

# On regular languages over power sets

*Tim Fernando*  
Trinity College Dublin, Ireland

## ABSTRACT

The power set of a finite set is used as the alphabet of a string interpreting a sentence of Monadic Second-Order Logic so that the string can be reduced (in straightforward ways) to the symbols occurring in the sentence. Simple extensions to regular expressions are described matching the succinctness of Monadic Second-Order Logic. A link to Goguen and Burstall's notion of an institution is forged, and applied to conceptions within natural language semantics of time based on change. Various reductions of strings are described, along which models can be miniaturized as strings.

*Keywords:*  
*regular language,*  
*power set, MSO,*  
*institution*

1

## INTRODUCTION

Working with more than one alphabet is established practice in finite-state language processing, attested by the popularity of auxiliary symbols (e.g., Kaplan and Kay 1994; Beesley and Karttunen 2003; Yli-Jyrä and Koskenniemi 2004; Hulden 2009). To avoid choosing an alphabet prematurely, implementations commonly treat the alphabet  $\Sigma$  as a dynamic entity that is left underspecified before the finite automaton is constructed in full.<sup>1</sup> Fixing  $\Sigma$  is not always necessary to determine the language denoted by an expression. This is the case with regular expressions; the expression  $\emptyset$  denotes the empty set for any alphabet  $\Sigma$ , and the expression  $ab$  denotes the singleton set  $\{ab\}$  for any alphabet  $\Sigma \supseteq \{a, b\}$ . Beyond regular expressions, however, there are expressions that denote different languages given different choices of

---

<sup>1</sup> I am indebted to an anonymous referee for raising this point.

the alphabet  $\Sigma$ . Consider  $ab$ 's negation (or complement)  $\overline{ab}$ , which denotes a language

$$\Sigma^* - \{ab\} = \{s \in \Sigma^* \mid s \neq ab\}$$

that is regular iff  $\Sigma$  is a finite set. To delay fixing  $\Sigma$  to some finite set is to leave open just what the denotation  $\Sigma^* - \{ab\}$  of  $\overline{ab}$  is. Relative to an alphabet  $\Sigma$ , a symbol  $c$ , understood as a string of length one, belongs to that denotation if and only if  $c \in \Sigma$ . ( $\Sigma$  contains *any* symbol, including  $c$ , in the open alphabet system implemented in Beesley and Karttunen 2003.)

Apart from negations, there are many more extensions to regular expressions describing denotations that vary with the choice of alphabet. Consider the sentences of Monadic Second-Order Logic (MSO), which, under a model-theoretic interpretation against strings, capture the regular languages, by a fundamental theorem due independently to Büchi, Elgot and Trakhtenbrot (e.g., Theorem 3.2.11, page 145 in Grädel 2007; Theorem 7.21, page 124 in Libkin 2010). Leaving the precise details of MSO for Section 2 below, suffice it to say (for now) that occurrences of a string symbol  $a$  are encoded in a unary predicate symbol  $P_a$  for an MSO-sentence such as  $\forall x P_a(x)$ , saying  $a$  occurs at every string position (satisfied by the string  $aaa$  but not by the string  $ab$  unless  $a = b$ ). We can check if a string over any finite alphabet  $\Sigma$  (hereafter, a  $\Sigma$ -string) satisfies an MSO-sentence  $\varphi$ , but the computation gets costlier as  $\Sigma$  is enlarged. Surely, however, only the symbols that appear in  $\varphi$  matter in satisfying  $\varphi$  or its negation? To investigate this question, let the *vocabulary* of  $\varphi$  be the set

$$\text{voc}(\varphi) := \{a \mid P_a \text{ occurs in } \varphi\}$$

of subscripts of unary predicate symbols appearing in  $\varphi$ . (For example,  $\forall x P_a(x)$ 's vocabulary  $\text{voc}(\forall x P_a(x))$  is  $\{a\}$ .) Now the question is: can we not reduce satisfaction of  $\varphi$  by a  $\Sigma$ -string to satisfaction of  $\varphi$  by a  $\text{voc}(\varphi)$ -string? A simple form such a reduction might take is a function  $f : \Sigma^* \rightarrow \text{voc}(\varphi)^*$  mapping a  $\Sigma$ -string  $s$  to a  $\text{voc}(\varphi)$ -string  $f(s)$  that satisfies  $\varphi$  if and only if  $s$  does

$$s \models \varphi \iff f(s) \models \varphi. \tag{1}$$

Unfortunately, already for  $\varphi$  equal to  $\forall x P_a(x)$  and  $\Sigma$  to  $\{a, b\}$ , it is clear no such function  $f$  can exist; the lefthand side of (1) fails for  $s = ab$ ,

whereas the righthand side cannot:  $a^n \models \forall x P_a(x)$  for all integers  $n \geq 0$ . Evidently,  $\text{voc}(\varphi)^*$  is too small to provide the variation necessary for the reduction (1). Enter  $(2^{\text{voc}(\varphi)})^*$ , where the power set  $2^A$  of a set  $A$  is the set of all subsets of  $A$ . For any MSO-sentence  $\varphi$  and string  $s = \alpha_1 \cdots \alpha_n$  of sets  $\alpha_i$ , we intersect  $s$  componentwise with  $\text{voc}(\varphi)$  for the  $2^{\text{voc}(\varphi)}$ -string

$$\rho_{\text{voc}(\varphi)}(\alpha_1 \cdots \alpha_n) := (\alpha_1 \cap \text{voc}(\varphi)) \cdots (\alpha_n \cap \text{voc}(\varphi)).$$

Then for any finite set  $\Sigma$ , we let  $\text{MSO}_\Sigma$  be the set of MSO-sentences with vocabulary contained in  $\Sigma$

$$\text{MSO}_\Sigma := \{\varphi \mid \varphi \text{ is an MSO-sentence and } \text{voc}(\varphi) \subseteq \Sigma\}$$

and interpret sentences  $\varphi \in \text{MSO}_\Sigma$  relative to  $2^\Sigma$ -strings  $s$  using a binary relation  $\models_\Sigma$  (defined in Section 2) such that

$$s \models_\Sigma \varphi \iff \rho_{\text{voc}(\varphi)}(s) \models_{\text{voc}(\varphi)} \varphi. \quad (2)$$

The subscripts  $\Sigma$  and  $\text{voc}(\varphi)$  on  $\models$  in the lefthand and righthand sides of (2) track the reduction effected by  $\rho_{\text{voc}(\varphi)}$  but could otherwise be dropped, had we not already used  $\models$  for the satisfaction relation mentioned in (1). Fixing  $\varphi$ 's denotation relative to  $\Sigma$  as the set

$$\mathcal{L}_\Sigma(\varphi) = \{s \in (2^\Sigma)^* \mid s \models_\Sigma \varphi\}$$

of  $2^\Sigma$ -strings that  $\models_\Sigma$ -satisfy  $\varphi$ , we may conclude from (2) that

(†) whatever finite set  $\Sigma$  we use to fix the denotation of  $\varphi$ , it all comes down to  $\text{voc}(\varphi)$ .

Our argument for (†) via (2) rests on modifying MSO-satisfaction  $\models$  as it is usually presented over  $\Sigma$ -strings (e.g., Libkin 2010) to one  $\models_\Sigma$  over  $2^\Sigma$ -strings. Without appealing to (†), which might be made precise some other way, we motivate the step from  $\Sigma$  to  $2^\Sigma$  in our presentation of MSO-models in Section 2, showing, among other things, how that step clarifies what predication and quantification amount to on strings (essentially, preimages and images under  $\rho_{\text{voc}(\varphi)}$ ).

Beyond MSO, the reduction (2) is an instance of a general condition built into an abstract model-theoretic approach to specification and programming based on *institutions* (Goguen and Burstall 1992). We adopt this perspective to generalize (2) in Section 3 from  $\rho_{\text{voc}(\varphi)}$

to functions on strings of sets, manipulating not only the vocabulary but also the length of strings (yielding, at the limit, infinite strings). At the center of this perspective are declarative methods for specifying sets of strings over different alphabets. We focus on methods, including but not limited to MSO, where the alphabets are power sets  $2^\Sigma$  of finite sets  $\Sigma$ .

A multiplicity of such alphabets is useful in the semantics of tense and aspect to measure time at different bounded granularities  $\Sigma$ , tracking finite sets of unary predicates named in  $\Sigma$ . Consider, for instance, Reichenbach's well-known account based on a reference time R, an event time E and a speech time S (Reichenbach 1947). We can picture various temporal relations between an event and a speech as strings of boxes that may or may not contain E or S. For example, the string  $\boxed{E} \boxed{S}$  portrays S after E (much like a film or comic strip), which we can verbalize using the simple past or the present perfect, illustrated by (a) and (b) respectively (where the event with time E is Ed's exhalation).

- (a) Ed exhaled.
- (b) Ed has exhaled.

To represent the difference between (a) and (b), we bring the reference time R into the picture, expanding  $\Sigma = \{E, S\}$  to  $\Sigma = \{R, E, S\}$  with

- (‡)  $\boxed{R, E} \boxed{S}$  for the simple past (a), and  
 $\boxed{E} \boxed{R, S}$  for the present perfect (b),

where a box is drawn instead of the usual curly braces  $\{, \}$  for a set construed as a symbol in a string of sets. The difference brought out in (‡) carries significance for anaphora (e.g., Kamp and Reyle 1993, where R is split many ways) and event structure (including an event's consequent state, in Moens and Steedman 1988). Both strings in (‡) can be constructed from simpler strings representing a Reichenbachian analysis of

- (i) tense as a relation between R and S, with  $\Sigma = \{R, S\}$  and

$\boxed{R} \boxed{S}$  for the past (a), and  $\boxed{R, S}$  for the present (b)

and

- (ii) aspect as a relation between R and E, with  $\Sigma = \{R, E\}$  and

$\boxed{R, E}$  for the simple (a), and  $\boxed{E} \boxed{R}$  for the perfect (b).

Complicating the picture, there are finer analyses of E into aspectual classes going back to Aristotle, Ryle and Vendler (e.g., Dowty 1979) that call for an expansion of  $\Sigma = \{R,E,S\}$  to refine the level of granularity (Fernando 2014). A wide ranging hypothesis that the semantics of tense and aspect is finite-state is defended in Fernando (2015), deploying regular languages over power sets, of the kind described below.

Applications to temporal semantics aside, the reader expecting a discussion of finite-state methods applied to phonology, morphology and/or syntax should be warned that such a discussion has been left for someone competent in such matters to take up elsewhere. The present paper claims neither to be the first nor the last word on regular languages over power sets. Its aim simply is to show how to get a handle on the dependence of certain declarative methods on the choice of a finite set  $\Sigma$  of symbols by stepping up to the power set  $2^\Sigma$  of  $\Sigma$  and reducing a string through some function  $\rho_{\text{voc}(\varphi)}$  or other. MSO provides an obvious point of departure (Section 2), leading to further declarative methods (Section 3).

## 2 MSO AND RELATED EXTENSIONS OF REGULAR EXPRESSIONS

It is convenient to fix an infinite set  $Z$  of symbols  $a$  that can appear in unary predicate symbols  $P_a$ , from which sentences of MSO are formed. An MSO-sentence  $\varphi$  can have within it only finitely many unary predicate symbols  $P_a$ , allowing us to break MSO up into fragments given by finite subsets  $\Sigma$  of  $Z$  (no single one of which encompasses all of MSO). In addition to the  $P_a$ 's, we assume a binary relation symbol  $S$  (for successors), from which we can form, for example, the MSO-sentence

$$\forall x(P_a(x) \supset \exists y(S(x, y) \wedge P_b(y)))$$

saying that every  $a$ -occurrence is succeeded by a  $b$ -occurrence. Formal definitions are given in Subsection 2.1 of a satisfaction relation  $\models_\Sigma$  between (finite)  $\text{MSO}_\Sigma$ -models and  $\text{MSO}_\Sigma$ -sentences, built from  $\text{MSO}_\Sigma$ -formulas with free variables analyzed by suitable expansions of  $\Sigma$ . These expansions are undone by functions  $\rho_\Sigma$  on strings that arguably provide the key to predication and quantification over strings. Indeed, the  $\rho_\Sigma$ 's pave an easy route to the regularity of MSO, as we show in Subsection 2.2. The functions can be tweaked for useful extensions

in Subsection 2.3 of regular expressions, and declarative methods in Section 3 that, like our presentation of MSO via  $\models_\Sigma$ , meet abstract requirements from Goguen and Burstall (1992).

In what follows, we write  $\text{Fin}(A)$  for the set of finite subsets of a set  $A$ . Often but not always,  $A$  is  $Z$ .

### 2.1 MSO-models, formulas and satisfaction

We restrict our attention to finite models, defining for any integer  $n \geq 0$ ,  $[n]$  to be the set of integers from 1 to  $n$ ,

$$[n] := \{1, 2, \dots, n\}$$

and  $S_n$  to be the successor (next) relation from  $i$  to  $i + 1$  for  $i \in [n - 1]$

$$S_n := \{(1, 2), (2, 3), \dots, (n - 1, n)\}.$$

Given  $\Sigma \in \text{Fin}(Z)$ , let us agree that an  $\text{MSO}_\Sigma$ -model  $M$  is a tuple

$$\langle [n], S_n, \{\llbracket P_a \rrbracket\}_{a \in \Sigma} \rangle$$

for some integer  $n \geq 0$ ,<sup>2</sup> such that for each  $a \in \Sigma$ ,  $\llbracket P_a \rrbracket$  is a subset of  $[n]$  interpreting the unary relation symbol  $P_a$ . For  $A \subseteq \Sigma$ , the  $A$ -reduct of  $M$  is the  $\text{MSO}_A$ -model  $\langle [n], S_n, \{\llbracket P_a \rrbracket\}_{a \in A} \rangle$ , keeping only the interpretations  $\llbracket P_a \rrbracket$  for  $a \in A$ .

There is a simple bijection  $\text{str}$  from  $\text{MSO}_\Sigma$ -models to  $2^\Sigma$ -strings, picturing an  $\text{MSO}_\Sigma$ -model  $M = \langle [n], S_n, \{\llbracket P_a \rrbracket\}_{a \in \Sigma} \rangle$  as the  $2^\Sigma$ -string  $\text{str}(M) = \alpha_1 \cdots \alpha_n$  with

$$\alpha_i := \{a \in \Sigma \mid i \in \llbracket P_a \rrbracket\} \quad (\text{for } i \in [n]),$$

which inverts to

$$\llbracket P_a \rrbracket = \{i \in [n] \mid a \in \alpha_i\} \quad (\text{for } a \in \Sigma).$$

For example, if  $\Sigma = \{a, b\}$  and  $M$  is  $\langle [4], S_4, \{\llbracket P_c \rrbracket\}_{c \in \Sigma} \rangle$  with  $\llbracket P_a \rrbracket = \{1, 2\}$  and  $\llbracket P_b \rrbracket = \{1, 3\}$ , then

$$\text{str}(M) = \boxed{a, b} \boxed{a} \boxed{b}$$

(with  $\alpha_i$  boxed, as noted in the introduction, to mark them out as string symbols). Strings of boxes with exactly one  $a \in \Sigma$  embed  $\Sigma^*$  into  $(2^\Sigma)^*$ ; let  $\iota : \Sigma^* \rightarrow (2^\Sigma)^*$  map  $a_1 \cdots a_n \in \Sigma^n$  to

$$\iota(a_1 \cdots a_n) := \boxed{a_1} \cdots \boxed{a_n}.$$

---

<sup>2</sup>We follow Libkin (2010) in allowing a model to have an empty domain/universe.

An advantage in working with  $(2^\Sigma)^*$  rather than  $\Sigma^*$  is that we can intersect a  $2^\Sigma$ -string  $\alpha_1 \cdots \alpha_n$  componentwise with any subset  $A$  of  $\Sigma$  for the  $2^A$ -string

$$\rho_A(\alpha_1 \cdots \alpha_n) := (\alpha_1 \cap A) \cdots (\alpha_n \cap A)$$

(generalizing  $\rho_{\text{voc}(\varphi)}$  in the introduction). The  $A$ -reduct of the  $\text{MSO}_\Sigma$ -model given by the string  $\alpha_1 \cdots \alpha_n$  is represented by  $\rho_A(\alpha_1 \cdots \alpha_n)$ ; i.e., for any  $\text{MSO}_\Sigma$ -model  $M$  and  $\text{MSO}_A$ -model  $M'$ ,

$$\rho_A(\text{str}(M)) = \text{str}(M') \iff M' \text{ is the } A\text{-reduct of } M.$$

The difference between an  $\text{MSO}_\Sigma$ -model  $M$  and the string  $\text{str}(M)$  is so slight that we can confuse  $M$  harmlessly with  $\text{str}(M)$  and refer to a  $2^\Sigma$ -string as an  $\text{MSO}_\Sigma$ -model.

To form  $\text{MSO}$ -formulas with free variables, let us fix an infinite set  $\text{Var}$  disjoint from  $Z$ ,  $\text{Var} \cap Z = \emptyset$ , treating each  $x \in \text{Var}$  as a first-order variable. Given finite subsets  $\Sigma$  of  $Z$  and  $V$  of  $\text{Var}$ , we define a  $\text{MSO}_{\Sigma,V}$ -model to be a  $2^{\Sigma \cup V}$ -string in which each  $x \in V$  occurs exactly once, and collect these in the set  $\text{Mod}_V(\Sigma)$

$$\text{Mod}_V(\Sigma) := \left\{ s \in (2^{\Sigma \cup V})^* \mid (\forall x \in V) \rho_{\{x\}}(s) \in \boxed{x}^* \right\}.$$

We define the set  $\text{MSO}_{\Sigma,V}$  of  $\text{MSO}_\Sigma$ -formulas  $\varphi$  with free variables in  $V$  by induction, alongside sets  $\mathcal{L}_{\Sigma,V}(\varphi)$  of strings in  $\text{Mod}_V(\Sigma)$  that satisfy  $\varphi$ , determining a satisfaction relation

$$\models_{\Sigma,V} \subseteq \text{Mod}_V(\Sigma) \times \text{MSO}_{\Sigma,V}$$

between strings  $s \in \text{Mod}_V(\Sigma)$  and formulas  $\varphi \in \text{MSO}_{\Sigma,V}$  according to

$$s \models_{\Sigma,V} \varphi \iff s \in \mathcal{L}_{\Sigma,V}(\varphi).$$

The inductive definition consists of six clauses.

- (a) If  $\{x, y\} \subseteq V$ , then  $x = y$  and  $S(x, y)$  are in  $\text{MSO}_{\Sigma,V}$ , with  $x = y$  satisfied by strings in  $\text{Mod}_V(\Sigma)$  where  $x$  and  $y$  occur in the same position

$$\mathcal{L}_{\Sigma,V}(x = y) := \left\{ s \in \text{Mod}_V(\Sigma) \mid \rho_{\{x,y\}}(s) \in \boxed{x, y}^* \right\}$$

and  $S(x, y)$  satisfied by strings in  $\text{Mod}_V(\Sigma)$  where  $x$  occurs immediately before  $y$

$$\mathcal{L}_{\Sigma,V}(S(x, y)) := \left\{ s \in \text{Mod}_V(\Sigma) \mid \rho_{\{x,y\}}(s) \in \boxed{x} \boxed{y}^* \right\}.$$

- (b) If  $a \in \Sigma$  and  $x \in V$ , then  $P_a(x)$  is in  $MSO_{\Sigma,V}$  and is satisfied by strings in  $Mod_V(\Sigma)$  where the occurrence of  $x$  coincides with one of  $a$

$$\mathcal{L}_{\Sigma,V}(P_a(x)) := \{s \in Mod_V(\Sigma) \mid \rho_{\{a,x\}}(s) \in \{\square, \boxed{a}\}^* \boxed{a,x} \{\square, \boxed{a}\}^*\}.$$

- (c) If  $\varphi \in MSO_{\Sigma,V}$  then so is  $\neg\varphi$  with  $\neg\varphi$  satisfied by strings in  $Mod_V(\Sigma)$  that do not satisfy  $\varphi$

$$\mathcal{L}_{\Sigma,V}(\neg\varphi) := Mod_V(\Sigma) - \mathcal{L}_{\Sigma,V}(\varphi).$$

- (d) If  $\varphi$  and  $\psi$  are in  $MSO_{\Sigma,V}$  then so is  $\varphi \wedge \psi$  with  $\varphi \wedge \psi$  satisfied by strings in  $Mod_V(\Sigma)$  that satisfy both  $\varphi$  and  $\psi$

$$\mathcal{L}_{\Sigma,V}(\varphi \wedge \psi) := \mathcal{L}_{\Sigma,V}(\varphi) \cap \mathcal{L}_{\Sigma,V}(\psi).$$

For quantification, we must be careful that a variable can be reused, as in

$$P_b(x) \wedge \exists x P_a(x),$$

which is equivalent to  $P_b(x) \wedge \exists y P_a(y)$  since  $\exists x P_a(x)$  and  $\exists y P_a(y)$  are.<sup>3</sup> To cater for reuse of  $q \in Var \cup Z$ , we define an equivalence relation  $\sim_q$  between strings  $s$  and  $s'$  of sets that differ at most on  $q$ , putting

$$s' \sim_q s \iff \hat{\rho}_q(s') = \hat{\rho}_q(s),$$

where the function  $\hat{\rho}_q$  removes  $q$  from a string  $\alpha_1 \cdots \alpha_n$  of sets

$$\hat{\rho}_q(\alpha_1 \cdots \alpha_n) := (\alpha_1 - \{q\}) \cdots (\alpha_n - \{q\}).$$

We can now state the last two clauses of our inductive definition of  $MSO_{\Sigma,V}$  and  $\mathcal{L}_{\Sigma,V}(\varphi)$ .

- (e) If  $\varphi \in MSO_{\Sigma,V \cup \{x\}}$  then  $\exists x \varphi$  is in  $MSO_{\Sigma,V}$  with  $\exists x \varphi$  satisfied by strings in  $Mod_V(\Sigma)$  that are  $\sim_x$ -equivalent to strings in  $Mod_{V \cup \{x\}}(\Sigma)$  satisfying  $\varphi$ :

$$\mathcal{L}_{\Sigma,V}(\exists x \varphi) := \{s \in Mod_V(\Sigma) \mid (\exists s' \in \mathcal{L}_{\Sigma,V \cup \{x\}}(\varphi)) s' \sim_x s\},$$

which simplifies in case  $x$  is not reused

$$\mathcal{L}_{\Sigma,V}(\exists x \varphi) = \frac{\{\rho_{\Sigma \cup V}(s) \mid s \in \mathcal{L}_{\Sigma,V \cup \{x\}}(\varphi)\}}{\quad} \quad \text{if } x \notin V.$$

<sup>3</sup>We can always avoid reuse in finite formulas, working with finitely many variables.



- (f) If  $\varphi \in \text{MSO}_{\Sigma \cup \{a\}, V}$  then  $\exists P_a \varphi$  is in  $\text{MSO}_{\Sigma, V}$  with  $\exists P_a \varphi$  satisfied by strings in  $\text{Mod}_V(\Sigma)$  that are  $\sim_a$ -equivalent to strings in  $\text{Mod}_V(\Sigma \cup \{a\})$  satisfying  $\varphi$  :

$$\mathcal{L}_{\Sigma, V}(\exists P_a \varphi) := \{s \in \text{Mod}_V(\Sigma) \mid (\exists s' \in \mathcal{L}_{\Sigma \cup \{a\}, V}(\varphi)) s' \sim_a s\},$$

which simplifies in case  $P_a$  is not reused

$$\mathcal{L}_{\Sigma, V}(\exists P_a \varphi) = \{\rho_{\Sigma \cup V}(s) \mid s \in \mathcal{L}_{\Sigma \cup \{a\}, V}(\varphi)\} \quad \text{if } a \notin \Sigma.$$

We adopt the usual abbreviations:  $\varphi \vee \psi$  for  $\neg(\neg\varphi \wedge \neg\psi)$ ,  $\forall x\varphi$  for  $\neg\exists x\neg\varphi$ , etc. Also, we render second-order quantification  $\exists P_a$  as  $\exists X$ , writing  $\exists X\varphi$  for  $\exists P_a \varphi_a^X$  where  $a$  does not occur in  $\varphi$ , and  $\varphi_a^X$  is  $\varphi$  with  $P_a$  replacing every occurrence of  $X$ . For example, we can express  $x < y$  as  $\exists X(X(y) \wedge \neg X(x) \wedge \text{closed}(X))$  where  $\text{closed}(X)$  abbreviates  $\forall x\forall y(X(x) \wedge S(x, y) \supset X(y))$ , which we can picture as

$$\mathcal{L}_{\{a\}, \emptyset}(\text{closed}(P_a)) = \boxed{\boxed{a}^*}$$

for the picture

$$\begin{aligned} \mathcal{L}_{\emptyset, \{x, y\}}(\exists P_a(P_a(y) \wedge \neg P_a(x) \wedge \text{closed}(P_a))) \\ &= \{\rho_{\{x, y\}}(s) \mid s \in \mathcal{L}_{\{a\}, \{x, y\}}(P_a(y) \wedge \neg P_a(x) \wedge \text{closed}(P_a))\} \\ &= \{\rho_{\{x, y\}}(s) \mid s \in \boxed{\boxed{x}^*} \boxed{\boxed{a}^*} \boxed{\boxed{a, y} \boxed{a}^*}\} \\ &= \boxed{\boxed{x}^*} \boxed{\boxed{y}^*} \end{aligned}$$

of  $x < y$ .

Next comes the pay-off in interpreting MSO-sentences over not just  $Z$ -strings but strings of sets. An easy proof by induction on  $\varphi \in \text{MSO}_{\Sigma, V}$  establishes

**Proposition 1** *Let  $\Sigma \in \text{Fin}(Z)$  and  $V \in \text{Fin}(\text{Var})$ . Then for all sets  $A \subseteq \Sigma$  and  $U \subseteq V$ ,*

$$\text{MSO}_{A, U} \subseteq \text{MSO}_{\Sigma, V}$$

and for all  $\varphi \in \text{MSO}_{A, U}$ ,

$$\mathcal{L}_{\Sigma, V}(\varphi) = \{s \in \text{Mod}_V(\Sigma) \mid \rho_{A \cup U}(s) \in \mathcal{L}_{A, U}(\varphi)\}.$$

To pick out  $\text{MSO}_{\Sigma, V}$ -formulas with *no* free variables, we let  $V = \emptyset$  for the set

$$\text{MSO}_{\Sigma} = \text{MSO}_{\Sigma, \emptyset}$$

of  $\text{MSO}_\Sigma$ -sentences, and write  $\models_\Sigma$  for  $\models_{\Sigma, \emptyset}$ , and  $\mathcal{L}_\Sigma(\varphi)$  for  $\mathcal{L}_{\Sigma, \emptyset}(\varphi)$  (where  $\varphi \in \text{MSO}_\Sigma$ ). An immediate corollary to Proposition 1 is that for all  $\varphi \in \text{MSO}_\Sigma$  and  $s \in \text{Mod}_\emptyset(\Sigma) = (2^\Sigma)^*$ ,

$$s \models_\Sigma \varphi \iff \rho_{\text{voc}(\varphi)}(s) \models_{\text{voc}(\varphi)} \varphi \quad (2)$$

where  $\text{voc}(\varphi)$  is the smallest subset  $A$  of  $Z$  such that  $\varphi \in \text{MSO}_A$

$$\text{voc}(\varphi) = \bigcap \{A \in \text{Fin}(Z) \mid \varphi \in \text{MSO}_A\}$$

(sharpening the description of  $\text{voc}(\varphi)$  in the introduction).

## 2.2 Regularity

For any finite sets  $A$  and  $B$ , the restriction

$$\rho_A^B := \rho_A \cap ((2^B)^* \times (2^B)^*)$$

of  $\rho_A$  to  $(2^B)^*$  is a regular relation – i.e. computed by a finite-state transducer (with one state, mapping  $\alpha \subseteq B$  to  $\alpha \cap A$ ). For the preimage (or inverse image) of a language  $L$  under a relation  $R$ , we borrow the notation

$$\langle R \rangle L := \{s \mid (\exists s' \in L) sRs'\}$$

from dynamic logic, instead of  $R^{-1}L$  which becomes awkward for long  $R$ 's. We can then rephrase the definition of  $\text{Mod}_V(\Sigma)$  as

$$\text{Mod}_V(\Sigma) = \bigcap_{x \in V} \langle \rho_{\{x\}}^{\Sigma \cup V} \rangle \boxed{x}^*. \quad (3)$$

Similarly we have

$$\mathcal{L}_{\Sigma, V}(S(x, y)) = \text{Mod}_V(\Sigma) \cap \langle \rho_{\{x, y\}}^{\Sigma \cup V} \rangle \boxed{x} \boxed{y}^* \quad \text{for } x, y \in V$$

and writing  $\theta_A^B$  for the inverse of  $\rho_A^B$ ,

$$\begin{aligned} \mathcal{L}_{\Sigma, V}(\exists x \varphi) &= \text{Mod}_V(\Sigma) \cap \langle \rho_{\Sigma \cup V - \{x\}}^{\Sigma \cup V} \rangle \langle \theta_{\Sigma \cup V - \{x\}}^{\Sigma \cup V \cup \{x\}} \rangle \mathcal{L}_{\Sigma, V \cup \{x\}}(\varphi) \\ &= \text{Mod}_V(\Sigma) \cap \langle \theta_{\Sigma \cup V}^{\Sigma \cup V \cup \{x\}} \rangle \mathcal{L}_{\Sigma, V \cup \{x\}}(\varphi) \quad \text{for } x \notin V. \end{aligned}$$

As regular languages are closed under intersection, complementation and preimages under regular relations (which are themselves closed under inverses), it follows that

**Proposition 2** For every  $\Sigma \in \text{Fin}(Z)$ ,  $V \in \text{Fin}(\text{Var})$  and  $\varphi \in \text{MSO}_{\Sigma, V}$ , the set  $\mathcal{L}_{\Sigma, V}(\varphi)$  of strings in  $\text{Mod}_V(\Sigma)$  that satisfy  $\varphi$  is a regular language.

The aforementioned Büchi–Elgot–Trakhtenbrot theorem (BET) side-steps free variables, making do with  $\text{MSO}_{\Sigma} = \text{MSO}_{\Sigma, \emptyset}$  and a fragment  $\models^{\Sigma} \subseteq \Sigma^* \times \text{MSO}_{\Sigma}$  of  $\models_{\Sigma} \subseteq (2^{\Sigma})^* \times \text{MSO}_{\Sigma}$  given by  $\Sigma$ -strings  $s$  and  $\varphi \in \text{MSO}_{\Sigma}$  such that

$$s \models^{\Sigma} \varphi \iff \iota(s) \models_{\Sigma} \varphi$$

(recalling from Subsection 2.1 that  $\iota(a_1 \cdots a_n) = \boxed{a_1} \cdots \boxed{a_n}$  for  $a_1 \cdots a_n \in \Sigma^n$ ). A language  $L \subseteq \Sigma^*$  is then characterized by BET as regular iff for some sentence  $\varphi \in \text{MSO}_{\Sigma}$ ,

$$L = \{s \in \Sigma^* \mid s \models^{\Sigma} \varphi\}.$$

There is a sense in which the difference between  $s$  and  $\iota(s)$  is purely cosmetic; a simple one-state finite-state transducer computes  $\iota$ . But the  $\text{MSO}_{\Sigma}$ -sentences valid in  $\models^{\Sigma}$  need not be valid in  $\models_{\Sigma}$ ; take the  $\text{MSO}_{\Sigma}$ -sentence

$$\text{spec}(\Sigma) := \forall x \bigvee_{a \in \Sigma} (P_a(x) \wedge \bigwedge_{a' \in \Sigma - \{a\}} \neg P_{a'}(x))$$

specifying in every string position  $x$ , exactly one symbol  $a$  from  $\Sigma$ . BET effectively presupposes  $\text{spec}(\Sigma)$  to extract from  $\varphi \in \text{MSO}_{\Sigma}$  the regular language  $\{s \in \Sigma^* \mid \iota(s) \models_{\Sigma} \varphi\}$  over  $\Sigma$ , rather than the full regular language  $\mathcal{L}_{\Sigma}(\varphi)$  over  $2^{\Sigma}$  from Proposition 2. To represent a regular language over  $2^{\Sigma}$ , BET provides a sentence *not* in  $\text{MSO}_{\Sigma}$  but in  $\text{MSO}_{2^{\Sigma}}$ , which we can translate into  $\text{MSO}_{\Sigma}$  by replacing every subformula  $P_{\alpha}(x)$  (for  $\alpha \subseteq \Sigma$ ) with the conjunction

$$\bigwedge_{a \in \alpha} P_a(x) \wedge \bigwedge_{a' \in \Sigma - \alpha} \neg P_{a'}(x)$$

in  $\text{MSO}_{\Sigma, \{x\}}$  interpretable by  $\models_{\Sigma, V}$ .<sup>4</sup> Insofar as computations are carried out on syntactic representations (e.g., MSO-formulas) rather than on semantic models (designed largely as theoretical aids to understanding), the explosion from  $\Sigma$  to  $2^{\Sigma}$  is computationally worrying in the syntactic step from  $\text{MSO}_{\Sigma}$  to  $\text{MSO}_{2^{\Sigma}}$  rather than in the semantic enrichment of  $\Sigma^*$  to  $(2^{\Sigma})^*$ .

<sup>4</sup> Conversely, we can translate  $\text{MSO}_{\Sigma}$  to  $\text{MSO}_{2^{\Sigma}}$  by replacing subformulas  $P_{\alpha}(x)$ , for  $\alpha \subseteq \Sigma$ , with the disjunction  $\bigvee \{P_a(x) \mid \alpha \subseteq \Sigma \text{ and } a \in \alpha\}$  in  $\text{MSO}_{2^{\Sigma}, \{x\}}$ .

Underlying Proposition 2 is a recipe from  $MSO_{\Sigma,V}$  to the regular expressions

$$\begin{aligned}\mathcal{L}_{\emptyset,\{x,y\}}(x=y) &= \boxed{x,y}^* \\ \mathcal{L}_{\emptyset,\{x,y\}}(S(x,y)) &= \boxed{x} \boxed{y}^* \\ \mathcal{L}_{\{a\},\{x\}}(P_a(x)) &= \{\boxed{a}\}^* \boxed{a,x} \{\boxed{a}\}^*\end{aligned}$$

closed under conjunction, complementation and preimages under  $\rho_A^B$  and  $\theta_A^B$ . These extended regular expressions are as succinct as the formulas in  $MSO_{\Sigma,V}$  they represent (up to a constant factor). That said, if we take the example of  $spec(\Sigma)$ , we can simplify the recipe for  $\mathcal{L}_{\Sigma}(spec(\Sigma))$  considerably to the image of  $\Sigma^*$  under  $\iota$

$$\mathcal{L}_{\Sigma}(spec(\Sigma)) = \{\boxed{a} \mid a \in \Sigma\}^*$$

linear in the size of  $\Sigma$  (as opposed to  $spec(\Sigma)$  with quadratically many occurrences of the variable  $x$ ). The representability of regular languages by regular expressions in general (i.e., Kleene's theorem) raises the question: what useful finite-state tools does MSO add to the usual regular operations? Apart from intersection and complementation (the usual extensions to regular expressions), one tool that  $MSO_{\Sigma}$  introduces is the idea of a string as a model, the proper formulation of which blows  $\Sigma$  up to its power set  $2^{\Sigma}$  (to represent all finite  $MSO_{\Sigma}$ -models, whether or not they satisfy  $spec(\Sigma)$ ). Exploiting that blow up, we can define regular relations such as  $\rho_A^B$  under which preimages of regular languages are also regular. We modify the relations  $\rho_A^B$  in the next subsection, Subsection 2.3, examining the MSO representation of accepting runs of a finite automaton, which is demonstrably more succinct than any available with regular expressions.

### 2.3 *Some parts and sorts*

Using sets as symbols provides a ready approach to meronymy (i.e., parts); we drop the subscript  $A$  on  $\rho_A$  for the non-deterministic relation  $\supseteq$  of componentwise inclusion between strings of the same length

$$\alpha_1 \cdots \alpha_n \supseteq \beta_1 \cdots \beta_m \iff n = m \text{ and } \alpha_i \supseteq \beta_i \text{ for } i \in [n]$$

called *subsumption* in Fernando (2004). For example,  $s \supseteq \rho_A(s)$  for all strings  $s$  of sets. A part of reduced length can be obtained by truncating

a string  $s$  from the front for a suffix  $s'$

$$s \text{ suffix } s' \iff (\exists s'') s = s''s'$$

or from the back for a prefix  $s'$

$$s \text{ prefix } s' \iff (\exists s'') s = s's''.$$

We can then compose the relations  $\supseteq$ , *suffix* and *prefix* for a notion  $\sqsupseteq$  of *containment*

$$\begin{aligned} s \sqsupseteq s' &\iff (\exists s_1, s_2) s \supseteq s_1 \text{ and } s_1 \text{ suffix } s_2 \text{ and } s_2 \text{ prefix } s' \\ &\iff (\exists u, v) s \supseteq us'v \end{aligned}$$

between strings of possibly different lengths. For every atomic  $\text{MSO}_{\Sigma, V}$ -formula  $\varphi$ , the satisfaction set  $\mathcal{L}_{\Sigma, V}(\varphi)$  consists of the strings in  $\text{Mod}_V(\Sigma)$  with characteristic  $\sqsupseteq$ -parts, given as follows.

**Proposition 3** *For all disjoint finite sets  $\Sigma$  and  $V$ ,*

$$\begin{aligned} \mathcal{L}_{\Sigma, V}(x = y) &= \text{Mod}_V(\Sigma) \cap \langle \sqsupseteq \rangle \boxed{x, y} && \text{for } x, y \in V \\ \mathcal{L}_{\Sigma, V}(S(x, y)) &= \text{Mod}_V(\Sigma) \cap \langle \sqsupseteq \rangle \boxed{x} \boxed{y} && \text{for } x, y \in V \\ \mathcal{L}_{\Sigma, V}(P_a(x)) &= \text{Mod}_V(\Sigma) \cap \langle \sqsupseteq \rangle \boxed{a, x} && \text{for } a \in \Sigma, \quad x \in V. \end{aligned}$$

Under Proposition 3, each set  $\mathcal{L}_{\Sigma, V}(\varphi)$  is the intersection of  $\text{Mod}_V(\Sigma)$  with a language  $\langle \sqsupseteq \rangle s_\varphi$ , where  $s_\varphi$  is a string of length  $\leq 2$  that pictures  $\varphi$ . The obvious picture of  $x < y$  is the set  $\boxed{x} \boxed{y}$  of arbitrarily long strings

$$\mathcal{L}_{\Sigma, V}(x < y) = \text{Mod}_V(\Sigma) \cap \langle \sqsupseteq \rangle \boxed{x}^* \boxed{y} \quad \text{for } x, y \in V$$

which is nonetheless easier to visualize (if not read) than the  $\text{MSO}_{\emptyset, \{x, y\}}$ -formula

$$\exists X (X(y) \wedge \neg X(x) \wedge (\forall u, v) (X(u) \wedge S(u, v) \supset X(v)))$$

expressing  $x < y$ . To compress the language  $\boxed{x} \boxed{y}$  to the string  $\boxed{x} \boxed{y}$ , we can replace containment  $\sqsupseteq$  by *weak containment*

$$\supseteq := \{(\alpha_1 \cdots \alpha_n, x_1 \cdots x_n) \mid x_i = \epsilon \text{ or } x_i \subseteq \alpha_i \text{ for } i \in [n]\}$$

with deletions ( $x_i$  equal to the empty string  $\epsilon$ ) allowed anywhere, not just in the front or back of  $\alpha_1 \cdots \alpha_n$  or inside any box  $\alpha_i$ . (For example,  $\boxed{x, a} \boxed{y} \succeq \boxed{x} \boxed{y}$  for all integers  $n \geq 0$ .) Proposition 3 holds with  $\sqsubseteq$  and  $S(x, y)$  replaced by  $\succeq$  and  $x < y$  respectively

$$\begin{aligned} \mathcal{L}_{\Sigma, V}(x = y) &= \text{Mod}_V(\Sigma) \cap \langle \succeq \rangle \boxed{x, y} && \text{for } x, y \in V \\ \mathcal{L}_{\Sigma, V}(x < y) &= \text{Mod}_V(\Sigma) \cap \langle \succeq \rangle \boxed{x} \boxed{y} && \text{for } x, y \in V \\ \mathcal{L}_{\Sigma, V}(P_a(x)) &= \text{Mod}_V(\Sigma) \cap \langle \succeq \rangle \boxed{a, x} && \text{for } a \in \Sigma, x \in V. \end{aligned}$$

Whether the part relation  $R$  is  $\sqsubseteq$  or  $\succeq$ ,<sup>5</sup> what matters for the regularity of  $\mathcal{L}_{\Sigma, V}(\varphi)$  is that the restriction of  $R$  to  $(2^{\Sigma \cup V})^*$

$$R \cap ((2^{\Sigma \cup V})^* \times (2^{\Sigma \cup V})^*)$$

is computable by a finite-state transducer (for all finite sets  $\Sigma$  and  $V$ ). Within  $\text{Mod}_V(\Sigma)$  are part relations  $\rho_{\{x\}}$  (for  $x \in V$ ) revealed by the equation

$$\text{Mod}_V(\Sigma) = \bigcap_{x \in V} \langle \rho_{\{x\}}^{\Sigma \cup V} \rangle \boxed{x}^*. \quad (3)$$

Moving from MSO to finite automata, let us rewrite pairs  $\Sigma, V$  as pairs  $A, Q$  of disjoint finite sets  $A$  and  $Q$ , and define an  $(A, Q)$ -automaton to be a triple  $\mathcal{A} = (\rightarrow_{\mathcal{A}}, F_{\mathcal{A}}, q_{\mathcal{A}})$  consisting of

- (i) a set  $\rightarrow_{\mathcal{A}}$  of triples in  $Q \times A \times Q$  specifying  $\mathcal{A}$ -transitions (where we write  $q \xrightarrow{a}_{\mathcal{A}} q'$  instead of  $(q, a, q') \in \rightarrow_{\mathcal{A}}$ )
- (ii) a set  $F_{\mathcal{A}} \subseteq Q$  of  $\mathcal{A}$ -final states, and
- (iii) an  $\mathcal{A}$ -initial state  $q_{\mathcal{A}} \in Q$ .

Given an  $(A, Q)$ -automaton  $\mathcal{A}$ , an  $\mathcal{A}$ -accepting run is a string

$$\boxed{a_1, q_1} \boxed{a_2, q_2} \cdots \boxed{a_n, q_n} \in (2^{A \cup Q})^*$$

such that  $q_{\mathcal{A}} \xrightarrow{a_1}_{\mathcal{A}} q_1$  and  $q_n \in F_{\mathcal{A}}$  and

$$q_{i-1} \xrightarrow{a_i}_{\mathcal{A}} q_i \text{ for } 1 < i \leq n$$

---

<sup>5</sup>For the present purposes, we can take a *part relation* to be any fragment  $R$  of  $\succeq$  (i.e., whenever  $sRs'$ ,  $s \succeq s'$ ). Thus,  $\rho_A$ , *suffix*, *prefix*,  $\sqsubseteq$  and  $\succeq$  are all part relations.

(where for  $n = 0$ , the empty string  $\epsilon$  is an  $\mathcal{A}$ -accepting run iff  $q_{\mathcal{A}} \in F_{\mathcal{A}}$ ). Let  $AccRuns(\mathcal{A})$  be the set of  $\mathcal{A}$ -accepting runs. Clearly, for all  $s \in A^*$ ,

$$\mathcal{A} \text{ accepts } s \iff (\exists s' \in AccRuns(\mathcal{A})) \iota(s) = \rho_A(s')$$

(recalling  $\iota(a_1 \cdots a_n) = \boxed{a_1} \cdots \boxed{a_n}$ ). That is,  $\mathcal{A}$  accepts the language

$$\mathcal{L}(\mathcal{A}) = \langle \iota_A \rangle \langle \theta_A^{A \cup Q} \rangle AccRuns(\mathcal{A})$$

(recalling  $\theta_A^B$  is the inverse of  $\rho_A^B$ ). As for the set  $AccRuns(\mathcal{A})$  of  $\mathcal{A}$ -accepting runs, we start by collecting strings of pairs from  $A$  and  $Q$  in

$$Pairs(A, Q) := \bigcup_{n \geq 0} \left\{ \boxed{a_1, q_1} \cdots \boxed{a_n, q_n} \mid a_1 \cdots a_n \in A^n \text{ and } q_1 \cdots q_n \in Q^n \right\}.$$

We refine  $Pairs(A, Q)$  to  $AccRuns(\mathcal{A})$ , taking into account

- (i) the set  $Init[\mathcal{A}]$  of strings that start with a pair  $a, q$  such that  $q_{\mathcal{A}} \xrightarrow{a}_{\mathcal{A}} q$

$$Init[\mathcal{A}] := \langle prefix \rangle \left\{ \boxed{a, q} \mid q_{\mathcal{A}} \xrightarrow{a}_{\mathcal{A}} q \right\}$$

- (ii) the set  $Final[\mathcal{A}]$  of strings ending with an  $\mathcal{A}$ -final state

$$Final[\mathcal{A}] := \langle \triangleright \rangle \langle suffix \rangle \left\{ \boxed{q} \mid q \in F_{\mathcal{A}} \right\}$$

and

- (iii) the set  $Bad[\mathcal{A}]$  of strings containing  $\boxed{q \mid a, q'}$  for triples  $(q, a, q')$  outside the set  $\xrightarrow{\mathcal{A}}$  of  $\mathcal{A}$ -transitions

$$Bad[\mathcal{A}] := \langle \triangleright \rangle \langle suffix \rangle \langle prefix \rangle \left\{ \boxed{q \mid a, q'} \mid (q, a, q') \in Q \times A \times Q \text{ and not } q \xrightarrow{a}_{\mathcal{A}} q' \right\}.$$

Note that  $\langle R \rangle \langle R' \rangle L = \langle R; R' \rangle L$  for all relations  $R$  and  $R'$  and sets  $L$ , where  $R; R'$  is the *relational composition* of  $R$  and  $R'$

$$R; R' := \{(s, s') \mid (\exists s'') sRs'' \text{ and } s''R's'\}$$

(and containment  $\sqsupseteq$  is the relational composition of  $\triangleright$ , *suffix* and *prefix*).

**Proposition 4** For all disjoint finite sets  $A$  and  $Q$ , and all  $(A, Q)$ -automata  $\mathcal{A}$ , the set  $AccRuns(\mathcal{A})$  of  $\mathcal{A}$ -accepting runs consists of all strings in  $Pairs(A, Q)$  that belong to  $Init[\mathcal{A}]$  and  $Final[\mathcal{A}]$  but not to  $Bad[\mathcal{A}]$

$$AccRuns(\mathcal{A}) = Pairs(A, Q) \cap Init[\mathcal{A}] \cap Final[\mathcal{A}] - Bad[\mathcal{A}].$$

Note that the language  $Pairs(A, Q)$  can be formed by defining for any finite sets  $C$  and  $D$ , the set

$$Spec_D(C) := \mathcal{L}_{C \cup D}(spec(C)) = \langle \rho_C^{C \cup D} \rangle \{ \boxed{c} \mid c \in C \}^*$$

of  $2^{C \cup D}$ -strings with exactly one element of  $C$  in each box, making

$$Pairs(A, Q) = Spec_Q(A) \cap Spec_A(Q).$$

The language  $\{ \boxed{c} \mid c \in C \}$  of  $\rho_C$ -parts of strings in  $Spec_D(C)$  includes strings of any finite length, whereas all strings  $\boxed{a, q}$ ,  $\boxed{q}$  and  $\boxed{q} \boxed{a, q'}$  pictured in  $Init_{\mathcal{A}}$ ,  $Final_{\mathcal{A}}$  and  $Bad_{\mathcal{A}}$  have length  $\leq 2$ . This is one sense in which the constraint  $Pairs(A, Q)$  is global (wide), while  $Init[\mathcal{A}] \cap Final[\mathcal{A}] - Bad[\mathcal{A}]$  is local (narrow). A second sense is that  $Pairs(A, Q)$  captures accepting runs of all  $(A, Q)$ -automata, just as  $Mod_V(\Sigma)$  in Proposition 3 captures all  $MSO_{\Sigma, V}$ -models. That is,  $Pairs(A, Q)$  and  $Mod_V(\Sigma)$  are general, sortal constraints that provide a context (or background) for more specific constraints to differentiate strings of the same sort; this differentiation is effected in Propositions 4 and 3 by attributes or parts that pick out substrings of length bounded by 2. Table 1 outlines the situation.

Table 1:

	sortal (taxonomic)	differential (meronymic)
Proposition 3	$Mod_V(\Sigma)$	$\langle \exists \rangle s_\varphi$
Proposition 4	$Pairs(A, Q)$	$Init[\mathcal{A}] \cap Final[\mathcal{A}] - Bad[\mathcal{A}]$
	general	specific (to $\varphi, \mathcal{A}$ )
length of part	unbounded ( $\rho_A$ )	bounded ( $\leq 2$ )

A further difference between the second and third columns of Table 1 is that whereas the sortal constraints  $Mod_V(\Sigma)$  and  $Pairs(A, Q)$  employ deterministic part relations  $\rho_A$ , the differential constraints  $\langle \exists \rangle s_\varphi$  and  $Init[\mathcal{A}] \cap Final[\mathcal{A}] - Bad[\mathcal{A}]$  employ non-deterministic relations  $\exists$ , *prefix* and the relational composition  $\triangleright$ ; *suffix*. Although it is



clear from Subsection 2.1 that the work done by  $\sqsupseteq$ , *prefix* and  $\sqsupseteq$ ; *suffix* can be done by  $\rho_A$ , non-determinism nevertheless arises when introducing existential quantification through the inverse  $\theta_A^B$  of  $\rho_A^B$  (used for the step from  $\mathcal{A}$ -accepting runs to the language  $\mathcal{L}(\mathcal{A})$  accepted by  $\mathcal{A}$ ). But while  $\sqsupseteq$ , *prefix* and  $\sqsupseteq$ ; *suffix* search inside a string,  $\theta_A^B$  searches outside. The search by  $\theta_A^B$  is bounded only because the set  $B$  (that serves as its superscript) is finite (with elements of  $B$  not in  $A$  amounting to auxiliary symbols).

Non-determinism aside, the relations  $\sqsupseteq$ , *prefix* and  $\sqsupseteq$ ; *suffix* differ from  $\rho_A$  and its inverse in relating strings of different lengths. Indeed, Table 1 arose above from the observation that parts with length  $\leq 2$  suffice for the constraints in the third column. That said, in the next section, we compress strings deterministically without setting any pre-determined bounds (such as 2) on the resulting length, for sorts and parts alike.

### 3 COMPRESSION AND INSTITUTIONS

Having established through Proposition 1 the reduction

$$s \models_{\Sigma} \varphi \iff \rho_{\text{voc}(\varphi)}(s) \models_{\text{voc}(\varphi)} \varphi \quad (2)$$

(for all  $\varphi \in \text{MSO}_{\Sigma}$  and  $s \in (2^{\Sigma})^*$ ), we proceeded to part relations other than  $\rho_A$  in Table 1. The present section calls attention to string functions that can (unlike  $\rho_A$ ) shorten a string, pointing the equivalence (2) and Table 1 in the direction of institutions (Goguen and Burstall 1992). As the length  $n$  of a string determines the domain  $[n] = \{1, \dots, n\}$  of the model encoded by the string, compression alters ontology over and above  $A$ -reducts produced by  $\rho_A$ .

#### 3.1 From compression to inverse limits

We can strip off empty boxes at the front and back of a string  $s$  by defining

$$\text{unpad}(s) := \begin{cases} \text{unpad}(s') & \text{if } s = \boxed{\phantom{x}} s' \text{ or else } s = s' \boxed{\phantom{x}} \\ s & \text{otherwise} \end{cases}$$

so that  $\text{unpad}(s)$  neither begins nor ends with  $\boxed{\phantom{x}}$ , making

$$\boxed{\phantom{x}}^* \boxed{x} \boxed{\phantom{x}}^* = \langle \text{unpad} \rangle \boxed{x}.$$

Using *unpad*-preimages, we can eliminate Kleene stars from the right side of

$$\text{Mod}_V(\Sigma) = \bigcap_{x \in V} \langle \rho_{\{x\}}^{\Sigma \cup V} \rangle \boxed{x}^* \quad (3)$$

and from the extended regular expressions from Proposition 3 for the sets  $\mathcal{L}_{\Sigma, V}(\varphi)$  of strings satisfying formulas  $\varphi \in \text{MSO}_{\Sigma, V}$ . Regular expressions with complementation instead of Kleene star are known in the literature as *star-free regular expressions*, denoting, by a theorem of McNaughton and Papert, the first-order definable sets (Theorem 7.26, page 127, Libkin 2010). We can formulate a notion of  $\Sigma$ -*extended star-free expressions* matching the regular expressions over  $2^\Sigma$ , but while it is easy enough to introduce the constructs  $\langle \exists \rangle$  and  $\langle \text{unpad} \rangle$ , we need subsets and supersets of  $\Sigma$  to relativize complementation and define the constructs  $\langle \rho_A^B \rangle$  and  $\langle \theta_A^B \rangle$ , where  $\theta_A^B$  is the inverse of  $\rho_A^B$ . On the positive side, this complication is potentially interesting as it suggests a hierarchy between the star-free regular languages and regular languages over  $2^\Sigma$ . Be that as it may, our present concerns lie elsewhere.

Rather than separating the set *Var* of first-order variables from the set *Z* of subscripts *a* on unary predicates  $P_a$ , we can formulate the requirement on a symbol *a* that it occur exactly once in  $\text{MSO}_{\{a\}}$

$$\text{nom}(a) := \exists x \forall y (P_a(y) \equiv x = y)$$

characteristic of *nominals* in the sense of Hybrid Logic (e.g., Braüner 2014, or “world variables” in Prior 1967, pages 187–197), with

$$\mathcal{L}_{\{a\}}(\text{nom}(a)) = \langle \text{unpad} \rangle \boxed{a}.$$

From  $\text{nom}(a)$ , it is a small step to the condition  $\text{interval}(a)$  that *a* occur in a string without gaps, which we can express in  $\text{MSO}_{\{a\}}$  as

$$\text{interval}(a) := \exists x P_a(x) \wedge \neg \exists y \text{gap}_a(y)$$

where  $\text{gap}_a(y)$  says *a* does not occur at position *y* even though it occurs before and after *y*

$$\text{gap}_a(y) := \neg P_a(y) \wedge \exists u \exists v (u < y \wedge y < v \wedge P_a(u) \wedge P_a(v))$$

so that

$$\mathcal{L}_{\{a\}}(\text{interval}(a)) = \langle \text{unpad} \rangle \boxed{a}^+. \quad (4)$$

We can eliminate  $\cdot^+$  from the right of (4) by defining a function  $bc$  that given a string  $s$ , compresses blocks  $\alpha^n$  of  $n > 1$  consecutive occurrences in  $s$  of the same symbol  $\alpha$  to a single  $\alpha$ , leaving  $s$  otherwise unchanged

$$bc(s) := \begin{cases} bc(\alpha s') & \text{if } s = \alpha \alpha s' \\ \alpha bc(\beta s') & \text{if } s = \alpha \beta s' \text{ with } \alpha \neq \beta \\ s & \text{otherwise} \end{cases}$$

so that  $\boxed{a}^+$  is  $\langle bc \rangle \boxed{a}$ . In general,  $bc$  outputs only stutter-free strings, where a string  $\alpha_1 \alpha_2 \cdots \alpha_n$  is *stutter-free* if  $\alpha_i \neq \alpha_{i+1}$  for  $i$  from 1 to  $n-1$ . Construing boxes in a string as moments of time, we can view  $bc$  as implementing “McTaggart’s dictum that ‘there could be no time if nothing changed’” (Prior 1967, page 85). The restriction of  $bc$  to any finite alphabet is computable by a finite-state transducer, as are, for all  $\Sigma \in \text{Fin}(Z)$  and  $A \subseteq \Sigma$ , the composition  $\rho_A^\Sigma; bc$  for  $bc_A^\Sigma$

$$bc_A^\Sigma(s) := bc(\rho_A^\Sigma(s)) \quad \text{for } s \in (2^\Sigma)^*$$

and the composition  $bc_A^\Sigma; \text{unpad}$  for  $\pi_A^\Sigma$

$$\pi_A^\Sigma(s) := \text{unpad}(bc_A^\Sigma(s)) = bc(\text{unpad}(\rho_A^\Sigma(s))) \quad \text{for } s \in (2^\Sigma)^*.$$

For  $a \in \Sigma$ , the  $(2^\Sigma)$ -strings in which  $a$  is an interval are those that  $\pi_{\{a\}}^\Sigma$  maps to  $\boxed{a}$

$$\mathcal{L}_\Sigma(\text{interval}(a)) = \langle \pi_{\{a\}}^\Sigma \rangle \boxed{a}.$$

The functions  $\pi_A^\Sigma$  compose nicely

$$\text{whenever } A \subseteq B \subseteq \Sigma, \quad \pi_A^\Sigma = \pi_B^\Sigma; \pi_A^B \quad (5)$$

from which it follows that

$$\begin{aligned} \mathcal{L}_\Sigma\left(\bigwedge_{a \in A} \text{interval}(a)\right) &= \bigcap_{a \in A} \mathcal{L}_\Sigma(\text{interval}(a)) \\ &= \bigcap_{a \in A} \langle \pi_{\{a\}}^\Sigma \rangle \boxed{a} \\ &= \langle \pi_A^\Sigma \rangle \text{Interval}(A) \end{aligned}$$

where  $\text{Interval}(A)$  is the  $\pi_A^A$ -image of  $\bigcap_{a \in A} \langle \pi_{\{a\}}^A \rangle \boxed{a}$

$$\text{Interval}(A) := \left\{ \pi_A^A(s) \mid s \in \bigcap_{a \in A} \langle \pi_{\{a\}}^A \rangle \boxed{a} \right\}.$$

Conflating a string  $s$  with the language  $\{s\}$ , observe that  $Interval(\{a\}) = \boxed{a}$ . For  $a \neq a'$ , the set  $Interval(\{a, a'\})$  consists of thirteen strings, one per interval relation in Allen (1983), which can be partitioned

$$Interval(\{a, a'\}) = \mathcal{L}(a \circ a') \cup \mathcal{L}(a < a') \cup \mathcal{L}(a' < a)$$

between the nine-element set

$$\mathcal{L}(a \circ a') := \{\boxed{a}, \boxed{a'}, \epsilon\} \boxed{a, a'} \{\boxed{a}, \boxed{a'}, \epsilon\}$$

describing overlap  $\circ$  between  $a$  and  $a'$  insofar as for all  $s \in Interval(\Sigma)$  with  $a, a' \in \Sigma$ ,

$$s \models_{\Sigma} \exists x (P_a(x) \wedge P_{a'}(x)) \iff \pi_{\{a, a'\}}^{\Sigma}(s) \in \mathcal{L}(a \circ a')$$

and the two-element sets

$$\begin{aligned} \mathcal{L}(a < a') &:= \{\boxed{a \mid a'}, \boxed{a \mid \mid a'}\} \\ \mathcal{L}(a' < a) &:= \{\boxed{a' \mid a}, \boxed{a' \mid \mid a}\} \end{aligned}$$

describing complete precedence  $<$  insofar as for all  $s \in Interval(\Sigma)$  with  $a, a' \in \Sigma$ ,

$$s \models_{\Sigma} \forall x \forall y ((P_a(x) \wedge P_{a'}(y)) \supset x < y) \iff \pi_{\{a, a'\}}^{\Sigma}(s) \in \mathcal{L}(a < a')$$

and similarly for  $a' < a$ . Event structures are built around the relations  $\circ$  and  $<$  in Kamp and Reyle (1993) (pages 667–674) to express the Russell-Wiener event-based conception of time, a particular elaboration of McTaggart's dictum mentioned above. The sets  $Interval(A)$  above provide representations of finite event structures (Fernando 2011).

Requiring that event structures be finite flies against the popularity of, for instance, the real line  $\mathbb{R}$  in temporal semantics (e.g., Kamp and Reyle 1993, page 670). But we can approximate any infinite set  $Z$  by its set  $Fin(Z)$  of finite subsets, using the inverse system  $(Interval(A))_{A \in Fin(Z)}$ ,

$$\pi_{A,B} : Interval(B) \rightarrow Interval(A), \quad s \mapsto \pi_A^B(s) \quad \text{for } A \subseteq B \in Fin(Z)$$

for the *inverse limit*

$$\{\mathbf{a} : Fin(Z) \rightarrow Fin(Z)^* \mid \mathbf{a}(A) = \pi_{A,B}(\mathbf{a}(B)) \text{ whenever } A \subseteq B \in Fin(Z)\}$$

consisting of maps  $\mathbf{a} : \text{Fin}(Z) \rightarrow \text{Fin}(Z)^*$  that respect the projections  $\pi_{A,B}$ . An element of that inverse limit, in case  $\mathbb{R} \subseteq Z$ , is the map  $\mathbf{a}_{\mathbb{R}}$  such that for all  $r_1 \cdots r_n \in \mathbb{R}^*$ ,

$$\mathbf{a}_{\mathbb{R}}(\{r_1, r_2, \dots, r_n\}) = \boxed{r_1} \boxed{r_2} \cdots \boxed{r_n} \quad \text{for } r_1 < r_2 < \dots < r_n$$

copying  $\mathbb{R}$ . Notice that compressing strings via  $\pi_{A,B}$  allows us to lengthen the strings in the inverse limit. If we remove the compression  $bc$  in  $\pi_{A,B}$ , we are left with the map  $\rho_A$  that leaves the ontology intact (insofar as the domain of an MSO-model is given by the string length), whilst restricting the vocabulary (for  $A$ -reducts).

### 3.2 From inverse systems to institutions

We have left out from the language  $\text{Interval}(\{a\}) = \boxed{a}$  the string  $\boxed{\boxed{a}}$  (among many others) that satisfies  $\text{interval}(a)$ , having built  $\text{unpad}$  into  $\pi_A^A$ . Notice that  $a$  is bounded to the left in  $\boxed{\boxed{a}}$

$$\boxed{\boxed{a}} \models_{\{a\}} \exists x \exists y (S(x, y) \wedge P_a(y) \wedge \neg P_a(x))$$

but not in  $\boxed{a}$ . The functions  $\pi_A^B$  underlying  $\text{Interval}(A)$  abstract away information about boundedness, which is fine if we assume intervals are bounded (as in Allen 1983). But what if we wish to study intervals that may or may not be left-bounded? Or, for that matter, strings where  $a$  may or may not be an interval? The line we pursue in this subsection harks back to Table 1 at the end of Section 2, encoding presuppositions in the second column (e.g.,  $\text{Mod}_V(\Sigma)$ ), and assertions in the third column (e.g.,  $\langle \exists \rangle s_\varphi$ ). For instance, we presuppose a string  $s$  is stutter-free (i.e.,  $s = bc(s)$ ) and assert that  $a$  is an interval in  $s$ , to replace  $\text{Interval}(A)$  by the intersection

$$\underbrace{\{bc(s) \mid s \in (2^A)^*\}}_{\text{presupposition}} \cap \underbrace{\bigcap \left\{ \langle \pi_{\{a\}}^A \rangle \boxed{a} \mid a \in A \right\}}_{\text{assertion}}$$

of which  $\boxed{\boxed{a}}$  and  $\boxed{a}$  are members, for  $a \in A$ . More generally, the idea is to refine the inverse system from the previous subsection to certain concrete instances of institutions (in the sense of Goguen and Burstall 1992) given by suitable functions on strings.

More precisely, let  $Z$  be a large set of symbols, and  $f$  be a function on  $\text{Fin}(Z)$ -strings (e.g.,  $bc$ ). For any finite subset  $A$  of  $Z$ , let  $P_f(A)$  be the image of  $(2^A)^*$  under  $f$

$$P_f(A) := \{f(s) \mid s \in (2^A)^*\}$$

and let  $f_A$  be the composition  $f_A = \rho_A; f$

$$f_A(s) := f(\rho_A(s)) \quad \text{for } s \in \text{Fin}(Z)^*.$$

Thus,  $P_f(A)$  is the image of  $\text{Fin}(Z)^*$  under  $f_A$ . More importantly, for every pair  $(B, A)$  of finite subsets of  $Z$  such that  $A \subseteq B$ , we define the function  $P_f(B, A) : P_f(B) \rightarrow P_f(A)$  sending  $s \in P_f(B)$  to  $f_A(s) \in P_f(A)$

$$P_f(B, A)(s) := f_A(s) \quad \text{for } s \in P_f(B).$$

Now, to say  $P_f$  is an inverse system over  $\text{Fin}(Z)$  is to require that for all  $A \in \text{Fin}(Z)$ ,

(c1)  $P_f(A, A)$  is the identity function on  $P_f(A)$ ; i.e.,

$$f_A(f(s)) = f(s) \quad \text{for all } s \in (2^A)^*$$

and whenever  $A \subseteq B \subseteq C \in \text{Fin}(Z)$ ,

(c2)  $P_f(C, A)$  is the composition  $P_f(C, B); P_f(B, A)$ ; i.e.,

$$f_A(f(s)) = f_A(f_B(f(s))) \quad \text{for all } s \in (2^C)^*.$$

Functions  $f$  validating conditions (c1) and (c2) include the identity function on  $\text{Fin}(Z)^*$  (in which case  $f_A$  is  $\rho_A$ ), *unpad* and *bc* (see Fernando 2014, where inverse systems  $P_f$  are referred to as presheaves). The condition (c2) reduces to the condition

$$\text{whenever } A \subseteq B \subseteq \Sigma, \quad \pi_A^\Sigma = \pi_B^\Sigma; \pi_A^B \tag{5}$$

from the previous subsection, for  $f$  equal to the composition  $bc; \text{unpad}$  (meeting also the requirement (c1)). To capture the entry  $\text{Mod}_V(\Sigma)$  in the second column and row of Table 1 in terms of  $P_f$ , we must treat a first-order variable in  $V$  as a symbol  $a \in Z$  (as in the previous subsection), and build into  $f$  both the uniqueness and existence conditions that  $\text{nom}(a)$  expresses, for  $a \in V$ . To ensure that no  $a \in V$  occur more than once in a string  $s$ , we delete occurrences in  $s$  of  $a$  after its first, setting for all  $\alpha_1 \cdots \alpha_n \in \text{Fin}(Z)^*$ ,

$$u_V(\alpha_1 \cdots \alpha_n) := \beta_1 \cdots \beta_n \quad \text{where } \beta_i := \alpha_i - \left( V \cap \bigcup_{j=1}^{i-1} \alpha_j \right) \text{ for } i \in [n].$$

To ensure each  $a \in V$  occurs at least once in the string, we put  $V$  at the very end

$$e_V(s\alpha) := s(\alpha \cup V)$$

with  $e_V(\epsilon) := V$  for the empty string  $\epsilon$ . Now, if  $f$  is the composition  $e^V; u^V$  then

$$\text{Mod}_V(\Sigma) = P_f(\Sigma \cup V)$$

and (c1) and (c2) hold.

The third column of Table 1 calls for further ingredients. Let us define a  $Z$ -form to be a function  $sen$  with domain  $Fin(Z)$  mapping  $A \in Fin(Z)$  to a set  $sen(A)$  such that for all  $B \in Fin(Z)$ ,

$$sen(A) \cap sen(B) \subseteq sen(A \cap B)$$

and

$$sen(A) \subseteq sen(B) \text{ whenever } A \subseteq B.$$

Given a  $Z$ -form  $sen$ , we can associate every  $\varphi \in \bigcup \{sen(A) \mid A \in Fin(Z)\}$  with the finite subset

$$\text{voc}(\varphi) = \bigcap \{A \in Fin(Z) \mid \varphi \in sen(A)\}$$

of  $Z$  such that

$$\varphi \in sen(A) \iff \text{voc}(\varphi) \subseteq A$$

for all  $A \in Fin(Z)$ . Next, given a function  $f$  on  $Fin(Z)^*$  and a  $Z$ -form  $sen$ , let us agree that a  $(f, sen)$ -specification  $\mathcal{L}$  is a function with domain  $Fin(Z)$  mapping  $A \in Fin(Z)$  to a function  $\mathcal{L}_A$  with domain  $sen(A)$  mapping  $\varphi \in sen(A)$  to a set  $\mathcal{L}_A(\varphi)$  of strings in  $P_f(A)$ . The intuition is that  $\mathcal{L}_A(\varphi)$  consists of the strings in  $P_f(A)$  that  $A$ -satisfy  $\varphi$

$$s \in \mathcal{L}_A(\varphi) \iff s \text{ } A\text{-satisfies } \varphi \quad (\text{for all } s \in P_f(A)).$$

Putting the ingredients together, let us define a  $(Z, f)$ -quadruple to be a 4-tuple  $(Fin(Z), P_f, sen, \mathcal{L})$  such that

- (i)  $P_f$  is an inverse system over  $Fin(Z)$
- (ii)  $sen$  is a  $Z$ -form, and
- (iii)  $\mathcal{L}$  is a  $(f, sen)$ -specification.

Note that once  $Z$  and  $f$  are fixed, only the third and fourth components  $sen$  and  $\mathcal{L}$  of a  $(Z, f)$ -quadruple  $(Fin(Z), P_f, sen, \mathcal{L})$  may vary. To link up with institutions, as defined in Goguen and Burstall (1992), we view

- (i)  $Fin(Z)$  as a category with morphisms given by  $\subseteq$
- (ii)  $P_f$  as a contravariant functor from  $Fin(Z)$  to the category **Set** of sets and functions, and
- (iii)  $sen$  as a (covariant) functor from  $Fin(\Phi)$  to **Set** such that whenever  $A \subseteq B \in Fin(Z)$ ,  $sen(A, B)$  is the inclusion  $sen(A) \hookrightarrow sen(B)$ .

The one remaining condition a  $(Z, f)$ -quadruple must meet to be an institution is that for all  $A \subseteq B \in Fin(Z)$  and  $\varphi \in sen(A)$ ,

$$s \in \mathcal{L}_B(\varphi) \iff f_A(s) \in \mathcal{L}_A(\varphi) \quad (\text{for all } s \in P_f(B))$$

which we can put as the equation

$$\mathcal{L}_B(\varphi) = P_f(B) \cap \langle f_A \rangle \mathcal{L}_A(\varphi).$$

In fact, the special case  $A = \text{voc}(\varphi)$  suffices.

**Proposition 5** *Given a set  $Z$  and function  $f$  on  $Fin(Z)^*$ , a  $(Z, f)$ -quadruple  $(Fin(Z), P_f, sen, \mathcal{L})$  is an institution iff for all  $\Sigma \in Fin(Z)$  and  $\varphi \in sen(\Sigma)$ ,*

$$\mathcal{L}_\Sigma(\varphi) = P_f(\Sigma) \cap \langle f_{\text{voc}(\varphi)} \rangle \mathcal{L}_{\text{voc}(\varphi)}(\varphi). \quad (6)$$

If  $f$  is the identity on  $Fin(Z)^*$ , and  $sen(\Sigma)$  is  $MSO_\Sigma$ , then (6) becomes the equivalence

$$s \models_\Sigma \varphi \iff \rho_{\text{voc}(\varphi)}(s) \models_{\text{voc}(\varphi)} \varphi \quad (2)$$

for all  $\varphi \in MSO_\Sigma$  and  $s \in (2^\Sigma)^*$ . (6) also represents the division in Table 1 between column 2 ( $P_f(\Sigma)$ ) and column 3 ( $\langle f_{\text{voc}(\varphi)} \rangle \mathcal{L}_{\text{voc}(\varphi)}(\varphi)$ ), whilst leaving open the possibility that  $f$  is not the identity function on  $Fin(Z)^*$  nor is  $\varphi$  an MSO-formula.

Under (6), we may assume without loss of generality that  $sen$  and  $\mathcal{L}$  have the following form. For every  $\Sigma \in Fin(Z)$ , there is a set  $\text{Expr}(\Sigma)$  of expressions  $e$  with denotations  $\llbracket e \rrbracket \subseteq (2^\Sigma)^*$  such that  $sen(\Sigma) = 2^\Sigma \times \text{Expr}(\Sigma)$  consists of pairs  $(A, e)$  of subsets  $A \subseteq \Sigma$  and  $e \in \text{Expr}(\Sigma)$  with  $\text{voc}(A, e) = A$  and

$$\mathcal{L}_\Sigma(A, e) = P_f(\Sigma) \cap \langle f_A \rangle \llbracket e \rrbracket. \quad (7)$$



An instructive example is provided by  $A$  equal to  $\{a\}$ , and  $e$  equal to the extended regular expression  $\langle \exists \rangle \boxed{a} \boxed{a}$  or equivalently, the  $\text{MSO}_{\{a\}}$ -sentence

$$\exists x \exists y (S(x, y) \wedge P_a(x) \wedge P_a(y)).$$

The righthand side of (7) can never hold with  $f = bc$ ; there is *no*  $s \in (2^\Sigma)^+$  such that  $bc_{\{a\}}(s) \supseteq \boxed{a} \boxed{a}$ . A slight revision, however, makes the right hand side  $bc$ -satisfiable; introduce a symbol  $b \neq a$  for  $A$  equal to  $\{a, b\}$  and  $e$  equal to  $\langle \exists \rangle \boxed{a, b} \boxed{a}$  or the  $\text{MSO}_{\{a, b\}}$ -sentence

$$\exists x \exists y (S(x, y) \wedge P_a(x) \wedge P_a(y) \wedge P_b(x)).$$

In general, we can neutralize block compression  $bc$  on a string  $s$  by adding a fresh symbol to alternating boxes in  $s$ , which  $bc$  then leaves unchanged, since

$$bc(s) = s \iff s \text{ is stutter-free}$$

(recalling that  $\alpha_1 \cdots \alpha_n$  is *stutter-free* if  $\alpha_i \neq \alpha_{i+1}$  for  $1 \leq i < n$ ). Similarly, we can add negations  $\bar{a}$  of symbols  $a$  in  $A$  through a function  $cl_A$

$$cl_A(\alpha_1 \cdots \alpha_n) := \beta_1 \cdots \beta_n \text{ where } \beta_i := \alpha_i \cup \{\bar{a} \mid a \in A - \alpha_i\} \text{ for } i \in [n]$$

to express  $bc_A^\Sigma$  in terms of  $\pi_B^\Sigma$

$$bc_A^\Sigma = cl_A; \pi_{c(A)}^\Sigma; \rho_A \text{ where } c(A) := A \cup \{\bar{a} \mid a \in A\}$$

treating  $\bar{a} \in c(A) - A$  as an auxiliary symbol, and

$$bc_A^\Sigma; cl_A = cl_A; \pi_{c(A)}^\Sigma.$$

Returning to (7) with  $f = bc$ , we can say  $a$  is bounded to the left

$$\mathcal{L}_\Sigma(\{a\}, \exists x (\neg P_a(x) \wedge \forall y (P_a(y) \supset x < y))) = \langle bc_{\{a\}}^\Sigma \rangle \langle \text{prefix} \rangle \square$$

applying *prefix* after  $bc$ , and say  $a$  overlaps  $a'$

$$\mathcal{L}_\Sigma(\{a, a'\}, \exists x (P_a(x) \wedge P_{a'}(x))) = \langle bc_{\{a, a'\}}^\Sigma \rangle \langle \exists \rangle \boxed{a, a'}$$

applying containment  $\supseteq$  after  $bc$ . It is clear that *unpad* is just one of many relations that can come after  $bc_A^\Sigma$  (leading, in this case, to  $\pi_A^\Sigma = bc_A^\Sigma; \text{unpad}$ ). The projection  $\rho_A^\Sigma$  in  $bc_A^\Sigma = \rho_A^\Sigma; bc$  changes the granularity from  $\Sigma$  to  $A$  before  $bc$  reduces the ontology to suit  $A$ , and part

relations (such as *prefix*, containment  $\sqsupseteq$  or *unpad*) pick out a temporal span to frame a string (such as  $\square$  or  $\boxed{a, a'}$ ) picturing an assertion (e.g., left-boundedness, overlap). We are dividing here the choice of an expression  $e_\varphi$  denoting the language  $\mathcal{L}_{\text{voc}(\varphi)}(\varphi)$  in Proposition 5 between a relation  $R$  and a string  $s$  for  $e_\varphi = \langle R \rangle s$ . Such a choice presupposes the finite approximability of the model of interest via the inverse limit of  $P_f$  (the discreteness of strings mirroring the bounded granularity of natural language statements, rife with talk of “the next moment”). Finite approximability is not only plausible but arguably implicit in accounts such as Reichenbach (1947) of tense and aspect.

4

## CONCLUSION

There is no question that as declarative devices specifying sets of strings accepted by finite automata, regular expressions are more popular than MSO. What MSO offers, however, is a model-theoretic perspective on strings with computable notions of entailment (inclusions between regular languages being decidable), in addition to Boolean connectives that expose deficiencies in succinctness of regular expressions (e.g., Gelade and Neven 2012). Mapping a finite automaton  $\mathcal{A}$  to a regular expression denoting the language  $\mathcal{L}(\mathcal{A})$  accepted by  $\mathcal{A}$  can have exponential cost (Ehrenfeucht and Zeiger 1976; Holzer and Kutrib 2010). A more concise representation of  $\mathcal{L}(\mathcal{A})$  existentially quantifies away the internal states from the accepting runs of  $\mathcal{A}$  (analyzed in Proposition 4 above). Not only can this be carried out in MSO (proving one half of the Büchi–Elgot–Trakhtenbrot theorem), but it is well-known that MSO-sentences can be far more succinct than finite automata (e.g., Libkin 2010, pages 124–125, and 135–136). To match the succinctness of MSO, regular expressions over alphabets  $2^\Sigma$  (for finite sets  $\Sigma$ ) are extended with preimages and images under homomorphisms  $\rho_A$  that output  $A$ -reducts, for  $A \subseteq \Sigma$ .

The step from  $\Sigma$  up to  $2^\Sigma$  is justified by the various notions of part between strings of sets, given by  $\rho_A$ , subsumption  $\supseteq$ , *prefix*, *suffix*, block compression *bc* and *unpad*, all computable (over  $2^\Sigma$ ) by finite-state transducers. Reducts between vocabularies are composed with compression within a fixed vocabulary to fit ontology against the vocabulary. An inverse limit construction (turning compression around to extension) takes us beyond the finite models of MSO to infinite time-

lines, approximated at granularity  $\Sigma$  by strings over the alphabet  $2^\Sigma$ . Different finite sets  $\Sigma$  induce different notions  $\models_\Sigma$  of satisfaction that form institutions, under certain minimal smoothness conditions (used to establish the Büchi–Elgot–Trakhtenbrot theorem in Section 2).

## ACKNOWLEDGEMENTS

My thanks to Mark-Jan Nederhof for his editorship and four anonymous journal referees for their comments and help.

## REFERENCES

- James F. ALLEN (1983), Maintaining knowledge about temporal intervals, *Communications of the ACM*, 26(11):832–843.
- Kenneth R. BEESLEY and Lauri KARTTUNEN (2003), *Finite State Morphology*, CSLI Publications, Stanford.
- Torben BRAÜNER (2014), Hybrid Logic, The Stanford Encyclopedia of Philosophy, <http://plato.stanford.edu/archives/spr2014/entries/logic-hybrid/>.
- David R. DOWTY (1979), *Word Meaning and Montague Grammar*, Reidel, Dordrecht.
- Andrzej EHRENFEUCHT and Paul ZEIGER (1976), Complexity measures for regular expressions, *J. Comput. Syst. Sci.*, 12(2):134–146.
- Tim FERNANDO (2004), A finite-state approach to events in natural language semantics, *Journal of Logic and Computation*, 14(1):79–92.
- Tim FERNANDO (2011), Finite-state representations embodying temporal relations, in *Proceedings 9th International Workshop on Finite State Methods and Natural Language Processing*, pp. 12–20.
- Tim FERNANDO (2014), Incremental semantic scales by strings, in *Proceedings EACL 2014 Workshop on Type Theory and Natural Language Semantics (TTNLS)*, pp. 63–71.
- Tim FERNANDO (2015), The semantics of tense and aspect: A finite-state perspective, in S. LAPPIN and C. FOX, editors, *Handbook of Contemporary Semantic Theory*, pp. 203–236, Wiley-Blackwell, second edition.
- Wouter GELADE and Frank NEVEN (2012), Succinctness of the complement and negation of regular expressions, *ACM Trans. Comput. Log.*, 13(1):4.1–4.19.
- Joseph GOGUEN and Rod BURSTALL (1992), Institutions: Abstract model theory for specification and programming, *J. ACM*, 39(1):95–146.
- Erich GRÄDEL (2007), Finite model theory and descriptive complexity, in *Finite Model Theory and Its Applications*, pp. 125–230, Springer.

Markus HOLZER and Martin KUTRIB (2010), The complexity of regular(-like) expressions, in *Developments in Language Theory*, pp. 16–30, Springer.

Mans HULDEN (2009), Regular expressions and predicate logic in finite-state language processing, in *Finite-State Methods and Natural Language Processing*, pp. 82–97, IOS Press.

Hans KAMP and Uwe REYLE (1993), *From Discourse to Logic*, Kluwer Academic Publishers, Dordrecht.

Ronald M. KAPLAN and Martin KAY (1994), Regular models of phonological rule systems, *Computational Linguistics*, 20(3):331–378.

Leonid LIBKIN (2010), *Elements of Finite Model Theory*, Springer.

Marc MOENS and Mark STEEDMAN (1988), Temporal ontology and temporal reference, *Computational Linguistics*, 14(2):15–28.

Arthur N. PRIOR (1967), *Past, Present and Future*, Clarendon Press, Oxford.

Hans REICHENBACH (1947), *Elements of Symbolic Logic*, London, Macmillan.

Anssi YLI-JYRÄ and Kimmo KOSKENNIEMI (2004), Compiling contextual restrictions on strings into finite-state automata, in *Proceedings of the Eindhoven FASTAR Days*.

*This work is licensed under the Creative Commons Attribution 3.0 Unported License.*

<http://creativecommons.org/licenses/by/3.0/>

