Matching μ-Logic

Xiaohong Chen
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801–2302
Email: xc3@illinois.edu

Grigore Roşu
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801–2302
Email: grosu@illinois.edu

***Abstract*—Matching logic is a logic for specifying and reasoning about structure by means of patterns and pattern matching. This paper makes two contributions. First, it proposes a sound and complete proof system for matching logic in its full generality. Previously, sound and complete deduction for matching logic was known only for particular theories providing equality and membership. Second, it proposes matching μ-logic, an extension of matching logic with a least fixpoint μ-binder. It is shown that matching μ-logic captures as special instances many important logics in mathematics and computer science, including first-order logic with least fixpoints, modal μ-logic as well as dynamic logic and various temporal logics such as infinite/finite-trace linear temporal logic and computation tree logic, and notably reachability logic, the underlying logic of the $\mathbb{K}$ framework for programming language semantics and formal analysis. Matching μ-logic therefore serves as a unifying foundation for specifying and reasoning about fixpoints and induction, programming languages and program specification and verification.**

I. Introduction

Matching logic [1] (shortened as ML) is a first-order logic (FOL) variant for specifying and reasoning about structure by means of patterns and pattern matching. In the practice of *program verification*, ML is used to specify static properties of programs in reachability logic [2] (shortened as RL), which takes an operational semantics of a programming language as axioms and yields a program verifier that can prove any reachability properties of any programs written in that language. As a successful implementation of ML and RL, the $\mathbb{K}$ framework (http://kframework.org) has been used to define the formal semantics of various real languages such as C [3], Java [4], JavaScript [5], and to verify complex program properties [6].

A sound and complete Hilbert-style proof system $\mathcal{P}$ of ML is given in [1], whose proof of completeness is by a reduction to pure predicate logic. However, the proof system $\mathcal{P}$ is only applicable to theories where a set of special *definedness symbols* are given together with appropriate axioms, which can be used to define both equality and membership as derived constructs. This leaves the question of whether there is any proof system of ML that is applicable to *all theories*, open. Our first contribution is to answer this question by proposing a new proof system $\mathcal{H}$ of ML, and show that it is (locally) complete *without requiring definedness or any other symbols*.

Our second and main contribution was stimulated by limitations of RL itself as a logic to reason about dynamic behavior of programs. Specifically, as its name suggests, RL can only define and reason about reachability claims. In particular, it is not capable of expressing liveness or many other interesting properties that temporal or dynamic logics can naturally express. Therefore, we propose *matching μ-logic* (shortened as MmL), which extends ML with a least fixpoint μ-binder. It turns out that MmL subsumes not only RL, but also a variety of common logics/calculi that are used to reason about fixpoints and induction, especially for program verification and model checking, including first-order logic with least fixpoints (LFP) [7], modal μ-logic [8] (as well as various temporal logics [9], [10] and dynamic logic (DL) [11]–[13]). For each of these logics/calculi, we prove a *conservative extension result*, showing that our definitions are faithful.

We organize the rest of the paper as follows. We start with a quick but comprehensive overview of ML in Section II, and then present the new proof system $\mathcal{H}$ in Section III. We present MmL in Section IV, and show how to define recursive symbols as syntactic sugar in Section V. Then we discuss how MmL subsumes all the following: first-order logic with least fixpoints (Section VI); modal μ-logic and its fragment logics (Section VII); reachability logic (Section VIII). We compare with related work and conclude the paper with a proposal of future work in Sections IX and X, respectively.

Due to space limitations, all proofs can be found in [14].

II. Matching Logic Preliminaries

Matching logic (ML) [1] is a variant of many-sorted FOL that makes no distinction between function and predicate symbols, allowing them to uniformly build *patterns*. Patterns define both structural and logical constraints, and are interpreted in models as sets of elements (those that *match* them).

A. Matching logic syntax

Definition 1. A *matching logic signature* or simply a *signature* $\mathbb{\Sigma} = (S, \textsc{Var}, \Sigma)$ is a triple with a nonempty set S of *sorts*, an S-indexed set $\textsc{Var} = \{\textsc{Var}_s\}_{s \in S}$ of countably infinitely many *sorted variables* denoted $x{:}s, y{:}s$, etc., and an $(S^* \times S)$-indexed set $\Sigma = \{\Sigma_{s_1 \ldots s_n, s}\}_{s_1, \ldots, s_n, s \in S}$ of countably many *many-sorted symbols*. When $n = 0$, we write $\sigma \in \Sigma_{\lambda, s}$ and say σ is a *constant*. Matching logic $\mathbb{\Sigma}$*-patterns* or simply *($\mathbb{\Sigma}$-)patterns* are defined inductively for all sorts $s, s', s_1, \ldots, s_n \in S$ as follows:

$$\begin{aligned} \varphi_s ::= x{:}s \in \textsc{Var}_s \mid \varphi_s \wedge \varphi_s \mid \neg \varphi_s \mid \exists x{:}s'.\varphi_s \\ \mid \sigma(\varphi_{s_1}, \ldots, \varphi_{s_n}) \quad \text{if } \sigma \in \Sigma_{s_1 \ldots s_n, s} \end{aligned}$$

978-1-7281-3608-0/19/$31.00 ©2019 IEEE

We use $\textsc{Pattern}^{\mathsf{ML}}(\mathbb{\Sigma}) = \{\textsc{Pattern}^{\mathsf{ML}}_s(\mathbb{\Sigma})\}_{s\in S}$ to denote the S-indexed set of $\mathbb{\Sigma}$-patterns generated by the above grammar (modulo α-equivalence, see later). We feel free to drop the signature $\mathbb{\Sigma}$ and simply write $\textsc{Pattern}^{\mathsf{ML}} = \{\textsc{Pattern}^{\mathsf{ML}}_s\}_{s\in S}$.

Intuitively speaking, patterns evaluate to the sets of elements that *match* them. A variable $x{:}s$ is a pattern that is matched by exactly one element; $\varphi_1 \wedge \varphi_2$ is matched by elements matching both φ_1 and φ_2; $\neg\varphi$ is matched by elements not matching φ; $\exists x{:}s'.\varphi$ is a pattern that allows us to abstract away irrelevant parts (i.e., $x{:}s'$) of the structures, which can match patterns $\sigma(\varphi_{s_1},\ldots,\varphi_{s_n})$. This intuition is formalized in Definition 4.

We often abbreviate $\mathbb{\Sigma} = (S, \textsc{Var}, \Sigma)$ as (S,Σ) or just Σ. When we write a pattern, we assume it is well-formed without explicitly specifying the necessary conditions. When $\sigma \in \Sigma_{\lambda,s}$ is a constant, we write σ to mean the pattern $\sigma()$. We adopt the following derived constructs as syntactic sugar:

$$\begin{array}{ll}
\varphi_1 \vee \varphi_2 \equiv \neg(\neg\varphi_1 \wedge \neg\varphi_2) & \forall x{:}s.\varphi \equiv \neg\exists x{:}s.\neg\varphi \\
\varphi_1 \to \varphi_2 \equiv \neg\varphi_1 \vee \varphi_2 & \top_s \equiv \exists x{:}s.x{:}s \\
\varphi_1 \leftrightarrow \varphi_2 \equiv (\varphi_1 \to \varphi_2) \wedge (\varphi_2 \to \varphi_1) & \bot_s \equiv \neg\top_s
\end{array}$$

Intuitively, $\varphi_1 \vee \varphi_2$ is matched by elements matching φ_1 or φ_2; $\top_s$ is matched by all elements (in the sort universe s); and $\bot_s$ is matched by no elements. The formal semantics of these derived constructs is given in Proposition 5. Standard precedences are adopted to avoid parentheses. The scope of "$\forall$" and "$\exists$" goes as far as possible to the right. We drop sort s whenever possible, so we write $x, \top, \bot$ instead of $x{:}s, \top_s, \bot_s$.

Like in FOL, "$\forall$" and "$\exists$" are *binders*, and we adopt the standard notions of *free variables*, *α-renaming*, and *capture-avoiding substitution*. We let $FV(\varphi)$ denote the set of free variables in φ. When $FV(\varphi) = \emptyset$, we say φ is *closed*. We regard α-equivalent patterns φ and φ' *as the same*, and write $\varphi \equiv \varphi'$. We let $\varphi[\psi/x]$ be the result of substituting ψ for every free occurrence of x in φ, where α-renaming happens implicitly to prevent variable capture. We let $\varphi[\psi_1/x_1,\ldots,\psi_n/x_n]$ be the result of simultaneously substituting $\psi_1,\ldots,\psi_n$ for $x_1,\ldots,x_n$.

B. Matching logic semantics

ML symbols are interpreted as *relations*, and thus ML patterns evaluate to *sets of elements* (those "matching" them).

Definition 2. Given $\mathbb{\Sigma} = (S,\Sigma)$, a *matching logic $\mathbb{\Sigma}$-model* $M = (\{M_s\}_{s\in S}, \{\sigma_M\}_{\sigma\in\Sigma})$, or simply a *($\mathbb{\Sigma}$)-model*, contains

- a nonempty carrier set M_s for each sort $s \in S$;
- an interpretation $\sigma_M : M_{s_1} \times \cdots \times M_{s_n} \to \mathcal{P}(M_s)$ for each $\sigma \in \Sigma_{s_1\ldots s_n,s}$, where $\mathcal{P}(M_s)$ is the powerset of M_s.

We overload the letter M to also mean the S-indexed set $\{M_s\}_{s\in S}$. The usual FOL models are special cases of ML models, where $|\sigma_M(a_1,\ldots,a_n)| = 1$ for all $a_1 \in M_{s_1},\ldots,a_n \in M_{s_n}$. Partial FOL models [15] are also special cases with $|\sigma_M(a_1,\ldots,a_n)| \leq 1$, as we can capture the undefinedness of the partial function σ_M on $a_1,\ldots,a_n$ by $\sigma_M(a_1,\ldots,a_n) = \emptyset$.

We tacitly use the same letter σ_M to mean its *pointwise extension*, $\sigma_M : \mathcal{P}(M_{s_1}) \times \cdots \times \mathcal{P}(M_{s_n}) \to \mathcal{P}(M_s)$, defined as:

$$\sigma_M(A_1,\ldots,A_n) = \bigcup\{\sigma_M(a_1,\ldots,a_n) \mid a_1 \in A_1,\ldots,a_n \in A_n\}$$

for all $A_1 \subseteq M_{s_1},\ldots,A_n \subseteq M_{s_n}$.

Proposition 3. *For all $A_i, A'_i \subseteq M_{s_i}$, $1 \leq i \leq n$, the pointwise extension σ_M has the following* property of propagation*:*

$\sigma_M(A_1,\ldots,A_n) = \emptyset$ *if* $A_i = \emptyset$ *for some* $1 \leq i \leq n$,

$\sigma_M(A_1 \cup A'_1,\ldots,A_n \cup A'_n) = \bigcup_{1\leq i\leq n, B_i\in\{A_i,A'_i\}} \sigma_M(B_1,\ldots,B_n)$,

$\sigma(A_1,\ldots,A_n) \subseteq \sigma(A'_1,\ldots,A'_n)$ *if* $A_i \subseteq A'_i$ *for all* $1 \leq i \leq n$.

Definition 4. Let $\mathbb{\Sigma} = (S, \textsc{Var}, \Sigma)$ and let M be a $\mathbb{\Sigma}$-model. Given a function $\rho\colon \textsc{Var} \to M$, called an *$M$-valuation*, let its *extension* $\bar{\rho}\colon \textsc{Pattern}^{\mathsf{ML}} \to \mathcal{P}(M)$ be inductively defined as:

- $\bar{\rho}(x) = \{\rho(x)\}$, for all $x \in \textsc{Var}_s$;
- $\bar{\rho}(\varphi_1 \wedge \varphi_2) = \bar{\rho}(\varphi_1) \cap \bar{\rho}(\varphi_2)$, for $\varphi_1,\varphi_2 \in \textsc{Pattern}_s$;
- $\bar{\rho}(\neg\varphi) = M_s \setminus \bar{\rho}(\varphi)$, for all $\varphi \in \textsc{Pattern}_s$;
- $\bar{\rho}(\exists x.\varphi) = \bigcup_{a\in M_{s'}} \rho[a/x](\varphi)$, for all $x \in \textsc{Var}_{s'}$;
- $\bar{\rho}(\sigma(\varphi_1,\ldots,\varphi_n)) = \sigma_M(\bar{\rho}(\varphi_1),\ldots,\bar{\rho}(\varphi_n))$, for $\sigma \in \Sigma_{s_1\ldots s_n,s}$;

where "$\setminus$" is set difference and $\rho[a/x]$ denotes the M-valuation ρ' with $\rho'(x) = a$ and $\rho'(y) = \rho(y)$ for all $y \not\equiv x$.

Proposition 5. *The following propositions hold:*

- $\bar{\rho}(\top_s) = M_s$ *and* $\bar{\rho}(\bot_s) = \emptyset$*;*
- $\bar{\rho}(\varphi_1 \vee \varphi_2) = \bar{\rho}(\varphi_1) \cup \bar{\rho}(\varphi_2)$*;*
- $\bar{\rho}(\varphi_1 \to \varphi_2) = M_s \setminus (\bar{\rho}(\varphi_1) \setminus \bar{\rho}(\varphi_2))$*, for* $\varphi_1,\varphi_2 \in \textsc{Pattern}_s$*;*
- $\bar{\rho}(\varphi_1 \leftrightarrow \varphi_2) = M_s \setminus (\underline{\bar{\rho}(\varphi_1)} \triangle \bar{\rho}(\varphi_2))$*, for* $\varphi_1,\varphi_2 \in \textsc{Pattern}_s$*;*
- $\bar{\rho}(\forall x.\varphi) = \bigcap_{a\in M_{s'}} \rho[a/x](\varphi)$*, for all* $x \in \textsc{Var}_{s'}$*;*

where "$\triangle$" is set symmetric difference.

Definition 6. We say pattern φ is *valid* in M, written $M \vDash_{\mathsf{ML}} \varphi$, iff $\bar{\rho}(\varphi) = M$ for all $\rho\colon \textsc{Var} \to M$. Let Γ be a set of patterns called *axioms*. We write $M \vDash_{\mathsf{ML}} \Gamma$ iff $M \vDash_{\mathsf{ML}} \psi$ for all $\psi \in \Gamma$. We write $\Gamma \vDash_{\mathsf{ML}} \varphi$ and say that φ is *valid* in Γ iff $M \vDash_{\mathsf{ML}} \varphi$ for all $M \vDash_{\mathsf{ML}} \Gamma$. We abbreviate $\emptyset \vDash_{\mathsf{ML}} \varphi$ as $\vDash_{\mathsf{ML}} \varphi$. We call the pair $(\mathbb{\Sigma}, \Gamma)$ a *matching logic $\mathbb{\Sigma}$-theory*, or simply a *($\mathbb{\Sigma}$-)theory*. We say that M is *a model of the theory* $(\mathbb{\Sigma}, \Gamma)$ iff $M \vDash_{\mathsf{ML}} \Gamma$.

C. Important notations

Several mathematical instruments of practical importance, such as definedness, totality, equality, membership, set containment, functions and partial functions, and constructors, can all be defined using patterns. We give a compact summary of the definitions and notations that are needed in this paper.

Definition 7. For any (not necessarily distinct) sorts s, s', let us consider a unary symbol $\lceil\_\rceil_s^{s'} \in \Sigma_{s,s'}$, called the *definedness symbol*, and the pattern/axiom $\lceil x{:}s\rceil_s^{s'}$, called (Definedness). We define *totality* "$\lfloor\_\rfloor_s^{s'}$", *equality* "$=_s^{s'}$", *membership* "$\in_s^{s'}$", and *set containment* "$\subseteq_s^{s'}$" as derived constructs:

$$\begin{array}{ll}
\lfloor\varphi\rfloor_s^{s'} \equiv \neg\lceil\neg\varphi\rceil_s^{s'} & \varphi_1 =_s^{s'} \varphi_2 \equiv \lfloor\varphi_1 \leftrightarrow \varphi_2\rfloor_s^{s'} \\
x \in_s^{s'} \varphi \equiv \lceil x \wedge \varphi\rceil_s^{s'} & \varphi_1 \subseteq_s^{s'} \varphi_2 \equiv \lfloor\varphi_1 \to \varphi_2\rfloor_s^{s'}
\end{array}$$

and feel free to drop the (not necessarily distinct) sorts s, s'.

For all M satisfying (Definedness), $(\lceil\_\rceil_s^{s'})_M(a) = M_{s'}$ for all $a \in M_s$ [1, Proposition 5.2]. Thus, for all ρ, we have $\bar{\rho}(\lceil\varphi\rceil_s^{s'}) = M_{s'}$ if $\bar{\rho}(\varphi) \neq \emptyset$, and $\bar{\rho}(\lceil\varphi\rceil_s^{s'}) = \emptyset$ otherwise; i.e., $\lceil\varphi\rceil_s^{s'}$ says, in sort universe s', if φ is defined in universe s. Definition 7 constructs have expected semantics: $\bar{\rho}(\lfloor\varphi\rfloor_s^{s'}) = M_{s'}$ if $\bar{\rho}(\varphi) = M_s$, and $\bar{\rho}(\lfloor\varphi\rfloor_s^{s'}) = \emptyset$ otherwise; $\bar{\rho}(\varphi_1 =_s^{s'} \varphi_2) = M_{s'}$ if $\bar{\rho}(\varphi_1) = \bar{\rho}(\varphi_2)$, and $\bar{\rho}(\varphi_1 =_s^{s'} \varphi_2) = \emptyset$ otherwise; etc.

Functions and *partial functions* can be defined by axioms:

$$\text{(Function)} \qquad \exists y \,.\, \sigma(x_1, \ldots, x_n) = y$$

$$\text{(Partial Function)} \qquad \exists y \,.\, \sigma(x_1, \ldots, x_n) \subseteq y$$

(Function) requires $\sigma(x_1, \ldots, x_n)$ to contain exactly one element and (Partial Function) requires it to contain at most one element (recall that y evaluates to a singleton set). For brevity, we use the function notation $\sigma\colon s_1 \times \cdots \times s_n \to s$ to mean we automatically assume the (Function) axiom of σ. Similarly, partial functions are written as $\sigma\colon s_1 \times \cdots \times s_n \rightharpoonup s$.

Constructors are extensively used in building programs and data, as well as semantic structures to define and reason about languages and programs. They can be characterized in the "no junk, no confusion" spirit [16]. Let $\mathbb{\Sigma} = (S, \Sigma)$ be a signature and $C = \{c_i \in \Sigma_{s_i^1 \ldots s_i^{m_i}, s_i} \mid 1 \leq i \leq n\} \subseteq \Sigma$ be a set of symbols called *constructors*. Consider the following axioms/patterns:

(No Junk) for all sorts $s \in S$:

$$\bigvee_{c_i \in C \text{ with } s_i = s} \exists x_i^1 {:} s_i^1 \ldots \exists x_i^{m_i} {:} s_i^{m_i} \,.\, c_i(x_i^1, \ldots, x_i^{m_i})$$

(No Confusion I) for all $i \neq j$ and $s_i = s_j$:

$$\neg(c_i(x_i^1, \ldots, x_i^{m_i}) \wedge c_j(x_j^1, \ldots, x_j^{m_j}))$$

(No Confusion II) for all $1 \leq i \leq n$:

$$(c_i(x_i^1, \ldots, x_i^{m_i}) \wedge c_i(y_i^1, \ldots, y_i^{m_i})) \to c_i(x_i^1 \wedge y_i^1, \ldots, x_i^{m_i} \wedge y_i^{m_i})$$

Intuitively, (No Junk) says everything is constructed; (No Confusion I) says different constructs build different things; and (No Confusion II) says constructors are injective. We refer to the the last two axioms as (No Confusion).

D. Defining first-order logic in matching logic

Given a FOL signature (S, Σ, Π) with *function symbols* Σ and *predicate symbols* Π, the *syntax* of FOL is given by:

$$t_s ::= x \in \textsc{Var}_s \mid f(t_{s_1}, \ldots, t_{s_n}) \text{ with } f \in \Sigma_{s_1 \ldots s_n, s}$$

$$\varphi ::= \pi(t_{s_1}, \ldots, t_{s_n}) \text{ with } \pi \in \Pi_{s_1 \ldots s_n} \mid \varphi \to \varphi \mid \neg\varphi \mid \forall x . \varphi$$

To subsume the syntax, we define a ML signature $\mathbb{\Sigma}^{\mathsf{FOL}} = (S^{\mathsf{FOL}}, \Sigma^{\mathsf{FOL}})$, where $S^{\mathsf{FOL}} = S \cup \{Pred\}$ contains a distinguished sort *Pred* for FOL formulas and $\Sigma^{\mathsf{FOL}} = \{f\colon s_1 \times \cdots \times s_n \to s \mid f \in \Sigma_{s_1 \ldots s_n, s}\} \cup \{\pi \in \Sigma^{\mathsf{FOL}}_{s_1 \ldots s_n, Pred} \mid \pi \in \Pi_{s_1 \ldots s_n}\}$ contains FOL function symbols as ML functions and FOL predicate symbols as ML symbols that return *Pred*. Let Γ^{FOL} be the resulting $\mathbb{\Sigma}^{\mathsf{FOL}}$-theory. Notice that we use the function notations so Γ^{FOL} contains the (Function) axioms for all $f \in \Sigma^{\mathsf{FOL}}$.

Proposition 8. *All FOL formulas φ are $\mathbb{\Sigma}^{\mathsf{FOL}}$-patterns of sort Pred, and we have $\vDash_{\mathit{FOL}} \varphi$ iff $\Gamma^{\mathit{FOL}} \vDash_{\mathit{ML}} \varphi$ (see [1]).*

E. Matching logic proof system $\mathcal{P}$ with definedness symbols

ML has a *conditional* sound and complete Hilbert-style proof system [1, Fig. 5], here referred to as $\mathcal{P}$. We let $\Gamma \vdash_{\mathcal{P}} \varphi$ denote its provability relation. $\mathcal{P}$ can prove all patterns φ that are valid in Γ *under the condition that* Γ contains definedness symbols and (Definedness) axioms. In fact, $\mathcal{P}$ proof rules use equality "=" and membership "$\in$", both requiring definedness symbols. This means that $\mathcal{P}$ *is not applicable at all* to any theories that do not contain definedness symbols.

We wrap up this section by reviewing the soundness and completeness theorem of $\mathcal{P}$. In Section III, we propose a new ML proof system $\mathcal{H}$ that is sound and (locally) complete *without requiring the theories to contain definedness symbols*.

Theorem 9 (Soundness and completeness of $\mathcal{P}$, see [1])**.** *For all theories Γ containing the definedness symbols and axioms (Definition 7) and all patterns φ, we have $\Gamma \vDash_{\mathit{ML}} \varphi$ iff $\Gamma \vdash_{\mathcal{P}} \varphi$.*

III. A New Proof System of Matching Logic

Our first main contribution is a new ML proof system $\mathcal{H}$ that is sound and (locally) complete *without requiring definedness symbols and axioms*, and thus extends the completeness result in [1], re-stated in Theorem 9. We first need the following:

Definition 10. A *context C* is a pattern with a distinguished placeholder variable $\square$. We write $C[\varphi]$ to mean the result of *replacing $\square$ with φ without any α-renaming*, so free variables in φ may become bound in $C[\varphi]$, *different* from capture-avoiding substitution. A *single symbol context* has the form $C_\sigma \equiv \sigma(\varphi_1, \ldots, \varphi_{i-1}, \square, \varphi_{i+1}, \ldots, \varphi_n)$ where $\sigma \in \Sigma_{s_1 \ldots s_n, s}$ and $\varphi_1, \ldots, \varphi_{i-1}, \varphi_{i+1}, \ldots, \varphi_n$ are patterns of appropriate sorts. A *nested symbol context* is inductively defined as follows:

- $\square$ is a nested symbol context, called the *identity context*;
- if C_σ is a single symbol context, and C is a nested symbol context, then $C_\sigma[C[\square]]$ is a nested symbol context.

Intuitively, a context C is a nested symbol context iff the path to $\square$ in C contains only symbols and no logic connectives.

The proof system $\mathcal{H}$ (Fig. 1, above the double line) has four categories of proof rules. The first consists of all propositional tautologies as axioms and (Modus Ponens). The second completes the (complete) axiomatization of pure predicate logic (two rules); see, e.g., [17]. The third category contains four rules that capture the property of propagation (Proposition 3). The fourth category contains two technical proof rules that are needed for the completeness result of $\mathcal{H}$. Note that unlike $\mathcal{P}$, all proof rules of $\mathcal{H}$ are general rules and do not depend on any special symbols such as the definedness symbols.

Definition 11. For an axiom set Γ and a pattern φ, we write $\Gamma \vdash_{\mathcal{H}} \varphi$ iff φ can be proved by $\mathcal{H}$ with the patterns in Γ as additional axioms. We abbreviate $\emptyset \vdash_{\mathcal{H}} \varphi$ as $\vdash_{\mathcal{H}} \varphi$.

There are two interesting observations about $\mathcal{H}$. First, (Framing) allows us to lift local reasoning through symbol contexts, and thus supports *compositional reasoning* in ML. Second, the propagation axioms plus (Framing) inspire a close relationship between ML and modal logics, where the *ML symbols* and the *modal logic modalities* are dual:

Proposition 12. *Let $\sigma \in \Sigma_{s_1 \ldots s_n, s}$ and define its "dual" as $\bar{\sigma}(\varphi_1, \ldots, \varphi_n) \equiv \neg\sigma(\neg\varphi_1, \ldots, \neg\varphi_n)$. Then we have:*

- (K)*: $\vdash_{\mathcal{H}} \bar{\sigma}(\varphi_1 \to \varphi_1', \ldots, \varphi_n \to \varphi_n')$*
  $\qquad \to (\bar{\sigma}(\varphi_1, \ldots, \varphi_n) \to \bar{\sigma}(\varphi_1', \ldots, \varphi_n'))$;
- (N)*: $\vdash_{\mathcal{H}} \varphi_i$ implies $\vdash_{\mathcal{H}} \bar{\sigma}(\varphi_1, \ldots, \varphi_i, \ldots, \varphi_n)$.*

These rules also appear in [18], [19] as proof rules of polyadic modal logic. When $n = 1$, we obtain the standard (K) *rule and* (N) *rule of normal modal logic [20].*

$\mathcal{H}_\mu$ / $\mathcal{H}$	(Propositional Tautology)	φ if φ is a propositional tautology over patterns of the same sort
	(Modus Ponens)	$\dfrac{\varphi_1 \quad \varphi_1 \to \varphi_2}{\varphi_2}$
	(∃-Quantifier)	$\varphi[y/x] \to \exists x.\varphi$
	(∃-Generalization)	$\dfrac{\varphi_1 \to \varphi_2}{(\exists x.\varphi_1) \to \varphi_2}$ if $x \notin FV(\varphi_2)$
	(Propagation$_\bot$)	$C_\sigma[\bot] \to \bot$
	(Propagation$_\vee$)	$C_\sigma[\varphi_1 \vee \varphi_2] \to C_\sigma[\varphi_1] \vee C_\sigma[\varphi_2]$
	(Propagation$_\exists$)	$C_\sigma[\exists x.\varphi] \to \exists x.C_\sigma[\varphi]$ if $x \notin FV(C_\sigma[\exists x.\varphi])$
	(Framing)	$\dfrac{\varphi_1 \to \varphi_2}{C_\sigma[\varphi_1] \to C_\sigma[\varphi_2]}$
	(Existence)	$\exists x.\, x$
	(Singleton Variable)	$\neg(C_1[x \wedge \varphi] \wedge C_2[x \wedge \neg\varphi])$ where C_1 and C_2 are nested symbol contexts.
$\mathcal{H}_\mu$	(Set Variable Substitution)	$\dfrac{\varphi}{\varphi[\psi/X]}$
	(Pre-Fixpoint)	$\varphi[\mu X.\,\varphi/X] \to \mu X.\varphi$
	(Knaster-Tarski)	$\dfrac{\varphi[\psi/X] \to \psi}{\mu X.\,\varphi \to \psi}$

Fig. 1. Sound and complete proof system $\mathcal{H}$ of matching logic (above the double line) and the proof system $\mathcal{H}_\mu$ of matching μ-logic

We present three important properties about $\mathcal{H}$, all proved in [14]. The first property is the soundness theorem of $\mathcal{H}$.

Theorem 13 (Soundness of $\mathcal{H}$)**.** *$\Gamma \vdash_{\mathcal{H}} \varphi$ implies $\Gamma \vDash_{ML} \varphi$.*

The second property is a version of *deduction theorem* of $\mathcal{H}$ which requires definedness symbols and axioms.

Theorem 14 (Deduction theorem)**.** *For all axiom sets Γ containing* (Definedness) *axioms (see Definition 7) and patterns ψ, φ with ψ closed, we have $\Gamma \cup \{\psi\} \vdash_{\mathcal{H}} \varphi$ iff $\Gamma \cup \vdash_{\mathcal{H}} \lfloor\psi\rfloor \to \varphi$.*

The proof is standard, by induction on the proof length of $\Gamma \cup \{\psi\} \vdash_{\mathcal{H}} \varphi$. Here, we give it an intuitive semantic explanation. Suppose $\Gamma \cup \{\psi\} \vDash_{\mathsf{ML}} \varphi$. Then for all models $M \vDash_{\mathsf{ML}} \Gamma$, if ψ holds then φ also holds (we ignore valuations as ψ is closed). This means $M \vDash_{\mathsf{ML}} \lfloor\psi\rfloor \to \varphi$, as $\lfloor\psi\rfloor$ evaluates to $\emptyset$ iff ψ does not hold in M. Note that $M \vDash_{\mathsf{ML}} \psi \to \varphi$ is too strong as a conclusion, for it requires the evaluation of ψ is always contained in φ, even in models where ψ does not hold.

The third property is that we can prove all proof rules of $\mathcal{P}$ using $\mathcal{H}$ with (Definedness) as axioms. This immediately gives us the following (global) completeness result of $\mathcal{H}$:

Theorem 15. *For all axiom sets Γ containing* (Definedness) *axioms and all patterns φ, we have $\Gamma \vDash_{ML} \varphi$ implies $\Gamma \vdash_{\mathcal{H}} \varphi$.*

Finally, we state our main completeness result for $\mathcal{H}$:

Theorem 16 (Local completeness of $\mathcal{H}$)**.** *$\vDash_{ML} \varphi$ implies $\vdash_{\mathcal{H}} \varphi$.*

Here, "local" means the theory is empty (i.e., no additional axioms); in comparison, Theorem 15 holds for non-empty theories. The proof of Theorem 16 is rather complex (see [14]). We drew inspiration from [21], where a similar result is proved for *hybrid modal logic*, using a mixture of modal and first-order techniques: the ideas of *canonical models* from modal logic and *witnessed sets* from first-order logic. Theorem 16 can be seen as a nontrivial generalization. Specifically, we extend hybrid modal logic with ∀-binder [21] in two directions. First, we consider multiple sorts, each coming with its own universe of worlds and logical infrastructure; the approach in [21] has only one sort, that of "formulas". Second, we allow arbitrarily many modalities of arbitrary arities (see Proposition 12); the approach in [21] only considers the usual, unary "necessity" modality "□" (and its dual "◇"). Polyadic, non-hybrid (i.e., without ∀-binder) variants of modal logic are known (see, e.g., [18]), but at our knowledge our work in this paper is the first to combine polyadic modalities and FOL quantifiers.

The full global completeness of $\mathcal{H}$ is left as future work. See Section IX-B for more discussion.

IV. From Matching Logic to Matching μ-Logic

We extend ML with the least fixpoint μ-binder. We call the extended logic *matching μ-logic* (MmL), and study its syntax, semantics, and proof system. Many definitions, notations, and properties of ML that are introduced in Section II and III also work for MmL, so we only focus on parts where they differ.

A. Matching μ-logic syntax

Definition 17. A *matching μ-logic signature* $\mathbb{\Sigma} = (S, \textsc{Var}, \Sigma)$ or simply a *signature* is the same as a matching logic signature *except that* $\textsc{Var} = \textsc{EVar} \cup \textsc{SVar}$ is now a disjoint union of two S-indexed sets of variables: the *element variables* denoted as $x{:}s$, $y{:}s$, etc. in EVar, and the *set variables* denoted as $X{:}s, Y{:}s$, etc in SVar. *Matching μ-logic $\mathbb{\Sigma}$-patterns*, or simply *($\mathbb{\Sigma}$)-patterns*, are defined inductively for all sorts $s, s' \in S$ as:

$$\begin{aligned} \varphi_s &::= x{:}s \in \textsc{EVar}_s \mid X{:}s \in \textsc{SVar}_s \mid \cdots \\ &\mid \mu X{:}s.\varphi_s \quad \text{if } \varphi_s \text{ is positive in } X{:}s, \end{aligned}$$

where the "..." part is the same as in ML. Note that we only quantify over element variables, not set variables. We say φ_s is *positive* in $X{:}s$ if every free occurrence of $X{:}s$ is under an even number of negations. We let $\textsc{Pattern}(\mathbb{\Sigma}) = \{\textsc{Pattern}_s\}_{s \in S}$ denote the set of all matching μ-logic $\mathbb{\Sigma}$-patterns and feel free to drop the signature $\mathbb{\Sigma}$.

From now on, we tacitly assume we are talking about MmL unless we explicitly say otherwise. Intuitively, element variables are like ML variables in that they evaluate to *elements*, while set variables evaluate to *sets*. The least fixpoint pattern $\mu X{:}s.\, \varphi_s$ gives *the least solution* (under set containment) of the equation $X{:}s = \varphi_s$ of set variable $X{:}s$ (this should be taken as merely intuition at this stage, because we may not have equality in the theories). The condition of positive occurrence guarantees the existence of such a least solution. The notion of free variables, α-renaming, and capture-avoiding substitution are extended to set variables and the μ-binder. The dual version of the least fixpoint μ-binder is the *greatest fixpoint* ν-binder, defined as $\nu X{:}s.\varphi_s \equiv \neg \mu X{:}s.\neg\varphi_s[\neg X{:}s/X{:}s]$, given that φ_s is positive in $X{:}s$, (which implies that $\neg\varphi_s[\neg X{:}s/X{:}s]$ is also positive in $X{:}s$, justifying the definition).

B. Matching μ-logic semantics

We first review a variant of the Knaster-Tarski theorem [22]:

Theorem 18 (Knaster-Tarski)**.** *Let M be a nonempty set and $\mathcal{F}\colon \mathcal{P}(M) \to \mathcal{P}(M)$ be a monotone function, i.e., $\mathcal{F}(A) \subseteq \mathcal{F}(B)$ for all subsets $A \subseteq B$ of M. Then $\mathcal{F}$ has a unique least fixpoint $\mu\mathcal{F}$ and a unique greatest fixpoint $\nu\mathcal{F}$, given as:*

$$\mu\mathcal{F} = \bigcap\{A \in \mathcal{P}(M) \mid \mathcal{F}(A) \subseteq A\},$$
$$\nu\mathcal{F} = \bigcup\{A \in \mathcal{P}(M) \mid A \subseteq \mathcal{F}(A)\}.$$

We call A a pre-fixpoint *of $\mathcal{F}$ whenever $\mathcal{F}(A) \subseteq A$, and a* post-fixpoint *of $\mathcal{F}$ whenever $A \subseteq \mathcal{F}(A)$.*

MmL models are exactly ML models where sorts are associated with their carrier sets and symbols are interpreted as relations. Valuations are extended such that element variables are mapped to elements and set variables are mapped to subsets. Patterns are evaluated in the same way for the ML constructs, but extended with the evaluation of least fixpoint patterns $\mu X{:}s.\, \varphi$ as the *true least fixpoints* in models. Formally:

Definition 19. Let $\mathbb{\Sigma} = (S, \textsc{Var}, \Sigma)$ be a signature with $\textsc{Var} = \textsc{EVar} \cup \textsc{SVar}$, and $M = (\{M_s\}_{s \in S}, \{\sigma_M\}_{\sigma \in \Sigma})$ be a $\mathbb{\Sigma}$-model. A *valuation* $\rho\colon \textsc{Var} \to (M \cup \mathcal{P}(M))$ is a function such that $\rho(x) \in M_s$ for all $x \in \textsc{EVar}_s$ and $\rho(X) \in \mathcal{P}(M_s)$ for all $X \in \textsc{SVar}_s$. Its *extension* $\bar{\rho}\colon \textsc{Pattern} \to \mathcal{P}(M)$ is defined as in Definition 4, extended with:

- $\bar{\rho}(x) = \{\rho(x)\}$ for all $x \in \textsc{EVar}_s$;
- $\bar{\rho}(X) = \rho(X)$ for all $X \in \textsc{SVar}_s$;
- $\bar{\rho}(\mu X.\, \varphi) = \mu\mathcal{F}^{\rho}_{\varphi,X}$ for all $X \in \textsc{SVar}_s$, where $\mathcal{F}^{\rho}_{\varphi,X}(A) = \overline{\rho[A/X]}(\varphi)$ for all $A \subseteq M_s$.

Here $\rho[A/X]$ is the ρ' with $\rho'(X) = A$ and $\rho'(Y) = \rho(Y)$ for all $Y \not\equiv X$. Note $\mathcal{F}^{\rho}_{\varphi,X}$ is monotone, since φ is positive in X. The notions $M \vDash \varphi$, $M \vDash \Gamma$, and $\Gamma \vDash \varphi$ are defined as expected.

Proposition 20. *For all axiom sets Γ of matching logic patterns (without μ) and all matching logic patterns φ (without μ), we have $\Gamma \vDash_{\mathsf{ML}} \varphi$ if and only if $\Gamma \vDash \varphi$.*

C. Example: capturing precisely term algebras

Many approaches to specifying formal semantics of programming languages are applications of *initial algebra semantics* [23]. In this subsection, we show how *term algebras*, a special case of initial algebras, can be *precisely captured* using MmL patterns as axioms. For simplicity, we discuss only *monosorted term algebras*, but the result can be extended to the many-sorted settings without any major technical difficulties using the techniques introduced in Section V.

Definition 21. Let $\mathbb{\Sigma} = (\{\mathit{Term}\}, \Sigma)$ be a signature with one sort *Term* and at least one constant. $\mathbb{\Sigma}$*-terms* are defined as:

$$t ::= c \in \Sigma_{\lambda,\mathit{Term}} \mid c(t_1, \ldots, t_n) \text{ for } c \in \Sigma_{\mathit{Term}\ldots\mathit{Term},\mathit{Term}}$$

The $\mathbb{\Sigma}$*-term algebra* $T^{\mathbb{\Sigma}} = (\{T^{\mathbb{\Sigma}}_{\mathit{Term}}\}, \{c_{T^{\mathbb{\Sigma}}}\}_{c \in \Sigma})$ consists of:

- a carrier set $T^{\mathbb{\Sigma}}_{\mathit{Term}}$ of all $\mathbb{\Sigma}$-terms;
- a function $c_{T^{\mathbb{\Sigma}}}\colon T^{\mathbb{\Sigma}}_{\mathit{Term}} \times \cdots \times T^{\mathbb{\Sigma}}_{\mathit{Term}} \to T^{\mathbb{\Sigma}}_{\mathit{Term}}$ for all $c \in \Sigma_{\mathit{Term}\ldots\mathit{Term},\mathit{Term}}$ defined as $c_{T^{\mathbb{\Sigma}}}(t_1, \ldots, t_n) = c(t_1, \ldots, t_n)$.

Proposition 22. *Let $\mathbb{\Sigma} = (\{\mathit{Term}\}, \Sigma)$ be a signature with one sort Term and at least one constant. Define a $\mathbb{\Sigma}$-theory $\Gamma^{\mathsf{term}}_{\mathbb{\Sigma}}$ with* ($\textsc{Function}$) *and* ($\textsc{No Confusion}$) *axioms (see Section II-C) for all symbols in Σ, plus the following axiom:*

$$(\textsc{Inductive Domain}) \quad \mu D.\, \bigvee_{c \in \Sigma} c(D, \ldots, D)$$

Then for all $\mathbb{\Sigma}$-models $M \vDash \Gamma^{\mathsf{term}}_{\mathbb{\Sigma}}$, M is isomorphic to $T^{\mathbb{\Sigma}}$. In addition, for all extended signatures $\mathbb{\Sigma}^+ \supseteq \mathbb{\Sigma}$ and $\mathbb{\Sigma}^+$-models $M \vDash \Gamma^{\mathsf{term}}_{\mathbb{\Sigma}}$, we have $M\big|_{\mathbb{\Sigma}}$ is isomorphic to $T^{\mathbb{\Sigma}}$, where $M\big|_{\mathbb{\Sigma}}$ is the reduct model *of M over the sub-signature $\mathbb{\Sigma}$.*

($\textsc{Inductive Domain}$) forces that for all models M, the carrier set M_{Term} must be the *the smallest set* that is closed under all symbols in Σ, while ($\textsc{Function}$) and ($\textsc{No Confusion}$) force all symbols in Σ to be interpreted as injective functions, and different symbols construct different terms.

Proposition 22 immediately tells us that MmL cannot have a proof system that is both sound and complete for all theories, because one can capture precisely the model $(\mathbb{N}, +, \times)$ of natural numbers with addition and multiplication with a finite number of MmL axioms, and the model $(\mathbb{N}, +, \times)$, by Gödel's first incompleteness theorem [24], is not axiomatizable.

Proposition 23. *Let $\mathbb{\Sigma} = (\{\mathit{Nat}\}, \{0 \in \Sigma_{\lambda,\mathit{Nat}}, \mathit{succ} \in \Sigma_{\mathit{Nat},\mathit{Nat}}\})$ and the $\mathbb{\Sigma}$-theory $\Gamma^{\mathsf{term}}_{\mathbb{\Sigma}}$ be defined as in Proposition 22, where the* ($\textsc{Inductive Domain}$) *takes the following form:*

$$(\textsc{Inductive Domain}) \quad \mu D \,.\, 0 \vee \mathit{succ}(D)$$

Let the signature $\mathbb{\Sigma}^{\mathbb{N}}$ extend $\mathbb{\Sigma}$ with two functions:

$$\mathit{plus}\colon \mathit{Nat} \times \mathit{Nat} \to \mathit{Nat} \qquad \mathit{mult}\colon \mathit{Nat} \times \mathit{Nat} \to \mathit{Nat}$$

and the $\mathbb{\Sigma}^{\mathbb{N}}$-theory $\Gamma^{\mathbb{N}}$ extend $\Gamma^{\mathsf{term}}_{\mathbb{\Sigma}}$ with the standard axioms:

$$\mathit{plus}(0, y) = y \qquad \mathit{plus}(\mathit{succ}(x), y) = \mathit{succ}(\mathit{plus}(x, y))$$
$$\mathit{mult}(0, y) = 0 \qquad \mathit{mult}(\mathit{succ}(x), y) = \mathit{plus}(y, \mathit{mult}(x, y))$$

Then, $\Gamma^{\mathbb{N}}$ captures precisely $(\mathbb{N}, +, \times)$, meaning that for all models $M \vDash \Gamma^{\mathbb{N}}$, M is isomorphic to $(\mathbb{N}, +, \times)$.

We finish this subsection by comparing Proposition 22 with the nontrivial result that the term algebra T^{Σ} has a *complete axiomatization* in FOL where the only predicate symbol is equality [25]. We refer to this complete FOL axiomatization as $\Gamma_{\mathsf{FOL}}(T^{\Sigma})$. This means that for all FOL formulas φ, $\Gamma_{\mathsf{FOL}}(T^{\Sigma}) \vDash_{\mathsf{FOL}} \varphi$ iff $T^{\Sigma} \vDash_{\mathsf{FOL}} \varphi$. This result is *weaker* than Proposition 22, because by Löwenheim-Skolem theorem [26], the FOL theory $\Gamma_{\mathsf{FOL}}(T^{\Sigma})$ has models of arbitrarily large cardinalities (if Σ contains non-constant constructors), meaning that there are models $M \vDash_{\mathsf{FOL}} \Gamma_{\mathsf{FOL}}(T^{\Sigma})$ with *strictly more elements* than T^{Σ}, and thus cannot be isomorphic to T^{Σ}. It is just the case that the FOL models of $\Gamma_{\mathsf{FOL}}(T^{\Sigma})$ satisfy exactly the same FOL formulas as T^{Σ}. Proposition 22, on the other hand, shows that the MmL theory $\Gamma^{\mathsf{term}}_{\Sigma}$ captures T^{Σ} *up to isomorphism*. Many automatic reasoning approaches [27], [28] for algebraic datatypes and co-datatypes exploit this complete axiomatization $\Gamma_{\mathsf{FOL}}(T^{\Sigma})$. These approaches can be generalized to MmL settings and provide (semi-)decision procedures for the corresponding MmL theories. We leave this as future work.

D. Matching μ-logic proof system

Proposition 23 implies that MmL cannot have a sound and complete proof system. The best we can do then is to aim for a proof system that is *good enough in practice*. We take the ML proof system $\mathcal{H}$ and extend it with three additional proof rules (see Fig. 1). Rules (Pre-Fixpoint) and (Knaster-Tarski) are standard proof rules about least fixpoints as in modal μ-logic [8]; sometimes (Knaster-Tarski) is referred to as *Park induction* [29]–[31]. Rule (Set Variable Substitution) allows us to prove from $\vdash \varphi$ any substitution $\vdash \varphi[\psi/X]$ for $X \in \mathrm{SVar}$. That X is a set variable is crucial. In general, we *cannot* prove from $\vdash \varphi$ that $\vdash \varphi[\psi/x]$ for $x \in \mathrm{EVar}$, because it does not hold semantically. As shown in [1], it only holds when ψ is *functional*, that is, when ψ evaluates to a singleton set. Indeed, suppose that ψ is not functional, say it is the pattern $0 \vee succ(0)$ over the signature of natural numbers in Proposition 23, which evaluates to a set of two elements. Then we can pick φ to be the tautology $\exists y \,.\, x = y$, and then $\varphi[\psi/x]$ becomes $\exists y \,.\, \psi = y$, which states that ψ evaluates to a singleton set (the valuation of y), which is a contradiction.

We let $\mathcal{H}_{\mu}$ denote the extended proof system in Fig. 1, and from here on we write $\Gamma \vdash \varphi$ instead of $\Gamma \vdash_{\mathcal{H}_{\mu}} \varphi$.

Theorem 24 (Soundness of $\mathcal{H}_{\mu}$)**.** *$\Gamma \vdash \varphi$ implies $\Gamma \vDash \varphi$.*

E. Instance: Peano arithmetic

We illustrate the power of (Pre-Fixpoint) and (Knaster-Tarski) by showing that they derive the (Induction) schema in the FOL axiomatization of Peano arithmetic [32], [33]:

$$(\textsc{Induction}) \quad \varphi(0) \wedge \forall x.(\varphi(x) \to \varphi(succ(x))) \to \forall x.\varphi(x)$$

where $\varphi(x)$ is a FOL formula with a distinguished variable x.

We encode the FOL syntax of Peano arithmetic following the technique in Section II-D, that is, we define a signature $\Sigma^{\mathsf{Peano}} = (\{Nat, Pred\}, \Sigma^{\mathbb{N}})$ where $\Sigma^{\mathbb{N}}$ is defined in Proposition 23 that contains the functions $0, succ, plus, mult$, and let Γ^{Peano} contain the same equation axioms as $\Gamma^{\mathbb{N}}$. The Σ^{Peano}-patterns of sort $Pred$ are those built from equalities between two patterns of sort Nat, as well as connectives and quantifiers.

Proposition 25. *Under the above notations, we have:*

$$\Gamma^{\mathsf{Peano}} \vdash \varphi(0) \wedge \forall x.(\varphi(x) \to \varphi(succ(x))) \to \forall x.\varphi(x).$$

V. Defining Recursive Symbols as Syntactic Sugar

Intuitively, the least fixpoint pattern $\mu X.\varphi$ specifies *a recursive set* that satisfies the equation $X = \varphi$, where φ may contain recursive occurrences of X. For example, the pattern $\mu X.\, 3 \vee plus(X, X)$ specifies the set of all nonzero multiples of 3, which intuitively defines a *recursive constant*:

$$m3 \in \Sigma_{\lambda,Nat} \qquad\qquad m3 =_{\mathrm{lfp}} 3 \vee plus(m3, m3).$$

Here, "$=_{\mathrm{lfp}}$" is merely a notation, meaning that we want $m3$ to be *the least set* that satisfies the equation. Note that the total set of all natural numbers is a trivial solution.

The challenge is how to generalize the above and define *recursive non-constant symbols*. For example, suppose we want to define a unary symbol $collatz \in \Sigma_{Nat,Nat}$ as follows:

$$collatz(n) =_{\mathrm{lfp}}$$
$$n \vee (even(n) \wedge collatz(n/2)) \vee (odd(n) \wedge collatz(3n+1))$$

with the intuition that $collatz(n)$ gives the set of all numbers in the Collatz sequence[1] starting from n. However, the μ-binder in MmL can only be applied on *set variables*, not on *symbols*, so the following attempt is syntactically wrong:

$$collatz(n) = \mu\, \sigma(n)\,. \qquad // \ \mu \text{ can only bind a set variable}$$
$$n \vee (even(n) \wedge \sigma(n/2)) \vee (odd(n) \wedge \sigma(3n+1))$$

One possible solution could be to *extend* MmL with the above syntax and allow the μ-binder to quantify *symbol variables*, not only *set variables*. The semantics and proof system could be extended accordingly. This is exactly how *first-order logic with least fixpoints* extends FOL [7]. But do we really have to? After all, our proof rules (Pre-Fixpoint) and (Knaster-Tarski) in Fig. 1 are nothing but a logical incarnation of the Knaster-Tarski theorem, which has been repeatedly demonstrated to serve as a solid if not the main foundation for recursion. Therefore, we conjecture that the $\mathcal{H}$ proof system in Fig. 1 is sufficient in practice, and thus would rather resist extending MmL. That is, we conjecture that it should be possible to *define* one's desired approach to recursion/induction/fixpoints using ordinary MmL *theories*; as an analogy, in Section II-C we showed how we can define definedness, totality, equality, membership, set containment, functions, partial functions, constructors, etc. (see [1] for more) as theories, without a need to extend ML.

In particular, we can solve the above recursive symbol challenge by using the *principle of currying-uncurrying* to "mimic" the unary symbol $collatz \in \Sigma_{Nat,Nat}$ with a set variable $collatz : Nat \otimes Nat$, where $Nat \otimes Nat$ is the *product sort* (defined later; the intuition is that $Nat \otimes Nat$ has the product set $\mathbb{N} \times \mathbb{N}$ as its carrier set), and thus reducing the challenge of defining a least relation in $[\mathbb{N} \to \mathcal{P}(\mathbb{N})]$ to defining a least subset of $\mathcal{P}(\mathbb{N} \times \mathbb{N})$, which can be done with the MmL μ-binder.

[1] A Collatz sequence starting from $n \geq 1$ is obtained by repeating the following procedure: if n is even then return $n/2$; otherwise, return $3n + 1$.

A. Principle of currying-uncurrying and product sorts

The principle of currying-uncurrying [34], [35] is used in various settings (e.g., simply-typed lambda calculus [36]) as a means to reduce the study of multi-argument functions to the simpler single-argument functions. We here present the principle in its adapted form that fits best with our needs.

Proposition 26. *Let $M_{s_1}, \ldots, M_{s_n}, M_s$ be nonempty sets. The* principle of currying-uncurring *means the isomorphism*

$$\mathcal{P}(M_{s_1} \times \cdots \times M_{s_n} \times M_s) \underset{uncurry}{\overset{curry}{\rightleftarrows}} [M_{s_1} \times \cdots \times M_{s_n} \to \mathcal{P}(M_s)]$$

defined for all $a_1 \in M_{s_1}, \ldots, a_n \in M_{s_n}, b \in M_s, \alpha \subseteq M_{s_1} \times \cdots \times M_{s_n} \times M_s$, and $f \colon M_{s_1} \times \cdots \times M_{s_n} \to \mathcal{P}(M_s)$ as:

$$curry(\alpha)(a_1, \ldots, a_n) = \{b \in M_s \mid (a_1, \ldots, a_n, b) \in \alpha\}$$
$$uncurry(f) = \{(a_1, \ldots, a_n, b) \mid b \in f(a_1, \ldots, a_n)\}.$$

The tuple set $uncurry(f)$ is also called the graph of f.

In other words, we can mimic an n-ary symbol $\sigma \in \Sigma_{s_1 \ldots s_n, s}$ with a set variable of the *product sort* $s_1 \otimes \cdots \otimes s_n \otimes s$, whose (intended) carrier set is exactly the product set $M_{s_1} \times \ldots M_{s_n} \times M_s$. This inspires the following definition.

Definition 27. Let s, s' be two sorts, not necessarily distinct. The *product sort $s \otimes s'$* is a sort that is different from s and s'. *Pairing* $\langle \_, \_ \rangle_{s,s'} \colon s \times s' \to s \otimes s'$ is a function and *projection* $\_(\_)_{s,s'} \colon s \otimes s' \times s \rightharpoonup s'$ is a partial function, and we drop sorts s, s' for simplicity. Define three axioms:

(Injectivity) $\langle k_1, v_1 \rangle = \langle k_2, v_2 \rangle \to (k_1 = k_2) \wedge (v_1 = v_2)$

(Key-Value) $\langle k_1, v \rangle(k_2) = (k_1 = k_2) \wedge v$

(Product) $\exists k \exists v. \langle k, v \rangle$

that force the carrier set of $s \otimes t$ to be the product of the ones of s and t and pairing/projection to be interpreted as expected. Note that we assume definedness symbols/axioms because we have used the function and partial function notations as well as equality in the axioms.

The product of multiple sorts and the associated pairing/projection operations can be defined as derived constructs as follows. Given (not necessarily distinct) sorts $s_1, \ldots, s_n, s$ and patterns $\varphi_1, \ldots, \varphi_n, \varphi, \psi$ of appropriate sorts, we define:

$$s_1 \otimes \cdots \otimes s_n \otimes s \equiv s_1 \otimes (s_2 \otimes (\cdots \otimes (s_n \otimes s) \ldots))$$
$$\langle \varphi_1, \ldots, \varphi_n, \varphi \rangle \equiv \langle \varphi_1, \langle \ldots, \langle \varphi_n, \varphi \rangle \ldots \rangle \rangle$$
$$\psi(\varphi_1, \ldots, \varphi_n) \equiv \psi(\varphi_1) \ldots (\varphi_n).$$

Note that we tacitly use the same syntax $\_(\_, \ldots, \_)$ for both symbol applications and projections to blur their distinction. In particular, if $\sigma : s_1 \otimes \cdots \otimes s_n \otimes s$ is a set variable of the product sort, then $\sigma(\varphi_1, \ldots, \varphi_n)$ is a well-formed pattern of sort s iff $\varphi_1, \ldots, \varphi_n$ have the appropriate sorts $s_1, \ldots, s_n$.

B. Defining recursive symbols in matching μ-logic

Definition 28. Let $\mathbb{\Sigma} = (S, \Sigma)$ be a signature and $\sigma \in \Sigma_{s_1 \ldots s_n, s}$, containing the product sorts and pairing/projection symbols. We use the notation $\sigma(x_1, \ldots, x_n) =_{\mathsf{lfp}} \varphi$ to mean the axiom:

$$\sigma(x_1, \ldots, x_n) =$$
$$(\mu \sigma{:}s_1 \otimes \cdots \otimes s_n \otimes s. \exists x_1 \ldots \exists x_n. \langle x_1, \ldots, x_n, \varphi \rangle)(x_1, \ldots, x_n)$$

where $\exists x_1 \ldots \exists x_n. \langle x_1, \ldots, x_n, \varphi \rangle$ captures the *graph of φ as a function w.r.t. $x_1, \ldots, x_n$*. Note that in the axiom, all occurrences of $\sigma \in \Sigma_{s_1 \ldots s_n, s}$ in φ are tacitly regarded as the set variable $\sigma{:}s_1 \otimes \cdots \otimes s_n \otimes s$, which are then bound by μ-binder. A symbol $\sigma \in \Sigma_{s_1 \ldots s_n, s}$ obeying this axiom is called *recursive*.

Recursive symbols can be used to define various (co)inductive data structures and relations. In Section VI, we will see how first-order logic with least fixpoints (LFP) can be captured as notations using recursive symbols. In [14], it is shown how recursive definitions in separation logic, such as lists and trees, can also be defined by recursive symbols. However, Definition 28 is not ideally convenient when it comes to *reasoning* about recursive symbols because it is complex and contains many details about the product sorts. Instead, we want to reason about recursive symbols in a similar way to how we reason about the basic least fixpoint patterns $\mu X. \varphi$, using a generalized form of (Pre-Fixpoint) and (Knaster-Tarski). This is achieved by the following theorem.

Theorem 29. *Let $\sigma \in \Sigma_{s_1 \ldots s_n, s}$ be a recursive symbol defined as $\sigma(x_1, \ldots, x_n) =_{\mathsf{lfp}} \varphi$, Γ be a theory, ψ be a pattern, and*

$$\Gamma \vdash (\exists z_1 \ldots \exists z_n. z_1 \in \varphi_1 \wedge \cdots \wedge z_n \in \varphi_n \wedge \psi[z_1/x_1, \ldots, z_n/x_n]) \to \psi[\varphi_1/x_1, \ldots, \varphi_n/x_n] \quad \textit{for all } \varphi_1, \ldots, \varphi_n \qquad (\dagger)$$

Then the following hold:

- *Pre-Fixpoint: $\Gamma \vdash \varphi \to \sigma(x_1, \ldots, x_n)$;*
- *Knaster-Tarski: $\Gamma \vdash \varphi[\psi/\sigma] \to \psi$ implies $\Gamma \vdash \sigma(x_1, \ldots, x_n) \to \psi$, where $\varphi[\psi/\sigma]$ is the result of substituting all patterns of the form $\sigma(\varphi_1, \ldots, \varphi_n)$ in φ with $\psi[\varphi_1/x_1, \ldots, \varphi_n/x_n]$.*

Condition (†) is a logic incarnation of the property of propagation (Proposition 3) of ψ as a function w.r.t. $x_1, \ldots, x_n$, which requires, intuitively, that ψ "behaves like a symbol".

VI. Instance: First-Order Logic with Least Fixpoints

First-order logic with least fixpoints (LFP) [7] extends the syntax of first-order logic formulas with:

$$\varphi ::= [\mathsf{lfp}_{R, x_1, \ldots, x_n} \varphi](t_1, \ldots, t_n)$$

where R is a *predicate variable* and φ is a formula that is positive in R. Intuitively, "$[\mathsf{lfp}_{R, x_1, \ldots, x_n} \varphi]$" behaves as the least fixpoint predicate of the operation that maps R to φ. Due to its complexity and our limited space, we skip the formal definition of the semantics and simply denote the validity relation in LFP as $\vDash_{\mathsf{LFP}} \varphi$. A comprehensive study on LFP can be found in [37]. As an example, the following LFP formula holds iff x is a nonzero multiple of 3:

$$[\mathsf{lfp}_{R,z} z = 3 \vee \exists z_1 \exists z_2. R(z_1) \wedge R(z_2) \wedge z = plus(z_1, z_2)](x)$$

Given the notations of recursive symbols defined in Section V, it is straightforward to subsume LFP by extending the theory Γ^{FOL} defined in Section II-D with product sorts and pairing/projection symbols, and the syntactic sugar:

$$[\mathsf{lfp}_{R, x_1, \ldots, x_n} \varphi](t_1, \ldots, t_n) \equiv$$
$$(\mu R \colon s_1 \otimes \ldots \otimes s_n \otimes Pred. \exists x_1 \ldots \exists x_n. \langle x_1, \ldots, x_n, \varphi \rangle)(t_1, \ldots, t_n)$$

for all predicate variables R with argument sorts $s_1, \ldots, s_n$. A minor difference here is that we add one additional axiom, $\forall x{:}Pred \forall y{:}Pred.x = y$, to constrain that the carrier set of sort *Pred* is a singleton set so that all MmL models can be regarded as FOL/LFP models. This fact is used to prove the "only if" part in the next theorem.[2] We denote the resulting theory Γ^{LFP}.

Theorem 30. *If φ is an LFP formula, then $\vDash_{\mathsf{LFP}} \varphi$ iff $\Gamma^{\mathsf{LFP}} \vDash \varphi$.*

VII. Instances: Modal μ-Calculus and Temporal Logics

We have seen how MmL symbols and patterns can be used to specify both structure and constraints, such as terms (Section IV-C) and FOL (Section II-D), as well as various induction, recursion and least fixpoints schemas (Sections IV-E and V) over these. These suffice to express and prove program assertions, including complex state abstractions (see also how separation logic falls as a fragment of ML in [1]), in contexts where MmL is chosen as a static state assertion formalism in program verification frameworks based on Hoare logic [38], dynamic logic [11], or reachability logic [2]. However, as explained in Section I, our ultimate goal is to support not only static state assertions, but any program properties, including ones that are usually specified using Hoare, dynamic, or reachability logics. We start the discussion in this section by showing how MmL symbols and patterns can also be used to specify *dynamic transition relations*, which are often captured by modalities in modal μ-logic and dynamic logic; in Section VIII we then discuss how MmL also subsumes reachability logic, which subsumes Hoare logic [6].

A. Modal μ-logic syntax, semantics, and proof system

The *syntax* of modal μ-logic [8] is parametric in a countably infinite set PVar of propositional variables. Modal μ-logic *formulas* are given by the grammar[3]:

$$\varphi ::= p \in \textsc{PVar} \mid \varphi \wedge \varphi \mid \neg\varphi \mid \circ\varphi \mid \mu X.\,\varphi \text{ if } \varphi \text{ is positive in } X$$

where $p, X \in$ PVar are propositional variables. As a convention, p is used for free variables while X is used for bound ones. Derived constructs are defined as usual, e.g., $\bullet\varphi \equiv \neg\circ\neg\varphi$. Modal μ-logic semantics is given using *transition systems* $\mathbb{S} = (S, R)$, with S a nonempty set of *states* and $R \subseteq S \times S$ a *transition relation*, and valuations $V \colon \textsc{PVar} \to \mathcal{P}(S)$, as follows:

- $[\![p]\!]^{\mathbb{S}}_V = V(p)$;
- $[\![\varphi_1 \wedge \varphi_2]\!]^{\mathbb{S}}_V = [\![\varphi_1]\!]^{\mathbb{S}}_V \cap [\![\varphi_2]\!]^{\mathbb{S}}_V$;
- $[\![\neg\varphi]\!]^{\mathbb{S}}_V = S \setminus [\![\varphi]\!]^{\mathbb{S}}_V$;
- $[\![\circ\varphi]\!]^{\mathbb{S}}_V = \{s \in S \mid s\,R\,t \text{ implies } t \in [\![\varphi]\!]^{\mathbb{S}}_V \text{ for all } t \in S\}$;
- $[\![\mu X.\,\varphi]\!]^{\mathbb{S}}_V = \bigcap\{A \subseteq S \mid [\![\varphi]\!]^{\mathbb{S}}_{V[A/X]} \subseteq A\}$;

A modal μ-logic formula φ is *valid*, denoted $\vDash_\mu \varphi$, if for all transition systems $\mathbb{S}$ and all valuations V, we have $[\![\varphi]\!]^{\mathbb{S}}_V = S$.

A proof system of modal μ-logic is firstly given in [8] and then proved to be complete in [39]. It extends the proof system of propositional logic with the following proof rules:

$$\text{(K)} \quad \circ(\varphi_1 \to \varphi_2) \to (\circ\varphi_1 \to \circ\varphi_2) \qquad \text{(N)} \quad \frac{\varphi}{\circ\varphi}$$

$$(\mu_1) \quad \varphi[\mu X.\varphi/X] \to \mu X.\,\varphi \qquad (\mu_2) \quad \frac{\varphi[\psi/X] \to \psi}{\mu X.\,\varphi \to \psi}$$

We denote the corresponding provability relation as $\vdash_\mu \varphi$. Notice that (K) and (N) are provable in MmL (Proposition 12), and (μ_1) and (μ_2) are our (Pre-Fixpoint) and (Knaster-Tarski). This means that we can easily mimic all modal μ-logic proofs in MmL (i.e. "(2) ⇒ (3)" in Theorem 31).

B. Defining modal μ-logic in matching μ-logic

To subsume the syntax of modal μ-logic, we define a signature (of *transition systems*) $\Sigma^{\mathsf{TS}} = (\{State\}, \{\bullet \in \Sigma^{\mathsf{TS}}_{State,State}\})$ where symbol "$\bullet$" is called *one-path next*. We regard propositional variables in PVar as MmL set variables. We write $\bullet\varphi$ instead of $\bullet(\varphi)$, and define $\circ\varphi \equiv \neg\bullet\neg\varphi$. Then all modal μ-logic formulas φ are MmL Σ^{TS}-patterns of sort *State*. Finally, note that no axioms are needed; let Γ^μ be the empty Σ^{TS}-theory.

An important observation is that the Σ^{TS}-models are *exactly* the transition systems, where $\bullet \in \Sigma^{\mathsf{TS}}_{State,State}$ is interpreted as the transition relation R. Specifically, for any transition system $\mathbb{S} = (S, R)$, we can regard $\mathbb{S}$ as a Σ^{TS}-model where S is the carrier set of *State* and $\bullet_{\mathbb{S}}(t) = \{s \in S \mid s\,R\,t\}$ contains all *R-predecessors* of t. This might seem counter-intuitive at first glance: why "one-path next" is interpreted as the predecessors instead of the successors of R? See the following illustration:

$$\begin{array}{cccccccl} \cdots & s & \xrightarrow{R} & s' & \xrightarrow{R} & s'' & \cdots & \text{// states} \\ & \bullet\bullet\varphi & & \bullet\varphi & & \varphi & & \text{// patterns} \end{array}$$

In other words, $\bullet\varphi$ is matched by states that *have at least one next state* that satisfies φ, conforming to the intuition. Another interesting observation is about $\bullet\varphi$ and its dual, $\circ\varphi \equiv \neg\bullet\neg\varphi$, called *all-path next*. The difference is that $\circ\varphi$ is matched by s if *for all* states t such that $s\,R\,t$, we have t matches φ. In particular, if s has no successor, then s matches $\circ\varphi$ for any φ. This is formally summarized in Proposition 32.

We now feel free to take any transition system $\mathbb{S}$ as an MmL Σ^{TS}-model. The following *conservative extension theorem* shows that our definition of modal μ-logic in MmL is *faithful*, both syntactically and semantically. What is insightful about the theorem is its *proof*, which can be applied to other logics discussed in this paper to obtain similar results.

Theorem 31. *The following properties are equivalent for all modal μ-logic formulas φ: (1) $\vDash_\mu \varphi$; (2) $\vdash_\mu \varphi$; (3) $\Gamma^\mu \vdash \varphi$; (4) $\Gamma^\mu \vDash \varphi$; (5) $M \vDash \varphi$ for all Σ^{TS}-models M such that $M \vDash \Gamma^\mu$; (6) $\mathbb{S} \vDash_\mu \varphi$ for all transition systems $\mathbb{S}$.*

Proof sketch: We only need to prove "(2) ⇒ (3)" and "(5) ⇒ (6)", as the rest are already proved/known. "(1) ⇒ (2)" follows by the completeness of modal μ-logic, which is nontrivial but known [39]. "(2) ⇒ (3)" follows by proving all modal μ-logic proof rules as theorems in MmL (Proposition 12). "(3) ⇒ (4)" follows by the soundness of MmL (Theorem 24). "(4) ⇒ (5)" follows by Definition 19. "(5) ⇒

[2] We do not need that axiom in defining FOL in ML, as seen in Section II-D, because there the "if" part is proved via a proof theoretical approach, using the completeness proof system of FOL and the fact that we can mimic FOL proofs in ML (see [1]). Since LFP does not have a complete proof system, we have to add additional axioms to further constrain on the MmL models.

[3] The modal μ-logic literature often uses $\Box\varphi$ and $\Diamond\varphi$ instead of $\circ\varphi$ and $\bullet\varphi$. We here use the latter to avoid confusion with the "always" $\Box\varphi$ and "eventually" $\Diamond\varphi$ in LTL and CTL.

(6)" follows by proving its contrapositive statement, "$\not\vDash_\mu \varphi$ implies $\Gamma^\mu \not\vDash \varphi$", by taking a transition system $\mathbb{S} = (S, R)$ and a valuation V such that $[\![\varphi]\!]_V^{\mathbb{S}} \neq S$, and showing that if we regard $\mathbb{S}$ as a Σ^{TS}-model and V as an $\mathbb{S}$-valuation in MmL, then $\mathbb{S} \vDash \Gamma^\mu$ and $\overline{V}(\varphi) \neq S$, which means that $\Gamma^\mu \not\vDash \varphi$. Finally, "(6) ⇒ (1)" follows by definition. ■

Therefore, modal μ-logic can be regarded as an empty theory in a vanilla MmL without quantifiers, over a signature containing only one sort and only one symbol, which is unary. It is worth mentioning that variants of modal μ-logic with more modal modalities have been proposed (see [40] for a survey). At our knowledge, however, all such variants consider only unary modal modalities and they are only required to obey the usual (K) and (N) proof rules of modal logic. In contrast, MmL allows polyadic symbols while still obeying the desired (K) and (N) rules (see Proposition 12), allows arbitrary further constraining axioms in MmL theories, and also quantification over element variables and many-sorted universes.

C. Studying transition systems in MmL

The above suggests that MmL may offer a unifying playground to specify and reason about transition systems, by means of Σ^{TS}-theories/models. We can define various temporal/dynamic operations and modalities as *derived constructs* from the basic "one-path next" symbol "$\bullet$" and the μ-binder, without the need to extend the syntax and semantics of the logic. We can constrain the models/transition systems of interest using *additional axioms*, without the need to modify/extend the proof system of the logic. In what follows, we show that by defining proper constructs as syntactic sugar and adding proper axioms, we can capture *faithfully* LTL (both finite- and infinite-trace), CTL, dynamic logic (DL), and reachability logic (RL).

Let us add more temporal modalities as derived constructs (we have seen "all-path next" $\circ\varphi$ in Section VII-B):

$$\text{"eventually" } \Diamond\varphi \equiv \mu X.\, \varphi \vee \bullet X$$

$$\text{"always" } \Box\varphi \equiv \nu X.\, \varphi \wedge \circ X$$

$$\text{"(strong) until" } \varphi_1 \, U \, \varphi_2 \equiv \mu X.\, \varphi_2 \vee (\varphi_1 \wedge \bullet X)$$

$$\text{"well-founded" } \mathsf{WF} \equiv \mu X. \circ X \quad \text{// no infinite paths}$$

Proposition 32. *Let $\mathbb{S} = (S, R)$ be a transition system regarded as a Σ^{TS}-model, and let ρ be any valuation and $s \in S$. Then:*

- *$s \in \bar{\rho}(\bullet\varphi)$ if there exists $t \in S$ such that $s \, R \, t$, $t \in \bar{\rho}(\varphi)$; in particular, $s \in \bar{\rho}(\bullet\top)$ if s has an R-successor;*
- *$s \in \bar{\rho}(\circ\varphi)$ if for all $t \in S$ such that $s \, R \, t$, $t \in \bar{\rho}(\varphi)$; in particular, $s \in \bar{\rho}(\circ\bot)$ if s has no R-successor;*
- *$s \in \bar{\rho}(\Diamond\varphi)$ if there exists $t \in S$ such that $s \, R^* \, t$, $t \in \bar{\rho}(\varphi)$;*
- *$s \in \bar{\rho}(\Box\varphi)$ if for all $t \in S$ such that $s \, R^* \, t$, $t \in \bar{\rho}(\varphi)$;*
- *$s \in \bar{\rho}(\varphi_1 \, U \, \varphi_2)$ if there exists $n \geq 0$ and $t_1, \ldots, t_n \in S$ such that $s R t_1 R \cdots R t_n$, $t_n \in \bar{\rho}(\varphi_2)$, and $s, t_1, \ldots, t_{n-1} \in \bar{\rho}(\varphi_1)$;*
- *$s \in \bar{\rho}(\mathsf{WF})$ if s is R-well-founded, meaning that there is no infinite sequence $t_1, t_2, \cdots \in S$ with $s \, R \, t_1 \, R \, t_2 \, R \ldots$;*

where $R^* = \bigcup_{i \geq 0} R^i$ is the reflexive transitive closure of R.

D. Instances: temporal logics

Since MmL can define modal μ-logic (as an empty theory over a unary symbol), it is not surprising that it can also define various temporal logics such as LTL and CTL as theories whose axioms constrain the underlying transition relations. What is interesting, in our view, is that the resulting theories are simple, intuitive, and faithfully capture both the syntax (provability) and the semantics of these temporal logics.

1) Instance: infinite-trace LTL: The LTL syntax, namely

$$\varphi ::= p \in \textsc{PVar} \mid \varphi \wedge \varphi \mid \neg\varphi \mid \circ\varphi \mid \varphi \, U \, \varphi$$

is already subsumed in MmL with the derived constructs we give in Section VII-C. Other common LTL modalities such as "always $\Box\varphi$" are defined from the "until U" modality in the usual way. Infinite-trace LTL takes as models transition systems whose transition relations are *linear* and *infinite into the future*. We assume readers are familiar with the semantics and proof system of infinite-trace LTL (see [10], e.g.) and skip their formal definitions. We use "$\vDash_{\mathsf{infLTL}}$" and "$\vdash_{\mathsf{infLTL}}$" to denote infinite-trace LTL validity and provability, respectively.

To capture the characteristics of both "*infinite future*" and "*linear future*", we add the following two patterns as axioms:

$$(\textsc{Inf}) \quad \bullet\top \qquad\qquad (\textsc{Lin}) \quad \bullet X \to \circ X$$

and denote the resulting Σ^{TS}-theory as Γ^{infLTL}. Note that by (Set Variable Substitution), we can prove from axiom (Lin) that $\bullet\varphi \to \circ\varphi$ for all patterns φ. Intuitively, (Inf) forces all states s to have at least one successor, and thus all traces can be extended to an infinite trace, and (Lin) forces all states s to have only a *linear future*. The following theorem shows that our definition of infinite-trace LTL is faithful both syntactically and semantically, proved exactly as Theorem 31.

Theorem 33. *The following properties are equivalent for all infinite-trace LTL formulas φ: (1) $\vdash_{\mathsf{infLTL}} \varphi$; (2) $\vDash_{\mathsf{infLTL}} \varphi$; (3) $\Gamma^{\mathsf{infLTL}} \vdash \varphi$; (4) $\Gamma^{\mathsf{infLTL}} \vDash \varphi$.*

Therefore, infinite-trace LTL can be regarded as a theory containing two axioms, (Inf) and (Lin), over the same signature as the theory corresponding to modal μ-logic.

2) Instance: finite-trace LTL: Finite execution traces play an important role in program verification and monitoring. Finite-trace LTL considers models that are *linear* but have only *finite future*. The following *syntax* of finite-trace LTL:

$$\varphi ::= p \in \textsc{PVar} \mid \varphi \wedge \varphi \mid \neg\varphi \mid \circ\varphi \mid \varphi \, U_w \, \varphi$$

differs from infinite-trace LTL in that the "until U_w" is *weak until*, meaning that $\varphi_1 \, U_w \, \varphi_2$ does not force that φ_2 holds eventually. Again, we assume readers are familiar with the semantics and proof system of finite-trace LTL (if not, see [10]) and use "$\vDash_{\mathsf{finLTL}}$" and "$\vdash_{\mathsf{finLTL}}$" to denote its validity and provability, respectively.

To subsume the above syntax, we define in MmL:

$$\text{"weak until" } \varphi_1 \, U_w \, \varphi_2 \equiv \mu X. \varphi_2 \vee (\varphi_1 \wedge \circ X).$$

To capture the characteristics of both *finite future* and *linear future*, we add the following two patterns as axioms:

$$(\textsc{Fin}) \quad \mathsf{WF} \equiv \mu X. \circ X \qquad\qquad (\textsc{Lin}) \quad \bullet X \to \circ X$$

and call the resulting Σ^{TS}-theory Γ^{finLTL}. Intuitively, (Fin) forces all states to be well-founded, meaning that there is no infinite execution trace in the underlying transition systems.

Theorem 34. *The following properties are equivalent for all finite-trace LTL formula φ: (1) $\vdash_{\mathsf{finLTL}} \varphi$; (2) $\vDash_{\mathsf{finLTL}} \varphi$; (3) $\Gamma^{\mathsf{finLTL}} \vdash \varphi$; (4) $\Gamma^{\mathsf{finLTL}} \vDash \varphi$.*

Therefore, finite-trace LTL can be regarded as a theory containing two axioms, (Fin) and (Lin), over the same signature as the theory corresponding to modal μ-logic.

3) Instance: CTL: CTL models are transition systems that are *infinite into the future* and allow states to have a *branching future* (rather than linear). The following syntax of CTL:

$$\varphi ::= p \in \text{PVar} \mid \varphi \wedge \varphi \mid \neg\varphi \mid \mathsf{AX}\varphi \mid \mathsf{EX}\varphi \mid \varphi\, \mathsf{AU}\, \varphi \mid \varphi\, \mathsf{EU}\, \varphi$$

is extended with the following derived constructs:

$$\mathsf{EF}\varphi \equiv \textit{true}\ \mathsf{EU}\ \varphi \qquad \mathsf{AG}\varphi \equiv \neg\mathsf{EF}\neg\varphi$$
$$\mathsf{AF}\varphi \equiv \textit{true}\ \mathsf{AU}\ \varphi \qquad \mathsf{EG}\varphi \equiv \neg\mathsf{AF}\neg\varphi$$

The names of the CTL modalities suggest their meaning: the first letter means either "on all paths" (A) or "on one path" (E), and the second letter means "next" (X), "until" (U), "always" (G), or "eventually" (F). For example, "AX" is "all-path next", "EU" is "one-path until", etc. We refer readers to [41] for CTL definitions, semantics and proof system. Here we denote its validity and provability as "$\vDash_{\mathsf{CTL}}$" and "$\vdash_{\mathsf{CTL}}$", respectively.

To define CTL as an MmL theory, we add only the axiom (Inf) for infinite future and use the following syntactic sugar:

$$\mathsf{AX}\varphi \equiv \circ\varphi \qquad \varphi_1\ \mathsf{AU}\ \varphi_2 \equiv \mu X.\, \varphi_2 \vee (\varphi_1 \wedge \circ X)$$
$$\mathsf{EX}\varphi \equiv \bullet\varphi \qquad \varphi_1\ \mathsf{EU}\ \varphi_2 \equiv \mu X.\, \varphi_2 \vee (\varphi_1 \wedge \bullet X)$$

The resulting Σ^{TS}-theory is denoted as Γ^{CTL}.

Theorem 35. *For all CTL formulas φ, the following are equivalent: (1) $\vdash_{CTL} \varphi$; (2) $\vDash_{CTL} \varphi$; (3) $\Gamma^{CTL} \vdash \varphi$; (4) $\Gamma^{CTL} \vDash \varphi$.*

Therefore, CTL can be regarded as a theory over the same signature as the theory corresponding to modal μ-logic, but containing one axiom, (Inf). It may be insightful to look at all three temporal logics discussed in this section through the lenses of MmL, as theories over a unary symbol signature: modal μ-logic is the empty and thus the least constrained theory; CTL comes immediately next with only one axiom, (Inf), to enforce infinite traces; infinite-trace LTL further constrains with the linearity axiom (Lin); finally, finite-trace LTL replaces (Inf) with (Fin). We believe that MmL can serve as a convenient and uniform framework to define and study temporal logics. For example, finite-trace CTL can be trivially obtained as the theory containing only the axiom (Fin), LTL with both finite and infinite traces is the theory containing only the axiom (Lin), and CTL with unrestricted (finite or infinite branch) models is the empty theory (i.e., modal μ-logic).

E. Instance: dynamic logic

Dynamic logic (DL) [11]–[13], [42] is a common logic used for program reasoning. The DL syntax is parametric in a set PVar of *propositional variables* and a set APgm of *atomic programs*, each belonging to a different formula syntactic category:

$$\varphi ::= p \in \text{PVar} \mid \varphi \to \varphi \mid \textit{false} \mid [\alpha]\varphi$$
$$\alpha ::= a \in \text{APgm} \mid \alpha \,;\, \alpha \mid \alpha \cup \alpha \mid \alpha^* \mid \varphi?$$

The first line defines *propositional formulas*. The second line defines *program formulas*, which represent programs built from atomic ones with the primitive regular expression constructs. Define $\langle\alpha\rangle\varphi \equiv \neg[\alpha](\neg\varphi)$. Intuitively, $[\alpha]\varphi$ holds if all executions of α lead to φ, while $\langle\alpha\rangle\varphi$ holds if there is one execution of α that leads to φ. Common program constructs such as if-then-else, while-do, etc., can be defined as derived constructs using the four primitive ones; see [11]–[13]. We let "$\vDash_{\mathsf{DL}}$" and "$\vdash_{\mathsf{DL}}$" denote the validity and provability of DL.

It is known that DL can be embedded in the variant of modal μ-logic with multiple modalities (see, e.g., [40]). The idea is to define a modality $[a]$ for every atomic program $a \in \text{APgm}$, and then to define the four program constructs as least/greatest fixpoints. We can easily adopt the same approach and associate an empty MmL theory to DL, over a signature containing as many unary symbols as atomic programs. However, MmL allows us to propose a better embedding, unrestricted by the limitations of modal μ-logic. Indeed, the embedding in [40] suffers from at least two limitations that we can avoid with MmL. First, sometimes transitions are not just labeled with discrete programs, such as in *hybrid systems* [43] and *cyber-physical systems* [44] where the labels are *continuous values* such as elapsing time. We cannot introduce for every time $t \in \mathbb{R}_{\geq 0}$ a modality $[t]$, as only countably many modalities are allowed. Instead, we may want to axiomatize the domains of such possibly continuous values and treat them as any other data. Second, we may want to quantify over such values, be they discrete or continuous, and we would not be able to do so (even in MmL) if they are encoded as modalities/symbols.

Let us instead define a signature (of *labeled transition systems*) $\Sigma^{\mathsf{LTS}} = (\{\mathit{State}, \mathit{Pgm}\}, \Sigma^{\mathsf{LTS}}_{\lambda,\mathit{Pgm}} \cup \{\bullet \in \Sigma^{\mathsf{LTS}}_{\mathit{Pgm}\,\mathit{State},\mathit{State}}\})$ where the "one-path next $\bullet$" is a *binary symbol* taking an additional *Pgm* argument, and for all atomic programs $a \in \text{APgm}$ we add a constant symbol $a \in \Sigma^{\mathsf{LTS}}_{\lambda,\mathit{Pgm}}$. Just as all Σ^{TS}-models are exactly transition systems (Section VII-B), we have that all Σ^{LTS}-models are exactly labeled transition systems. We define compound programs as derived constructs as follows:

$$\langle\alpha\rangle\varphi \equiv \bullet(\alpha, \varphi) \qquad [\alpha]\varphi \equiv \neg\langle\alpha\rangle\neg\varphi$$
$$(\textsc{Seq})\ [\alpha \,;\, \beta]\varphi \equiv [\alpha][\beta]\varphi \qquad (\textsc{Choice})\ [\alpha \cup \beta]\varphi \equiv [\alpha]\varphi \wedge [\beta]\varphi$$
$$(\textsc{Test})\ [\psi?]\varphi \equiv (\psi \to \varphi) \qquad (\textsc{Iter})\ [\alpha^*]\varphi \equiv \nu X.\, (\varphi \wedge [\alpha]X)$$

Like for the embedding of modal μ-logic (Section VII-B), no axioms are needed. Let Γ^{DL} denote the empty Σ^{LTS}-theory.

Theorem 36. *For all DL formulas φ, the following are equivalent: (1) $\vdash_{\mathsf{DL}} \varphi$; (2) $\vDash_{\mathsf{DL}} \varphi$; (3) $\Gamma^{\mathsf{DL}} \vdash \varphi$; (4) $\Gamma^{\mathsf{DL}} \vDash \varphi$.*

We point out that the iterative operator $[\alpha^*]\varphi$ is axiomatized with *two* axioms in the proof system of DL (see, e.g., [13]):

$$(\text{DL-}\textsc{Iter}_1) \qquad \varphi \wedge [\alpha][\alpha^*]\varphi \leftrightarrow [\alpha^*]\varphi$$
$$(\text{DL-}\textsc{Iter}_2) \qquad \varphi \wedge [\alpha^*](\varphi \to [\alpha]\varphi) \to [\alpha^*]\varphi$$

while we just regard it as syntactic sugar, via (Iter). One may argue that (Iter) desugars to the ν-binder, though, which obeys the proof rules (Pre-Fixpoint) and (Knaster-Tarski) that essentially have the same appearance as (DL-Iter$_1$) and (DL-Iter$_2$). We agree. And that is exactly why we think that we should have *one uniform and fixed logic*, such as MmL, where general fixpoint axioms are given to specify and reason about *any fixpoint properties* of *any domains* and to develop general-purpose automatic tools and provers. When it comes to specific domains and special-purpose logics, we can define them as theories/notations in MmL, as what we have done in this section for modal μ-logic and all its fragment logics. Often, these special-purpose logics are simpler than MmL and more computationally efficient. In particular, modal μ-logic and all its fragment logics shown in this section are not only *complete* but also *decidable* [20], while MmL does not have any complete proof system and thus its validity is not semi-decidable. Therefore, the existing decision procedures and completeness results of these special-purpose logics give decision procedures and (global) completeness results (such as Theorem 31) for the corresponding MmL theories.

VIII. Instance: Reachability Logic

Reachability logic (RL) [2] is an approach to program verification using operational semantics. Different from other approaches such as Hoare-style verification, RL has a *language-independent* proof system that offers sound and relatively complete deduction for all programming languages. RL is the logic underlying the $\mathbb{K}$ framework [45], which has been used to define the formal semantics of various real languages such as C [3], Java [4], and JavaScript [5], yielding program verifiers for all these languages at no additional cost [6].

In spite of its generality w.r.t. languages, reachability logic is unfortunately limited to specifying and deriving only reachability properties. This limitation was one of the factors that motivated the development of MmL. Fig. 2 shows a few RL proof rules; notice that unlike Hoare logic proof rules, RL proof rules are not specific to any particular programming language. The programming language is given through its operational semantics as a set of axiom rules, to be used via the (Axiom) proof rule. The characteristic feature of RL is its (Circularity) rule, which supports reasoning about circular behavior and recursive program constructs. In this subsection, we show how RL is faithfully defined in MmL and all its proof rules, including (Circularity), can be proved in MmL.

A. RL syntax, semantics, and proof system

RL is parametric in a model of ML (without μ) called the *configuration model*. Specifically, fix a signature (of *static program configurations*) Σ^{cfg} which may have various sorts and symbols, among which there is a distinguished sort *Cfg*. Fix a Σ^{cfg}-model M^{cfg} called the *configuration model*, where M^{cfg}_{Cfg} is the set of all configurations. RL formulas are called *reachability rules*, or simply *rules*, and have the form $\varphi_1 \Rightarrow \varphi_2$ where φ_1, φ_2 are ML (without μ) Σ^{cfg}-patterns. A *reachability system S* is a finite set of rules, which yields a transition system $\mathbb{S} = (M^{\mathsf{cfg}}_{Cfg}, R)$ where $s\,R\,t$ iff there exist a rule $\varphi_1 \Rightarrow \varphi_2 \in S$ and an M^{cfg}-valuation ρ such that $s \in \bar{\rho}(\varphi_1)$ and $t \in \bar{\rho}(\varphi_2)$. A rule $\psi_1 \Rightarrow \psi_2$ is *S-valid*, denoted $S \vDash_{\mathsf{RL}} \psi_1 \Rightarrow \psi_2$, iff for all M^{cfg}_{Cfg}-valuations ρ and configurations $s \in \bar{\rho}(\psi_1)$, either there is an infinite trace $s\,R\,t_1\,R\,t_2\,R\ldots$ in $\mathbb{S}$ or there is a configuration t such that $s\;R^*\;r$ and $t \in \bar{\rho}(\psi_2)$. Therefore, validity in RL is defined in the spirit of *partial correctness*.

$$\text{(Axiom)}\quad \frac{\varphi_1 \Rightarrow \varphi_2 \in A}{A \vdash_C \varphi_1 \Rightarrow \varphi_2}$$

$$\text{(Transitivity)}\quad \frac{A \vdash_C \varphi_1 \Rightarrow \varphi_2 \quad A \cup C \vdash \varphi_2 \Rightarrow \varphi_3}{A \vdash_C \varphi_1 \Rightarrow \varphi_3}$$

$$\text{(Consequence)}\quad \frac{M^{\mathsf{cfg}} \vDash \varphi_1 \to \varphi_1' \quad A \vdash_C \varphi_1' \Rightarrow \varphi_2' \quad M^{\mathsf{cfg}} \vDash \varphi_2' \to \varphi_2}{A \vdash_C \varphi_1 \Rightarrow \varphi_2}$$

$$\text{(Circularity)}\quad \frac{A \vdash_{C \cup \{\varphi_1 \Rightarrow \varphi_2\}} \varphi_1 \Rightarrow \varphi_2}{A \vdash_C \varphi_1 \Rightarrow \varphi_2}$$

Fig. 2. Some selected proof rules in the proof system of reachability logic

The sound and relatively complete proof system of RL derives *reachability logic sequents* of the form $A \vdash_C \varphi_1 \Rightarrow \varphi_2$ where A (called *axioms*) and C (called *circularities*) are finite sets of rules. Initially we start with $A = S$ and $C = \emptyset$. As the proof proceeds, more rules can be added to C via (Circularity) and then moved to A via (Transitivity), which can then be used via (Axiom). We write $S \vdash_{\mathsf{RL}} \psi_1 \Rightarrow \psi_2$ to mean that $S \vdash_{\emptyset} \psi_1 \Rightarrow \psi_2$. Notice (Consequence) consults the configuration model M^{cfg} for validity, so the completeness result is *relative to* M^{cfg}. We recall the following result [2]:

Theorem 37. *For all reachability systems S satisfying some reasonable technical assumptions (see [2]) and all rules $\psi_1 \Rightarrow \psi_2$, we have $S \vDash_{\mathsf{RL}} \psi_1 \Rightarrow \psi_2$ iff $S \vdash_{\mathsf{RL}} \psi_1 \Rightarrow \psi_2$.*

B. Defining reachability logic in matching μ-logic

We define the extended signature $\Sigma^{\mathsf{RL}} = \Sigma^{\mathsf{cfg}} \cup \{\bullet \in \Sigma_{Cfg,Cfg}\}$ where "$\bullet$" is a unary symbol called *one-path next*. To capture the semantics of reachability rules $\varphi_1 \Rightarrow \varphi_2$, we define:

$$\begin{aligned} \text{"weak eventually"} \quad & \Diamond_w \varphi \equiv \nu X.\, \varphi \vee \bullet X \quad \text{// equal to } \neg\mathsf{WF} \vee \Diamond\varphi \\ \text{"reaching star"} \quad & \varphi_1 \Rightarrow^* \varphi_2 \equiv \varphi_1 \to \Diamond_w \varphi_2 \\ \text{"reaching plus"} \quad & \varphi_1 \Rightarrow^+ \varphi_2 \equiv \varphi_1 \to \bullet\Diamond_w \varphi_2 \end{aligned}$$

Notice that the "weak eventually" $\Diamond_w \varphi$ is defined similarly to the "eventually" $\Diamond\varphi \equiv \mu X.\, \varphi \vee \bullet X$, but instead of using least fixpoint μ-binder, we define it as a greatest fixpoint. One can prove that $\Diamond_w \varphi = \neg\mathsf{WF} \vee \Diamond\varphi$, that is, a configuration γ satisfies $\Diamond_w \varphi$ if either it satisfies $\Diamond\varphi$, or it is not well-founded, meaning that there exists an infinite execution path from γ. Also notice that "reaching plus" $\varphi_1 \Rightarrow^+ \varphi_2$ is a stronger version of "reaching star", requiring that $\Diamond_w \varphi_2$ should hold *after at least one step*. This *progressive condition* is crucial to the soundness of RL reasoning: as shown in (Transitivity), circularities are flushed into the axiom set only *after one reachability step is established*. This leads us to the following translation from RL sequents to MmL patterns.

Definition 38. Given a rule $\varphi_1 \Rightarrow \varphi_2$, define the MmL pattern $\boxdot(\varphi_1 \Rightarrow \varphi_2) \equiv \Box(\varphi_1 \Rightarrow^+ \varphi_2)$ and extend it to a rule set A as follows: $\boxdot A \equiv \bigwedge_{\varphi_1 \Rightarrow \varphi_2 \in A} \boxdot(\varphi_1 \Rightarrow \varphi_2)$. Define the translation RL2MmL from RL sequents to MmL patterns as follows:

$$\mathrm{RL2MmL}(A \vdash_C \varphi_1 \Rightarrow \varphi_2) = (\forall \boxdot A) \wedge (\forall \circ \boxdot C) \to (\varphi_1 \Rightarrow^\star \varphi_2)$$

where $\star = *$ if C is empty and $\star = +$ if C is nonempty. We use $\forall\varphi$ as a shorthand for $\forall\vec{x}.\varphi$ where $\vec{x} = FV(\varphi)$. Recall that the "$\circ$" in $\forall\circ\boxdot C$ is "all-path next".

Hence, the translation of $A \vdash_C \varphi_1 \Rightarrow \varphi_2$ depends on whether C is empty or not. When C is nonempty, the RL sequent is *stronger* in that it requires *at least one step* being made in $\varphi_1 \Rightarrow \varphi_2$. Axioms (those in A) are also *stronger* than circularities (those in C) in that axioms *always* hold, while circularities only hold *after at least one step* because of the leading all-path next "$\circ$"; and since the "next" is an "all-path" one, it does not matter which step is actually made, as circularities hold on *all* next states.

Theorem 39. *Let $\Gamma^{\mathsf{RL}} = \{\varphi \in \textsc{Pattern}^{ML}_{Cfg} \mid M^{\mathsf{cfg}} \vDash \varphi\}$ be the set of all ML patterns (without μ) of sort Cfg that hold in M^{cfg}. For all RL systems S and rules $\varphi_1 \Rightarrow \varphi_2$ satisfying the same technical assumptions in [2], the following are equivalent: (1) $S \vdash_{\mathsf{RL}} \varphi_1 \Rightarrow \varphi_2$; (2) $S \vDash_{\mathsf{RL}} \varphi_1 \Rightarrow \varphi_2$; (3) $\Gamma^{\mathsf{RL}} \vdash \mathrm{RL2MmL}(S \vdash_\emptyset \varphi_1 \Rightarrow \varphi_2)$; (4) $\Gamma^{\mathsf{RL}} \vDash \mathrm{RL2MmL}(S \vdash_\emptyset \varphi_1 \Rightarrow \varphi_2)$.*

Therefore, provided that an oracle for validity of ML patterns (without μ) in M^{cfg} is available, the MmL proof system is capable of deriving any reachability property that can be derived with the RL proof system. This result makes MmL an even more fundamental logic foundation for the $\mathbb{K}$ framework and thus for programming language specification and verification than RL, because it can express significantly more properties than partial correctness reachability.

IX. Future and Related Work

We discuss future work, open problems, and related work.

A. Relation to modal logics

Due to the duality between MmL symbols and modal logic modalities (Section III, Proposition 12), ML can be regarded as a nontrivial extension of modal logics. There are various directions to extend the basic propositional modal logic in the literature [20]. One is the *hybrid extension*, where first-order quantifiers "$\forall$" and "$\exists$" are added to the logic, as well as *state variables/names* that allow us to specify one particular state. Another is the *polyadic extension*, where modalities can take not just one argument, but any number of arguments, and there can be multiple modalities. MmL can be seen as a combination of both extensions, further extended with multiple sort universes. The local completeness of $\mathcal{H}$ (Theorem 16) also extends the completeness results of its fragment logics, including hybrid modal logic [21] and many-sorted polyadic modal logic [19].

B. Stronger completeness results of $\mathcal{H}$

There are various notions of completeness for modal logics (see, e.g., [46, Appendix B.6]). We recall three of them, adapted to the context of ML and its proof system $\mathcal{H}$, from the strongest to the weakest:

- Global completeness: $\Gamma \vDash_{\mathsf{ML}} \varphi$ implies $\Gamma \vdash_{\mathcal{H}} \varphi$;
- Strong local completeness: $\Gamma \vDash^{\mathrm{loc}}_{\mathsf{ML}} \varphi$ implies $\Gamma \vdash^{\mathrm{loc}}_{\mathcal{H}} \varphi$;
- Weak local completeness: $\vDash_{\mathsf{ML}} \varphi$ implies $\vdash_{\mathcal{H}} \varphi$;

where $\Gamma \vDash^{\mathrm{loc}}_{\mathsf{ML}} \varphi$, called *local semantic entailment*, means that $\bigcap_{\psi\in\Gamma} \bar{\rho}(\psi) \subseteq \bar{\rho}(\varphi)$ for all models M and valuations ρ; $\Gamma \vdash^{\mathrm{loc}}_{\mathcal{H}} \varphi$, called *local provability*, means that there exists a finite subset $\Gamma_0 \subseteq_{fin} \Gamma$ such that $\vdash_{\mathcal{H}} \wedge\Gamma_0 \to \varphi$, where $\wedge\Gamma_0$ is the conjunction of all patterns in Γ_0. Theorem 16 is a weak local completeness result for $\mathcal{H}$, but the way we actually prove it is by proving the strong local completeness theorem and then let $\Gamma = \emptyset$ (see [14]). What is unknown and left as future work is global completeness. Theorem 15 shows that global completeness holds for ML when Γ contains definedness symbols and axioms. We conjecture global completeness holds in general.

C. Decidability of matching μ-logic without FOL quantifiers

Modal μ-logic is known for its high expressiveness as well as its *decidability*, given that it can capture the true least/greatest fixpoints in models, As a result, modal μ-logic stands out from other fixpoint logics, such as LFP. As seen in Section VII, modal μ-logic can be seen as the syntactic fragment of MmL without FOL quantifiers (i.e., $\exists$-binder) or element variables that contains only one sort and one unary symbol. A natural question is whether the decidability result stil holds if we consider the MmL fragment without FOL quantifiers or element variables but containing multiple sorts and symbols of arbitrary arities. We conjecture it holds.

D. Alternative semantics of matching μ-logic

MmL cannot have a sound and complete proof system because we can precisely define $(\mathbb{N}, +, \times)$ (see Proposition 23). On the other hand, the proof system $\mathcal{H}_\mu$ turned out to be strong enough to prove all the proof rules of all the proof systems of all the logics discussed in this paper. Therefore, a natural question is whether we can find alternative models for MmL that make $\mathcal{H}_\mu$ complete. A promising direction towards such an alternative semantics is to consider the so-called *Henkin semantics* or *general semantics*, where the least fixpoint pattern $\mu X.\,\varphi$ does not evaluate to the true least fixpoint in models, but to *the least fixpoint that is definable in the logic*.

X. Conclusion

We made two main contributions in this paper. Firstly, we proposed a new sound and complete proof system $\mathcal{H}$ for matching logic (ML). Secondly, we extended ML with the least fixpoint μ-binder and proposed matching μ-logic (MmL). We showed the expressiveness of MmL by defining a variety of common logics about induction/fixpoints/verification in MmL. We hope that MmL may serve as a promising unifying foundation for specifying and reasoning about induction, fixpoints, as well as model checking and program verification.

Acknowledgments: We thank the anonymous reviewers for their valuable comments on drafts of this paper. The work presented in this paper was supported in part by NSF CNS 16-19275. This material is based upon work supported by the United States Air Force and DARPA under Contract No. FA8750-18-C-0092.

References

[1] G. Roşu, "Matching logic," *Logical Methods in Computer Science*, vol. 13, no. 4, pp. 1–61, 2017.

[2] G. Roşu, A. Ştefănescu, Ş. Ciobâcă, and B. M. Moore, "One-path reachability logic," in *Proceedings of the 28th Symposium on Logic in Computer Science (LICS'13)*. IEEE, 2013, pp. 358–367.

[3] C. Hathhorn, C. Ellison, and G. Roşu, "Defining the undefinedness of C," in *Proceedings of the 36th annual ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI'15)*. ACM, 2015, pp. 336–345.

[4] D. Bogdănaş and G. Roşu, "K-Java: A complete semantics of Java," in *Proceedings of the 42nd Symposium on Principles of Programming Languages (POPL'15)*. ACM, 2015, pp. 445–456.

[5] D. Park, A. Ştefănescu, and G. Roşu, "KJS: A complete formal semantics of JavaScript," in *Proceedings of the 36th annual ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI'15)*. ACM, 2015, pp. 346–356.

[6] A. Ştefănescu, D. Park, S. Yuwen, Y. Li, and G. Roşu, "Semantics-based program verifiers for all languages," in *Proceedings of the 2016 ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA'16)*. ACM, 2016, pp. 74–91.

[7] Y. Gurevich and S. Shelah, "Fixed-point extensions of first-order logic," *Annals of Pure and Applied Logic*, vol. 32, pp. 265–280, 1986.

[8] D. Kozen, "Results on the propositional μ-calculus," in *Proceedings of the 9th International Colloquium on Automata, Languages and Programming (ICALP'82)*. Springer, 1982, pp. 348–359.

[9] A. Pnueli, "The temporal logic of programs," in *Proceedings of the 18th Annual Symposium on Foundations of Computer Science (SFCS'77)*. IEEE, 1977, pp. 46–57.

[10] G. Roşu, "Finite-trace linear temporal logic: Coinductive completeness," *Formal Methods in System Design*, vol. 53, no. 1, pp. 138–163, 2018.

[11] M. J. Fischer and R. E. Ladner, "Propositional dynamic logic of regular programs," *Journal of Computer and System Sciences*, vol. 18, no. 2, pp. 194–211, 1979.

[12] D. Harel, "Dynamic logic," in *Handbook of Philosophical Logic*, ser. Synthese Library. Springer, 1984, vol. 165, pp. 497–604.

[13] D. Harel, J. Tiuryn, and D. Kozen, *Dynamic logic*. MIT Press, 2000.

[14] X. Chen and G. Roşu, "Matching μ-logic," University of Illinois at Urbana-Champaign, Tech. Rep., 2019. [Online]. Available: http://hdl.handle.net/2142/102281

[15] F. Lucio-Carrasco and A. Gavilanes-Franco, "A first order logic for partial functions," in *Proceedings of the 6th Annual Symposium on Theoretical Aspects of Computer Science (STACS'89)*. Springer, 1989, pp. 47–58.

[16] K. Futatsugi, J.-P. Jouannaud, and J. Meseguer, Eds., *Algebra, meaning, and computation*, 1st ed., ser. Theoretical Computer Science and General Issues. Springer, 2006, vol. 4060.

[17] J. R. Shoenfield, *Mathematical logic*. Addison-Wesley Pub. Co, 1967.

[18] P. Blackburn, M. d. Rijke, and Y. Venema, *Modal logic*. Cambridge University Press, 2001.

[19] I. Leustean and N. Moanga, "A many-sorted polyadic modal logic," *CoRR*, vol. abs/1803.09709, 2018. [Online]. Available: http://arxiv.org/abs/1803.09709

[20] P. Blackburn, J. van Benthem, and F. Wolter, Eds., *Handbook of modal logic*, 1st ed. Elsevier, 2006, vol. 3.

[21] P. Blackburn and M. Tzakova, "Hybrid completeness," *Logic Journal of IGPL*, vol. 6, no. 4, pp. 625–650, 1998.

[22] A. Tarski, "A lattice-theoretical fixpoint theorem and its applications," *Pacific Journal of Mathematics*, vol. 5, no. 2, pp. 285–309, 1955.

[23] J. A. Goguen, J. W. Thatcher, E. G. Wagner, and J. B. Wright, "Initial algebra semantics and continuous algebras," *Journal of the ACM*, vol. 24, no. 1, pp. 68–95, 1977.

[24] K. Gödel, *On formally undecidable propositions of principia Mathematica and related systems*. Courier corporation, 1992.

[25] A. I. Malc'ev, "Axiomatizable classes of locally free algebras of various type," *The Metamathematics of Algebraic Systems: Collected Papers*, vol. 1967, pp. 262–281, 1936.

[26] L. Löwenheim, "Über möglichkeiten im relativkalkül," *Mathematische Annalen*, vol. 76, no. 4, pp. 447–470, 1915.

[27] L. Kovács, S. Robillard, and A. Voronkov, "Coming to terms with quantified reasoning," in *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages (POPL'17)*. ACM, 2017, pp. 260–270.

[28] J. C. Blanchette, N. Peltier, and S. Robillard, "Superposition with datatypes and codatatypes," in *Proceedings of the 9th International Joint Conference on Automated Reasoning (IJCAR'18)*. Springer, 2018, pp. 370–387.

[29] D. Park, "Fixpoint induction and proofs of program properties," *Machine Intelligence*, vol. 5, pp. 59–78, 1969.

[30] P. Hitchcock and D. Park, "Induction rules and termination roofs," in *Proceedings of the 1st International Colloquium on Automata, Languages and Programming (ICALP'72)*. Springer, 1972, pp. 225–251.

[31] Z. Ésik, "Completeness of Park induction," *Theoretical Computer Science*, vol. 177, no. 1, pp. 217–283, 1997.

[32] G. Peano, *Arithmetices principia: Nova methodo exposita*. Fratres Bocca, 1889.

[33] E. Mendelson, *Introduction to mathematical logic*. Springer, 1979.

[34] M. Schönfinkel, "Über die Bausteine der mathematischen Logik," *Mathematische annalen*, vol. 92, no. 3-4, pp. 305–316, 1924.

[35] H. B. Curry, *Combinatory logic*. Amsterdam: North-Holland Pub. Co., 1958.

[36] A. Church, "A formulation of the simple theory of types," *The Journal of Symbolic Logic*, vol. 5, no. 2, pp. 56–68, 1940.

[37] S. Kreutzer, "Pure and applied fixed-point logics," Ph.D. dissertation, Bibliothek der RWTH Aachen, 2002.

[38] C. A. R. Hoare, "An axiomatic basis for computer programming," *Communications of the ACM*, vol. 12, no. 10, pp. 576–580, 1969.

[39] I. Walukiewicz, "Completeness of Kozen's axiomatisation of the propositional μ-calculus," *Information and Computation*, vol. 157, no. 1-2, pp. 142–182, 2000.

[40] G. Lenzi, "The modal μ-calculus: A survey," *Task quarterly*, vol. 9, no. 3, pp. 293–316, 2005.

[41] E. A. Emerson, "Temporal and modal logic," in *Formal Models and Semantics*. Elsevier, 1990, pp. 995–1072.

[42] V. Pratt, "Semantical consideration on Floyd-Hoare logic," in *Proceedings of the 17th Annual Symposium on Foundations of Computer Science (SFCS'76)*. IEEE, 1976, pp. 109–121.

[43] R. Alur, C. Courcoubetis, T. A. Henzinger, and P.-H. Ho, "Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems," in *Hybrid systems*. Springer, 1993, pp. 209–229.

[44] E. A. Lee, "Cyber physical systems: Design challenges," in *Proceedings of the 11th IEEE Symposium on Object Oriented Real-Time Distributed Computing (ISORC'08)*. IEEE, 2008, pp. 363–369.

[45] G. Rosu, "K—A semantic framework for programming languages and formal analysis tools," in *Dependable Software Systems Engineering*. IOS Press, 2017.

[46] M. Marx and Y. Venema, *Multi-dimensional modal logic*, ser. Applied Logic Series, D. M. Gabbay, Ed. Springer, 1997, vol. 4.