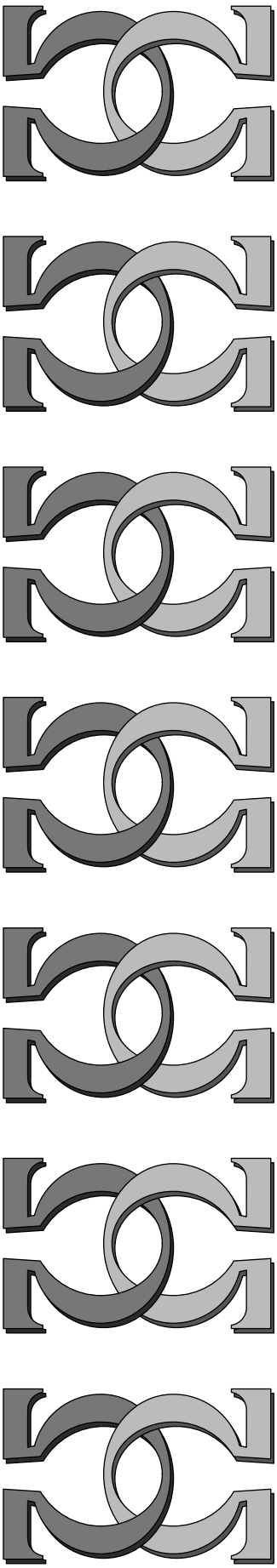

CDMTCS Research Report Series

X-Families: An Approach to the Study of Families of Syntactically Similar Languages

Carlos Martín-Vide
Rovira i Virgili University

Gheorghe Păun
Institute of Mathematics of
the Romanian Academy

Grzegorz Rozenberg
Leiden University

Arto Salomaa
Academy of Finland and Turku University

CDMTCS-076
January 1998

Centre for Discrete Mathematics and
Theoretical Computer Science

X-Families:

An Approach to the Study of Families of Syntactically Similar Languages

Carlos MARTÍN-VIDE
Research Group on Mathematical Linguistics and Language Engineering
Rovira i Virgili University
Pl. Imperial Tàrraco 1, 43005 Tarragona, Spain

Gheorghe PĂUN
Institute of Mathematics of the Romanian Academy
PO Box 1 – 764, 70700 Bucureşti, Romania

Grzegorz ROZENBERG
Department of Computer Science, Leiden University
PO Box 9512, 2300 Leiden, The Netherlands

Arto SALOMAA
Academy of Finland and Turku University
Department of Mathematics
20014 Turku, Finland

Abstract. All classes of grammars investigated in formal language theory generate a language by starting from finite sets of axioms and iteratively applying certain *production* rules which transform "correct" strings into "correct" strings. If the set of rules is fixed and the axiom set is varying over the family of finite languages, then to any grammar we associate a family of languages. When using grammars of certain type X, we call this family an X-family. The aim of this paper is to propose the investigation of such families of languages. We only formulate here some of the basic problems and we start the study of M-families, those obtained when using Marcus contextual grammars as starting point. Several properties of M-families are given, examples and counterexamples are produced, and some decidability results are proven.

1 Introduction: X-Families

The formal language theory deals mainly with the syntax of individual languages. Not so many tools are developed for studying the syntax of *families of languages*, or, otherwise stated, to study families of languages with certain syntactic similarities. In linguistics, this problem is a central one; dialects, stages in the development of a language, languages which form well-defined families are topics much investigated.

A theoretical attempt to deal with this problem is that of grammar forms, introduced in [1]; see details in [11], [14]. The basic idea was to *interpret* the symbols of a usual Chomsky grammar by means of a morphism and to consider the family of languages generated by sets of rules obtained by such interpretations. In this way, we obtain a family associated with the starting grammar. The area was flourishing for a while, but the applications turned out not to be significant. Maybe two of the explanations are that the notion of an interpretation is too general and that the obtained families are "too similar" to Chomsky families. In particular, the regular, linear, and context-free languages form *grammatical families* in the sense of grammar form theory. This should be contrasted with the fact that the natural languages does not fit easily the Chomsky hierarchy. In terms of [3] (page 4), "It is entirely possible, for example, that a realistic theory of natural languages would define a class of languages which is incommensurate with the Chomsky types, e.g., a few regular languages, a few non-regular context-free languages, a few non-context-free context-sensitive languages, and so on."

A simple idea of dealing with families of syntactically similar languages (such as the dialects of a natural language) is the following one. All grammars, of all types investigated in formal language theory, contain three basic components: an alphabet, a set of starting strings (axioms), and a set of production rules. One starts from the axioms and one produces a language by iteratively using the production rules. For instance, in the case of Chomsky grammars, we have only one axiom, which is a symbol, and the productions are rewriting rules. In L systems and in pure grammars one usually deals with a string axiom, but in Marcus contextual grammars and in H systems one uses a finite set of axioms (and context adjoining rules in the case of contextual grammars and splicing rules in H systems). We refer for details to the corresponding chapters in [13], as well as to the monographs [8], [10]. Also for Chomsky grammars, L systems and pure grammars we can start from a finite set of axioms and the basic properties of the obtained grammars and associated languages remain the same. (Actually, L systems with finite sets of axioms were already considered.)

Now, consider a generative mechanism of a type X and let us remove its axiom set. We say that we obtain a *scheme* of type X, in short, an X-scheme. Take such a scheme σ. We can associate with it an infinite *class* of devices of type X by adding to σ any finite set of axioms over the alphabet of σ. In this way, we obtain a family of languages associated with σ, those generated by these generative mechanisms corresponding to σ. We say that this is an X-family. By the definition, the syntactic

similarity of the languages in a given X-family is obvious: the productions used for generating these languages are the same (modulo their applicability to the particular set of axioms).

Such families can be considered for Chomsky grammars, Marcus contextual grammars, pure grammars, L systems, H systems. We identify them as X-families with X equal to C, M, P, L, H, respectively.

We want to point out that, as regards *L schemes*, quite much work has already been done along these lines, [12]. Briefly, an *XL scheme* (for instance, 0L, D0L, ET0L scheme) is an XL system without the axiom. We recall here the following interesting result, [12], concerning growth type combinations of D0L schemes.

D0L growth functions are customarily classified using numbers as follows. Exponential functions are *of type 3*. Functions becoming ultimately 0 are *of type 0*. (Thus, the corresponding D0L sequence becomes the empty word at some stage). Functions not of type 0 and bounded from above by a constant are *of type 1*. All remaining growth functions are *of type 2*. (This class consists of polynomially bounded functions that are not bounded by a constant.) The *growth type combination* of a D0L scheme S is the subset of $\{0,1,2,3\}$, consisting of all numbers i such that S provided with some (nonempty) axiom yields a D0L system of growth type i.

The result we are referring to lists all possible growth type combinations. They are:

$$\{0\},\ \{1\},\ \{3\},\ \{0,1\},\ \{0,3\},\ \{1,3\},\ \{1,2\},$$
$$\{0,1,2\},\ \{0,1,3\},\ \{1,2,3\},\ \{0,1,2,3\}.$$

In other words, in a D0L scheme the growth type 2 never occurs without type 1; this is the only restriction concerning the growth type combination.

Of course, the growth type combination of an L scheme gives an information about the family of languages associated to that scheme, but it does not "describe" the family. So, further investigations remain to be done (for instance, concerning problems as those mentioned in the next section).

2 Basic Problems

In the way sketched above we obtain many X-families of languages. When studying them, specific problems will occur, but there are several classes of problems which can be formulated for all these families (and which are of a basic interest, in view of the motivation we have started with). We mention here some of these problems.

Given an X-scheme σ, we denote by $\mathcal{F}_X(\sigma)$ the X-family associated with σ.

1. (Family Equivalence Problem) Given two X-schemes σ_1 and σ_2, can we decide algorithmically whether or not the equality $\mathcal{F}_X(\sigma_1) = \mathcal{F}_X(\sigma_2)$ holds ?

2. (Family Inclusion Problem) Given two X-schemes σ_1 and σ_2, can we decide algorithmically whether or not the inclusion $\mathcal{F}_X(\sigma_1) \subseteq \mathcal{F}_X(\sigma_2)$ holds ? (Of

course, the positive answer to the second problem implies the positive answer to the first problem.)

3. (Language Equivalence and Inclusion Problems) Given an X-scheme σ and two languages $L_1, L_2 \in \mathcal{F}_X(\sigma)$, are the relations $L_1 = L_2$ and $L_1 \subseteq L_2$ decidable ? (Note that this is not the general decidability problem for languages generated by grammars of type X: we do not consider arbitrary languages L_1, L_2, but languages in the same X-family, hence syntactically similar. It is expected that undecidable problems in the general case are decidable in this restricted case, or, at least, easier to solve.)

4. (Family Membership Problem) Given a language L and an X-scheme σ, decide whether or not $L \in \mathcal{F}_X(\sigma)$ (whether or not a set of axioms exists such that L is generated by σ supplemented with this set of axioms).

5. (Similarity Problem) Given two languages L_1, L_2, decide whether or not there is an X-scheme σ such that both languages L_1, L_2 are in $\mathcal{F}_X(\sigma)$. In the affirmative case, construct effectively a scheme σ such that $L_1, L_2 \in \mathcal{F}_X(\sigma)$.

6. (Closure Properties) What closure properties has an X-family ? Again, one can expect results which are different from the results known for families of languages generated by grammars of type X.

7. (Comparison Problems) Compare the X-families with previously considered families in language theory, such as families in the Chomsky hierarchy, in the hierarchies of Marcus contextual languages, of pure and L languages, of languages generated by H systems. Families defined by descriptional complexity parameters are also good candidates (especially because when considering an X-scheme, we fix the set of productions, hence all associated languages will have the production-complexity bounded by the complexity of the scheme).

Several other problems can be considered (characterizations of X-families, operations with families and their effect on the property of being an X-family, etc). Some suggestions will appear also from the sections below, where we consider as a case-study the M-families, associated to Marcus contextual grammars.

3 Contextual Grammars

The contextual grammars, introduced by [4], are a class of generative mechanisms which produce families of languages which are "incommensurate" with the families in the Chomsky hierarchy. Such a grammar consists of a given set of strings (axioms, well-formed starting phrases) and a set of contexts associated with certain strings, called *selectors*. When a string selector is present in a larger string, then the associated context can be adjoined to the selector occurrence, producing a larger

string. In this way, starting from the axioms and iteratively adjoining contexts we get a language. The mathematical theory of contextual grammars is well developed: see [8]. Up to now, this theory has dealt only with the generation of individual languages, as in "classic" formal language theory. However, because the contextual grammars have as an intrinsic feature the use of a finite set of axioms, they are able to generate in a very natural way families of syntactically similar languages.

We briefly investigate here such families of languages (*M-families*), mainly for illustrating the problems mentioned in the previous section. For instance, we find that no previously considered family of languages (neither in the Chomsky area nor in the contextual area) is an M-family. This shows that this notion of similarity is a new one, indeed, which cannot be reduced to other notions of similarity. Moreover, we find that certain properties of M-families (closure and decidability) are different from the properties of families of contextual languages studied before. This supports again the conclusion drawn before.

First, some general notations.

For an alphabet V we denote by V^* the set of all strings over V; the empty string is denoted by λ and $V^* - \{\lambda\}$ (the set of all non-empty strings over V) is denoted by V^+. Any subset of V^* is called a language (over V). The length of $x \in V^*$ is denoted by $|x|$ and its mirror image (reversal) is denoted by $mi(x)$. By *FIN, REG, LIN, CF, CS, RE* we denote the families of finite, regular, linear, context-free, context-sensitive, and recursively enumerable languages, respectively. For further elements of formal language theory we refer to [13].

A *contextual grammar* is a construct $G = (V, A, P)$, where V is an alphabet, A is a finite language over V, and P is a finite set of triples $(x, (u, v))$ of strings over V. The strings in A are called *axioms*, each triple $(x, (u, v))$ of P is called a *production*; x is the *selector* and (u, v) is the *context* of the production $(x, (u, v))$.

For $w, z \in V^*$ we write $w \Longrightarrow z$ (with respect to G) if and only if $w = w_1 w_2 w_3, z = w_1 u w_2 v w_3$, for some $w_1, w_2, w_3 \in V^*$, and $(w_2, (u, v)) \in P$. (The context (u, v) is adjoined to the selector x found as a substring of w; the resulting string is z.) We denote by $\Longrightarrow^*$ the reflexive and transitive closure of the relation $\Longrightarrow$. The language generated by G is defined by

$$L(G) = \{z \in V^* \mid w \Longrightarrow^* z, w \in A\}.$$

Some remarks are here in order. In [4] the contexts are adjoined at the ends of strings (such grammars are called *external*); the variant considered here, called *internal*, is introduced in [9]. Moreover, we have defined here only grammars with finite sets of selector strings. In the literature one can find also contextual grammmars with regular selection (the productions are of the form $(R, (u, v))$, where R is a regular language and (u, v) is a context), and even with arbitrary selection.

We denote by ICC the family of languages generated by (internal) contextual grammars as above (with finite selection). Proofs and references concerning the following results can be found in [8]:

Theorem 1. (i) $REG \subset ICC \subset CS$. (ii) *$ICC$ is incomparable with LIN and CF.* (iii) *ICC contains non-semilinear languages.* (iv) *The family ICC is an anti-AFL, that is, it is closed under none of the six AFL operations: union, concatenation, Kleene closure, morphisms, inverse morphisms, intersection with regular languages.*

Further results concerning the languages in ICC will be mentioned in the following sections, when necessary. We conclude this section with an **example** which will be useful below: for the grammar

$$G = (\{a, b, c\}, \{acb, bca\}, P),$$

with P containing the productions

$$\begin{aligned}
&(c, (a, a)), (c, (b, b)),\\
&(acb, (a, \lambda)), (acb, (b, \lambda)), (acb, (\lambda, a)), (acb, (\lambda, b)),\\
&(bca, (a, \lambda)), (bca, (b, \lambda)), (bca, (\lambda, a)), (bca, (\lambda, b)),
\end{aligned}$$

we obtain

$$L(G) = \{xcy \mid x, y \in \{a, b\}^+, x \neq mi(y)\}.$$

The reader might check this equality.

4 M-Families of Languages

A pair $\sigma = (V, P)$, where V is an alphabet and P is a finite set of contextual productions $(x, (u, v))$ over V, such that there is at least one production with $uv \neq \lambda$, is called a *contextual scheme*, or an *M-scheme*. For a finite language $A \subseteq V^*$, we define the language $L(\sigma, A) = L(G_{\sigma,A})$, where $G_{\sigma,A}$ is the contextual grammar (V, A, P). Thus, with an M-scheme σ we associate the family of languages

$$\mathcal{F}_M(\sigma) = \{L(\sigma, A) \mid A \in FIN\}.$$

A family of languages $\mathcal{L}$ is called an *M-family* if there is an M-scheme σ such that $\mathcal{L} = \mathcal{F}_M(\sigma)$.

Therefore, the languages in an M-family are generated by contextual grammars which have the same productions, but different axioms. The set of productions are related to the syntactic structure of languages, that is why we do not allow M-schemes with a trivial set of productions, empty or containing only the context (λ, λ). No syntax is then provided by the scheme, the axiom sets remain unchanged.

Of course, changing the set of axioms it is possible that the set of applicable productions changes, too. We shall further discuss this problem in a subsequent section.

By the definition, we have $\mathcal{F}_M(\sigma) \subseteq ICC$ for all σ.

We consider here one **example**. Take the M-scheme $\sigma = (\{a, b, c\}, P)$, where P is the same as in the example at the end of the previous section. Here are some languages in $\mathcal{F}_M(\sigma)$:

$$L(\sigma, \{acb, bca\}) = \{xcy \mid x, y \in \{a, b\}^+, x \neq mi(y)\},$$
$$L(\sigma, \{c\}) = \{xc\ mi(x) \mid x \in \{a, b\}^+\},$$
$$L(\sigma, \{c, acb, bca\}) = \{xcy \mid x, y \in \{a, b\}^+\} \cup \{c\},$$
$$L(\sigma, \{c^k\}) = (L(\sigma, \{c\}))^k,\ k \geq 1.$$

The syntactic similarity of these languages (and of other which can be obtained in this way, for various axiom sets) is obvious. It is worth noting that the family $\mathcal{F}_M(\sigma)$ is infinite (see the last languages, different from each other for different values of k) and that in constructing the language $L(\sigma, c)$ the productions with the selectors acb and bca are never used. This can be easily seen in this example, but it is not a simple matter in the general case: there is no algorithm which can tell us whether or not for a given set of axioms all productions are applicable in generating the corresponding language (Section 6 below deals with this question).

5 Closure Properties

We have mentioned in Theorem 1 (iv) the poor closure properties of the family ICC. The M-families contain similar languages, hence we may expect differences from this point of view. This is true only for union:

Theorem 2. *Each M-family is closed under union.*

Proof. Clearly, for all $\sigma = (V, P)$ and for all $A_1, A_2 \subseteq V^*$, we have

$$L(\sigma, A_1) \cup L(\sigma, A_2) = L(\sigma, A_1 \cup A_2),$$

hence $\mathcal{F}_M(\sigma)$ is closed under union. □

This simple result has several consequences of interest: ICC is not an M-family; no family of contextual languages which is not closed under union is an M-family; if two contextual languages L_1, L_2 are such that $L_1 \cup L_2 \notin ICC$, then there is no M-scheme σ such that $L_1, L_2 \in \mathcal{F}_M(\sigma)$ (the two languages cannot be members of the same M-family, there is no way to find a syntactic similarity between them in terms of M-schemes).

Trivially, the M-families of languages are not necessarily closed to the unary (or to external) operations mentioned in Theorem 1 (iv): Kleene closure, morphisms, inverse morphisms, intersection with regular languages. For instance, if $L \in ICC$ and $h(L) \notin ICC$, for some morphism h, then any M-family containing L is not closed under morphisms (all M-families are included in ICC). The closure under concatenation (and intersection) needs new proofs.

Theorem 3. *There are M-families which are not closed under concatenation or under intersection.*

Proof. Consider the M-scheme

$$\sigma_1 = (\{a, b\}, \{(ab, (a, b)), (ba, (\lambda, a))).$$

The language $L(\sigma_1, \{ab\}) = \{a^n b^n \mid n \geq 1\}$ is in $\mathcal{F}_M(\sigma_1)$, but $L(\sigma_1, \{ab\})L(\sigma_1, \{ab\})$ not: if this language can be generated by σ_1 starting from a set A of axioms, this set must contain strings of the form $a^i b^i a^j b^j$ for some $i, j \geq 1$; to such a string we can apply the production $(ba, (\lambda, a))$ and in this way we obtain strings of the form $a^i b^i a^{j+k} b^j$ for all $k \geq 1$. Such strings are not in $L(\sigma_1, \{ab\})L(\sigma_1, \{ab\})$, a contradiction.

For the intersection we consider the M-scheme

$$\sigma_2 = (\{a, b, c, d, e\}, P),$$

with P containing the following productions:

$$(adbdc, (\lambda, c)),\ (adb, (a, b)),\ (d, (\lambda, e)),$$
$$(aebec, (a, \lambda)),\ (bec, (b, c)),\ (e, (d, \lambda)).$$

For the languages

$$L(\sigma_2, \{adbdc\}) = \{a^n de^r b^n de^s c^m \mid n, m \geq 1, r, s \geq 0\},$$
$$L(\sigma_2, \{aebec\}) = \{a^n d^r eb^m d^s ec^m \mid n, m \geq 1, r, s \geq 0\},$$

we obtain

$$L(\sigma_2, \{adbdc\}) \cap L(\sigma_2, \{aebec\}) = \{a^n deb^n dec^n \mid n \geq 0\}.$$

This language is not even in the family ICC: no context can be removed from its strings in such a way to obtain a string in the same language. □

Note that in the proof above we have used again the freedom of choosing the axioms, without care whether or not all productions are applicable for a set of axioms or another one.

6 Undecidability of Axioms Matching

Given an M-scheme $\sigma = (V, P)$ and a finite language $A \subseteq V^*$, we say that A is a *matching* set of axioms for σ if for every production $(x, (u, v)) \in P$ there is a string $w \in L(\sigma, A)$ such that $w = w_1 x w_2$. (That is, all productions from P are applicable when starting from the axioms in A.)

As we have mentioned above, the axiom sets we have used were in many cases non-matching. However, this is not an easy to check property, because it is of

a dynamic nature: we have to take into consideration all strings generated when starting from a given set of axioms (not, say, only the axioms or any other specified finite set of strings). This is formally confirmed below:

Theorem 4. *There is no algorithm which can decide whether or not a given set of axioms is matching with respect to an arbitrary M-scheme.*

Proof. Let us consider an instance of the Post Correspondence Problem (PCP) over the alphabet $\{a, b\}$, that is, two n-tuples of non-empty strings over $\{a, b\}$, $x = (x_1, \ldots, x_n), y = (y_1, \ldots, y_n)$. We consider the following language

$$\begin{aligned} L_0 \;=\; & \{cd^m cx_{i_1}x_{i_2}\ldots x_{i_k}cy_{i_1}y_{i_2}\ldots y_{i_k} \mid k \geq 1, 1 \leq i_j \leq n, \text{ for all } 1 \leq j \leq n, \\ & \text{and } m = 1, \text{ if } x_{i_1}x_{i_2}\ldots x_{i_k} = y_{i_1}y_{i_2}\ldots y_{i_k}, \text{ and } m = 0, \text{ otherwise}\}. \end{aligned}$$

It is clear that L_0 is a context-sensitive language. Consider a context-sensitive grammar $G = (N, T, S, P)$ generating the language L_0; of course, $T = \{a, b, c, d\}$.

We construct the M-scheme σ with the alphabet

$$V = N \cup T \cup \{f, [,], \vdash\},$$

and the following productions:

1. $(u, ([,]v))$, for each rule $u \to v$ in P;
2. $(\alpha[u], (\vdash, \alpha))$, for all $u \to v \in P$ and $\alpha \in N \cup T$,
3. $(\alpha \vdash \beta, (\vdash, \alpha))$, for all $\alpha, \beta \in N \cup T$,
4. $(\alpha f, (\vdash, \alpha))$, for all $\alpha \in T$,
5. $(fcdc, (f, f))$.

Consider also the set of axioms $\{Sf\}$ and let us examine the work of σ when starting from Sf.

The productions of type 1 are meant to simulate the rewriting rules of P. When simulating a rule $u \to v$, we pass from a string w_1uw_2 to a string $w_1[u]vw_2$; the left member of the rule is bracketed with [and], and the right member of the rule is adjoined to the right of the string $[u]$. The symbols in between [and] are interpreted as "dead": if a further rule of P is simulated on dead symbols, then all resulting symbols will be dead. This means that none of them can exit the brackets [,]. Symbols which are not dead (we say that they are "alive") can circulate to the right, by using rules in group 2 and in group 3. By rules in group 2, alive symbols can jump over blocks of the form $[u]$; we obtain derivation steps of the form $w_1\alpha[u]w_2 \Longrightarrow w_1 \vdash \alpha[u]\alpha w_2$. Also $\vdash$ is considered a "killer": it kills the symbol placed immediately to its right. The rules of type 3 can transport alive symbols across pairs $\vdash \beta$ (hence over dead symbols due to the presence of $\vdash$): $w_1\alpha \vdash \beta w_2 \Longrightarrow w_1 \vdash \alpha \vdash \beta\alpha w_2$. Note that if a dead symbol in a block $\vdash \alpha$ is used

by the selector of a rule of type 3 and interpreted as alive, then we get a pair of adjacent symbols $\vdash$: $w_1 \vdash \alpha \vdash \beta w_2 \Longrightarrow w_1 \vdash\vdash \alpha \vdash \beta\alpha w_2$. This will be a barrier, no selector can use this sequence, hence symbols placed to the left of such a pair $\vdash\vdash$ cannot be transported to the right, across $\vdash\vdash$. This is important, on the one hand, because it might block the simulation of the rules in P (only moving alive symbols we can create substrings u for the rules in P), and because we want to collect all terminal alive symbols produced to the left of f and to move them to the right of f, by means of rules of type 4. (The construction of σ is based on a similar construction used in [2], with a different goal, for arbitrary type-0 Chomsky grammars. Thus, details concerning the correctness of the construction can also be found in [2].)

Now, the rule of type 5 can be applied if and only if: (i) a derivation in G has been correctly and completely simulated by the rules of types 1, 2, 3 (no nonterminal symbol is alive, because the symbols cd^mc have to be moved across f and they are the last to do this, hence they have to cross the whole string in the left of f; if this string contains blocks $\vdash\vdash$, or alive nonterminals, or substrings of the form $[u_1[u_2]u_3]$, then the operation is not possible), (ii) we have $m = 1$. This means that the string in L_0 corresponding to this derivation is of the form $cdcx_{i_1}x_{i_2}\ldots x_{i_k} = y_{i_1}y_{i_2}\ldots y_{i_k}$, for some $k \geq 1$ and $1 \leq i_j \leq n, 1 \leq j \leq k$. This means that $PCP(x, y)$ has a solution, which is undecidable. Therefore, the appearance of the subword $fcdc$ (hence the use of the corresponding production of σ) is not algorithmically predictable. Whether or not Sf is a matching set of axioms for σ is not decidable. □

Because of this result, we will not impose that the axiom sets we use are matching and we continue to work in the general, non-restricted case.

7 Non-M-Families

We have announced in the Introduction that no previously considered family of languages is an M-family. We will make here a list of such non-M-families.

- ICC (we have already mentioned this, as a consequence of Theorem 2).
- No finite family of languages can be an M-family, because of the following **fact**: every M-family is infinite. This can be easily seen: take the languages $A_i = V^i$, for $i \geq 1$; for every σ we have $L(\sigma, A_i) \neq L(\sigma, A_j)$ for $i \neq j$ (if, for instance, $j > i$, then $V^i \in L(\sigma, A_i) - L(\sigma, A_j)$).
- No family containing only finite languages is an M-family, because of the following **fact**: every M-family contains an infinite language. This is due to the fact that we work with non-trivial M-schemes, having at least a non-null context, and to the fact that if a production $(x, (u, v))$ can be used once, then it can be used indefinitely (the selector remains after adjoining the context). Thus, the family FIN or any subfamily of it cannot be an M-family.

- Let us denote by ICC_k the family of languages which can be generated by contextual grammars with at most k productions, $k \geq 1$. No family ICC_k, $k \geq 1$, is an M-family, because these families are not closed under union. To see this, let us consider the languages

$$L_1 = \bigcup_{i=1}^{k} (ab^i a)^+,$$
$$L_2 = (ab^{i+1}a)^+.$$

  We have $L_1 \in ICC_k$ (it is generated by the grammar $G = (\{a, b\}, \{ab^i a \mid 1 \leq i \leq k\}, \{(ab^i a, (\lambda, ab^i a)) \mid 1 \leq i \leq k\})$) and $L_2 \in ICC_1$, but $L_1 \cup L_2 \notin ICC_k$, which is easy to be seen.

- REG and any family including REG is not an M-family, because for every M-family $\mathcal{F}_M(\sigma)$ there is $k \geq 1$ (the number of productions in σ) such that $\mathcal{F}_M(\sigma) \subseteq ICC_k$. On the other hand, for every $k \geq 1$, there are languages $L \in REG$ such that $L \notin ICC_k$.

- The family of one-letter regular languages is not an M-family. Assume the contrary, and consider an M-scheme $\sigma = (\{a\}, P)$ such that $\mathcal{F}_M(\sigma) = REG \cap 2^{a^*}$. Let $m = \max\{|uv| \mid (x, (u, v)) \in P\}$. All languages in $\mathcal{F}_M(\sigma)$ have the *m-bounded growth property*: for any string x in a language there is a string y in the same language of the lenght differing by at most m from the length of x. Such a property does not hold for the language $L = (a^{m+1})^+$, hence this language cannot be in $\mathcal{F}_M(\sigma)$.

Although easy to prove, these assertions are significant for our approach: the property of being an M-family is not related to other properties of languages than the syntactic similarity as defined by the M-schemes.

8 Decidabilty Results

As for operations with languages, it is expected that certain problems which can be formulated for languages in the same M-family have different answers than for the general case, when we refer to arbitrary contextual languages.

This is indeed the case with the inclusion (and probably the equivalence) problem:

Theorem 5. (i) *The inclusion is undecidable for languages in ICC, but* (ii) *it is decidable for languages in any M-family.*

Proof. (i) Let us consider again the contextual grammar G given at the end of Section 2, as well as the grammar $G' = (\{a, b, c\}, \{c\}, P)$, with the productions

$$(c, (x_i, mi(y_i)),\ 1 \leq i \leq n,$$

where $x = (x_1, \ldots, x_n), y = (y_1, \ldots, y_n)$ are two n-tuples of non-empty strings over $\{a, b\}$. It is easy to see that $L(G')$ contains strings of the form $wc\ mi(w)$ if and only if the Post Correspondence Problem for x, y has a solution; if $PCP(x, y)$ has no solution, then all strings in $L(G')$ are of the form wcz with $w \neq mi(z)$.

Remember that $L(G) = \{wcz \mid w, z \in \{a, b\}^+, w \neq mi(z)\}$. Therefore, $L(G') \subseteq L(G)$ if and only if $PCP(x, y)$ has no solution, which is not decidable.

(ii) Consider an M-scheme $\sigma = (V, P)$ and take two contextual grammars $G_1 = (V, A_1, P), G_2 = (V, A_2, P)$ derived from it. It is clear that, because G_1, G_2 have the same productions, $L(G_1) \subseteq L(G_2)$ if and only if $A_1 \subseteq L(G_2)$. Because $\mathcal{F}_M(\sigma) \subseteq ICC \subset CS$, the membership is decidable for the language $L(G_2)$. The set A_1 is finite, hence we can check algorithmically whether or not $A_1 \subseteq L(G_2)$. □

Corollary 1. *The equivalence is decidable for languages in the same M-family.*

Proof. The decidability of the inclusion implies the decidability of the equivalence. □

It is of interest to note that the decidability of the equivalence for arbitrary languages in ICC is an *open problem* ([8]).

9 Further Problems

The inclusion problem investigated above is related to the following problem of a more general interest.

Consider the following M-scheme:

$$\sigma = (\{a, b\}, \{(\lambda, (\lambda, \beta)) \mid \beta \in \{a, b\}\}).$$

As we have mentioned in Section 6, the family $\mathcal{F}_M(\sigma)$ is infinite, but, on the other hand, every language in this family is included in $L(\sigma, \{\lambda\}) = \{a, b\}^*$. We say that the M-family $\mathcal{F}_M(\sigma)$ has an *attractor* (the language $\{a, b\}^*$): a language in the family which includes all languages in the family.

Such a property is not valid for the next M-scheme:

$$\sigma' = (\{a, b\}, \{(ab, (ab))\}).$$

All languages $L(\sigma', \{(ab)^i\}) = \{a^n b^n \mid n \geq 1\}^i$ are in $\mathcal{F}_M(\sigma')$, but there is no language in this family which includes all these languages. This suggests the following questions:

1. Is the property of having an attractor decidable ? When an attractor exists, can it be effectively constructed ?

In view of the fact that the union of two languages in an M-family is also in the family, in general, that given two sets of axioms A_1, A_2 such that $A_1 \subseteq A_2$, the language associated with A_1 is included in that associated with A_2, it is of interest

to consider a notion of the following type: for an M-family $\mathcal{F}_M(\sigma)$ we say that a family $\mathcal{L}$ is a *cover* of $\mathcal{F}_M(\sigma)$ if (i) $\mathcal{L} \subseteq \mathcal{F}_M(\sigma)$, (ii) for all two different languages $L_1, L_2 \in \mathcal{L}$ none of the inclusions $L_1 \subseteq L_2, L_2 \subseteq L_1$ holds, (iii) for every $L_1 \in \mathcal{F}_M(\sigma)$ there is $L_2 \in \mathcal{L}$ such that $L_1 \subseteq L_2$, and (iv) $\mathcal{L}$ is maximal with these properties.

Of course, if $\mathcal{F}_M(\sigma)$ has an attractor L, then $\{L\}$ is a cover for $\mathcal{F}_M(\sigma)$ and it is the unique one.

2. Given an M-scheme σ, is the cover of $\mathcal{F}_M(\sigma)$ unique ?

3. Can a cover of $\mathcal{F}_M(\sigma)$ (the unique one, if this is the case) be effectively constructed ?

4. What properties has a cover (in comparison with the properties of $\mathcal{F}_M(\sigma)$) ?

5. Because the cover can contain only one language – the case of M-families having an attractor – it follows that a cover of an M-family is not necessarily an M-family. Are there M-families for which the covers are M-families, too ?

6. Returning to the matching sets of axioms, one may also ask what happens when one considers only M-families defined by using matching sets of axioms. This restricts the arguments in many of the proofs in the previous sections.

All the investigations above as well as the previous problems can be considered for other variants of contextual grammars and associated schemes. For instance, we can consider M-schemes with maximal (or minimal) use of selectors, as introduced in [6]. In view of the relevance of these variants of contextual grammars for natural language study (see, e.g., [5]), this case deserves a special attention. In general, the usefulness of M-families (and of other X-families) for linguistics is a problem which should be addressed.

Recently ([7]), a combination of Chomsky and Marcus grammars was proposed, under the name of MC-grammars: take an alphabet (separated into a nonterminal and a terminal alphabet), a finite set of axioms, a finite set of rewriting rules (as in Chomsky grammars) and a finite set of context adjoining rules (as in a Marcus grammar). A language is generated by applying both types of rules, either nondeterministically chosen or in a prescribed sequence.

This idea can be extended also to X-families, leading to XY-families: first, consider hybrid grammars, mixing productions of types X and Y for all possibilities X, Y in the set {C, M, P, L, H}, then consider XY-schemes (such grammars without a set of axioms), and then define the associated families of languages. (Such hybrid grammars seem to be well suited for linguistic interpretations, taking into account the two main components of natural languages, the syntactic and the semantic one, and other similar two-level approaches to natural languages.)

All problems considered above can be formulated also for XY-families (a level where, presumably, they are more difficult than for X-families).

10 Conclusions

We have here only introduced the notion of an X- and XY-family, as an attempt to study families of syntactically similar languages, we have formulated several basic research topics for this area, and we have considered as a case-study the M-families. (Of course, the problems investigated for M-families, such as the matching questions for axioms, can be formulated for any type of X- and XY-family.)

We appreciate that this area of research is very fruitful, well motivated, and connected with many known problems and results in formal language theory, but also raises a wealth of new problems. In short, we believe that the study of X- and XY-families deserves a more intensive research.

References

[1] A. B. Cremers, S. Ginsburg, Context-free grammar forms, *J. Computer System Sci.*, 11 (1975), 86 – 116.

[2] A. Ehrenfeucht, Gh. Păun, G. Rozenberg, On representing recursively enumerable languages by internal contextual languages, *Theoretical Computer Sci.*, 182 (1997).

[3] A. Manaster Ramer, *Uses and Misuses of Mathematics in Linguistics*, X Congreso de Lenguajes Naturales y Lenguajes Formales, Sevilla, 1994 (unpublished manuscript).

[4] S. Marcus, Contextual grammars, *Rev. Roum. Math. Pures Appl.*, 14 (1969), 1525 – 1534.

[5] S. Marcus, C. Martín-Vide, Gh. Păun, Contextual grammars as generative models of natural languages, *Computational Linguistics*, to appear.

[6] C. Martín-Vide, A. Mateescu, J. Miquel-Verges, Gh. Păun, Internal contextual grammars: minimal, maximal, and scattered use of selectors, *Proc. of the Fourth Bar-Ilan Symp. on Foundations of AI, BISFAI '95* (M. Koppel, E. Shamir, eds.), The AAAI Press, Menlo Park, Ca., 1996, 159 – 168.

[7] C. Martín-Vide, A. Mateescu, Gh. Păun, Hybrid grammars; The Chomsky-Marcus case, *Bulletin of the EATCS*, to appear.

[8] Gh. Păun, *Marcus Contextual Grammars*, Kluwer Academic Publ., Dordrecht, Boston, London, 1997.

[9] Gh. Păun, X. M. Nguyen, On the inner contextual grammars, *Rev. Roum. Math. Pures Appl.*, 25 (1980), 641 – 651.

[10] Gh. Păun, G. Rozenberg, A. Salomaa, *DNA Computing. New Computing Paradigms*, in preparation.

[11] Gh. Păun, A. Salomaa, Families generated by grammars and L systems, chapter 12 in vol. 1 of [13], 811 – 861.

[12] G. Rozenberg, A. Salomaa, *The Mathematical Theory of L Systems*, Academic Press, New York, 1973.

[13] G. Rozenberg, A. Salomaa (eds.), *Handbook of Formal Languages*, 3 volumes, Springer-Verlag, Berlin, Heidelberg, 1997.

[14] D. Wood, *Grammar and L Forms: An Introduction*, *Lect. Notes Computer Sci.* 91, Springer-Verlag, Berlin, 1980.