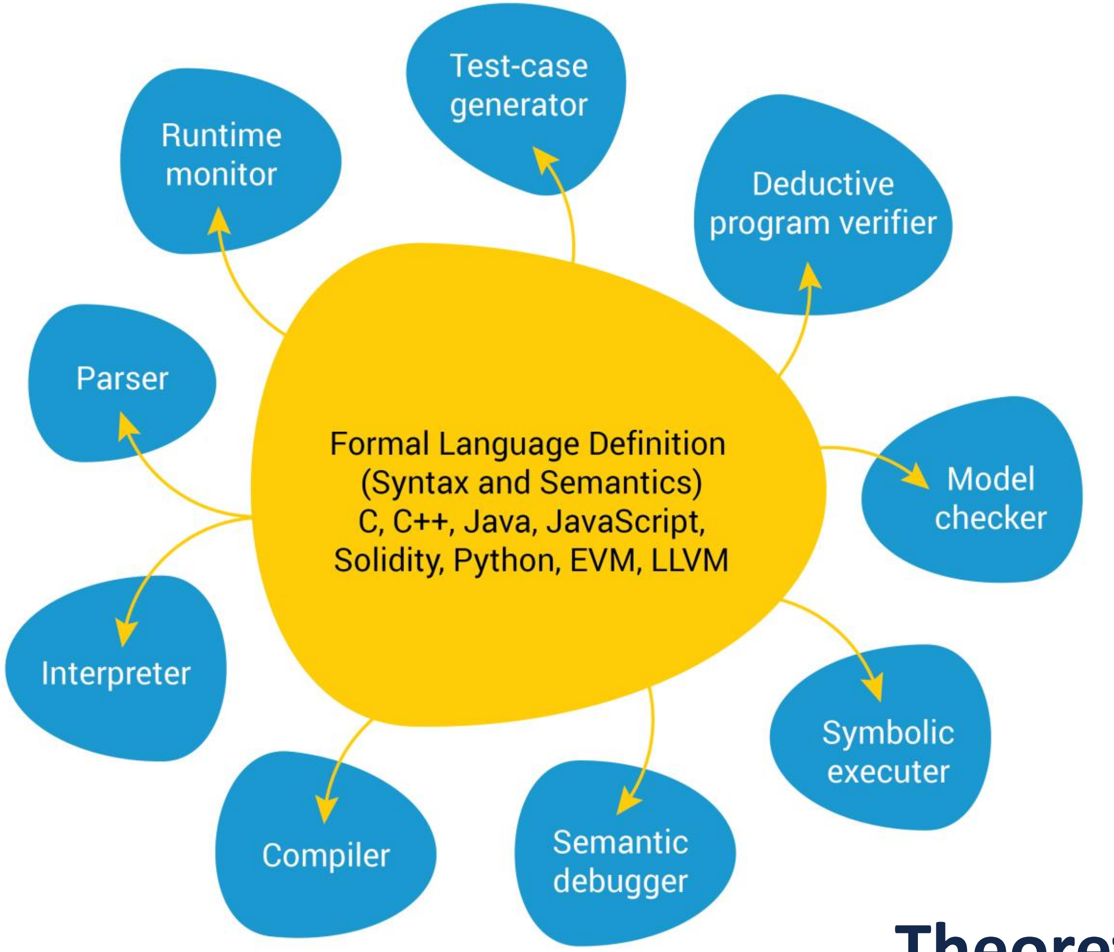
A General Approach to Define Binders using Matching Logic

Xiaohong Chen and Grigore Rosu

{xc3,grosu}@illinois.edu

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

Motivation: K Formal Semantics Framework

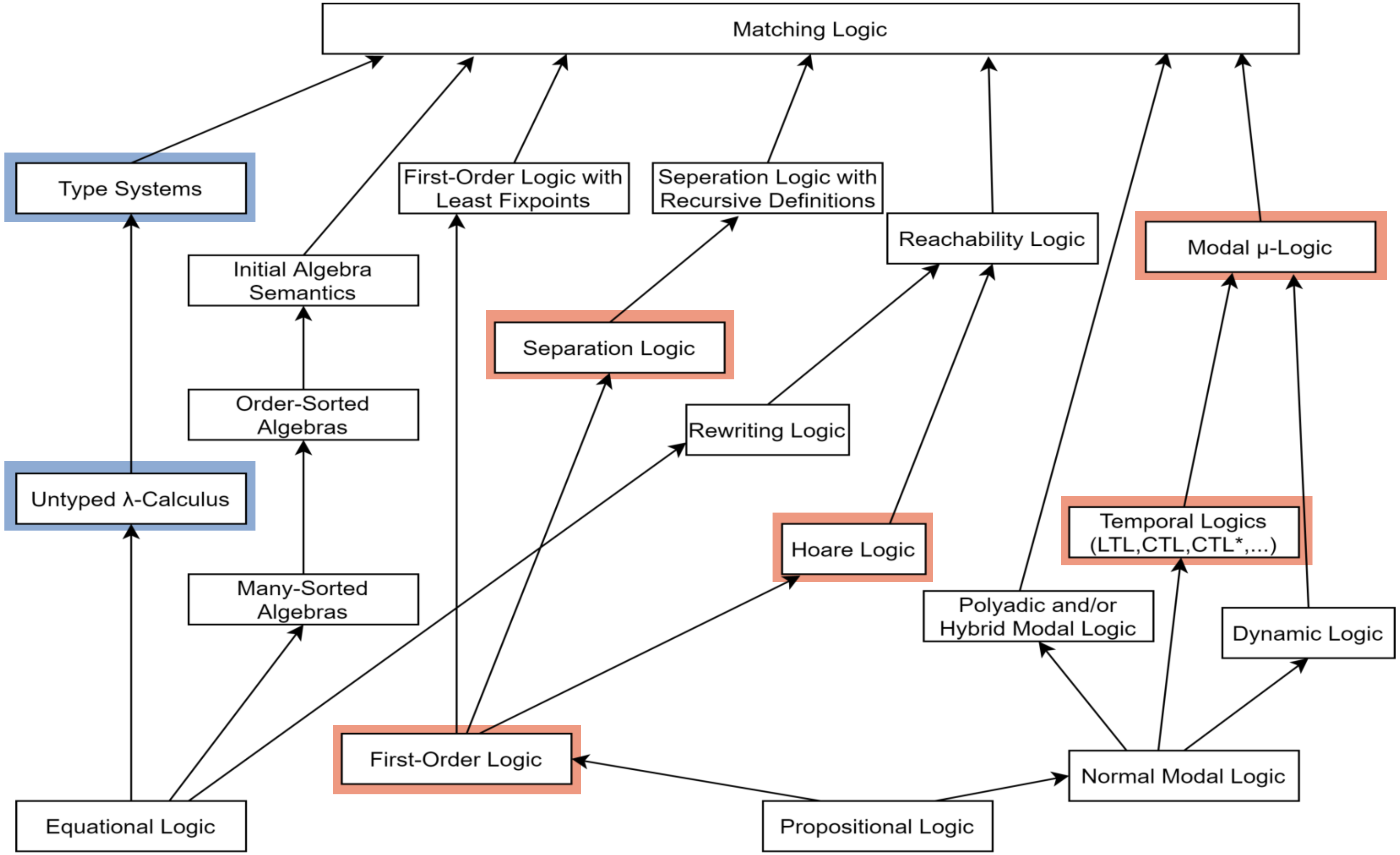


- K is a formal semantics framework
- K has been used to define real-world languages
  - C, Java, JavaScript, Python, EVM, Solidity, …
- K makes it easy to define PL syntax & semantics
  - Including defining binders

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

Theoretical Question: What does [binder] mean?

Matching Logic: Logical Foundation of K



- Previous work [...,LMCS'17,LICS'19]
  - 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
  - Various type systems
  - ...

Main Contribution

1. A simple variant of matching logic that is more suitable for defining binders (sections 3-5).

Then, taking λ-calculus as an example:

2. A matching logic theory Γ^λ (section 6) and an encoding of λ-expressions:
$$\lambda x. e \equiv \mathrm{lambda}\ [x{:}\,Var]\ e$$

3. A set of theorems that establish the correctness of the encoding:
   a. (Conservative Extension, Theorem 36) $\vdash_\lambda e_1 = e_2$ *iff* $\Gamma^\lambda \vdash e_1 = e_2$
   b. (Deductive Completeness, Theorem 36) $\Gamma^\lambda \vdash e_1 = e_2$ *iff* $\Gamma^\lambda \vDash e_1 = e_2$
   c. (Representative Completeness, 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, Lemma 32).
      For any λ-calculus 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. Generalization to other systems with binders: System F, pure type systems, … in a unifying way (section 9).

Straightforward Encoding: Binders = (1) Creating a binding + (2) Building a term

Matching Logic Overview

- A simple logic focused on pattern matching

patterns $\varphi ::= x \mid X \mid \sigma \mid \varphi_1\ \varphi_2 \mid \bot \mid \varphi_1 \to \varphi_2 \mid \exists x.\, \varphi$

Variables | symbols | (built-in) application | propositional constraints | quantification

- Patterns can be matched by zero, one, or more elements.

  - $zero$
  - one
  - $zero \vee one$
  - $succ\ n$
  - $even$
  - $succ\ even$
  - $mult3$
  - $even \wedge mult3$
  - $1 \mapsto 2$
  - $1 \mapsto 2 * 2 \mapsto 3$
  - $\exists x.\, 1 \mapsto x \wedge x \geq 42$
  - $h_1 -\!\!* \ h_2 \equiv \exists h\,.\, h \wedge (h * h_1 \subseteq h_2)$

Matching Logic Theories

- A theory is a collection of symbols, notations, and a set of axioms about them.
- **Example**: Γ^{Nat}, the theory of natural numbers
  - $zero, succ, plus$: symbols
  - $\exists x. zero = x$, i.e., $zero$ is matched by exactly one element (i.e., it is a FOL-style term)
  - $\forall x \exists y. succ\ x = y$, i.e., $succ$ is a FOL-style function
  - $\forall x. zero \neq succ\ x$
  - $\forall x \forall y. succ\ x = succ\ y \ \rightarrow x = y$
  - $x + y \equiv plus\ x\ y$, just a notation for better readability
  - $\forall y.\ plus\ zero\ y = y$
  - $\forall x \forall y. plus\ (succ\ x)\ y = succ\ (plus\ x\ y)$
- In the paper, we defined many basic theories:
  - Γ^{Pair}, the theory of pairs. $\langle x, y \rangle$ represents the pair of x and y
  - Γ^{Sort}, the theory of sorts. $\forall x{:}\, Nat. \varphi$ and $\exists x{:}\, Nat. \varphi$
  - $\Gamma^{Function}$, the theory of functions. $succ{:}\, Nat \ \rightarrow Nat$ and $plus{:}\, Nat \times Nat \rightarrow Nat$

Theory of λ-Calculus Γ^{λ}

<u>λ expressions</u> $e ::= x \mid e_1\, e_2 \mid \lambda x.\, e$

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

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

the <u>**set**</u> of all pairs (i.e., the <u>graph</u>): $\exists x{:}\, Var.\, \langle x, e\rangle$ the binding of x in e is created by the $\exists$-binder of matching logic

Therefore, we let

- $[x{:}\, Var]\; e \equiv \text{intension}\ \exists x{:}\, Var.\, \langle x, e\rangle$
- $\lambda x.\, e \equiv \text{lambda}\ [x{:}\, Var]\; e$

Encoding of λ-Expressions and Its Correctness

	λ-calculus	$\xrightarrow{\text{encoding}}$	matching logic theory Γ^{λ}
variables	x	$\longrightarrow$	x
application	$e_1\ e_2$	$\longrightarrow$	$e_1\ e_2$
abstraction	$\lambda x.\, e$	$\longrightarrow$	$\lambda x.\, e \equiv \text{lambda}\ [x{:}\,Var]\ e$
β-reduction	$(\lambda x.\, e)e' = e[e'/x]$	$\longrightarrow$	$(\lambda x.\, e)e' = e[e'/x]$, an axiom of Γ^{λ}

Equivalence Theorem(s)

$$\vdash_{\lambda} e_1 = e_2 \quad \xrightleftharpoons{\text{if and only if}} \quad \Gamma^{\lambda} \vdash e_1 = e_2 \quad \xrightleftharpoons{\text{if and only if}} \quad \Gamma^{\lambda} \vDash e_1 = e_2$$

λ-calculus reasoning — matching logic reasoning — matching logic models

Generalization

It's easy to generalize Γ^{λ} to other binder-featured systems.

- $\nu x.\, e \equiv \mathrm{nu}\ [x]\ e$; new process name creation in π-calculus;
- $\Pi t.\, e \equiv \mathrm{Pi}\ [t]\ e$; Π-type constructor in System F;
- $\lambda x{:}\, e_1.\, e_2 \equiv \mathrm{lambda}\ ([x]\ e_2)\ e_1$; typed functions in pure type systems.

We give a systematic treatment of all the above via Term-Generic Logic (TGL); check our paper for more details (section 9).

Conclusion

- We proposed a general approach to defining binders in matching logic, the foundation of K.
- We proposed a simple variant of matching logic.
- We studied untyped λ-calculus and proposed the encoding
$$\lambda x. e \equiv \text{lambda}\ [x{:}\,Var]\ e$$
- We proved the correctness of the encoding.
- We generalized the encoding to other systems with binders in a systematic way.
- For more details, read our papers

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