Some Results Regarding Tree Homomorphic Feature Structure Grammar and the Empty String*

Tore Burheim
Telenor Research and Development
PO.Box 23, N-2007 Kjeller
Tore.Burheim@fou.telenor.no

Abstract

In this article we focus on some restrictions we may impose on Tree Homomorphic Feature Structure Grammar (THFSG) regarding the empty string. After defining the restrictions, we prove some closure properties of the two new classes of languages these restrictions give us. Furthermore, we establish that the membership problem is PSPACE-complete for THFSGs without empty productions in the phrase structure rules.

1 Introduction and some definitions

In many linguistic theories the empty string plays the prominent role of the empty category. This is also the case for early Lexical-Functional Grammar (LFG) [KB82]. However, later the use of the empty category in LFG has been questioned. Kaplan and Zaenen [KZ89] argue that by the introduction of functional uncertainty (implemented as regular expressions in equation schemata) there is no place for the empty category in LFG. Grammaticality is enforced through the completeness, coherence and consistence constraints on the functional structure. Bresnan [Bre95] on the other side, argues against this view. With no intention of going into this discussion, we may at least say that, from a linguistic point of view, it remains a matter of dispute whether the empty category is needed in LFG-like grammars which allow regular expressions in the equation schemata.

From a more technical perspective, the empty string requires special care with respect to parsing: The question is, when do we introduce an empty category/string? Even if it sounds a bit overwhelming, we may introduce an unlimited number of empty categories all over the string. This is a problem e.g. for bottom-up parsing. Both the linguistic and the technical perspectives make it interesting to study grammar formalisms with respect to the empty string.

Tree Homomorphic Feature Structure Grammar (THFSG) [Bur97b] is an LFG-like feature structure grammar formalism which may allow regular expressions in the equation schemata.[1]

*This article is a short version of [Bur98].

[1]In [Bur97a] it was proved that we may introduce regular expressions in the equation schemata for THFSG without extending the class of languages described.

As LFG, it has a context-free phrase structure backbone and it allows the empty string on the right hand side in the phrase structure rules. In the work presented here, we study the classes of languages we get by making two different restrictions regarding the empty string. First we define the class of languages we get by just removing the empty string from the languages defined by THFSG. Then we define a special version of THFSG in which we do not allow the empty string in the phrase structure rules, the so-called ε-free THFSG. But first, some words about THFSG itself.

1.1 Tree Homomorphic Feature Structure Grammar

THFSG was introduced by Burheim in [Bur97b]. It is based on Lexical Functional Grammar [KB82, DKMZ95] and work by Colban [Col91]. The formalism has a context-free phrase structure backbone and add equations to the nodes in the phrase-structure tree as is done in LFG. These equations describe feature structures. In the formal framework there are two main differences from LFG: First, due to a restriction that is imposed on the equations in the grammar, the referred part of the feature structure is a tree which includes a homomorphic image of the phrase structure tree. On the other side, with this restriction we do not need the off-line parsability constraint to make the grammar decidable.

We give a brief introduction to THFSG. Since it is based on feature structures we will start with an informal definition of feature structures.

A *feature structure* over a set of attribute symbols $\mathcal{A}$ and value symbols $\mathcal{V}$ is a four-tuple $\langle Q, f_D, \delta, \theta \rangle$ where Q is a finite set of nodes, $f_D : D \to Q$ is a function, called the name mapping, $\delta : Q \times \mathcal{A} \to Q$ is a partial function, called the transition function, and $\theta : Q \to \mathcal{V}$ is a partial function called the atomic value function. We extend the transition function by its transitive and reflexive closure to be a function from pairs of nodes and *strings* of attribute symbols, and we assume that D is implicit defined by f. A feature structure is *well defined* if it is describable (from named nodes), acyclic and atomic.

THFSG uses equations to talk about feature structures, such that a feature structure may or may not satisfy each equation. A feature structure *satisfies* the equation $x_1 u_1 \doteq x_2$ if and only if $\delta(f(x_1), u_1) = f(x_2)$, and the equation $x_3 u_3 \doteq v$ if and only if $\alpha(\delta(f(x_3), u_3)) = v$, where $x_1, x_2, x_3 \in D$, $u_1, u_3 \in \mathcal{A}^*$ and $v \in \mathcal{V}$. These path and value equations are the only kind of equations we use in THFSG. The well defined feature structure M satisfies the set of equations E, if and only if M satisfies every equation in E.

A *Tree Homomorphic Feature Structure Grammar*, THFSG, over the set of attribute symbols $\mathcal{A}$ and value symbols $\mathcal{V}$, is a 5-tuple $\langle \mathcal{K}, \mathcal{S}, \Sigma, \mathcal{P}, \mathcal{L} \rangle$ where $\mathcal{K}$ and Σ are two finite and disjoint sets of symbols, called categories and terminals, and $S \in \mathcal{K}$ is the start symbol. Moreover $\mathcal{P}$ is a finite set of production rules

$$\begin{array}{cccccc} A_0 & \to & A_1 & \ldots & A_m & \\ & & E_1 & & E_m & \end{array} \qquad (1)$$

where $m \geq 1$, $A_0, \ldots, A_m \in \mathcal{K}$, and for every i, $1 \leq i \leq m$, is E_i a finite set with *one and only one* equation schema on the form $\uparrow u_1 = \downarrow$ where $u_1 \in \mathcal{A}^*$, and a finite number of equation schemata on the form $\uparrow u_2 = v$ where $u_2 \in \mathcal{A}^+$ and $v \in \mathcal{V}$. At last $\mathcal{L}$ is a finite set of lexicon rules

$$A \rightarrow \begin{array}{c} t \\ E \end{array} \tag{2}$$

where $A \in \mathcal{K}$, $t \in (\Sigma \cup \{\varepsilon\})$, and E is a finite set of equation schemata on the form $\uparrow u_3 = v$ where $u_3 \in \mathcal{A}^+$ and $v \in \mathcal{V}$.

The constituent structure (c-structure) is a phrase structure tree decorated with symbols and equation schemata sets according to the production and lexicon rules the usual way. As in LFG, the up and down arrows in the equation schemata are metavariables. To instantiate the arrows we substitute them with nodes in the c-structure, which then become the name domain in the name mapping function. This function is then a mapping from the nodes in the c-structure to the nodes in the feature structure. A c-structure is *feature consistent* if and only if the union of all the equations in the c-structure is satisfied by a well defined feature structure.

2 THFSG and ε

In this section we study some relations between different restrictions we may impose on THFSG regarding the empty string ε and some closure properties on the classes of languages we get when imposing these restrictions. Two restrictions may be imposed: the absence of the empty string in the grammar and the absence of the empty string in the language generated. Let us start with the definition:

Definition 1

- ***ε-free form:*** *A* THFSG *is in ε-free form if and only if the empty string ε does not occur on the right hand side in any lexicon rule in the grammar.*

- ***ε-free*** **THFSG*-languages:*** *The class of ε-free* THFSG*-languages, $\mathcal{C}(\text{THFSG})_{\not\varepsilon}$, is defined as*

$$\mathcal{C}(\text{THFSG})_{\not\varepsilon} = \{L \in \mathcal{C}(\text{THFSG}) \mid \varepsilon \notin L\} \tag{3}$$

THFSGs in ε-free form only have lexicon rules as

$$A \rightarrow \begin{array}{c} t \\ E \end{array} \tag{4}$$

where t is a symbol in the alphabet Σ, and *not* the empty string. THFSGs in ε-free form are almost identical to the grammar formalism GF1 defined by Colban [Col91]. The only difference is that GF1 only allows path equation schemata on the forms $\uparrow \doteq \downarrow$ and $\uparrow a \doteq \downarrow$ where a is a single attribute symbol. This restriction on attribute strings in GF1 is used in a normal form[2] for THFSG [Bur97b]. Following the lines in the proof of normal form, it is easy to see that we may rewrite any grammar with longer attribute strings into an equivalent grammar with at most one single attribute symbol in each path equation schema, without violating the ε-free restriction. Hence the ε-free THFSG describe the same class of languages as Colban's GF1.

[2]This normal form also requires exactly two elements on the right hand side in the production rules

When studying classes of languages, we often want to study what happens when we impose different algebraic operations on a class, i.e. we want to study its closure properties. In this context, trios and Abstract Families of Languages (AFL) are of particular interest. But before we go into details on trios and AFL's, let us clarify what we mean by a *class* of languages. A class of languages is a set of languages $\mathcal{C}_\Gamma$ over a countable set of symbols Γ, such that for each language $L \in \mathcal{C}_\Gamma$ there exists a finite alphabet $\Sigma \subseteq \Gamma$ such that $L \subseteq \Sigma^*$. A class of languages given by a grammar formalism $\mathcal{C}_\Gamma(\mathcal{GF})$ is a class of languages such that for each language L' in $\mathcal{C}_\Gamma(\mathcal{GF})$ there exists a grammar G in $\mathcal{GF}$ such that $L(G) = L'$, and for each grammar G in $\mathcal{GF}$ $L(G)$ is in $\mathcal{C}_\Gamma(\mathcal{GF})$. In the rest of this paper we assume that Γ is given and omit Γ as subscript.

A *trio* is a class of languages closed under ε-free homomorphism, inverse homomorphism, and intersection with regular languages. A *full trio* is a class of languages closed under arbitrary homomorphism, inverse homomorphism, and intersection with regular languages. By intersection with regular languages, we mean the traditional binary set theoretic operation. A *string homomorphism* is a function $h : \Delta^* \to \Sigma^*$ such that for every $w \in \Delta^*$ and $a \in \Delta$ we have

$$h(\varepsilon) = \varepsilon \tag{5}$$
$$h(aw) = h(a)h(w) \tag{6}$$

A homomorphism is ε-free if $h(a) \neq \varepsilon$ for all $a \neq \varepsilon$. The string homomorphic image of a language $L \subseteq \Delta^*$ under a string homomorphism $h : \Delta^* \to \Sigma^*$ is the language $\{h(w) \mid w \in L\}$. The inverse string homomorphic image of a language $L' \subseteq \Sigma^*$ is the language $\{w \mid h(w) \in L'\}$.

An *Abstract Family of Languages* is a trio which is also closed under concatenation, union, and positive closure. A *full* abstract family of languages is a full trio closed under concatenation, union, and Kleene closure. By union we mean the traditional binary set-theoretic operation. The concatenation of two languages L_1 and L_1, is the language $\{w_1w_2 \mid w_1 \in L_1 \textit{ and } w_2 \in L_2\}$. By positive closure of a language L we mean the language $\{w_1 \ldots w_n \mid n \geq 1 \textit{ and } w_1, \ldots, w_n \in L\}$, and by Kleene closure of a language L we mean the language $\{w_1 \ldots w_n \mid n \geq 0 \textit{ and } w_1, \ldots, w_n \in L\}$. The only difference between Kleene closure and the positive closure is the empty string in Kleene closure.

To show that a class of languages is a full trio, it is sufficient to show closure under NFT-mapping. This due to the fact that the NFT-image of a class of languages gives us the least full trio containing the class [Gin75]. If we restrict the NFT-mapping to ε-free mappings, we get the least trio. A *Nondeterministic Finite Transducer* (NFT) is a 6-tuple $M = \langle Q, \Delta, \Sigma, \delta, q_0, F\rangle$ where Q is a finite set of states, Δ is an input-alphabet, Σ is an output-alphabet, δ is a function from $Q \times (\Delta \cup \{\varepsilon\})$ to finite subsets of $Q \times \Sigma^*$, $q_0 \in Q$ is the initial state, and $F \subseteq Q$ is a set of final states. An NFT is ε-free if δ is a function from $Q \times (\Delta \cup \{\varepsilon\})$ to finite subsets of $Q \times \Sigma^+$. We extend the transformation function δ as follows: (1) For every $q \in Q$, $\langle q, \varepsilon\rangle \in \delta(q, \varepsilon)$. (2) If $\langle q_2, x\rangle \in \delta(q_1, w)$ and $\langle q_3, y\rangle \in \delta(q_2, a)$, then $\langle q_3, xy\rangle \in \delta(q_1, wa)$ for every $q_1, q_2, q_3 \in Q$, $a \in (\Delta \cup \{\varepsilon\})$, and $w \in \Delta^*$. Let $\widetilde{\mathcal{NFT}}$ be the set of NFTs, and $\mathcal{NFT}$ be the set of ε-free NFTs.

For any NFT $M = \langle Q, \Delta, \Sigma, \delta, q_0, F\rangle$, the image under M of a string $w \in \Delta^*$ and a language $L \subseteq \Delta^*$ are

$$M(w) = \{x \mid \exists q \in F : (q, x) \in \delta(q_0, w)\} \tag{7}$$
$$M(L) = \bigcup_{w \in L} M(w) \tag{8}$$

The inverse images under M of a string $x \in \Sigma^*$ and a language $L' \subseteq \Sigma^*$ are

$$\begin{aligned} M^{-1}(x) &= \{w \mid x \in M(w)\} & (9)\\ M^{-1}(L') &= \bigcup_{w \in L'} M^{-1}(w) & (10) \end{aligned}$$

For any class $\mathcal{C}$ of languages, the NFT-image and ε-free NFT-image of $\mathcal{C}$, $\widetilde{\mathcal{M}}(\mathcal{C})$ and $\mathcal{M}(\mathcal{C})$, are defined as

$$\begin{aligned} \widetilde{\mathcal{M}}(\mathcal{C}) &= \{M(L) \mid L \in \mathcal{C} \textit{ and } M \in \widetilde{\mathcal{NFT}}\} & (11)\\ \mathcal{M}(\mathcal{C}) &= \{M(L) \mid L \in \mathcal{C} \textit{ and } M \in \mathcal{NFT}\} & (12) \end{aligned}$$

An important result from [Gin75] is that for any class $\mathcal{C}$ of languages:

$$\begin{aligned} \widetilde{\mathcal{M}}(\mathcal{C}) &= \{h_2(h_1^{-1}(L) \cap R) \mid L \in \mathcal{C},\ R \in \mathcal{R}, \\ &\qquad h_1, h_2 \textit{ are homomorphisms}\} & (13)\\ \mathcal{M}(\mathcal{C}) &= \{h_2(h_1^{-1}(L) \cap R) \mid L \in \mathcal{C},\ R \in \mathcal{R},\ h_1 \textit{ is a homomorphism} \\ &\qquad \textit{and } h_2 \textit{ is an } \varepsilon\textit{-free homomorphism}\} & (14) \end{aligned}$$

where $\mathcal{R}$ is the class of regular languages. From this result, Ginsburg [Gin75] shows that $\widetilde{\mathcal{M}}(\mathcal{C})$ is the least full trio containing $\mathcal{C}$ and $\mathcal{M}(\mathcal{C})$ is the least trio containing $\mathcal{C}$. An interesting case arises when $\mathcal{C}$ only consists of a single language.

Now let us state some closure properties for ε-free THFSG-languages. In [Bur97b], it was proved that the class of THFSG-languages, $\mathcal{C}(\mathsf{THFSG})$, is a full Abstract Family of Languages. From this fact, we get the following result directly.

Lemma 1 *The class of ε-free THFSG-languages, $\mathcal{C}(\mathsf{THFSG})_{\not\varepsilon}$, is an abstract family of languages and*

$$\mathcal{C}(\mathsf{THFSG})_{\not\varepsilon} = \{L - \{\varepsilon\} \mid L \in \mathcal{C}(\mathsf{THFSG})\} \qquad (15)$$

Proof: From [Gin75, p21], we know that if $\mathcal{C}$ is an abstract family of languages, then $\{L \in \mathcal{C} \mid \varepsilon \notin L\} = \{L - \{\varepsilon\} \mid L \in \mathcal{C}\}$ is an abstract family of languages. Since $\mathcal{C}(\mathsf{THFSG})$ is a (full) abstract family of languages [Bur97b], we know that $\mathcal{C}(\mathsf{THFSG})_{\not\varepsilon}$ is an abstract family of languages and we know that

$$\mathcal{C}(\mathsf{THFSG})_{\not\varepsilon} = \{L - \{\varepsilon\} \mid L \in \mathcal{C}(\mathsf{THFSG})\} \qquad (16)$$

■

From the definition of ε-free THFSG-languages and Lemma 1, we have that

$$\{L \in \mathcal{C}(\mathsf{THFSG}) \mid \varepsilon \notin L\} = \{L - \{\varepsilon\} \mid L \in \mathcal{C}(\mathsf{THFSG})\} \qquad (17)$$

Another way to view this is that

$$\mathcal{C}(\mathsf{THFSG}) = \{L, L \cup \{\varepsilon\} \mid L \in \mathcal{C}(\mathsf{THFSG})_{\not\varepsilon}\} \qquad (18)$$

Since $\mathcal{C}(\mathsf{THFSG})_{\not\varepsilon}$ is an abstract family of languages, and hence a trio we have that it is closed under ε-free NFT-mapping

$$\mathcal{M}(\mathcal{C}(\mathsf{THFSG})_{\not\varepsilon}) = \mathcal{C}(\mathsf{THFSG})_{\not\varepsilon} \qquad (19)$$

It is straightforward to establish the closure under arbitrary NFT-mapping:

Lemma 2 *The NFT-image of* $\mathcal{C}(\mathsf{THFSG})_{\not\varepsilon}$ *is* $\mathcal{C}(\mathsf{THFSG})$, *that is*

$$\widetilde{\mathcal{M}}(\mathcal{C}(\mathsf{THFSG})_{\not\varepsilon}) = \mathcal{C}(\mathsf{THFSG}) \tag{20}$$

Proof: Since $\mathcal{C}(\mathsf{THFSG})$ is closed under NFT-mapping and $\mathcal{C}(\mathsf{THFSG})_{\not\varepsilon}$ is a subset of $\mathcal{C}(\mathsf{THFSG})$, we have that $\widetilde{\mathcal{M}}(\mathcal{C}(\mathsf{THFSG})_{\not\varepsilon}) \subseteq \mathcal{C}(\mathsf{THFSG})$. Since $\mathcal{C}(\mathsf{THFSG}) = \{L, L \cup \{\varepsilon\} \mid L \in \mathcal{C}(\mathsf{THFSG})_{\not\varepsilon}\}$ and it is straightforward to define an NFT M such that $M(w) = \{w, \varepsilon\}$ for each string w, we have that $\mathcal{C}(\mathsf{THFSG}) \subseteq \widetilde{\mathcal{M}}(\mathcal{C}(\mathsf{THFSG})_{\not\varepsilon})$. ∎

Now we turn our attention to ε-free THFSGs. Recall that these are THFSGs without the empty string on the right hand side in lexicon rules. Immediately, we see that ε-free THFSGs define only ε-free THFSG-languages, hence

$$\mathcal{C}(\varepsilon\text{-}\mathit{free}\mathsf{THFSG}) \subseteq \mathcal{C}(\mathsf{THFSG})_{\not\varepsilon} \tag{21}$$

However, ε-free languages may be generated by THFSGs with ε in the lexicon rules.

When we now turn our attention to closure properties for ε-free THFSGs, let us first establish that the class of languages they define is an Abstract Family of Languages.

Lemma 3 *The class of languages described by ε-free* THFSG*s,*
$\mathcal{C}(\varepsilon\text{-}\mathit{free}\mathsf{THFSG})$, *is an abstract family of languages.*

Proof: The proof that $\mathcal{C}(\varepsilon\text{-}\mathit{free}\mathsf{THFSG})$ is closed under union, concatenation, and positive closure, is almost identical to the proof of closure under union, concatenation, and Kleene star for $\mathcal{C}(\mathsf{THFSG})$ [Bur97b]. The only difference is that we do not need, and are not allowed to introduce, the lexicon rule which generates the empty string for Kleene closure.

The proof that $\mathcal{C}(\varepsilon\text{-}\mathit{free}\mathsf{THFSG})$ is closed under ε-free NFT-mapping is identical to the proof that $\mathcal{C}(\mathsf{THFSG})$ is closed under NFT-mapping in [Bur97b], except that we must restrict the NFT to be ε-free. Hence the class $\mathcal{C}(\varepsilon\text{-}\mathit{free}\mathsf{THFSG})$ is an Abstract Family of Languages. ∎

Since an Abstract Family of Languages is also a trio, and the ε-free NFT-image of a class of languages is the least trio containing the class, we have that $\mathcal{C}(\varepsilon\text{-}\mathit{free}\mathsf{THFSG})$ is closed under ε-free NFT-mapping, that is

$$\mathcal{M}(\mathcal{C}(\varepsilon\text{-}\mathit{free}\mathsf{THFSG})) = \mathcal{C}(\varepsilon\text{-}\mathit{free}\mathsf{THFSG}) \tag{22}$$

Now, if we look for the least full abstract family of languages containing $\mathcal{C}(\varepsilon\text{-}\mathit{free}\mathsf{THFSG})$ we get the following result.

Lemma 4 *The NFT-image of* $\mathcal{C}(\varepsilon\text{-}\mathit{free}\mathsf{THFSG})$ *is* $\mathcal{C}(\mathsf{THFSG})$, *that is*

$$\widetilde{\mathcal{M}}(\mathcal{C}(\varepsilon\text{-}\mathit{free}\mathsf{THFSG})) = \mathcal{C}(\mathsf{THFSG}) \tag{23}$$

Proof: Since $\mathcal{C}(\varepsilon\text{-}\mathit{free}\mathsf{THFSG}) \subseteq \mathcal{C}(\mathsf{THFSG})$, and $\mathcal{C}(\mathsf{THFSG})$ is closed under NFT-mapping, we know that $\widetilde{\mathcal{M}}(\mathcal{C}(\varepsilon\text{-}\mathit{free}\mathsf{THFSG})) \subseteq \mathcal{C}(\mathsf{THFSG})$.

To prove that $\mathcal{C}(\mathsf{THFSG}) \subseteq \widetilde{\mathcal{M}}(\mathcal{C}(\varepsilon\text{-}\mathit{free}\mathsf{THFSG}))$, we show that for each THFSG G there exists an ε-free THFSG G' and an NFT M such that $M(L(G')) = L(G)$. Let $G = \langle \mathcal{K}, \mathcal{S}, \Sigma, \mathcal{P}, \mathcal{L} \rangle$, and assume that $a \notin (\mathcal{K} \cup \Sigma)$. Then let $G' = \langle \mathcal{K}, \mathcal{S}, \Sigma \cup \{a\}, \mathcal{P}, \mathcal{L}' \rangle$ where $\mathcal{L}'$ is the least set such that for each

$$\begin{array}{ccc} A & \rightarrow & t \\ & & E \end{array} \tag{24}$$

in $\mathcal{L}$,

$$A \rightarrow \begin{array}{c} t' \\ E \end{array} \tag{25}$$

is a rule in $\mathcal{L}'$, where $t' = a$ whenever $t = \varepsilon$, and $t' = t$ else. Clearly, G' is an ε-free THFSG. Now define a string homomorphism h such that

$$h(t) = \begin{cases} \varepsilon & \text{if } t = a \\ t & \text{otherwise} \end{cases} \tag{26}$$

It is straightforward to see that $h(L(G')) = L(G)$. Since any string homomorphism may be represented by an NFT we know that there exists an NFT M such that $M(L(G')) = L(G)$. ∎

The results in this section with respect to NFT-images may be summarised in the following theorem.

Theorem 1

$$\begin{tikzcd} & \mathcal{C}(\text{THFSG}) & \\ \mathcal{C}(\varepsilon\text{-}\mathit{free}\text{THFSG}) \arrow[loop above, "\mathcal{M}"] \arrow[ur, "\widetilde{\mathcal{M}}"] & \subseteq & \mathcal{C}(\text{THFSG})_{\not\varepsilon} \arrow[loop above, "\mathcal{M}"'] \arrow[ul, "\widetilde{\mathcal{M}}"'] \end{tikzcd} \tag{27}$$

Proof: Directly from Lemmas 1, 2, 3, 4, and the fact that ε-free THFSGs only generate ε-free languages. ∎

3 ε-free THFSG and PSPACE

As noticed earlier, ε-free THFSG is almost identical to the grammar formalism GF1 defined by Colban [Col91]. Colban shows that the membership problem for GF1 is NP-hard. In this section we strengthen this result by stating that the membership problem for ε-free THFSG is PSPACE-complete. The difference between GF1 and ε-free THFSG —the length of attribute strings in the equations— does not make any difference here, hence the given result is also valid for GF1. In fact, consulting the proof of normal form in [Bur97b] we see that there is a polynomial (even linear) algorithm which transforms any ε-free THFSG into a GF1 grammar. Conversely, GF1 is a special case of ε-free THFSG.

We show the PSPACE completeness as usual in two steps, first showing that the problem is PSPACE-hard, and then that it is in PSPACE.

Lemma 5 *The membership problem for ε-free* THFSG *is PSPACE-hard.*

The proof is by transforming the Finite State Automata Intersection Problem (FSAI) into the membership problem for ε-free THFSG. The FSAI is given on the following instances. Given a finite set of deterministic finite state automata $A_1, \ldots, A_n$ over a common alphabet Σ, let $L(A_i)$ be the language accepted by automaton $A_i, 1 \leq i \leq n$. The FSAI question is then: is there a string $u \in \Sigma^* : u \in L(A_1) \cap \ldots \cap L(A_n)$? FSAI was shown by [Koz77] to be PSPACE-complete. Given any instance of this problem we show how to define an ε-free THFSG

G such that $t^n \in L(G)$ for a given atomic symbol t, if and only if there exists a string in the intersection as described.[3]

Since ε-free THFSG is a special case of THFSG we have the following corollary directly from Lemma 5:

Corollary 1 *The membership problem for* THFSG*s is PSPACE-hard.*

With the result from Lemma 5 we need the following lemma to establish that the membership problem is PSPACE-complete.

Lemma 6 *The membership problem for ε-free* THFSG*s is in PSPACE.*

To prove this, we show how to construct a polynomial space nondeterministic Turing Machine which decides the language defined by an ε-free THFSG. Since the construction can be done in polynomial time for any ε-free THFSG, and NPSPACE=PSPACE [Sav70], this implies that the given membership problem is in PSPACE. The main idea in the proof is that we may traverse both the c-structure and feature structure top-down simultaneously with only a limited view of the rest of the structures. This due to the homomorphism between the two structures and the restrictions on the equation schemata. [4]

From Lemma 5 and 6 we have the following result.

Theorem 2 *The membership problem for ε-free* THFSG*s is PSPACE-complete.*

4 Summary and remarks

In this article we have proved a set of closure properties for THFSG with two Restrictions regarding the empty string. We have shown that both the class of ε-free THFSG-languages and the class of languages defined by ε-free THFSGs are abstract families of languages, and that the least full-abstract family of languages including each of these two classes is the class of languages defined by THFSG. We also saw that the membership problem for ε-free THFSGs is PSPACE-complete.

There is one important open question here, and that is whether or not $\mathcal{C}$(ε-*free*THFSG) is equal to $\mathcal{C}$(THFSG)$_{\not\varepsilon}$. If this is the case, the only difference between these two subclasses and $\mathcal{C}$(THFSG) is the absence of the empty string in the languages generated. This would make it possible to define an ε-isolated normal form for THFSG in which the start symbol S never occurs on the right hand side of production rules, and the empty string only occurs on the right hand side of lexicon rules if the S occurs to the left. This kind of normal form is often used. However, the question is left for further research.

In the beginning of this article we noticed that in later LFG the empty category does not play a prominent role, even if there is some disagreement about the need for it [DKMZ95, p134] [KZ89, Bre95]. Functional uncertainty implemented as regular expressions in the equation schemata is used to handle linguistic phenomena previously handled by empty categories (e.g. long-distance dependencies). If we look at the use of regular expressions to implement functional

[3]The proof is given in [Bur98].

[4]The proof is given in [Bur98].

uncertainty as introduced in [KZ89], we only find one regular expression in each set of equation schemata, and this is a path equation schema.

Now, what about regular expressions in conjunction with ε-free THFSG? In [Bur97b] it is proved that we may introduce regular expressions in the equation schemata in THFSG without extending the class of languages described, and that the emptiness and membership problems remain decidable. The proof of equivalence between THFSG with and without the possibility of regular expressions, introduces the empty string on the right hand side in new lexicon rules. However, by examining the proof it is straightforward to see that if we restrict regular expressions to only occur in path equation schemata, a corresponding introduction of the empty string in lexicon rules is no longer needed. As a result, the class of languages we get by allowing regular expressions in the path equation schemata in ε-free THFSG is the same as the class of languages defined by clean ε-free THFSG.

Acknowledgements

I would like to thank Tore Langholm for his advice during the work that led to this paper, and for comments on early versions of the manuscript. I would also like to thank Kjell Jørgen Hole and Marianne Fjelltveit Hole for proofreading.

References

[Bre95] Joan Bresnan. Linear order, syntactic rank, and empty categories: On weak crossover. In Mary Dalrymple, Ronald M. Kaplan, John T. Maxwell III, and Annie Zaenen, editors, *Formal Issues in Lexical-Functional Grammar*, number 47 in CSLI Lecture Notes. CSLI-Publications, 1995.

[Bur97a] Tore Burheim. Emptiness, membership and regular expressions for tree homomorphic feature structure grammars. In Tilman Becker and Hans-Ulrich Krieger, editors, *Proceedings of the Fift Meeting on Mathematics of Language - MOL5*, D-97-02, Saarbrücken, Germany, 1997. Deutsches Forschungszentrum für Künstliche Intelligenz, DFKI. Short version.

[Bur97b] Tore Burheim. A grammar formalism and cross-serial dependencies. In Patrick Blackburn and Maarten de Rijke, editors, *Specifying Syntactic Structure*. CSLI-Publications, 1997.

[Bur98] Tore Burheim. Three homomorphic feature structure grammar and the empty string. Manuscript., 1998.

[Col91] Erik A. Colban. *Three Studies in Computational Semantics*. PhD thesis, University of Oslo, 1991.

[DKMZ95] Mary Dalrymple, Ronald M. Kaplan, John T. Maxwell III, and Annie Zaenen. *Formal Issues in Lexical-Functional Grammar*. Number 47 in CSLI Lecture Notes. CSLI Publications, 1995.

[Gin75] Seymour Ginsburg. *Algebraic and Automata-Theoretic Properties of Formal Languages.* North-Holland Publishing Company, Amsterdam, The Netherlands, 1975.

[KB82] Ronald M. Kaplan and Joan Bresnan. Lexical-Functional Grammar: A formal system for grammatical representation. In Joan Bresnan, editor, *The Mental Representation of Grammatical Relations.* The MIT-Press, Cambridge, Massachusetts, USA, 1982.

[Koz77] D. Kozen. Lower bounds for natural proof systems. In *Proc. of 18th. Annual Symposium on Foundations of Computer Science*, pages 254–266, Long Beach, USA, 1977. IEEE Computer Society.

[KZ89] Ronald M. Kaplan and Annie Zaenen. Long-distance dependencies, constituent structure, and functional uncertainty. In Mark R. Baltin and Anthony S. Kroch, editors, *Alternative conceptions of phrase structure*, pages 17–42. The University of Chicago Press, 1989.

[Sav70] W.J. Savitch. Relationship between nondeterministic and deterministic tape complexity. *Journal of Computing System Science*, 4:177–192, 1970.