Bericht Nr. 82, Januar 1997

Arbeitspapiere des
Sonderforschungsbereichs 340

ISSN 0947-6954/97

Sprachtheoretische Grundlagen für die Computerlinguistik

Hugo Volger :

Principle Languages and Principle Based Parsing

Universität Stuttgart Universität Tübingen
IBM Deutschland Informationssysteme GmbH

Arbeitspapiere des SFB 340
Bericht Nr. 82, Januar 1997

PRINCIPLE LANGUAGES AND PRINCIPLE BASED PARSING

HUGO VOLGER

ABSTRACT. This paper contains a study of principle-based grammar formalisms i.e. formalisms based on the principle-and-parameter paradigm, where grammars are sets of principles or constraints. In this approach the set XG of principles of a grammar for a language X consists of the properties which characterize the parse trees of the grammatical sentences of X. In addition, there is a universal set of parametrized principles, the universal grammar UG, from which the grammar XG is obtained by an appropriate specialization of the parameter values.

We advocate the use of a dedicated principle language to describe the class of parse trees of a given grammar. Combined with a compiler derived from a formula evaluator such a language yields a flexible approach to principle checking. In Passau we have been using a first order language for attributed trees called $Clat(A)$ where A is the set of attribute values. In this paper we discuss two expressivity results for Clat of A.Palm [23] and compare it with related results for the other dedicated principle languages. The result for a finite value set is not new but the explicit description of the translation may yield new results for special cases. However, the result for stacks as values is new.

In addition, we suggest that a principle-based parser should respect the central tenets of the principle-and-parameter paradigm. Therefore we require that such a parser should use a set of universal parsing operations. However, the applicability of a parsing operation in a particular situation is controlled by the so called internal grammar introduced by K.-M.Schneider [30]. It is a set of nodefree relations derived from the set of principles of the grammar. In the case of local principles he obtained a quite successful implementation. However, in the non-local case a feasible extension of the approach remains to be found.

To appear in: Kolb, H.-P. & U. Mönnich (eds.), *The Mathematics of Syntactic Structure.*

ISSN 0947-6954/97

1. Introduction

Within the last 15 years Chomsky's GB-theory (cf. [9]), the *Theory of Government and Binding*, also referred to under the more general term of *principles-and- parameters-model*, has been established as a grammar formalism adaquate to describe syntactical knowledge in a way that gives answers to questions concerning language acquisition and universal properties of languages.

In a joint project of the author and his colleague Sascha Felix in linguistics our group in Passau has started to develop a set of tools for the study of the GB-theory. First of all a linguist working with GB needs a principle checker which at any moment can test the candidate parse trees in his database w.r.t. his favorite set of principles. Our principle compiler for our language Clat serves this purpose quite nicely. In addition, a more ambitious linguist might like to have an associated principle-based parser i.e. a parser producing parse trees satisfying his set of principles. At least we can offer a parser which is principle-based w.r.t. a class of universal-existential local principles.

As a logician the author is interested in general properties of principle-based grammar formalisms. This applies to GB and to a lesser degree to grammar formalisms with principle-based aspects like GPSG, HPSG, PATRII and others ([19], [24], [33]). Therefore this report should be viewed as a conceptual study of the two basic tasks of principle checking and principle-based parsing. The results which will be reported here are based on results of A.Palm [22],[23] and K.-M.Schneider [30],[31].

1.1. **The principles-and-parameter paradigm.**

In the well known rule-based approach the syntactical representations i.e. parse trees of grammatical sentences are generated by means of a set of rules. As a consequence this procedural approach provides information which can be used for the parsing process. But general properties of parse trees are difficult to prove. However, in the principle-based approach to be studied here the parse trees of grammatical sentences are singled out by a set of principles i.e. wellformedness conditions. By definition this declarative approach yields information on general properties of parse trees. But now the information for the parsing process is difficult to obtain.

The principles-and-parameters paradigm on which the GB-theory is based can be summarized as follows:

1. Constraints:
   The grammar of a language X consists of a set XG of *principles* (= wellformedness conditions) for the parse trees. The parse trees of the

grammatical sentences of X are the trees satisfying the principles in X.

2. Universality:
There is a *universal grammar* UG i.e. a set of parametrized principles such that each grammar XG can be obtained by an appropriate specialization of the parameter values.
3. Modularity:
The set of principles is organized into a set of *modules* which are related by their dependency relation. The set of modules satisfied by a sentence yields a degree structure on the notion of grammaticality. Thus grammaticality and parsability can be distinguished i.e. certain ungrammatical sentences can still be parsed, and conversely.

A typical example of a UG-principle is the case filter. It requires that a case must be assigned to each noun phrase. The direction of the assignment and the posssible requirement of adjacency in a given grammar are determined by parameters. The basic GB-module is the XBar-theory. Other examples of modules are Θ-theory or binding-theory.

The claims on universality and modularity described above have cognitive and physiological motivations from language learning resp. neurological disorders. These arguments need not be not be repeated here. However, they suggest that the principles of the universal grammar should be considered as phenomenological constraints on possible constituent trees. More precisely, the principles should be viewed as the result of analyzing the output of the unknown human parser.

Therefore it does not come as a surprise that the principle-and-parameter-paradigm of GB-theory does not state where the candidate trees come from. The formalism does not offer any hints for the parsing process. We are looking for a principle-based parser i.e. a parser for natural languages which is cognitively realistic in the following sense. It should exhibit the same behaviour with respect to difficulties and reaction times as the unknown human parser.

1.2. **Principle Checking.** The implementation of principle-based grammar formalisms involves two main tasks. The first task to be solved is principle checking i.e. to develop a procedure for testing candidate parse trees whether they satisfy a given set of principles. First one realizes that the GB-theory does not come with a formalization.

To start working with it one has to choose a *principle language* in which the existing and hopefully also future principles of GB can be formalized. Having done this a principle compiler can be developped which transforms a set of principles into an executable program which can test potential parse trees for grammaticality. This yields a uniform approach to principle checking.

We suggest to use dedicated principle languages i.e. languages which describe finite attributed trees. In Passau we have worked quite successfully with a first order principle language with an efficient principle compiler. Below we present a powerful expressivity result for our language which was obtained by A.Palm and we compare it with results for other languages which have been used.

1.3. **Principle-based Parsing.** The second task to be solved is principle-based parsing i.e. to write for a given set of principles a parser which is adapted to the principle-and-parameter approach of the grammar formalism. However, the principle-and-parameter paradigm of the GB-theory does not state where the trees come from which could be submitted to the principle-checker. As the theory does not make any claims w.r.t. parsing the requirements of a principle-based parser have to be clarified.

Principles are constraints satisfied by grammatical parse trees. Viewing principles as filters for potential parse trees constitutes a straightforward operationalization of this notion. This view suggests a generate-and-test approach to parsing. However, as mentioned above, the theory does not offer any hint how the input sentence could be used to generate a set of candidate trees which can be submitted to the filters. Generating all trees from an unspecific class of trees leads to the problem of overgeneration which plagues the generate-and-test-parsers. Alternating generation - and filter phases as in requires a tuning for which linguistical motivations are hard to come by (cf. Fong [17]). We doubt whether this simple operationalization which underlies the generate-and-test approach to parsing is an essential part of the principle-based approach.

The universality and modularity are the central aspects the principles-and-parameters paradigm. Hence it is seems reasonable to extend the paradigm by the requirement that a *principle-based parser* should reflect these two aspects in an essential manner. By this approach we might hope to arrive at useful approximations of the unknown human parser.

We think that a principle-based parser should be a *parser generator* for the universal grammar. It produces a parser for a particular grammar by a partial evaluation w.r.t. the parameter values of the grammar. Thus the parser generator separates the aspects of parsing from those of grammatical knowledge. Following ideas of K.-M.Schneider in [30] we suggest to collect the information of a particular grammar in the so called *internal* *grammar* i.e. a set of nodefree relations over attribute values. Analyzing the different types of information contained in the principle formulas one can separate the parser operation which a parser might apply from the information in the internal grammar which controls the applicability of the parser operation. In the case of local universal-existential principles a quite efficient parser which is principle-based in the sense just discussed has been implemented by K.-M.Schneider [30].

2. Languages for Principle Checking

A *principle language* is a language in which properties of finite attributed trees can be expressed and a *principle checker* is a compiler which transforms properties expressed in the principle language into executable test programs. Clearly, the success of principle checking depends on the choice of an powerful language and the implementation of an efficient compiler. Below we shall present the principle language we are using in Passau and discuss several other principle languages which have appeared in the literature.

2.1. **Ad-hoc vs. dedicated principle languages.** In the late 80's several GB-parsers have been implemented which relied on a general programming language e.g. Prolog (cf. [1],[17]). They used a general purpose programming language to formalize the principles and to implement the principle checker. Such a language should be called an *ad-hoc principle language*. Using a such language, call it $\mathcal{L}$, has the following disadvantages. First the data type of attributed trees has to be expressed in $\mathcal{L}$. Then the principles must be formalized using the facilities provided by $\mathcal{L}$ and a principle checking procedure must be written. Finally a compiler for $\mathcal{L}$ is used. Efficiency problems are to be expected. Moreover, usually the formalization of a principle depends on the type of principle to be expressed. Hence a change of the set of principles requires a major revision of the principle checking component.

We favor a second approach which makes use of a *dedicated* principle language i.e. a formal language in which properties of trees can be expressed. Formulas of such a language describe classes of attributed trees i.e. subclasses of the class $Trees(A)$ of ordered trees with finite branching whose nodes are labelled with values from a given set A of attribute values. A is assumed to be a finite set unless explicitly stated otherwise. The class of one-b-trees will be used as a running example. It consists of those trees in $Trees(\{a, b\})$ which have exactly one node which is labelled with b. As another example we shall use the class of even-depth trees.

It consists of those trees in $Trees(\{a\})$ whose branches are all of even length.

A principle compiler for such a language is a formula evaluator i.e. a program which describes the evaluation of formulas in attributed trees. The combination of a dedicated principle language with its evaluator yields an *uniform approach* to principle checking which has the following advantages: It is *simple to use* since the dedicated language simplifies the task of formalizing the principles. It is *flexible* since a change of the set of principles requires just one compilation of the new set of principles and the available data can be tested with the new test program. It yields an *efficient compiler* since the evaluator can be implemented in a well chosen implementation language.

It should be noted that we restrict the discussion to languages for attributed trees i.e. to grammar formalisms which use attributed trees as syntactical representations. More generally, one could consider grammar formalism based on feature graphs. The associated attribute-value languages were studied by Rounds and Kasper, Keller and others (cf.[6]). In these languages the constituent tree structure has to be coded into the feature graph. Clearly, many results for tree languages can be extended to feature graph languages.

2.2. **The first-order principle language Clat.** In Passau (cf. [14],[22]) we have developped a tool for principle checking. It is based on the dedicated principle language called $Clat$, constraint language for attributed trees, together with a quite efficient principle compiler. $Clat(A)$ is a first order language for the class $Trees(A)$ of ordered trees whose nodes are labelled with values from a given set A of attribute values. In general the value set A may be a relational structure. However, unless explicitly stated otherwise (e.g. in 2.5 and 4.2) the value set A is assumed to be a finite set without further structure.

The language has variables for nodes and hence quantification for nodes but no variables for attribute values. For convenience we assume a constant *root* for the root. Under the finiteness assumption on A adding variables and quantification for attribute values does not extend the expressivity of the language. As usual the formulas are built up from the atomic formulas(= atomic constraints) using the propositional connectives and the existential and universal quantifier. Note that we use the symbols $\Rightarrow$ and *biimp* to denote the implication and biimplication since the simple arrows will be used for the directions in the tree. Conjunctions of literals i.e. atomic or negated atomic formulas are called constraints. –

Actually the original version of this language in [14] contained also some definable operations like conditional functional uncertainties.

For the following we need a more detailed description of $Clat(A)$. Let $\Delta = \{\downarrow, \rightarrow, \uparrow, \leftarrow\}$, the set of *local directions*, be the set of structural predicates for the local relations in ordered trees. Here $\downarrow$ resp. $\rightarrow$ are the predicates for the immediate dominance - resp. the immediate left sibling relation and $\uparrow$ and $\leftarrow$ are the predicates for the converse relations of $\downarrow$ and $\rightarrow$. Then $\Delta^+ = \{\uparrow^+, \rightarrow^+, \downarrow^+, \leftarrow^+\}$, the set of *nonlocal directions*, consists of the irreflexive transitive closures of the relations in Δ i.e. the predicates for the dominance - and the left sibling relation with their converses. However, it should be noted that only in finite trees the dominance - resp. left sibling relation coincides with the irreflexive transitive closure of $\downarrow$ resp. $\rightarrow$.

In addition, we have as value predicates the unary value predicates $|x| = a$ for each $a \in A$ and the binary value equality $|x| = |y|$. Thus $Clat(A)$ is the first order language $FO(\Delta^+, A)$ with:

1. binary structural predicates from Δ^+
2. unary value predicates for the elements of A and the value equality
3. possibly other predicates and operations of the structure A.

Assuming an axiomatization of the class $Trees(\{a, b\})$ the class of finite one-b trees consists of the finite models of the following sentence of $FO(\Delta^+, \{a, b\})$:

$$\exists x(|x| = b \wedge \forall y(y = x \vee |y| = a))$$

However, the class of even-depth trees cannot be expressed in $FO(\Delta^+, \{a\})$ since discrete orders of even length are not first order expressible (cf.[12]).

The linear precedence relation $\prec$ and its converse $\succ$ are first order definable from Δ^+. Moreover, in finite ordered trees also the following definition can be used: $\prec := \uparrow^+ \cdot \rightarrow^+ \cdot \downarrow^+$. In addition, in finite ordered trees we have the *unique structural relation* lemma which is crucial for later results:

$$\forall x \forall y((x \prec y) \oplus (x \downarrow^+ y) \oplus (x = y) \oplus (x \uparrow^+ y) \oplus (x \succ y))$$

where $\vee$ denotes the exclusive disjunction. It can be used to eliminate negative literals in structural constraints.

If the set A of attribute values is finite i.e. $A = \{a_1, \ldots, a_k\}$ then the *unique value* lemma plays a similar role:

$$\forall x((|x| = a_1) \oplus \ldots \oplus (|x| = a_k))$$

. Again this may be used to eliminate negated equations of the form $x \neq a_i$. In addition, the value equality is definable from the value predicates.

2.3. **Variants of the language.** The following variations of the set of structural predicates yield an equivalent language. The set Δ^+ could be replaced by $\{\downarrow^+, \rightarrow^+\}$ since the converse of a relation is first order definable. Moreover, the local predicates in Δ could be added since they are definable from the nonlocal versions. The linear precedence relation $\prec := \uparrow^+ \cdot \rightarrow^+ \cdot \downarrow^+$ and the left sibling relation $\rightarrow^+$ are first order definable from each other in the presence of $\downarrow^+$. Hence we may replace the set $\{\downarrow^+, \rightarrow^+\}$ by $\Gamma = \{\downarrow^+, \prec\}$.

We obtain a version $Clat_0(A)$ of $Clat(A)$ which is more local if we consider the language $FO(\Delta, A)$ which has only local structural relations. It is well known that the relation $\downarrow^+$ is not first order definable from $\downarrow$. Thus this more local version is strictly weaker. However, it may be questioned whether this fragment contains only local formulas as the negation of a local structural relation yields a disjunction of nonlocal relations.

The unary value predicates $|x| = a$ for the elements a of a finite value set A define a finite partition on the set of nodes of a tree. Rather than using the finite value set A and its associated set of value predicates in our language one might just use unary predicates Q_a for a in A. Assuming the unique value lemma as axiom one obtains an equivalent language. More generally, the variant of the language with finite set $Q_1, \ldots, Q_k$ of unary predicates without any further axiom is equivalent to the language with a value set of 2^k elements.

If one wants to consider *trees with projection lines* rather than ordinary trees as intended models then a simple variant of the above can be used. One uses the same language but in this case the predicates $\downarrow^+$ resp. $\downarrow$ will have to be interpreted by the projection – resp. immediate projection relation. This variant will be used later on in connection with linear index grammars.

2.4. **Products of value sets and modularity.** In GB we do not have one homogeneous value set A but rather several value sets like $Case$, $Number$, $\Theta - Roles$ and others. To obtain an adequate description one assumes that the value set is of the form $A = \hat{A}_1 \times \cdots \times \hat{A}_k$ where $\hat{A}_i$ is the extension of the value set A_i by a special element which expresses that there exists no value from A_i. Thus the values of the nodes are k-tuples of values in this situation.

The consideration of products of value sets yields a strong notion of modularity. To simplify the discussion we consider a value set $A_1 \times A_2$ of two factors. There are projection maps pr_i from $Trees(A_1 \times A_2)$ to $Trees(A_i)$ for $i = 1, 2$ which are obtained by forgetting the other component of the node values. The value predicates $|x| = (a_1, a_2)$ in $FO(\Delta^+, A_1 \times A_2)$ are definable from the component value predicates $|x|_1 = a_1$ and $|x|_2 = a_2$ in $FO(\Delta^+, A_1)$ resp. $FO(\Delta^+, A_1 \times A_2)$. The converse is true if the value sets A_1 and A_2 are both finite:

$$|x|_1 = a_1 \Leftrightarrow \bigvee_{a_2 \in A_2} |x| = (a_1, a_2) \qquad |x|_2 = a_2 \Leftrightarrow \bigvee_{a_1 \in A_1} |x| = (a_1, a_2)$$

$$|x| = (a_1, a_2) \Leftrightarrow |x|_1 = a_1 \wedge |x|_2 = a_2$$

Hence there are inclusion maps in_i from $FO(\Delta^+, A_i)$ to $FO(\Delta^+, A_1 \times A_2)$ for $i = 1, 2$ which are obtained by replacing $|x| = a_i$ by $|x|_i = a_i$. They behave as desired since we have:

$$T \models in_i(\varphi) \text{ iff } pr_i(T) \models \varphi$$

Hence a formula $\varphi \in FO(\Delta^+, A_1 \times A_2)$ is said to be *strongly modular w.r.t.* $A_1 \times A_2$ if there exists $i \in \{1, 2\}$ and a formula $\psi \in FO(\Delta^+, A_i)$ such that φ is equivalent to $in_i(\psi)$. This states that φ can be evaluated in $Trees(A_i)$ rather than $Trees(A_1 \times A_2)$.

2.5. **Infinite value structures.** A value set A may be considered as a relational structure $(A, (a \in A))$ with constants for the elements of A. More generally, we may consider value structures B where B is a relational structure over a signature Σ. In this case the atomic value constraints are atomic formulas over Σ whose terms generated by node values $|x|$ for a node variable x or constants in Σ.

A simple example is the counter structure $Counter = (N, s, p, 0)$ where s resp. d are the sucessor - resp. predecessor function. Below in 4.2 we shall consider the stack structure $Stacks(I) = (I^*, (push_i : i \in I), pop, empty)$ with stack elements from I. Now $p(|x|) = 0$ and $s(|x|) = |y|$ are atomic value constraints for the counter structure and $pop(|x|) = empty$ and $push_i(|x|) = |y|$ are atomic value constraintsa for the stack structure.

One place where infinite value structures have to be used is the treatment of coreferences between nodes in a tree. In GB coreferent nodes are used in the theory of binding and each movement creates a coreference between the moved element and its trace. Viewed abstractly, a coreference on a tree T is a partial function on T i.e. a functional binary relation G on T. A coreference pair for G is a pair (t_1, t_2) of nodes in G and t_1 is called the antecedent and t_2 is called the succedent of the pair. In general, one gets coreference chains as the succedent of a pair may be an antecedent of another pair. The pairs in coreference chain are determined by the transitive closure G^+ of the relation G.

To be able to talk in our language about coreferent nodes w.r.t. to a given coreference we must require that the coreference G is definable by a formula $\gamma(x, y)$ in $FO(\Gamma, A)$. Using indexing we can describe the pairs of an arbitrary coreference in our language. To do this we consider the infinite value structure $(N \times 2, g)$ where $g((n, 1)) = (n, 0)$ and $g((n, 0)) = (n, 0)$. Then $\gamma(x, y) \ := \ |x|_1 = |y|_1 \wedge |x|_2 = 1 \wedge |y|_2 = 0$ defines the coreference pairs where $|x|_1$ is the index of the pair (x, y).

Using a stronger form of indexing we can also describe the coreference chains. To do this we consider the infinite value structure $(N \times N, g^+)$ where $g^+((n, i)) = (n, i-1)$ for $i > 0$ and $g^+((n, 0)) = (n, 0)$. Then $\gamma^+(x, y) \ := \ |x|_1 = |y|_1 \wedge |x|_2 > |y|_2$ defines the coreferent elements of a chain where $|x|_1$ is the index of the coreference chain containing x and y. This is closely related to the free indexation considered in Rogers [27]. Later on we shall use a more restricted form of indexing which is based on the stack structure $Stacks(I)$ mentioned above.

Our experience with $Clat(A)$ has shown that many principles of GB can be expressed in $Clat(A)$ with a finite value set A and that all principles of GB can be expressed with an appropriate structure A.

3. A survey of dedicated principle languages

In the literature several other dedicated principle languages have been considered. They may divided into three groups. The first group consists of extensions of first order logic which are monadic tree languages i.e. all the nonstructural predicates are monadic. The next group consists of multimodal propositional languages which may be viewed as fragments of languages of the first group. The last group contains classes of grammars since grammars may be viewed as formulas of a principle language. These languages may be viewed as fragments of languages of the second group.

3.1. **Extensions of first order principle languages.** Extending the language $FO(\Gamma, A)$ by monadic second order (= set) variables one obtains the powerful language $MSO(\Gamma, A)$ of *monadic second order logic*. It was studied by Rogers in [26] under the name of L^2_{KP}. For later use we note that a set variable X can be simulated by an extension of the value set A to $A \times 2$ if $X(y)$ is replaced by $|y| = 1$. But we have to apply the projection onto A to the trees in $Trees(A \times 2)$. Thus the extension from A to $A \times 2$ corresponds to an existential quantifier over one set variable.

In particular, the language $MSO(\Gamma, A)$ contains as fragment the *monadic least fixpoint language* $MLFP(\Gamma, A)$. It is the extension of $FO(\Gamma, A)$ by least fixpoints for definable set operators which are strongly monotonic.

These operators are determined by a formula $\psi(x, X)$ containing two free variables x, X where x is a first order variable and a X set variable which may occur only positively. X receives the input set and the elements interpreting x which make the formula true produce the output set of the operator. Using quantification over set variables the least fixpoint $[LFP_{y,Y}\psi(y, Y)](x)$ can be defined in $MSO(\Gamma, A)$ by the following formula:

$$Y(x) \wedge \forall y(\psi(y, Y) \Rightarrow Y(y)) \wedge \forall Z(\forall z(\psi(z, Z) \Rightarrow Z(z)) \Rightarrow \forall z(Y(z) \Rightarrow Z(z)))$$

The monadic least fixpoint language in turn contains as fragment the *monadic transitive closure language* $MTC(\Gamma, A)$. It is the extension of $FO(\Gamma, A)$ by transitive closures of definable binary relations. Clearly, the transitive closure $[TC_{u,v}\varphi(u, v)](x, y)$ is definable in $MLFP(\Gamma, A)$ as a least fixpoint of an appropriate operator as follows:

$$[LFP_{z,Z}(\varphi(x, z) \vee \exists u(Z(u) \wedge \varphi(u, z)))](y)$$

It should be noted that the monadic transitive closure language is a proper extension of the first order language $FO(\Gamma, A)$. We can describe the class of *even depth trees* i.e. the finite trees where all branches have even length as follows:

$$\forall z(\neg\exists y \downarrow (x, y)) \Rightarrow [TC_{x,y}(\exists u(\downarrow (x, u) \wedge \downarrow (u, y)))](root, z)$$

However, it is well known that the set of discrete orders of even length is not definable in $FO(\Gamma, A)$ (cf.[12]).

3.2. **Multimodal propositional principle languages.** Rather than looking at monadic tree languages which are extensions of first order logic one may look at multimodal propositional languages. The trees together with the unary relations for the values from A may be viewed as a family of propositional structures parametrized by the nodes. The propositional structures are truthvalue valuations for propositional variables associated with the values from A. Thus we obtain tree frames in which extensions of propositional logic by modal operators for the structural relations of the trees may be interpreted. This motivates the study of propositional multimodal logics whose modal operators for the directions correspond to first order tree quantification along the associated structural relations.

Blackburn et al. in [5] studied and axiomatized the *propositional multimodal logic* $PML(\Delta^+ \cup \Delta, A)$ with the set $\Delta^+ \cup \Delta$ of structural relations as modal operators and the set A as propositional variables. Replacing a modal operator for the structural relation d by the restricted existential quantification along d one can define the model preserving standard translation st_1 from $PML(\Delta^+ \cup \Delta, A)$ to $FO^2(\Delta^+ \cup \Delta, A)$ the fragment of $FO(\Delta^+ \cup \Delta, A)$ consisting of sentences with at most 2 bound variables. $st_1(\varphi)$ is defined as $\forall x st_x(\varphi)$ and $st_x(a) = |x| = a$ and $st_x(\langle d\rangle\varphi) = \exists y(d(x,y) \wedge st_y(\varphi))$ show how to define st_x. This shows that the modal logic may be viewed as a fragment of the first order logic $FO^2(\Delta^+ \cup \Delta, A)$. – As a local fragment one obtains the fairly weak modal logic $PML(\Delta, A)$ with a translation st_0 from $PML(\Delta, A)$ to the local first order logic $FO(\Delta, A)$.

Kracht [20] studied the even stronger *propositional dynamic logic* $PDL(\Delta, A)$ i.e. a propositional multimodal logic with A as set of propositional variables and the set of modal operators is generated the regular operations from Δ and the test operator ? applied to formulas. The presence of the iteration $(-)^*$ is an obstacle for a translation into first order logic. But we can extend st_1 to a translation st_2 into $MTC(\Gamma, A)$. To do this one expresses the set of nodes reached by an iterated modal operator from a given start node by means of transitive closure.

Later on we shall use $PDL_1(\Delta, A))$, the star-depth-1 fragment of $PDL(\Delta, A)$ of Palm in [23]. It consists of those formulas whose modal operators do not contain nestings of the iteration operator $(-)^*$. It suffices to admit only those modal operators which can obtained by applying concatenation and union to simple expressions of the form:

- d resp. d^*
- $\psi?$ resp. $(\psi?)^*$
- $d \cdot \psi?$ resp. $(d \cdot \psi?)^*$
- $\psi? \cdot d$ resp. $(\psi? \cdot d)^*$

where $d \in \Delta$ and $\psi \in PDL_1(\Delta, A)$. This fragment can already be translated into the language $FO(\Gamma, A)$ since it contains the transitive structural relations in Δ^+.

In $PDL(\Delta, \{a, b\})$ the one-b example can be defined by the following conjunction:

$$\langle\uparrow^*\rangle(\neg\langle\uparrow\rangle 1 \wedge \langle\downarrow^*\rangle b) \wedge (b \Rightarrow ([\uparrow^+]a \wedge [\downarrow^+]a \wedge [\uparrow^*][\rightarrow^+][\downarrow^*]a \wedge [\uparrow^*][\leftarrow^+][\downarrow^*]a)$$

which is equivalent to a formula in $PDL_1(\Delta, \{a, b\})$. On the other hand the even-depth example can be defined as follows:

$$\neg\langle\downarrow\rangle 1 \;\Rightarrow\; \langle(\uparrow\uparrow)^*\rangle(\neg\langle\uparrow\rangle 1)$$

However, this formula is not equivalent to a formula in $PDL_1(\Delta, \{a, b\})$ since this would imply that the even depth example is first order expressible.

3.3. **Grammars as principle languages.** A phrase structure grammar defines a class of attributed trees with its set of symbols as value set. Thus a grammar G may be viewed as a formula of a somewhat non-standard language for attributed trees. The language is determined by the class of grammars to be considered. The satisfaction relation for a grammar decides whether a given attributed tree is a derivation tree for the grammar.

In particular, the contextfree grammars over A, i.e. grammars in $CFG(A)$, define classes of finite attributed trees with bounded branching and labels in A. Thus $CFG(A)$ may be viewed as the set of formulas of a principle language. Moreover, there is a model preserving translation tr_0 from $CFG(A)$ to the modal logic $PML(\Delta, A)$ and a corresponding translation tr_1 from $CFG(A^+ \cup A)$ to $PML(\Delta^+ \cup \Delta)$ and hence to $PDL_1(\Delta, A)$ The translations are based on the formula below which states that the sequence of values of the children of a given node with value a_0 is the sequence $a_1 \ldots a_k$

$$a_0 \Rightarrow \langle\downarrow\rangle(first \wedge a_1 \wedge \langle\rightarrow\rangle(a_2 \wedge \langle\rightarrow\rangle(\ldots \langle\rightarrow\rangle(a_k \wedge last)\ldots)))$$

where $first := \neg\langle\leftarrow\rangle 1$ and $last := \neg\langle\rightarrow\rangle 1$. It may be noted in passing that the composed translations $st_0 \cdot tr_0$ and $st_1 \cdot tr_1$ yield universal first order formulas whenever there are additional unary predicates for the root and the leaves of a tree in the language.

To describe the binary trees among the one-b trees the symbol a has to be split into two symbols $(a, 0)$ and $(a, 1)$. Here $(a, 1)$ describes those a's which dominate the unique b. The symbols $(a, 1)$ and b are start symbols. Then the finite binary one-b trees are determined by the following set of productions:

$$(a,1) \rightarrow (a,1)(a,0)|(a,0)(a,1)|b(a,0)|(a,0)b$$

$$(a,0) \rightarrow (a,0)(a,0) \qquad b \rightarrow (a,0)(a,0)$$

It can be shown that the extension of the set $\{a, b\}$ cannot be avoided in this example.

More generally, one can consider the class $ECFG(A)$ of *extended contextfree grammars* where regular expressions rather than finite sequences over A are used as the right side of productions (cf. Thatcher [34]). The interpretation of the regular expression in an extended contextfree production yields a derived set of contextfree productions which will be infinite in general. As before the grammars in $ECFG(A)$ define classes of attributed trees with labels in A. Moreover, in this case parse trees with unbounded branching may occur as the one-b-example below will show.

Once again there is a model preserving translation tr_2 from $ECFG(A)$ to $PDL(\Delta, A)$. In this case the translation is based on the formula below which states that the sequence of values of the children of a given node with value a is an instance of the regular expression r:

$$a \Rightarrow \langle\downarrow\rangle(first \wedge \langle\tau(r)\rangle last)$$

The modal operator $\tau(r)$ associated with the regular expression r which tests whether the value sequence of the children of a node is an instance of r is built up inductively from tests $?a$ for the symbols in A

To describe the one-b example without any bound on the branching the symbol a has to be split as above. Then the finite one-b trees are determined by the following set of extended productions:

$$(a,1) \rightarrow (a,0)^*(a,1)(a,0)^*|(a,0)^*b(a,0)^*$$

$$(a,0) \rightarrow (a,0)^* \qquad b \rightarrow (a,0)^*$$

The same splitting can be used to describe the even-depth trees. The symbol $(a,0)$ which is a start - and a terminal symbol describes nodes at an even distance from the root. Then the even depth trees are determined by the following set of productions:

$$(a,0) \rightarrow (a,1)^* \qquad (a,1) \rightarrow (a,0)^*$$

However, it should be noted that the word languages defined by extended contextfree grammars are still contextfree. Thus $CFG(A)$ and $ECFG(A)$ are weakly equivalent but not strongly equivalent.

The principle languages and their translations discussed above are collected in the proposition below:

Proposition 1.

$$\begin{array}{ccccc}
FO(\Delta, A) & \leftarrow & PML(\Delta, A) & \leftarrow & CFG(A) \\
\downarrow & & \downarrow & & \downarrow \\
FO(\Delta^+ \cup \Delta, A) & \leftarrow & PDL_1(\Delta, A) & \leftarrow & CFG(A^+ \cup A) \\
\downarrow & & \downarrow & & \downarrow \\
MTC(\Gamma, A) & \leftarrow & PDL(\Delta, A) & \leftarrow & ECFG(A) \\
\downarrow & & & & \\
MLFP(\Gamma, A) & & & & \\
\downarrow & & & & \\
MSO(\Gamma, A) & & & &
\end{array}$$

Thus all the principle languages considered above are able to describe parse trees of contextfree grammars. Below we shall see that up to a projection the converse is true as well.

4. The expressivity of principle languages

The characterization of the expressivity in the finite of the language $FO(\Gamma, A)$ or any other language discussed above asks for a characterization of those classes of finite attributed trees which are definable by some sentence ψ of the language. Note that we work in finite model theory i.e. we consider only finite models since parse trees are finite structures. In addition, we presuppose the first order axioms of ordered trees i.e. we consider finite models of sentences of the form $\psi \wedge \theta$ where θ axiomatizes ordered trees. Note that there is no first order axiomatization of finite ordered trees (with finite branching). But the first order theory of finite ordered trees (with finite branching) can be characterized by set of first order formulas (cf. [3]). However, already in $MTC(\Delta^+, A)$ there exists a finite axiomatization of the finite ordered trees.

The following method to extend the expressivity of a language over a given value set A will be used several times. It makes uses of the corresponding language over a larger value set $A \times B$. More precisely, we admit classes of trees in $Trees(A)$ which are the projection onto A of a class of trees definable over $A \times B$. Such a class is called definable over A up to a projection. We have noted earlier in 3.1 that the extension from A to $A \times 2$ corresponds to the addition of an existential quantifier for one set variable to the formulas over A. Thus the extension from A to $A \times B$ can be simulated by the addition of an block of length r of existentially quantified set variables whenever $B \subseteq 2^r$.

4.1. **The finite case.** Let A be a finite set. As noted above in proposition 1, we can translate any extended contextfree grammar G into a sentence $\psi(G)$ in $FO(\Gamma, A)$ whose finite tree models are the parse trees of G. However, A.Palm (to appear in the forthcoming dissertation [23]) has shown that up to a projection the converse is true as well. Thus we do not get beyond contextfree languages in this case. Moreover, any implementation of the translation opens the way for the use of parsers for contextfree grammars.

More precisely, he has shown:

Theorem 2. *Let A be a finite value set. Each formula ψ in $Clat(A) = FO(\Gamma, A)$ can be translated into an extended contextfree grammar $G(\psi)$*

over the value set $A \times D$ for some finite set D which up to the projection onto A describes the same class of finite attributed trees. If ψ implies bounded branching then $G(\psi)$ will be contextfree. The translation is obtained as the combination of the following sequence of translations:

1. *Each formula ψ in $FO(\Gamma, A)$ can be translated into a formula $\tau_1(\psi)$ in $PDL_1(\Gamma, A)$ which describes the same class of attributed trees.*
2. *Each formula φ in $PDL(\Gamma, A)$ can be translated into a formula $\tau_2(\varphi)$ in $PML(\Delta, A \times B)$ for some finite set B which up to the projection onto A describes the same class of attributed trees.*
3. *Each formula μ in $PML(\Delta, A \times B)$ can be translated into an extended contexfree grammar $\tau_3(\mu)$ in $ECFG(A \times B \times C)$ for some finite set C which up to the projection onto $A \times B$ describes the same class of attributed trees.*

First of all we have to discuss the context of this result. Actually, under the assumption of bounded branching this characterization result follows from classical results in Thatcher [34] resp. Doner [11] for the stronger logic $MSO(\Gamma, A)$ (cf. Rogers [26], [27]). Their result is proved in a roundabout manner. The first step uses a translation into the (weak) second order theory SkS of the full k-branching tree $\mathcal{N}_k$ with k successor operations where k is the bound on the branching. This step is based on the observation that a k-branching tree and valuations of second order variables can be described in SkS by means of additional unary predicates. The next step makes use of the equivalence of definability in SkS and recognizability i.e. definability by a tree automaton. Finally a recognizable set of trees is up to a projection the set of derivation trees of a contextfree grammar.

Recently at the workshop in Prague Rogers has claimed the full characterization result for $MSO(\Gamma, A)$ for a finite value set A without the assumption of bounded branching (to appear in [28]). In this case he obtains generalized regular contextfree grammars i.e. grammars with a set of contextfree productions generated by a regular grammar. However, it can be verified that these grammars correspond to the extended contextfree grammars of Wright as defined above.

Thus it has turned out that under the finiteness assumption for the value set all the nonlocal principle languages mentioned above are equally expressive up to a projection of the value sets involved (cf. also Kracht [20]). More precisely, the formulas of the strongest language $MSO(\Gamma, A)$ and hence of all the other nonlocal languages can be translated into an extended contextfree grammar over some extended value set $A \times B$ such that both describe the same classes of finite attributed trees. Thus all the translations between nonlocal languages in proposition 1 have an inverse up to a projection of the value sets. We conjecture that in the local case we get a translation into contextfree grammars.

Now it seems that Palm's result for $FO(\Gamma, A)$ has been superseded by Roger's new result for $MSO(\Gamma, A)$. However, Palm has given an explicit construction of the translation map by a sequence of intermediate steps via propositional dynamic logic and propositional modal logic and a node pattern formalism. This approach pays off as follows. On one hand the set B of additional attribute values can be tightly controlled. In addition special results for special classes of formulas can be extracted from the construction. More important, by an analysis of his proof he is able to extend his result to certain infinite value structures A under the locality assumption which requires that the formula implies that the value of a node may only change locally.

It remains an open question for which principle languages beside the first order language the expressivity result can be extended in a similar manner beyond the finite case. Thus it seems to me that our first order principle language strikes a fine balance between expressivity on one hand and the existence of an efficient principle compiler and a feasible translation into formal grammars on the other hand. Another open problem is the determination of the subclass of extended contextfree grammars which is needed for the translation of the first order language in the finite case.

Below we present a short sketch of the stepwise construction of the translation in the proof of theorem 2. Making heavy use of the unique structural relation lemma and the unique value lemma mentioned in 2.2 Palm proceeds by a sort of quantifier elimination. By this method he can translate formulas of $FO(\Delta^+, A)$ into formulas of $PDL_1(\Delta, A))$, the star-depth-1 fragment of $PDL(\Delta, A)$ mentioned in subsection 3.2. It consists of formulas whose modal operators do not contain nestings of the iteration operator $(-)^*$. The conditional modal operators of the form $p\cdot(\varphi?)$ where p is modal operator in $\Delta^+ \cup \Delta$ and φ is a formula, which are the conditional functional uncertainties of our first version of $Clat(A)$, play an important role in this context.

Adding new attribute values by splitting the given ones he can eliminate the iteration operator $(-)^*$ for formulas in $PDL_1(\Delta, A)$ and hence afterwards the tests. Thus he arrives at the multi-modal logic $PML(\Delta, A \times B_1)$. Translating the full logic $PDL(\Delta, A)$ yields a larger set $A \times B$ of attribute values. Now the modal formulas are transformed into node pattern grammars. Node patterns are special typed feature structure in the sense of Carpenter [8] which can be used to specify the local neighbourhood of a node.

The set of node patterns is responsible for another splitting of the set of attribute values. Making use of this splitting the node pattern grammar in turn is translated into the associated finite *regular rational attributed tree* which is just a graphical representation of an extended contextfree grammar. This finishes the sequence of translations. The final very useful finite representation of the given formula is universal in the sense that all the possible finite tree models can be read off from it. In the case of bounded branching the regular rational attributed tree can be replaced by an rational and-or-attributed tree.

4.2. **The stack case.** The above expressivity results rely on the finiteness of the value structure. However, certain linguistic phenomena like indexation seem to require the use of infinite value structures. In this situation the problem of characterizing the expressivity of the principle languages discussed above is wide open. If the value structure to be used is too strong the definable classes of attributed trees will yield all recursively enumerable word languages. The linguists will prefer those value structures where the definable classes of attributed trees will yield mildly contextfree languages.

For our first order principle language Palm has obtained a first expressivity result for an infinite value structure which gets us beyond the class of contextfree languages. It yields the class of linear index grammars and other related classes. Making use of the detailed description of the translation in the finite case he has been able to extend his result to the case where the value structure is a product of a finite set A and a stack structure $Stacks(I) = (I^*, (push_i : i \in I), pop, nop, empty)$ with elements from a finite index set I. Note that $Stacks(I)$ is infinite but finitely generated.

His translation result is restricted to formulas in $FO(\Gamma, A \times Stacks(I))$ which satisfy a certain locality condition. This locality condition requires that only for pairs of nodes belonging to the same neighbourhood equations between their stack values may be introduced. The translation yields a grammar in a class of generalized index grammars determined by the locality notion which is used. These grammars are generalizations of contextfree grammars which make use of an index stack. In the contextfree rules some symbols may be adorned with a stack operation and in the derivation tree some nodes carry an index stack. Special cases are the index grammars of Aho [2] and the linear index grammars of Gazdar [18]. The latter formalism constitutes one of several equivalent grammar formalisms for the description of the so called mildly contextsensitive languages (cf. [37]).

A formula $\varrho(x, y)$ in $FO(\Gamma, A)$ is called a *coreference formula* over A if it defines a partial function on trees in $Trees(A)$ or it is the converse of such a formula. Such a coreference formula defines a notion of locality between a pair of nodes satisfying it. As usual the first argument is called the antecedent of the pair and the second one is called the succedent. In general, one gets coreference chains as the succedent of a pair may be an antecedent of another pair. The movements in GB provide examples of such coreference formulas and their chains. The two examples to be used below have distance 1 i.e. the antecedent and the succedent of a coreference pair must have the distance 1. The *linear coreference* is defined by the immediate projection relation $\downarrow (x, y)$ in trees with projection lines whereas the relation $\uparrow (x, y)$ in arbitrary trees yields the *uniform coreference*.

For a given coreference formula $\varrho(x, y)$ in $FO(\Gamma, A)$ we shall consider the ϱ-local trees in $Trees(A \times Stacks(I))$ i.e. trees where the stack value must be empty for all non antecedent nodes. Thus in the linear case the stack is empty at the beginning of the projection line and is inherited by the unique successor along the projection line. Whereas in the uniform case the stack is inherited by all successors.

Now a formula ψ in $FO(A \times Stacks(I))$ is called *locally stack constrained* w.r.t. ϱ if ψ implies ϱ-locality and equations between the stack values of nodes may appear only in the form $\varrho(x, y) \wedge |y|_2 = f(|x|_2)$ where f is a stack operation and $| - |_2$ yields the stack value. In the linear case we obtain the linear index trees i.e. trees with projection lines which carry a stack where at the beginnning of a projection line the stack must be empty and applying the immediate projection the stack may be changed by a stack operation. In the uniform case all nodes carry a stack and the stack is changed uniformly from a node to its immediate successsors.

Having a coreference formula $\varrho(x, y)$ over A with distance 1 we can define the class $IG_\varrho(A)$ of *ϱ-index grammars* as follows. They consist of contextfree rules over A where the succedent position w.r.t. ϱ in the rule is adorned by a stack operation from $Stacks(I)$. The derivation trees of such a grammar carry stacks along the ϱ-chains which must respect the stack operation prescribed by the rule which was applied. Clearly any such grammar can be translated into a formula in $FO(\Gamma, A \times Stacks(I))$ which describes the derivation trees. The following result shows that the converse is true up to a projection of the value sets.

Theorem 3. *Let A a finite set and $Stacks(I)$ be the finitely generated stack structure over a finite index set I and let $\varrho(x, y)$ be a coreference formula over A with distance 1. Then each formula ψ in $FO(\Gamma, A \times Stacks(I))$ which is locally constrained w.r.t. ϱ can be translated into an extended ϱ-index grammar over $(A \times B)$ with index set I which up to the projection onto A describes the same class of finite indexed attributed trees.*

The key idea of the proof is to view the ϱ-local trees in $Trees(B \times Stacks(I))$ as trees in $Trees(B \times stacks(I))$ where $stacks(I)$ is the finite set of stack operations $\{push_i : i \in I\} \cup \{pop, nop, empty\}$. This is achieved by attaching the element f of $stacks(I)$ appearing in the equation $|y|_2 = f(|x|_2)$ at the succedent node of the coreference pair. As $B \times stacks(I)$ is finite we can apply the result of the finite case. However, a postprocessing is needed since not all trees in $Trees(B \times stacks(I))$ correspond to ϱ-local trees in $Trees(B \times Stacks(I))$ as negative stacks may appear along a ϱ-path.

5. Principle-based parsing

The translations discussed in the previous section could be used to transform a given set of principles into a contextfree grammar if the set of attribute values happens to be finite and bounded branching is assumed. Then any parser for the contextfree grammar will yield a parser for the set of principles. However, these parsers are not principle-based in the sense discussed in 1.3. As we have indicated there we are looking for principle-based parsers i.e. parsers which respect the universality and the modularity of of the grammar formalism. Such a parser must be derivable from a parser generator for the UG which preserves the module structure present in the UG.

The parser generator or parsing scheme for the UG produces a parser for a given language X by a specialization w.r.t. the parameter values in its set of principles XG. The information derived from XG which the parser generator must access in this case will be called the *internal grammar* $I(XG)$ of X. Clearly, the parsing operations and the parsing strategy should be universal i.e. definable at the level of the UG. In addition, the parsing strategy should preserve the module structure present in the UG. Then changing the grammar by disabling a module of principles will produce a parser disregarding this module. However, in the concrete parser for X the applicability of a parsing operation in a certain situation is controlled by the input sentence and the information in $I(XG)$.

5.1. **The internal grammar.** Our decision to use a dedicated principle language consisting of formulas describing attributed trees motivates the following approach first used by Schneider in [30]. It may be summarized as follows:

1. The parsing operation to be used is determined by the syntactical form of the principle the parser tries to satisfy. Thus parsing operations correspond to syntactically defined classes of principles.
2. The internal grammar refers to attribute values but it does not refer to nodes. Thus it consists of a set of nodefree relations on attribute values. The relations are indexed by the principle class to which they belong.
3. If a first order language is considered the principle classes can determined by the quantifier prefix and the structural part of the matrix of a prenex normal form of the principle .
4. To obtain the internal grammar one has to separate the structural constraints of the matrix i.e. the conditions on the node configurations from the value constraints i.e. the conditions on attribute values.

In order to separate the structural - from the value information we derive a normal form for first order principles. Let $\varphi = Q_1 x_1 \dots Q_k x_k \psi(\bar{x})$ be a first order principle in prenex normalform, where ψ is quantifier free matrix and $\bar{x}$ denotes the sequence $x_1, \dots, x_k$. Using the conjunctive normalform of $\psi(\bar{x})$ we can separate the structural part from the attribute value part and obtain $\bigwedge_{i=1}^{m}(\delta_i(\bar{x}) \vee \beta_i(\bar{x}))$ where δ_i resp. β_i are structural - resp. value clauses i.e. disjunctions of negated or unnegated atomic constraints. Using $\bar{\delta}_i = \neg\delta_i$ the principle can be rewritten in the form:

$$\bar{Q}\bar{x}(\bigwedge_{i=1}^{m}(\bar{\delta}_i(\bar{x}) \Rightarrow \beta_i(\bar{x}))) \tag{1}$$

Thus the condition on the structural configuration of the k nodes in the structural part may be viewed as a restricted quantification applied to the value part. Experience has shown that for concrete principles in GB the syntactical complexity is quite low. In many cases quantifier prefix $\bar{Q}$ has length at most 2 i.e. the principle refers to at most 2 nodes.

Making use of the fact that we are dealing with finite trees we can apply the unique structurual relation property mentioned in 2.2. Eliminating the negated structural atomic constraints in each structural constraint $\bar{\delta}_i(\bar{x})$ we obtain a form of the principle in 1 where the structural premisses in the implications are negationfree:

$$\bar{Q}\bar{x}(\bigwedge_{i=1}^{m}(\varrho_i(\bar{x}) \Rightarrow \beta_i(\bar{x}))) \tag{2}$$

Here $\varrho_i(\bar{x})$ is of the form $\bigvee_{l=1}^{r_i} \varrho_{i,l}$ where each $\varrho_{i,l}$ is a conjunction of atomic structural constraints. At the cost of enlarging the conjunction in (2) we may get rid of the disjunctions in ϱ_i by moving them out of the implication as conjunctions.

Clearly, each value constraint $\beta_i(\bar{x})$ involving the nodes $x_1, \ldots, x_k$ determines a k-ary relation B_i on A. It describes the set of admissible values if the structural constraint $\varrho_i(\bar{x})$ is assumed. Now the contribution of the principle φ of shape $(\bar{Q}, \varrho_1, \ldots, \varrho_m)$ to the internal grammar consists of the sequence of relations $(R_i^{\varphi}(\bar{Q}, \varrho_i) : 1 = i, \ldots, m)$ which are defined by the formulas $\beta_i(\bar{x})$ i.e. the nodefree relations B_i on A^k. Given a grammar (or a module of the grammar) i.e. a finite set of principles one wants to collect all the relations of a particular shape into one relation in the internal grammar. Since the prenex normalform of a conjunction of formulas in prenex normalform requires a renaming of existential variables of the conjuncts it is easier to compute the prenex normalform of the conjunction of the principles considered.

The information on the set of possible values of a sequence of nodes $x_1, \ldots, x_k$ considered by the parser is again a k-ary relation B on A. This suggests to use unification in an extended set of values to take care of the value constraints which must be satisfied by a sequence of nodes. In this situation it is helpful to view A as well as A^k as a flat type structure where the elements are pairwise incompatible and there is a smallest element $\perp$ which represents inconsistency i.e. conflicting information and a largest element $\top$ which represents no information. Introducing for each $\bar{a}$ in A^k a new formal value $\neg\bar{a}$ we obtain an ordered type structure $\widehat{A^k}$ for the representation of partial information on the values in A^k, where smaller elements represent more information.

Using this approach the value constraint $\beta_i(\bar{x})$ is transformed as follows. With $|\bar{x}| = \bar{a}$ as an abbreviation of $\bigwedge_{j=1}^{k} |x_j| = a_i$ the value constraint $\beta_i(\bar{x})$ can be represented by the disjunction $\bigvee_{\bar{a} \in B_i} |\bar{x}| = \bar{a}$. By the finiteness assumption for the value set A we know that A^k and hence B_i are finite. Hence $\beta_i(\bar{x})$ can also be represented by the conjunction $\bigwedge_{\bar{a} \notin B_i} |\bar{x}| \neq \bar{a}$. By this method $\beta_i(\bar{x})$ can be represented as a conjunction $\gamma_i(\bar{x})$ of atomic value constraints which may contain the new formal values. This yields the following normalform of the principle of shape

$(\bar{Q}, \varrho_1, \dots, \varrho_m)$

(3)
$$\bar{Q}\bar{x}(\bigwedge_{i=1}^{m}(\varrho_i(\bar{x}) \Rightarrow \gamma_i(\bar{x})))$$

where $\rho_i(\bar{x})$ is a negationfree structural constraint and $\gamma_i(\bar{x})$ is conjunction of atomic value constraints. To get a more compact representation one considers the type structure $\tilde{A}$. It is an extension of A obtained by introducing negations of values in A and special values for value equations and inequations as in Schneider [30]. With the help of the type structure $\tilde{A}$ the value of a node can be computed incrementally. Starting from the empty information *top* one can take care of a value constraintby unifying it with the actual value.

5.2. **Parsing in the strongly local case.** Having described the internal grammar we have to deal with the parsing operations. During the parsing process a parser must deal with incomplete parse trees, the objects to which the parsing operations are applied. Since the parsing process need not be deterministic it has to handle finite sets of of incomplete parse trees which represent the set of options to be considered. They can be described as disjunctions of conjunctions of atomic constraints i.e. finite sets of constraints. The structural part of such a constraint determines a quasi tree in the sense Rogers and Vijay-Shanker (cf. [29],[36]).

For the general case we have nothing to offer if we stick to our requirements for a principle-based parser. However, in the strongly local case the situation is much better. A principle φ is called strongly local if in its normalform $\bar{Q}\bar{x}(\bigwedge_{i=1}^{m}(\varrho_i(\bar{x}) \Rightarrow \beta_i(\bar{x})))$ in the sense of (2) the positive structural constraints $\varrho_i(\bar{x})$ contain only the immediate structural relations in Δ. Since any set of positive local structural constraints on a sequence of node may be interpreted as a set of trees the parser can work with a set of trees in this situation.

As outlined above, we consider parser operations for the universal grammar which do not presuppose information from the internal grammar. Thus they are general operations operating on attributed trees. But the applicability of a parser operation in a particular situation is controlled by the input sentence and the internal grammar.

In the strongly local case the state of the parsing process is given by a stack of trees and the part of the input sentence still to be read. Here we shall discuss a set up with three basic parser operations read, insert and expand which use at most two trees at the top of the stack. The question how many trees below the top tree can take part in decisions of the parser is a linguistical question which needs to be decided on linguistical evidence.

The *read operation* creates a one-node tree from the next symbol (=word) of the input sentence and pushes this tree on top of the stack. The *insert operation* is a structure integrating operation which takes care of the restriction principles. It integrates the top tree of the stack into the second top-most tree of the stack and, after removing both trees from the stack, places the new tree on the top of the stack. Thus $Ins(t_2, t_1, u, r))$ inserts the tree t_1 into the tree t_2 at the position u in t_2 in the direction r. Its applicability is controlled by the internal grammar by means of the restriction grammar relation for r.

The *expand operation* is a structure generating operation which takes care of the generation principles. It creates a new node and adds it to the top node of the stack. Thus $expand(t_1, x, u, r)$ expands the tree t_1 at the node u by a new node with label x in the direction r. Again its applicability is controlled by the internal grammar by means of the licensing grammar relation for r.

In our approach the state of parsing process is described by *configurations* which are pairs of the form (s, w) where s is a stack of trees and w is the part of the input still to be read. The parser proceeds by transitions between the configurations. A *transition* from the configuration (s, w) to the configuration (s', w') is a quadruple of the form (s, x, op, s'), where $x \in \Sigma \cup \{\epsilon\}$ is an input symbol and $op \in \{read, insert, expand\}$ is a parser operation such that s' is the result of applying op to s and $w = w'x$. Clearly, we must have $x = \epsilon$ whenever $op \neq read$.

A *computation* for the input sentence w is a sequence of configurations which starts with the initial configuration (s_0, w), where s_0 is the empty stack, and successor configurations are obtained by transitions. A computation *sucessful* if the last configuration is a stop configuration with a stack consisting of one tree, a *parse tree* of w, the result of the computation.

To obtain a concrete parser generator we have to fix a parsing strategy. A *parsing strategy* regulates the choice of transitions which can be used by the parser in a particular configuration. If we adhere to our general concept it follows that the parsing strategy works at the level of the universal grammar. Therefore the definition of a strategy should not depend on the particular internal grammar with which it is used. However, the strategy may require the test of the applicability of a certain parser operation. By this route the internal grammar interacts with the strategy.

Using this approach we obtain a parsing schema which yields a parser generator after fixing a parsing strategy. Moreover, in our approach we can address the question of *correctness* and *completeness* of a given parser with respect to a parsing strategy. A computation is called admissible for an internal grammar if all transitions of the computation are admissible i.e. the parser operation used satisfies the applicability constraints of the internal grammar. Thus we can study the question whether an admissible computation yields only such parse trees which satisfy the given set of principles and whether the existing parse trees can be obtained.

Using the approach described so far Schneider has implemented a parser for an interesting fragment of the German grammar (cf. [30]). Obviously, he resorted to ad-hoc solutions in dealing with some nonlocal principles. The parser turned out to be quite efficient although no special tuning for efficiency was used. In addition, he has obtained a correctness result (cf. [32]). Restricting the principles to a special class of strictly local $\forall\exists$-formulas satisfying a set of consistency properties he has proved a correctness result which is indepedent of the parsing strategy considered.

References

[1] S. Abney and J. Cole. A government-binding parser. In *Proceedings of NELS 15*, 1985.

[2] A. Aho. Indexed grammars - an extension of contextfree grammars. *Journal of the Association for Computing Machinery*, 15:647–671, 1968.

[3] R. Backofen, J. Rogers, and K. Vijay-Shanker. A first-order axiomization of the theory of finite trees. *Journal of Logic, Language and Information*, 4:5–39, 1995.

[4] G. Barton, Jr. *Toward a Principle-based Parser*. MIT Press, Cambridge, Mass., 1984.

[5] P. Blackburn, W. Meyer-Viol, and M. de Rijke. A proof system for finite trees. *x*, z, 199y.

[6] P. Blackburn and E. Spaan. A modal perspective on the computational complexity of attribute value grammar. *Journal of Logic, Language and Information*, 2:129–169, 1993.

[7] J. Büchi. Weak second-order arithmetic and finite automata. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik*, 6:66–92, 1960.

[8] B. Carpenter. *The Logic of Typed Feature Structures; with Applications to Unification Grammars and Constrained Resolution*, volume 32 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1992.

[9] N. Chomsky. *Lectures on government and binding*. Foris, Dordrecht, 1981.

[10] N. Chomsky. *Knowledge of language*. Praeger, New York, 1986.

[11] J. Doner. Tree acceptors and some of their applications. *Journal of Computer and System Sciences*, 4:406–451, 1970.

[12] H.-D. Ebbinghaus and J. Flum. *Finite Model Theory*. Springer, Berlin, 1995.

[13] G. Fanselow and S. Felix. *Sprachtheorie, eine Einführung in die generative Grammatik, Band 1: Grundlagen und Zielsetzungen.* Francke, Tübingen, 1987.

[14] M. Feldmeyer. Clat: Eine Sprache zur Beschreibung von attribuierten Bäumen durch globale Constraints, 1991.

[15] S. Felix. The structure of functional categories. *Linguistische Berichte*, 125:46–71, 1990.

[16] S. Fong. The computational implementation of principle-based parsers. In S. Abney and C. Tenney, editors, *Principle-based parsing: Computation and psycholinguistics.* Kluwer, Dordrecht, 1991.

[17] S. Fong. *Computational properties of principle-based parsers.* PhD thesis, MIT, Dept. of Electrical Engineering and Computer Science, 1991.

[18] G. Gazdar. Applicability of indexed grammars to natural language, 1985.

[19] G. Gazdar, E. Klein, G. Pullum, and I. Sag. *Generalized Phrase Structure Grammar.* Blackwell, Oxford, 1985.

[20] M. Kracht. Mathematical aspects of command relations. In *Proceedings of 6th Conference of the EACL*, pages 40–249, 1993.

[21] M. Kracht. Syntactic codes and grammar refinement. *Journal of Logic, Language and Information*, 41–60:5–39, 1995.

[22] A. Palm. Erweiterung der Sprache CLAT für attribuierte Bäume, 1992.

[23] A. Palm. *The expressivity of tree languages.* PhD thesis, Universität Passau, Fakultät für Mathematik und Informatik, 1997. to appear.

[24] C. Pollard and I. Sag. *Head-Driven Phrase Structure Grammar.* Chicago University Press, Chicago, 1994.

[25] M. Rabin. Decidability of second-order theories and automata on infinite trees. *Transactions of the American Mathematical Society*, 141:1–35, 1969.

[26] J. Rogers. *Studies in the logic of trees with applications to grammar formalisms.* PhD thesis, University of Delaware, 1994.

[27] J. Rogers. On descriptive complexity, language complexity, and gb. In P. Blackburn and M. de Rijke, editors, *Specifying Syntactic Structures.* CSLI Publications, 1996.

[28] J. Rogers. Topics in model-theoretic syntax. In *Arbeitspapiere des Sonderforschungsbereichs 340*, 1997. Bericht 80-85.

[29] J. Rogers and K. Vijay-Shanker. Reasoning with description of trees. In *Proceedings of 25th Annual meeting of the ACL*, pages 2–80, 1992.

[30] K.-M. Schneider. Entwurf und Implementierung eines GB-Parsers für ein Fragment des deutschen, 1993.

[31] K.-M. Schneider. An attribute-value vector formalism in a principle-based parser, 1995.

[32] K.-M. Schneider. Description of a parser model in a many-sorted first-order language, 1995.

[33] S. Shieber. *An Introduction to Unification-based Approaches to Grammar*, volume 4 of *CSLI Lecture Notes.* Center for the Study of Language and Information, Stanford, 1986.

[34] J. Thatcher. Characterizing derivation trees of contextfree grammars through a generalization of finite automata theory. *Journal of Computer and System Sciences*, 1:317–322, 1967.

[35] J. Thatcher and J. Wright. Generalized finite automata theory with an application to a decision problem of second order logic. *Mathematical Systems Theory*, 2:57–81, 1968.

[36] K. Vijay-Shanker. Using descriptions of trees in a tree adjoining grammar. *Journal of Computational Linguistics*, 18:481–517, 1992.

[37] K. Vijay-Shanker and D. Weir. The equivalence of four extensions of context-free grammars. *mathematical Systems Theory*, 27:511–546, 1994.

Universität Passau, D 94030 Passau, Tel.49-851-5093132
E-mail address: volger@byrd.fmi.uni-passau.de