoding of $\lambda x.\, e$ is $\mathbf{lambda}\ [x{:}\, Var]\ e$

where $\mathbf{lambda}$ is a normal symbol/constructor

Theory Γ^{λ} and Its Correctness

	λ-calculus	encoding →	matching logic (within theory Γ^{λ})
variables	x	→	x
application	$e_1\ e_2$	→	$e_1\ e_2$
abstraction	$\lambda x. e$	→	$\lambda x. e \equiv \mathrm{lambda}\,[x{:}Var]\ e$
beta-deduction	$(\lambda x. e)e' = e[e'/x]$	→	$(\lambda x. e)e' = e[e'/x]$

abstraction defined as a notation (syntactic sugar)

beta-reduction added as an axiom verbatim

equivalence: $\vdash_{\lambda} e_1 = e_2$ ⇄ (if and only if) $\Gamma^{\lambda} \vdash e_1 = e_2$ ⇄ (if and only if) $\Gamma^{\lambda} \vDash e_1 = e_2$

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 λ-calculus thoroughly and gave the encoding $\lambda x. e \equiv \mathbf{lambda}\ [x{:}Var]\ e$. 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 π-calculus.

The conference paper: http://fsl.cs.illinois.edu/FSL/papers/2020/chen-rosu-2020-icfp/chen-rosu-2020-icfp-public.pdf
The companion technical report (containing all proof details): http://hdl.handle.net/2142/106608