Connecting Constrained Constructor Patterns and Matching Logic

Xiaohong Chen[1], Dorel Lucanu[2], and Grigore Roșu[1]

[1] University of Illinois at Urbana-Champaign, USA
[2] Alexandru Ioan Cuza University of Iași, Romania
{xc3,grosu}@illinois.edu dlucanu@info.uaic.ro

Abstract. Constrained constructor patterns are pairs of a constructor term pattern and a quantifier-free first-order logic constraint, built from conjunction and disjunction. They are used to express state predicates for reachability logic defined over rewrite theories. Matching logic has been recently proposed as a unifying foundation for programming languages, specification and verification. It has been shown to capture several logical systems and/or models that are important for programming languages, including first-order logic with fixpoints and order-sorted algebra. In this paper, we investigate the relationship between constrained constructor patterns and matching logic. The comparison result brings us a mutual benefit for the two approaches. Matching logic can borrow computationally efficient proofs for some equivalences, and the language of the constrained constructor patterns can get a more logical flavor and more expressiveness.

1 Introduction

The subject of this paper is inspired by a comment given by José Meseguer in a private message: "I strongly conjecture that there is a deep connection between matching logic and the constrained constructor patterns. It would be great to better understand the details of such a connection."

Constrained constructor patterns are the bricks of the *rewrite-theory-generic* reachability logic framework [11], by which we mean that the reachability logic framework as considered in [11] is parametric in the underlying rewriting theory. The order-sorted specifications $(\Sigma, E \cup B)$, used as support for rewrite theories, consist of an order-sorted signature Σ, a set of particular equations B used to reason modulo B, and a set of equations E that can be turned into a set of rewrite rules $\overrightarrow{E}$ convergent modulo B, assuming that the theory $(\Sigma, E \cup B)$ is sufficiently complete [9]. In this paper, we work under the assumptions that ensure all the properties mentioned below (now we implicitly assume them). The definition of constrained constructor patterns is based on the strong relationship between the initial $(\Sigma, E \cup B)$-algebra $T_{\Sigma/E\cup B}$ and its canonical constructor $(\Omega, E_\Omega \cup B_\Omega)$-algebra $C_{\Omega/E_\Omega,B_\Omega}$. This relationship is briefly explained as follows:

1. $T_{\Sigma/E\cup B}$ is isomorphic to the canonical term-algebra $C_{\Sigma/E,B}$, consisting of B-equivalence classes of $\overrightarrow{E}$-irreducible-modulo-B Σ-terms;

2. $\Omega \subseteq \Sigma$ is the subsignature of constructors;
3. $C_{\Sigma/E,B}|_{\Omega} = C_{\Omega/E_{\Omega},B_{\Omega}}$.

A constrained constructor pattern predicate is a pair $u|\varphi$, where u is a constructor term pattern and φ is a quantifier-free first-order logic (FOL) formula. The set of constrained constructor patterns includes the constrained constructor pattern predicates and is closed under conjunction and disjunction. The semantics defined by $u|\varphi$ is given by the subset of states $[\![u|\varphi]\!] \subseteq C_{\Omega/E_{\Omega},B_{\Omega}}$ matching u, i.e., for each $a \in [\![u|\varphi]\!]$ there is a valuation ρ such that φ holds (written $\rho \vDash \varphi$) and $a = u\rho$.

There are several additional operations over constrained constructor patterns required to express reachability properties and to support their verification in a computational efficient way. These include (parameterized) subsumption, over-approximation of the complements, and parameterized intersections. The definitions of these operations exploits the cases when the matching and unification modulo $E \cup B$ can be efficiently solved, using, e.g., the theory of variants [4,7].

Matching Logic (ML) [10,3,2] is a variant of first-order logic (FOL) with fixpoints that makes no distinction between functions and predicates. It uses instead symbols and application to uniformly build patterns that can represent static structures and logical constraints at the same time. Semantically, ML patterns are interpreted as the sets of elements that match them. The functional interpretation is obtained by adding axioms like $\exists y. \mathtt{s}\, x = y$ that forces the pattern $\mathtt{s}\, x$ to be evaluated to a singleton. The conjunction and disjunction are interpreted as the intersection, respectively union. For instance, the ML pattern $\exists x{:}Nat.\, \mathtt{s}\, x \wedge (x = 2 \vee x = 5)$, when interpreted over the natural numbers, denotes the set $\{3, 6\}$ since $\mathtt{s}\, x$ is matched by the successor of x, constants 2 and 5 are matched by the numbers 2 and 5, respectively, and $x = n$ is a "predicate": it matches either the entire carrier set when x and n are matched by the same elements, or otherwise the empty set.

The **main contribution** of the paper is an insightful comparison of constrained constructor patterns and matching logic. Since order-sorted algebras can be captured in matching logic [2], we were tempted to think that this comparison is a natural one, because a constrained constructor pattern $u|\varphi$ can be seen as a special ML pattern $u \wedge \varphi$. When we started to formalize this intuition, we realized a few interesting challenges that we need to address:

- How to capture the logical reasoning modulo equations in B in ML?
- How to formalize the canonical model containing only constructor terms?
- What properties does the ML model corresponding to an OSA canonical model have?
- Which are the most suitable ML patterns that capture constrained constructor pattern operations?
- How to express the equivalence between a constrained constructor pattern and its ML encoding?

In order to better understand the relationship between the two approaches, we consider a running example, the QLOCK mutual exclusion protocol [5,11], and show how to define it in ML. This example gives us a better view of the specificity of ML axioms and how the OSA canonical model is reflected in ML. In this paper, we only consider the static structure of QLOCK. Since the ML axiomatization includes the complete specifications of natural numbers, (finite) list and (finite) multisets, and it specifies their carrier sets using least fixpoints, we can derive from the specifications an induction proof principle for them.

Structure of the paper. We define constrained constructor patterns and introduce the QLOCK example in Section 2. In Section 3, we introduce matching logic (ML) in details, as it was recently proposed. In Section 4 we discuss the axiomatization of free constructors and the encoding of OSA in ML, and a complete specification of the QLOCK configurations. In Section 5, we show the ML encoding of the constrained constructor patterns and their operations, which is our main contribution. We conclude in Section 6.

2 Constrained Constructor Patterns

We assume the readers are familiar with order-sorted equational and first-order logics (see, e.g., [8]). Here we briefly recall the definitions of constructor pattern predicates [11].

Definition 1. *An* order-sorted signature $\Sigma = (S, \leq, F)$ *contains a sort set S, a partial ordering $\leq\ \subseteq S \times S$ called* subsorting*, and a function (family) set $F = \{F_{s_1 \ldots s_n, s}\}_{s_1,\ldots,s_n,s \in S}$. We allow* subsort overloading*, i.e., $f \in F_{s_1 \ldots s_n, s} \cap F_{s'_1 \ldots s'_n, s'}$ with $s_1 \leq s'_1, \ldots, s_n \leq s'_n, s \leq s'$. An* order-sorted algebra $A = (\{A_s\}_{s \in S}, \{f_A\}_{f \in F})$ *contains (1) a nonempty carrier set A_s for every $s \in S$; we require $A_s \subseteq A_{s'}$ whenever $s \leq s'$; and (2) a function interpretation $f \colon M_{s_1} \times \cdots \times M_{s_n} \to M_s$ for every $f \in F_{s_1 \ldots s_n, s}$. Note that overloaded functions must coincide on the overlapped parts.*

A function $f \in F_{s_1 \ldots s_n, s}$ is denoted as $f : s_1 \times \cdots \times s_n \to s$. Let $X = \{X_s\}_{s \in S}$ be an S-indexed set of *sorted variables* denoted $x{:}s, y{:}s$. We use $T_\Sigma(X)$ to denote the *Σ-term algebra on X*, whose elements are (ground and non-ground) terms. We use $T_\Sigma = T_\Sigma(\emptyset)$ to denote the Σ-algebra of ground terms.

An (equational) *order-sorted theory* $(\Sigma, B \cup E)$ consists of an order-sorted signature Σ and a union set $B \cup E$ of (possibly conditional) Σ-equations (explained below). We assume that $F = \Omega \cup \Delta$, where Ω contains *constructors* and Δ contains *defined functions*. We assume that B contains a special class of axioms that usually express properties like associativity, commutativity, and identity of functions in Σ. Let $B_\Omega \cup E_\Omega$ be the axioms (equations) that only contain constructors in Ω. Then, $(B \setminus B_\Omega) \cup (E \setminus E_\Omega)$ is the set of axioms (equations) that specify defined functions in Δ.

Given $(\Sigma, B \cup E)$, its *initial model* is isomorphic to the *canonical term algebra* $C_{\Sigma/E,B}$ that contains $(B_\Omega \cup E_\Omega)$-equivalence classes of ground Ω-terms. For a ground Σ-term t that may contain defined functions, we let $\mathit{canf}(t) = [u]_{B_\Omega \cup E_\Omega}$

denote its *canonical form* in $C_{\Sigma/E,B}$, i.e., $u =_{B \cup E} t$ and $u \in T_\Omega$. Let $\rho\colon X \to T_\Omega$ be a valuation. We define its extension $\_\rho\colon T_\Sigma \to T_\Omega$ in the usual way.

Given $s \in S$, an *s-sorted constrained constructor pattern* is an expression $u|\varphi$, where $u \in T_\Omega(X)$ has sort s and φ is a quantifier-free Σ-formula; see [11, pp. 204]. The set of *constrained constructor pattern predicates* $\mathit{PatPred}(\Omega, \Sigma)$, is the smallest set that includes $\perp$ and constrained constructor patterns, and is closed under disjunction and conjunction. The *semantics* of a constrained constructor pattern predicate A is the set $[\![A]\!]_C$ of canonical terms that *satisfies* it:

$$[\![\perp]\!]_C = \emptyset \qquad [\![A \vee B]\!]_C = [\![A]\!]_C \cup [\![B]\!]_C \qquad [\![A \wedge B]\!]_C = [\![A]\!]_C \cap [\![B]\!]_C$$
$$[\![u|\varphi]\!]_C = \{\mathit{canf}(u\rho) \mid \rho\colon X \to T_\Omega, C_{\Sigma/E,B} \models \varphi\rho\}$$

2.1 A Running Example: QLOCK

QLOCK is a mutual exclusion protocol [5] that allows an unbounded number of (numbered) processes that are in one of the three states: “normal” (doing their own things), “waiting” for a resource, and “critical” when using the resource. A QLOCK state is a tuple $\langle n|w|c|q\rangle$ where n, w, c are multisets of identities of the processes that are in “normal”, “waiting”, and “critical” states, respectively, and q is the waiting queue, i.e., an associative list. In this paper, we are only interested in understanding how constrained constructor patterns express state predicates, so we only consider the static structure of QLOCK states, whose OSA specification [11] is given below:

$S = \{\mathit{Nat}, \mathit{List}, \mathit{MSet}, \mathit{NeMSet}, \mathit{Conf}, \mathit{State}, \mathit{Pred}\}$
$\leq\, = \{\mathit{Nat} < \mathit{List}, \mathit{Nat} < \mathit{NeMSet} < \mathit{MSet}\} \cup =_S$
Σ_Ω (constructors):
  $\mathbb{0} :\ \to \mathit{Nat},\ \mathbb{s}\_ : \mathit{Nat} \to \mathit{Nat}$
  $\mathit{nil} :\ \to \mathit{List},\ \_;\_ : \mathit{List} \times \mathit{List} \to \mathit{List}$
  $\mathit{empty} :\ \to \mathit{MSet},\ \_\,\_ : \mathit{MSet} \times \mathit{MSet} \to \mathit{MSet},$
  $\_\,\_ : \mathit{NeMSet} \times \mathit{NeMSet} \to \mathit{NeMSet}$
  $\_|\_|\_|\_ : \mathit{MSet} \times \mathit{MSet} \times \mathit{MSet} \times \mathit{List} \to \mathit{Conf}$
  $\langle\_\rangle : \mathit{Conf} \to \mathit{State}$
  $\mathit{tt} :\ \to \mathit{Pred}, \mathit{ff} :\ \to \mathit{Pred}$
$\Sigma(\text{QLOCK}) = \Sigma_\Omega \cup \{\mathit{dupl} : \mathit{MSet} \to \mathit{Pred}, \mathit{dupl} : \mathit{NeMSet} \to \mathit{Pred}\}$
B_Ω:
  associativity for list concatenation $\_;\_$ with the identity *nil*
  associativity/commutativity for multiset union $\_;\_$ with the identity *empty*
$E_\Omega = \emptyset$
$E = \{\mathit{dupl}(s\,u\,u) = \mathit{tt}\}$, where s is any multiset (could be empty).

The corresponding canonical model, denoted QLK, is given as:

$\mathsf{QLK}_{\mathit{Nat}} = \{\mathbb{0}, \mathbb{s}\,\mathbb{0}, \mathbb{s}^2\,\mathbb{0}, \ldots\}$
$\mathsf{QLK}_{\mathit{List}} = \mathsf{QLK}_{\mathit{Nat}} \cup \{\mathit{nil}\} \cup \{n_1; \ldots; n_k \mid n_i \in \mathsf{QLK}_{\mathit{Nat}}, 1 \leq i \leq k, k \geq 2\}$
$\mathsf{QLK}_{\mathit{NeMSet}} = \mathit{Nat} \cup \{[n_1, \ldots, n_k] \mid n_i \in \mathsf{QLK}_{\mathit{Nat}}, 1 \leq i \leq k, k \geq 2\}$
$\mathsf{QLK}_{\mathit{MSet}} = \mathsf{QLK}_{\mathit{NeMSet}} \cup \{\mathit{empty}\}$

$\mathsf{QLK}_{Conf} = \{x_1|x_2|x_3|y \mid x_1, x_2, x_3 \in \mathsf{QLK}_{MSet}, y \in \mathsf{QLK}_{List}\}$
$\mathsf{QLK}_{State} = \{\langle x \rangle \mid x \in \mathsf{QLK}_{Conf}\}$
$\mathsf{QLK}_{Pred} = \{tt, \mathit{ff}\}$

An example of a constrained constructor pattern predicate is $\langle n|w|c|q\rangle | dupl(n\, w\, c) \neq tt$, since no process can be waiting and critical at the same time.

3 Matching Logic

We give a compact introduction to matching logic (ML) syntax and semantics, and the important mathematical instruments that can be defined as theories and/or notations. For full details, we refer readers to [10,3,2].

3.1 Matching Logic Syntax and Semantics

ML is an unsorted logic whose formulas, called *patterns*, are constructed from constant symbols, two sets of variables (explained below), propositional constructs $\bot$ and $\to$, a binary application function, the FOL-style existential quantifier $\exists$, and the least fixpoint operator μ. In models, patterns are interpreted as the *sets* of elements that *match* them. Important mathematical instruments and structures, as well as various logical systems can be captured in ML.

Definition 2. *We assume two countably infinite sets of variables EV and SV, where EV is the set of* element variables *denoted $x, y, \ldots$ and SV is the set of* set variables *denoted $X, Y, \ldots$. Given an (at most) countable set of constant symbols Σ, the set of Σ-patterns, written* Pattern, *is inductively generated by the following grammar for every $\sigma \in \Sigma$, $x \in EV$, and $X \in SV$:*

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

where in $\mu X.\, \varphi$ we require that φ is positive in X, i.e., X is not nested in an odd number of times on the left-hand side of an implication $\varphi_1 \to \varphi_2$. This syntactic requirement is to make sure that φ is monotone with respect to the set X, and thus the least fixpoint denoted by $\mu X.\, \varphi$ exists.

Both $\exists$ and μ are binders, and we assume the standard notions of free variables, α-equivalence, and capture-avoiding substitution. Specifically, we use $FV(\varphi)$ to denote the set of (element and set) variables that occur free in φ. We regard α-equivalent patterns as syntactically identical. We write $\varphi[\psi/x]$ (resp. $\varphi[\psi/X]$) for the result of substituting ψ for x (resp. X) in φ, where bound variables are implicitly renamed to prevent variable capturing. We define the following logical constructs as syntactic sugar:

$$\begin{array}{lll} \neg\varphi \equiv \varphi \to \bot & \varphi_1 \vee \varphi_2 \equiv \neg\varphi_1 \to \varphi_2 & \varphi_1 \wedge \varphi_2 \equiv \neg(\neg\varphi_1 \vee \neg\varphi_2) \\ \top \equiv \neg\bot & \forall x.\, \varphi \equiv \neg\exists x.\, \neg\varphi & \nu X.\, \varphi \equiv \neg\mu X.\, \neg\varphi[\neg X/X] \end{array}$$

We assume the standard precedence between logical constructs and that application $\varphi_1\,\varphi_2$ binds the tightest. We abbreviate the sequential application $(\cdots((\varphi_1\,\varphi_2)\;\varphi_3)\cdots\varphi_n)$ as $\varphi_1\,\varphi_2\,\varphi_3\,\cdots\,\varphi_n$.

ML has a *pattern matching* semantics where patterns are interpreted in models as the *sets* of elements that *match* them.

Definition 3. *Given a symbol set Σ, a Σ-*model $(M, \_\bullet\_, \{\sigma_M\}_{\sigma\in\Sigma})$ *contains:*

- *M: a nonempty carrier set;*
- *$\_\bullet\_\colon M \times M \to \mathcal{P}(M)$ as the interpretation of application, where $\mathcal{P}(M)$ is the powerset of M;*
- *$\sigma_M \subseteq M$: a subset of M as the* interpretation *of $\sigma \in \Sigma$.*

By abuse of notation, we write M for the above model.

For notational simplicity, we extend $\_\bullet\_$ from over elements to over sets, *pointwisely*, as follows:

$$\_\bullet\_\colon \mathcal{P}(M) \times \mathcal{P}(M) \to \mathcal{P}(M) \quad A \bullet B = \bigcup_{a\in A, b\in B} a \bullet b \;\text{ for } A, B \subseteq M$$

Note that $\emptyset \bullet A = A \bullet \emptyset = \emptyset$ for any $A \subseteq M$.

Definition 4. *Given a symbol set Σ and a Σ-model M, an M-valuation $\rho\colon (EV \cup SV) \to (M \cup \mathcal{P}(M))$ is a function that maps element variables to elements of M and set variables to subsets of M, i.e., $\rho(x) \in M$ and $\rho(X) \subseteq M$ for every $x \in EV$ and $X \in SV$. We extend ρ from over variables to over patterns, denoted $\bar{\rho}\colon$ Pattern $\to \mathcal{P}(M)$, as follows:*

$$\bar{\rho}(x) = \{\rho(x)\} \quad \bar{\rho}(X) = \rho(X) \quad \bar{\rho}(\sigma) = \sigma_M \quad \bar{\rho}(\bot) = \emptyset \quad \bar{\rho}(\varphi_1\,\varphi_2) = \bar{\rho}(\varphi_1) \bullet \bar{\rho}(\varphi_2)$$
$$\bar{\rho}(\varphi_1 \to \varphi_2) = M \setminus (\bar{\rho}(\varphi_1) \setminus \bar{\rho}(\varphi_2)) \quad \bar{\rho}(\exists x.\,\varphi) = \bigcup_{a\in M} \overline{\rho[a/x]}(\varphi) \quad \bar{\rho}(\mu X.\,\varphi) = \mu\mathcal{F}^{\rho}_{X,\varphi}$$

where $\mathcal{F}^{\rho}_{X,\varphi}\colon \mathcal{P}(M) \to \mathcal{P}(M)$ is a monotone function defined as $\mathcal{F}^{\rho}_{X,\varphi}(A) = \overline{\rho[A/X]}(\varphi)$ for $A \subseteq M$, and $\mu\mathcal{F}^{\rho}_{X,\varphi}$ denotes its unique least fixpoint given by the Knaster-Tarski fixpoint theorem [12].

Definition 5. *Given M and φ, we say M* satisfies *φ, written $M \vDash \varphi$, iff $\bar{\rho}(\varphi) = M$ for all ρ. Given $\Gamma \subseteq$* Pattern, *we say M* satisfies *Γ, written $M \vDash \Gamma$, iff $\bar{\rho}(\varphi) = M$ for all ρ and $\varphi \in \Gamma$. We call Γ a* theory *and patterns in Γ* axioms.

3.2 Important Mathematical Instruments

Several mathematical instruments of practical importance, such as definedness, totality, equality, membership, set containment, functions and partial functions, constructors, and sorts can all be defined using patterns. We give a compact summary of their definitions in ML and introduce proper notations for them.

Definedness Symbol and Axiom. ML patterns are interpreted as subsets of M. This is different from the classic FOL, whose formulas evaluate to either true or false. However, it is easy to restore the classic two-value semantics in ML, by using M, the entire carrier set, to represent the logical true, and $\emptyset$, the empty set, to represent the logical false. Since M is nonempty, no confusion is possible. We call φ a *predicate* in M if $\bar{\rho}(\varphi) \in \{\emptyset, M\}$ for all ρ. In the following, we define a set of predicate patterns that represent the important mathematical instruments. These patterns are constructed from a special symbol called *definedness*.

Definition 6. *Let* $\lceil \_ \rceil$ *be a symbol, which we call the* definedness *symbol. We write* $\lceil \varphi \rceil$ *instead of* $\lceil \_ \rceil\ \varphi$*. Let* (Definedness) *be the axiom* $\forall x. \lceil x \rceil$*. We define the following important notations:*

$$\begin{array}{ll} \textit{totality}\ \lfloor \varphi \rfloor \equiv \neg \lceil \neg\varphi \rceil & \textit{equality}\ \varphi_1 = \varphi_2 \equiv \lfloor \varphi_1 \leftrightarrow \varphi_2 \rfloor \\ \textit{membership}\ x \in \varphi \equiv \lceil x \wedge \varphi \rceil & \textit{inclusion}\ \varphi_1 \subseteq \varphi_2 \equiv \lfloor \varphi_1 \to \varphi_2 \rfloor \end{array}$$

We also define their negations:

$$\varphi_1 \neq \varphi_2 \equiv \neg(\varphi_1 = \varphi_2) \quad x \notin \varphi \equiv \neg(x \in \varphi) \quad \varphi_1 \not\subseteq \varphi_2 \equiv \neg(\varphi_1 \subseteq \varphi_2)$$

In the following, when we say that we consider a theory Γ that contains certain axioms, we implicitly assume that the symbol set contains all symbols that occur in those axioms.

Sorts. ML is an unsorted logic and has no built-in support for sorts or many-sorted functions. However, we can define sorts as constant symbols and use patterns to axiomatize their properties. Specifically, for every sort s, we define a corresponding constant symbol also denoted s that represents its sort name. For technical convenience, we include the following axiom

$$(\textsc{Sort Name}) \quad \exists x. s = x$$

to specify that s is matched by exactly one element, which is the name of the sort s. To get the carrier set of s, we define a symbol $[\![\_]\!]$, which we call the *inhabitant* symbol, and we write $[\![\varphi]\!]$ instead of $[\![\_]\!]\ \varphi$. The intuition is that $[\![s]\!]$ is matched by exactly the elements that have sort s, i.e., it represents the carrier set of s. We also include a symbol *Sort* that is matched by all sort names, by including an axiom $s \in \mathit{Sort}$.

We can specify properties about sorts by patterns. E.g., the following axiom

$$(\textsc{Nonempty Inhabitant}) \quad [\![s]\!] \neq \bot$$

specifies that the carrier set of s is nonempty. The following axiom

$$(\textsc{Subsort}) \quad [\![s_1]\!] \subseteq [\![s_2]\!]$$

specifies that the carrier set of s_1 is a subset of that of s_2, i.e., s_1 is a *subsort* of s_2. We define *sorted negation* $\neg_s \varphi \equiv (\neg\varphi) \wedge [\![s]\!]$, which is matched by all elements of sort s that do not match φ. We define *sorted quantification* that restricts the ranges of $x, x_1, \ldots, x_n$ in the quantification:

$$\begin{array}{ll} \forall x{:}s.\, \varphi \equiv \forall x.\, x \in [\![s]\!] \to \varphi & \forall x_1, \ldots, x_n{:}s.\, \varphi \equiv \forall x_1{:}s.\, \ldots \forall x_n{:}s.\, \varphi \\ \exists x{:}s.\, \varphi \equiv \forall x.\, x \in [\![s]\!] \wedge \varphi & \exists x_1, \ldots, x_n{:}s.\, \varphi \equiv \exists x_1{:}s.\, \ldots \exists x_n{:}s.\, \varphi \end{array}$$

We can specify sorting restrictions of symbols. For example:

$$(\textsc{Sorted Symbol}) \quad \sigma\, [\![s_1]\!] \cdots [\![s_n]\!] \subseteq [\![s]\!]$$

requires $\sigma\, x_1 \cdots x_n$ to have sort s, given that $x_1, \ldots, x_n$ have sorts $s_1, \ldots, s_n$, respectively. For notational simplicity, we write $\sigma \in \Sigma_{s_1 \ldots s_n, s}$ to mean that we assume the axiom (Sorted Symbol) for σ.

Functions and Partial Functions. ML symbols are interpreted as relations, when they are applied to arguments. Indeed, $\sigma\, x_1 \cdots x_n$ is a pattern that can be matched zero, one, or more elements. In practice, we often want to specify that σ is a function (or partial function), in the sense that $\sigma\, x_1 \cdots x_n$ can be matched by exactly one (or at most one) element. That can be specified by the following axioms, respectively:

$$\begin{array}{ll} (\textsc{Function}) & \forall x_1{:}s_1.\, \ldots \forall x_n{:}s_n.\, \exists y{:}s.\, \sigma(x_1, \ldots, x_n) = y \\ (\textsc{Partial Function}) & \forall x_1{:}s_1.\, \ldots \forall x_n{:}s_n.\, \exists y{:}s.\, \sigma(x_1, \ldots, x_n) \subseteq y \end{array}$$

Recall that y is an element variable, so it is matched by exactly one element. For notational simplicity, we use the function notation $\sigma \colon s_1 \times \cdots \times s_n \to s$ to mean that we assume the axiom (Function) for σ. Similarly, we use the partial function notation $\sigma \colon s_1 \times \cdots \times s_n \rightharpoonup s$ to mean that we assume the axiom (Partial Function) for σ.

Constructors. *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 [6].[3] Specifically, let *Term* be a sort of *terms* and Σ be a set of constructors denoted c. We associate an arity $n_c \geq 0$ with every c. Consider the following axioms:

$$(\textsc{Function}, \text{ for all } c) \quad c \colon \underbrace{\mathit{Term} \times \cdots \times \mathit{Term}}_{n_c \text{ times}} \to \mathit{Term}$$

$$(\textsc{No Junk}) \quad \bigvee_{c \in C} \exists x_1, \ldots, x_{n_c}{:}\mathit{Term}.\, c\, x_1 \cdots x_{n_c}$$

$$(\textsc{No Confusion I}, \text{ for all } c \neq c')$$

$$\forall x_1, \ldots, x_{n_c}{:}\mathit{Term}.\, \forall y_1, \ldots, y_{n_{c'}}{:}\mathit{Term}.\, \neg \left(c\, x_1 \cdots x_{n_c} \wedge c'\, y_1 \cdots y_{n_{c'}} \right)$$

[3] This answers a question asked by Jacques Carette on the *mathoverflow* site (`https://mathoverflow.net/questions/16180/formalizing-no-junk-no-confusion`) ten years ago: Are there logics in which these requirements ("no junk, no confusion") can be internalized?

(No Confusion II, for all c)

$$\forall x_1, \ldots, x_{n_c}{:}\mathit{Term}.\, \forall y_1, \ldots, y_{n_c}{:}\mathit{Term}.$$
$$(c\, x_1 \cdots x_{n_c} \wedge c\, y_1 \cdots y_{n_c}) \to c\,(x_1 \wedge y_1) \cdots (x_{n_c} \wedge y_{n_c})$$

(Inductive Domain) $\mu T.\ \bigvee_{c \in C} c\, \underbrace{T \cdots T}_{n_i \text{ times}}$

Intuitively, (No Confusion I) says different constructs build different things; (No Confusion II) says constructors are injective; and (Inductive Domain) says the carrier set of *Term* is the smallest set that is closed under all constructors. We refer to the first two axioms as (No Confusion). Technically, (No Junk) is not necessary as it is implied by (Inductive Domain).

4 Encoding Order-Sorted Algebras

As seen in Section 3.2, the subset relation between the carrier sets of sorts can be captured in ML by patterns. Therefore, OSA and subsorting can be naturally captured in ML; see [2] for details. Specifically, to capture OSA, we define for every sorts $s \in S$ a corresponding sort, also denoted s, in ML. For every $s \leq s'$, we include a subsorting axiom $[\![s]\!] \subseteq [\![s']\!]$. We define for every OSA function $f \in F_{s_1 \ldots s_n, s}$ a corresponding symbol, also denoted f, and include the (Function) axiom, i.e., $f\colon s_1 \times \cdots \times s_n \to s$. This is summarized in Figure 1.

Let $\Sigma = (S, \leq, F)$ be an order-sorted signature and Σ^{ML} be the corresponding ML signature. Let $A = (\{A_s\}_{s \in S}, \{f_A\}_{f \in F})$ be an OSA. We define its derived ML Σ^{ML}-model, denoted A^{ML}, as in [2], which includes the standard interpretations of the definedness and inhabitant symbols, sorts, functions, and elements in A.

Theorem 1 (See [2]). *For every formula φ, we have $A^{\mathsf{ML}} \models \varphi^{\mathsf{ML}}$ iff $A \models \varphi$.*

4.1 QLOCK Example in ML

We have shown the OSA specification of QLOCK's static structures in Section 2.1 and the ML encoding of OSA in Section 4. Putting them together, we get an ML specification for QLOCK, which we show below in full details.

Notations

- $\overline{x}$: a syntactic sugar for $x_1, \ldots, x_n$
- $\forall \overline{x}{:}\overline{s}$: a syntactic sugar for $\forall x_1{:}s_1. \ldots \forall x_n{:}s_n$, where we assume $\bar{x}$ and $\bar{s}$ have the same length n.

ML Signature $\Sigma(\text{QLOCK})^{\mathsf{ML}}$ contains the following symbols (we remind readers of the mathematical instruments defined in Section 3.2):

- a definedness symbol $\lceil \_ \rceil$;
- an inhabitant symbol $[\![\_]\!]$;
- a symbol S for sort names;
- a symbol for each sort: *Nat*, *List*, *MSet*, *NeMSet*, *Conf*, *State*, *Pred*;
- a symbol for each function: *nil*, *conc*, *union*, *conf*, *state*, *dupl*, $\mathbb{0}$, $\mathbb{s}$;

	Order-Sorted Algebra	**Matching Logic**
Signature	$\Sigma = (S, \leq, F)$	$\Sigma^{\mathsf{ML}} = \{\lceil \_ \rceil, [\![\_]\!], \mathit{Sort}\} \cup S \cup F$
Axioms	OSA metalanguage	ML axioms
	$s \in S$	$s \in \mathit{Sort}$ $\exists y.\, s = y$ $[\![s]\!] \neq \bot$
	$s \leq s'$	$[\![s]\!] \subseteq [\![s']\!]$
	$f \in F_{s_1 \ldots s_n, s}$	$f \colon s_1 \times \cdots \times s_n \to s$
	$x{:}s$ (sorted variable)	$x \in [\![s]\!]$
Terms	t	t^{ML}
	$f(t_1, \ldots, t_n)$	$f\, t_1 \cdots t_n$
Sentences	φ	φ^{ML}
	$\{x_1, \ldots, x_n\}$ = variables in φ	$x_1 \in [\![s_1]\!] \wedge \cdots \wedge x_n \in [\![s_n]\!] \to (\varphi = \top)$
Model	A	$M \equiv A^{\mathsf{ML}}$
	$f_A : A_{s_1} \times \cdots A_{s_n} \to A_s$ $f_A(a_1, \ldots, a_n)$	$f_M\, a_1\, \cdots a_n = \{f_A(a_1, \ldots, a_n)\}$

Fig. 1. Given an order-sorted signature $\Sigma = (S, \leq, F)$ and a Σ-OSA A, we derive a ML signature Σ^{ML} and a corresponding Σ^{ML}-model $M \equiv A^{\mathsf{ML}}$.

ML Axioms Γ^{QLOCK} includes the (Definedness) axiom (see Definition 6) and the following axioms:

ML axioms for sort names

- the sort symbols are functional constants:

$$\exists y.\, y = \mathit{Nat} \qquad \exists y.\, y = \mathit{List} \qquad \exists y.\, y = \mathit{MSet} \quad \exists y.\, y = \mathit{NeMSet}$$

$$\exists y.\, y = \mathit{Conf} \quad \exists y.\, y = \mathit{State} \quad \exists y.\, y = \mathit{Pred}$$

- S is the set of sorts:

$$S = \mathit{Nat} \vee \mathit{List} \vee \mathit{MSet} \vee \mathit{NeMSet} \vee \mathit{Conf} \vee \mathit{State} \vee \mathit{Pred}$$

- for each sort $s \in S$, its carrier set is non-empty:

$$\forall s{:}S.\, [\![s]\!] \neq \bot$$

ML axioms for the natural numbers

- the constructors are functional:

$$\exists y{:}\mathit{Nat}.\, y = \mathbb{0} \qquad \forall x{:}\mathit{Nat}.\, \exists y{:}\mathit{Nat}.\, y = \mathbb{s}\, x$$

- "no confusion" axioms:

$$\forall x{:}\mathit{Nat}.\, \neg(\mathbb{0} \wedge \mathbb{s}\, x) \qquad \forall x, y{:}\mathit{Nat}.\, \mathbb{s}\, x \wedge \mathbb{s}\, y \to \mathbb{s}(x \wedge y)$$

- the domain of Nat is the smallest set that is closed under $\mathbb{0}$ and $\mathbb{s}$:

$$[\![\mathit{Nat}]\!] = \mu X.\, \mathbb{0} \vee \mathbb{s}(X)$$

  There is no need to add the "no junk" axiom $[\![\mathit{Nat}]\!] = \mathbb{0} \vee \mathbb{s}\, [\![\mathit{Nat}]\!]$ as it is a consequence of the above axiom.

Remark. Note that we use the sorted quantification in the above functional axioms. In other words, we only specify that $\mathbb{s}$ is a function when it is within the domain of Nat. Its behavior outside the domain of Nat is *unspecified.* This way, we allow maximal flexibility in terms of modeling, because each model (i.e., implementation) of the specification Γ can decide the behavior of $\mathtt{s}$ outside Nat. An "order-sorted-like"model will make $\mathtt{s}\,x$ return $\bot$, the empty set, whenever x is not in Nat, while an "error-algebra-like" model will make $\mathtt{s}\,x$ return $error$, a distinguished error element, to denote the "type error". Note that if we do not use the sorted quantification, but use the unsorted version, $\forall x.\, \exists y.\, y = \mathtt{s}\,x$, then we explicitly *exclude* the order-sorted model, which is not what we want.

Remark. We point out that the sorted quantification axioms do not *restrict* $\mathtt{s}$ to be only applicable within Nat. The pattern $\mathtt{s}\,x$ when x is outside the domain of Nat is still a well-formed pattern, whose semantics is not specified by the theory of natural numbers, but can be specified by other theories. For example, the theory of real numbers may re-use $\mathtt{s}$ and overload it as the increment-by-one function on reals. The theory of bounded arithmetic may re-use and overload $\mathtt{s}$ as the successor "function", which is actually a partial function and is undefined on the maximum value. The theory of transition systems may re-use and overload $\mathtt{s}$ as the successor "function", which is actually the underlying transition relation, and $\mathtt{s}\,x$ yields the set of all next states of the state x. In the last two cases, $\mathtt{s}$ is no longer a function because it is not true that $\mathtt{s}\,x$ always returns one element. Therefore, if we use not the sorted quantification axiom but the unsorted one, we cannot re-use $\mathtt{s}$ in the theories of bounded arithmetic or transition systems, without introducing inconsistency. Thus, by using sorted quantification for $\mathtt{s}$ in the theory of natural numbers, we do not restrict but actually encourage the re-use and overloading of $\mathtt{s}$ in other theories. On the other hand, ML is expressive enough if one wants to allow a restricted use of a symbol. For instance, if we want to restrict the use of $\mathtt{s}$ only to Nat, then we can add the axiom $\forall x.\, \lceil \mathtt{s}\,x \rceil \to x \in [\![Nat]\!]$.

ML axioms for Boolean values *Pred*

- the constructors are functional:
$$\exists y{:}Pred.\, y = tt \qquad \exists y{:}Pred.\, y = f\!f$$
- "no confusion" axiom: $\neg(tt \wedge f\!f)$
- the domain of $Pred$ consists only of $f\!f$ and tt:
$$[\![Pred]\!] = f\!f \vee tt$$

ML axioms for associative lists (over natural numbers)

- the constructors are functional:
$$\forall x,y{:}List.\, \exists z{:}List.\, z = conc\, x\, y \qquad \exists x{:}List.\, x = nil$$
- the associativity axiom:
$$\forall x,y,z{:}List.\, conc(conc\, x\, y)\, z = conc\, x\, (conc\, y\, z)$$
- the unity axioms:
$$\forall x{:}List.\, conc\, x\, nil = x \qquad \forall x{:}List.\, conc\, nil\, x = x$$

- the domain of *List* is the smallest set that includes $[\![Nat]\!]$ and closed under *conc* and *nil*:

  $$[\![List]\!] = \mu X.\, [\![Nat]\!] \vee nil \vee conc\, X\, X$$

  There is no need to add the subsort axiom $[\![Nat]\!] \subseteq [\![List]\!]$ to Γ since it is a consequence of the above axiom.

ML axioms for multisets (over natural numbers)

- the constructors are functional:

  $$\exists y{:}MSet.\, y = empty \qquad \forall x{,}y{:}MSet.\, \exists z{:}MSet.\, z = union\, x\, y$$

  $$\forall x{,}y{:}NeMSet.\, \exists z{:}NeMSet.\, z = union\, x\, y$$

- the associativity axiom:

  $$\forall x{,}y{,}z{:}MSet.\, union(union\, x\, y)\, z = union\, x\, (union\, y\, z)$$

- the unity and commutativity axioms:

  $$\forall x{:}MSet.\, union\, x\, empty = x \qquad \forall x, y{:}MSet.\, union\, x\, y = union\, y\, x$$

- the domain axiom:

  $$[\![NeMSet]\!] = \mu\, X.\, [\![Nat]\!] \vee union\, X\, X \quad [\![MSet]\!] = empty \vee [\![NeMSet]\!]$$

  The axioms $[\![Nat]\!] \subseteq [\![NeMSet]\!]$ and $[\![NeMSet]\!] \subseteq [\![MSet]\!]$, corresponding to subsorting relations $Nat < NeMSet$ and respectively $NeMSet < MSet$, are not needed, as they are consequences of the above.

ML axioms for configurations

- the constructors are functional:

  $$\forall x_1{,}x_2{,}x_3{:}MSet.\, \forall y{:}List.\, \exists z{:}Conf.\, conf\, x_1\, x_2\, x_3\, y = z$$

- "no confusion" axiom:

  $\forall x_1{,}x_2{,}x_3{,}x'_1{,}x'_2{,}x'_3{:}MSet.\, \forall y{,}y'{:}List.$

  $$conf\, x_1\, x_2\, x_3\, y \wedge conf\, x'_1\, x'_2\, x'_3\, y' \to conf\, (x_1 \wedge x'_1)(x_2 \wedge x'_2)(x_3 \wedge x'_3)(y \wedge y')$$

- the domain of *Conf* is the set that is closed under *conf*:

  $$[\![Conf]\!] = conf\, [\![MSet]\!]\, [\![MSet]\!]\, [\![MSet]\!]\, [\![List]\!]$$

ML axioms for states

- the constructors are functional:

  $$\forall x{:}Conf.\, \exists y{:}State.\, state\, x = y$$

- "no confusion" axiom:

  $$\forall x{,}x'{:}Conf.\, state\, x \wedge state\, x' \to state\, x \wedge x'$$

- the domain of *State* is the set that is closed under *state*:

  $$[\![State]\!] = state\, [\![Conf]\!]$$

The specification of the carrier set for the sorts *Nat*, *List*, *MSet*, and *NeMSet* as least fix points allows to formalize in ML of their induction proof principles. In what follows, $\varphi(x)$ says that the pattern φ depends on the variable x.

ML axioms that define *dupl*

We here give the complete specification of *dupl*:

$$\forall x{:}MSet.\, \exists y{:}Pred.\, dupl\, x = y$$
$$\forall s.\, \exists s', u.\, s =_{NeMSet} union\, s'(union\, u\, u) \rightarrow dupl\, s = tt$$
$$\forall s.\, \forall s', u.\, s \neq_{MSet} union\, s'(union\, u\, u) \rightarrow dupl\, s = f\!f$$

Proposition 1. $\mathsf{QLK}^{\mathsf{ML}} \models \Gamma^{\mathrm{QLOCK}}$.

Proof. By construction.

In the following, we show that *inductive reasoning* is available in $\mathsf{QLK}^{\mathsf{ML}}$ for natural numbers, (finite) lists, and (finite) multisets. We write $\varphi(x)$ to mean a pattern φ with a distinguished variable x and write $\varphi(t)$ to mean $\varphi[t/x]$.

Proposition 2 (Peano Induction).

$$\Gamma^{\mathrm{QLOCK}} \models \varphi(0) \wedge (\forall y{:}Nat.\, \varphi(y) \rightarrow \varphi(\mathtt{s}\, y)) \rightarrow \forall x{:}Nat.\, \varphi(x)$$

Proof. See [2].

Since the specifications for lists and multisets do not include "no confusion" axioms (due to the associativity, commutativity and identity axioms), their induction principles are given only for the ML model generated from the canonical OSA. This is sufficient for the purpose of this paper, because our goal is to show a faithful ML representation of constrained constructor patterns, whose semantics are given in the canonical model.

Proposition 3 (List and Multiset Induction).

$$\mathsf{QLK}^{\mathsf{ML}} \models \varphi(nil) \wedge \forall x{:}Nat. \varphi(x) \wedge (\forall \ell_1{,}\ell_2{:}List. \varphi(\ell_1) {\wedge} \varphi(\ell_2) {\rightarrow} \varphi(conc\, \ell_1\, \ell_2))) \rightarrow \forall \ell{:}List.\, \varphi(\ell)$$

$$\mathsf{QLK}^{\mathsf{ML}} \models \forall x{:}Nat. \varphi(x) \wedge (\forall m_1{,}m_2{:}NeMSet. \varphi(m_1) {\wedge} \varphi(m_2) {\rightarrow} \varphi(union\, m_1\, m_2)) \rightarrow \forall m{:}NeMSet.\, \varphi(m)$$

$$\mathsf{QLK}^{\mathsf{ML}} \models \varphi(empty) \wedge \forall x{:}Nat.\, \varphi(x) \wedge (\forall m_1{,}m_2{:}MSet.\, \varphi(m_1) \wedge \varphi(m_2) \rightarrow \varphi(union\, m_1\, m_2)) \rightarrow \forall m{:}MSet.\, \varphi(m)$$

Proof. By the inductive principle of the canonical model QLK and Theorem 1.

5 Encoding Constrained Constructor Patterns in ML

Let $(\Sigma, B \cup E)$ be an order-sorted theory with $(\Omega, B_\Omega \cup E_\Omega)$ being its subtheory of constructors. Recall that $C_{\Sigma/E,B}$ denotes the canonical constructor term algebra. Let $(\Sigma^{\mathsf{ML}}, \Gamma^{\Sigma,E,B})$ be the ML translation of $(\Sigma, E \cup B)$ with $\Gamma^{\Sigma,E,B} = B^{\mathsf{ML}} \cup E^{\mathsf{ML}}$, as discussed in Section 4.

Definition 7. *For a constrained constructor pattern $u|\varphi$, its ML translation is the pattern $u^{\mathsf{ML}} \wedge \varphi^{\mathsf{ML}}$. The ML translations of constrained constructor pattern predicates are defined in the expected way, where $\bot$ translates to $\bot$, conjunction translates to conjunction, and disjunction translates to disjunction.*

The canonical model $C_{\Sigma/E,B}$ has a corresponding $(\Sigma^{\mathsf{ML}}, \Gamma^{\Sigma,E,B})$-model $C^{\mathsf{ML}}_{\Sigma/E,B}$ by Theorem 1. For $\rho\colon X \to T_\Omega$ and a FOL formula φ, we have $C_{\Sigma/B,E} \models \varphi\rho$ iff $C^{\mathsf{ML}}_{\Sigma/E,B} \models (\varphi\rho)^{\mathsf{ML}}$ by the same theorem. This allows us to define the semantics of a constrained constructor pattern $[\![u|\varphi]\!]$ as the interpretation of the ML pattern $\exists\overline{x}{:}\overline{s}.\, u^{\mathsf{ML}} \wedge \varphi^{\mathsf{ML}}$ in $C^{\mathsf{ML}}_{\Sigma/E,B}$, where $\overline{x}{:}\overline{s} = FV(u \wedge \varphi)$.

Next we explain in ML terms some of the constrained constructor pattern operations discussed in [11]. We regard a substitution $\sigma \triangleq \{x_1 \mapsto t_1, \ldots, x_n \to t_n\}$ as the ML pattern $\sigma^{\mathsf{ML}} \triangleq x_1 = t_1 \wedge \cdots \wedge x_n = t_n$.

Constrained Constructor Pattern Subsumption. In [11], the following question is asked: When is the constrained constructor pattern $u|\psi$ an instance of a finite family $\{(v_i|\psi_i) \mid i \in I\}$, i.e., $[\![u|\varphi]\!] \subseteq \bigcup_{i\in I}[\![v_i|\psi_i]\!]$? Perhaps, at this level of abstraction, the above question is unclear, because we do not know yet what exactly it means by “when”. Let us elaborate it. The constrained constructor patterns are evaluated in the canonical model $C_{\Sigma/E,B}$, so the above question asks when there is a computationally efficient way to decide whether[4]

$$C_{\Sigma/E,B} \models [\![u|\varphi]\!] \subseteq \bigcup_{i\in I}[\![v_i|\psi_i]\!]$$

The answer is given by $E_\Omega \cup B_\Omega$-matching. Let $\textsc{match}(u, \{v_i \mid i \in I\})$ denote the set of pairs (i, β) with β a substitution such that $u =_{E_\Omega\cup B_\Omega} v_i\beta$, i.e., β matches v_i on u modulo $E_\Omega \cup B_\Omega$. Assuming that $u|\psi$ and $\{(v_i|\psi_i) \mid i \in I\}$ do not share variables, the constrained constructor pattern subsumption is formally defined as follows:

Definition 8 ([11]). *A family of constrained constructor patterns* $\{(v_i|\psi_i) \mid i \in I\}$ subsumes $u|\varphi$, *denoted* $u|\varphi \sqsubseteq \{(v_i|\psi_i) \mid i \in I\}$, *iff*

$$C_{\Sigma/B,E} \models \varphi \to \bigvee_{(i,\beta)\in\textsc{match}(u,\{v_i|i\in I\})} \psi_i\beta$$

Defined in this way, the constrained constructor pattern subsumption is computationally cheap in some cases; see [11]. One such case for example is when $E = \emptyset$ and Ω consists of associativity or associativity-commutativity and the terms are not too large. Note that $u|\varphi \sqsubseteq \{(v_i|\psi_i) \mid i \in I\}$ implies $[\![u|\varphi]\!] \subseteq \bigcup_{i\in I}[\![v_i|\psi_i]\!]$, but the inverse implication is not always true. The following counterexample is from [11], where a simple “inductive” instantiation of variable m by 0 and $s(k)$ can yield a proof by subsumption for the above set inclusion. Formally, let $\langle \_, \_ \rangle$ denote the pairing of natural numbers. Then we have $[\![\langle n, m\rangle|\top]\!] \subseteq [\![\langle x, 0\rangle|\top \vee \langle y, \mathsf{s}(z)\rangle|\top]\!]$, but $\langle n, m\rangle|\top \not\sqsubseteq \langle x, 0\rangle|\top \vee \langle y, \mathsf{s}(z)\rangle|\top$.

Let us discuss the ML counterpart of the subsumption. The ML pattern that corresponds to $[\![u|\varphi]\!] \subseteq \bigcup_{i\in I}[\![(v_i|\psi_i)]\!]$, is the following:

$$\left(\exists\overline{x}{:}\overline{s}.\, u^{\mathsf{ML}} \wedge \varphi^{\mathsf{ML}}\right) \subseteq \left(\bigvee_{i\in I} \exists\overline{y_i}{:}\overline{s_i}.\, v_i^{\mathsf{ML}} \wedge \psi_i^{\mathsf{ML}}\right)$$

[4] This is an informal notation because $[\![u|\varphi]\!] \subseteq \bigcup_{i\in I}[\![v_i|\psi_i]\!]$ is not exactly a formula.

where $\overline{x}{:}\overline{s} = FV(u|\varphi)$, and $\overline{y_i}{:}\overline{s_i} = FV(v_i|\psi_i)$. Since the two patterns do not share variables by assumption, the above is a well-formed ML pattern (we remind that $\varphi \subseteq \varphi'$ is the sugar-syntax of the ML pattern $\lfloor \varphi \to \varphi' \rfloor$).

The ML translation of the definition for $u|\varphi \sqsubseteq \{(v_i|\psi_i) \mid i \in I\}$ is

$$C^{\mathsf{ML}}_{\Sigma/B,E} \models \varphi^{\mathsf{ML}} \to \bigvee_{(i,\beta)\in \textsc{match}(u,\{v_i|i\in I\})} \left(\psi_i^{\mathsf{ML}} \wedge \beta^{\mathsf{ML}}\right)$$

where β^{ML} is the pattern describing the substitution β. We can prove now that the two ML patterns are equivalent:

Theorem 2. *The following holds:*

$$C^{\mathsf{ML}}_{\Sigma/E,B} \models \left(\exists \overline{x}{:}\overline{s}.\, u^{\mathsf{ML}} \wedge \varphi^{\mathsf{ML}}\right) \subseteq \left(\bigvee_{i\in I} \exists \overline{y_i}{:}\overline{s_i}.\, v_i^{\mathsf{ML}} \wedge \psi_i^{\mathsf{ML}}\right) \leftrightarrow$$
$$\left(\varphi^{\mathsf{ML}} \to \bigvee_{(i,\beta)\in \textsc{match}(u,\{v_i|i\in I\})} (\psi_i^{\mathsf{ML}} \wedge \beta^{\mathsf{ML}})\right)$$

Explanation. The key property is that of the match result (i, β), which satisfies that $u =_{E_\Omega \cup B_\Omega} v_i\beta$. In other words, β is the logical constraint that states that u can be matched by v_i. Thus, the reasoning is as follows. Intuitively, the LHS holds when $u^{\mathsf{ML}} \wedge \varphi^{\mathsf{ML}}$ is $\bot$, i.e., φ^{ML} is $\bot$, or when u^{ML} can be matched by v_i^{ML} for some i. This yields the RHS, which states that if φ^{ML} holds, then there exists i such that u is matched by the constraint term pattern $v_i|\psi_i$. The matching part is equivalent to the logical constraint β given by the matching function MATCH, and ψ_i is the logical constraint in the original constraint term pattern. Both need to be satisfied, and thus we have $\psi_i^{\mathsf{ML}} \wedge \beta^{\mathsf{ML}}$ on the RHS.

Regarding the counterexample, we show that

$$C^{\mathsf{ML}}_{\Sigma/E,B} \models \exists m,n{:}Nat.\, \langle n, m\rangle \subseteq \exists x,y,z{:}Nat.\, \langle x, 0\rangle \vee \langle y, \mathtt{s}(z)\rangle \qquad (*)$$

is proved in ML. Consider $\varphi(m) \triangleq \forall n,x,y,z{:}Nat.\, \langle n, m\rangle \subseteq \langle x, 0\rangle \vee \langle y, \mathtt{s}(z)\rangle$ and applying the induction principle for natural numbers, given by Proposition 2, we obtain

$$C^{\mathsf{ML}}_{\Sigma/E,B} \models \forall m{:}Nat.\, \exists n{:}Nat.\, \langle n, m\rangle \subseteq \exists x,y,z{:}Nat.\, \langle x, 0\rangle \vee \langle y, \mathtt{s}(z)\rangle$$

which implies $(*)$.

Over-Approximating Complements. In [11] it is showed that the complement of a constrained constructor pattern cannot be computed using negation, i.e, $[\![u|\top]\!]\setminus[\![u|\varphi]\!] = [\![u|\neg\varphi]\!]$ does not always hold, but the inclusion $[\![u|\top]\!]\setminus[\![u|\varphi]\!] \subseteq [\![u|\neg\varphi]\!]$ holds. Therefore an over-approximation of the difference is defined as:

$$[\![u|\varphi]\!] \setminus\!\!\setminus [\![u|\psi]\!] \triangleq [\![u|\varphi]\!] \cap [\![u|\neg\psi]\!] \quad (= [\![u|\varphi \wedge \neg\psi]\!])$$

Since ML has negation, the difference $[\![u|\top]\!] \setminus [\![u|\varphi]\!]$ is the same with the interpretation in $C^{\mathsf{ML}}_{\Sigma/E,B}$ of the ML pattern

$$\exists \overline{x}{:}\overline{s}.\, u^{\mathsf{ML}} \wedge \neg(\exists \overline{x}{:}\overline{s}.\, (u^{\mathsf{ML}} \wedge \varphi^{\mathsf{ML}}))$$

The constructor pattern predicate $[\![u|\top]\!]$ is the same with the interpretation in $C^{\mathsf{ML}}_{\Sigma/E,B}$ of the pattern $\exists\overline{x}{:}\overline{s}.\, u^{\mathsf{ML}}$, where $\overline{x}{:}\overline{s}$ is the set of variables occurring in u, and constructor predicate $[\![u|\neg\varphi]\!]$ is the same with the interpretation of $\exists\overline{x}{:}\overline{s}.\,(u^{\mathsf{ML}} \wedge \neg\varphi^{\mathsf{ML}})$.

The counterexample for equality as in [11] is $u \triangleq (x, y, z)$, as a multiset over $\{a, b, c\}$, $\varphi \triangleq x \neq y$. Using ML we may explain why $[\![u|\top]\!] \setminus [\![u|\varphi]\!] = [\![u|\neg\varphi]\!]$ does not hold in a more generic way. We use the notation from the QLOCK example. Apparently, the interpretations of $\exists x{,}y{,}z{:}\mathit{MSet}.\,(\mathit{union}\, x\, y\, z) \wedge x \neq y$ and $\exists x{,}y{,}z{:}\mathit{MSet}.\,(\mathit{union}\, x\, y\, z) \wedge x = y$ are disjoint because $a \neq b$ and $a = b$ are contradictory. This is not true because Γ includes the axioms ACU for the multisets; let us denote these axioms by ϕ. Then the two patterns are equivalent to $\exists x{,}y{,}z{:}\mathit{MSet}.\,(\mathit{union}\, x\, y\, z) \wedge x \neq y \wedge \phi$ and $\exists x{,}y{,}z{:}\mathit{MSet}.\,(\mathit{union}\, x\, y\, z) \wedge x = y \wedge \phi$, respectively. Obviously, $x \neq y \wedge \phi$ and $x = y \wedge \phi$ are not contradictory and the two patterns could match common elements.

The difference $[\![u|\varphi]\!] \setminus [\![u|\psi]\!]$ is the same as the interpretation of the pattern

$$\exists\overline{x}{:}\overline{s}.\,(u^{\mathsf{ML}} \wedge \varphi^{\mathsf{ML}}) \wedge \neg(\exists\overline{x}{:}\overline{s}.\,(u^{\mathsf{ML}} \wedge \psi^{\mathsf{ML}}))$$

and $[\![u|\varphi]\!] \setminus\!\!\setminus [\![u|\psi]\!]$ is the same as the interpretation of

$$\exists\overline{x}{:}\overline{s}.\,(u^{\mathsf{ML}} \wedge \varphi^{\mathsf{ML}}) \wedge \exists\overline{x}{:}\overline{s}.\,(u^{\mathsf{ML}} \wedge \neg\psi^{\mathsf{ML}}),$$

which is equivalent to $\exists\overline{x}{:}\overline{s}.\,(u^{\mathsf{ML}} \wedge \varphi^{\mathsf{ML}} \wedge \neg\psi^{\mathsf{ML}})$. We can prove that $[\![u|\varphi]\!] \setminus\!\!\setminus [\![u|\psi]\!]$ is indeed an over-approximation of the difference:

Proposition 4. *The following holds:*

$$C^{\mathsf{ML}}_{\Sigma/E,B} \models \exists\overline{x}{:}\overline{s}.\,(u^{\mathsf{ML}} \wedge \varphi^{\mathsf{ML}}) \wedge \neg(\exists\overline{x}{:}\overline{s}.\,(u^{\mathsf{ML}} \wedge \psi^{\mathsf{ML}})) \subseteq \exists\overline{x}{:}\overline{s}.\,(u^{\mathsf{ML}} \wedge \varphi^{\mathsf{ML}} \wedge \neg\psi^{\mathsf{ML}})$$

Parameterized Intersections. The intersection of two constrained constructor patterns that share a set of variables Y is defined as

$$(u|\varphi) \wedge_Y (v|\psi) \triangleq \bigvee_{\alpha \in \mathit{Unif}_{E_\Omega \cup B_\Omega}(u,v)} (u|\varphi \wedge \psi\alpha)$$

where $\mathit{Unif}_{E_\Omega \cup B_\Omega}(u, v)$ is a complete set of $E_\Omega \cup B_\Omega$-unifiers (the parameterized intersection is defined only when such a set exists). We have

$$[\![(u|\varphi) \wedge_Y (v|\psi)]\!] = \bigcup_{\rho \in [Y \to T_\Omega]} [\![u|\varphi]\!] \cap [\![v|\psi]\!]$$

For the case when $E = B = \emptyset$, it is shown in [1] that

$$\Gamma^{\Sigma} \models u \wedge v \leftrightarrow u \wedge \sigma^{\mathsf{ML}}$$

where σ is the most general unifier of u and v. We obtain as a consequence that $(u \wedge \varphi) \wedge (v \wedge \psi)$ is equivalent to $u \wedge \sigma^{\mathsf{ML}} \wedge \varphi \wedge \psi$, which is the ML translation of the corresponding constrained constructor pattern $(u|\varphi) \wedge_Y (v|\psi)$. We claim that this result can be generalized:

Theorem 3. *If $\{\sigma_1, \dots, \sigma_k\}$ is a complete set of $B_\Omega \cup E_\Omega$-unifiers for u_1 and u_2, then $C^{\mathsf{ML}}_{\Sigma/E,B} \models (u_1 \wedge u_2) \leftrightarrow (u_i \wedge (\sigma_1^{\mathsf{ML}} \vee \cdots \vee \sigma_k^{\mathsf{ML}}))$, for $i = 1, 2$.*

So, the parameterized intersection of two constrained constructor patterns is encoded in ML by the conjunction of the corresponding ML patterns.

Parameterized Containments. Given the constrained constructor patterns $u|\varphi$ and $\{(v_i|\psi_i) \mid i \in I\}$ with the shared variables Z, their set containment is defined as follows:

$$[\![u|\varphi]\!] \subseteq_Z [\![\bigvee_{i\in I}(v_i|\psi_i)]\!] \quad \text{iff} \quad \forall\rho \in [Z \to T_\Omega] \text{ s.t. } [\![(u|\varphi)\rho]\!] \subseteq [\![\bigvee_{i\in I}(v_i|\psi_i)\rho]\!]$$

The Z-parameterized subsumption of $u|\varphi$ by $\{(v_i|\psi_i) \mid i \in I\}$, denoted $u|\varphi \sqsubseteq_Z \bigvee_{i\in I}(v_i|\psi_i)$, holds iff $C_{\Sigma/E,B} \models \varphi \to \bigvee_{(i,\beta)\in\textsc{match}(u,\{v_i|i\in I\},Z)}(\psi_i\beta)$. The following result holds: if $u|\varphi \sqsubseteq_Z \bigvee_{i\in I}(v_i|\psi_i)$ then $[\![u|\varphi]\!] \subseteq_Z [\![\bigvee_{i\in I}(v_i|\psi_i)]\!]$.

Let us discuss the ML counterpart of the parameterized subsumption. The ML pattern expressing $[\![u|\varphi]\!] \subseteq \bigcup_{i\in I}[\![(v_i|\psi_i)]\!]$ is

$$\forall\overline{z}{:}\overline{s'}.\ \left(\exists\overline{x}{:}\overline{s}.\, u^{\mathsf{ML}} \wedge \varphi^{\mathsf{ML}} \subseteq \bigvee_{i\in I} \exists\overline{y_i}{:}\overline{s_i}.\, v_i^{\mathsf{ML}} \wedge \psi_i^{\mathsf{ML}}\right)$$

where $\overline{z}{:}\overline{s'}$ is the set of shared variables freely occurring in both $u|\varphi$ and $\{(v_i|\psi_i) \mid i \in I\}$, $\overline{x}{:}\overline{s}$ is the set of variables different of $\overline{z}{:}\overline{s'}$ that freely occur in $u|\varphi$, and $\overline{y_i}{:}\overline{s_i}$ is the set of variables different of $\overline{z}{:}\overline{s'}$ that freely occur in $v_i|\psi_i$.

The ML translation of $u|\varphi \sqsubseteq \{(v_i|\psi_i) \mid i \in I\}$ is

$$C^{\mathsf{ML}}_{\Sigma/B,E} \models \varphi^{\mathsf{ML}} \to \bigvee_{(i,\beta)\in\textsc{match}(u,\{v_i|i\in I\},Z)} \left(\psi_i^{\mathsf{ML}} \wedge \beta^{\mathsf{ML}}\right)$$

where $\textsc{match}(u, \{v_i \mid i \in I\}, Z)$ is a set of substitutions β defined over $var(v_i)\backslash Z$, and β^{ML} is the pattern describing the substitution β. We can prove now that the two ML patterns are equivalent.

Theorem 4.

$$C^{\mathsf{ML}}_{\Sigma/E,B} \models \left(\forall\overline{z}{:}\overline{s'}.\ \left(\exists\overline{x}{:}\overline{s}.\, u^{\mathsf{ML}} \wedge \varphi^{\mathsf{ML}} \subseteq \bigvee_{i\in I} \exists\overline{y_i}{:}\overline{s_i}.\, v_i^{\mathsf{ML}} \wedge \psi_i^{\mathsf{ML}}\right)\right) \leftrightarrow \left(\varphi^{\mathsf{ML}} \to \bigvee_{(i,\beta)\in\textsc{match}(u,\{v_i|i\in I\},Z)} (\psi_i^{\mathsf{ML}} \wedge \beta^{\mathsf{ML}})\right)$$

Explanation. The main idea is the same as Theorem 2 and to use the property (i, β); that is, $u =_{E_\Omega \cup B_\Omega} v_i\beta$ for any shared variables $z_i \in Z$, explaining the quantifier $\forall\overline{z}{:}\overline{s'}$ that appear on top of the LHS.

6 Conclusion

The paper establishes the exact relationship between two approaches that formalize state predicates of distributed systems: constrained constructor patterns [11] and matching logic [2]. The main conclusion from this comparison is that there is a mutual benefit. Matching logic can benefit from borrowing the computationally efficient reasoning modulo $E \cup B$. A first step is given in [1], but we think that there is more potential that can be exploited. On the other hand, the theory of constrained constructor patterns can get more expressiveness from its formalization as a fragment of the matching logic.

References

1. Arusoaie, A., Lucanu, D.: Unification in matching logic. In: Formal Methods—The Next 30 Years. Lecture Notes in Computer Science, vol. 11800, pp. 502–518. Porto, Portugal (2019). https://doi.org/10.1007/978-3-030-30942-8_30
2. Chen, X., Roşu, G.: Applicative matching logic. Tech. Rep. http://hdl.handle.net/2142/104616, University of Illinois at Urbana-Champaign (2019)
3. Chen, X., Rosu, G.: Matching μ-logic. In: Proceedings of the 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS 2019). pp. 1–13. IEEE, Vancouver, Canada (2019). https://doi.org/10.1109/LICS.2019.8785675
4. Escobar, S., Sasse, R., Meseguer, J.: Folding variant narrowing and optimal variant termination. J. Log. Algebr. Program. **81**(7-8), 898–928 (2012). https://doi.org/10.1016/j.jlap.2012.01.002
5. Futatsugi, K.: Fostering proof scores in CafeOBJ. In: Proceedings of the 12th International Conference on Formal Engineering Methods (ICFEM 2010). Lecture Notes in Computer Science, vol. 6447, pp. 1–20. Springer, Shanghai, China (2010). https://doi.org/10.1007/978-3-642-16901-4_1
6. Goguen, J.A., Thatcher, J.W., Wagner, E.G.: An initial algebra approach to the specification, correctness, and implementation of abstract data types. Tech. Rep. RC 6487, IBM Res. Rep. (1976), see also Current Trends in Programming Methodology, vol. 4: Data Structuring, R. T. Yeh, Ed. Englewood Cliffs, NJ: Prentice-Hall, 1978, pp. 80–149
7. Meseguer, J.: Variant-based satisfiability in initial algebras. Sci. Comput. Program. **154**, 3–41 (2018). https://doi.org/10.1016/j.scico.2017.09.001
8. Meseguer, J.: Generalized rewrite theories, coherence completion, and symbolic methods. J. Log. Algebr. Meth. Program. **110** (2020). https://doi.org/10.1016/j.jlamp.2019.100483
9. Meseguer, J.: Twenty years of rewriting logic. J. Log. Algebr. Program. **81**(7), 721–781 (2012). https://doi.org/10.1016/j.jlap.2012.06.003
10. Roşu, G.: Matching logic. Logical Methods in Computer Science **13**(4), 1–61 (2017)
11. Skeirik, S., Stefanescu, A., Meseguer, J.: A constructor-based reachability logic for rewrite theories. In: Proceedings of the 27th International Symposium on Logic-Based Program Synthesis and Transformation (LOPSTR 2017). Lecture Notes in Computer Science, vol. 10855, pp. 201–217. Springer, Namur, Belgium (2018). https://doi.org/10.1007/978-3-319-94460-9_12
12. Tarski, A.: A lattice-theoretical fixpoint theorem and its applications. Pacific Journal of Mathematics **5**(2), 285–309 (1955)