TR-I-0015

A Description of English Dialogues by Structural Correspondence Specification Language: SCSL

言語構造記述言語 SCSL による英語対話の記述

Rémi Zajac and Teruaki Aizawa
レミ・ザジャック　　相沢　輝昭

November, 1987

Abstract

This paper deals with an analysis and a description of English dialogues using the Structural Correspondence Specification Language (SCSL), developed at GETA. The corpus is a set of inter-keyboard communications, the topic of communications being international conference information.

This work was done when the first author was a visiting research scientist of ATR Interpreting Telephony Research Laboratories.

ATR Interpreting Telephony Research Laboratories
ATR 自動翻訳電話研究所

This is an excerpt from the report as a visiting research scientist

Period: from August 1, 1987 to September 30, 1987

Company: ATR Interpreting Telephony Research Laboratories
Osaka, Japan

Topic: A description of English dialogues by Structural Correspondence Specification Language: SCSL

Name: Rémi Zajac, Researcher of the GETA (Groupe d'Etudes pour la Traduction Automatique), Grenoble, France

A Report to
ATR Interpreting Telephony Research Laboratories,
September, 24, 1987.

Rémi ZAJAC

Table

Introduction

This report describes a part of the work I have done in ATR Interpreting Telephony Research Laboratories during August and September 1987. The final goal was to write a little grammar in the Structural Correspondance Specification Language (SCSL) in order to exemplify the features of this formalism. The corpus is a set of inter-keyboard communications, the topic of the commnunications being internationnal conference informations. The grammar is very elementary, all notions described are basic. I shall explain how I built this grammar and I hope that this will serve as an introduction to SCSL.

I first draw the limits of this work; I give an overall view of the grammar, I recall the basic principles used for the grammatical description of a sentence, I show how to write a board in SCSL and then, I give some examples for boards describing verb clause structures.

The references used for this work are listed below. SCSL [7] is an elaboration of the "Static Grammar" formalism [2, 3]. I briefly recall in section 3 the basic principles for the grammatical description of a sentence used at GETA, which are described in [1]. The SCSL grammar is based on [4], which is an English grammar model written in the "Static Grammar" formalism for pedagogical use.

1. Vauquois B., *Description de la structure intermédiaire*, Communication presented at Luxembourg Meeting, April 17-18, 1978.
2. Chappuy S., *Formalisation de la description des niveaux d'interprétation des langues naturelles. Etude menée en vue de l'analyse et de la génération au moyen de transducteurs*, Thèse de 3ième cycle "informatique", INPG, 1983.
3. Vauquois B., Chappuy S., *Static Grammars: a formalism for the description of linguistic models*, Proceedings of the Conference on Theoretical and Methodological issues in Machine Translation of Natural Languages, COLGATE University, Hamilton, N.-Y., U:S.A, August 14-16, 1985.
4. *Maquette pédagogique BEX-FEX*, Contrat ADI n°83/333, GETA, Grenoble, 1984.
5. *Longman Dictionary of Contemporary English*, 1987.
6. Quirk R., Greenbaum S., Leech G., Svartvik J., *A Comprehensive Grammar of the English Language*, Longman, 1985.
7. Zajac R., *SCSL : a linguistic specification language for MT*, COLING-86.

1. Goals and limits

Goals

This work is intended to be an introduction to SCSL and hence, the emphasis is put on the description of elementary grammatical notions using SCSL, rather than on the writing of a realistic English grammar. It is clear that a grammar for dialogs should integrate discourse notions, but these notions have not yet been clarified enough to be used for introducing a grammatical formalism. The grammar is limited to constituent structures, syntactic functions and predicate argument structure.

The two very basic notions of the language are the notion of **designation** of an individual, person or object, and the notion of the **formulation of a message** describing processes, relations, properties about individuals of the world. From these two *extra-grammatical* notions, we can derive the *grammatical* notions of **noun** and **sentence**. These two notions could be used as a kind of an axiomatic basis from which the grammar can be built without further introducing extra-grammatical notions.

The notion of sentence

In a dialog *some* utterances can be recognised by a locutor as "complete" when they are examinated out of the context of the dialog. But they are not complete as a message : no utterance can explicitly carry all the informations which are used for its interpretation. This notion of completeness is rather syntactic, that is, an utterance is complete when it can be transformed in systematic ways which modify the discourse value : declaration, interrogation, order, exclamation, etc. This implies that a complete utterance is not only a kind of a minimallly independent utterance, but also that it can be used as a constituent of more complex constructions.

The *simple sentence* could be defined as
- declarative
- most neutral with regard to communication purposes
- it cannot be analysed as a result of the integration of one or more simple sentences
- it is the best basis to describe interrogatives sentences, passives, emphazised sentences as the result of a set of transformations.

The *elementary sentence* is defined as a simple sentence where the suppression of an element is either impossible or destroys its properties of being a simple sentence. The simple sentence could then be seen as an expansion of the elementary sentence.

The *complex sentence* is analysed as an integration of two or more simple sentences.

The notion of noun

The most basic elements of the language are used to refer to individuals : proper nouns. Personal pronouns are very similar to proper nous in the sense that they are also used to refer to individuals (but the person refered is determinated by the context).

A **nominal expression** is a unit (eventually complex) which can be used in a way similar to proper nouns or personal pronouns to build a sentence. In simple cases, a nominal expression simply commutes with a proper noun or a pronoun.

The notion of predicate

From the two notions of simple sentence and nominal expression, the **predicate** can now be defined as the set of elements which structurate a set of nominal expressions to build a sentence. In the case of the simple sentence in English, the predicate is a verb (or more precisely a verb kernel which can integrate auxiliaries and modifiers) and the nominal expressions will be called the complements of the predicate.

The extension of the grammar

In this work, I have tried to describe the organisation of sentences as a product of a verb kernel and complements. For this, I have used commutation with nouns or pronouns, suppression of elements and discourse transformation (passive/active, declarative/interrogative, emphasis). I don't describe the internal structure of nominal expressions and verb kernels. I retain only a rough typology of nominal expressions that could be meaningful to differentiate different patterns of elementary sentences.

2. The Grammar

The Corpus

The corpus is a set of 16 inter-keyboard English communications translated from Japanese by a professional interpreter. Only the five first dialogs are studied as a basis for the grammar. The rest of the corpus should be analysed in order to extend the description and to account for structures that were not found in the first five dialogs (to facilitate this necessary extension, some generalisations have been made).

The first step was to recognise simple and complex sentences. A simple sentence is enclosed between braces { } in the commented corpus (annex 2). A complex sentence in composed of two or more simple sentences. There are two kinds of composition :

- coordination : {X} op1 {Y} where op1 is a coordinator ("and", "or", "nor", "but", "yet", "so", "," : the criterion is that a coordinator allows for subject elision in verb clause coordination).
- subordination : {X {op2 Y}} or {{op2 X} Y} where op2 is a subordinator.

The second step was to recognise the complements of a verbal kernel of a simple sentence. Complements are enclosed between square brackets []. The elements which are not part of a complement are the components of the predicate.

The third step was to classify sentences in seven different types :

1. VCL : declarative verb clauses, passive and active.
2. ICL : interrogative verb clause : yes/no questions, wh-questions, tag questions, declarative with a question mark.
3. DCL : directive verb clause.
4. FCL : formula clause such as "hello", "yes", "thank you very much", etc.
5. NPCL : noun phrase clause. It is often an answer to a previous question.
6. INP : interrogative noun phrase. It is often an echo of a previous clause.
7. CL : clauses that combine several different previous types : "Yes, it is, what can we do...".

The fourth step was to recognise obligatory and optional complements of a predicate and to draw a rough classification of nominal expressions : noun phrases (NP), adjectival phrases (AP), relative clauses (RELCL), participial clauses (PARTCL).

Here are some examples from the corpus. The reference to the corpus is enclosed between angle brackets <>. After the reference, the type of the sentence and the predicate-complements structure are found, with optionnal complements enclosed between parenthesis.

<01A02> VCLe == C1 V C2
{ [I] would like [to attend the conference] }

<01B03> VCLs == C1 V (C2) C3
{ [We] will send [you] [the registration form] }

< 01B04> VCLc == {C1 V (C2) (C3)} and {ø V (C2) (C3)}
{ { [you] may register [when you arrive at the conference] [in August] }
and
{pay [the registration fee] [at that time] } }

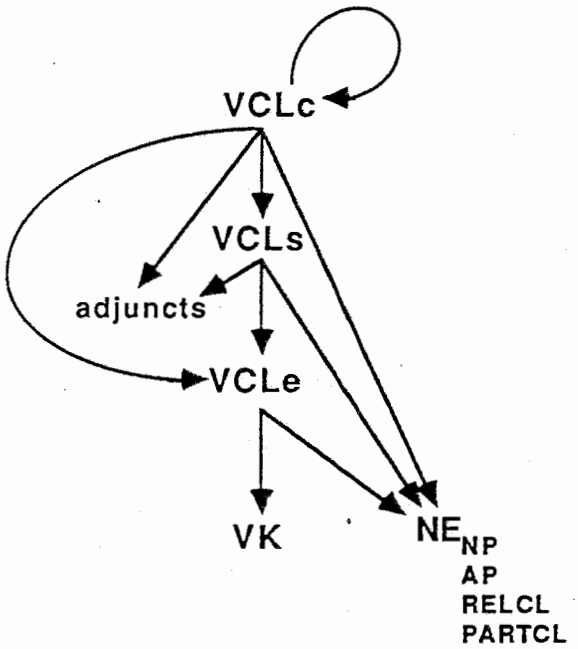
< 01A08.2> ICLc == {if {C1 V (C2)} , v C1 V C2}
{ If { [I] can come [only for the first day of the conference] }
will [you] return [part of the fee] ? }

Structure of the grammar

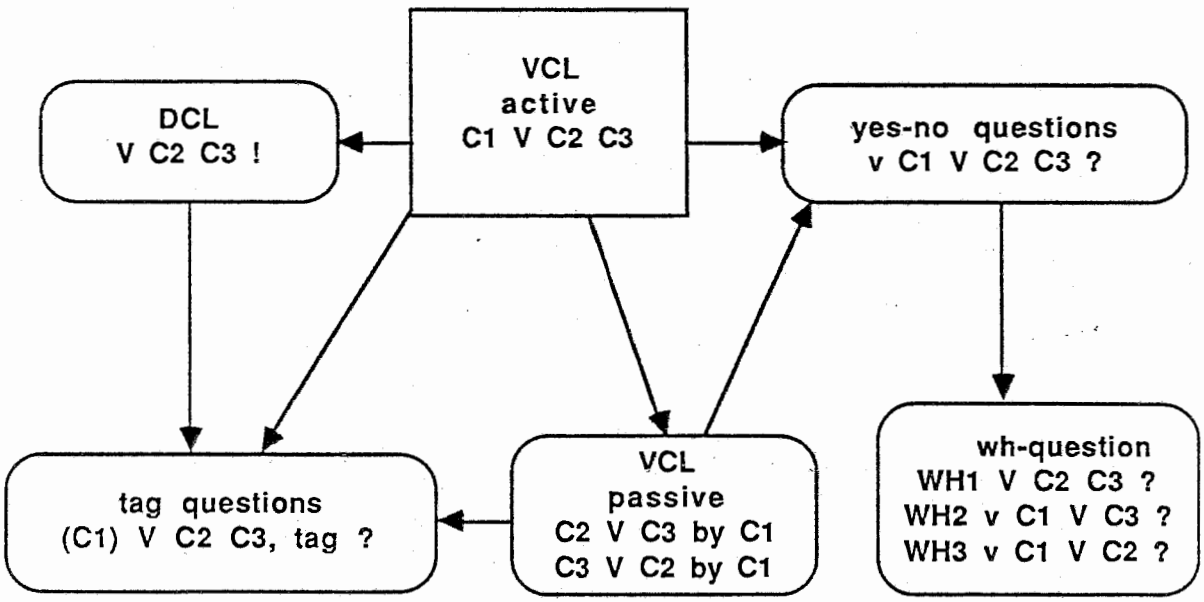
The overall structure of the grammar for describing verbal clauses is now clear : a VCLe (elementary verb clause) is composed of a verbal kernel and of nominal expressions (NP, AP, RELCL and PARTCL). This typology is a first rough approach to express some constraints on the nature of

complements. But I don't want, at this step of analysis, to refine this description.

A VCLs is composed of a VCLe and nominal expressions. A VCLc is a coordination or a subordination of several VCLe, VCLs or VCLc. Clause adjuncts may be integrated to VCLs or VCLc.



Examining the structure of declarative active and passive, interrogative and imperative, the relations between them can be drawn like this (V is the verb, v an auxiliary, Ci a complement) :



This can be used as a plan for building the grammar and some links may also say that, for example, a tag question is composed of a VCL plus a tag. The different transformations must be stated explicitly in order to ensure the coherence during the developpement.

3. Boards

Describing the grammatical structure using a labeled and decorated tree [1, 4]

The tree structure describes the constituent structure of a string : for the elementary sentence, the predicate-complement structure. The other levels of description (syntactic functions and predicate argument structure) are interpreted using the features attached to each node of the tree. The label of internal nodes will be the value of the syntagmatic class of the constituent (NP, AP, VCL,...).

To select the appropriate pattern for each particular verb, valencies have been declared as :

```
TYPE valencies <syntactic valency of a verb or state valency of a constituent> =
     SCALAR ( to-inf  <to- infinitive clause> ,
              ing-inf  <-ing infinitive clause> ,
              ed-inf <-ed infinitive clause> ,
              that    <that- clause> ,
              n       <noun phrase without preposition> ,
              about   <e.g., to talk about something> ,
              at      <e.g., to arrive at somewhere> ,
              for     <e.g., to pay for attending the conference> ,
              from    <e.g., to ride the subway from Kyoto station> ,
              in      <e.g., to participate in the conference> ,
              of      <e.g., to take care of something> ,
              on      <e.g., to record on tapes> ,
              to      <e.g., to pay to somebody> ,
              with    <e.g., to talk with somebody> ,
              by      <e.g., to pay by credit card> )
```

The valencies of a verb describe what are the prepositions which can govern a complement. These valencies are found in the description of each verb in the dictionary.

The state valency is described by the grammar and characterizes a constituent. In the case of noun phrases introduced by a preposition, the value of the valency is the preposition .

To describe the syntactic behavior of constituents, features for syntactic functions have been declared as :

```
TYPE sf-type <syntactic functions> =
     SCALAR ( gov <governor; central element of a phrase or a clause> ,
              subj    <subject; of the governor> ,
              obj1    <first object; of the governor> ,
              obj2    <second object; of the governor> ,
              atgov   <attribute of the governor> ,
              atsubj  <attribute of the subject> ,
              atobj   <attribute of the object> ,
              cpag    <agentive complement; introduced by "by" in a passive verb clause> ,
              coord   <rigth element of a coordination> ,
              circ    <circumstancial; complements of a verb clause without logical relationship
                       with the predicate> ,
              reg     <governs the function of a phrase; preposition or conjunction> ,
              0       <empty syntactic function>  )
```

The interpretation is the following : the governor of a constituent is always a leaf and the syntactic feature of a sub-constituent relates it to the governor. For example, the subject may be a noun phrase and will be the subject of a verb, the verb being on a leaf at the same level as the root of the subject noun phrase.

The features for logical relations, which describe the predicate-argument structure of a clause, have been declared :

```
TYPE logic-type <logical relations> =
     SCALAR ( pred   <predicate> ,
              arg0   <first argument> ,
              arg1   <second argument> ,
              arg2   <third argument> ,
              trl0   <transfer relation to first argument; e.g. the element is an attribute of the
                     subject of an active verb clause> ,
              trl1   <transfer relation to second argument; e.g. the element is an attribute of the
                     first object of an active verb clause> ,
              id     <same relation as the mother node> ,
              0      <empty relation> )
```

The interpretation is the following : the predicative unit of a constituent, generally a verb, relates the complements of the predicate, which are thus called the arguments. The argument places are defined by the verb class (its syntactic behavior) but the semantic role of arguments (semantic relations) depends on each particular verb; they are not described in this grammar.

The features attached to each node have been split into

- lexical properties (local to lexical units), generally attached to the leaves, but not only, e.g. for compounds or locutions;
- structural properties (role of a constituent in the whole structure), generally attached to internal nodes of the tree.

Writing a board

A board relates a tree to the corresponding string of constituents, which themselves relate sub-constituents recursively to strings of lexical units.

The "terminals" elements of the grammar are nodes or sub-trees (e.g. for compounds) computed by some morphological analyser from the string elements (grammatical informations such as gender, tense,...) and a dictionary which contains lexical units (and semantic features, syntactic behavior,...), or to be used by some morphological syntheziser to produce the string. By convention, the labels of the leaves will be the values of the lexical units.

Let us see how to write a board describing elementary verb clauses when the verb is a copular verb such as "to be" (see the complete board in the annex 1). We write the name of the board BOARD vcl-e-cop and the comments : the type of constituent described is an elementary verb clause, in the case of declarative clauses, with copular verbs. In this case, the typical pattern is C1 be C2, where C2 defines or modifies C1.

Examples from the corpus are : "The registration fee is $100 (US)", from the first conversation, locutor B, exchange 5; "I'am sorry", from the first conversation, locutor B, exchange 6; "Taking the subway and then the taxi is the fastest way", from the fourth conversation, locutor B, exchange 4, sentence 1.

The head of the board is then written :

```
BOARD  vcl-e-cop

TYPE  : elementary verb clause
CASE  : active declarative clause, copular verbs :'C1 be C2.
COM   : C2 defines or modifies  C1.

EXAMPLES : 1) "The registration fee is $100 (US)"  <01B05>
           2) "I'am sorry"  <01B06>
           3) "Taking the subway and then the taxi is the fastest way"  <04B04.1>
```

The pattern is C1 V C2, where C1 is the first complement, V the copular verb and C2 the second complement. We suppose that there are boards describing verbs kernels and nominal expressions. We have to write in the forest part, 3 tree patterns corresponding to C1, V and C2. C1 could be a noun phrase ("The registration fee"), a participial phrase ("Taking the subway and then the taxi") or a relative clause, hence, we write in the reference part that the references for C1 could be any board describing a noun phrase, a participial phrase or a relative clause : REFERENCES .1 : (np, partcl, relcl). We have also to define references for V, which is a verb kernel (vk), and for C2 which could be the same as C1 or an adjectival phrase (np, ap, partcl, relcl). For the complete description of these constituents, we have to refer to the boards describing these types of constituent.

We have now to refine the description of the patterns of the constituents, because not all verb kernels are allowed is this description, and the first element of C1 must not be an interrogative pronoun.

For the verb kernel, we have to say that the verb is a copular verb : we have to distinguish in the tree pattern a leaf which bears the syntactic function of governor of the verb kernel and have the category of verb and the sub-category of copular verb.

We write the tree pattern as .0($01, .02 ,$03) where the nodes are prefixed with a "." and the forests with a "$". The tree pattern has a root .0 which bears the label "vk" (verb kernel) and the associated decoration must contain the attribute voice(active), because the voice is an attribute of the verb kernel (which includes auxiliaries) and not of the verb itself, and we are describing only active

clauses : we write "vk".0 : voice(active) in the node constraints part of the forest part (keyword NOC). It has a leaf .02 which associated decoration must contain the attributes copular verb and governor function : .02 : v(copular) & sf(gov).

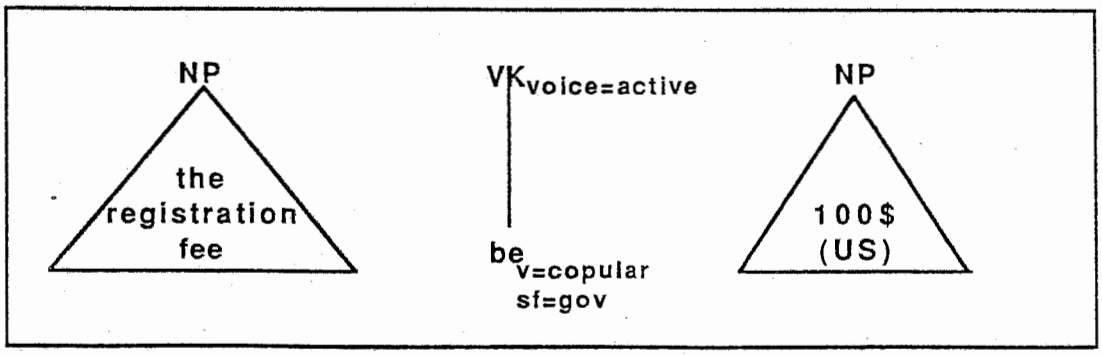
C1 can be absent when the clause is a part of a coordinated clause where the subject C1 is elided : the root of the pattern .1 suffixed with a "?". C1 cannot begin with an interrogative pronoun : we write the tree pattern as .1?(.11, $12), and the decoration of .11 must not have interrogative pronoun as attribute : .11 : ^(pr(rel)).

The labels for the roots of C1 and C2 are not necessary because the declaration of the references allows only for the proper boards, but they are written for better readability.

We have then to ensure that for each particular verb, we have the right type of complements. The kinf of complements which are allowed for each verb are defined in the dictionary as the valencies of the verb, and each constituent has a "state valency" : the state valency of the first complement (vl) must be equal to one of the first valency of the verb (vl0) and the state valency of the second complement must be equal to one of the second valency of the verb (vl1). In the forest pattern constraint part (keyword PAC), we write constraints linking different nodes of the forest pattern : vl(.1?) ∈ vl0(.02) & vl(.2) ∈ vl1(.02).

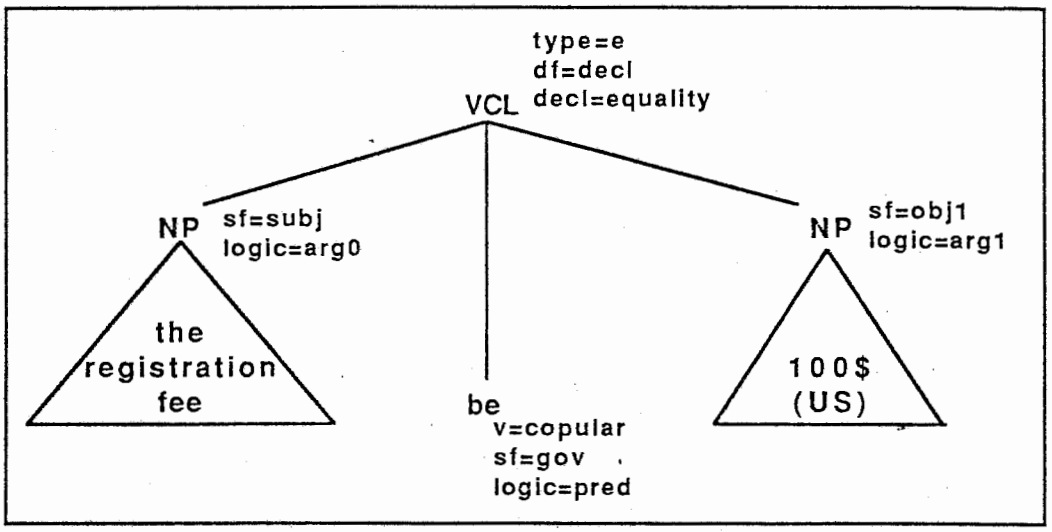
We have now completed the description of the family of strings, which is written :

```
REFERENCES .0 : (vk)
REFERENCES .1 : (np, partcl, relcl)
REFERENCES .2 : (np, ap, partcl, relcl)

<the tree part is not yet written>

FOREST      .1?(.11, $12) , .0($01, .02 ,$03 ) , .2($21)

     NOC    "np".1? V "partcl".1? V "relcl".1?
            .11       : ^(pr(rel))
            "vk".0    : voice(active)
            .02       : v(copular) & sf(gov)
            "np".2 V "ap".2 V "partcl".2 V "relcl".2

     PAC    vl(.1?) ∈ vl0(.02) & vl(.2) ∈ vl1(.02)
```



Instance of the forest part for the example <01B05> "The registration fee is $100 (US)".

We have to write now the corresponding family of trees. The governor of the clause is the governor of the verb kernel .02 : the tree could be understood as taking C1 and C2 as the left daughter and the right daughter of the verb kernel, as in this example :



The tree pattern is written : .0 (.1?(.11, $12) , $01 , .02 , $03 , .2($21)). The root has a label VCL (the syntagmatic class is VCL), the type of the clause is elementary, and it is a declarative clause : "vcl".0 : k(vcl) & type(e) & df(decl).

We can now define the syntactic function of each element : the first complement is the subject of the clause governed by the copular verb : .1? : sf(subj) and .02 : sf(gov). The syntactic function of C2 is attribute of the subject if C2 is an adjectival phrase, else it is the first object. These complex constraints are written in the pattern constraints part of the tree part :

```
PAC  (  k(.2)=ap  <=>  sf(.2)=atsubj)
     &
     (  (k(.2)=np  V  k(.2)=partcl V k(.2)=relcl)  <=>  sf(.2)=obj1)
```

The logical relations are then added : the predicate is the verb (.02 : logic(pred)), the first argument is C1 (.1? : logic(arg0)). If C2 is not an adjectival phrase, then it is the second argument (logic(.2)=arg1), else, it modifies C1, hence the logical relation should be the one of C1 (logic(.2)=trl0). These constraints are added in the PAC part of the tree.

We can sub-categorize the type of the clause : if C2 is an adjectival phrase, then the clause expresses a property (decl(.0)=property) else it expresses a definition or an equality (decl(.0)=equality). These constraints are added in the PAC part of the tree.

Finally, if C1 does not exist, the clause could only be used as a sub-constituent of a coordinated clause where the subject is elided : ^EXIST(.1?) <=> vcl(.1?)=seli.

We have now written the last part of this board :

```
TREE        .0 ( .1?(.11, $12) ,  $01 , .02 , $03 , .2($21) )

      NOC   "vcl".0    : k(vcl) & type(e) & df(decl)
            .1?        : sf(subj) & logic(arg0)
            .02        : sf(gov) & logic(pred)

      PAC  (k(.2)=ap  <=>
             (sf(.2)=atsubj & logic(.2)=trl0 & decl(.0)=property))
           &
           (          (k(.2)=np  V  k(.2)=partcl V k(.2)=relcl))  <=>
            (sf(.2)=obj1 & logic(.2)=arg1 & decl(.0)=equality))
           &
           (^EXIST(.1?) <=> vcl(.1?)=seli)
```

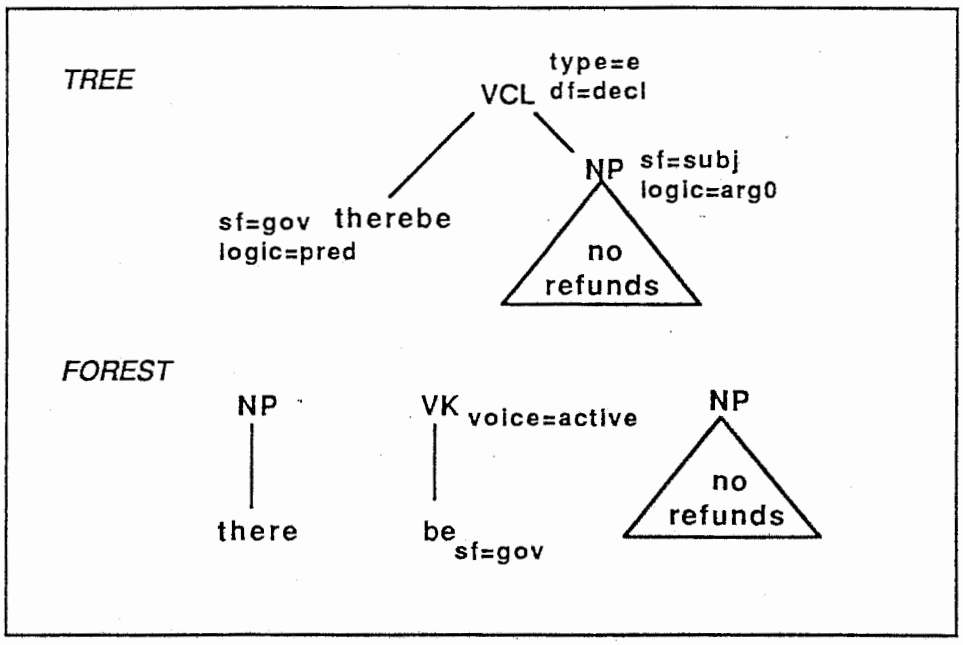
4. Examples

The Elementary Sentence

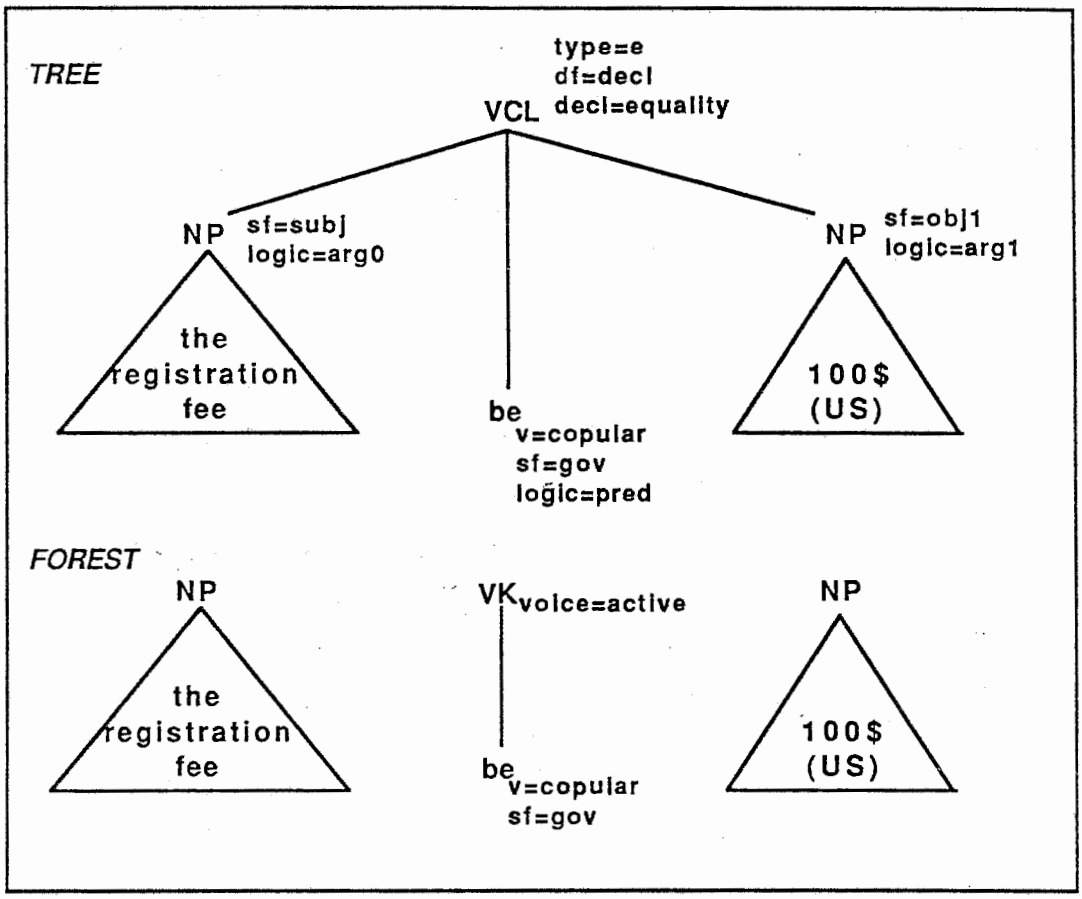
To write the boards for elementary sentences, I have extracted from the corpus the elementary and the simple sentences, and I have tried to make a classification of sentence patterns first based on the number of complements and then on the behavior of the complements under transformations (suppression, displacement, commutation, interrogative transformation, passivation, etc.). This kind of classification is partially done in the Longman grammar but as a classification of *verbs*. I took this classification to classify *sentence patterns* and I added 3 others :

1. bare predication pattern : there + be + NP
2. impersonal pattern (with copular verbs) : it + be + C2
3. copular verbs : C1 be C2
4. intransitive pattern : C1 V
5. monotransitive pattern : C1 V C2
8. complex transitive pattern : C1 V $C2_{o1}$ $C3_{att}$
6. ditransitive pattern 1 : C1 V $C2_{o2}$ $C3_{o1}$
7. ditransitive pattern 2 : C1 V $C2_{o1}$ $C3_{o2}$

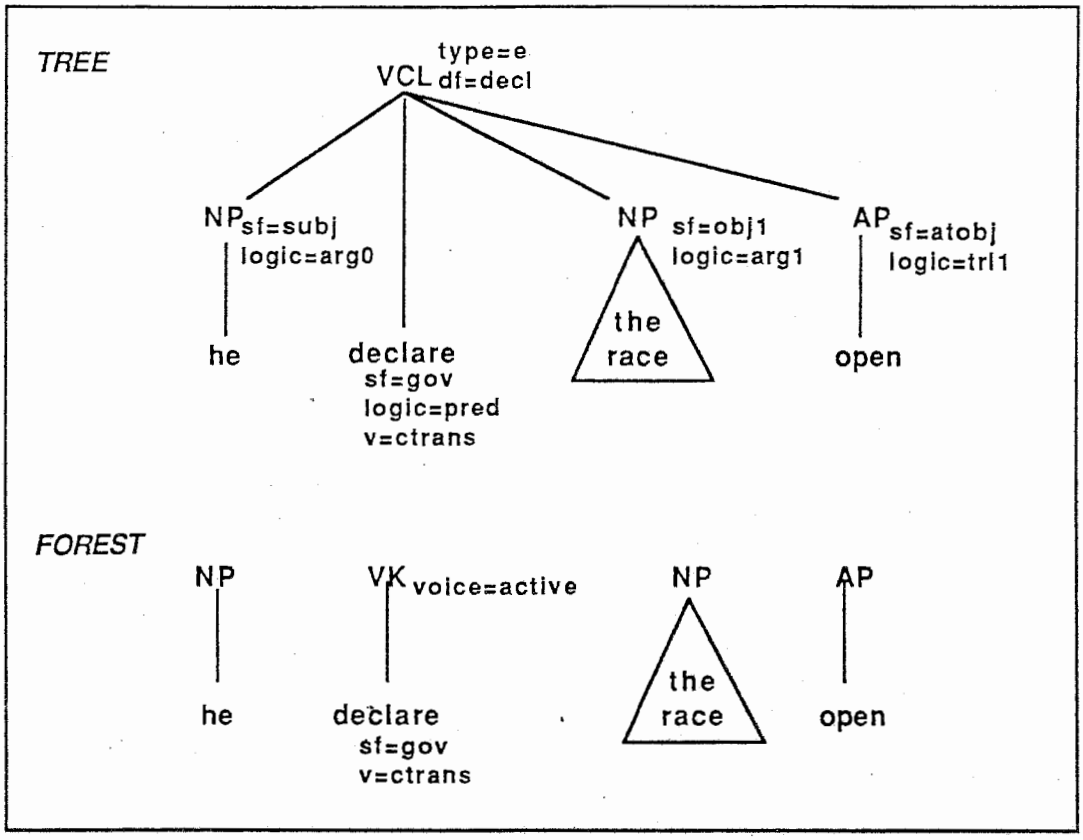
One board describes one pattern : there are 8 boards describing the elementary sentence patterns. To differentiate between the patterns 3 to 8 when a verb may have several different constructions, the type of verbs and the valencies of the complements must be checked. In order to account for the different behaviour of the patterns, two other levels of description must be added : the syntactic functions (subject, object, attribute,...) and the logical relations (predicate-argument relations).



Example of an instance of the board there *on the string "There are no refunds". The board describes bare existential sentence patterns. The string "There are" is described by a single node with the lexical unit "therebe" as a predicate.*



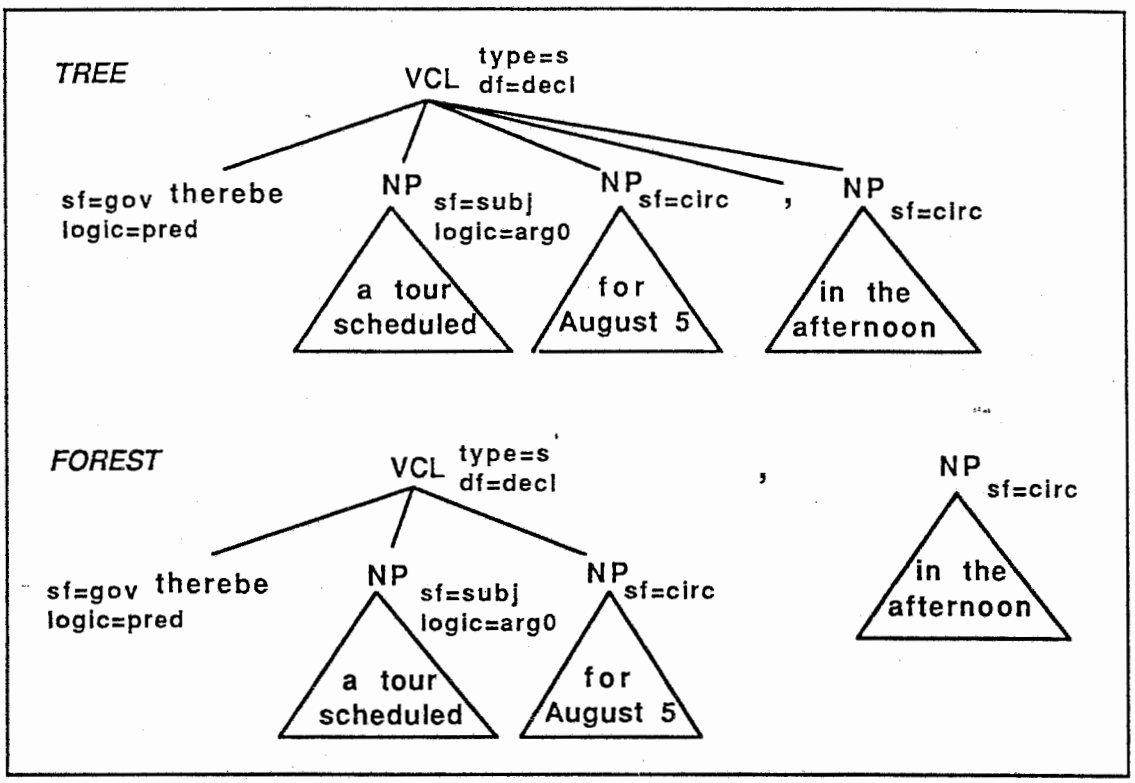
Example of an instance of the board vcl-e-cop *on the string "The registration fee is 100$ (US)". The board describes patterns with copular verbs. The second complement is a nominal phrase. Hence, the sentence expresses an equality (or a definition).*



Example of an instance of the board vcl-e-ct *on the string "He declared the race open". The board describes complex transitive patterns. The adjectival phrase is an attribute of the object : it has no proper logical relation with an other element, the logical relation to be referred to is the relation of the argument 1 : logic=trl1.*

The Simple Sentence

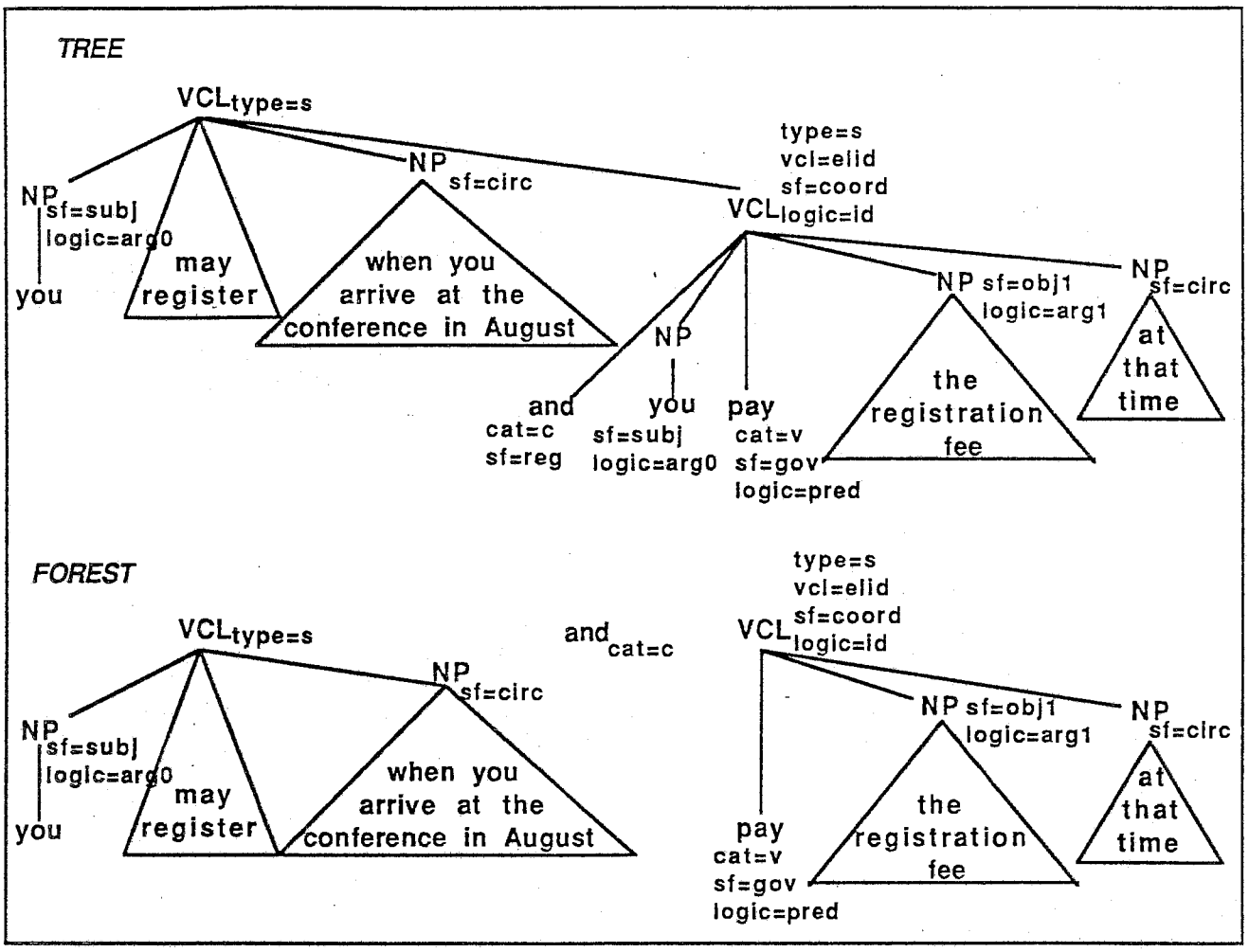
The simple sentence is described by two boards as an expansion of the elementary sentence with circumstancials and clause adjuncts on the left or on the right.



Example of an instance of the board vcl-s-r *on the string "There is a tour scheduled for August 5, in the afternoon". The sub-string "There is a tour scheduled on August 5" is described recursively by the same board.*

The Complex Sentence

The complex sentence is described by two boards for subordination (on the left or on the right), one board for coordination (the tree structure chosen for coordination is the same as the tree structure for right subordination) and one for coordination with elision of the subject. Two other boards are added for describing the expansion of coordination or subordination with circumstancials and clause adjuncts (but non were found in the corpus).



Example of an instance of the board vcl-c-coord-elid on the string "You may register when you arrive at the conference in August and pay the registration fee at that time". The two verb clauses are described by the board vcl-s-r, the second with elision of the subject, which is reinserted in the tree.

Conclusion

I hope that this report will give indications on how to write a grammar using SCSL. This toy grammar is rather simple, and not all features of SCSL have been used (constraints on the context for example). It should be regarded only as a starting point for some more extended grammar.

Annexe 1

GRAMMAR ekc

LANGUAGE : english
TYPOLOGY : inter-keyboard communication
DOMAIN : information about an internationnal conference
MEDIA : computer

COMMENT : the grammar describes declarative sentences found in a corpus of 16 conversations. It describes the structure of declarative sentences at the level of predicate/complements and there relations : the internal structure of the predicate and the complements are not described. The description accounts for the constituent structure, the syntactic functions and the logical relations.

TYPE logic-type <logical relations> =
SCALAR (pred <predicate> ,
arg0 <first argument> ,
arg1 <second argument> ,
arg2 <third argument> ,
trl0 <transfert relation to first argument; e.g. the element is an attribute of the subject of an active verb clause> ,
trl1 <transfert relation to second argument; e.g. the element is an attribute of the first object of an active verb clause> ,
id <same relation as the mother node> ,
0 <empty relation>)

TYPE sf-type <syntactic functions> =
SCALAR (gov <governor; central element of a phrase or a clause> ,
subj <subject; of the governor> ,
obj1 <first object; of the governor> ,
obj2 <second object; of the governor> ,
atgov <attribute of the governor> ,
atsubj <attribute of the subject> ,
atobj <attribute of the object> ,
coord <rigth element of a coordination> ,
circ <circumstancial; complements of a verb clause without logical relationship with the predicate> ,
reg <govern the function of a phrase; preposition or conjunction> ,
0 <empty syntactic function>)

TYPE sub-k <sub-category of syntagmatic classes> =
SCALAR (e <elementary constituent; no optionnal element> ,
s <simple constituent; there may be optionnal elements> ,
c <complex constituants; composed of simple constituants>)

TYPE be-clause <sub-category of declarative or interrogative> =
SCALAR (equality <definition; "oxygen is a gas"> ,
property <attribute; "this experiment is dangerous">)

```
TYPE df-type  <discourse functions>  =
      SCALAR ( decl   <declarative> : be-clause,
               inter  <interrogative> : be-clause,
               imp    <imperative> ,
               form   <formula; "yes", "goodbye">  )

TYPE voice-type <voices of verb clauses or verb phrase>  =
      SCALAR ( active, passive)


TYPE valencies <syntactic valency of a verb or state valency of a constituent>  =
      SCALAR ( to-inf <to- infinitive clause> ,
               ing-inf  <-ing infinitive clause> ,
               ed-inf       <-ed infinitive clause> ,
               that   <that- clause> ,
               n      <noun phrase without preposition> ,
               about  <e.g., to talk about something> ,
               at     <e.g., to arrive at somewhere> ,
               for    <e.g., to pay for attending the conference> ,
               from   <e.g., to ride the subway from Kyoto station> ,
               in     <e.g., to participate in the conference> ,
               of     <e.g., to take care of something> ,
               on     <e.g., to record on tapes> ,
               to     <e.g., to pay to somebody> ,
               with   <e.g., to talk with somebody> ,
               by     <e.g., to pay by credit card> )

TYPE k-type  <syntagmatic classes>  =
      SCALAR ( np     <noun phrase> :
                      SET ( type <complexity> : sub-k,
                            df <discourse function> : df-type,
                            svl <state valency> : valencies) ,
               ap     <adjectival phrase>
                      SET ( type <complexity> : sub-k,
                            df <discourse function> : df-type) ,
               vk     <verb kernel; predicative expression> :
                      SET ( type <complexity> : sub-k,
                            df <discourse function> : df-type,
                            voice : voice-type) ,
               vcl    <verb clause> :
                      SET ( type <complexity> : sub-k,
                            df <discourse function> : df-type,
                            voice : voice-type ,
                            imper <impersonnal pattern> ,
                            seli <elision of the subject>  ) ,
               relcl  <relative clause; introduced by a relative pronoun, that-clauses and
                      wh-clauses> :
                      SET ( type <complexity> : sub-k,
                            df <discourse function> : df-type,
                            svl <state valency> : valencies ) ,
               partcl <participial clauses; governed by an infinitive or a participle verb> :
                      SET ( type <complexity> : sub-k,
                            df <discourse function> : df-type,
                            svl <state valency> : valencies ) )
```

```
TYPE cptt-type <properties of constituents; these properties are on the internals nodes of the tree
     structure describing the constituent structure> =
     SET (     k       <syntagmatic class> : k-type ,
               sf      <syntactic function> : sf-type ,
               logic   <logical relation> : logic-type  )


TYPE ucl <upper case letters> =
     STRING   "A" + "B" + "C" + "D" + "E" + "F" + "G" + "H" + "I" + "J" + "K" + "L" + "M" +
              "N" + "O" + "P" + "Q" + "R" + "S" + "T" + "U" + "V" + "W" + "X" + "Y" + "Z"

TYPE lcl <lower case letters> =
     STRING   "a" + "b" + "c" + "d" + "e" + "f" + "g" + "h" + "i" + "j" + "k" + "l" + "m" +
              "n" + "o" + "p" + "q" + "r" + "s" + "t" + "u" + "v" + "w" + "x" + "y" + "z"

TYPE othl <others "letters"> =  "-"

TYPE digit = STRING  "0" + "1" + "2" + "3" + "4" + "5" + "6" + "7" + "8" + "9"

TYPE punct = STRING  "," + ";" + ":" + "." + "?" + "!" + "(" + ")"

TYPE special = STRING  """" + "'" + "$"

TYPE occ-type <occurrence of a word> =
     STRING   (ucl + lcl + othl + digit + special)+   +   punct

TYPE lu-type <lexical units; defined in the dictionary> =
     STRING   (lcl + othl)+   +   punct


TYPE vl-type <valency of the complements of a verb> =
     SET (     vl0 : valencies <valency of the first complement (subject)> ,
               vl1 : valencies <valency of the second complement> ,
               vl2 : valencies <valency of the third complement> )

TYPE v-type <verb classes and properties> =
     SET (     copular     <copular; "This rose is blue"> ,
               intrans     <intransitive; "Juliet loves"> ,
               1trans      <monotransitive; "Juliet loves roses"> ,
               ctrans            <complex transitive; "That music drives me mad"> ,
               2trans1     <ditransitive 1; "Juliet gave Romeo a rose"> ,
               2trans2     <ditransitive 2; "Juliet gave a rose to Romeo"> ,
               vl          <valencies> : vl-type )


TYPE pn-type <pronoun classes> =
     SCALAR ( rel    <relative; who, which> ,
              int    <interrogative; who, which> ,
              pers   <personal; he, you, it>  )

TYPE a-type <adjunct classes> =
     SCALAR ( adcl   <clause adjuncts; however, frankly> )
```

```
TYPE cat-type <word classes> =
     SCALAR ( v     <verb> : v-type ,
              n     <noun> ,
              pn    <pronoun> : pn-type ,
              s     <subordinbator> ,
              c     <coordinator; and, but, yet, so, nor> ,
              p     <punctuation> ,
              a     <adjunct> : a-type )

TYPE uppt-type <lexical units properties; these properties are attached to the leaves of the tree
               structure (exept for compounds)> =
     SET (    occ   <occurence of the lexical unit in the text> : occ-type ,
              lu    <lexical unit; defined in the dictionary> : lu-type ,
              cat   <word class> : cat-type,
              sf    <syntactic function> : SCALAR (gov, 0) ,
              logic <logical relation> : SCALAR (pred, 0) )


LABEL lbl <label of a node; syntagmatic class for internal nodes, lexical units for leaves> =
     STRING (lcl)+ + punct + special


DECO ppt <properties attached to a node> =
     SCALAR ( cppt  <constituent properties> : cppt-type ,
              uppt  <lexical unit properties> : uppt-type )


HIERARCHY     k-type <hierarchy between boards; defined on the syntagmatic classes> =
     (vcl(type(c)) , vcl(type(c)))
     (vcl(type(c)) , vcl(type(s)))
     (vcl(type(c)) , vcl(type(e)))
     (vcl(type(c)) , np)
     (vcl(type(c)) , ap)
     (vcl(type(c)) , relcl)
     (vcl(type(c)) , partcl)
          (vcl(type(s)) , vcl(type(e)))
          (vcl(type(s)) , np)
          (vcl(type(s)) , ap)
          (vcl(type(s)) , relcl)
          (vcl(type(s)) , partcl)
               (vcl(type(e)) , vk)
               (vcl(type(e)) , np)
               (vcl(type(e)) , ap)
               (vcl(type(e)) , relcl)
               (vcl(type(e)) , partcl)
```

BOARD vcl-e-there

TYPE : elementary verb clause.
CASE : Existential predication pattern : there + be + C2 (NP).

EXAMPLES : 1) "there are no refunds" <01B08>

REFERENCES .0 : (vk)
REFERENCES .2 : (np)

```
TREE        .0 ( $01,  .02,  $03,  .2($21) )

    NOC     "vcl".0    : k(vcl) & type(e) & df(decl)
            "therebe".02  : lu("therebe") & sf(gov) & logic(pred)
            "np".2     : sf(subj) & logic(arg0)

FOREST      .1(.11)  ,  .0($01,  .02,  $03)  ,  .2($21)

    NOC     "np".1
            "there".11 : cat(pr) & lu("there")
            "vk".0     : voice(active)
            "be".02    : cat(v) & sf(gov) & lu("be")
            "np".2     : vl(n)
```

BOARD vcl-e-it

TYPE : elementary verb clause
CASE : active declarative clause, intransitive pattern : C1 V.

EXAMPLES : 1) "I see" <01A08.1>
2) "she may" <03B05>
3) "Either one will do" <05B03.2>

REFERENCES .0 : (vk)
REFERENCES .1 : (np, partcl, relcl)

TREE .0 (.1?(.11, $12) , $01 , .02 , $03)

NOC "vcl".0 : k(vcl) & type(e) & df(decl)
.1? : sf(subj) & logic(arg0)
.02 : sf(gov) & logic(pred)

PAC ^EXIST(.1?) <=> vcl(.0)=seli

FOREST .1?(.11, $12) , .0($01, .02 ,$03)

NOC "np".1? V "partcl".1? V "relcl".1?
.11 : ^(pr(rel) V pr(int))
"vk".0 : voice(active)
.02 : v(intrans) & sf(gov)

PAC vl(.1?) ∈ vl0(.02)

BOARD vcl-e-cop

TYPE : elementary verb clause
CASE : active declarative clause, copular verbs : C1 be C2.
COM : C2 defines or modifies C1.

EXAMPLES : 1) "The registration fee is $100 (US)" <01B05>
2) "I'am sorry" <01B06>
3) "Taking the subway and then the taxi is the fastest way" <04B04.1>

REFERENCES .0 : (vk)
REFERENCES .1 : (np, partcl, relcl)
REFERENCES .2 : (np, ap, partcl, relcl)

TREE .0 (.1?(.11, $12) , $01 , .02 , $03 , .2($21))

NOC "vcl".0 : k(vcl) & type(e) & df(decl)
.1? : sf(subj) & logic(arg0)
.02 : sf(gov) & logic(pred)

PAC (k(.2)=ap <=>
(sf(.2)=atsubj & logic(.2)=trl0 & decl(.0)=property))
&
((k(.2)=np V k(.2)=partcl V k(.2)=relcl)) <=>
(sf(.2)=obj1 & logic(.2)=arg1 & decl(.0)=equality))
&
(^EXIST(.1?) <=> vcl(.1?)=seli)

FOREST .1?(.11, $12) , .0($01, .02 ,$03) , .2($21)

NOC "np".1? V "partcl".1? V "relcl".1?
.11 : ^(pr(rel) V (pr(int))
"vk".0 : voice(active)
.02 : v(copular) & sf(gov)
"np".2 V "ap".2 V "partcl".2 V "relcl".2

PAC vl(.1?) ∈ vl0(.02) & vl(.2) ∈ vl1(.02)

BOARD vcl-e-imper

TYPE : elementary verb clause.
CASE : impersonnal with copular verbs only : it + be + C2 C3.

EXAMPLES : 1) "It would be more convenient to pay by credit card" <02B09.1>

REFERENCES .0 : (vk)
REFERENCES .1 : (np)
REFERENCES .2 : (np, ap)
REFERENCES .3 : (partcl, relcl)

TREE .0 (.1(.11) , $01 , .02 , $03 , .2($21) , .3($31))

NOC "vcl".0 : k(vcl) & type(e) & df(decl)
.02 : sf(gov) & logic(pred)
.1? : sf(subj) & logic(0)
.2 : sf(atsubj) & logic(trl0)
.3 : sf(0) & logic(arg0)

FOREST .1(.11) , .0($01, .02 ,$03) , .2($21) , .3($31)

NOC "np".1 : vl(n)
"it".11 : pr(pers) & lu("it")
"vk".0
.01 : v(copular) & sf(gov)
"np".2 V "ap".2
"partcl".3 V "relcl".3

PAC vl(.3) ∈ vl0(.01)

BOARD vcl-e-1t

TYPE : elementary verb clause.
CASE : active declarative verb clause, monotransitive pattern : C1 V C2.
COM : C2 is object1. Passive : C2 V by C1.

EXAMPLES : 1) "I don't read the paper" <01A04.2>
2) "we would prefer that you pay in US dollars only" <01B06>
3) "You need to fill out the registration form" <02B07.1>

REFERENCES .0 : (vk)
REFERENCES .1 : (np, partcl, relcl)
REFERENCES .2 : (np, partcl, relcl)

TREE .0 (.1?(.11,$12) , $01 , .02 , $03 , .2($21))

NOC "vcl".0 : k(vcl) & type(e) & df(decl)
.1? : sf(subj) & logic(arg0)
.02 : cat(v) & sf(gov) & logic(pred)
.2 : sf(obj1) & logic(arg1)

PAC ^EXIST(.1?) <=> vcl(.0)=seli

FOREST .1?(.11,$12) , .0($01, .02, $03) , .2($21)

NOC "np".1? V "partcl".1? V "relcl".1?
.11 : ^(pr(rel) V pr(int))
"vk".0 : voice(active)
.02 : v(1trans) & sf(gov)
"np".2 V "partcl".2 V "relcl".2

PAC vl(.1?) ∈ vl0(.02) & vl(.2) ∈ vl1(.02)

BOARD vcl-e-ct

TYPE : elementary verb clause.
CASE : active declarative clause, complex transitive pattern : C1 V C2 C3
COM : C2 is object1, C3 is attribute. Passive : C2 V C3 by C1.

EXAMPLES : 1) "He declared the race open"
2) "My contract allows me to take one month's leave"

REFERENCES .0 : (vk)
REFERENCES .1 : (np, partcl, relcl)
REFERENCES .2 : (np, partcl, relcl)
REFERENCES .3 : (np, ap, partcl, relcl)

TREE .0 (.1?(.11, $12) , $01 , .02 , $03 , .2($21) , .3($31))

NOC "vcl".0 : k(vcl) & type(e) & df(decl)
.1? : sf(subj) & logic(arg0)
.02 : sf(gov) & logic(pred)
.2 : sf(obj1) & logic(arg1)
.3 : sf(atobj) & logic(trl1)

PAC ^EXIST(.1?) <=> vcl(.0)=seli

FOREST .1?(.11, $12) , .0($01, .02 ,$03) , .2($21) , .3($31)

NOC "np".1? V "partcl".1? V "relcl".1?
.11 : ^(pr(rel V pr(int))
"vk".2 : voice(active)
.02 : v(ctrans) & sf(gov)
"np".2 V "partcl".2 V "relcl".2
"np".3 V "ap".3 V "partcl".3 V "relcl".3

PAC vl(.1?) ∈ vl0(.02) & vl(.2) ∈ vl1(.02) & vl(.3) ∈ vl2(.02)

BOARD vcl-e-2t1

TYPE : elementary verb clause
CASE : active declarative clause, ditransitive pattern : C1 V C2 C3.
COM : C2 is object2, C3 is object1. Passive : C3 V C2 by C1.

EXAMPLES : 1) "We will send you the registration form" <01B03>

REFERENCES .0 : (vk)
REFERENCES .1 : (np, partcl, relcl)
REFERENCES .2 : (np, relcl)
REFERENCES .3 : (np, partcl, relcl)

TREE .0 (.1?(.11, $12) , $01 , .02 , $03 , .2($21) , .3($31))

NOC "vcl".0 : k(vcl) & type(e) & df(decl)
.1? : sf(subj) & logic(arg0)
.02 : sf(gov) & logic(pred)
.2 : sf(obj2) & logic(arg2)
.3 : sf(obj1) & logic(arg1)

PAC ^EXIST(.1?) <=> vcl(.0)=seli

FOREST .1?(.11, $12) , .0($01, .02 ,$03) , .2($21) , .3($31)

NOC "np".1? V "partcl".1? V "relcl".1?
.11 : ^(pr(rel) V pr(int))
"vk".0 : voice(active)
.02 : v(2trans1) & sf(gov)
"np".2 V "relcl".2
"np".3 V "partcl".3 V "relcl".3

PAC vl(.1?) ∈ vl0(.02) & vl(.2) ∈ vl1(.02) & vl(.3) ∈ vl2(.02)

BOARD vcl-e-2t2

TYPE : elementary verb clause
CASE : active declarative clause, ditransitive pattern : C1 V C2 C3.
COM : C2 is object1, C3 is object2. Passive : C2 V C3 by C1.

EXAMPLES : 1) "We will send the registration form to you"

REFERENCES .0 : (vk)
REFERENCES .1 : (np, partcl, relcl)
REFERENCES .2 : (np, partcl, relcl)

TREE .0 (.1?(.11, $12) , $01 , .02 , $03 , .2($21) , .3($31))

NOC "vcl".0 : k(vcl) & type(e) & df(decl)
.1? : sf(subj) & logic(arg0)
.02 : sf(gov) & logic(pred)
.2 : sf(obj1) & logic(arg1)
.3 : sf(obj2) & logic(arg2)

PAC ^EXIST(.1?) <=> vcl(.0)=elid

FOREST .1?(.11, $12) , .0($01, .02 ,$03) , .2($21) , .3($31)

NOC "np".1? V "partcl".1? V "relcl".1?
.11 : ^(pr(rel) V pr(int))
"vk".0 : voice(active)
.02 : v(2trans2) & sf(gov)
"np".2 V "partcl".2 V "relcl".2
"np".3 V "partcl".3 V "relcl".3

PAC vl(.1?) ∈ vl0(.02) & vl(.2) ∈ vl1(.02) & vl(.3) ∈ vl2(.02)

BOARD vcl-s-r

TYPE : simple verb clause.
CASE : active declarative clause, including circumstancials and clauses adjuncts on the right.

EXAMPLES : 1) "you may register when you arrive at the conference in August" <01B04>
2) "there is a tour scheduled for August 5, in the afternoon." <03B04.1>

REFERENCES .0 : (vcl)
REFERENCES .1 : (np, partcl, NONE)

TREE .0 ($01 , .2? , .1($11))

NOC "vcl".0 : type(s)

PAC (cat(.1)=adcl <=> sf(.1)=atg)
&
((k(.1)=np V k(.1)=partcl) <=> sf(.1)=circ)

FOREST .0($01) , .2? , .1($11)

NOC "vcl".0 : k(vcl) & (type(e) V type(s))
.1 : cat(adcl) V k(np) V k(partcl)
",".2? : cat(punct) & lu(",")

BOARD vcl-s-I

TYPE : simple verb clause
CASE : active declarative clause, including circumstancials and clauses adjuncts on the left.

EXAMPLES : 1) "In July, the conference will be held"

REFERENCES .0 : (vcl)
REFERENCES .1 : (np, partcl, NONE)

TREE .0 (.1($11) , .2? , $01)

NOC "vcl".0 : type(s)

PAC (cat(.1)=adcl <=> sf(.1)=atg)
&
((k(.1)=np V k(.1)=partcl) <=> sf(.1)=circ)

FOREST .1?($11) , .2? , .0($01)

NOC "vcl".0 : k(vcl) & (type(e) V type(s))
.1 : cat(adcl) V k(np) V k(partcl)
",".2? : cat(punct) & lu(",")

BOARD vcl-c-sub-r

TYPE : complex verb clause
CASE : active declarative clause with subordinate clauses on the right.

REFERENCES .0 : (vcl)
REFERENCES .1 : (vcl, partcl, ap)

```
TREE        .0 ( $01 ,  .1(.3?, .2, $11) )

      NOC   "vcl".0   : type(c)
            .1        : sf(circ)
            .2        : sf(reg)

FOREST       .0($01)  ,  .3? ,  .2  ,  .1($11)

      NOC   "vcl".0
            "ap".1 V "partcl".1  V  "vcl".1
            .2        : cat(s) & ^lu("that")
            ",".3?    : cat(p)
```

BOARD vcl-c-sub-l

TYPE : complex verb clause
CASE : active declarative clause with subordinate clauses on the left.

EXAMPLES : 1) "If I can come only for the first day, you will return part of the fee"

REFERENCES .0 : (vcl)
REFERENCES .1 : (vcl, partcl, ap)

TREE .0 (.1(.2, $11, .3?) , $01)

NOC "vcl".0 : type(c)
.1 : sf(circ)
.2 : sf(reg)

FOREST .2 , .1($11) , .3? , .0($01)

NOC "vcl".0
"ap".1 V "partcl".1 V "vcl".1
.2 : cat(s) & ^lu("that")
",".3? : cat(p)

BOARD vcl-c-coord

TYPE : complex verb clause
CASE : active declarative clause with coordinate clauses clauses.

EXAMPLES : 1) "The hotel names are Kyoto Hotel, Kyoto Prince Hotel, Kyoto Royal Hotel, they are all close to Kyoto station." <03B03>

REFERENCES .0 : (vcl)
REFERENCES .1 : (vcl)

TREE .0 ($01, .2, .1($11))

NOC "vcl".0 : type(c)
.1 : sf(coord) & logic(id)
.2 : sf(reg)

FOREST .0($01) , .2 , .1($11)

NOC "vcl".0
"vcl".1
.2 : cat(c)

BOARD vcl-c-coord-elid

TYPE : complex verb clause.
CASE : active declarative clause with subject elision in coordinate clauses.

EXAMPLES : 1) "you may register when you arrive at the conference in August and pay the registration fee at that time." <01B04>

REFERENCES .0 : (vcl)
REFERENCES .1 : (vcl)

TREE .0($01,
.02($021, .022, $023), $03) ,
.1(.2 , $11, .02(.022),.12, $13))

NOC "vcl".0 : type(c)
.1 : sf(coord) & logic(id)
.2 : sf(reg)

FOREST .0($01, .02($021, .022, $023), $03)) , .2 , .1($11, .12, $13)

NOC "vcl".0 : ^vcl(imper)
"np".02 : sf(subj)
.022 : sf(gov)
"vcl".1 : vcl(elid)
.12 : sf(gov)
.2 : cat(c)

PAC voive(.0)=voice(.1)

BOARD vcl-c-r

TYPE : complex verb clause.
CASE : active declarative clause, including circumstancials and clauses adjuncts on the right.

REFERENCES .0 : (vcl)
REFERENCES .1 : (np, partcl, NONE)

TREE .0 ($01 , .2? , .1($11))

NCC "vcl".0 : type(c)

PAC (cat(.1)=adcl <=> sf(.1)=atg)
&
((k(.1)=np V k(.1)=partcl) <=> sf(.1)=circ)

FOREST .0($01) , .2? , .1($11)

NCC "vcl".0 : k(vcl) & type(c)
.1 : cat(adcl) V k(np) V k(partcl)
",".2? : cat(punct) & lu(",")

BOARD vcl-c-l

TYPE : complex verb clause
CASE : active declarative clause, including circumstancials and clauses adjuncts on the left.

REFERENCES .0 : (vcl)
REFERENCES .1 : (np, partcl, NONE)

```
TREE        .0 ( .1($11) , .2? , $01 )

      NOC   "vcl".0   : type(c)

      PAC   (cat(.1)=adcl <=> sf(.1)=atg)
            &
            ((k(.1)=np V k(.1)=partcl) <=> sf(.1)=circ)

FOREST      .1?($11) , .2? , .0($01)

      NOC   "vcl".0   : k(vcl) & type(c)
            .1        : cat(adcl) V k(np) V k(partcl)
            ",".2?    : cat(punct) & lu(",")
```

```
<A01>   FCLe == P
        { Hello } .

<B01>   FCLe == P
        { Hello } .

<A02>   VCLe == C1 V C2 / NP to-infinitive
        { [I] would like [to attend the conference] } .

<B02>   ICLyn.s == v C1 V (C2) C3 / NP NP NP
        { Shall [I] send [you] [the information] } ?

<A03.1> FCLc == FCLe , FCLe
        FCLe == P
        FCLe == P
        { Yes} , { please } .
<A03.2> ICLwh.e == wh v C1 V / NP
        { [How] do [I] register } ?

<B03>   VCLs == C1 V (C2) C3 / NP NP NP
        { [We] will send [you] [the registration from] } .

<A04.1> FCLe == P
        { Yes } .
<A04.2> VCLs == CLM C1 V (C2) / NP NP
        CLM == but
        { But } { [I] don't read [the paper] } .
<A04.3> VCLe == C1 V C2 / NP to-infinitive
        { [I] would just like [to attend the conference] } .
<A04.4> ICLyn.e == V C1 C2 / NP AP
        { Is [it] [all right] } ?

<B04>   CL   == FCLe , VCLc
        FCLe == P
        VCLc == VCLs and VCLs
        VCLs == C1 V (C2) (C3) / NP wh-clause NP
        VCLs == V (C2) (C3) / NP NP
        { Yes } ,
        { [you] may register [when you arrive at the conference]
          [in August] } and
        { pay [the registration fee] [at that time] } .

<A05>   ICLwh.e == wh V C1 / NP
        { [How much] is [the registration fee] } ?

<B05>   VCLe == C1 V C2 / NP NP
        { [The registration fee] is [$100 (US)] } .

<A06>   ICLyn.s == v C1 V (C2) / NP NP
        { Can [I] pay [by yen] } ?

<B06>   VCLc == VCLe but VCLe
        VCLe == C1 V C2 / NP AP
        VCLe == C1 V C2 / NP that-clause
        { [I] 'm [sorry] } but
        { [we] would prefer [that you pay in US dollars only] } .

<A07>   ICLvc1.s == C1 V C2 (C3) / NP NP NP
        { [You] mean [$100] [for three days] } ?

<B07.1> FCLe == P
        { Yes } .
<B07.2> VCLe == C1 V C2 / NP AP
        { [The hotel and food expenses] would be [extra] } .
```

```
<A08.1> VCLe == C1 V / NP
        { [I] see } .
<A08.2> ICLyn.c == (if VCLs , ) ICLyn.e
        VCLs    == C1 V (C2) / NP NP
        ICLyn.e == v C1 V C2 / NP NP
        { { If / [I] can come [only for the first day of the conference] } ,
          will [you] return [part of the fee] } ?
<A08.3> INP
        { [At least 1/3 of it] } ?

<B08>   VCLc == VCLe but VCLe
        VCLe == C1 V C2 / NP AP
        VCLe == V C2 / NP
        { [I] 'm [sorry] } but { there are [no refunds] } .

<A09.1> FCLc == FCLe , FCLe
        FCLe == P
        { Well} , { all right } .
<A09.2> ICLwh.e == wh V C1 / NP
        { [What] is [the official language of the conference] } ?

<B09>   VCLe == C1 V C2 / NP NP
        { [The official languages] are [English and Japanese] } .

<A10.1> VCLe == C1 V C2 / NP AP
        { [I] 'm not [good at English] } .
<A10.2> ICLyn.e == v C1 V C2 / NP NP
        { Do [you] have [simultaneous interpretation] } ?

<B10>   FCLe == P
        { Yes } .

<A11.1> VCLe == C1 V C2 / NP to-infinitive
        { [I] 'd like [to talk with foreign speakers or participants] } .
<A11.2> ICLyn.s == v C1 V (C2) C3 (C4) / NP NP NP to-infinitve
        { Will [you] get [me] [an interpreter] [to help me] } ?

<B11>   VCLe == C1 V C2 / NP AP
        { [Interpreters] will be [available on request] } .

<A12.1> VCLe == C1 V / NP
        { [I] see } .
<A12.1> FCLe == P
        { Thank you very much } .

<B12>   ICLyn.e == v C1 V C2 / NP NP
        { Do [you] have [any other questions] } ?

<A13.1> FCLe == P
        { No } .
<A13.2> FCLe == P
        { Thank you } .

<B13>   FCLe == P
        { [Good day] } .

------------------------------------------------------------------------ 02

<A01>   FCLe == P
        { Hello } .

<B01>   ICLyn.e == v C1 V (C2) / NP NP
        { May [I] help [you] } ?

<A02.1> VCLs == C1 V (C2) / NP that-clause
```

```
        { [I] hear [that first international conference of interpreting
          telephony will be held in August] } .
<A02.2> ICLwh.e == wh v C1 V / NP
        { [Where] will [it] be held } ?

<B02>   VCLs == C1 V C2 (C3) / NP NP NP
        { [It] will be held [in Kyoto]
          [at the Kyoto international conference hall Japan] } .

<A03>   ICLwh.e == wh V C1 / NP
        { [How much] is [the registration fee] } ?

<B03>   VCLe == C1 V C2 / NP NP
        { [The registration fee] is [$100 US] } .

<A04>   INP
        { [$100] [for attending the conference] } ?

<B04>   FCLe == P
        { Yes } .

<A05>   ICLyn.e == V C1 C2 / NP AP
        { Is [the program of the conference] [available now] } ?

<B05>   VCLs == C1 V (C2) / NP NP
        { [Conference proceedings] will be published [before the conference] } .

<A06>   ICLyn.s == v C1 V (C2) / NP NP
        { Do [you] still accept [submission of papers] } ?

<B06>   VCLc == VCLe but VCLe
        VCLe == C1 V C2 / NP AP
        VCLe == C1 V C2 / NP NP
        { [I] 'm [sorry] } but
        { [the deadline for submitting papers] was [September 20, 1986] } .

<A07.1> VCLe == C1 V / NP
        { [I] see } .
<A07.2> ICLwh.s == wh v C1 V (C2) / NP AP
        { [What] do [I] have to do [to attend the conference] } ?

<B07.1> VCLe == C1 V C2 / NP to-infinitive
        { [You] need [to fill out the registration form] } .
<B07.2> ICLyn.s == v C1 V (C2) C3 / NP NP NP
        { Shall [I] send [you] [one] } ?

<A08.1> FCLc == FLCe , FCLe
        FLCe == P
        FLCe == P
        { Yes} , { please } .
<A08.2> ICLyn.e == CLM V C1 C2 / NP AP
        CLM == but
        { But } { is [it] [free of charge] } ?

<B08>   FCLe = P
        { Yes } .

<A09.1> VCLe == C1 V C2 / NP NP
        { [My address] is [1-2-61, Higashi ward, Osaka city] } .
<A09.2> ICLyn.s == v C1 V (C2) (C3) / NP NP NP
        { Do [I] pay [my registration fee] [to your bank account] } ?

<B09.1> VCLs == C1 V C2 (C3) / NP AP to-infinitive
        { [It] would be [more convenient] [to pay by credit card] } .
<B09.2> ICLyn.e == v C1 V C2 / NP NP
        { Do [you] have [one] } ?
```

```
<A10.1> FCLe == P
        { Yes } .
<A10.2> ICLyn.e == v C1 V / NP
        { Will [any credit card] do } ?

<B10.1> FCLe == P
        { Yes } .
<B10.2> DCLe == V (C1) C2 / NP NP
        { Give [us] [the credit card name and number] } .

<A11.1> VCLe == C1 V C2 / NP NP
        { [It] is [number 234-56789 of American Express] } .
<A11.2> ICLyn.s ==  CLM  v C1 V (C2) / NP NP
        CLM == and
        { And } { will [you] publish [the conference proceedings] } ?

<B11.1> FCLe == P
        { Yes } .
<B11.2> NPCL
        { [In July] } .
<B11.3> ICLyn.s == v C1 V (C2) (C3) / NP NP NP
        { May [we] charge [the registration fee]
          [to your American Express account] } ?

<A12.1> FCLe == P
        { Yes } .
<A12.2> CL   == FCLe and VCLs
        FCLe == P
        VCLs == C1 V (C2) / NP NP
        { Thank you } and { [I] am waiting [for the registration form] } .

<B12>   ICLyn.c == CLM :  ICLyn.s so VCLs
        CLM     == NP
        ICLyn.s == v C1 V (C2) C3 / NP NP NP
        VCLs    == C1 V C2 (C3) / NP NP NP
        { One last item } :
        { could [you] give [us] [your name and telephone number] } so
        { [we] can contact [you] [later] } ?

<A13.1> VCLe == C1 V C2 / NP NP
        { [This] is [ATR] } .
<A13.2> VCLe == C1 V C2 / NP NP
        { [The telephone number] is [06-211-5111] } .

<B13.1> FCLe == P
        { Thank you } .
<B13.2> ICLyn.e == v C1 V C2 / NP NP
        { Do [you] have [any other questions] } .

<A14.1> FCLe == P
        { No } .
<A14.2> FCLe == P
        { Thank you very much } .

<B14>   FCLe == P
        { Goodbye } .

------------------------------------------------------------------------ 03

<A01>   FCLe == P
        { Hello } .

<B01>   ICLyn.e == v C1 V C2 / NP NP
        { May [I] help [you] } ?
```

```
<A02>   ICLyn.s == v C1 V C2 (C3) (C4) / NP NP NP NP
        { Will [you] reserve [a room] [at a hotel]
          [for the participant of the conference] } ?

<B02.1> FCLe == P
        { Yes } .
<B02.2> VCLe == V C1 / NP
        { There are [three hotels with prices running from 6000-10000 yen
                     a night for a single and 9500-60000 for a double] } .
<B02.3> ICLwh.s == wh v C1 V (C2) / NP NP
        { [Which] do [you] prefer [an expensive or less expensive hotel] } .

<A03.1> CL      == FCLe , ICLwh.e
        FCLe    == P
        ICLwh.e == wh V C1 / NP
        { Well } , { [what] are [the names of the hotels] } ?
<A03.2> ICLyn.e == V C1 C2 / NP AP
        { Are [they] [close to the JNR Kyoto station] } ?

<B03>   VCLc == VCLe , VCLe
        VCLe == C1 V C2 / NP NP
        VCLe == C1 V C2 / NP AP
        { [The hotel names] are
          [Kyoto Hotel, Kyoto Prince Hotel, Kyoto Royal Hotel] } ,
        { [they] are [all close to Kyoto station] } .

<A04>   ICLyn.s == v C1 V (C2) / NP ing-clause
        { Are [sightseeing tours] planned [during the conference] } ?

<B04.1> CL   == FCLe VCLs
        FCLe == P
        VCLs == V C1 (C2) (C3) / NP NP NP
        { Yes }
        { there is [a tour schelduled] [for August 5] , [in the afternoon] } .
<B04.2> VCLe == C1 V C2 / NP NP
        { [This tour] will cover
          [the Golden temple, the Heian shrine and Nijo castle] } .
<B04.3> ICLyn.e == V C1 C2 / NP AP
        { Are [you] [interested] } ?

<A05.1> FCLe == P
        { Yes } .
<A05.2> VCLe == C1 V C2 / NP to-infinitive
        { [I] would like [to go with my daughter] } .
<A05.3> VCLs == C1 V (C2) / NP NP
        { [She] won't attend [the conference] } .
<A05.4> ICLyn.s == v C1 V (C2) / NP NP
        { Can [she] participate [in the tour] } ?

<B05>   CL   == FCLe VCLe
        FCLe == P
        VCLe == C1 V / NP
        { Yes } { [she] may } .

<A06>   ICLwh.e == wh V C1 / NP
        { [How much] is [the fee] } ?

<B06>   VCLe == C1 V C2 / NP AP
        { [The tour fee] is [included with the registration fee] } .

<A07>   ICLyn.s == v C1 V C2 (C3) / NP NP NP
        { Will [you] reserve [a twin at Kyoto Royal] [for me and my daughter]} ?

<B07>   ICLwh.s == WH v C1 V (C2) / NP to-infinitive
        { [What dates] would [you] like [to stay] } ?
```

```
<A08>   NPCL
        { [From the evening of August 4] [to the morning of August 8] } .

<B08.1> VCLs == C1 V (C2) (C3) (C4) / NP NP NP NP
        { [We] have confirmed [your reservations]
          [from August 4] [to August 8] } .
<B08.2> ICLs == v C1  CLM  V (C2) C3 / NP NP NP
        CLM  == FCLe
        { Could [you] { please } give [us] [your name and adress] } ?

<A09.1> VCLe == C1 V C2 / NP NP
        { [My name] is [Etsuko Takenaka] } .
<A09.2> VCLe == C1 V C2 / NP NP
        { [My address] is [1-3-15, Minamiyogo-cho, Matsuyama-city, Ehime] } .
<A09.3> VCLe == C1 V C2 / NP NP
        { [My daughter's name] is [Mariko] } .

<B09.1> FCLs == P C1 / NP
        { Thank you [Ms. Takenaka] } .
<B09.2> ICLyn.s == V C1 (C2) (C3) / NP NP NP
        { Is there [anything else we can do] [for you] [today] } ?

<A10>   ICLyn.c == ICLyn.s and ICLyn.s
        ICLyn.s == v C1 V (C2) (C3) / NP NP NP
        ICLyn.s == V C1 (C2) / NP NP
        { Can [I] record [the speeches] [on tapes] } and
        { take [some photos] [in the hall] } ?

<B10>   CL   == FCLe VCLe
        FCLe == P
        VCLe == C1 V / NP
        { Yes } { [you] may } .

<A11.1> FCLe == P
        { Thank you } .
<A11.2> ICLtag.e == CLM C1 V C2 tag / NP NP
        CLM      == and
        { And } { { [you] will take
                    [care of my hotel reservation and tour reservation] } ,
                { won't [you] } } ?

<B11>   CL   == FCLe VCLe
        FCLe == P
        VCLe == C1 V / NP
        { Yes } { [we] will } .

------------------------------------------------------------------------ 04

<A01>   FCLe == P
        { Hello } .

<B01>   ICLyn.e == v C1 V C2 / NP NP
        { May [I] assist [you] } ?

<A02>   VCLe == C1 V C2 / NP to-infinitive
        { [I] would like [to know how I can get to the Kyoto
          international conference hall from Kyoto station] } .

<B02.1> FCLe == P
        { OK } .
<B02.2> DCLs == C1 V C2 (C3) (C4) / NP NP NP NP
        { [You] ride [the subway] [from Kyoto station] [to Kitaoji station] } .
<B02.3> DCLc == DCLe and DCLe
        DCLe == V C1 / NP
        DCLe == V C1 / NP
        { Exit [the station] } and { take [the city bus number 5 or a taxi] } .
```

```
<A03.1> FCLe == P
        { Thank you } .
<A03.2> ICLwh.c == ICLwh.e (if VCLe)
        ICLwh.e == WH v C1 V / NP
        VCLe    == C1 V C2 / NP NP
        { [How long] will [it] take { if / [I] take [a taxi] } } ?

<B03>   NPCL
        { [About ten minutes] } .

<A04>   ICLyn.c == ICLyn.e (if VCLe)
        ICLyn.e == v C1 V C2 / NP NP
        VCLe    == C1 V C2 / NP NP
        { Will [it] take [longer]
          { if / [I] ride [the subway and a bus instead of a taxi] } } ?

<B04.1> VCLe == C1 V C2 / ing-clause NP
        { [Taking the subway and then taking a taxi] is [the fastest way] } .
<B04.2> VCLe == C1 V C2 / ing-clause NP
        { [Riding the bus] would take [about 20 minutes longer] } .

<A05>   FCLe == P
        { Thank you } .

<B05>   ICLyn.s == V C1 C2 (C3) / NP NP NP
        { Is [that] [all] [for today] } ?

<A06.1> FCLc == FCLe , FCLe
        FCLe == P
        FCLe == P
        { Yes } , { thank you } .
<A06.2> FCLe == P
        { Good-bye } .

<B06>   FCLe == P
        { Goodbye } .

---------------------------------------------------------------------------- 05

<A01.1> FCLe == P
        { Hello } .
<A01.2> ICLyn.e == V C1 C2 / NP NP
        { Is [it] [the first international conference of interpreting telephony
          secretariat] } ?

<B01>   CL      == FCLe VCLe ICLwh.s
        FCLe    == P
        VCLe    == C1 V / NP
        ICLwh.s == wh v C1 V (C2) (C3) / NP NP NP NP
        { Yes } { [it] is } , { [what] can [we] do [for you] [today] } ?

<A02.1> VCLe == C1 V C2 / NP to-infinitive
        { [I] 'd like [to participate in the conference] } .
<A02.2> ICLwh.s == WH V (C1) / to-infinitive
        { [What kind of procedures] are needed [to participate in] } ?

<B02.1> VCLe == CLM C1 V C2 / NP to-infinitive
        CLM == first
        { First } { [you] need [to register] } .
<B02.2> VCLe == C1 V C2 / NP NP
        { [We] need [your name telephone number credit card name and number] } .
<B02.3> VCLe == C1 V C2 / NP to-infinitive
        { [You] also need
          [to pay the registation fee, which is 100 dollars (US)] } .
```

```
<A03>   ICLyn.c == VCLe , so ICLyn.s
        VCLe    == C1 V C2 / NP NP
        ICLyn.s == v C1 V (C2) / NP NP
        { [I] don't have [any credit card] } , so
        { may [I] pay [by a money order] } ?

<B03.1> VCLs == C1 V (C2) / NP NP
        { [You] may pay [by money order or by cashier's check] } .
<B03.2> VCLe == C1 V / NP
        { [Either one] will do } .

<A04.1> VCLe == C1 V / NP
        { [I] see } .
<A04.2> VCLs == C1 V (C2) / NP NP
        { [I] will pay [by money order] } .
<A04.3> VCLe == C1 V C2 / NP NP
        { [My name] is [Mikiko Matsumoto] } .
<A04.4> VCLe == C1 V C2 / NP NP
        { [Telephone number] is [06-372-3551] } .
<A04.5> ICLwh.e == wh V C1 / NP
        { [When] is [the deadline for the resistration payment] } ?

<B04.1> FCLs == P C1 / NP
        { Thank you very much [Ms. Matsumoto] } .
<B04.2> VCLe == C1 V C2 / NP NP
        { [The deadline for the registration fee] is [June 30] } .
<B04.3> ICLyn.e == v C1 V C2 / NP NP
        { Do [you] have [any other questions] } ?

<A05.1> FCLe == P
        { OK } .
<A05.2> VCLs == C1 V (C2) / NP NP
        { [I] will pay [the day] } .
<A05.3> VCLe == C1 V C2 / NP NP
        { [I] have [one more question] } .
<A05.4> VCLe C1 V C2 / NP NP
        { [I] am [a stranger around Kyoto] } .
<A05.5> ICLyn.s == v C1 V (C2) C3 / NP NP wh-clause
        { Could [you] tell [me]
          [how long it takes by car from station to the Kyoto
           international conference hall] } ?

<B05>   VCLs == C1 V C2 (C3) / NP NP ing-clause
        { [It] would take [anywhere from 30 to 40 minutes]
          [depending on the traffic] } .

<A06.1> VCLe == C1 V / NP
        { [I] see } .
<A06.2> FCLe == P
        { Thank you very much } .

<B06>   ICLyn.s == V C1 (C2) (C3) (C4) / NP that-clause NP NP
        { Is there [anything else] [that we can do] [for you] [today] } ?

<A07.1> FCLe == P
        { No } .
<A07.2> FCLe == P
        { Thank you very much } .

<B07>   FCLe == P
        { Good day } .
```

---------- END