International Journal of Foundations of Computer Science
Vol. 21, No. 5 (2010) 799–815

DOI: 10.1142/S0129054110007568

BOOLEAN GRAMMARS AND GSM MAPPINGS*

TOMMI LEHTINEN
Department of Mathematics, University of Turku, Turku FI–20014, Finland, and
Turku Centre for Computer Science
tojleht@utu.fi

ALEXANDER OKHOTIN†
Academy of Finland, and
Department of Mathematics, University of Turku, Turku FI–20014, Finland
alexander.okhotin@utu.fi

Received 1 July 2008
Accepted 20 February 2010
Communicated by Erzsébet Csuhaj-Varjú and Zoltán Ésik

It is proved that the language family generated by Boolean grammars is effectively closed under injective gsm mappings and inverse gsm mappings (where gsm stands for a generalized sequential machine). The same results hold for conjunctive grammars, unambiguous Boolean grammars and unambiguous conjunctive grammars.

Keywords: Boolean grammars; conjunctive grammars; language equations; closure properties; gsm mappings; homomorphisms.

1991 Mathematics Subject Classification: 68Q45

1. Introduction

Boolean grammars [13] are an extension of the context-free grammars, in which the rules may contain explicit Boolean operations. The extended expressive power and the intuitive clarity of the new operations in Boolean grammars make them a much more powerful tool for specifying languages than the context-free grammars. Another important fact is that the main context-free parsing algorithms, such as the Cocke–Kasami–Younger [13], the recursive descent [16] and the generalized LR [14], can be extended to Boolean grammars without increasing their computational complexity [15].

Though the Boolean grammars easily inherit many good practical properties of context-free grammars, their theoretical properties present a greater challenge to the researcher. Up to date, no methods of proving that any particular language cannot be generated by a Boolean grammar are known, except for those based upon computational complexity, as all languages generated by Boolean grammars belong

*Research supported by the Academy of Finland under grant 118540.
†Corresponding author.

to $\mathsf{DTIME}(n^3) \cap \mathsf{DSPACE}(n)$ [13]. Accordingly, the family of languages they generate could not be separated from this complexity-theoretic upper bound.

Likewise, quite little progress has been made on the closure properties of the languages generated by Boolean grammars. This is the question of whether applications of certain operations to these languages always yield languages generated by Boolean grammars. Boolean grammars are trivially closed under Boolean operations and concatenation, since all these operations are included in their formalism. The same can be said of the Kleene star, which can be expressed by iterating a single nonterminal, as in the context-free case.

Among the standard operations on languages are *homomorphisms* and the more general *gsm mappings*: these are mappings $M : \Sigma^* \to \Gamma^*$ implemented by deterministic transducers (generalized sequential machines, gsm). As shown by Ginsburg and Rose [3], context-free languages are closed under gsm mappings; an easy proof of this fact given by Harrison [5] is by simulating a gsm within a pushdown automaton. On the contrary, the languages generated by Boolean grammars are already not closed under homomorphisms: in fact, all recursively enumerable languages can be obtained as homomorphic images of languages generated by a subclass of Boolean grammars, the *linear conjunctive grammars* [2, 12]. Their closure under non-erasing homomorphisms remains an open problem.

For a gsm mapping $M : \Sigma^* \to \Gamma^*$, the corresponding *inverse gsm mapping* is defined as $M^{-1}(L) = \{w \in \Sigma^* \mid M(w) \in L\}$ for every $L \subseteq \Gamma^*$. It is known from Ginsburg and Rose [3] that context-free languages are closed under inverse gsm mappings. This argument was adapted to unambiguous context-free languages by Ginsburg and Ullian [4]. More accessible proofs of these results based upon pushdown automata were given by Harrison [5]. An examination of this argument shows that it also applies to linear context-free languages, which are consequently closed under inverse gsm mappings. The aforementioned linear conjunctive languages are closed under this operation by an argument presented by Ibarra and Kim [7], done in terms of trellis automata [1].

This paper investigates the closure of Boolean grammars under *injective* gsm mappings and under inverse gsm mappings. In Section 4 it is established that for every injective gsm mapping $M : \Sigma^* \to \Gamma^*$ and for every Boolean grammar G over an alphabet Σ, the language $M(L(G))$ is generated by a Boolean grammar.

The closure under inverse gsm mappings means that for every gsm mapping $M : \Sigma^* \to \Gamma^*$ (not necessarily injective) and for every Boolean grammar G over Γ, the language $M^{-1}(L(G))$ of pre-images of words generated by G is generated by a Boolean grammar. This result is first established for the special case of homomorphisms known as *weak codings*, which is done in Section 5 by an explicit construction. Then the general case is proved by representing the language $M^{-1}(L(G))$ as a combination of an injective gsm mapping, an inverse weak coding and an intersection with a regular language. The overall proof is constructive.

Furthermore, if the Boolean grammar G is *unambiguous* [17], then all the above constructions also produce unambiguous grammars, and if G does not use negation (that is, it is a *conjunctive grammar* [11]), then negation can be eliminated in the constructed grammar. So, in effect, each of the four families of languages generated by (unambiguous) conjunctive and (unambiguous) Boolean grammars are proved to be closed under injective gsm mappings and under inverse gsm mappings.

2. Definition of Boolean Grammars

Definition 1 ([13]) *A Boolean grammar is a quadruple $G = (\Sigma, N, P, S)$, where Σ and N are disjoint finite nonempty sets of terminal and nonterminal symbols respectively; P is a finite set of* rules *of the form*

$$A \to \alpha_1 \& \ldots \& \alpha_m \& \neg\beta_1 \& \ldots \& \neg\beta_n, \tag{1}$$

where $m + n \geqslant 1$, $\alpha_i, \beta_j \in (\Sigma \cup N)^*$; $S \in N$ is the start symbol of the grammar.

For each rule (1), the terms α_i and $\neg\beta_j$ (for all i, j) are called *conjuncts*, *positive* and *negative* respectively. A conjunct with any sign is denoted $\pm\gamma$. The entire right-hand side of a rule (1) will sometimes be denoted by φ, and the whole rule by $A \to \varphi$.

A Boolean grammar is called a *conjunctive grammar* [11] if negation is never used, that is, $n = 0$ for every rule (1). It is a *context-free grammar* if neither negation nor conjunction are allowed, that is, $m = 1$ and $n = 0$ for each rule. Another important particular case of Boolean grammars is formed by *linear conjunctive grammars*, in which every rule is a conjunction of terms of the form uBv or w, with $u, v, w \in \Sigma^*$ and $B \in N$. Linear conjunctive grammars are equal in power to *linear Boolean grammars* with conjuncts $\pm uBv$ or w, as well as to *trellis automata*, also known as one-way real-time cellular automata [1, 12].

Intuitively, a rule (1) of a Boolean grammar can be read as follows: every word w over Σ that satisfies each of the syntactical conditions represented by $\alpha_1, \ldots, \alpha_m$ and none of the syntactical conditions represented by $\beta_1, \ldots, \beta_m$ therefore satisfies the condition defined by A. A formal definition of the language generated by a Boolean grammar can be given in several different ways [10, 13], that ultimately yield the same class of languages. We shall use the most straightforward one of these definitions, which begins with the interpretation of a grammar as a system of equations with formal languages as unknowns:

Definition 2. *Let $G = (\Sigma, N, P, S)$ be a Boolean grammar. The system of language equations associated with G is a resolved system of language equations over Σ in variables N, in which the equation for each variable $A \in N$ is*

$$A = \bigcup_{A \to \alpha_1 \& \ldots \& \alpha_m \& \neg\beta_1 \& \ldots \& \neg\beta_n \in P} \left[\bigcap_{i=1}^{m} \alpha_i \cap \bigcap_{j=1}^{n} \overline{\beta_j} \right] \tag{2}$$

Each instance of a symbol $a \in \Sigma$ in such a system defines a constant language $\{a\}$, while each empty word denotes a constant language $\{\varepsilon\}$. A solution of such a system

is a vector of languages $(\ldots, L_C, \ldots)_{C \in N}$*, such that the substitution of* L_C *for* C*, for all* $C \in N$*, turns each equation (2) into an equality.*

Now the following restriction is imposed upon these equations, so that their solutions can be used to define the languages generated by grammars:

Definition 3. *Let* $G = (\Sigma, N, P, S)$ *be a Boolean grammar, let (2) be the associated system of language equations and assume that it has a unique solution* $(\ldots, L_C, \ldots)_{C \in N}$ *with* $L_C \subseteq \Sigma^*$*. Further assume that for every subword-closed finite language* $M \subset \Sigma^*$ *(that is,* $xyz \in M$ *implies* $y \in M$*), the vector of languages* $(\ldots, L_C \cap M, \ldots)_{C \in N}$ *is the unique vector, such that a substitution of* L_C *for* C*, for each* $C \in N$*, turns every equation (2) into an equality modulo intersection with* M*.*

Then, for every $A \in N$*, the language* $L_G(A)$ *is defined as* L_A*, while the language generated by the grammar is* $L(G) = L_G(S) = L_S$*.*

Example 1. *The following Boolean grammar generates the language* $\{a^m b^n c^n \mid m, n \geqslant 0, m \neq n\}$*:*

$$\begin{aligned} S &\to AB\&\neg DC \\ A &\to aA \mid \varepsilon \\ B &\to bBc \mid \varepsilon \\ C &\to cC \mid \varepsilon \\ D &\to aDb \mid \varepsilon \end{aligned}$$

The rules for the nonterminals A, B, C and D are context-free. Then the propositional connectives in the rule for S specify the following combination of the conditions given by AB and DC:

$$\underbrace{\{a^n b^m c^m \mid m, n \geqslant 0, m \neq n\}}_{L(S)} = \{a^i b^j c^k \mid j = k \text{ and } i \neq j\} = \\ = \underbrace{\{a^i b^j c^k \mid j = k\}}_{L(AB)} \cap \overline{\underbrace{\{a^i b^j c^k \mid i = j\}}_{L(DC)}}.$$

Several examples of Boolean grammars for such languages as $\{a^n b^n c^n \mid n \geqslant 0\}$ $\{wcw \mid w \in \{a, b\}^*\}$ and $\{ww \mid w \in \{a, b\}^*\}$ can be found in a recent survey [15]. Conjunctive grammars for non-regular languages have also been developed, starting with $\{a^{4^n} \mid n \geqslant 0\}$ [8] and proceeding with some general representability results [9].

There exists an unambiguous subclass of Boolean grammars, which generalizes unambiguous context-free grammars.

Definition 4. *A Boolean grammar* $G = (\Sigma, N, P, S)$ *is unambiguous if*

(I) Different rules for every single nonterminal A *generate disjoint languages, that is, for every word* w *there exists at most one rule*

$$A \to \alpha_1 \& \ldots \& \alpha_m \& \neg\beta_1 \& \ldots \& \neg\beta_n,$$

such that $w \in L_G(\alpha_1) \cap \ldots \cap L_G(\alpha_m) \cap \overline{L_G(\beta_1)} \cap \ldots \cap \overline{L_G(\beta_n)}$*.*

(II) All concatenations are unambiguous, that is, for every conjunct $\pm s_1 \dots s_\ell$ *and for every word* w *there exists at most one factorization* $w = u_1 \dots u_\ell$*, such that* $u_i \in L_G(s_i)$ *for all* i*.*

While the languages generated by Boolean grammars can be recognized in cubic time [13] and no better upper bound is known, unambiguous Boolean grammars allow square-time parsing [17]. However, no proofs of inherent ambiguity of any languages generated by Boolean grammars are known. It is known that all linear conjunctive languages have unambiguous grammars.

The relation between the families of languages generated by Boolean grammars (*Bool*), conjunctive grammars (*Conj*) and linear conjunctive grammars (*LinConj*), their unambiguous variants (*UnambBool* and *UnambConj*), as well as other common families of formal languages, is shown in Fig. 1 [17], in which arrows without a question mark represent proper inclusions, while a question mark denotes an inclusion not known to be proper. A dotted line between two families indicates that they are incomparable. The rest of the classes in the figure are regular (*Reg*), linear context-free (*LinCF*), context-free (*CF*) and deterministic context-sensitive languages (*DetCS*).

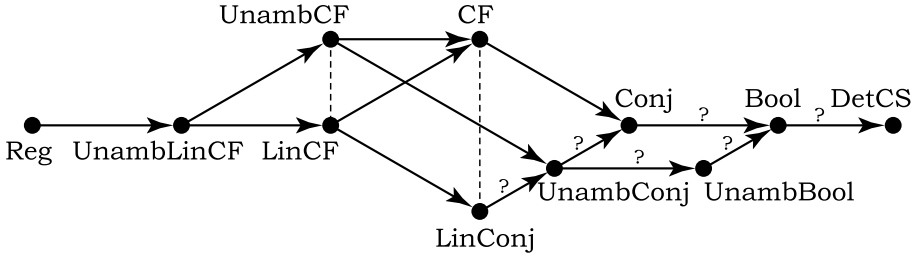

Fig. 1. The hierarchy of language families.

The following normal form for Boolean grammars, which generalizes the Chomsky normal form for the context-free grammars, is known.

Definition 5. *A Boolean grammar* $G = (\Sigma, N, P, S)$ *is in the binary normal form if every rule in* P *is of the form*

$$A \to B_1C_1\& \dots \&B_mC_m\&\neg D_1E_1\& \dots \&\neg D_nE_n\&\neg\varepsilon \quad (m \geqslant 1, n \geqslant 0)$$

$$A \to a$$

$$S \to \varepsilon \quad \text{(only if } S \text{ does not appear in right-hand sides of rules)}$$

Similarly, a conjunctive grammar in the binary normal form has all rules of the form $A \to B_1C_1\& \dots \&B_mC_m$ *with* $m \geqslant 0$*,* $A \to a$ *or* $S \to \varepsilon$*.*

These three types of rules shall sometimes be referred to as *long*, *short* and *epsilon* rules, respectively. Every grammar of this form is well-defined in the sense

of Definition 3, as well as according to other definitions of Boolean grammars [10, 13].

Proposition 1 ([11, 13, 17]) *For every Boolean grammar defined according to Definition 3 there exists and can be effectively constructed a Boolean grammar in the binary normal form generating the same language. Furthermore, if the given grammar is unambiguous, then so is the constructed grammar. The same holds for conjunctive and unambiguous conjunctive grammars.*

In the following, whenever a Boolean grammar is mentioned, it is assumed to be well-defined in the sense of Definition 3.

3. Generalized Sequential Machines

Generalized sequential machines are deterministic finite transducers that read a single symbol and write an arbitrary word at every step. They may be defined with or without accepting states. This paper uses a more general definition with accepting states, in which case a gsm computes a partial mapping.

Definition 6. *A generalized sequential machine (gsm) is a septuple* $M = (\Sigma, \Gamma, Q, q_0, \delta, \lambda, F)$*, where*

- Σ *is a finite nonempty* input alphabet*,*
- Γ *is a finite nonempty* output alphabet*,*
- Q *is a finite nonempty set of* states*,*
- $q_0 \in Q$ *is the* initial state*,*
- $\delta\colon Q \times \Sigma \to Q$ *is the* transition function*,*
- $\lambda\colon Q \times \Sigma \to \Gamma^*$ *is the* output function*, and*
- $F \subseteq Q$ *is the set of* accepting states*.*

The functions δ *and* λ *are extended to* $Q \times \Sigma^*$ *in the usual way, as* $\delta(q, \varepsilon) = q$*,* $\delta(q, aw) = \delta(\delta(q, a), w)$ *and as* $\lambda(q, \varepsilon) = \varepsilon$*,* $\lambda(q, aw) = \lambda(q, a)\lambda(\delta(q, a), w)$*. The gsm* M *computes a partial function* $M\colon \Sigma^* \to \Gamma^*$*, where* $M(w) = \lambda(q_0, w)$ *if* $\delta(q_0, w) \in F$*, and* $M(w)$ *is undefined otherwise.*

A state q of a gsm M is said to be *useful* if it is reachable from the initial state and there is a path from it to some accepting state, that is, $\delta(q_0, u) = q$ and $\delta(q, v) \in F$ for some $u, v \in \Sigma^*$. Denote by $\widehat{Q} \subseteq Q$ the set of useful states.

A gsm M is called *injective* if the function it computes is injective, that is, if $M(w) = M(w')$ implies $w = w'$. For any injective gsm, the structure of its transitions that output ε (erasing transitions) has the following property:

Lemma 1. *Let* M *be an injective gsm. Consider the graph* (V, E) *of its erasing transitions, with* $V = \widehat{Q}$ *and* $E = \{(q, q') \mid \exists a \in \Sigma : \delta(q, a) = q', \lambda(q, a) = \varepsilon\}$*. Then this graph is acyclic.*

Proof. Suppose the contrary, that is, there is a cycle passing through some state $q \in \widehat{Q}$. Then there is a word $w \neq \varepsilon$ with $\delta(q, w) = q$ and $\lambda(q, w) = \varepsilon$. Since q is

useful, there exist words $u, v \in \Sigma^*$ with $\delta(q_0, u) = q$ and $\delta(q, v) \in F$. Therefore, $\delta(q_0, uv) = \delta(q_0, uwv) \in F$ and $\lambda(q_0, uv) = \lambda(q_0, uwv)$, that is, $M(uv) = M(uwv)$, where $uv \neq uwv$. This contradicts the assumption that M is injective. □

Since the graph is acyclic, there is a bound on the length of its paths. In terms of the given injective gsm, every path in this graph corresponds to a sequence of erasing transitions. So it is known that the number of such consecutive transitions is bounded, and for every state $q \in \widehat{Q}$ the following bounds can be defined:

$$\mathrm{D}(q) = \max\{\, |w| \mid \delta(q, w) \in \widehat{Q}, \lambda(q, w) = \varepsilon \,\},$$

$$\mathfrak{D}(q) = \max\{\, |w| \mid \exists q' : \delta(q', w) = q, \lambda(q', w) = \varepsilon \,\}.$$

In other words, $\mathrm{D}(q)$ ($\mathfrak{D}(q)$, respectively) is the greatest number of erasing transitions starting from the state q (before entering the state q, respectively), which is well-defined because of Lemma 1.

For any gsm $M = (\Sigma, \Gamma, Q, q_0, \delta, \lambda, F)$, define the corresponding inverse gsm mapping as $M^{-1}(L) = \{w \in \Sigma^* \mid M(w) \in L\}$. Note that this is not an inverse in a strict mathematical sense, since $M^{-1}(M(L))$ is not necessarily equal to L, but rather a set of all pre-images of all words in L.

4. Injective gsm Mappings

Context-free languages are closed under gsm mappings, which can be naturally proved using the pushdown automaton representation. Already for conjunctive languages and homomorphisms the corresponding statement is false [11]. The argument from the context-free case does not work, not only for the lack of any automaton representation of conjunctive grammars, but also due to more fundamental properties of intersection. While homomorphism respects union and concatenation in the sense that $h(K \cup L) = h(K) \cup h(L)$ and $h(KL) = h(K) \cdot h(L)$, it does not respect intersection: $h(K \cap L)$ is not necessarily equal to $h(K) \cap h(L)$.

However, all *injective* functions respect intersection, and this allows proving the closure of these families of languages under injective gsm mappings.

Theorem 1. *For every generalized sequential machine M implementing a partial injective function $M : \Sigma^* \to \Gamma^*$ and for every Boolean (conjunctive, unambiguous Boolean, unambiguous conjunctive) grammar G over Σ, there exists and can be effectively constructed a Boolean (conjunctive, unambiguous Boolean, unambiguous conjunctive) grammar over Γ generating the language $M(L(G))$.*

Let $M = (\Sigma, \Gamma, Q, q_0, \delta, \lambda, F)$ be an injective gsm, and let $\widehat{Q} \subseteq Q$ be the set of its useful states. Assume $G = (\Sigma, N, P, S)$ is in the binary normal form without the rule $S \to \varepsilon$. If ε is generated by G, this would only mean that $M(\varepsilon) = \varepsilon$ should be in $M(L(G))$ (provided that $q_0 \in F$), and it will suffice to add a single rule to the final grammar to complete the construction.

Construct $G' = (\Gamma, N'_1 \cup N'_2 \cup \{S'\}, P', S')$ with $N'_1 = \widehat{Q} \times N \times \widehat{Q}$ and $N'_2 = \widehat{Q} \times N \times N \times \widehat{Q}$. For all $\varphi = B_1C_1\& \ldots \&B_mC_m\&\neg D_1E_1\& \ldots \&\neg D_nE_n\&\neg\varepsilon$ appearing as the right-hand side of a rule in P, denote

$$(q, \varphi, q') = \mathop{\&}_{1 \leqslant i \leqslant m} (q, B_i, C_i, q') \& \mathop{\&}_{1 \leqslant j \leqslant n} \neg(q, D_j, E_j, q') \& \neg\varepsilon \tag{3}$$

The set of rules P' of the new grammar is comprised of the following rules:

$$(q, A, q') \to (q, \varphi, q') \quad (A \to \varphi \in P), \tag{4a}$$

$$(q, B, C, q') \to (q, B, q'')(q'', C, q') \quad (q'' \in \widehat{Q}), \tag{4b}$$

$$(q, A, \delta(q, a)) \to \lambda(q, a) \quad (A \to a \in P;\ q, \delta(q, a) \in \widehat{Q};\ \lambda(q, a) \neq \varepsilon) \tag{4c}$$

$$(q, A, q') \to \varepsilon \quad (\exists w \in L_G(A) : \lambda(q, w) = \varepsilon,\ \delta(q, w) = q') \tag{4d}$$

$$S' \to (q_0, S, q_f) \quad (q_f \in F) \tag{4e}$$

Now the task is to prove that G' is well-defined, that it actually generates $M(L(G))$, and that it preserves unambiguity. The case of a conjunctive G also requires some remarks. These properties of G' are established in the statements below:

Claim 1. *The system (2) corresponding to G' has a strongly unique solution.*

Proof. It has to be proved that the solution modulo every finite subword-closed $K \subseteq \Gamma^*$ is unique. The proof is done by induction on $|K|$.

Basis: $K = \{\varepsilon\}$. The unique solution L modulo K is defined as follows. For nonterminals in N'_1, $L_{(q,A,q')} = \{\varepsilon\}$ if there is a rule $(q, A, q') \to \varepsilon$ in P', and $L_{(q,A,q')} = \varnothing$ otherwise. For nonterminals in N'_2, $L_{(q,B,C,q')} = \{\varepsilon\}$ if there exists a state q'' with the rules $(q, B, q'') \to \varepsilon$ and $(q'', C, q') \to \varepsilon$, and $L_{(q,B,C,q')} = \varnothing$ otherwise.

Induction step: Assume that the solution modulo K is unique, and let $x \notin K$ be a word with all its proper subwords belonging to K. It has to be proved that the solution modulo $K \cup \{x\}$ is also unique. Suppose it is not, and let L and L' be two different solutions modulo $K \cup \{x\}$, with $x \in L_X$ and $x \notin L'_X$ for some $X \in N$. The nonterminal X is chosen with the minimal $\mathrm{D}(q) + \mathfrak{C}(q')$.

If $X = (q, A, q') \in N_1$, then $x \in L_{(q,A,q')}$ is generated either by a rule $(q, A, \delta(q, a)) \to \lambda(q, a)$, which means that $x \in L'_{(q,A,q')}$ by the same rule, or by a long rule $(q, A, q') \to (q, \varphi, q')$, in which case the solutions differ on some nonterminal $(q, B, C, q') \in N_2$ as well. This leaves only the second case to consider.

If $X = (q, B, C, q') \in N_2$, then there is a rule $(q, B, C, q') \to (q, B, q'')(q'', C, q')\&\neg\varepsilon$ with $x \in L_{(q,B,q'')}L_{(q'',C,q')}$. Let $x = x_1x_2$, where $x_1 \in L_{(q,B,q'')}$ and $x_2 \in L_{(q'',C,q')}$. If $x_1, x_2 \in K$, then, by the induction hypothesis, $x_1 \in L'_{(q,B,q'')}$ and $x_2 \in L'_{(q'',C,q')}$, and thus $x \in L'_{(q,B,C,q')}$.

Otherwise, let $x_1 = x$ and $x_2 = \varepsilon$. Then $\varepsilon \in L_{(q'',C,q')}$ by a rule $(q'', C, q') \to \varepsilon$, and such a rule exists only if there is a nonempty word $w \in L_G(C)$ with $\lambda(q'', w) = \varepsilon$

and $\delta(q'', w) = q'$. Then $\mathfrak{C}(q') \geqslant \mathfrak{C}(q'') + |w|$, and therefore $\mathfrak{C}(q'') < \mathfrak{C}(q')$. Consider the variable (q, B, q''): since $\mathrm{D}(q) + \mathfrak{C}(q'') < \mathrm{D}(q) + \mathfrak{C}(q')$, by the assumption, the solutions L and L' do not differ on this variable and thus $x \in L_{(q,B,q'')} = L'_{(q,B,q'')}$. On the other hand, by the basis of induction, $\varepsilon \in L_{(q'',C,q')}$ is equivalent to $\varepsilon \in L'_{(q'',C,q')}$. Therefore, $x \in L'_{(q,B,q'')} L'_{(q'',C,q')} \subseteq L'_{(q,B,C,q')}$, a contradiction.

In the other case of $x_1 = \varepsilon$ and $x_2 = x$, by a similar argument, $\mathrm{D}(q'') < \mathrm{D}(q)$. Then for the variable (q'', C, q') it holds that $\mathrm{D}(q'') + \mathfrak{C}(q') < \mathrm{D}(q) + \mathfrak{C}(q')$, so $L'_{(q'',C,q')} = L_{(q'',C,q')}$ by assumption and $x \in L'_{(q'',C,q')}$. At the same time, $\varepsilon \in L'_{(q,B,q'')}$ according to the basis, and $x \in L'_{(q,B,q'')} L'_{(q'',C,q')}$ yields the same contradiction as above. □

Claim 2. *$x \in L_{G'}\big((q, A, q')\big)$ if and only if $x = \lambda(q, w)$ for some $w \in L_G(A)$ with $\delta(q, w) = q'$.*

Proof. The proof is an induction on the lexicographically ordered pairs $(|x|, \mathrm{D}(q) + \mathfrak{C}(q'))$. Note that the second component is well-defined by Lemma 1.

Basis: If $x = \varepsilon$, then it can be generated only by a rule $(q, A, q') \to \varepsilon$, and the definition of these rules implies that there is a word $w \in L_G(A)$ with $\delta(q, w) = q'$ and $\lambda(q, w) = \varepsilon$.

Induction step: Let $|x| \geqslant 1$. Assuming the induction hypothesis, let us first prove the following two statements:

Claim 2.1. *$x \in L_{G'}\big((q, B, C, q')\big)$ if and only if $x = \lambda(q, w)$ for some $w \in L_G(BC)$ with $\delta(q, w) = q'$.*

Proof. To prove this, first assume that $x \in L_{G'}\big((q, B, C, q')\big)$. Then $x = yz$ with $y \in L_{G'}\big((q, B, q'')\big)$ and $z \in L_{G'}\big((q'', C, q')\big)$ for some intermediate state q''. If $y, z \neq \varepsilon$, then, by the induction hypothesis, $y = \lambda(q, u)$ and $z = \lambda(q'', v)$ for some words $u \in L_G(B)$ and $v \in L_G(C)$ with $\delta(q, u) = q''$ and $\delta(q'', v) = q'$. Then $x = \lambda(q, uv)$ with $uv \in L_G(BC)$ and $\delta(q, uv) = q'$.

If $y = \varepsilon$ and $z = x$, then, by the construction (4d), there is such a word $u \in L_G(B)$, that $\lambda(q, u) = \varepsilon$ and $\delta(q, u) = q''$. Then $\mathrm{D}(q) \geqslant |u| + \mathrm{D}(q'')$, and since u is not the empty word, $\mathrm{D}(q) > \mathrm{D}(q'')$. Thus the induction hypothesis is applicable to $x \in L_{G'}\big((q, C, q'')\big)$. The hypothesis asserts that there is a word $v \in L_G(C)$, with $\lambda(q'', v) = x$ and $\delta(q'', u) = q'$. Again, $x = \lambda(q, uv)$ with $uv \in L_G(BC)$ and $\delta(q, uv) = q'$.

The case of $y = x$ and $z = \varepsilon$ is handled similarly.

Assume conversely that $x = \lambda(q, w)$ for some $w \in L_G(BC)$ with $\delta(q, w) = q'$. Then $w = uv$ for some nonempty words $u \in L_G(B)$ and $v \in L_G(C)$. Let $q'' = \delta(q, u)$ be the intermediate state. If $\lambda(q, u) = \varepsilon$, then $\mathrm{D}(q) > \mathrm{D}(q'')$ and if $\lambda(q, v) = \varepsilon$, then $\mathfrak{C}(q') > \mathfrak{C}(q'')$. In both cases the induction hypothesis is applicable to $\lambda(q, u) \in L_{G'}\big((q, B, q'')\big)$ and $\lambda(q'', v) \in L_{G'}\big((q'', C, q')\big)$, as is in the case $\lambda(q, u), \lambda(q, v) \neq \varepsilon$. Thus $\lambda(q, u) \in L_{G'}\big((q, B, q'')\big)$ and $\lambda(q'', v) \in L_{G'}\big((q'', C, q')\big)$, and consequently $x = \lambda(q, u)\lambda(q'', v) \in L_{G'}\big((q, B, C, q')\big)$, which proves Claim 2.1. □

Claim 2.2. *Let* $\varphi = B_1C_1\& \ldots \& B_mC_m \& \neg D_1E_1 \& \ldots \& \neg D_nE_n \& \neg\varepsilon$ *and* $A \to \varphi$ *be a rule in* P*, assume* $x \neq \varepsilon$*. Then* $x \in L_{G'}\big((q, \varphi, q')\big)$ *if and only if* $x = \lambda(q, w)$ *for some* $w \in L_G(\varphi)$ *with* $\delta(q, w) = q'$*.*

Proof. Recall that (q, φ, q') is a notation for a Boolean expression (3). Assume that $x \in L_{G'}\big((q, \varphi, q')\big)$; then, by (3), $x \in L_{G'}\big((q, B_i, C_i, q')\big)$ for $i = 1, \ldots, m$. By Claim 2.1, there are words $w_i \in L_G(B_iC_i)$ with $\delta(q, w_i) = q'$ and $\lambda(q, w_i) = x$. Since M is injective, all the words $w_1, \ldots, w_m$ are actually the same word, denoted by w. Furthermore, $w \notin L_G(D_jE_j)$ for $j = 1, \ldots, n$ again by Claim 2.1, so $w \in L_G(\varphi)$.

Conversely, let $x = \lambda(q, w)$ for some word $w \in L_G(\varphi)$ with $\delta(q, w) = q'$. Then, for each $i = 1, \ldots, m$, $w \in L_G(B_iC_i)$ and hence $x \in L_{G'}\big((q, B_i, C_i, q')\big)$ by Claim 2.1. Similarly, $w \notin L_G(D_jE_j)$ for $j = 1, \ldots, n$, and it has to be proved that $x \notin L_{G'}\big((q, D_j, E_j, q')\big)$. Suppose the contrary, that $x \in L_{G'}\big((q, D_j, E_j, q')\big)$. Then, by Claim 2.1, there is $\widetilde{w} \in L_G(D_jE_j)$ with $\delta(q, \widetilde{w}) = q'$ and $\lambda(q, \widetilde{w}) = x$. Since M is injective, $w = \widetilde{w}$, and hence $w \in L_G(D_jE_j)$, which contradicts the above. Therefore, $x \in L_{G'}\big((q, \varphi, q')\big)$ and Claim 2.2 is proved. □

Returning to the proof of the induction step in Claim 1, first assume that $x \in L_{G'}\big((q, A, q')\big)$. If x is generated by a rule $(q, A, \delta(q, a)) \to x$, then, by the definition of these rules, $x = \lambda(q, a)$ and $a \in L_G(A)$. If x is generated by a rule $(q, A, q') \to (q, \varphi, q')$, then by Claim 2.2 there is $w \in L_G(\varphi)$ with $\delta(q, w) = q'$ and $\lambda(q, w) = x$. It follows that $w \in L_G(A)$ by the rule $A \to \varphi$, which exists by the construction of (4a).

Conversely, assume that $x = \lambda(q, w)$ with $w \in L_G(A)$ and $\delta(q, w) = q'$. If $w = a \in \Sigma$, then there is a rule $A \to a \in P$, and accordingly x is in the language $L_{G'}\big((q, A, q')\big)$ by the rule $(q, A, \delta(q, a)) \to \lambda(q, a)$. If $|w| > 1$, then it is generated by a long rule $A \to \varphi$. Now $x \in (q, \varphi, q')$ by Claim 2.2 and thus $x \in L_{G'}\big((q, A, q')\big)$ by the rule $(q, A, q') \to (q, \varphi, q')$. This ends the proof of the induction step and of the entire Claim 2.

Using this statement, it is easy to show that the constructed grammar generates the image of $L(G)$ under M. Consider that S' has the rules $S' \to (q_0, S, q_f)$ for all $q_f \in F$. So it follows from Claim 2 that $x \in L(G')$ if and only if $x = \lambda(q_0, w)$ for some $w \in L_G(S)$ with $\delta(q_0, w) = q_f \in F$, which means that $x = M(w)$ for some $w \in L(G)$. This shows that $L(G') = M\big(L(G)\big)$, which proves Theorem 1 for the case of Boolean grammars.

To see that the same construction works for conjunctive grammars, note that no new negations except $\neg\varepsilon$ are added. If G is conjunctive, then G' can be made conjunctive as well by replacing each conjunct $\neg\varepsilon$ with a reference to a nonterminal generating Σ^+. So Theorem 1 has been proved for conjunctive grammars as well.

To complete the proof of Theorem 1, it remains to consider unambiguous conjunctive and Boolean grammars. It is sufficient to establish the following statement:

Claim 3. *If* G *is unambiguous, then* G' *is unambiguous.*

Proof. Let us first prove the uniqueness of the factorizations. The only conjuncts in the rules of P' with multiple nonterminals are those of the form $(q, B, q'')(q'', C, q')$. So let $x = yz = \tilde{y}\tilde{z}$ be two factorizations of x with $y, \tilde{y} \in L_{G'}\big((q, B, q'')\big)$ and $z, \tilde{z} \in L_{G'}\big((q'', C, q')\big)$. Then, by Claim 2 (applied four times), $y = \lambda(q, u)$, $\tilde{y} = \lambda(q, \tilde{u})$, $z = \lambda(q'', v)$ and $\tilde{z} = \lambda(q'', \tilde{v})$ for some $u, \tilde{u} \in L_G(B)$ and $v, \tilde{v} \in L_G(C)$ with $\delta(q, u) = \delta(q, \tilde{u}) = q''$ and $\delta(q'', v) = \delta(q'', \tilde{v}) = q'$. Combining the images in each pair, one obtains $\lambda(q, uv) = yz$ and $\lambda(q, \tilde{u}\tilde{v}) = \tilde{y}\tilde{z}$, that is, $\lambda(q, uv) = \lambda(q, \tilde{u}\tilde{v}) = x$. Since M is injective, this implies $uv = \tilde{u}\tilde{v}$.

Consider that $u, \tilde{u} \in L_G(B)$ and $v, \tilde{v} \in L_G(C)$. Since the original grammar contains a conjunct BC, the factorization of any word into $L_G(B)L_G(C)$ must be unique, so $u = \tilde{u}$ and $v = \tilde{v}$. This implies $y = \tilde{y}$ and $z = \tilde{z}$, that is, the factorization of x is unique.

The second condition to be demonstrated is that different rules for any nonterminal generate disjoint languages. Consider any nonterminal in N'_1.

If the empty word is in $L_{G'}\big((q, A, q')\big)$, then it can only be generated by a rule $(q, A, q') \to \varepsilon$, since other rules explicitly deny it by a conjunct $\neg\varepsilon$.

In the case $x \neq \varepsilon$ and $x \in L_{G'}\big((q, A, q')\big)$, there is a word $w \in L_G(A)$ with $\delta(q, w) = q'$ and $\lambda(q, w) = x$. There is only one such w, since M is injective. There are two cases: the case of x generated by a rule $(q, A, \delta(q, a)) \to x$ for some $a \in \Sigma$, and the case of x generated by $(q, A, q') \to (q, \varphi, q')$. In the former case, $w = a$, and there is a unique such rule. In the latter case, w is generated by a rule $A \to \varphi$. Let x be generated by two different rules $(q, A, q') \to (q, \varphi, q')$ and $(q, A, q') \to (q, \psi, q')$. Then w is generated by two different rules $A \to \varphi$ and $A \to \psi$ by Claim 2.2, contradicting the unambiguity of G.

Next, consider the rules for $x \in L_{G'}\big((q, B, C, q')\big)$. If there are two rules for it, say $(q, B, C, q') \to (q, B, q'')(q'', C, q')$ and $(q, B, C, q') \to (q, B, q''')(q''', C, q')$, then there are two corresponding factorizations $x = yz = \tilde{y}\tilde{z}$. Then, by Claim 2 four times, there are such words $u, \tilde{u} \in L_G(B)$ and $v, \tilde{v} \in L_G(C)$, that $\delta(q, u) = q''$ and $\delta(q, \tilde{u}) = q'''$, and that $\lambda(q, u) = y$, $\lambda(q'', v) = z$, $\lambda(q, \tilde{u}) = \tilde{y}$ and $\lambda(q''', \tilde{v}) = \tilde{z}$. Now $\lambda(q, uv) = \lambda(q, \tilde{u}\tilde{v})$ and it follows from the injectivity of M that $uv = \tilde{u}\tilde{v}$. Since G in unambiguous, $u = \tilde{u}$ and $v = \tilde{v}$, so that $q'' = q'''$ and the rules are the same. □

5. Inverse Weak Coding

The main result of this paper is the closure of these four language families under inverse gsm mappings. A special case of this result is established in this section, and later it will be used to prove the general statement.

Let Σ be an alphabet and $h : \Sigma^* \to \Gamma^*$ *a weak coding*, that is, $|h(a)| \leqslant 1$ for all $a \in \Sigma$. Denote $\Sigma_0 = \{a \in \Sigma \mid h(a) = \varepsilon\}$ and $\Sigma_1 = \{a \in \Sigma \mid h(a) \in \Gamma\}$.

Theorem 2. *For every weak coding $h : \Sigma^* \to \Gamma^*$ and for every Boolean (conjunctive, unambiguous Boolean, unambiguous conjunctive) grammar G over Γ there exists a Boolean (conjunctive, unambiguous Boolean, unambiguous conjunctive) grammar over Σ generating the language $h^{-1}(L(G))$.*

Let $G = (\Gamma, N, P, S)$ be a Boolean grammar in binary normal form; for technical reasons, assume that $S \to \varepsilon \notin P$. The goal is to construct a grammar $G' = (\Sigma, N', P', S')$ for the language $h^{-1}(L(G))$. Let $N' = N \cup \{S', T\}$ be the set of nonterminals. Then P' contains the following rules:

$$S' \to TST \tag{5a}$$

$$T \to a_0 T \mid \varepsilon \quad (\text{for all } a_0 \in \Sigma_0) \tag{5b}$$

$$A \to B_1 T C_1 \& \ldots \& B_m T C_m \& \neg D_1 T E_1 \& \ldots \& \neg D_n T E_n \& \neg\varepsilon \quad (\text{for all } A \to B_1 C_1 \& \ldots \& B_m C_m \& \neg D_1 E_1 \& \ldots \& \neg D_n E_n \& \neg\varepsilon \in P) \tag{5c}$$

$$A \to a \quad (\text{for all } a \in \Sigma_1 \text{ and } A \to h(a) \in P) \tag{5d}$$

If ε should be in $L(G)$, then a rule $S' \to T$ can be added to G'; this case is not considered in the below proof.

It is easy to see that the constructed grammar is well-defined:

Claim 4. *The system (2) corresponding to G' has a strongly unique solution.*

Proof. The equation for T has a unique solution $T = \Sigma_0^*$ modulo every language. Now, for the entire system, it has to be proved that for every finite subword-closed language M the solution modulo M is unique. This is proved by induction on $|M|$.

Basis: The unique solution modulo $\{\varepsilon\}$ has $L_A = \varnothing$ for all $A \in N$ and $L_T = \{\varepsilon\}$, and accordingly $L_{S'} = \varnothing$.

Induction step: Let $M = M_0 \cup \{w\}$, with $w \notin M_0$ and with all subwords of w in M_0. By the induction hypothesis, the solution modulo M_0 is unique.

Let $(L_{S'}, L_S, \ldots, L_A, \ldots)$ and $(L'_{S'}, L'_S, \ldots, L'_A, \ldots)$ be any two different solutions modulo M. If $L_{S'}$ and $L'_{S'}$ are different, then so are L_S and L'_S, so it is sufficient to consider the case of the solutions being different on a variable $A \in N$. Assume, without loss of generality, that $w \in L_A$ and $w \notin L'_A$. If w is in L_A by a rule (5d), then it is in L'_A by the same rule. Let w be in L_A by a rule (5c), while the same rule (5c) does not produce w in L'_A. Then there is a conjunct $B_i T C_i$ with $w \in L_{B_i} L_T L_{C_i}$ and $w \notin L'_{B_i} L'_T L'_{C_i}$ (or, symmetrically, a conjunct $D_j T E_j$ with $w \notin L_{D_j} L_T L_{E_j}$ and $w \in L'_{D_j} L'_T L'_{E_j}$). Accordingly, w can be factorized as uxv, with $u \in L_{B_i}$, $x \in L_T$ and $v \in L_{C_i}$. Since $\varepsilon \notin L_B, L_C$, u and v are shorter than w and hence are in M_0. Then, since the solution modulo M_0 is unique, $u \in L'_{B_i}$ and $v \in L'_{C_i}$, and therefore $w \in L'_{B_i} L'_T L'_{C_i}$, which contradicts the assumption. □

Next, note that each nonterminal $A \in N$ in the grammar G' may generate only nonempty words that begin and end with symbols from Σ_1.

Claim 5. *For every $A \in N$, $L_{G'}(A) \subseteq \Sigma_1 \cup \Sigma_1(\Sigma_0 \cup \Sigma_1)^* \Sigma_1$.*

Proof. Suppose the contrary, that there is a word $w \in L_{G'}(A)$ that has the first or the last symbol from Σ_0 (the case of $w = \varepsilon$ is clearly impossible by (5)).

Let w be the shortest such word. It cannot be generated by a rule (5d), because that would imply $w \in \Sigma_1$. So w must be generated by a rule (5c), and by the first

conjunct of this rule, $w = uxv$ with $u \in L_{G'}(B)$, $t \in \Sigma_0^*$ and $v \in L_{G'}(C)$. Since $u, v \neq \varepsilon$, both u and v are shorter than w. Then, if the first symbol of w is from Σ_0, then u is a shorter word with the first symbol from Σ_0, and if w ends with a symbol from Σ_0, then so does v, which is shorter. In both cases this contradicts the choice of w. □

The correctness of the construction is established by the following correspondence of nonterminals in G and G'.

Claim 6. *Let $w \in \Sigma_1 \cup \Sigma_1(\Sigma_0 \cup \Sigma_1)^*\Sigma_1$ and $A \in N$. Then $w \in L_{G'}(A)$ if and only if $h(w) \in L_G(A)$.*

Proof. The proof is done by induction on the length of w.

Basis: $w = a \in \Sigma_1$. Then $a \in L_{G'}(A)$ if and only if there is a rule $A \to a$ in P', which, by (5d), exists if and only if $h(a) \in L_G(A)$.

Induction step: Let $w \in \Sigma_1(\Sigma_0 \cup \Sigma_1)^*\Sigma_1$ and let us first prove that under the induction hypothesis the following statement holds:

Claim 6.1. *$w \in L_{G'}(BTC)$ if and only if $h(w) \in L_G(BC)$.*

Proof. If $w \in L_{G'}(BTC)$, there is a factorization $w = uxv$, where $u \in L_{G'}(B)$, $x \in L_{G'}(T)$ and $v \in L_{G'}(C)$. Then $u, v \neq \varepsilon$, and hence $|u|, |v| < |w|$. By the induction hypothesis for u and v, $h(u) \in L_G(B)$ and $h(v) \in L_G(C)$. Also $h(x) = \varepsilon$ by (5b), so $h(uxv) = h(u)h(x)h(v) = h(u)h(v) \in L_G(BC)$.

Conversely, if $h(w) \in L_G(BC)$, there is a factorization $w = u'v'$, such that $h(u') \in L_G(B)$ and $h(v') \in L_G(C)$. This implies $u', v' \neq \varepsilon$ and thus $|u'|, |v'| < |w|$. Let $x \in \Sigma_0^*$ be the longest suffix of u' comprised of symbols from Σ_0, that is, $u' = ux$ with $u \in \Sigma_1 \cup \Sigma_1(\Sigma_0 \cup \Sigma_1)^*\Sigma_1$. Then $h(u) = h(u')$ and, by the induction hypothesis, $u \in L_{G'}(B)$. Similarly, let $y \in \Sigma_0^*$ be the longest prefix of v' containing only symbols from Σ_0: then $v' = yv$ with $v \in \Sigma_1 \cup \Sigma_1(\Sigma_0 \cup \Sigma_1)^*\Sigma_1$, and the induction hypothesis gives $v \in L_{G'}(C)$. Combining these facts, $w = uxyv \in L_{G'}(BTC)$, which completes the proof of Claim 6.1. □

Next, a similar statement for the right hand sides of the long rules will be established:

Claim 6.2. *$w \in L_{G'}(B_1TC_1\& \ldots \&B_mTC_m\&\neg D_1TE_1\& \ldots \&\neg D_nTE_n\&\neg\varepsilon)$ if and only if $h(w) \in L_G(B_1C_1\& \ldots \&B_mC_m\&\neg D_1E_1\& \ldots \&\neg D_nE_n\&\neg\varepsilon)$.*

Proof. Suppose $w \in L_{G'}(B_1TC_1\& \ldots \&B_mTC_m\&\neg D_1TE_1\& \ldots \&\neg D_nTE_n\&\neg\varepsilon)$. This is the case if and only if $w \in L_{G'}(B_iTC_i)$ for all applicable i and $w \notin L_{G'}(D_jTE_j)$ for all applicable j. By Claim 6.1 this is equivalent to $h(w) \in L_G(B_iC_i)$ for all i and $h(w) \notin L_G(D_jE_j)$ for all j, which is equivalent to $h(w) \in L_G(B_1C_1\& \ldots \&B_mC_m\&\neg D_1E_1\& \ldots \&\neg D_nE_n\&\neg\varepsilon)$. Thus Claim 6.2 has been proved. □

Getting back to the induction step in the proof of Claim 6, consider that $w \in L_{G'}(A)$ is equivalent to the existence of a rule

$$A \to B_1TC_1\& \ldots \& B_mTC_m\& \neg D_1TE_1\& \ldots \& \neg D_nTE_n\& \neg\varepsilon$$

in P'. Such a rule exists if and only if P contains a rule

$$A \to B_1C_1\& \ldots \& B_mC_m\& \neg D_1E_1\& \ldots \& \neg D_nE_n\& \neg\varepsilon.$$

On the other hand, by Claim 6.2, this is equivalent to $h(w) \in L_G(A)$. This completes the proof of Claim 6.

Now Theorem 2 can be proved for Boolean grammars. It is claimed that the language $L_{G'}(TST)$ equals $h^{-1}(L_G(S))$. First consider the set of pre-images of ε, which is Σ_0^*: no word of this form is in $L_{G'}(TST)$, and neither is ε in $L_G(S)$.

It remains to show that a word with a nonempty image is in $L_{G'}(TST)$ if and only if it is in $h^{-1}(L_G(S))$. So let $w \in \Sigma^*$ be such that $h(w) \neq \varepsilon$. Suppose $h(w) \in L_G(S)$. Then w can be factorized as $w = xw'y$, where $x, y \in \Sigma_0^*$ and $w' \in \Sigma_1 \cup \Sigma_1(\Sigma_0 \cup \Sigma_1)^*\Sigma_1$ with $h(w') = h(w) \in L_G(S)$. By Claim 6 and the definition of T, this is equivalent to $w' \in L_{G'}(S)$ and $x, y \in L_{G'}(T)$. By the rule (5a), this holds if and only if $w \in L_{G'}(S')$, and thus indeed $L(G') = h^{-1}\big(L(G)\big)$.

If G is conjunctive, then no new negations are added and G' is conjuctive, so Theorem 2 holds for conjunctive grammars as well. It remains to show that the construction preserves unambiguity, which will complete the proof of Theorem 2.

Claim 7. *If G is unambiguous, then G' is unambiguous.*

Proof. Consider factorizations of words in G' according to its conjuncts. For the rule (5a), if $w \in L_{G'}(TST)$, then $w = xw'y$, where $x, y \in \Sigma_0^*$ and $w' \in \Sigma_1 \cup \Sigma_1(\Sigma_0 \cup \Sigma_1)^*\Sigma_1$, is the unique factorization of w with respect to $L_{G'}(T)$ and $L_{G'}(S)$.

All the other conjuncts that have multiple nonterminals are of the form BTC. So, let $w \in L_{G'}(BTC)$. Suppose $w = u_1x_1v_1 = u_2x_2v_2$, with $u_1, u_2 \in L_{G'}(B)$, $x_1, x_2 \in L_{G'}(T)$ and $v_1, v_2 \in L_{G'}(C)$. Then by Claim 6, $h(w) = h(u_1v_1) = h(u_2v_2) \in L_G(BC)$. In addition, $u_1, u_2 \in \Sigma_1 \cup \Sigma_1(\Sigma_0 \cup \Sigma_1)^*\Sigma_1$ by Claim 5. It follows that if $|u_1| < |u_2|$, then also $|h(u_1)| < |h(u_2)|$. This means there would be two different factorizations of $h(w)$ with respect to $L_G(B)$ and $L_G(C)$, which contradicts the unambiguity of G. This proves that conjuncts of G' yield unique factorizations.

If a word w is generated by some rule (5c) of G', then, by Claim 6.2, there is a corresponding rule of G that generates $h(w)$. Because G is unambiguous, $h(w)$ is generated by a unique rule for A. From this it follows that the languages generated by distinct rules are disjoint, and furthermore that G' is unambiguous. □

6. Inverse gsm Mappings

The above constructions for injective gsm mappings and inverse weak codings of Boolean grammars can now be combined to establish their closure under inverse gsm mappings. This will be done using the following general result:

Lemma 2. *Let $\mathcal{L}$ be a family of languages closed under (1) partial injective gsm mappings and (2) inverse weak codings. Then $\mathcal{L}$ is closed under inverse partial gsm mappings.*

Provided that the closure under the above three operations is effective, the closure under inverse partial gsm mappings is effective as well.

The construction is actually from Ginsburg and Ullian [4], who used it in the special case of unambiguous context-free languages to establish their closure under inverse gsm mappings. The representation they obtained applies for every family of languages.

Proof. Let $L \subseteq \Gamma^*$ be any language and let $M = (\Sigma, \Gamma, Q, q_0, \delta, \lambda, F)$ be a gsm. We assume that Σ and Γ are disjoint.

The first step is to define a simulation of M, which maps words in Σ^* to words over $\Delta = \Sigma \cup \Gamma$. Define a gsm $M_1 = (\Sigma, \Delta, Q, q_0, \delta, \lambda_1, F)$, where

$$\lambda_1(q, a) = a\lambda(q, a).$$

Now the image $M_1(w)$ contains the computation history of M on w, including the symbols of w and the outputs. Then $M_1(\Sigma^*)$ becomes the language of valid computation histories of M; note that it is regular as an image of a gsm.

Next, define a weak coding $h : \Delta \to \Gamma$ that converts such a computation history into an output word of M. Let

$$h(c) = \begin{cases} \varepsilon, \text{ if } c \in \Sigma, \\ c, \text{ if } c \in \Gamma. \end{cases}$$

Now $h(M_1(w)) = M(w)$ for all $w \in \Sigma^*$.

It is also possible to reconstruct the input word of M and M_1 from a computation history of this form. Let $A = (\Delta, Q_2, q_0^2, \delta_2, F_2)$ be a DFA recognizing the language $M_1(\Sigma^*)$. This DFA is augmented to a gsm $M_2 = (\Delta, \Gamma, Q_2, q_0^2, \delta_2, \lambda_2, F_2)$, maintaining the same set of states and the same transition function and adding the output $\lambda_2(q, a) = a$ for all $a \in \Sigma$ and $\lambda_2(q, b) = \varepsilon$ for all $b \in \Gamma$ in every state $q \in Q_2$. Then $M_2(M_1(w)) = w$ for every $w \in \Sigma^*$ accepted by M_1. On the other hand, M_2 contains a recognizer for $M_1(\Sigma^*)$, so every word $x \notin M_1(\Sigma^*)$ is rejected. Therefore, every word w in the domain of M_1 has a unique pre-image $M_2^{-1}(w) = M_1(w)$, that is, M_2 is injective.

Now the pre-image of every language $L \subseteq \Gamma^*$ under M can be represented as follows:

$$M^{-1}(L) = M_2\big(h^{-1}(L)\big). \tag{6}$$

Here h^{-1} attempts to re-construct computation histories of M on words w with $M(w) \in L$, then recognition of $M_1(\Sigma^*)$ by M_2 filters out ill-formed computation histories, and finally M_2 extracts the actual words w from these computation histories.

If $L \in \mathcal{L}$, then $h^{-1}(L)$ is in $\mathcal{L}$ as a pre-image under weak coding and $M_2\big(h^{-1}(L)\big)$ belongs to $\mathcal{L}$ by the closure under injective gsm mappings. So $M^{-1}(L)$ is in $\mathcal{L}$, which establishes the closure of $\mathcal{L}$ under inverse gsm mappings. □

The families of languages generated by Boolean (conjunctive, unambiguous Boolean, unambiguous conjunctive) grammars meet the assumptions of Lemma 2: they are closed under injective partial gsm mappings by Theorem 1, under inverse weak codings by Theorem 2 and under intersection, since it is an explicit operation. It follows that each of the four families is closed under inverse partial gsm mappings:

Theorem 3. *For every generalized sequential machine* $M = (\Sigma, \Gamma, Q, q_0, \delta, \lambda, F)$ *implementing a partial function* $M : \Sigma^* \to \Gamma^*$ *and for every Boolean (conjunctive, unambiguous Boolean, unambiguous conjunctive) grammar* G *over* Γ *there exists a Boolean (conjunctive, unambiguous Boolean, unambiguous conjunctive) grammar over* Σ *generating the language* $M^{-1}(L(G))$.

The construction of a grammar for $M^{-1}(L(G))$ is a combination of the constructions in Theorem 1 and in Theorem 2 according to Lemma 2, where (6) gives an explicit formula for this combination.

7. Conclusion

It has been shown that Boolean grammars and some of their subfamilies are closed under injective gsm mappings and inverse gsm mappings. It remains unknown whether they are closed under non-erasing gsm mappings or under inverses of nondeterministic transducers. These problems are left for future study; in both cases non-closure appears quite likely.

The closure properties of Boolean grammars and their subfamilies are given in Table 1. The bottom right corner of the table has been established in this paper. The closure properties of the unambiguous families remain to be studied. In addition, it remains unknown whether conjunctive languages are closed under complementation.

Table 1. Closure properties of Boolean grammars, compared to other classes.

	$\cup$	$\cap$	$\sim$	$\cdot$	$*$	R	h	$h_{\varepsilon\text{-free}}$	h^{-1}	gsm^{-1}
Reg	+	+	+	+	+	+	+	+	+	+
UnambCF	− [4]	−	− [6]	− [4]	?	+	+ [4]	+ [4]	+ [4]	+ [4]
LinCF	+	−	−	−	−	+	+	+	+	+ [5]
CF	+	−	−	+	+	+	+	+	+	+ [3]
LinConj	+	+	+ [12]	−	− [12]	+	− [2]	−	+ [2]	+ [7]
UnambConj	?	+	?	?	?	+	−	?	+	+
UnambBool	+	+	+	?	?	+	−	?	+	+
Conj	+	+	?	+	+	+	−	?	+	+
Bool	+	+	+	+	+	+	−	?	+	+

References

[1] K. Culik II, J. Gruska, A. Salomaa, "Systolic trellis automata", I–II, *International Journal of Computer Mathematics*, 15 (1984), 195–212 and 16 (1984), 3–22.

[2] K. Culik II, J. Gruska, A. Salomaa, "Systolic trellis automata: stability, decidability and complexity", *Information and Control*, 71 (1986) 218–230.

[3] S. Ginsburg, G. Rose, "Operations which preserve definability in languages", *Journal of the ACM*, 10:2 (1963), 175–195.

[4] S. Ginsburg, J. Ullian, "Preservation of unambiguity and inherent ambiguity in context-free languages", *Journal of the ACM*, 13:3 (1966), 364–368.

[5] M. A. Harrison, *Introduction to Formal Language Theory*, Addison-Wesley, 1978.

[6] T. N. Hibbard, J. Ullian, "The independence of inherent ambiguity from complementedness among context-free languages", *Journal of the ACM*, 13:4 (1966), 588–593.

[7] O. H. Ibarra, S. M. Kim, "Characterizations and computational complexity of systolic trellis automata", *Theoretical Computer Science*, 29 (1984), 123–153.

[8] A. Jeż, "Conjunctive grammars can generate non-regular unary languages", *International Journal of Foundations of Computer Science*, 19:3 (2008), 597–615.

[9] A. Jeż, A. Okhotin, "Conjunctive grammars over a unary alphabet: undecidability and unbounded growth", *Theory of Computing Systems*, 46:1 (2010), 27–58.

[10] V. Kountouriotis, Ch. Nomikos, P. Rondogiannis, "Well-founded semantics for Boolean grammars", *Information and Computation*, 207:9 (2009), 945–967.

[11] A. Okhotin, "Conjunctive grammars", *Journal of Automata, Languages and Combinatorics*, 6:4 (2001), 519–535.

[12] A. Okhotin, "On the equivalence of linear conjunctive grammars to trellis automata", *RAIRO Informatique Théorique et Applications*, 38:1 (2004), 69–88.

[13] A. Okhotin, "Boolean grammars", *Information and Computation*, 194:1 (2004), 19–48.

[14] A. Okhotin, "Generalized LR parsing algorithm for Boolean grammars", *International Journal of Foundations of Computer Science*, 17:3 (2006), 629–664.

[15] A. Okhotin, "Nine open problems for conjunctive and Boolean grammars", *Bulletin of the EATCS*, 91 (2007), 96–119.

[16] A. Okhotin, "Recursive descent parsing for Boolean grammars", *Acta Informatica*, 44:3–4 (2007), 167–189.

[17] A. Okhotin, "Unambiguous Boolean grammars", *Information and Computation*, 206:9–10 (2008), 1234–1247.