ANALYZING FAMILIES OF GRAMMARS

Eberhard Bertsch
Universität des Saarlandes

The problem of designing general parsing procedures for arbitrary families of languages has hardly been studied. One will usually be satisfied with general procedures for the class of LR(k)-languages, the class of precedence languages, the class of regular languages, etc. For the types of abstract families of languages (AFL) studied by Cremers and Ginsburg [3] , the availability of general procedures may be of equal importance, however. To put it more precisely, if two grammars are related by belonging to the same family, it will be desirable to be able to use the same parser for both of them.
In the present paper we investigate the relationship between structurally similar grammars from the parsing point of view. Apart from providing an interesting parsing method, our approach will yield time bounds for the word problems of related grammars [2] .
Our technique is basically a three-pass mechanism which will analyze a given language L(G), if the first pass analyzes a language L(G′) such that G can be mapped on G′. The second pass is based on a construction in [7] which is there used to obtain a decidability result. It takes the output of the first pass as its input. The third pass produces a derivation of the original word, if one exists, or else prints an error message.

Some definitions of structural relatedness

Definitions 1 and 2 are slight modifications of concepts defined in [3].

Definition A context-free grammar form is a 6-tuple $F=(V,T,V_F,T_F,P,S)$ where (i) $T \subseteq V$, $T_F \subseteq T$, $(V_F \setminus T_F) \subseteq (V \setminus T)$, $T_F \subseteq T$
(ii) $G_F=(V_F,T_F,P,S)$ is a context-free grammar.

G_F is called the form grammar of the grammar form F.

Definition A strict interpretation of a grammar form $F=(V,T,V_F,T_F,P,S)$ is a 5-tuple $I=(\mu,V_I,T_I,P_I,S_I)$ where

(1) μ is a substitution on V^* such that
  (a) $\mu(a)$ is a finite subset of T for each $a \in T_F$
  (b) $\mu(A)$ is a finite subset of $V \setminus T$ for each $A \in V_F \setminus T_F$
  (c) $\mu(A) \cap \mu(B) = \emptyset$ for $A,B \in V_F$, $A \neq B$

(2) P_I is a subset of

$$\mu(P) = \bigcup_{p \in P} \mu(p)$$

where $\mu(A \longrightarrow B) := \{ u \longrightarrow v \mid u \in \mu(A),\ v \in \mu(B) \}$

(3) S_I is in $\mu(S)$

(4) T_I is the set of symbols in T occurring in P_I.
  V_I is the set of symbols in V occurring in P_I augmented by S_I.

The context-free grammar $G_I=(V_I,T_I,P_I,S_I)$ is called the grammar of I.

Definition 3 is an essential concept for the categorial treatment of grammars presented in [5], [6]. By M we mean the set of derivations m of a grammar G, s(m) is the source of m, t(m) is the target of m.

Definition For two grammars $G_1=(V_1,T_1,P_1,S_1)$ and $G_2=(V_2,T_2,P_2,S_2)$, an x-functor $\varphi: G_1 \longrightarrow G_2$ is a pair of mappings (φ_1,φ_2) where $\varphi_1: V_1^* \longrightarrow V_2^*$, $\varphi_2: M_1 \longrightarrow M_2$ such that φ_1 and φ_2 are homomorphisms, that is

$$\left.\begin{aligned} \varphi_1(AB) &= \varphi_1(A)\,\varphi_1(B) \quad \text{for } A,B \in V_1^* \\ \varphi_2(m_1 \circ m_2) &= \varphi_2(m_1) \circ \varphi_2(m_2) \\ \varphi_2(m_1 \times m_2) &= \varphi_2(m_1) \times \varphi_2(m_2) \end{aligned}\right\} \text{ for } m_1,m_2 \in M_1$$

and φ_1 is compatible with φ_2, that is

$$\left.\begin{aligned} \varphi_1(s(m)) &= s(\varphi_2(m)) \\ \varphi_1(t(m)) &= t(\varphi_2(m)) \end{aligned}\right\} \quad \text{for } m \in M_1$$

φ is a length-preserving functor if $\varphi_1(V_1 \setminus T_1) \subseteq V_2 \setminus T_2$, $\varphi_2(P_1) \subseteq P_2$, $\varphi_1(S_1) = S_2$, $\varphi_1(T_1) = T_2$.
Using a proposition in [5], it can be shown that the following theorem connects functors with interpretations of grammar forms:
Theorem: Let G_F be the form grammar of a c.f.grammar form F. G_I is the grammar of a strict interpretation of F, if and only if there exists a length-preserving x-functor $\varphi: G_I \longrightarrow G_F$.

Our parsing mechanism

The intuitive meaning of a tree-automaton is easy to understand [7] , [8] , [2] . We will therefore skip the formal definition and explain the underlying idea in less rigorous terms.

Imprecise definition A tree-automaton $\mathcal{A}$ relative to a grammar G possesses an alphabet T and a transition function h. Given any derivation tree m of G, whose terminal nodes are $X_1,\ldots,X_n$ in left-to-right order, h assigns a value $h_m(X_1\ldots X_n)$ to the top node of m. This value is calculated by going from bottom to top and assigning a letter of T to each node. Which letter is taken, depends on which letters have been assigned to the immediately dependent nodes.

Now we need some notation which will be used in the proof of the following theorem. Suppose that we have two grammars $G_1=(V_1,T_1,P_1,s_1)$, $G_2=(V_2,T_2,P_2,s_2)$ and a length-preserving x-functor $\varphi : G_1 \longrightarrow G_2$. We construct an automaton $\mathcal{A}= (T,h)$ relative to G_2. T is identified with $2^{(V_1)}$. h is given by

$$h_p(X_1\ldots X_{|t(p)|}) := \{ A\in V_1 \mid \exists\ \bar{p} = A \longrightarrow A_1\ldots A_{|t(p)|}\in P_1 \text{ with } \varphi_2(\bar{p}) = p \text{ and } A_i\in X_i\ (i \le |t(p)|) \}$$

for all $p\in P_2$.

By a lemma in [7], we know that then for all $m\in M_2$ $h_m(X_1\ldots X_{|t(m)|}) = \{ A\in V_1^* \mid \exists\ \bar{m}\in M_1$ with $s(\bar{m}) = A$, $t(\bar{m}) = A_1\ldots A_{|t(m)|}$, $\varphi_2(\bar{m}) = m$ and $A_i\in X_i\ (i\le |t(m)|)\}$.

The main result

This gives us our next theorem, whose proof will also contain a description of how the tree-automaton is to be employed.

Theorem: Let $\varphi : G_1 \longrightarrow G_2$ be a length-preserving functor, where G_1 and G_2 are c.f.grammars and G_2 is unambiguous. Suppose there is an algorithm which will construct a parse for $w\in L(G_2)$ and reject $w\notin L(G_2)$ in less than $f(|w|)$ steps. Then there is a constant c and an algorithm which will accept $w\in L(G_1)$ and reject $w\notin L(G_1)$ in less than $cf(|w|)$ steps.

Proof: Let $w \in T_1^*$. If there is an $m \in M_1$ with $t(m)=w$ and $s(m)=s_1$, then by the definition of φ there exists an $m' \in M_2$ with $\varphi_2(m)=m'$, $s(m')=s_2$, $t(m')=w' \in T_2^*$. By the unambiguity of G_2, there is no $m'' \in M_2$ with $s(m'')=s_2$, $t(m'')=w'$, such that $\varphi_2(m) \neq m''$. So we construct $w'=\varphi_1(w)$ in realtime and try to find a parse of w' in G_2. If there is none, we have $w \notin L(G_1)$. If there is one, we have to check whether there is a coimage m of $m' \in M_2$ such that $s(m)=s_1, t(m)=w$. By the construction of our finite automaton, this amounts to calculating $h_{m'}(\{a_1\}\{a_2\} \ldots \{a_n\})$ where $w=a_1 \ldots a_n$. m' has a coimage whose root is s_1 if and only if s_1 is an element of the resulting set. Let n_m be the number of nodes in a derivation tree $m \in M_2$. Simulating tree-automata in time cn_m poses no problem. We simply represent parse trees by words of the bracketed c.f.language [4] which is generated by the rules $P_b := \{\alpha \rightarrow [_p \beta]_p \mid p = \alpha \rightarrow \beta \in P_2\}$. Then each transition of our tree-automaton corresponds in an obvious way to a reduction step on a sentential form. As shown in [4], a reduction sequence can be obtained in linear time.

So far we have just specified how a yes-no decision can be reached. To use our mechanism as a parser in the strict sense of the word, some additional considerations are necessary. What we have to do, is to store a trace of all direct transitions and walk back from top to bottom. Within our categorial framework, this can be compactly stated as follows:

Suppose $h_m(X_1 \ldots X_z)=X$ where $m=\bar{m} \circ \hat{m} \circ \tilde{m}$, $h_{\bar{m}}(X_1 \ldots X_z)=\bar{X}_{11} \ldots \bar{X}_{1k_1} \ldots \bar{X}_{lk_l}$ such that $h_{\hat{m}}(\bar{X}_{11} \ldots \bar{X}_{1k_1} \ldots \bar{X}_{lk_l})=\hat{X}_1 \ldots \hat{X}_l$ and $\hat{m}=(m_1 \times \ldots \times m_l)$, $m_i \in P_1 \cup Id_{V_2}$ with
$h_{m_i}(\bar{X}_{i_1} \ldots \bar{X}_{ik_i})=\hat{X}_i$ for $i \leq l$.

Suppose further that $\mathrm{Tree}(\tilde{m}, \hat{X}_1 \ldots \hat{X}_l)$ has been defined. If $m_i \in P_1$, take a rule $p_i \in P_1$ with $s(p) \in \hat{X}_i, t(p) \in \bar{X}_{i1} \ldots \bar{X}_{ik_i}$ $(i \leq l)$ and construct $\mathrm{Tree}(\hat{m} \circ \tilde{m}, \bar{X}_{11} \ldots \bar{X}_{lk_l}) := (m_1' \times \ldots \times m_l') \circ \mathrm{Tree}(\tilde{m}, \hat{X}_1 \ldots \hat{X}_l)$ where $m_i'=p_i$ if $m_i \in P_1$ and $m_i' \in Id_{V_2}$ otherwise.

To start this top-to-bottom process, we have to set $\mathrm{Tree}(p, X_1 \ldots X_n)$ as a rule $p_{start}=A \rightarrow A_1 \ldots A_n$ in P_2 with $A \in X$, $A_i \in X_i$ for $i \leq n$ and $h_p(X_1 \ldots X_n)=X$.

List of references (necessarily incomplete, as the subject matter touches several rapidly growing fields of study):

1) Bertsch,E.: Surjectivity of functors on grammars, appearing in Mathematical Systems Theory, vol.10, no.1
2) Bertsch,E.: An observation on relative parsing time, appearing in Journal of the ACM, vol.22, no.3
3) Cremers,A. and S.Ginsburg: Context-free grammar forms, in Lecture Notes in Computer Science, vol.14
4) Ginsburg,S. and M.Harrison: Bracketed context-free languages, Journal of Computer and System Sciences (1967)
5) Hotz,G.: Eindeutigkeit und Mehrdeutigkeit formaler Sprachen, Elektr. Inform. und Kybern. (1966)
6) Hotz,G. and V.Claus: Automatentheorie und formale Sprachen, vol.3, BI-HTB 823a, Duden-Verlag (1973)
7) Schnorr,C.: Transformational Classes of Grammars, Information and Control (1969)
8) Thatcher,J.: Characterizing derivation trees of context-free grammars through a generalization of finite automata-theory, Journal of Computer and System Sciences (1967)