Overlap and Independence in Multiset Comprehension Patterns*

Iliano Cervesato[1] and Edmund S.L. Lam[1]

Carnegie Mellon University
sllam@qatar.cmu.edu and iliano@cmu.edu

Abstract

Rule-based programming, a model of computation by which rules modify a global state by concurrently rewriting disjoint portions of it, is gaining popularity as a simple, effective, and declarative means for implementing concurrent [1, 4] and distributed applications [5]. The state is often a multiset of first-order facts, with the rules specifying transformations over them. A rule consists of a head pattern which identifies a fragment of the state and a body which prescribes actions, often replacing this fragment with new facts. In this paper, we study rule interaction. Specifically, we identify conditions when two rule heads overlap by possibly targeting the same facts, which in many languages would prevent applying them concurrently — non-overlapping rules are independent and can always be executed in parallel without the risk of interference. A precise way of characterizing which rules are independent and which overlap is at the basis of numerous optimizations in the runtime of rule-based languages. It is also a useful debugging tool for programmers. We carry out this study in the context of Comingle [5], a rule-based language for programming mobile distributed applications. Head patterns in Comingle consist not only of partially instantiated facts, as found in most rule-based languages, but also of constraints and multiset comprehension patterns, which adds a challenging twist to our endeavor and interesting avenues of future work.

1 Multiset Comprehension Patterns

In this section, we formalize the syntax and matching semantics of a fragment of Comingle [5]. But first, some notation. We write $\overline{o}$ for a multiset of syntactic objects o. We denote the extension of a multiset $\overline{o}$ with an object o as "$\overline{o}, o$", with $\varnothing$ indicating the empty multiset. We also write "$\overline{o}_1, \overline{o}_2$" for the union of multisets $\overline{o}_1$ and $\overline{o}_2$. We write $\vec{o}$ for a tuple of o's and $[\vec{t}/\vec{x}]o$ for the simultaneous substitution within object o of all free occurrences of variable x_i in $\vec{x}$ with the corresponding term t_i in $\vec{t}$. A generic substitution is denoted θ. Substitution implicitly α-renames bound variables as needed to avoid capture.

Syntax. The top part of Figure 1 defines the abstract syntax of Comingle's head patterns. Computation in Comingle happens by rewriting *facts* F of the form $p(\vec{t})$ where p is a predicate symbol and $\vec{t}$ is a tuple of *terms*. The semantics of Comingle is largely agnostic to the specific language of terms as long as it is predicative — in this paper, we assume a first-order term language, later extended with primitive multisets.

Comingle *head patterns*, written H, have the form $\overline{E} \mid g$. We refer to $\overline{E}$ as the *template* of the pattern and to g as its *guard*. The template $\overline{E}$ consists of *atoms* F and of *comprehension patterns* [5] of the form $\wr F \mid g \int_{\vec{x} \to ts}$. An atom F is a fact $p(\vec{t})$ that may contain variables in the

*This paper was made possible by grants NPRP 4-1593-1-260 and NPRP 4-341-1-059, from the Qatar National Research Fund (a member of the Qatar Foundation). The statements made herein are solely the responsibility of the authors.

Syntax

Variables: x *Terms:* t *Guards:* g *Predicates symbols:* p

Facts $F ::= p(\vec{t})$
Expressions $E ::= F \mid \wr F \mid g \int_{\vec{x}\to t}$
Head Patterns $H ::= \overline{E} \mid g$

Matching: $\overline{E} \triangleq_{\mathtt{head}} St$, $E \triangleq_{\mathtt{head}} St$

$$\frac{\overline{E} \triangleq_{\mathtt{head}} St \quad E \triangleq_{\mathtt{head}} St'}{\overline{E}, E \triangleq_{\mathtt{head}} St, St'}\ (\mathsf{h}_{mset\text{-}1}) \qquad \frac{}{\varnothing \triangleq_{\mathtt{head}} \varnothing}\ (\mathsf{h}_{mset\text{-}2}) \qquad \frac{}{F \triangleq_{\mathtt{head}} F}\ (\mathsf{h}_{fact})$$

$$\frac{\models [\vec{t}/\vec{x}]g \quad \wr F \mid g \int_{\vec{x}\to \overline{ts}} \triangleq_{\mathtt{head}} St}{\wr F \mid g \int_{\vec{x}\to \vec{t},\overline{ts}} \triangleq_{\mathtt{head}} St, [\vec{t}/\vec{x}]F}\ (\mathsf{h}_{comp\text{-}1}) \qquad \frac{}{\wr F \mid g \int_{\vec{x}\to \varnothing} \triangleq_{\mathtt{head}} \varnothing}\ (\mathsf{h}_{comp\text{-}2})$$

Residual: $\overline{E} \triangleq^{\neg}_{\mathtt{head}} St$, $E \triangleq^{\neg}_{\mathtt{head}} St$

$$\frac{\overline{E} \triangleq^{\neg}_{\mathtt{head}} St \quad E \triangleq^{\neg}_{\mathtt{head}} St}{\overline{E}, E \triangleq^{\neg}_{\mathtt{head}} St}\ (\mathsf{h}^{\neg}_{mset\text{-}1}) \qquad \frac{}{\varnothing \triangleq^{\neg}_{\mathtt{head}} St}\ (\mathsf{h}^{\neg}_{mset\text{-}2}) \qquad \frac{}{F \triangleq^{\neg}_{\mathtt{head}} St}\ (\mathsf{h}^{\neg}_{fact})$$

$$\frac{F' \not\sqsubseteq \wr F \mid g \int_{\vec{x}\to ts} \quad \wr F \mid g \int_{\vec{x}\to ts} \triangleq^{\neg}_{\mathtt{head}} St}{\wr F \mid g \int_{\vec{x}\to ts} \triangleq^{\neg}_{\mathtt{head}} St, F'}\ (\mathsf{h}^{\neg}_{comp\text{-}1}) \qquad \frac{}{\wr F \mid g \int_{\vec{x}\to ts} \triangleq^{\neg}_{\mathtt{head}} \varnothing}\ (\mathsf{h}^{\neg}_{comp\text{-}2})$$

where $F' \sqsubseteq \wr F \mid g \int_{\vec{x}\to ts}$ iff $F' = \theta F$ and $\models \theta g$ for some $\theta = [\vec{t}/\vec{x}]$

Head match: $St \rightarrowtail^{H} St$

$$\frac{\theta\overline{E} \triangleq_{\mathtt{head}} St^{+} \quad \theta\overline{E} \triangleq^{\neg}_{\mathtt{head}} St^{-} \quad \models \theta g}{St^{+}, St^{-} \rightarrowtail^{\overline{E}|g} St^{-}}\ (\mathsf{match})$$

Figure 1: Matching a Rule Head

terms $\vec{t}$. Guards in patterns and comprehensions are Boolean-valued expressions constructed from terms and are used to constrain the values that the variables can assume. Just like for terms we keep guards abstract, writing $\models g$ to express that ground guard g is satisfiable. Two common types of guards are term equality $t = t'$ and multiset membership $t \in ts$. We drop the guard from patterns and comprehensions when it is the always-satisfiable constant $\top$. A comprehension pattern $\wr F \mid g \int_{\vec{x}\to ts}$ represents a multiset of all facts that match the atom F and satisfy guard g under the bindings of variables $\vec{x}$ that range over ts, a multiset of tuples called the *comprehension range*. We call F the *subject* of the comprehension. The scope of $\vec{x}$ is the atom F and the guard g. We implicitly α-rename bound variables to avoid capture.

Given two head patterns $H_1 = \overline{E}_1 \mid g_1$ and $H_2 = \overline{E}_2 \mid g_2$ without (free) variables in common, we define the *parallel composition* of H_1 and H_2, written $H_1 \parallel H_2$, as the head pattern $\overline{E}_1, \overline{E}_2 \mid g_1 \wedge g_2$.

Matching. During computation, Comingle head patterns are matched against a *state*, written St, which is a multiset of ground facts $p(\vec{t})$. We describe the successful match of a head pattern $H = \overline{E} \mid g$ against a state St by means of the judgment $St \rightarrowtail^{H} St'$, with St' collecting the portion of the state St that was not matched by the pattern.

Let $\overline{E}$ be a ground template — we will deal with the more general case momentarily. Intuitively, matching $\overline{E}$ against a state St means splitting St into two parts, St^+ and St^-, and checking that $\overline{E}$ matches St^+ completely. The latter is achieved by the judgment $\overline{E} \triangleq^{}_{\mathtt{head}} St^+$ defined in the upper central part of Figure 1. Rules $\mathsf{h}_{mset\text{-}*}$ partition St^+ into fragments to be matched by each atom in $\overline{E}$: plain facts F must occur identically (rule h_{fact}) while for comprehension atoms $\wr F \mid g \int_{\vec{x} \to ts}$ the state fragment must contain a distinct instance of F for every element of the comprehension range ts that satisfies the comprehension guard g (rules $\mathsf{h}_{comp\text{-}*}$).

In Comingle, comprehension patterns must match *maximal* fragments of the state. Therefore, no comprehension pattern in $\overline{E}$ should match any fact in St^-. This check is captured by the judgment $\overline{E} \triangleq^{\neg}_{\mathtt{head}} St^-$ in the lower central part of Figure 1: it holds unless St^- contains a fact F' and $\overline{E}$ a pattern $\wr F \mid g \int_{\vec{x} \to ts}$ for which there exists a substitution θ such that $F' = \theta F$ and guard θg is satisfied. Rules $\mathsf{h}^{\neg}_{mset\text{-}*}$ test each individual atom and rule $\mathsf{h}^{\neg}_{fact}$ ignore facts. Rules $\mathsf{h}^{\neg}_{comp\text{-}*}$ deal with comprehensions $\wr F \mid g \int_{\vec{x} \to ts}$: they check that no fact in St^- matches any instance of F while satisfying g.

Rule (match) in Figure 1 describes head matching in Comingle. This involves identifying a ground instance of the pattern obtained by means of a substitution θ. The instantiated guard must be satisfiable ($\models \theta g$) and we must be able to partition the state into two parts St^+ and St^-. The instance of the template must match St^+ ($\theta\overline{E} \triangleq^{}_{\mathtt{head}} St^+$), while the remaining fragment St^- must not match any comprehension in it ($\theta\overline{E} \triangleq^{\neg}_{\mathtt{head}} St^-$). In Comingle, rules also contain a body B, another template, and the body instance θB is then unfolded into ground facts St_B which replaces St^+ in the state. A formal description and examples of use can be found in [5].

2 Overlapping Patterns

Two head patterns $H_1 = \overline{E}_1 \mid g_1$ and $H_2 = \overline{E}_2 \mid g_2$ without variables in common *overlap* if the former consumes facts that may prevent the latter from being applicable. This happens if there is a state St such that $St \overset{H_1}{\rightarrowtail} St_1$ and $St \overset{H_2}{\rightarrowtail} St_2$ for some St_1 and St_2, but there is no St' such that $St \overset{H_1 \| H_2}{\rightarrowtail} St'$. Head patterns H_1 and H_2 (without common variables) are *independent* if they do not overlap. For example, $H_1 = p(a, X), q(X)$ and $H_2 = p(Y, Y), r(Z)$ overlap, for instance in state $p(a, a), q(a), r(b)$, while H_1 and $H_2' = p(b, Y), r(Z)$ are independent.

The above definitions do not provide any guidance as to how to determine whether two head patterns overlap (or are independent). In the rest of this section, we devise effective conditions to make this determination in some common cases. We proceed incrementally, starting from simplified patterns and progressing towards the general case, which remains unsolved in its full generality.

Multisets We start with head patterns of the form $H = \overline{F}$, featuring an empty guard and no comprehensions. Then, H_1 and H_2 overlap if and only if one contains a fact unifiable in the other template. In symbols, if $H_1 = p(\vec{t_1}), \overline{F}'_1$ and $H_2 = p(\vec{t_2}), \overline{F}'_2$, and moreover there is a substitution θ such that $\theta\vec{t_1} = \theta\vec{t_2}$.

This was the situation in our earlier example. Note that the pair of facts $p(\vec{t_1})$ and $p(\vec{t_2})$ may not be unique in H_1 and H_2. While this makes determining independence a combinatorial problem, head patterns in typical rule-based language contain just a handful of atoms.

In a traditional first-order language, the substitution θ can be conveniently chosen to be the most general unifier (mgu) of $p(\vec{t_1})$ and $p(\vec{t_2})$, but no such concept exists for more complex term languages, for example languages featuring term-level multisets [2] or higher-order terms.

Guards We next add guards, but still no comprehensions. Our head patterns then assume the form $H = \overline{F} \mid g$. Now, head patterns $H_1 = \overline{F}_1 \mid g_1$ and $H_2 = \overline{F}_2 \mid g_2$ without common variables overlap if their template contains a unifiable fact and both guards hold for the witness substitution. In symbols, H_1 and H_2 overlap if $H_1 = p(\vec{t_1}), \overline{F}'_1 \mid g_1$ and $H_2 = p(\vec{t_2}), \overline{F}'_2 \mid g_2$, and moreover there is a substitution θ such that $\theta\vec{t_1} = \theta\vec{t_2}$ and $\models \theta g_1$ and $\models \theta g_2$. For example, in a term language over integers, the patterns $H_1 = p(X) \mid X > 3$ and $H_2 = p(Y) \mid Y < 10$ overlap for the state $p(7)$ for instance, while H_1 and $H'_2 = p(Y) \mid Y < 3$ are independent.

An effective approach to finding overlaps, for instance in this last example, is to compute unifiers θ between candidate facts $p(\vec{t_1})$ and $p(\vec{t_2})$, and then pass the partially instantiated guards θg_1 and θg_2 to an SMT solver such as Z3 [3] to ascertain satisfiability.

Multisets of facts, possibly constrained through guards, are found in a majority of rule-based languages, for example CHR [4]. This makes the above approach widely applicable. Comprehensions in Comingle complicate this picture, however, as we explore next.

Open-ended Comprehensions We will now allow multiset comprehensions among the atoms of a head pattern, but impose the restriction that their comprehension range does not appear anywhere else in the template, in particular not in guards. Therefore, we shall accept the pattern $p(X), \wr p(x) \mid x > 0 \int_{x \to Xs}$, but not $p(X), \wr p(x) \mid x > 0 \int_{x \to Xs} \mid |Xs| = 0$ where $|M|$ returns the number of elements in multiset M.

Given two head patterns $H_1 = \overline{E}_1 \mid g_1$ and $H_2 = \overline{E}_2 \mid g_2$ with the above characteristics (and without common variables), it may come as a surprise that the procedure we just saw in the absence of comprehensions still determines overlaps. The reason is that an open-ended comprehension — one whose range we do not constrain through guards or other mechanisms — can never fail: at worst it will return an empty multiset through its range.

As a simple example, consider the templates $H_1 = p(X)$, which consists of a single fact, and $H_2 = \wr p(x) \int_{x \to Xs}$. Template H_1 succeeds in any state that contains at least one fact headed by the predicate symbol p. For example, $p(a) \rightarrowtail^{H_1} \varnothing$. Instead, H_2 always succeeds: indeed $p(a) \rightarrowtail^{H_2} \varnothing$ and $\varnothing \rightarrowtail^{H_2} \varnothing$. Putting them together, we have that $p(a) \rightarrowtail^{H_1 \| H_2} \varnothing$. Note however that H_2 matches different portions of this state when applied in isolation (it matches $p(a)$) and when composed with H_1 (it matches $\varnothing$), a behavior that does not manifest in the absence of comprehensions. This weak form of interaction, although not an overlap according to our definition, has implications when implementing Comingle [5].

General Comprehensions Constraints over comprehension ranges, whether in guards or through shared variables, significantly complicate determining whether head patterns are independent. Consider the following pair of patterns:

A) $H_1 = \wr p(x) \int_{x \to Xs}, q(Y) \mid Y \in Xs$ *and* $H_2 = p(Z)$

H_1 and H_2 overlap in states such as $p(a), q(a)$, in which each succeeds separately, but fail when composed as H_2 grabs $p(a)$ forcing the comprehension range Xs to be instantiated to the empty multiset, thereby failing the guard.

Membership constraints are not the only guards that lead to overlap in the presence of comprehensions. Consider the following cardinality constraint:

B) $H_1 = \wr p(x) \int_{x \to Xs} \mid |Xs| > 0$ *and* $H_2 = p(Z)$

Here, the state $p(a)$ witnesses the overlap as the combined head patterns fails since the range Xs will be instantiated to the empty multiset, which violates the constraint.

Explicit guards are not even needed to cause overlap. Consider these head patterns:

C) $H_1 = \wr p(x) \int_{x \to Xs}, \wr q(y) \int_{y \to Xs},$ *and* $H_2 = p(Z)$

and the state $p(a), q(a)$. On it, they succeed independently, but fail when composed.

In each of these examples, comprehension ranges appear more than once in one of the head patterns. This is not a sufficient condition, however, for the presence of overlap. The following three examples use the same operations on comprehension ranges seen earlier (membership test, cardinality constraints, and variable sharing) and yet the patterns appearing on each row are independent.

D) $H_1 = \wr p(x) \mid x < 3 \int_{x \to Xs}, q(Y) \mid Y \in Xs$ *and* $H_2 = p(Z) \mid Z > 5$
E) $H_1 = \wr p(x) \int_{x \to Xs} \mid |Xs| \leq 8$ *and* $H_2 = p(Z)$
F) $H_1 = \wr p(x) \int_{x \to Xs}, \wr q(y) \mid y \in Xs \int_{y \to Ys},$ *and* $H_2 = p(Z)$

Situation (D) differs from (A) in that the two bounds are such that no fact $p(n)$ can match both patterns. Example (E) imposes an upper bound on the comprehension range that matches H_1, while (B) forced a lower bound (that H_2 encroaches upon). Finally, the second occurrence of Xs in (F) filters out values for Ys rather than requiring that some terms be present.

Negation-as-absence is a common variant of (B) whose patterns are independent:

G) $H_1 = \wr p(x) \int_{x \to Xs} \mid |Xs| = 0$ *and* $H_2 = p(Z)$

Here, H_1 asks for the state not to contain predicates of the form $p(n)$ (alternatively, this constraint could be written as the guard $Xs = \varnothing$). Here H_1 and H_2 are independent as there is no state in which both can succeed.

These examples illustrate the rich space of behaviors that constrained comprehension patterns opens to. A general algorithmic approach to determining when such patterns overlap (or are independent) has not been found yet, however. This will be the focus of future research.

3 Conclusions

In this paper, we have explored overlap and independence, two forms of head pattern interaction in rule-based languages. We specifically focused on the head constructions of Comingle [5], a language for programming mobile distributed applications with roots in advanced forms of multiset rewriting. While the form of head patterns found in traditional languages lend itself to a simple procedural characterization of these properties, a similar recipe for the full range of constructs found in Comingle has so far proved elusive. Specifically, the interaction of constraints (or guards) and multiset comprehensions was shown to be diverse and varied. In future work, we intend to study this interaction at a deeper level, and to devise effective algorithms for overlap an independence in full Comingle.

References

[1] M. P. Ashley-Rollman et al. A Language for Large Ensembles of Independently Executing Nodes. In *ICLP'09*, pages 265–280. Springer LNCS 5649.

[2] I. Cervesato. Solution Count for Multiset Unification with Trailing Multiset Variables. In *UNIF'02*.

[3] L. De Moura and N. Bjørner. Z3: An Efficient SMT Solver. In *TACAS'08*. Springer-Verlag.

[4] T. Frühwirth. *Constraint Handling Rules*. Cambridge University Press, 2009.

[5] E. S. Lam, I. Cervesato, and N. F. Haque. Comingle: Distributed Logic Programming for Decentralized Mobile Ensembles. In *COORDINATION'15*. Springer LNCS 9037.