An Ideal Language Framework Vision

We pursue it with the K framework [www.kframework.org]

K scales.

JavaScript ES5: by Park etal [PLDI’15]
Passes existing conformance test suite (2872 programs)
Found (confirmed) bugs in Chrome, IE, Firefox, Safari

Java 1.4: by Bogdanas etal [POPL’15]

x86: by Dasgupta etal [PLDI’19]

C11: Ellison etal [POPL’12, PLDI’15]
192 different types of undefined behavior
10,000+ program tests (gcc torture tests, obfuscated C, …)
Commercialized by startup (Runtime Verification, Inc.)

EVM [CSF’18], Solidity, IELE, Plutus, Vyper, …
An Ideal Language Framework Vision

We pursue it with the K framework [www.kframework.org]

Currently, K uses Matching Logic to specify static structures of programs and Reachability Logic to specify dynamic (reachability) properties.

Can we unify Matching Logic and Reachability Logic so that K has a uniform foundation?

Can we have a more powerful logic that goes beyond Reachability Logic (e.g., liveness properties as in LTL/CTL)?
Family of Logics

Matching $\mu$-Logic serves as a unifying foundation.

- General-purpose logic
- "Static" logic
- "Dynamic" logic

Diagram:
- First-Order Logic with Least Fixpoints
- Separation Logic with Recursive Definitions
- Rewriting Logic
- Hoare Logic
- Matching Logic
- Separation Logic
- Propositional Logic
- Matching $\mu$-Logic
- Reachability Logic
- Modal $\mu$-Logic
- Temporal Logics (LTL, CTL, CTL*, ...)
- Dynamic Logic
- Polyadic and/or Hybrid Modal Logic
- Normal Modal Logic
Talk Overview

• Background:
  • Towards an Ideal Language Framework
  • The need for a uniform and more powerful logic

• Matching $\mu$-Logic:
  • Syntax, Semantics, Proof System

• Applications (“static” and “dynamic”)
  • Reasoning about Constructors and Term Algebras;
  • Reasoning about Transition Systems;
    • Subsuming Modal Logic variants
    • Subsuming Reachability Logic

• Conclusion
  • Matching $\mu$-Logic serves as a unifying foundation
Matching $\mu$-Logic Has Simple Syntax
(7 syntactic constructs)

Patterns (of each sort $s$)

$\varphi_s ::= x : s \in \text{EVar}_s \mid X : s \in \text{SVar}_s$

Set variables

$\sigma(\varphi_{s_1}, \ldots, \varphi_{s_n})$ with $\sigma \in \Sigma_{s_1 \ldots s_n, s}$

Element variables

$\varphi_s \land \varphi_s \mid \neg \varphi_s$

Structures

$\exists z : s' . \varphi_s \mid \mu Z : s . \varphi_s$ if $\varphi_s$ is positive in $Z : s$
Matching \( \mu \)-Logic Semantics: Pattern Matching

\[
\varphi_s ::= x : s \in EVar_s \mid X : s \in SVar_s \mid \sigma(\varphi_{s_1}, \ldots, \varphi_{s_n}) \\
\mid \varphi_s \land \varphi_s \mid \neg \varphi_s \mid \exists z : s' . \varphi_s \mid \mu Z : s . \varphi_s
\]

A Model

\[
\mathcal{M} = (\{ M_s \}_{s \in \text{Sorts}}, \{ \sigma M \}_{\sigma \in \text{Symbols}})
\]

\[
\sigma_M : M_{s_1} \times \cdots \times M_{s_n} \rightarrow \mathcal{P}(M_s)
\]

**Pattern** = The set of elements that **match** it.

\( \land \) as intersection, \( \neg \) as complement, \( \exists \) as union over all \( x \), \( \mu \) as the least fixpoint (examples given later)
Derived Constructs

\[ \varphi_1 \lor \varphi_2 \equiv \neg (\neg \varphi_1 \land \neg \varphi_2) \]

\[ \varphi_1 \rightarrow \varphi_2 \equiv \neg \varphi_1 \lor \varphi_2 \]

\[ \varphi_1 \leftrightarrow \varphi_2 \equiv (\varphi_1 \rightarrow \varphi_2) \land (\varphi_2 \rightarrow \varphi_1) \]

\[ \forall x : s. \varphi \equiv \neg \exists x : s. \neg \varphi \]

\[ T_s \equiv \exists x : s. x : s \]

\[ \bot_s \equiv \neg T_s \]

\[ \nu X : s. \varphi_s \equiv \neg \mu X : s. \neg \varphi_s [\neg X : s / X : s] \]
Recovering Known Notions, Axiomatically

**Definedness**

\[
\left[ \_ \right]_{s}^{s'} \in \sum_{s,s'}
\]

\[
\left[ x:s \right]_{s}^{s'}
\]

\[
\left[ \varphi \right]_{s}^{s'}
\]

- Definedness symbol
- Definedness pattern/axiom
- Either the empty set (when \(\varphi\) empty) or otherwise the total set (when \(\varphi\) non-empty)

**Totality**

\[
\left[ \varphi \right]_{s}^{s'} \equiv \neg \left[ \neg \varphi \right]_{s}^{s'}
\]

Sort subscripts and superscripts can be inferred from the context, so we do not write them

**Equality, Membership, Inclusion**

\[
x \in_{s}^{s'} \varphi \equiv \left[ x \land \varphi \right]_{s}^{s'}
\]

\[
\varphi_1 \subseteq_{s}^{s'} \varphi_2 \equiv \left[ \varphi_1 \rightarrow \varphi_2 \right]_{s}^{s'}
\]

\[
\varphi_1 =_{s}^{s'} \varphi_2 \equiv \left[ \varphi_1 \leftrightarrow \varphi_2 \right]_{s}^{s'}
\]

- Gray area matches \(\varphi_1 \rightarrow \varphi_2\)
- Gray area matches \(\varphi_1 \leftrightarrow \varphi_2\)
Functions (recovering “operation” symbols)

\[ \sigma \in \sum_{s_1 \ldots s_n, s} \]
\[ \exists y . \sigma(x_1, \ldots, x_n) = y \]

Similarly we can define partial functions, total relations, etc. Algebraic specification and FOL subsumed notationally. For example:

\[ 0 : \rightarrow \text{Nat} \]
\[ \text{succ} : \text{Nat} \rightarrow \text{Nat} \]
\[ \text{plus} : \text{Nat} \times \text{Nat} \rightarrow \text{Nat} \]
\[ \text{plus}(0, y) = y \]
\[ \text{plus}(\text{succ}(x), y) = \text{succ}(\text{plus}(x, y)) \]
Examples of Patterns

- $x$: $s$ or simply $x$ matched by singletons (of sort $s$);
- $X$: $s$ or simply $X$ matched by any sets (of sort $s$);
- $\text{succ}(x)$ matched by the successor of $x$;
- $0 \lor \exists x. \text{succ}(x)$ matched by zero or successors;
- $\mu N. 0 \lor \text{succ}(N)$ matched by all natural numbers;
  - The smallest solution of the equation $N = 0 \lor \text{succ}(N)$
  - $N = \{0, \text{succ}(0), \text{succ}(\text{succ}(0)), \ldots\}$
Term Algebras in Matching $\mu$-Logic

Consider the term algebra generated by a set of constructors $C = \{c_i \in \sum_{\text{arity}(c_i) \text{ times}} \text{Term} \ldots \text{Term}, \text{Term} \}$

For example,
Natural numbers = Term algebra freely generated by $\{0, \text{succ}\}$.
Lists (inductive data structure) = Term algebra freely generate by $\{\text{nil}, \text{cons}\}$.

The set of all terms is precisely captured by the following pattern/axiom:

$\mu D. \bigvee_{c \in C} c(D, \ldots, D)$

For example,
The set of all natural numbers $\mu N. 0 \lor \text{succ}(N)$
The set of all lists $\mu L. \text{nil} \lor \text{cons}(T_N, L)$. 
Separation logic = Matching logic [Map]

• Consider map model, with some useful axioms

\[
\begin{align*}
\_ \mapsto \_ : \text{Nat} \times \text{Nat} & \to \text{Map} \\
\text{emp} : & \to \text{Map} \\
\_ \star \_ : \text{Map} \times \text{Map} & \to \text{Map} \\
0 \mapsto a = \bot \\
\end{align*}
\]

\[
\begin{align*}
\text{emp} \star H & = H \\
H_1 \star H_2 & = H_2 \star H_1 \\
(H_1 \star H_2) \star H_3 & = H_1 \star (H_2 \star H_3) \\
x \mapsto a \star x & \mapsto b = \bot
\end{align*}
\]

\[
\begin{align*}
\_ \mapsto [\_] : \text{Nat} \times \text{Seq} & \to \text{Map} \\
x \mapsto [\varepsilon] & = \text{emp} \\
x \mapsto [a, S] & = x \mapsto a \star (x + 1) \mapsto [S]
\end{align*}
\]

• Then we can define map patterns “a la SL”

\[
\begin{align*}
\text{list} \in \Sigma_{\text{Nat} \times \text{Seq}, \text{Map}} \\
\text{list}(0, \varepsilon) & = \text{emp} \\
\text{list}(x, n \cdot S) & = \exists z. x \mapsto [n, z] \star \text{list}(z, S)
\end{align*}
\]
Separation Logic Derivations Using the Generic Matching Logic Proof System (to be shown next)

• Sample derivation for the “separation logic” theory:

\[ 1 \leftrightarrow 5 \cdot 2 \leftrightarrow 0 \cdot 7 \leftrightarrow 9 \cdot 8 \leftrightarrow 1 \]
\[ 1 \leftrightarrow [5, 0] \cdot list(0, \varepsilon) \cdot 7 \leftrightarrow [9, 1] \]
\[ list(1, 5 \cdot \varepsilon) \cdot 7 \leftrightarrow [9, 1] \]
\[ \exists z \cdot 7 \leftrightarrow [9, z] \land list(z, 5) \]

\[ = 1 \leftrightarrow [5, 0] \cdot 7 \leftrightarrow [9, 1] \]
\[ \rightarrow (\exists z \cdot 1 \leftrightarrow [5, z] \cdot list(z, \varepsilon) \cdot 7 \leftrightarrow [9, 1] \]
\[ = list(1, 5) \cdot 7 \leftrightarrow [9, 1] \]
\[ = list(7, 9 \cdot 5) \]

• Local reasoning globalized (“structural framing” for free!)
  • Above derivation can be lifted to whole configuration
Matching $\mu$-Logic Has Simple Proof System (FOL plus 9 additional proof rules)

<table>
<thead>
<tr>
<th>Rule Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>(Propositional Tautology)</strong></td>
<td>$\varphi$ if $\varphi$ is a propositional tautology over patterns $\varphi_1 \varphi_1 \rightarrow \varphi_2$</td>
</tr>
<tr>
<td><strong>(Modus Ponens)</strong></td>
<td>$\varphi_2$</td>
</tr>
<tr>
<td><strong>(∃-Quantifier)</strong></td>
<td>$\varphi[y/x] \rightarrow \exists x.\varphi$</td>
</tr>
<tr>
<td><strong>(∃-Generalization)</strong></td>
<td>$\varphi_1 \rightarrow \varphi_2$ if $x \notin FV(\varphi_2)$</td>
</tr>
<tr>
<td><strong>(Propagation$_\bot$)</strong></td>
<td>$C_\sigma[\bot] \rightarrow \bot$</td>
</tr>
<tr>
<td><strong>(Propagation$_\lor$)</strong></td>
<td>$C_\sigma[\varphi_1 \lor \varphi_2] \rightarrow C_\sigma[\varphi_1] \lor C_\sigma[\varphi_2]$</td>
</tr>
<tr>
<td><strong>(Propagation$_\exists$)</strong></td>
<td>$C_\sigma[\exists x.\varphi] \rightarrow \exists x.\ C_\sigma[\varphi]$ if $x \notin FV(C_\sigma[\exists x.\varphi])$</td>
</tr>
<tr>
<td><strong>(Framing)</strong></td>
<td>$C_\sigma[\varphi_1] \rightarrow C_\sigma[\varphi_2]$</td>
</tr>
<tr>
<td><strong>(Existence)</strong></td>
<td>$\exists x. x$</td>
</tr>
<tr>
<td><strong>(Singleton Variable)</strong></td>
<td>$\neg(C_1[x \land \varphi] \land C_2[x \land \neg \varphi])$ where $C_1$ and $C_2$ are nested symbol contexts.</td>
</tr>
<tr>
<td><strong>(Set Variable Substitution)</strong></td>
<td>$\varphi[\psi/X]$</td>
</tr>
<tr>
<td><strong>(Pre-Fixpoint)</strong></td>
<td>$\varphi[\mu X.\varphi/X] \rightarrow \mu X.\varphi$</td>
</tr>
<tr>
<td><strong>(Knaster-Tarski)</strong></td>
<td>$\varphi[\psi/X] \rightarrow \psi$</td>
</tr>
</tbody>
</table>

- **Standard FOL reasoning**
- **Frame reasoning about symbols**
- **Technical rules (completeness)**
- **Standard fixpoint reasoning**
Matching $\mu$-Logic Reasoning is Powerful

Peano Induction Rule:
\[
\varphi(0) \land \forall y. (\varphi(y) \to \varphi(\text{succ}(y))) \to \forall x. \varphi(x)
\]

... is a theorem in Matching $\mu$-Logic:
Let $\Phi \equiv \exists z. (z \land \varphi(z))$.

\[
\forall x. \varphi(x) \iff \forall x. x \in \Phi,
\]
(Inductive Domain)

\[
\iff (\mu N. 0 \lor \text{succ}(N)) \to \Phi, \text{ by }
\]
\[
\iff 0 \lor \text{succ}(\Phi) \to \Phi
\]

\[
\iff \text{both } 0 \to \Phi, \text{ i.e., } \varphi(0) \text{ holds, and } \text{succ}(\Phi) \to \Phi, \text{ i.e., } \varphi(y) \to \varphi(\text{succ}(y))
\]

(FOL formula $\varphi(x)$)

matched by all numbers where $\varphi$ holds, i.e., $\varphi(z)$ iff $z \in \Phi$

(Knaster-Tarski)
\[
\frac{\varphi[\psi / X] \to \psi}{\mu X. \varphi \to \psi}
\]
Outline

We’ve seen that matching $\mu$-Logic supports
✓ Heap patterns $list(x)$
✓ Term algebras; inductive structures and reasoning
✓ Standard FOL reasoning and fixpoint reasoning
✓ Frame reasoning for free
✓ Peano induction as an instance

We’ll see next...
• Transition systems
• Modal $\mu$-logic, LTL, CTL as instances
• Reachability logic as instances
Transition Systems

\[ S = (S, R) \] with a *binary transition relation* \( R \subseteq S \times S \)

In matching \( \mu \)-Logic, define:

- A sort *State* of states with carrier set \( M_{State} = S \);
- A symbol \( \bullet \in \Sigma_{State,State} \), called *one-path next*, with interpretation
  \[
  \bullet_S : S \to \mathcal{P}(S) \quad \text{with} \quad \bullet_S(t) = \{ s \in S \mid s R t \}
  \]

\[
\begin{array}{cccccc}
\ldots & s & \xrightarrow{R} & s' & \xrightarrow{R} & s'' & \ldots \\
\text{states} &&&&&& \\
\bullet \bullet \varphi & \bullet \varphi & \varphi & \ldots \\
\text{patterns}
\end{array}
\]

Transition Systems

= Matching \( \mu \)-Logic with one sort and one unary symbol.
Useful Sugar about Transition Systems

“one-path next” \( \Diamond \varphi \)

“all-path next” \( \square \varphi \equiv \neg \Diamond \neg \varphi \)

“exists a next state in \( \varphi \)”

“all next states in \( \varphi \)”

“eventually” \( \Diamond \varphi \equiv \mu X. \varphi \lor \Diamond X \)

“always” \( \square \varphi \equiv \nu X. \varphi \land \square X \)

“(strong) until” \( \varphi_1 U \varphi_2 \equiv \mu X. \varphi_2 \lor (\varphi_1 \land \Diamond X) \)

“well-founded” \( \text{WF} \equiv \mu X. \Diamond X \quad // \text{no infinite paths} \)
Modal \( \mu \)-Logic, LTL, CTL, ... = Matching \( \mu \)-Logic theories

\((\text{INF})\) \(\Diamond \top\) \quad \text{All states are some states' predecessors: no stopped states}

\((\text{FIN})\) \(\text{WF} \equiv \mu X . \circ X\) \quad \text{All states are well-founded}

\((\text{LIN})\) \(\Diamond X \rightarrow \circ X\) \quad \text{If } s \text{ can go to } t \text{ then } s \text{ can only go to } t: \text{linear future (no branching)}

<table>
<thead>
<tr>
<th>Target logic</th>
<th>Assumption on traces</th>
<th>Axioms required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modal ( \mu )-Logic</td>
<td>Any traces</td>
<td>No axioms</td>
</tr>
<tr>
<td>Infinite-trace LTL</td>
<td>Infinite and linear traces</td>
<td>(INF)+(LIN)</td>
</tr>
<tr>
<td>Finite-trace LTL</td>
<td>Finite and linear traces</td>
<td>(FIN)+(LIN)</td>
</tr>
<tr>
<td>CTL</td>
<td>Infinite traces</td>
<td>(INF)</td>
</tr>
</tbody>
</table>
Reachability Logic: Semantics of K

A reachability rule:

$\varphi \Rightarrow \varphi'$ with $\varphi$ and $\varphi'$ configuration patterns

Can express operational semantics:

$\langle \langle x = i; s \rangle_k, \langle x \mapsto j, e \rangle_{env}, c \rangle_{cfg}$

$\Rightarrow \langle \langle s \rangle_k, \langle x \mapsto i, e \rangle_{env}, c \rangle_{cfg}$

Can express Hoare triples:

$\{\psi\} \text{code} \{\psi'\}$

$\exists X_{code} (\langle \text{code}, \sigma_{X_{code}} \rangle \land \psi_X)$

$\Rightarrow \exists X_{code} (\langle \text{skip}, \sigma_{X_{code}} \rangle \land \psi'_X)$
K = (Best Effort) Implementation of Reachability Logic

Evaluated it with the existing semantics of C, Java, JavaScript, Ethereum VM, ..., and many tricky programs. Performance acceptable.
Reachability Logic
= Matching $\mu$-Logic fragment

Semantically, $\varphi_1 \Rightarrow \varphi_2$ means that for all configurations $\gamma_1$ matching $\varphi_1$:

- either there exists $\gamma_2$ matching $\varphi_2$ such that $\gamma_1$ reaches $\gamma_2$ (in finite steps);
- or there exists an infinite execution path from $\gamma_1$;

“weak eventually” $\Diamond_w \varphi \equiv \nu X . \varphi \lor \bullet X$

“reachability” $\varphi_1 \Rightarrow \varphi_2 \equiv \varphi_1 \rightarrow \Diamond_w \varphi_2$
Matching $\mu$-Logic as unifying semantic foundation
Completeness?

On the fragment \textit{without} fixpoints $\mu$-binder:

- \textbf{Theorem}. For theory $\Gamma$ that contains \textit{definedness}
  \[ \Gamma \models \varphi \text{ implies } \Gamma \vdash \varphi. \]

- \textbf{Theorem}. For the empty theory $\emptyset$,
  \[ \emptyset \models \varphi \text{ implies } \emptyset \vdash \varphi. \]

- \textbf{Conjecture}. For all theories $\Gamma$,
  \[ \Gamma \models \varphi \text{ implies } \Gamma \vdash \varphi. \]

On the fragment \textit{without} \textit{FOL} $\exists$-binders:

- \textbf{Conjecture}. $\emptyset \models \varphi$ implies $\emptyset \vdash \varphi$.

- \textbf{Conjecture}. Fragment is \textit{decidable}.

On \textit{full} Matching $\mu$-logic:

- \textbf{Conjecture}. $\Gamma \models_{\text{Henkin}} \varphi$ implies $\Gamma \vdash \varphi$.

\[ [\_]_{s}^{s'} \in \Sigma_{s,s'} \]
\[ [x:s]_{s}^{s'} \]

- \textbf{Conjecture}. Fragment is \textit{nontrivial extension of hybrid modal logic} called \textit{global completeness}.

- \textbf{Conjecture}. Extension of modal $\mu$-logic.

- Henkin semantics or General semantics.