GRAALAN – Grammar Abstract Language Basics

Stefan Diaconescu[1]

[1] SOFTWIN SRL, Str. Fabrica de glucoza, Nr. 5, Sect. 2, Postal Code 020331, Bucharest, Romania
sdiaconescu@softwin.ro

Abstract. This paper gives an outline about most important features of GRAALAN (Grammar Abstract Language) used for linguistic knowledge description. GRAALAN is an implementation of theoretical concepts of GDGF (Generative Dependency Grammars with Features) and AVT (Attribute Value Trees). GDGF is based on dependency trees (DT) and a generative process. GDG eliminates some issues of Dependency Grammars DG (for example the missing of phrasal categories) and Generative Grammars GG (the problem of discontinuous structures) and merges the advantages of the two types of grammar (GG - the representation of phrasal categories; GDG - the handling of discontinuous structures as gaps and non projective constructions). GRAALAN can be used in the morphological description, morphological analysis, text annotation, grammar analysis, generation and even in the automatic translation process.

1 Introduction

A lot of language models and language grammar types were proposed trying to solve the natural language problem [1]. There are three of the linguistic models that seem to be more successful and used in some applications [2]: TAG – Tree Adjoining Grammar [3], HPSG – Head-Driven Phrase Structure Grammar [4] and LFG – Lexical Functional Grammar [5].

During the last years another idea was more and more analyzed and studied: the dependency. Actually, it is a quite old idea (usually [6] is used as reference but the dependency idea in the grammar is millennial), but new valences and strength became attractive. The new type of Generative Dependency Grammars (GDG) [7] try to move farther these researches, making a connection between two directions that seems to be almost irreconcilable till now: the generative approach and the dependency approach. GDG is a sort of combination between Dependency Grammar (DG) [6], [11], [12], [13] and Generative Grammar (GG) [14]. The association of AVT (Attribute Value Trees) to the elements of the GDG (non-terminals, terminals, pseudoterminals and procedural actions) produces the GDG with Features [7]. This new research direction will have as final target to find a more adequate language model that could be used in natural language processing and that have the potential to produce many and better applications. GDG investigations have already some results that seem to be promising and deserve to be continued and developed. GRAALAN (Grammar Abstract Language) is a language that implements the principles of GDG [7], [8], [9].

2 GRAALAN features

GRAALAN is a declarative language used to describe linguistic knowledge and is based on GDG and AVT. GRAALAN is a sequence of sections. Each section describes a certain linguistic chapter: a lexicon, the morphological structure of a language (the “Morphological Configurator”), the grammar of a language, the bilingual dictionaries, etc.

We will name *programme* a description of linguistic information realised in GRAALAN.

The global structure of a GRAALAN program is:

```
1. {GRAALAN programme} => {section}{GRAALAN programme}| {section}
2. {section} =>   {morphological configurator section}|
                  {grammar description section}|
                  {multiword expression dictionary section}|
                  {bilingual syntactic dictionary section}|
                  {bilingual morphological dictionary section}|
                  …etc
```

Not all the sections must be written directly. Some of them can be generated using some specific tools.

We will present in the next sections only some elements concerning the morphological description [8], the grammar description [7] and Multi Word Expressions (MWE) dictionary description [9].

3 GRAALAN morphology description

3.1 Informal definitions

We will use a formalism that is of type FS (feature structures) in the description of the morphological knowledge and grammar knowledge. This formalism considers that a word has associated a structure formed by some attributes and each attribute has different values. FS types are used for example in the PATR grammars or in HPSG. In fact PATR or HPSG use AVM – Attribute Value Matrix. Instead of AVM GRAALAN define an AVT – Attribute Value Tree.

We will consider here that each word belong to a lexical class. There are different lexical classes accepted in different classical grammars for different languages. For example, there are usually accepted 10 lexical classes in Romanian grammars: noun, article, pronoun, adjective, numeral, verb, adverb, preposition, conjunction, interjection. Each class can have many lexical categories. Each lexical category can have many values. We will name classes and categories as “classes” or “attributes”.

We can associate an AVT not only to a word (or “terminal”) but also to some syntactical construction. In this case we will define many types of categories that will be considered “syntactical categories”. Therefore we will give to the notion of category a very general sense: it can be a lexical class, a lexical category or a syntactical category. Such a category will be represented in an AVT by an “attribute”.

We will consider also that the attributes can be indexed or not indexed. If an attribute is indexed, than in a certain context (for example in a grammar description rule) this attribute must take the same value for each apparition. If an attribute is not indexed, then in the same context this attribute can take any value from his value set.

A very simplified AVT can be described as follows:

```
1. <AVT> -> {<label>:<not labeled AVT>}|{<label>}|<not labeled AVT>
2. <not labeled AVT> -> < indexed AVT >|< non indexed AVT >
3. <indexed AVT> -> [<AVT content>]
4. <non indexed AVT> -> (<AVT content>)
5. <AVT content> -> <label>:<feature content>|<feature content>|<label>
6. <feature content> -> <attribute> = <attribute value list>
7. <attribute value list> -> <attribute value element>, <attribute value list>|
<attribute value element>
8. <attribute value element> -> <attribute value><AVT>|<attribute value>
```

The rule 1 expresses the fact that an AVT can have a label – the AVT label definition. The AVT label can be used later alone in the same context and this mean that the label must be substituted with its definition.

The rule 5 expresses the fact that an attribute with his list of values can have a label – the attribute list label definition. This label can be used later alone in the same context and this means that the label must be substituted with its definition. In this case the label is also a marker for the attribute so two attributes with different labels (markers) must be considered as different attributes. Usually a context is formed by the left hand of a rule and one alternant from the right hand of the rule in a grammar description. The labels defined here are a sort of generalization of reentrancy from HPSG.

The rule 8 defines a point of branching of the tree. We can see that such a branching appears after an attribute value.

In fact, in GRAALAN each attribute (that corresponds to a category) and each value has associated other information too. The category can for example have associated: the name of the category, the abbreviation of the category name, the fact if the category can be inflected or not. For example the category "Pronoun Type" has in Romanian 9 values but it can not be inflected and the category "Pronoun number" can take two values (singular, plural) but it can be inflected. The category value can have associated for example: the name of the category value, the abbreviation of the category value name, the indication if the corresponding value is associated to the entry in the lexicon (for example the value Nominative of the category Case will be associated to a noun in the lexicon but the category Genitive will be not).

3.2 Morphology description creating and using

The morphological knowledge is created in two steps: the Morphological Configurator creating and the morphological description creating.

a) The Morphological Configurator can be created by a linguist in two ways: using a usual text editor or a special text editor (that offers some features like syntax highlighting). The Morphological Configurator is then compiled with a specific compiler and some errors can be detected and corrected. The final result is stored in a LKB (Linguistic Knowledge Base).

b) The Morphological Description of the words of the language is created by the linguist with a special tool that uses the Morphological Configurator. The resulting information (lexicon, inflection rules, and root dictionary) is stored in the LKB. From LKB the information is taken to realize different type of applications: spellers, grammar checkers, inflection applications, lemmatizers, translation applications.

Example of morphological description:

We will present a fragment of a Morphological Configurator for the Romanian Language written as a big AVT in GRAALAN:

```
//******************************************************************
//
Section Morphological Configurator
Source RO
Exploitation RO;

//*******************************************************************
//                              Clasa SUBSTANTIV
//*******************************************************************
[clasa /name = Clasa, abbreviation = Cls, inflection = no/
   = substantiv /name  = Substantiv, abbreviation = Subst,
                          lemma = yes, lexicon = entry/
   [rol /name = Rol, abbreviation = Rol, inflection = yes/
            = substantiv /name = Substantiv,
                          abbreviation = Subst, lemma = yes,
                          lexicon = entry /
                 [tip substantiv /name = TipSubstantiv,
                          abbreviation = TipSubst,
                          inflection = no/
                   = propriu /name = Propriu, abbreviation = Pr,
                          lemma = yes, lexicon = entry/
                   , comun /name = Comun,
                          abbreviation = Com,
                          lemma = yes,
                          lexicon = entry/
                 ]
                 [GEN_NEFLEXIONAT: gen /name = Gen,
                          abbreviation = Gen, inflection = no/
                   = masculin /name = Masculin,
                          abbreviation = Masc, lemma = yes,
                          lexicon      = entry/
                   , feminin /name = Feminin, abbreviation= Fem,
                          lemma = yes, lexicon      = entry/
```

```
                  , neutru / name = Neutru, abbreviation = Neu,
                         lemma = yes, lexicon      = entry/
                ]
           ................................
   ]

,articol ...
,pronume ...
,adjectiv ...
,numeral ...
,verb ...
,adverb ...
,prepozitie ...
,conjunctie ...
,interjectie ...
]
```

4 GRAALAN grammar description

4.1 Informal definition

The formalism of grammar description in GRAALAN allows the treatment of the following elements:

a) *The syntax.* By "syntax" we will understand here the way of word combining in order to build sentences, phrases.

b) *The dependencies.* We will accept here that the aspects concerning the dependencies can be reduced to the way different part of speech that have some roles in the sentences, phrases interact by some coordinate or subordinate relations.

c) *The agreement.* By agreement we will understand the way different parts of speech "match" one another (by dependency relations) from point of view of different lexical categories: gender, case, persons, etc.

d) *The errors.* The grammar description must allow indicating some errors (at least the more frequent ones) that are fond in the statements. The bad built sentences must therefore recognize (more or less) but they must be marked as being incorrect.

e) *Reversibility.* we understand by grammar reversibility the property of the grammar to allow the transformation of a source text (named surface structure) into a deep structure (dependency tree in our case) and also the transformation of this tree into a surface structure.

We will give here an informal definition of GRAALAN Grammar description. A more formal definition of GDG on which is based GRAALAN grammar description is given in [7].

A GRAALAN grammar description is a sequence of labeled rules.

A rule has two parts: the left part and the right part.

The left part of a rule contains a non terminal and an AVT. The AVT contains syntactic/lexical categories with their values.

The right part of a rule contains one or many Alternantes.

An alternant is formed by a set of sections: the syntactic section, the dependency section and the agreement section.

a) *The syntactic section* is a sequence of (eventually labeled) NTPA where the four letters N, T, P, A have the following meanings:

N means *non-terminals* i.e. the syntactic categories that can be described having a name and a structure. A non-terminal must be described in a grammar rule (must appear one time in the left part of a rule). The name of a non-terminal will be written between <...>.

T means *terminals* i.e. a set of the words that can be found in the lexicon or can be obtained by applying some flexional rules on words from the lexicon. The terminal will be written between "...".

P means *pseudo-terminals* i.e. non-terminals that contain only terminals. When we will describe a dependency tree or a grammar we will not cover all the words from the lexicon because in this case the number of rules from the grammar can be too big. So we can say that some non-terminals that we name pseudo-terminals (for example some nouns or some verbs) will never be described in the grammar (we have not rule with these non terminals in the left part). The name of a pseudo-terminal will be written between %...%.

A means *procedural actions* i.e. the set of the routines that can be used to represent a certain portion of the text that we analyze. For example a number represented like a sequence of digits or a mathematical formula or even an image with a certain significance that appear in a text can be "replaced" in grammars or dependency trees by a certain procedural action. The name of a procedural action will be written between #...#.

Each NTPA can have associated information about how this NTPA is linked with others NTPA by certain relations from the dependency section (these relations are indicated by their labels in the dependency section). There are three positions concerning the relations: Coordinated (CR), Subordinated (SR) and Governor (RR) (i.e. the other NTPA is subordinated to the current NTPA).

Each NTPA can have associated an AVT describing syntactic/lexical categories with their values.

b) *The dependency section* contains the description of the relations between the NTPA from the syntactic section (referred by their labels). There are two types of relations:

The subordinate relation SR is a relation between two N, T, P, A, or a coordinate relation CR, respecting some rules. One of the two elements is considered to be subordinated to the other element by the relation SR. The name of an SR will be written between @...@. Each SR can have associated information about how this SR is linked with others NTPA or relation by certain relations from the current dependency section (these relations are indicated by their labels in the current dependency section). There are two positions concerning the relations of the subordinate relation: coordinated subordinated (SR) and governor (RR).

The coordinate relation CR is a relation between (usually) two N, T, P, A, or SR that are said to be coordinated by CR respecting some rules. These two elements are named "fixed entries". A CR can group also others "supplementary entries". The name of a CR will be also written between @...@. Each CR can have associated information about how this CR is linked with others NTPA or relations by certain relations from the current dependency section (these relations are indicated by their labels in the current dependency section). There are three positions concerning the relations: coordinated (CR), subordinated (SR) and governor (RR).

c) *The agreement section* contains a list of agreement rules. An agreement rule is an expression of type “if (conditional expression) then (actions)” working in a tetravalent logic (with the logical values: true, false, not applicable, and undefined) [10]. The conditional expression is a logical expression expressed between the categories values of the NTPA from the syntactic section. The actions indicate some error messages or how the analyze will be continued after an agreement error.

4.2 Grammar description creating and using

The grammar description can be created by a linguist in two ways using a usual text editor or a special text editor (that offers some features like syntax highlighting). The grammar description is then compiled with a specific compiler and some errors can be detected and corrected. The final result is stored in LKB (Linguistic Knowledge Base).

From LKB the information is taken to realize different type of applications: grammar checkers, translation applications.

Example of grammar description:

We will give in the following the description of a simplified toy grammar that can generate amongst others the phrase:

"Profesorul si discipolul au venit si au vazut ca pastorii si strainii se bucura si dantuie." (The teacher and the disciple came and saw that the shepherds and the strangers enjoy and dance.)

The GRAALAN description in a simplified form will be:

```
RuleR1:
<fraza>::=
   Alternant A1:
      Syntax Label1: <grup subiectiv>
                       (gen = masculin, feminin, neutru)
                       (numar = singular, plural)
                       (persoana = I, II, III)
                    Governor Label3
             Label2: <grup predicativ>
                       (gen = masculin, feminin, neutru)
                       (persoana = I, II, III)
                       (numar = singular, plural)
                    Subordinate Label3
      Dependencies Label3: @relatie subiect/predicat@(1)
      Ageement
         if
         (
             Labe2  [gen = masculin, feminin, neutru]
                    [numar = singular, plural]
                    [persoana = I, II, III]
                           <-
```

```
              Label1 [gen = masculin, feminin, neutru]
                     [numar = singular, plural]
                     [persoana = I, II, III]
         )
         true ( OK )
         else ( Message “Agreement error”…
         )
RuleR2:
<grup subiectiv>::=
   Alternant A1:
      Syntax Label1: <subiect>
                     Coordinate Label3(1) ","!
             Label2: <grup subiectiv>
                     Coordinate Label3(2)
      Dependencies Label3:@rel. de coord. prin virgula@(2)
   Alternant A2:
      Syntax Label1: <subiect>
                     Coordinate Label3(1) "si"!
             Label2: <grup subiectiv>
                     Coordinate Label3(2)
      Dependencies Label3:@rel. de coord. prin conjunctie@(2)
   Alternant A3:
      Syntax <subiect>
RuleR3: <subiect>::=
   Alternant A1:
      Syntax <substantiv>
RuleR4: <substantiv>::=
   Alternant A1: Syntax "profesorul"
   Alternant A2: Syntax "discipolul"
   Alternant A3: Syntax "pastorii"
   Alternant A4: Syntax "strainii"
RuleR5: <grup predicativ>::=
   Alternant A1:
      Syntax Label1: <grup predicativ elementar>
                     Coordinate Label3(1) ","!
             Label2: <grup predicativ>
                     Coordinate Label3(2)
      Dependencies Label3: @rel. de coord. prin virgula@(2)
   Alternant A2:
      Syntax Label1: <grup predicativ elementar>
                     Coordinate Label3(1) "si"!
             Label2: <grup predicativ>
                     Coordinate   Label3(2)
      Dependencies Label3: @rel. de coord. prin conjunctie@(2)
   Alternant A3:
      Syntax <grup predicativ elementar>
RuleR6: <grup predicativ elementar>::=
   Alternant A1:
      Syntax Label1: <verb>
                     Governor Label3 "ca"!
             Label2: <complement>
                     Subordinate Label3
      Dependencies
          Label3: @relatie predicat complement introdus prin ca@
   Alternant A2:
      Syntax <verb>
RuleR7: <verb>::=
   Alternant A1: Syntax "au venit"
   Alternant A2: Syntax "au vazut"
   Alternant A3: Syntax "se bucura"
   Alternant A4: Syntax "dantuie"
RuleR8: <complement>::=
   Alternant A1:
      Syntax <fraza>
```

We used the sign "!" to avoid some particles to be considered "heads".

Observation: In fact, the agreement in Romanian language for a multiple subject (for example the multiple subject "Profesorul (the professor) si (and) discipolul (the disciple)" with different predicates "au venit (they came) si (and) au vazut (they saw)" is much more complicated, but GRAALAN allows expressing all these elementary rules in a quite compact way.

5 GRAALAN multi word expression dictionary description

5.1 Informal definition

The Multiword Expression Dictionary (between two languages L_1 and L_2) contains linguistic correspondences between equivalent expressions belonging to the two languages. This dictionary will be created with special graphical tools that will create simultaneously the both L_1 -> L_2 and L_2 -> L_1 dictionaries.

The two expressions will be described as dependency trees with some correspondences between the tree nodes. Some correspondences will be established between the elements of the two trees. The MWE dictionary must contains rules that can be used to transform the source dependency tree into the target dependency tree. From this point of view, the syntactic elements are:

a) *Invariable* syntactic elements. This type of elements will have always the same form in all the contexts where they appear.

b) *Partial variable* elements. Such an element will contain always the same word in any context where it appears but it can have different inflected forms.

c) *Total variable* element: Such an element can contain in each context where it appears any word belonging to some lexical classes.

The transformations that can be realized on the source elements to obtain the target elements are:

a) *Correspondences*. The correspondences between two TA (terminals or procedural actions) indicate that the two elements in the two expressions are relative equivalent. This means that the TA from the target expression will take all the relations of the corresponding TA from the source expression (if this source TA has relations to some extensions of the expression).

b) *Transfers*. The transfer between two elements TA of the two expressions indicates the fact that the two elements are not equivalent but if TA from the source expression has some relations to the extension of the source expression than this relations will be transferred to the target element.

c) *Combinations*. It is a more complicated but very powerfull transformation that allows the transformation of an expression that contains some extensions that interfere with the expresion itself.

5.2 MWE dictionary creating and using

The Multi Word Expression dictionary can be created by a linguist in two ways: using a usual text editor or a special graphical application. The graphical application receives pairs of expressions from the linguist, analyzes these expressions (using the previous grammar description and morphological description for the two implied languages) and creates the dependency trees corresponding to the two expressions. The linguist can graphically indicate transformations between the two dependency trees. The graphical application generates the corresponding description using the two dependency trees and the transformations between them. The result is then compiled with a specific compiler (some errors can be detected and corrected if the description was not generated by the graphical application). The final result is stored in LKB.

From LKB the information is taken to realize different type of applications: grammar checkers, MWE dictionary, translation applications.

Example of MWE description:

Let us have the expression "a fi în gratiile cuiva" (to be apprecieted by someone) in Romanian language and the equivalent expression "être dans des bonnes grâces de qn." in French language.

The dependency tree of the expression "a fi în gratiile cuiva" can be represented as follows:

a fi (Relatie verb/complement circumstantial (în (Relatie prepozitie (gratiile (Relatie subst./atribut (cuiva()))))))

The dependency tree of the expression "être dans des bonnes grâces de qn." can be represented as follows:

être (Relatie verb/complement circumstantial (dans (Relatie prepozitie ((grâces (Relatie subst./atribut (bonnes()), Relatie subst./atribut (de (Relatie prepozitie (qn.())))))))))

Something of the following form will be in the GRAALAN MWE dictionary description:

```
Expression A "a fi în gratiile cuiva"
   Syntax Label1: "a fi" partial variabil [clasa = verb]
                   Governor Label5
          Label2: "în" invariabil [clasa = prepozitie]
                   Subordinate Label5 Governor Label6
          Label3: "gratia" invariabil [clasa = substantiv]
                   [gen =feminin] [numar = plural]
                   [caz = acuzativ]
                   Subordinate Label6 Governor L7
          Label4: "cuiva" total variabil
                   Subordinate Label7
   Dependencies
          Label5: @relatie verb complement circumstantial@
          Label6: @relatie prepozitie@
          Label7: @relatie substantiv/atribut@
Expression B "être dans des bonnes grâces de qn."
   Syntax Label1: "être" partial variabil [clasa = verb]
                    Governor Label5
          Label2: "dans" invariabil [clasa = prepozitie]
                    Subordinate Label5 Governor Label6
          Label3: "bon" invariabil [clasa = adjectiv]
                    [gen = feminin] [numar = plural]
          Label4: "grâce" invariabil [clasa = substantiv]
                    [gen =feminin] [numar = plural]
                    [caz = acuzativ]
                    Subordinate Label6 Governor Label7
          Label5: "de" invariabil [clasa = prepozitie]
          Label6: "qn." total variabil
                    Subordinate Label7
   Dependencies
          Label7: @relatie verb complement circumstantial@
          Label8: @relatie prepozitie@
          Label9: @relatie substantiv/atribut@
          Label10: @relatie substantiv/atribut@
          Label11: @relatie prepozitie@
   Transform A->B
      Correspondencies
             Label1/Label1, Label3/Label4, Label4/Lbel6
      Combination Label4 @relatie "et"@ "et"
   Transform B->A
      Correspondencies
             Label1/Label1, Label4/Label3, Label6/Label4
```

5 Conclusions

We presented the most important features of GRAALAN: the morphological, grammar and multi word expression correspondences descriptions. These were not a complete description but the aim was to give only a "flavor" of what and how linguistic knowledge can be described in GRAALAN. GRAALAN offers an integrated method to represent linguistic knowledge. In the real implementation GRAALAN compiler is completed with some graphical tools that make easier the work of a linguist.

GRAALAN can be used for natural language processing of a single language (in the applications like morphological description, morphological analysis, text annotation, grammar analysis, generation) but also for pairs of languages (like dictionaries and automatic machine translation).

The GDG with feature structure of the Romanian language was started to build using GRAALAN. The Romanian language has a very difficult grammar with quite free word order. For example, only to express the accord between a multiple governor and a subordinate (in gender, number, animation, order) there are about 1350 situations that must be observed. The description of the Morphological Configurator and (partially) the description of the grammar were realised so far for the Romanian Language.

Some correspondences descriptions between Romanian and French languages was started too.

References

[1] Ron Cole (Ed), *Survey of the State of the Art in Human Language Technology.*

[2] S. Kahane, *Grammaire d'Unification Sens-Texte. Vers un modèle mathématique articulé de la langue.*

[3] A. Joshi, L. Levi, L. Takabashi, *Tree Adjunct Grammars*, in Journal of the Computer and System Sciences, 1975.

[4] C. Pollard, I. Sag, *Head-Driven Phrase Structure Grammar*, Stanford CSLI, 1994.

[5] R. Kaplan, J. Bresnan, *Lexical Functional Grammar. A Formal System for Grammatical Representation*, in J. Bresnan (ed), The Mental Representation of Grammatical Relations, MIT Press, 1982.

[6] L. Tesnière, *Éléments de syntaxe structurelle*, Paris, Klincksieck, 1959.

[7] Stefan Diaconescu: *Natural Language Understanding Using Generative Dependency Grammar*, in Max Bramer, Alun Preece and Frans Coenen (Eds), Proceedings of ES2002, Cambridge UK, December 2002, Springer.

[8] Stefan Diaconescu: *Morphological Categorization Attribute Value Tree and XML*, in Mieczyslav A. Klopotek, Slawomir T. Wierzchon, Krzysztof Trojanowski (Eds), Proceedings of the International IIS: IIPWM'03 Conference, Poland, 2003.

[9] Stefan Diaconescu: Multiword Expression Translation Using Generative Dependency Grammar, in Proceedings of ESTAL 2004, Alicante, Spain

[10] Stefan Diaconescu, *Natural Language Agreement Description for Reversible Grammars*, in Tamás D. Gedeon, Lance Chun Che Fung (Eds.), Proceedings of AI 2003, Perth, Australia, December 2003, Proceedings, pp. 161-172.

[11] Gaifman, H., *Dependency Systems and phrase structure systems*, Information and Control, 1965, pp. 304-337.

[12] Mel'cuk, I.A., Pertsov, N.V. *Surface Syntax of English: A formal model within the Meaning-Text Framework*, Amsterdam/PA: John Benjamins, 1987.

[13] Hudson, R *English Word Grammar*, Oxford UK, Basil Blackwell, 1990.

[14] Chomsky, Noam *Generative Grammar: Its Basis, Development and Prospects*. Studies in English Linguistics and Literature, Special Issue, Kyoto University of Foreign Studies, 1988.

Biography

Name: Stefan Diaconescu

Address: Str. Fabrica de glucoza, Nr. 5, Sect.2, Bucharest, Romania

Education & Work experience: Doctor in in applied informatics – Artificial intelligence. Director of R&D Department, SOFTWIN

Tel: +(40) 21-233-0780

E-mail: sdiaconescu@softwin.ro