RULE TYPES IN A SYSTEMIC FUNCTIONAL GRAMMAR: AN XML DEFINITION OF THE CARDIFF LEXICOGRAMMAR GENERATOR[*]

Víctor M. CASTEL
(Consejo Nacional de Investigaciones Científicas y Técnicas / Universidad Nacional de Cuyo)

ABSTRACT: *The objective of this paper is to provide an XML definition of the structure of the Cardiff Lexicogrammar Generator (CLGG) rules so that grammar development is facilitated and enhanced. At present, given its inherent intricacy and high degree of delicacy, only Robin Fawcett and Gordon Tucker can "safely" manipulate versions of CLGG (Micro, Mini, Midi, etc.). One crucial reason for this state of affairs, though not, of course, the only one, is the significant complexity of rule writing. The paper then addresses the problem of constructing an XML schema to account for the well-formedness and validity of CLGG rules.*

KEY-WORDS: *Cardiff Grammar Generator XML schema, Cardiff Grammar Generator rule typology, Text generation.*

1. Introduction

The Cardiff Lexicogrammar Generator, GENESYS (henceforward, CLGG), in its computationally implemented Mini version, is a set of rules capable of generating linguistic representations like the structure of Figure 1: (Fawcett et al. 1993, Fawcett 2000, 2004a; Castel 2006a, b)

```
Σ
    [entity, spoken, consultative, situation, congruent_situation, independent,
    information, giver, negative, unmarked_negative, present_trp,
    validity_unassessed_present, unmodulated_present, no_pastness_from_trp, action,
    one_role_process, agent_only, simple_agent_only, activeness_ago, working,
    period_marked, agent_only_unmarked, a_subject_theme_unmarked, outsider_sth,
    count_sth, singular_sth, simple_singular_sth, time_position_unspecified,
    not_co_ordinated_with_a_previous_situation, simplex_situation, dominant,
    strongly_dominant, no_contrastive_newness_sit]
    Cl
        St
          ||
        S/Ag
          [entity, spoken, consultative, thing, congruent_thing, stereotypical_thing,
          outsider, cultural_classification, no_ad_hoc_description, physical_thing,
          physical_thing_specified, living_thing, creature, human_cr, whole_human,
          human_specified_by_stage_and_gender, adult, man_c, count_cc,
          singular_cc, particularized_singular, sing_not_selected_from_by_quantity,
          recoverable_cc, not_co_ordinated_with_a_previous_thing, simplex_thing]
          ngp
              dd
                  the
              h
                  man
        O/PdX
          is+n’t
        M
          work+ing
        MN
          MT
        K
          1+
        E
          ||
```

OUTPUT STRING: || the man isn’t working MT 1+ ||

Key: Σ = variable ranging over genre elements; Cl = Clause; St = Starter; S = Subject; Ag = Agent; ngp = nominal group; dd = deictic determiner; h = head; O = Operator; PdX = Period Auxiliary; M = Main Verb; MN = Mood-bearing Tonic; K = Key; E = Ender.

Figure 1: Example of linguistic representation generated by CLGG.

CLGG rules are organized into two components: the semantic component and the form component. The semantic component contains System Network Rules (SNRs), and Same Pass Preference Resetting Rules (SPRs). The form component contains Realization Rules (RRs) proper and Graphological Rules (GRs).

SNRs and SPRs jointly construct selection expressions, i.e. sets of semantic features. Cf. the expressions in square brackets dominated by “Σ” and “S/Ag” in Figure 1. The task of RRs is to define form representations, i.e. tree structures which account for syntactic, lexical, and punctuational (or intonational) properties of linguistic units realizing a given selection expression. Cf. for example the syntactic unit “ngp” which realizes the element “S/Ag” which is associated with an appropriate selection expression; notice that the unit “ngp” dominates the elements “dd” and “h” which in turn dominate the lexical items “the” and “man”, respectively. The task of GRs is to take care of graphological realization.

At a very abstract level, CLGG rules involved in the generation of linguistic representations like the instance illustrated in Figure 1 are all implications which can be represented as in (1i), read as in (1ii), and interpreted as in (1iii):

(1i) $p \Rightarrow q$,

(1ii) if p, then q,

(1iii) if p is true, then carry out q,

where p and q are variables ranging over conditions and consequences, respectively. Condition p can be a single semantic feature, or a disjunction of semantic features, or a conjunction of semantic features. Consequence q can be a(n) (conjunction of) operation(s), and/or a(n) (conjunction of) implication(s) like (1i).

The purpose of this paper is not to establish how the truth value of p is determined nor to specify how *carry out q* is computationally perfomed.[i] The objective is rather more modest but perhaps more useful for grammar development, namely: to provide an XML definition of the structure of CLGG rules so that grammar writing is facilitated and enhanced. At present, given its inherent intricacy and high degree of delicacy, only Robin Fawcett and Gordon Tucker can "safely" manipulate versions of CLGG (Micro, Mini, Midi, etc.). One crucial reason for this state of affairs, though not, of course, the only one, is the complexity of rule writing as illustrated by the rule sample (1)-(20) in §2. Thus, the paper addresses neither CLGG rule function nor rule interaction. It does not address the problem of output construction either. It simply addresses the problem of constructing an XML schema to account for the well-formedness and validity of CLLG rules.[ii]

2. CLGG Rule Sample

The rule types presented in this paper have been defined inductively from the appropriate CLGG text files as specified in Fawcett (2004a). Here follows an extract from this source which highlights instances of the different rule classes:[iii]

System Network Rules[iv]

(1) sn1: entity --> MODE And ENTITY_TYPE.

(2) sn2: MODE --> 70% spoken (0.1) Or 30% written (0.2).

(3) sn33: (giver Or new_content_seeker Or

proposal_for_action_by_addressee) --> POLARITY.

(4) sn50: ((relational Or action Or mental Or environmental Or influential) And (information Or proposal_for_action)) --> 10% period_marked (10.1) Or 90% not_period_marked.

(5) sn62: (time_position_presented And (proposal_for_action Or (future_trp And (no_pastness_from_trp Or past_from_trp)))) --> 45% time_position_recoverable_from_present_unit (sp20_4, 20.4) Or 45% deictic_future_time_position (sp20_4, 20.6) Or 10% recoverable_future_time_position (sp20_4, 20.61) Or 0% others.

(6) sn71: (following_situation_is_in_separate_information_unit And spoken) --> 70% unmarked_co_ordination_int_sit (99.7) Or 5% co_ordination_with_dominance_int_sit (99.71) Or 15% co_ordination_with_reservations_int_sit (99.72).

(7) sn74: another_co_ordinated_situation --> (99% one_following_situation (19.1) Or 1% two_or_more_following_situations (19.2)) And (95% another_additive_situation (19.81) Or 5% another_alternative_situation (19.61)).

(8) sn274: (agent_only_unmarked Or affected_only_unmarked Or (agent_unmarked And agent_subject_theme) Or (agent_unmarked And affected_covert) Or (affected_unmarked And affected_subject_theme) Or (agent_covert And affected_unmarked) Or at_carrier_unmarked) --> a_subject_theme_unmarked.

Same Pass Preference Resetting Rules

(9) sp1_1: congruent_situation --> written --> for same_pass prefer sn12 [99.98% information, 0.02% proposal_for_action] And sn14 [99.9% giver, 0.1% seeker & 0% confirmation_seeker].

(10) sp60: congruent_thing --> fills At And {on_previous_pass} no_ad_hoc_description --> apply carrier_attribute_agreement_subrule.

Same Pass Preference Resetting Subrules

(11) carrier_attribute_agreement_subrule --> {on_previous_pass} (interactant Or human_tc Or human_ssth Or whole_human Or name_of_person) --> for same_pass prefer human_tc And human_ssth And BASIC_TYPICALLY_HUMAN_PREF_BLOCK And TYPICALLY_HUMAN_CC_PREF_BLOCK, {on_previous_pass} (non_human_tc Or non_human_cr Or non_human_ssth) --> for same_pass prefer non_human_tc And non_human_cr And

non_human_ssth, {on_previous_pass} (singular_performer or singular_addressee Or singular_tc Or singular_ssth Or singular_loc_rth Or singular_pos_rth Or singular_cc Or pl_with_singular_quantity) --> for same_pass prefer singular_performer And singular_addressee And singular_tc And singular_ssth And singular_loc_rth And singular_pos_rth And singular_cc And pl_with_singular_quantity, {on_previous_pass} (plural_performer Or plural_addressee Or plural_tc Or plural_ssth Or plural_loc_rth Or plural_pos_rth Or plural_cc Or pl_with_plural_quantity) --> for same_pass prefer plural_performer And plural_addressee And plural_tc And plural_ssth And plural_loc_rth And plural_pos_rth And plural_cc And pl_with_plural_quantity, {on_previous_pass} (mass_tc Or mass_ssth Or mass_loc_rth Or mass_pos_rth Or mass_cc) --> for same_pass prefer mass_tc And mass_ssth And mass_loc_rth And mass_pos_rth And mass_cc, {on_previous_pass} (performer or addressee) --> for same_pass prefer sn78 [0.1% interactant, 99.9% outsider], {on_previous_pass} token_classification --> for same_pass prefer sn78 [0.1% interactant, 99.9% outsider] And sn136 [0.1% recoverable_thing, 94.9% cultural_classification, 5% name_of_thing].

Realization Rules

(12) 0.2: written --> for any_re_entry prefer written.

(13) 1.1: congruent_situation --> Cl, S @ 33, spoken And not_co_ordinated_with_a_previous_situation And fills Z --> St @ 3, St > ||, not at_being --> M @ 100, (information And (at_being Or unmarked_passive Or future_trp Or validity_assessed Or retrospective_from_trp Or period_marked Or negative Or (seeker And not ncs_theme_on_a_subject_theme_sought_r) Or confirmation_seeker Or contrastive_newness_on_polarity)) --> apply Operator_placement_subrule, (information And (negative Or confirmation_seeker Or (seeker And not ncs_theme_on_a_subject_theme_sought_r) Or contrastive_newness_on_polarity)) --> apply do_support_subrule, (simplex_situation Or final_co_ordinated_situation) --> E @ 250, apply Ender_subrule, (spoken And (simplex_situation Or final_co_ordinated_situation)) --> no_contrastive_newness_sit --> MN @ 200, MN > MT, K @ 201.

(14) 1.33: request --> O @ 31, for S prefer thing And congruent_thing And stereotypical_thing And interactant And addressee, for S re_enter_at entity.

(15) 10.1: period_marked --> (information And not (future_trp Or validity_assessed Or retrospective_from_trp Or past_from_trp)) --> PdX by O, apply finite_be_forms, (future_trp Or validity_assessed Or retrospective_from_trp Or past_from_trp Or proposal_for_action) --> PdX @ 96, PdX > be, unmarked_passive --> PaX >+ +ing.

Realization Subrules

(16) Ender_subrule --> spoken --> E > ||, written --> unmarked_mood_wr --> E > ., (seeker Or confirmation_seeker Or request) --> E > ?, (fun_mood_wr Or enthusiastic_mood_wr) --> E > !.

Block Preference Subrules

(17) BASIC_SING_OUTSIDER_PREF_BLOCK --> singular_tc And singular_loc_rth And singular_pos_rth And singular_ssth And sn157 [99.999% singular_cc, 0.001% plural_cc, 0% class_of_count_thing].

Graphological Rules

(18) gr5: +ing --> e --> apply ing_subrule_1, Else apply ing_subrule_4.

Graphological Subrules

(19) ing_subrule_1 --> be --> delete +, ing sfx_is_written_as ing, Else apply ing_subrule_1_1.

(20) ing_subrule_4 --> VC --> apply ing_subrule_4_1, Else delete +, ing sfx_is_written_as ing.

As this rule sample suggests, CLGG is a text which has been written using a special vocabulary and a special syntax. Being a structured text, its underlying “grammar” can be captured in terms of an XML schema as shown in §§3-4.[v]

3. CLGG Rule Classes

CLGG rules can be classified in accordance with the following diagram:[vi]

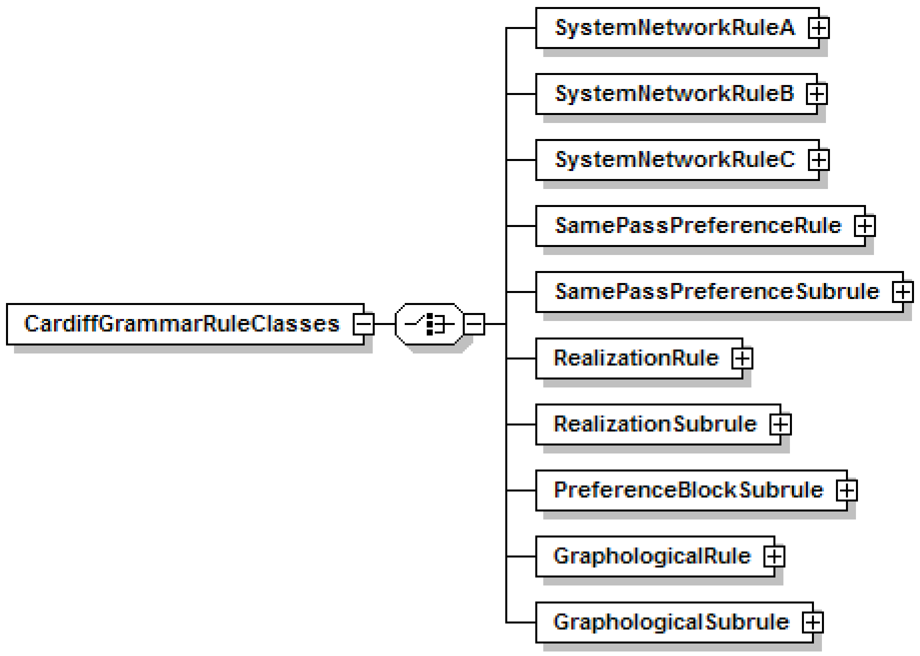


Figure 2: CLGG rule classes.

The Choice Compositor in the diagram is used to indicate that any given CLGG rule belongs in one of the rule options. For examples of each class, see (1)-(20) above, where (1) is a SystemNetworkRuleA, (2) a SystemNetworkRuleB, and (3) a SystemNetworkRuleC.

All the rule classes share the property of being defined by the complex type RuleType, i.e. a sequence of a RuleCode element, an Implication element and a RuleEnder element.

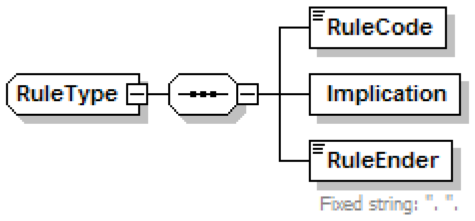


Figure 3: Rule elements.

The RuleCode is a regular expression which varies depending on the rule class. If it is a System Network Rule, the RuleCode is defined by the regular expression "sn\p{N}+[:]\p{Zs}". If it is a Same Pass Preference Resetting Rule, the RuleCode is defined by the regular expression "sp\p{N}+([_]\p{N}+)?[:]\p{Zs}". If it is a Realization Rule, the RuleCode is defined by the regular expression

“\p{N}+([.]\p{N}+)?[:]\p{Zs}”. If it is a Graphological Rule, the RuleCode is defined by the regular expression “gr\p{N}+[:]\p{Zs}”. In (21), the RuleCode is underlined, the Implication is in italics, and the RuleEnder is in bold.

(21) <u>sn2:</u> *MODE --> 70% spoken (0.1) Or 30% written (0.2)***.**

The Implication element is defined by the complex type ImplicationType, which is a sequence of a Condition, the ImplicationOperator, a Consequence1, and an optional sequence of the ElseOperator and a Consequence2.

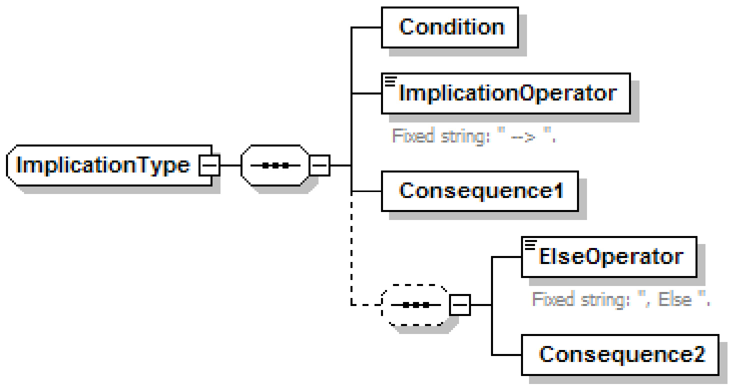


Figure 4: The structure of implications.

Example (22) illustrates the Condition element with the underlined expression, the Consequence1 element with the expression in italics, and the Consequence2 element with the expression in bold.

(22) gr5: <u>+ing</u> --> *e --> apply ing_subrule_1*, Else **apply ing_subrule_4**.

3.1. Conditions

The Condition element varies depending on the rule class, but all the variations are defined as restrictions on the complex type ConditionType:

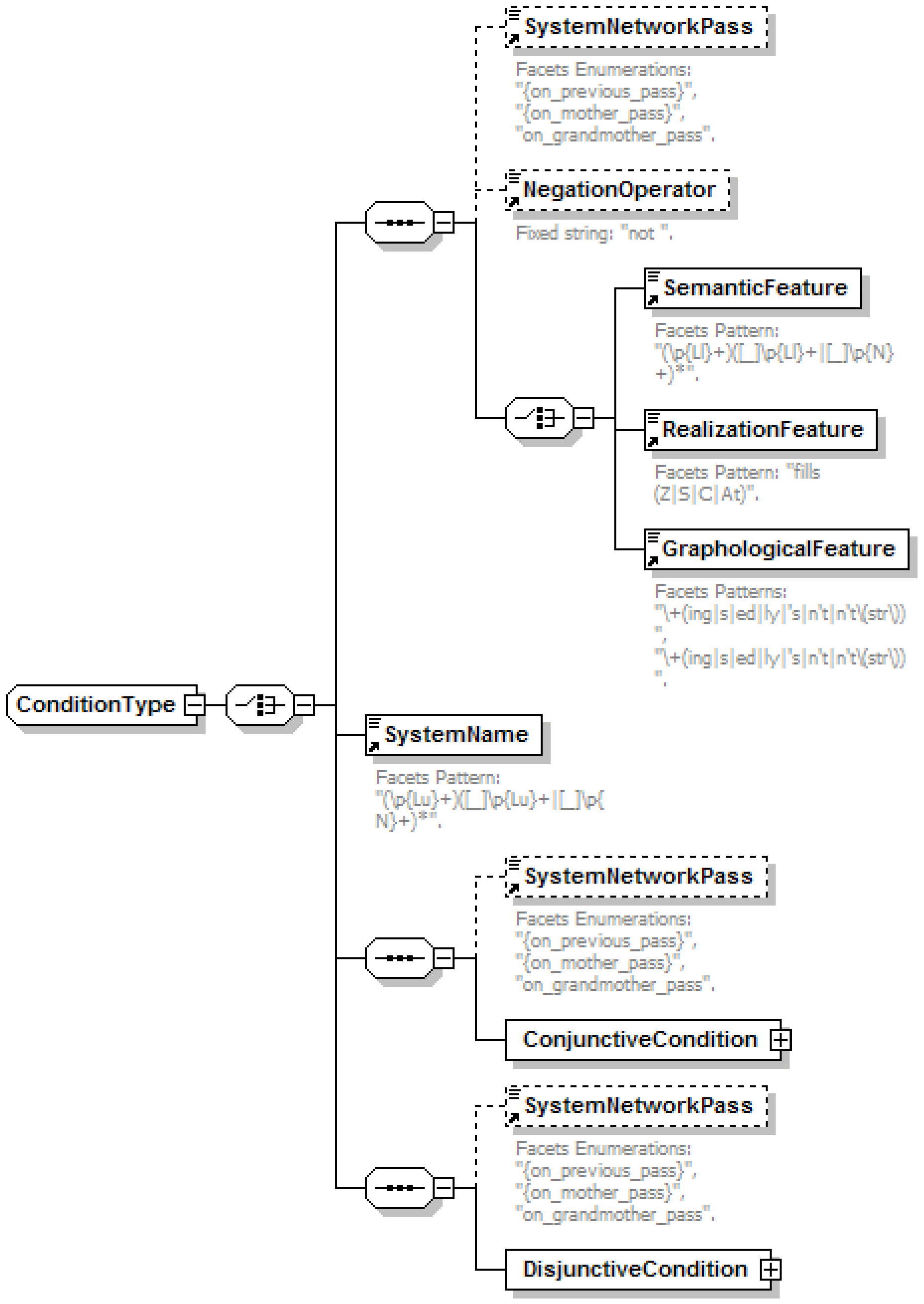

SystemNetworkPass
Facets Enumerations: "{on_previous_pass}", "{on_mother_pass}", "on_grandmother_pass".
NegationOperator
Fixed string: "not ".
SemanticFeature
Facets Pattern: "(\p{Ll}+)([_]\p{Ll}+|[_]\p{N}+)*".
RealizationFeature
Facets Pattern: "fills (Z|S|C|At)".
GraphologicalFeature
Facets Patterns: "\+(ing|s|ed|ly|'s|n't|n't\(str\))", "\+(ing|s|ed|ly|'s|n't|n't\(str\))".
ConditionType
SystemName
Facets Pattern: "(\p{Lu}+)([_]\p{Lu}+|[_]\p{N}+)*".
SystemNetworkPass
Facets Enumerations: "{on_previous_pass}", "{on_mother_pass}", "on_grandmother_pass".
ConjunctiveCondition
SystemNetworkPass
Facets Enumerations: "{on_previous_pass}", "{on_mother_pass}", "on_grandmother_pass".
DisjunctiveCondition

Figure 5: Condition choices.

The complex type ConditionType defines the CLGG potential for defining rule conditions. Notice that according to this definition a Condition can be (1) a sequence of an optional SystemNetworkPass, an optional NegationOperator, and a SemanticFeature or a

RealizationFeature or a GraphologicalFeature, (2) a SystemName, (3) a sequence of an optional SystemNetworkPass and a ConjunctiveCondition, or (4) a sequence of an optional SystemNetworkPass and a DisjunctiveCondition. The expressions in italics in (23) illustrate the Condition element:

(23a) sn4: *SPOKEN_TENOR* --> 10% formal (0.32) Or 70% consultative (0.33) Or 20% casual (0.34).

(23b) sn8: *situation* --> 100% congruent_situation (sp1_1, 1.1) Or 0% reified_situation.

(23c) sn21: *(written & giver)* --> 0.1% fun_mood_wr Or 99.9% unmarked_mood_wr.

(23d) sn33: *(giver Or new_content_seeker Or proposal_for_action_by_addressee)* --> POLARITY.

(23e) *am_are_subrule* --> *not (polarity_seeker Or confirmation_seeker)* --> O > am, *(polarity_seeker Or (confirmation_seeker And (formal Or very_formal) Or (consultative And written)))* --> *spoken* --> O > am(str), *written* --> O > am, *(confirmation_seeker And (casual Or (consultative And spoken)))* --> O > are.

(23f) sp60: *congruent_thing* --> *fills At And {on_previous_pass} no_ad_hoc_descriptio*n --> apply carrier_attribute_agreement_subrule.

The character strings realizing the elements SystemNetworkPass, NegationOperator, SemanticFeature, RealizationFeature, GraphologicalFeature, and SystemName are as specified in the corresponding annotations.

A DisjunctiveCondition is defined by the complex type DisjunctiveConditionType:

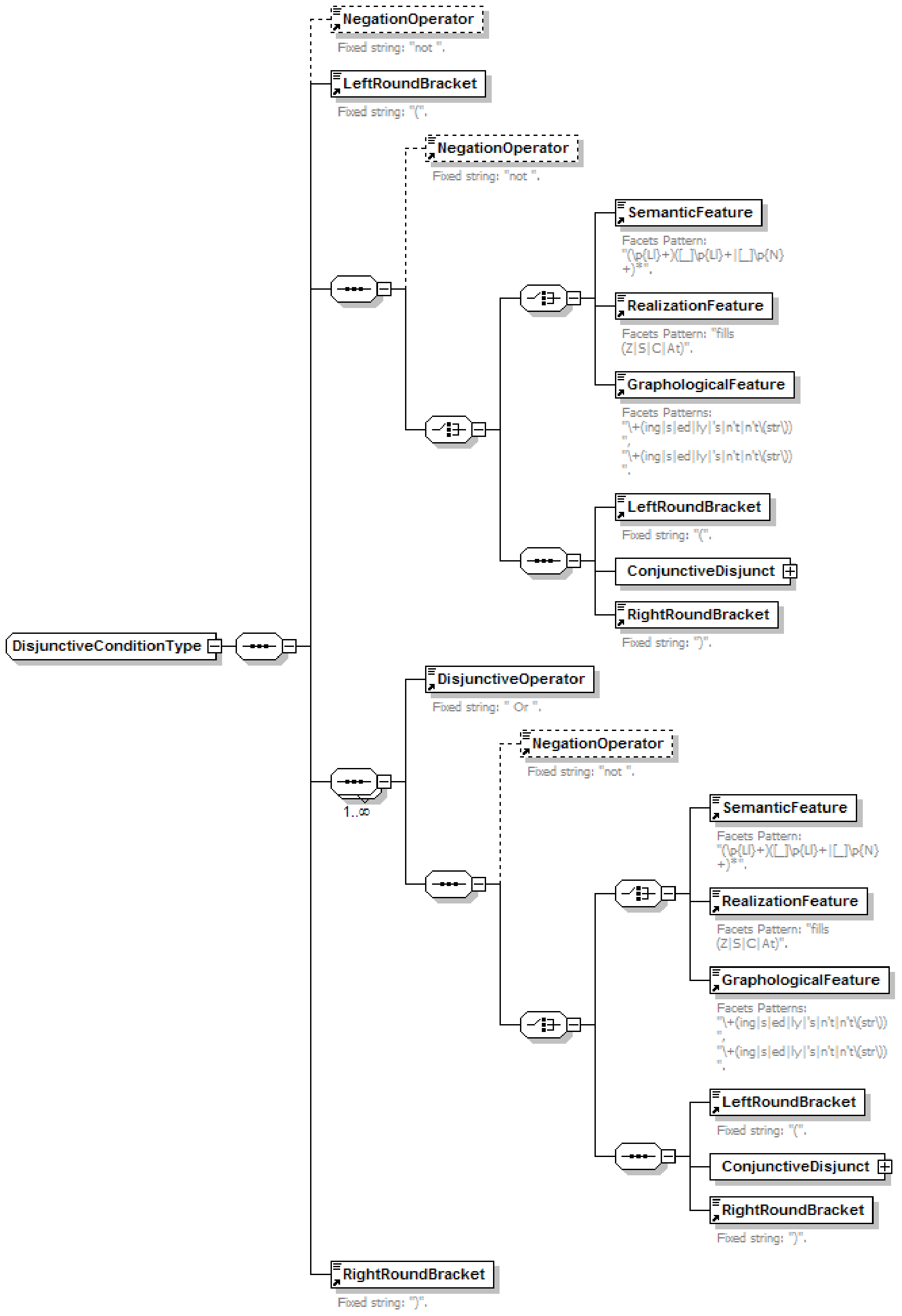

NegationOperator
Fixed string: "not ".
LeftRoundBracket
Fixed string: "(".
NegationOperator
Fixed string: "not ".
SemanticFeature
Facets Pattern: "(\p{Ll}+)([_]\p{Ll}+|[_]\p{N}+)*".
RealizationFeature
Facets Pattern: "fills (Z|S|C|At)".
GraphologicalFeature
Facets Patterns: "\+(ing|s|ed|ly|'s|n't|n't\(str\))", "\+(ing|s|ed|ly|'s|n't|n't\(str\))".
LeftRoundBracket
Fixed string: "(".
ConjunctiveDisjunct
RightRoundBracket
Fixed string: ")".
DisjunctiveConditionType
DisjunctiveOperator
Fixed string: " Or ".
1..∞
NegationOperator
Fixed string: "not ".
SemanticFeature
Facets Pattern: "(\p{Ll}+)([_]\p{Ll}+|[_]\p{N}+)*".
RealizationFeature
Facets Pattern: "fills (Z|S|C|At)".
GraphologicalFeature
Facets Patterns: "\+(ing|s|ed|ly|'s|n't|n't\(str\))", "\+(ing|s|ed|ly|'s|n't|n't\(str\))".
LeftRoundBracket
Fixed string: "(".
ConjunctiveDisjunct
RightRoundBracket
Fixed string: ")".
RightRoundBracket
Fixed string: ")".

Figure 6: The structure of disjunctive conditions.

A DisjunctiveCondition, then, is a sequence of an optional NegationOperator, a LeftRoundBracket, a SemanticFeature or a RealizationFeature or a GraphologicalFeature or a sequence of a LeftRoundBracket, a ConjunctiveDisjunct (of type ConjunctiveConditionType; see below) and a RightRoundBracket, a DisjunctiveOperator, and a sequence of an optional NegationOperator, a SemanticFeature or a RealizationFeature or a GraphologicalFeature or a sequence of a LeftRoundBracket, a ConjunctiveDisjunct (of type ConjunctiveConditionType; see below) and a RightRoundBracket, and a RightRoundBracket. For an example, cf. the expression in italics in (23d).

A ConjunctiveCondition is defined by the complex type ConjunctiveConditionType:

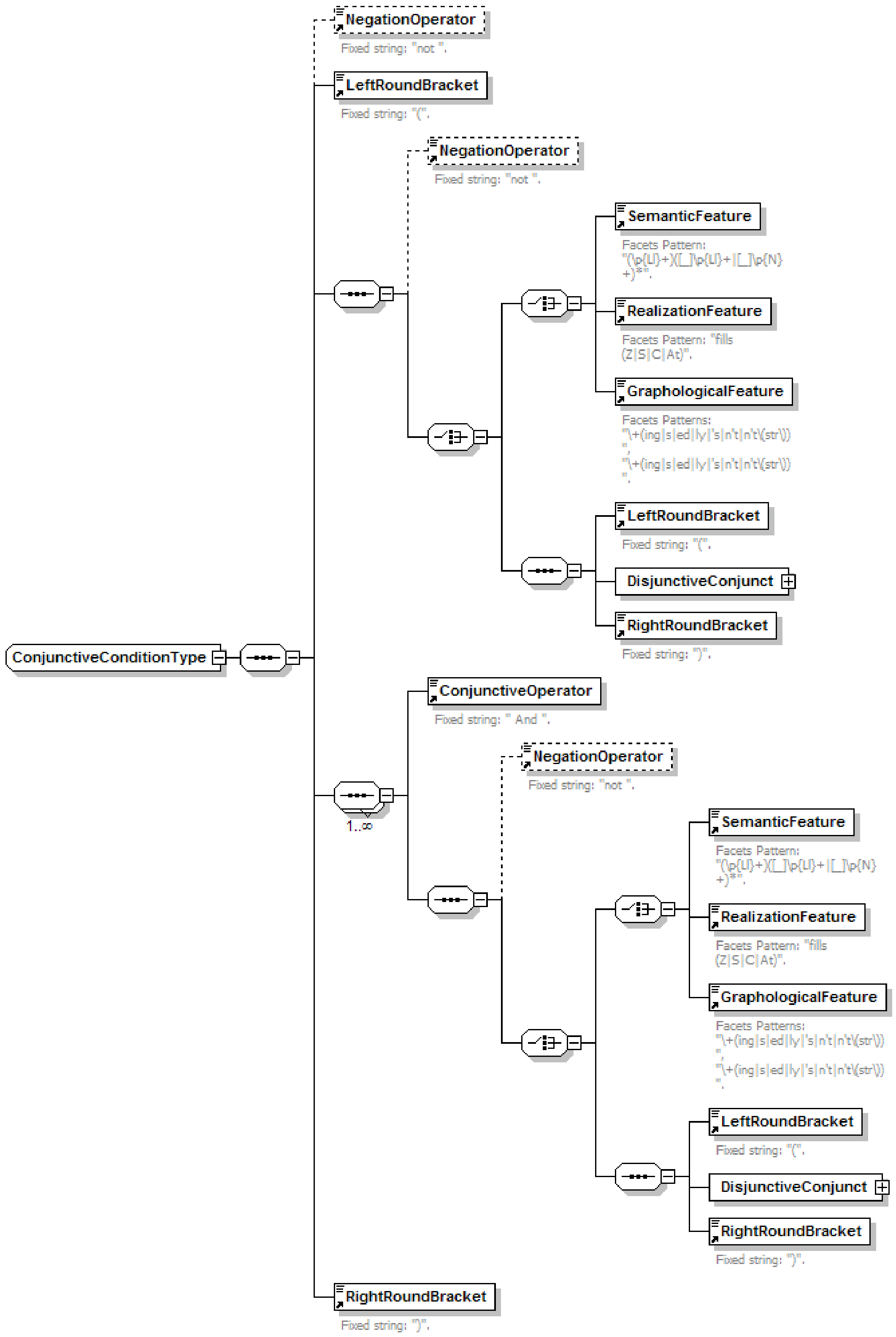

NegationOperator
Fixed string: "not ".
LeftRoundBracket
Fixed string: "(".
NegationOperator
Fixed string: "not ".
SemanticFeature
Facets Pattern: "(\p{Ll}+)([_]\p{Ll}+|[_]\p{N}+)*".
RealizationFeature
Facets Pattern: "fills (Z|S|C|At)".
GraphologicalFeature
Facets Patterns: "\+(ing|s|ed|ly|'s|n't|n't\(str\))", "\+(ing|s|ed|ly|'s|n't|n't\(str\))".
LeftRoundBracket
Fixed string: "(".
DisjunctiveConjunct
RightRoundBracket
Fixed string: ")".
ConjunctiveConditionType
ConjunctiveOperator
Fixed string: " And ".
1..∞
NegationOperator
Fixed string: "not ".
SemanticFeature
Facets Pattern: "(\p{Ll}+)([_]\p{Ll}+|[_]\p{N}+)*".
RealizationFeature
Facets Pattern: "fills (Z|S|C|At)".
GraphologicalFeature
Facets Patterns: "\+(ing|s|ed|ly|'s|n't|n't\(str\))", "\+(ing|s|ed|ly|'s|n't|n't\(str\))".
LeftRoundBracket
Fixed string: "(".
DisjunctiveConjunct
RightRoundBracket
Fixed string: ")".
RightRoundBracket
Fixed string: ")".

Figure 7: The structure of conjunctive conditions.

A ConjunctiveCondition, then, is a sequence of an optional NegationOperator, a LeftRoundBracket, a SemanticFeature or a RealizationFeature or a GraphologicalFeature or a sequence of a LeftRoundBracket, a DisjunctiveConjunct (of type DisjunctiveConditionType; see above) and a RightRoundBracket, a ConjunctiveOperator, and a sequence of an optional NegationOperator, a SemanticFeature or a RealizationFeature or a GraphologicalFeature or a sequence of a LeftRoundBracket, a DisjunctiveConjunct (of type DisjunctiveConditionType; see above) and a RightRoundBracket, and a RightRoundBracket. For an example, cf. the expression in italics in (23c).

Each rule class picks up from ConditionType, by restriction, a specific subset of possible conditions. Thus, System Network Rule conditions do not allow for the following elements: SystemNetworkPass, NegationOperator, RealizationFeature, and GraphologicalFeature (and this is true also within the elements ConjunctiveCondition and DisjunctiveCondition). Realization Rules do not allow for the elements GraphologicalFeature and SystemName. Graphological Rules do not allow for the elements SystemNetworkPass, SemanticFeature, RealizationFeature and SystemName.

3.2. Consequences

Both Consequence1 and Consequence2 are of type ConsequenceType:

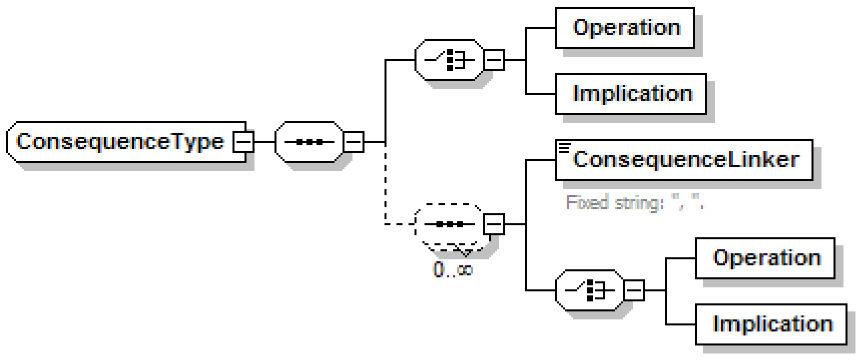


Figure 8: The structure of consequences.

Thus, these elements are sequences made up of an Operation or an Implication (see ImplicationType above), a ConsequenceLinker, and an Operation or an Implication (see ImplicationType above).

The complex type OperationType defines the Operation element:

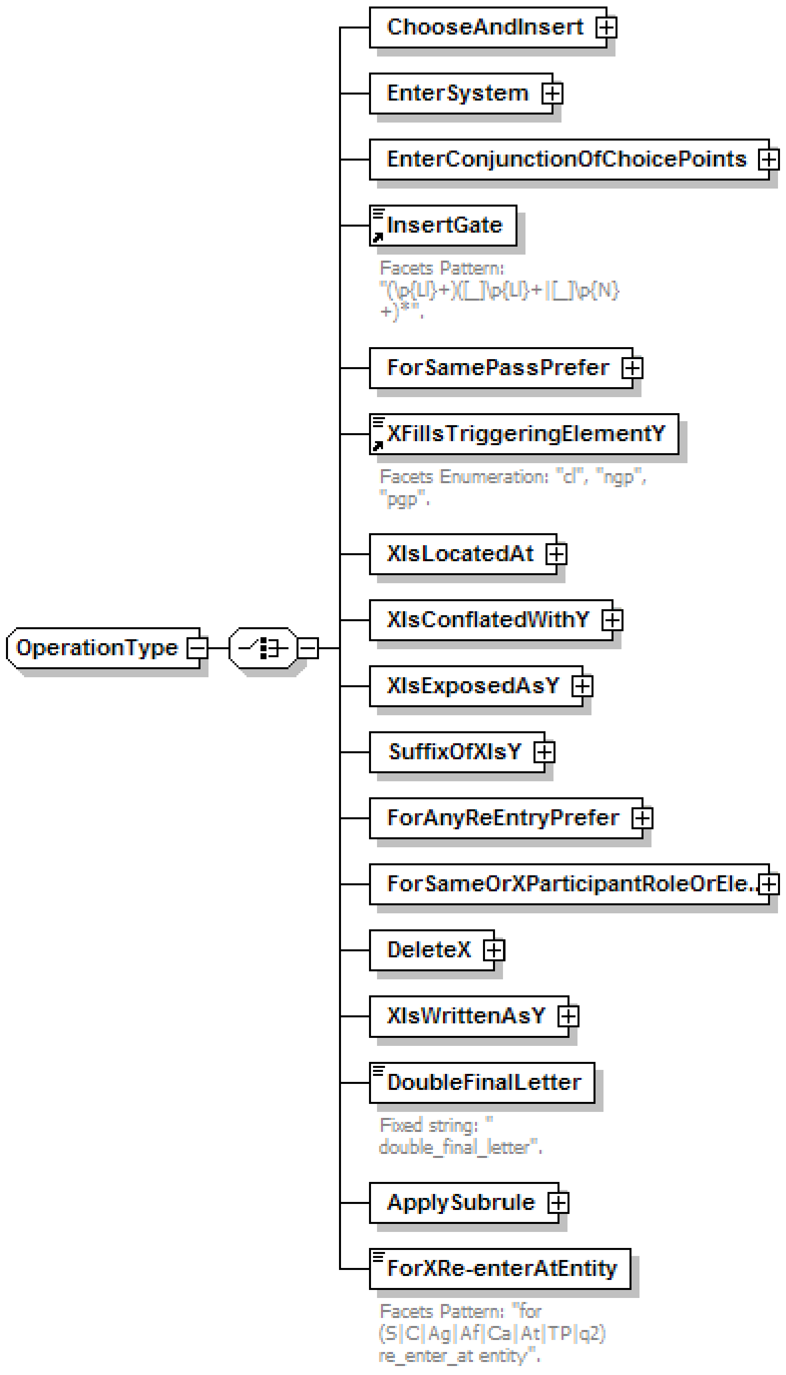

ChooseAndInsert
EnterSystem
EnterConjunctionOfChoicePoints
InsertGate
Facets Pattern: "(\p{Ll}+)([_]\p{Ll}+|[_]\p{N}+)*".
ForSamePassPrefer
XFillsTriggeringElementY
Facets Enumeration: "cl", "ngp", "pgp".
XIsLocatedAt
OperationType
XIsConflatedWithY
XIsExposedAsY
SuffixOfXIsY
ForAnyReEntryPrefer
ForSameOrXParticipantRoleOrEle...
DeleteX
XIsWrittenAsY
DoubleFinalLetter
Fixed string: "double_final_letter".
ApplySubrule
ForXRe-enterAtEntity
Facets Pattern: "for (S|C|Ag|Af|Ca|At|TP|q2) re_enter_at entity".

Figure 9: Operation classes.

ChooseAndInsert, EnterSystem, EnterConjunctionOfChoicePoints, and InsertGate are System Network Rule operations.

The Operation ChooseAndInsert is defined as a sequence of a sequence of an AssociatedProbability element, a SemanticFeature element and an optional AssociatedRules element, followed by a sequence of a DisjunctiveOperator element and a sequence of an AssociatedProbability element, a SemanticFeature element and an optional AssociatedRules element (cf. the expression in italics in (24)):

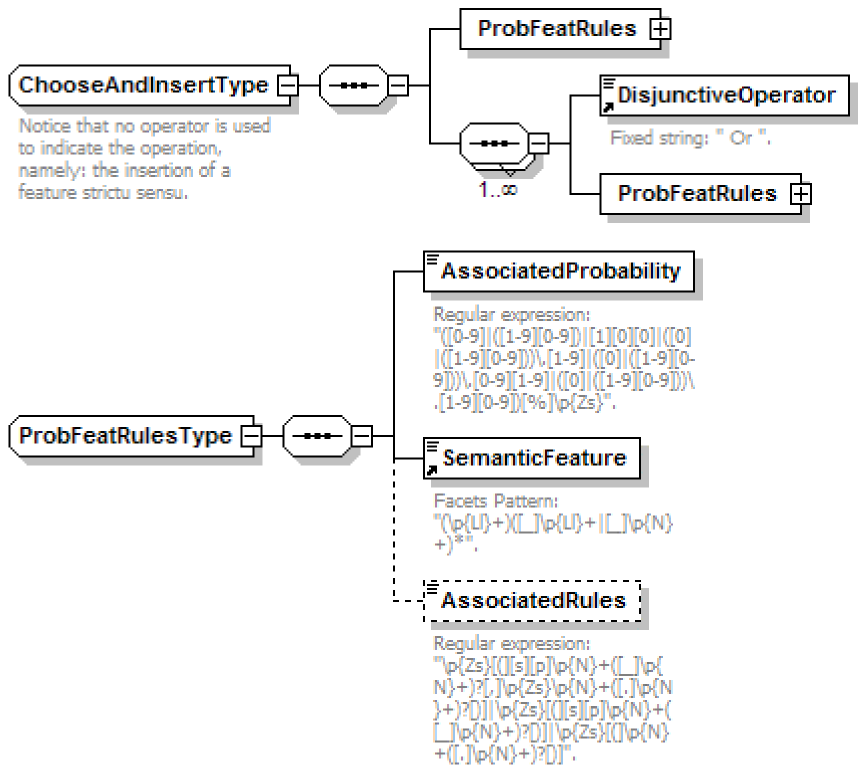

ProbFeatRules
ChooseAndInsertType
Notice that no operator is used to indicate the operation, namely: the insertion of a feature strictu sensu.
DisjunctiveOperator
Fixed string: " Or ".
1..∞
ProbFeatRules
AssociatedProbability
Regular expression: "([0-9]|([1-9][0-9])|[1][0][0]|([0]|([1-9][0-9]))\.[1-9]|([0]|([1-9][0-9]))\.[0-9][1-9]|([0]|([1-9][0-9]))\.[1-9][0-9])[%]\p{Zs}".
ProbFeatRulesType
SemanticFeature
Facets Pattern: "(\p{Ll}+)([_]\p{Ll}+|[_]\p{N}+)*".
AssociatedRules
Regular expression: "\p{Zs}[(][s][p]\p{N}+([_]\p{N}+)?[,]\p{Zs}\p{N}+([.]\p{N}+)?[)]|\p{Zs}[(][s][p]\p{N}+([_]\p{N}+)?[)]|\p{Zs}[(]\p{N}+([.]\p{N}+)?[)]".

Figure 10: The structure of the ChooseAndInsert operation.

(24) sn4: SPOKEN_TENOR --> *10% formal (0.32) Or 70% consultative (0.33) Or 20% casual (0.34).*

The Operation EnterSystem is defined by the complex type EnterSystemType (cf. the expression in italics in (25)):

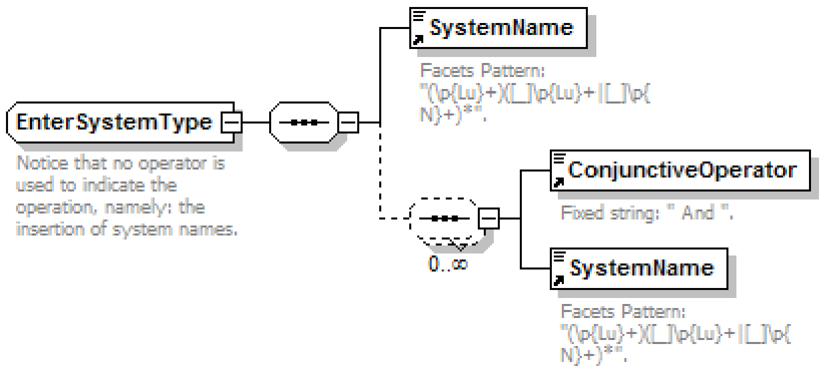

SystemName
Facets Pattern: "(\p{Lu}+)([_]\p{Lu}+|[_]\p{N}+)*".
EnterSystemType
Notice that no operator is used to indicate the operation, namely: the insertion of system names.
ConjunctiveOperator
Fixed string: " And ".
0..∞
SystemName
Facets Pattern: "(\p{Lu}+)([_]\p{Lu}+|[_]\p{N}+)*".

Figure 11: The structure of the EnterSystem operation.

(25) sn1: entity --> *MODE And ENTITY_TYPE.*

The Operation EnterConjunctionOfChoicePoints is defined by the complex type EnterConjunctionOfChoicePointsType (cf. the expression in italics in (26)):

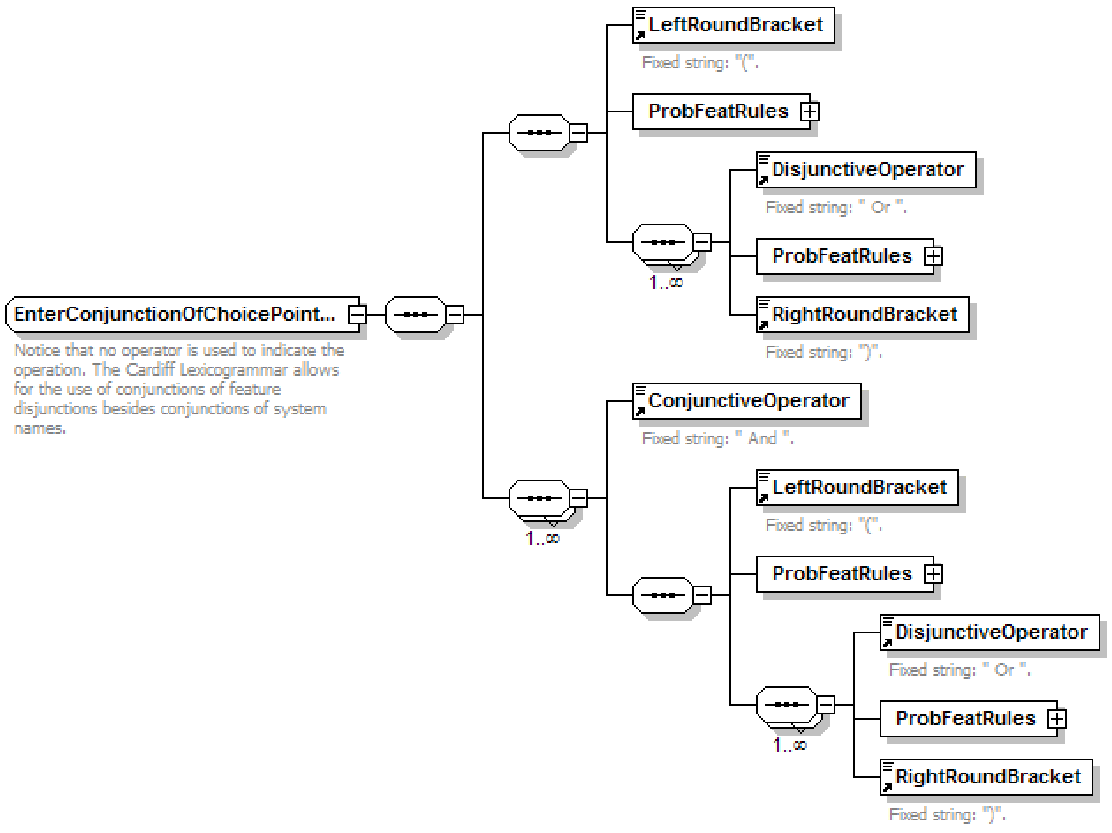


Figure 12: The structure of the EnterConjunctionOfChoicePoints

operation.

See complex type ProbFeatRulesType above for the element ProbFeatRules.

(26) sn74: another_co_ordinated_situation --> *(99% one_following_situation (19.1) Or 1% two_or_more_following_situations (19.2)) And (95% another_additive_situation (19.81) Or 5% another_alternative_situation (19.61)).*

The Operation InsertGate is defined by the global element InsertGate (cf. the expression in italics in (27)):

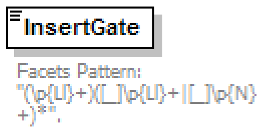


Figure 13: The structure of the InsertGate operation.

(27) sn274: (agent_only_unmarked Or affected_only_unmarked Or (agent_unmarked And agent_subject_theme) Or (agent_unmarked And affected_covert) Or (affected_unmarked And affected_subject_theme) Or (agent_covert And affected_unmarked) Or at_carrier_unmarked) --> *a_subject_theme_unmarked*.

ForSamePassPrefer is an operation of Same Pass Preference Resetting Rules which is defined by the complex type ForSamePassPreferType (cf. the expression in italics in (28) below):

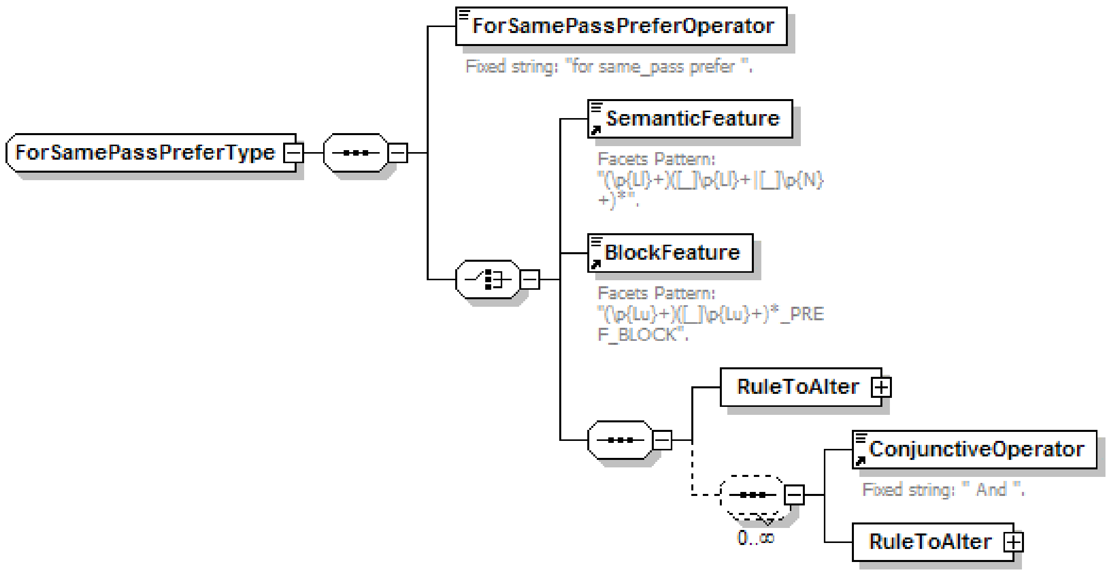


Figure 14a: The structure of the ForSamePassPrefer operation.

The element RuleToAlter is in turn defined by the complex type RuleToAlterType:

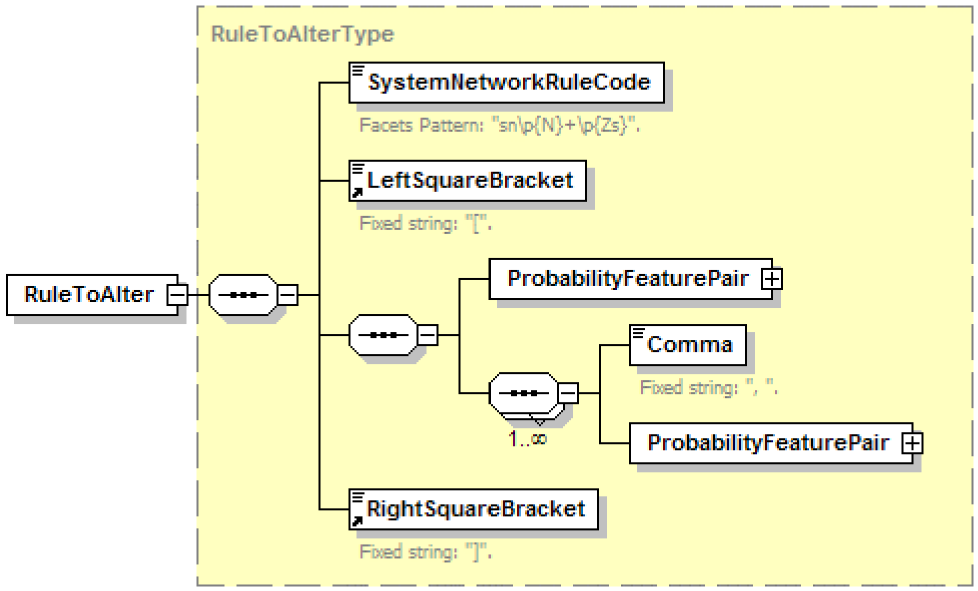


Figure 14b: The structure of the ForSamePassPrefer operation.

And the element ProbabilityFeaturePair is defined as follows:

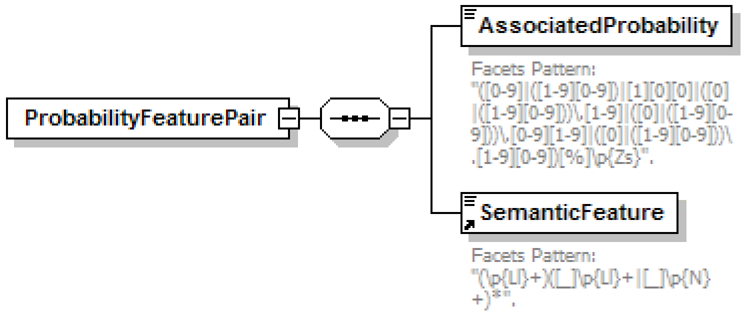

ProbabilityFeaturePair
AssociatedProbability
Facets Pattern:
"([0-9]|([1-9][0-9])|[1][0][0]|([0]|([1-9][0-9]))\.[1-9]|([0]|([1-9][0-9]))\.[0-9][1-9]|([0]|([1-9][0-9]))\.[1-9][0-9])[%]\p{Zs}".
SemanticFeature
Facets Pattern:
"(\p{Ll}+)([_]\p{Ll}+|[_]\p{N}+)*".

Figure 14c: The structure of the ForSamePassPrefer operation.

(28) sp1_1: congruent_situation --> written --> *for same_pass prefer sn12 [99.98% information, 0.02% proposal_for_action] And sn14 [99.9% giver, 0.1% seeker & 0% confirmation_seeker]*.

XFillsTriggeringElementY, XIsLocatedAt, XIsConflatedWithY, XIsExposedAsY, SuffisOfXIsY, ForAnyReEntryPrefer, ForSameOrXParticipantRoleOrElementPrefer, ForXRe-enterAtEntity, and ApplySubrule are Realization Rule operations.

The global element XfillsTriggeringElementY defines the corresponding Operation (cf. the expression in italics in (29)):

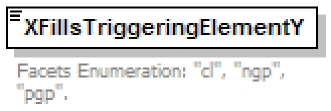


Figure 15: The structure of the XFillsTriggeringElementY operation.

(29) 60: congruent_thing --> *ngp*.

The following element defines the Operation XisLocatedAt (cf. the expressions in italics in (30)):

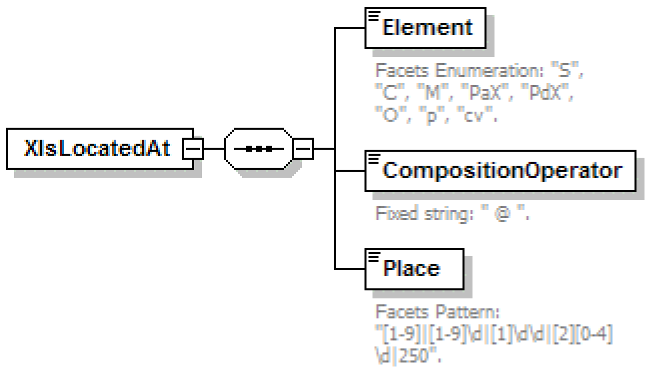

XIsLocatedAt
Element
Facets Enumeration: "S", "C", "M", "PaX", "PdX", "O", "p", "cv".
CompositionOperator
Fixed string: " @ ".
Place
Facets Pattern: "[1-9]|[1-9]\d|[1]\d\d|[2][0-4]\d|250".

Figure 16: The structure of the XisLocatedAt operation.

(30) 90: minor_relationship_with_thing --> pgp, *p @ 7*, *cv @ 8*.

The Operation XisConflatedWithY is defined by the following element (cf. the expressions in italics in (31)):

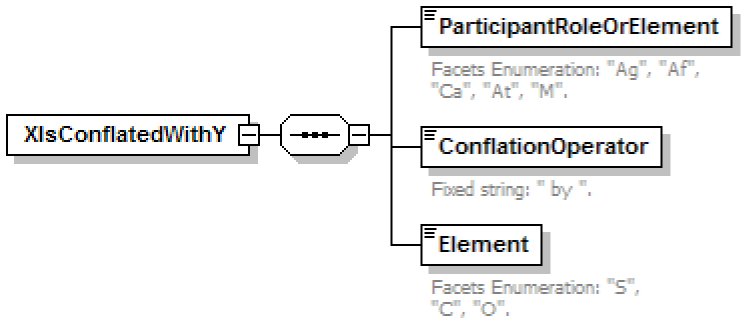


Figure 17: The structure of the XIsConflatedWithY operation.

(31) 6.482: affected_covert --> *Ag by S*, not proposal_for_action --> agent_unmarked --> apply Ag_preferences_subrule, agent_sought --> apply Ag_sought_preferences_subrule, for Ag re_enter_at entity, C @

120, *Af by C*.

The following element defines the Operation XisExposedAsY (cf. the expressions in italics in (32)):

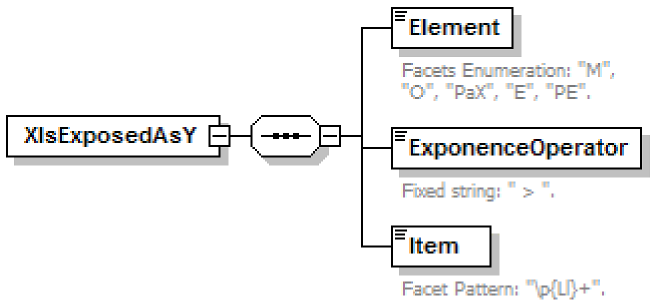

XIsExposedAsY
Element
Facets Enumeration: "M", "O", "PaX", "E", "PE".
ExponenceOperator
Fixed string: " > ".
Item
Facet Pattern: "\p{Ll}+".

Figure 18: The structure of the XisExposedAsY operation.

(32) Ender_subrule --> spoken --> *E* > ||, written --> unmarked_mood_wr --> *E* > ., (seeker Or confirmation_seeker Or request) --> *E* > *?*, (fun_mood_wr Or enthusiastic_mood_wr) --> *E* > *!*.

The Operation SuffixOfXIsY is defined by the following element (cf. the expressions in italics in (33)):

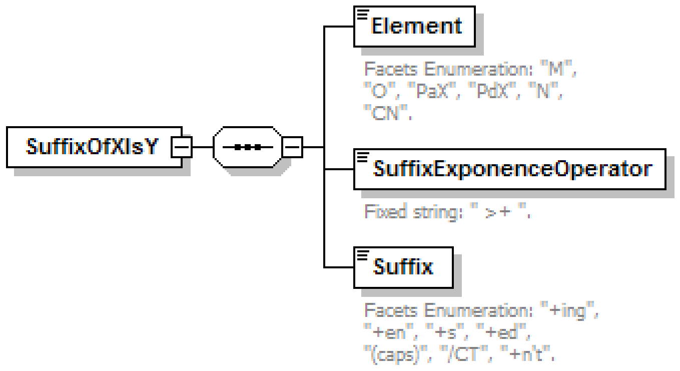

SuffixOfXIsY
Element
Facets Enumeration: "M", "O", "PaX", "PdX", "N", "CN".
SuffixExponenceOperator
Fixed string: " >+ ".
Suffix
Facets Enumeration: "+ing", "+en", "+s", "+ed", "(caps)", "/CT", "+n't".

Figure 19: The structure of the SuffixOfXIsY operation.

(33) 98.2: contrastive_newness_on_process --> CN by M, spoken --> *M* >+ */CT*, written --> *M* >+ *(caps)*.

The complex type ForAnyReEntryPreferType defines the Operation ForAnyReEntryPrefer (cf. the expression in italics in (34)):

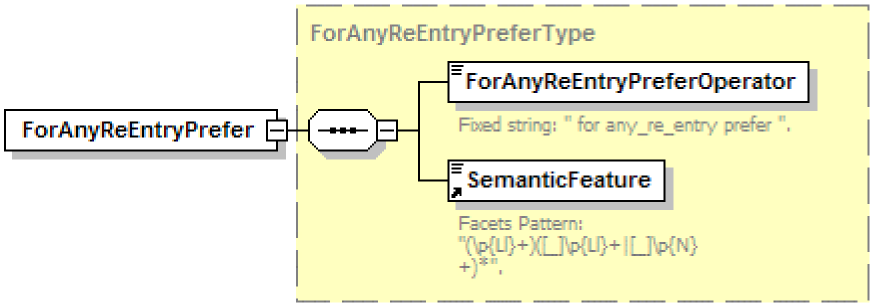

ForAnyReEntryPreferType
ForAnyReEntryPrefer
ForAnyReEntryPreferOperator
Fixed string: " for any_re_entry prefer ".
SemanticFeature
Facets Pattern:
"(\p{Ll}+)([_]\p{Ll}+|[_]\p{N}+)*".

Figure 20: The structure of the ForAnyReEntryPrefer operation.

(34) 0.31: very_formal --> *for any_re_entry prefer very_formal*.

The Operation ForSameOrXParticipantRoleOrElementPrefer abbreviates two separate operations, namely: ForSameParticipantRoleOrElementPrefer (cf. the expressions in italics in (36)) and ForXElementOrParticipantRolePrefer (cf. the expression in italics in (35)), and it is defined by the following complex type:

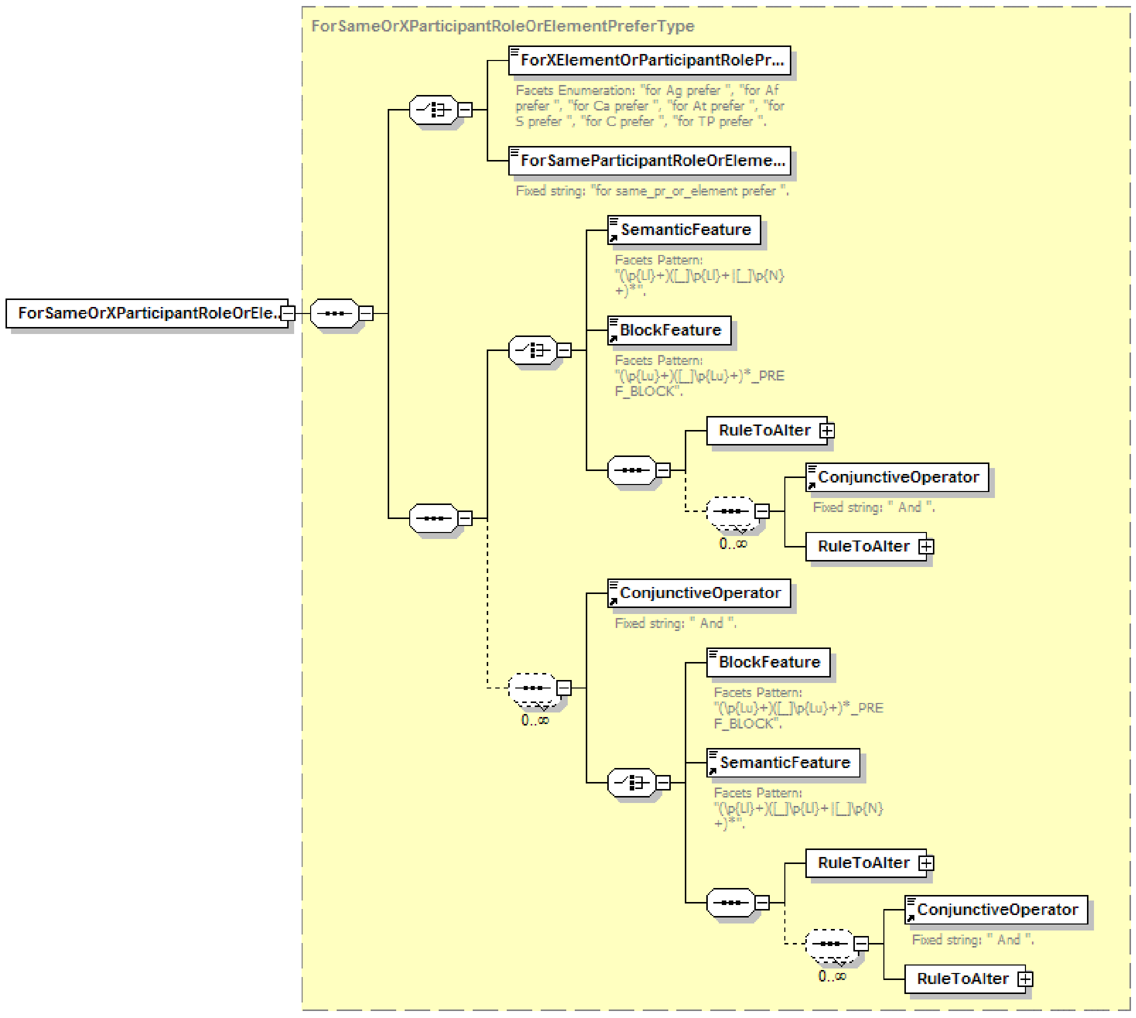

ForSameOrXParticipantRoleOrElementPreferType
ForSameOrXParticipantRoleOrEle...
ForXElementOrParticipantRolePr...
Facets Enumeration: "for Ag prefer ", "for Af prefer ", "for Ca prefer ", "for At prefer ", "for S prefer ", "for C prefer ", "for TP prefer ".
ForSameParticipantRoleOrEleme...
Fixed string: "for same_pr_or_element prefer ".
SemanticFeature
Facets Pattern: "(\p{Ll}+)([_]\p{Ll}+|[_]\p{N}+)*".
BlockFeature
Facets Pattern: "(\p{Lu}+)([_]\p{Lu}+)*_PREF_BLOCK".
RuleToAlter
ConjunctiveOperator
Fixed string: " And ".
RuleToAlter
0..∞
ConjunctiveOperator
Fixed string: " And ".
0..∞
BlockFeature
Facets Pattern: "(\p{Lu}+)([_]\p{Lu}+)*_PREF_BLOCK".
SemanticFeature
Facets Pattern: "(\p{Ll}+)([_]\p{Ll}+|[_]\p{N}+)*".
RuleToAlter
ConjunctiveOperator
Fixed string: " And ".
RuleToAlter
0..∞

Figure 21: The structure of the ForSameOrXParticipantRoleOrElementPrefer operation.

For the definition of RuleToAlter, see complex type RuleToAlterType in Figure 14b above.

(35) 6.20051: changing_as_such --> M > change, apply r, (affected_unmarked Or affected_only_unmarked) --> *for Af prefer sn244 [95% artefact, 5% natural_object]*.

(36) 19.61: another_alternative_situation --> & @ 4, & > or, one_following_situation --> *for same_pr_or_element prefer final_alternative_situation*, two_or_more_following_situations --> *for same_pr_or_element prefer another_alternative_situation*.

DeleteX, XisWrittenAsY, DoubleFinalLetter, and ApplySubrule are

Graphological Rule operations.

The Operation DeleteX is defined as follows (cf. the expression in italics in (37)):

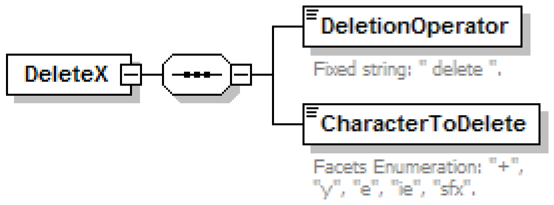

DeletionOperator
Fixed string: " delete ".
DeleteX
CharacterToDelete
Facets Enumeration: "+", "y", "e", "ie", "sfx".

Figure 22: The structure of the DeleteX operation.

(37) gr1: +n’t --> *delete* +.

The following element defines the Operation XisWrittenAsY (cf. the expression in italics in (38)):

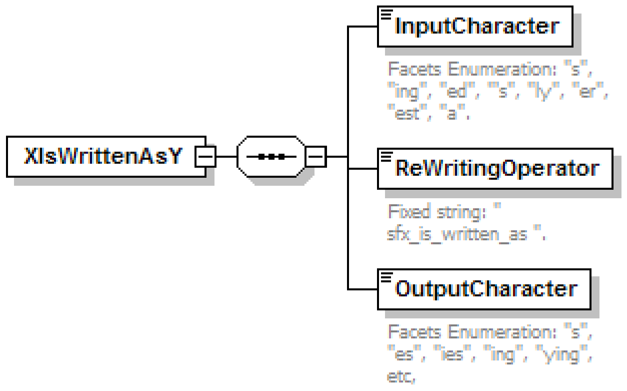

InputCharacter
Facets Enumeration: "s", "ing", "ed", "'s", "ly", "er", "est", "a".
XIsWrittenAsY
ReWritingOperator
Fixed string: " sfx_is_written_as ".
OutputCharacter
Facets Enumeration: "s", "es", "ies", "ing", "ying", etc.

Figure 23: The structure of the XIsWrittenAsY operation.

(38) ing_subrule_1_1 --> ie --> delete +, delete ie, *ing sfx_is_written_as ying*, Else apply ing_subrule_2.

The Operation DoubleFinalLetter is defined by the following element (cf. the expression in italics in (39)):

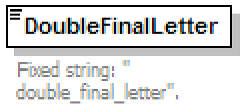


Figure 24: The structure of the DoubleFinalLetter operation.

(39) ing_subrule_8 --> c --> delete +, ing sfx_is_written_as king, Else *double_final_letter*, delete +, ing sfx_is_written_as ing.

The Operation ApplySubrule is defined as follows (cf. the expressions in italics in (40)-(42)):

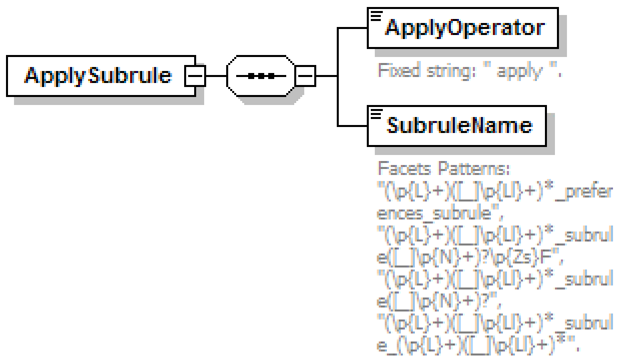

ApplySubrule
ApplyOperator
Fixed string: " apply ".
SubruleName
Facets Patterns:
"(\p{L}+)([_]\p{Ll}+)*_preferences_subrule",
"(\p{L}+)([_]\p{Ll}+)*_subrule([_]\p{N}+)?\p{Zs}F",
"(\p{L}+)([_]\p{Ll}+)*_subrule([_]\p{N}+)?",
"(\p{L}+)([_]\p{Ll}+)*_subrule_(\p{L}+)([_]\p{Ll}+)*".

Figure 25: The structure of the ApplySubrule operation.

(40) 2.12: at_carrier_unmarked --> *apply Ca_preferences_subrule*, for Ca re_enter_at entity.

(41) r --> (information Or (proposal_for_action And (period_marked Or unmarked_passive))) --> *apply rvs_subrule_1*.

(42) sp60: congruent_thing --> (fills At And {on_previous_pass} no_ad_hoc_description) --> *apply carrier_attribute_agreement_subrule*.

The element ForXRe-enterAtEntity defines the corresponding Operation (cf. the expression in italics in (43)):

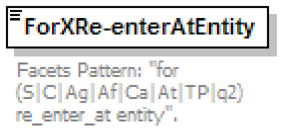

ForXRe-enterAtEntity
Facets Pattern: "for (S|C|Ag|Af|Ca|At|TP|q2) re_enter_at entity".

Figure 26: The structure of the ForXRe-enterAtEntity operation.

(43) 1.33: request --> O @ 31, for S prefer thing & congruent_thing & stereotypical_thing & interactant & addressee, *for S re_enter_at entity*.

4. CLGG Subrule Classes

Except for System Network Rules (A, B, and C), all other CLGG rules have associated subrules. The essential difference between rules and subrules is that the latter lack a RuleCode (cf. Figure 3), and the first Condition of the Implication element is a single feature realized by an appropriate regular expression containing, in most cases, the expression "subrule".

4.1. Same Pass Preference Resetting Subrules

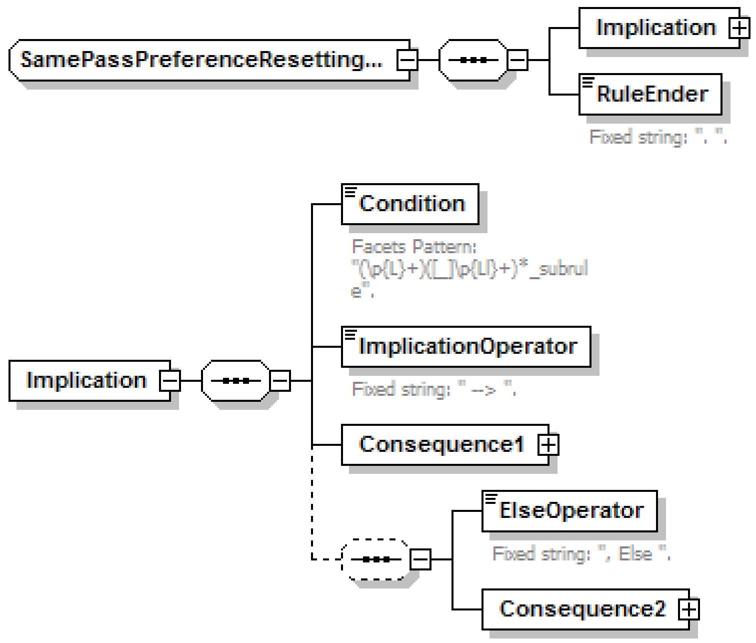
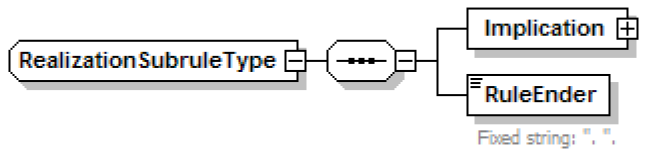


Figure 27: The structure of Same Pass Preference Resetting Subrules.

Example (44), which is an abbreviation of (11) above, illustrates this class of subrule:

(44) carrier_attribute_agreement_subrule --> {on_previous_pass} (interactant Or human_tc Or human_ssth Or whole_human Or name_of_person) --> for same_pass prefer human_tc And human_ssth And BASIC_TYPICALLY_HUMAN_PREF_BLOCK And TYPICALLY_HUMAN_CC_PREF_BLOCK, {on_previous_pass} (non_human_tc Or non_human_cr Or non_human_ssth) … .

4.2. Realization Subrules



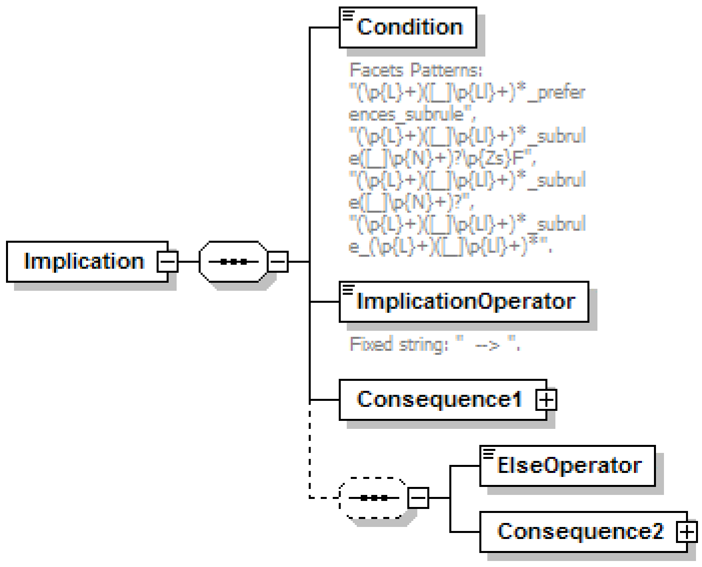
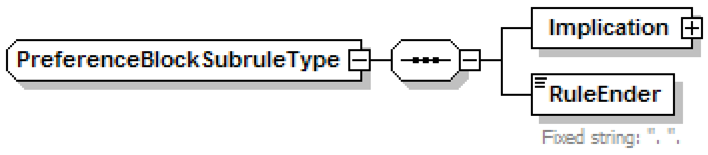

Condition
Facets Patterns: "(\p{L}+)([_]\p{Ll}+)*_preferences_subrule", "(\p{L}+)([_]\p{Ll}+)*_subrule([_]\p{N}+)?\p{Zs}F", "(\p{L}+)([_]\p{Ll}+)*_subrule([_]\p{N}+)?", "(\p{L}+)([_]\p{Ll}+)*_subrule_(\p{L}+)([_]\p{Ll}+)*".
Implication
ImplicationOperator
Fixed string: " --> ".
Consequence1
ElseOperator
Consequence2

Figure 28: The structure of Realization Subrules.

Example (45) illustrates this subrule class:

(45) Operator_placement_subrule --> (giver Or (seeker And ncs_theme_on_a_subject_theme_sought_r)) --> O @ 35, (seeker And not ncs_theme_on_a_subject_theme_sought_r) Or confirmation_seeker) --> O @ 31.

4.3. Preference Block Subrules (subrules of both Same Pass Preference Resetting Rules and Realization Rules)



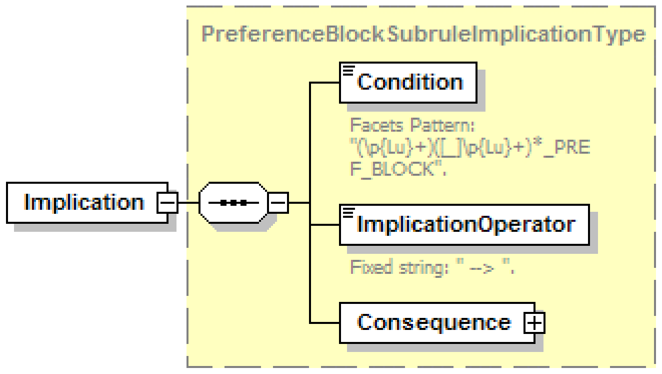

PreferenceBlockSubruleImplicationType
Implication
Condition
Facets Pattern: "(\p{Lu}+)([_]\p{Lu}+)*_PREF_BLOCK".
ImplicationOperator
Fixed string: " --> ".
Consequence

Figure 29: The structure of Preference Block Subrules.

Example (46) illustrates this class of subrule:

(46) BASIC_SING_OUTSIDER_PREF_BLOCK --> singular_tc And singular_loc_rth And singular_pos_rth And singular_ssth And sn157 [99.999% singular_cc, 0.001% plural_cc, 0% class_of_count_thing].

4.4. Graphological Subrules

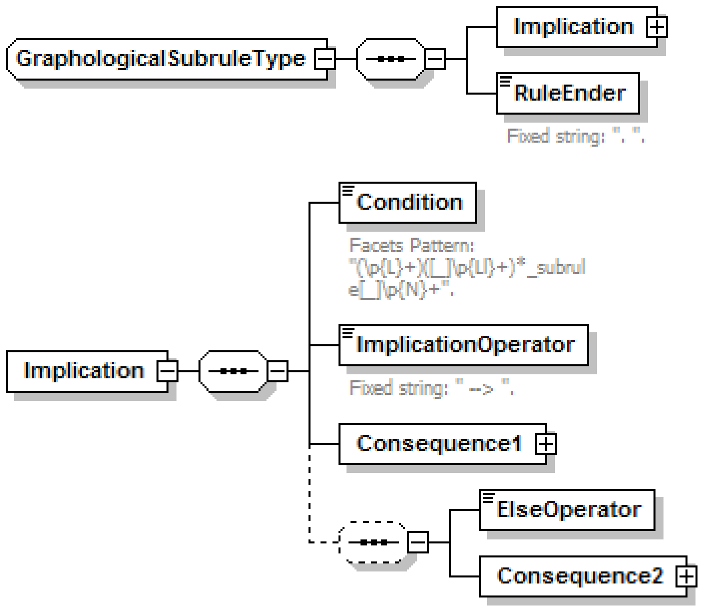


Figure 30: The structure of Graphological Subrules.

Example (47) illustrates this class of subrule:

(47) s_subrule_1 --> y --> (ay Or ey Or oy Or uy) --> delete +, s sfx_is_written_as s, Else delete +, delete y, s sfx_is_written_as ies, Else delete +, s sfx_is_written_as s.

5. Conclusions

In its current state of development, the Cardiff Grammar has no means to help linguists to write generation-oriented grammar versions based on CLGG, for the "grammar" of CLGG rules is nowhere but in the minds of Robin Fawcett and Gordon Tucker. As shown in §§2-4, CLGG rules are highly structured texts. As such, their well-formedness and validity can be guaranteed by providing an XML definition of them. This is precisely the issue that this paper has addressed and solved. I have provided an XML schema of CLGG rule classes from the point of view of their structural properties. Among other useful applications, this schema can certainly be used to design a specific rule editor which should significantly facilitate the writing of Cardiff like grammars.

REFERENCES

CASTEL, Víctor M. 2006a. *An Implementation of GENESYS: The Cardiff Grammar Generator*. Cardiff, Wales/Mendoza, Argentina: Cardiff University and CONICET/UNCUYO. For access permission, write to rp.fawcett@virgin.net.

_____. 2006b. *The Cardiff Grammar Generator Online Help*. Cardiff, Wales / Mendoza, Argentina: Cardiff University and CONICET/UNCUYO. Available at http://www.cricyt.edu.ar/institutos/incihusa/ul/webhelp/Victor_M._Castel.htm.

_____. 2006c. *Cardiff Grammar Generator Rule Classes*. MSWord file of the Cardiff Grammar Generator Rule Classes XML Schema. Available at http://www.cricyt.edu.ar/institutos/incihusa/ul/webhelp/Victor_M._Castel.htm.

_____. 2006d. *An XML Schema of the Cardiff Grammar Generator Rule Classes*. XMLSPY[5] ".xsd" file of the Cardiff Grammar Generator Rule Classes. Available at http://www.cricyt.edu.ar/institutos/incihusa/ul/webhelp/Victor_M._Castel.htm.

_____. 2005. "Determinación dinámica de valores de verdad de condiciones de reglas de generación de textos". In: Víctor M. Castel. Comp. 2005. *Desarrollo, implementación y utilización de modelos para el procesamiento automático de textos*. Mendoza: Editorial de la Facultad de Filosofía y Letras, Universidad Nacional de Cuyo. Available at http://ffyl.uncu.edu.ar/?id_rubrique=197&id_sector=42.

FAWCETT, Robin P. 2004a. *The Mini-Grammar of GENESYS Version 5*. Cardiff: Computational Linguistics Unit, Cardiff University.

_____. 2004b. Realizing Meaning in Intonation and Punctuation in English: The GENESYS Model. *Communal Working Papers* **19**. Cardiff: Computational Linguistics Unit, Cardiff University.

_____. 2000. *A theory of syntax for systemic functional linguistics*. Amsterdam: John Benjamins.

FAWCETT, Robin P. Gordon H. TUCKER, y Yuen Q. LIN. 1993. How a systemic functional grammar works: The role of realization in realization. In: Helmut Horacek & Michael Zock. Eds. 1993. *New concepts in natural language generation*. London: Pinter.

---

[*] I am grateful to Ana M. Miret for valuable comments on various aspects of the paper.

[i] For an algorithm which takes care of the assignment of truth values to CLGG rule conditions, see Castel (2005).

[ii] In the sense “well-formedness” and “validity” are understood in XML.

[iii] No definition is provided here for intonation rules, for they have not been specified yet as rules proper in the available source (Fawcett 2004b), i.e. the Mini version of CLGG has an algorithm for this class of rules but there is no formal declarative statement of them.

[iv] I use “Or” and “And” instead of “/” and “&”, respectively, in the definition of System Network Rules to maximize uniformity with Realization Rules. I have also adapted and modified other minor aspects of the original specification in Fawcett (2004a).

[v] The XML definition of CLGG rules given in §§3-4 below has been constructed with XMLSPY[5] Professional Edition.

[vi] For a more complete definition of the CLGG rules discussed in this paper, see Castel (2006c, d), available at http://www.cricyt.edu.ar/institutos/incihusa/ul/webhelp/Victor_M._Castel.htm.