```
;;;===========================================================;;;
;;;-----------------------------------------------------------;
;;; Rule 1 for START                                          ;
;;;                                                           ;
;;; Generate a parenthetical sentence.                        ;

(<start> --> (<l-paren> <sent> <r-paren>)
         (((x0 parenthetical) =c +)
          ((x0 parenthetical) = *REMOVE*)
         ((x1 root) <= "[")
         ((x3 root) <= "]")
          (x2 = x0)))

(<l-paren> --> (%)
           (((x1 value) = (x0 root))))
(<r-paren> --> (%)
           (((x1 value) = (x0 root))))


;;;-----------------------------------------------------------;
;;; Rule 2 for START                                          ;
;;;                                                           ;
;;; Generate a labelled sentence.                             ;

(<start> --> (<label> <colon> <sent>)
         (((x0 label) = *defined*)
          (x1 = (x0 label))
         (x3 = x0)))

(<label> --> (<np>)
         ((x1 = x0)))
(<colon> --> (%)
         (((x1 value) <= ": ")))

;;;-----------------------------------------------------------;
;;; Rule 3 for START                                          ;
;;;                                                           ;
;;; Generate a hyphened sentence.                             ;


(<start> --> (<hyphen> <sent>)
         (((x0 hyphen) =c +)
         (x2 = x0)))

(<hyphen> --> (%)
          (((x1 value) <= "--")))

 ;;;-----------------------------------------------------------;
 ;;; Rule 4 for START                                          ;
 ;;;                                                           ;
 ;;; Generate a number bulleted sentence.                      ;
```

```
(<start> --> (<bullet> <sent>)
         (((x0 number-bullet) = *DEFINED*)
          ((x1 integer) == (x0 number-bullet))
          (x2 = x0)))

(<bullet> --> (<number> <period>)
           ((x1 = x0)))

; <period> will be defined later.
;;;-------------------------------------------------------------;
;;;                                                             ;
;;; Rule 5 for START                                            ;
;;;                                                             ;
;;; Generate ordinary sentences and none phrases.               ;

(<start> --> (<sent>)
        ((x1 = x0)))

;;;===========================================================;;;
;;;                          SENT                             ;;;
;;;===========================================================;;;
;;;-------------------------------------------------------------;
;;;                                                             ;
;;; Rule 1 for SENT                                             ;
;;;                                                             ;
;;; There three kinds of endings of sentences: colon, period, or
;;; none. Period is the default. Colon is for colonized sentences.
;;; And the none is for headings.
;;; sent 1

(<sent> --> (<s> <colon>)
         (((x0 colon) = *defined*)
          ((x0 colon) = *remove*)
          (x1 = x0)))
; <colon> has been defined previously.

;;; sent 2

(<sent> --> (<s> <period>)
         (((x0 colon) = *UNDEFINED*)
          ((x0 heading) = *UNDEFINED*)
          (x1 = x0)))

;;; sent 3

(<sent> --> (<np>)
         ((*EOR* (((x0 heading) = *defined*))
                 (((x0 hyphen) = *defined*))
                 (((x0 label) = *defined*)))
          ((x0 cat) =c n)
          (x1 = x0)))

;;; sent 4
```

```
(<sent> --> (<vp>)
        ((*EOR* (((x0 heading) = *defined*))
              (((x0 hyphen) = *defined*)))
         ((x0 cat) =c v)
         (x1 = x0)))

(<period> --> (%)
          (((x1 value) <= ". ")))


;;;============================================================;;;
;;;                           S                                ;;;
;;;============================================================;;;
;;;------------------------------------------------------------;
;;;                                                            ;
;;; Rule 1 for S                                               ;
;;;                                                            ;
;;; Discourse cohesion (before S)                              ;

(<s> --> (<cohesion> <s>)
     (((x0 discourse-cohesion) = *DEFINED*)
      (x1 == (x0 discourse-cohesion))
      (x2 = x0)))

(<cohesion> --> (<adv> <comma>)
             (((x0 cat) =c adv)
              (x1 = x0)))

(<comma> --> (%)
         (((x1 value) <= ", ")))

;;;------------------------------------------------------------;
;;;                                                            ;
;;; Rule 2 for S                                               ;
;;;                                                            ;
;;; Precondition clause (before S)                             ;

(<s> --> (<pre-condition> <s>)
     (((x0 pre-condition) = *DEFINED*)
      (x1 == (x0 pre-condition))
      (x2 = x0)))

(<pre-condition> --> (<conj> <simp-s> <comma>)
                  (((x1 root) <= "如果")
                   ((x1 cat) = conj)
                   (x2 = x0)))

;;;------------------------------------------------------------;
;;;                                                            ;
;;; Rule 3 for S                                               ;

(<s> --> (<reason-event> <s>)
     (((x0 reason) = *DEFINED*)
      (x1 == (x0 reason))
      (x2 = x0)))
```

```
(<reason-event> --> (<conj> <simp-s> <comma>)
                    (((x1 root) <= "因为")
                     ((x1 cat) = conj)
                     (x2 = x0)))

(<conj> --> (%)
          (((x0 cat) =c conj)
           ((x1 value) = (x0 root))))

;;;------------------------------------------------------------;
;;; Rule 4 for S                                                ;
;;;                                                             ;
;;; When clause (before S)                                      ;

(<s> --> (<when-event> <s>)
     (((x0 when-event) = *DEFINED*)
      (x1 == (x0 when-event))
      (x2 = x0)))

(<when-event> --> (<conj> <simp-s> <shi> <comma>)
                  (((x1 cat) = conj)
                   ((x1 root) <= "当")
                   (x2 = x0)))
(<shi> --> (%)
        (((x1 value) <= "时")))

;;;------------------------------------------------------------;
;;; Rule 5 for S                                                ;
;;;                                                             ;
;;; Although clause (before S)                                  ;

(<s> --> (<although-event> <s>)
     (((x0 although-event) = *defined*)
      (x1 == (x0 although-event))
      (x2 = x0)))

(<although-event> --> (<conj> <simp-s> <comma>)
                      (((x1 cat) = conj)
                       ((x1 root) <= "尽管")
                       (x2 = x0)))

;;;------------------------------------------------------------;
;;; Rule 6 for S                                                ;
;;;                                                             ;
;;; During clause (before S)                                    ;

(<s> --> (<during-event> <s>)
     (((x0 during-event) = *defined*)
      (x1 == (x0 during-event))
      (x2 = x0)))

(<during-event> --> (<conj> <simp-s> <comma>)
```

```
                     (((x0 cat) =c v)
                     ((x1 cat) = conj)
                      ((x1 root) <= "当")
                      (x2 = x0)))

(<during-event> --> (<pp>)
      (((x0 cat) =c n)
       (x1 = x0)
       ((x1 surf root) <= "期间")
        ((x1 prep root) <= "在")))

;;;------------------------------------------------------------;
;;; Rule 7 for S                                               ;
;;;                                                            ;
;;; Until clause (after S)                                     ;

(<s> --> (<s> <until-event>)
     (((x0 until-event) = *defined*)
      (x2 == (x0 until-event))
      (x1 = x0)))

(<until-event> --> (<conj> <simp-s>)
                (((x1 cat) = conj)
                 ((x1 root) <= "直到")
                 (x2 = x0)))

;;;------------------------------------------------------------;
;;; Rule 8 for S                                               ;
;;;                                                            ;
;;; But clause (after S)                                       ;

(<s> --> (<s> <comma> <but-event>)
     (((x0 but-event) = *defined*)
      (x3 == (x0 but-event))
      (x1 = x0)))

(<but-event> --> (<conj> <simp-s>)
                (((x1 cat) = conj)
                 ((x1 root) <= "但是")
                 (x2 = x0)))

;;;------------------------------------------------------------;
;;; Rule 9 for S                                               ;
;;;                                                            ;
;;; Sequential clause (after S)                                ;

(<s> --> (<s> <comma> <sequential-event>)
     (((x0 sequential-event) = *defined*)
      (x3 == (x0 sequential-event))
      (x1 = x0)))

; (<sequential-event> --> (<conj> <simp-s>)
```

```
;               (((x1 cat) = conj)
;               ((x1 root) <= "而")
;               (x2 = x0)))

 (<sequential-event> --> (<simp-s>)
                ((x1 = x0)))

 ;;;-------------------------------------------------------------;
 ;;;                                                             ;
 ;;; Rule 10 for S                                               ;
 ;;;                                                             ;
 ;;;                                                             ;
 ;;; or clause (after S)                                         ;

 (<s> --> (<s> <conj> <s>)
      (((x0 or-event) = *DEFINED*)
       ((x2 cat) = conj)
       ((x2 root) <= "或")
       (x3 == (x0 or-event))
       (x1 = x0)))

 ;;;-------------------------------------------------------------;
 ;;;                                                             ;
 ;;; Rule 11 for S                                               ;
 ;;;                                                             ;
 ;;;                                                             ;
 ;;; Simple sentence                                             ;

 (<s> --> (<simp-s>)
      (((x0 cat) =c v)
       (x1 = x0)))

 ;;;-------------------------------------------------------------;
 ;;;
 ;;; Special rule 1

 (<simp-s> --> (<dec-act-s>)
         (((x0 root) =c "推荐")
          ((x0 passive) =c +)
         ((x0 mood) =c dec)
         ((x1 cat) = v)
         ((x1 root) <= "是")
          (*EOR* (((x1 modal) == (x0 modal))) (((x0 modal) = *undefined*)))
          (*EOR* (((x1 tense) == (x0 tense))) (((x0 tense) = *undefined*)))
          (*EOR* (((x1 mood) == (x0 mood))) (((x0 mood) = *undefined*)))
          (*EOR* (((x1 tentative) == (x0 tentative))) (((x0 tentative) = *undefined*)
))
          (*EOR* (((x1 obligation) == (x0 obligation))) (((x0 obligation) = *undefine
d*)))
          (*EOR* (((x1 compulsion) == (x0 compulsion))) (((x0 compulsion) = *undefine
d*)))
          (*EOR* (((x1 negation) == (x0 negation))) (((x0 negation) = *undefined*)))
          (*EOR* (((x0 theme) = *defined*)((x1 theme) == (x0 theme)))
                 (((x0 patient) = *defined*)((x1 theme) == (x0 patient))))
         ((x1 predicated-of-theme) <= (x0))))

 ;;;-------------------------------------------------------------;
```

```
;;; Special rule 2

(<simp-s> --> (<dec-act-s>)
        (((x0 root) =c "适用")
         ((x0 passive) =c +)
        ((x0 mood) =c dec)
        ((x1 cat) = v)
        ((x1 root) <= "是")
         (*EOR* (((x1 modal) == (x0 modal))) (((x0 modal) = *undefined*)))
         (*EOR* (((x1 tense) == (x0 tense))) (((x0 tense) = *undefined*)))
         (*EOR* (((x1 mood) == (x0 mood))) (((x0 mood) = *undefined*)))
         (*EOR* (((x1 tentative) == (x0 tentative))) (((x0 tentative) = *undefined*)
))
         (*EOR* (((x1 obligation) == (x0 obligation))) (((x0 obligation) = *undefine
d*)))
         (*EOR* (((x1 compulsion) == (x0 compulsion))) (((x0 compulsion) = *undefine
d*)))
         (*EOR* (((x1 negation) == (x0 negation))) (((x0 negation) = *undefined*)))
        (*EOR* (((x0 purpose) = *defined*))
               (((x0 location) = *defined*)))
         (*EOR* (((x0 theme) = *defined*)((x1 theme) == (x0 theme)))
                (((x0 patient) = *defined*)((x1 theme) == (x0 patient))))
        ((x1 predicated-of-theme) <= (x0))))

;;;-----------------------------------------------------------;
;;; Special rule 3

(<simp-s> --> (<dec-act-s>)
        (((x0 root) =c "限制")
         ((x0 passive) =c +)
        ((x0 mood) =c dec)
        ((x1 cat) = v)
        ((x1 root) <= "是")
         (*EOR* (((x1 modal) == (x0 modal))) (((x0 modal) = *undefined*)))
         (*EOR* (((x1 tense) == (x0 tense))) (((x0 tense) = *undefined*)))
         (*EOR* (((x1 mood) == (x0 mood))) (((x0 mood) = *undefined*)))
         (*EOR* (((x1 tentative) == (x0 tentative))) (((x0 tentative) = *undefined*)
))
         (*EOR* (((x1 obligation) == (x0 obligation))) (((x0 obligation) = *undefine
d*)))
         (*EOR* (((x1 compulsion) == (x0 compulsion))) (((x0 compulsion) = *undefine
d*)))
         (*EOR* (((x1 negation) == (x0 negation))) (((x0 negation) = *undefined*)))
        (*EOR* (((x0 condition) = *defined*))
               (((x0 location) = *defined*)) (((x0 agent) = *undefined*)))
         (*EOR* (((x0 theme) = *defined*)((x1 theme) == (x0 theme)))
                (((x0 patient) = *defined*)((x1 theme) == (x0 patient))))
        ((x1 predicated-of-theme) <= (x0))))

;;;-----------------------------------------------------------;
;;; Special rule 4
```

```
(<simp-s> --> (<dec-act-s>)
       (((x0 root) = "设计")
        ((x0 passive) =c +)
       ((x0 mood) =c dec)
        ((x0 purpose) = *defined*)
       ((x1 cat) = v)
       ((x1 root) <= "是")
        (*EOR* (((x1 modal) == (x0 modal))) (((x0 modal) = *undefined*)))
        (*EOR* (((x1 tense) == (x0 tense))) (((x0 tense) = *undefined*)))
        (*EOR* (((x1 mood) == (x0 mood))) (((x0 mood) = *undefined*)))
        (*EOR* (((x1 tentative) == (x0 tentative))) (((x0 tentative) = *undefined*)
))
        (*EOR* (((x1 obligation) == (x0 obligation))) (((x0 obligation) = *undefine
d*)))
        (*EOR* (((x1 compulsion) == (x0 compulsion))) (((x0 compulsion) = *undefine
d*)))
        (*EOR* (((x1 negation) == (x0 negation))) (((x0 negation) = *undefined*)))
        (*EOR* (((x0 theme) = *defined*)((x1 theme) == (x0 theme)))
             (((x0 patient) = *defined*)((x1 theme) == (x0 patient))))
       ((x1 predicated-of-theme) <= (x0))))

;;;-------------------------------------------------------------;
;;; Special rule 5

(<simp-s> --> (<dec-act-s>)
       (((x0 root) = "制造")
        ((x0 passive) =c +)
       ((x0 mood) =c dec)
        ((x0 agent) = *defined*)
       ((x1 cat) = v)
       ((x1 root) <= "是")
        (*EOR* (((x1 modal) == (x0 modal))) (((x0 modal) = *undefined*)))
        (*EOR* (((x1 tense) == (x0 tense))) (((x0 tense) = *undefined*)))
        (*EOR* (((x1 mood) == (x0 mood))) (((x0 mood) = *undefined*)))
        (*EOR* (((x1 tentative) == (x0 tentative))) (((x0 tentative) = *undefined*)
))
        (*EOR* (((x1 obligation) == (x0 obligation))) (((x0 obligation) = *undefine
d*)))
        (*EOR* (((x1 compulsion) == (x0 compulsion))) (((x0 compulsion) = *undefine
d*)))
        (*EOR* (((x1 negation) == (x0 negation)))); (((x0 negation) = *undefined*))
)
        (*EOR* (((x0 theme) = *defined*)((x1 theme) == (x0 theme)))
             (((x0 patient) = *defined*)((x1 theme) == (x0 patient))))
       ((x0 subcat) <= 'v0)
       ((x1 predicated-of-theme) <= (x0))))

;;;-------------------------------------------------------------;
;;; Rule 6 for Simp-s                                                ;
;;;                                                             ;
;;; Purpose clause (befor S)                                    ;
```

```
;(<simp-s> --> (<simp-s> <purpose-event>)
;     (((x0 purpose) = *defined*)
;      ((x0 purpose mood) = *undefined*)
;      ((x0 purpose cat) =c v)
;      (x2 == (x0 purpose))
;      ((x2 conj root) <= "以便")
;      (x1 = x0)))

;;;-------------------------------------------------------------;
;;; Rule 7 for Simp-s                                               ;

(<simp-s> --> (<purpose-event> <simp-s>)
     (((x0 purpose) = *defined*)
      ((x0 passive) = *undefined*)
;     ((x0 purpose cat) =c n)
      (x1 == (x0 purpose))
      (x2 = x0)))

;;;-------------------------------------------------------------;
;;; Rule 8 for Simp-s                                               ;

(<simp-s> --> (<pp> <comma> <simp-s>)
     (((x0 search-for) = *defined*)
      (x1 == (x0 search-for))
      (x3 = x0)
      ((x1 prep root) <= "为查明")))

(<purpose-event> --> (<conj> <np>)
        (((x0 cat) =c n)
         ((x1 root) <= "为了")
         ((x1 cat) = conj)
         (x2 = x0)))

(<purpose-event> --> (<conj> <vp> <comma>)
        (((x0 cat) =c v)
         ((x1 root) <= "为了")
         ((x1 cat) = conj)
         (x2 = x0)))
;;        ((x2 mood) = dec)))

;;;==========================================================;;;
;;;                    SIMPLE SENTENCE                         ;;;
;;;==========================================================;;;
;;;-------------------------------------------------------------;
;;; Rule 9 for Simple setence                                     ;
;;;                                                               ;
;;; Declarative sentence

(<simp-s> --> (<dec-s>)
          ((*EOR* (((x0 mood) =c dec))(((x0 mood) = *undefined*)))
           (x1 = x0)))
```

```
;;;-----------------------------------------------------------------;
;;; Rule 10 for Simple setence                                      ;
;;;                                                                 ;
;;; Imperative sentence

(<simp-s> --> (<imp-s>)
          (((x0 mood) =c imp)
           (x1 = x0)))

;;;==============================================================;;;
;;;                 DECLARATIVE SENTENCE                         ;;;
;;;==============================================================;;;
;;;-----------------------------------------------------------------;
;;; RULE 1 for dec-s                                                ;
;;;                                                                 ;
;;; Declarative sentence with active voice.

(<dec-s> --> (<dec-act-s>)
          ((*EOR* (((x0 mood) =c dec))(((x0 mood) = *undefined*)))
           ((x0 passive) = *UNDEFINED*)
           (x1 = x0)))

;;;-----------------------------------------------------------------;
;;; RULE 2 for Simple setence                                       ;
;;;                                                                 ;
;;; Declarative sentence with passive voice.

(<dec-s> --> (<dec-pas-s>)
          (
;         ((x0 mood) =c dec)
;          ((x0 passive) =c +)
           (x1 = x0)))

;;;==============================================================;;;
;;;                 IMPERATIVE  SENTENCE                         ;;;
;;;==============================================================;;;
;;;-----------------------------------------------------------------;
;;; RULE 1 for Simple setence                                       ;
;;;                                                                 ;
;;; Imperative sentence

(<imp-s> --> (<vp>)
         (((x0 mood) =c imp)
          (x1 = x0)))

;;;==============================================================;;;
;;;                 DECLARATIVE ACTIVE SENTENCE                  ;;;
;;;==============================================================;;;
;;;-----------------------------------------------------------------;
;;; Special rule 1
;;; for sentence pattern: <np> is for <np>.
```

```
;;; dec-act-s 1
(<dec-act-s> --> (<np> <v> <ppadjunct>)
        (((x0 root) =c "是")
         ((x0 theme) = *defined*)
        ((x0 purpose) = *defined*)
         (x1 == (x0 theme))
         (x3 == (x0 purpose))
         ((x3 prep root) <= "用于")
        (x2 = x0)))

;;;------------------------------------------------------------;
;;;                                                            ;
;;; RULE 2 for Simple declarative setence                      ;
;;;                                                            ;
;;;                                                            ;
;;; Declarative sentence(THERE BE pattern)

(<dec-act-s> --> (<pp> <predicate1>)
           (((x0 theme pleonastic-pron) =c there)
            ((x0 predicate) = *DEFINED*)
            (x1 == (x0 condition))
            ((x1 prep root) = "在")
            ((x1 surf root) = "下")
            (x2 = x0)))

;;;------------------------------------------------------------;
;;;                                                            ;
;;; RULE 3 for Simple declarative setence                      ;

(<dec-act-s> --> (<pp> <predicate1>)
           (((x0 theme pleonastic-pron) =c there)
            ((x0 predicate) = *DEFINED*)
            (x1 == (x0 location))
            ((x1 prep root) = "在")
            (x2 = x0)))

;;;------------------------------------------------------------;
;;;                                                            ;
;;; RULE 4 for Simple declarative setence                      ;

(<dec-act-s> --> (<pp> <predicate1>)
           (((x0 theme pleonastic-pron) =c there)
            ((x0 predicate) = *DEFINED*)
            (x1 == (x0 between-items))
            ((x1 prep root) <= "在")
            ((x1 surf root) <= "之间")
            (x2 = x0)))

;;;------------------------------------------------------------;
;;;                                                            ;
;;; RULE 4 for Simple declarative setence                      ;

(<dec-act-s> --> (<predicate1>)
           (((x0 theme pleonastic-pron) =c there)
            ((x0 predicate) = *DEFINED*)
            (x1 = x0)))
```

```
(<predicate1> --> (<predicate1>)
      (((x0 root) =c "是")
        ((x0 root) <= "有")
        (x1 = x0)))

(<predicate1> --> (<future-aux> <predicate1>)
      (((x0 tense) =c future)
        ((x0 tense) = *REMOVE*)
        (x2 = x0)))

(<predicate1> --> (<modal> <predicate1>)
           (((x0 modal) = *defined*)
           ((x0 modal-processed) = *undefined*)
           (x1 = x0)
           (x2 = x0)
            ((x2 modal-processed) <= +)))

(<predicate1> --> (<predicate1>)
           (((x0 modal-processed) = *undefined*)
            ((x0 negation) = *defined*)
           (x1 = x0)
            ((x1 root) <= "没有")
            ((x1 modal-processed) <= +)))

(<predicate1> --> (<v> <np>)
               ((x2 == (x0 predicate))
                (x1 = x0)))

;(<np> --> (<vp>)
;      (((x0 cat) = v)
;       (x1 = x0)))

;;;-------------------------------------------------------------;
;;; RULE 5 for Simple declarative setence                       ;
;;;                                                             ;
;;; Declarative sentence(PREDICATE)
;;; Eample: sometimes it is necessary to remove an angled
;;; component.

(<dec-act-s> --> (<vp> <predicate2>)
           (((x0 root) = "是")
            ((x0 theme person) =c third)
            ((x0 theme number) =c sg)
            ((x0 event) = *DEFINED*)
            (x1 == (x0 event))
            (x2 == x0)))

(<predicate2> --> (<modal> <predicate2>)
           (((x0 modal) = *defined*)
           ((x0 modal-processed) = *undefined*)
           (x1 = x0)
           (x2 = x0)
```

```
            ((x2 modal-processed) <= +)))

(<predicate2> --> (<predicate2>)
           (((x0 modal-processed) = *undefined*)
           ((x0 negation) = *defined*)
           (x1 = x0)
            ((x1 modal-processed) <= +)
            ((x1 root) <= "不是")))

(<predicate2> --> (<adv> <predicate2>)
              (((x0 event-frequency) = *defined*)
               (x1 == (x0 event-frequency))
               (x2 = x0)))

(<predicate2> --> (<v> <adjp>)
              (((x0 predicated-of-event) = *DEFINED*)
               (x2 == (x0 predicated-of-event))
               (x1 = x0)))

;;;-----------------------------------------------------------;
;;; RULE 6 for Simple declarative setence                     ;
;;;                                                           ;
;;; Declarative sentence(PREDICATE)
;;; Example: this is correct mark.

(<dec-act-s> --> (<pp> <dec-act-s>)
        (;((x0 root) =c "是")
         (*EOR* (((x0 predicate) = *DEFINED*))
                (((x0 predicated-of-theme) = *defined*)))
         ((x0 location) = *DEFINED*)
         (x1 == (x0 location))
         ((x1 prep root) <= "在")
        (x2 = x0)))

;;;-----------------------------------------------------------;
;;; RULE 7 for Simple declarative setence                     ;

(<dec-act-s> --> (<np> <predicate3>)
           (;((x0 root) =c "是")
            (*EOR* (((x0 predicate) = *DEFINED*))
                   (((x0 predicated-of-theme) = *defined*)))
            (x1 == (x0 theme))
            (x2 = x0)))

(<predicate3> --> (<pp> <predicate3>)
      (((x0 range-of-predicate) = *DEFINED*)
        (x1 == (x0 range-of-predicate))
        ((x1 prep root) <= "对")
        ((x1 surf root) <= "来说")
        (x2 = x0)))

(<predicate3> --> (<modal> <predicate3>)
```

```
          (((x0 modal) = *defined*)
          ((x0 modal-processed) = *undefined*)
          (x1 = x0)
          (x2 = x0)
           ((x2 modal-processed) <= +)))

 (<predicate3> --> (<future-aux> <predicate3>)
      (((x0 tense) =c future)
        ((x0 tense) = *REMOVE*)
        (x2 = x0)))

 (<predicate3> --> (<predicate3>)
          (((x0 modal-processed) = *undefined*)
          ((x0 negation) = *defined*)
          (x1 = x0)
           ((x1 modal-processed) = +)
           ((x1 root) <= "不是")))

 (<predicate3> --> (<v> <np>)
          (((x0 predicate) = *DEFINED*)
           (x2 == (x0 predicate))
           (x1 = x0)))

 ;;; Example: This is very important.

 (<predicate3> --> (<adjp>)
          (((x0 predicated-of-theme) = *DEFINED*)
          ((x0 predicated-of-theme intensifier) = *undefined*)
          ((x0 predicated-of-theme than-object) = *undefined*)
          ((x0 predicated-of-theme as-possible) = *undefined*)
          ((x0 predicated-of-theme default-intensifier) = *defined*)
          ((x0 predicated-of-theme intensifier) <= (x0 predicated-of-theme default-
intensifier))
          ((x0 predicated-of-theme no-de) = +)
;;;          ((x0 no-shi) =c +)
           (x1 == (x0 predicated-of-theme))))

 (<predicate3> --> (<adjp>)
          (((x0 predicated-of-theme) = *DEFINED*)
;;;          ((x0 no-shi) =c +)
          ((x0 predicated-of-theme no-de) = +)
           (x1 == (x0 predicated-of-theme))))

 (<predicate3> --> (<v> <vp> <de>)
          (((x0 predicated-of-theme) = *DEFINED*)
          ((x0 predicated-of-theme cat) =c v)
           (*EOR* (((x0 root) =c "是")) (((x0 root) =c "不是")))
           (x2 == (x0 predicated-of-theme))
          (x1 = x0)))

 ;;;-------------------------------------------------------------;
 ;;; special rule for some verbs
```

```
;;; **
;;;
;;; vp 1

(<vp> --> (<vp> <pp>)
        (((x0 root) =c "使用")
;;       ((x0 passive) =c +)
        (*EOR* (((x0 cofactor) = *defined*)(x2 == (x0 cofactor)))
               (((x0 location) = *defined*)(x2 == (x0 location))))
         ((x2 prep root) <= "于")
        (x1 = x0)))

;;; vp 2

(<vp> --> (<vp> <pp>)
        (((x0 root) =c "适用")
         ((x0 passive) =c +)
        (*EOR* (((x0 purpose) = *defined*)(x2 == (x0 purpose)))
               (((x0 location) = *defined*)(x2 == (x0 location))))
         ((x2 prep root) <= "于")
        (x1 = x0)))

;;; vp 3

(<vp> --> (<vp> <pp>)
        (((x0 root) =c "限制")
         ((x0 passive) =c +)
        (*EOR* (((x0 condition) = *defined*)(x2 == (x0 condition)))
               (((x0 location) = *defined*)(x2 == (x0 location))))
         ((x2 prep root) <= "于")
        (x1 = x0)))

;;; vp 4

(<vp> --> (<pp> <vp>)
        (((x0 root) =c "设计")
         ((x0 passive) =c +)
         ((x0 purpose) = *defined*)
        (x1 == (x0 purpose))
         ((x1 prep root) <= "为")
        (x2 = x0)))

;;; vp 5

(<vp> --> (<pp> <vp>)
        (((x0 root) =c "制造")
         ((x0 passive) =c +)
        ((x0 agent) = *defined*)
        (x1 == (x0 agent))
         ((x1 prep root) <= "由")
        (x2 = x0)))

;;; vp 6
```

```
(<vp> --> (<pp> <vp> <pp>)
      ((*EOR* (((x0 subcat) =c v0))(((x0 root) =c "是")))
        ((x0 under) = *DEFINED*)
        ((x0 location) = *DEFINED*)
        (x1 == (x0 location))
        ((x1 prep root) <= "在")
        (x3 == (x0 under))
        ((x3 prep root) <= "到")
        ((x3 surf root) <= "下面")
        (x2 = x0)))

;;;------------------------------------------------------------;
;;; general rule for some verbs
;;; vp 7

(<vp> --> (<vp> <pp>)
      ((*EOR* (((x0 subcat) =c v0))(((x0 root) =c "是")))
        ((x0 goal) = *DEFINED*)
        (x2 == (x0 goal))
        ((x2 prep root) <= "到")
        (x1 = x0)))

;;;vp 8

(<vp> --> (<vp> <np>)     ;;; special rule for example 97
      (((x0 root) =c "沿着")
        ((x0 path) = *DEFINED*)
        (x2 == (x0 path))
        (x1 = x0)))

;;;vp 9

(<vp> --> (<vp> <pp>)
      ((*EOR* (((x0 subcat) =c v0))(((x0 root) =c "是")))
        ((x0 path) = *DEFINED*)
        (x2 == (x0 path))
        ((x2 prep root) <= "过")
        (x1 = x0)))

;;;vp 10

(<vp> --> (<pp> <vp>) ;;;exception of the following  rule
      (((x0 root) =c "受益")
        ((x0 location) = *DEFINED*)
        (x1 == (x0 location))
        ((x1 prep root) <= "在")
        (x2 = x0)))

;;;vp 11

(<vp> --> (<vp> <pp>)
      ((*EOR* (((x0 subcat) =c v0))(((x0 root) =c "是")))
```

```
        ((x0 location) = *DEFINED*)
        (x2 == (x0 location))
        ((x2 prep root) <= "在")
        (x1 = x0)))

;;;vp 12

(<vp> --> (<vp> <np>)
      ((*EOR* (((x0 subcat) =c v0))(((x0 root) =c "是")))
        ((x0 extent) = *DEFINED*)
        (x2 == (x0 extent))
        (x1 = x0)))

;;;vp 13

(<vp> --> (<vp> <np>)
      (((x0 root) =c "导致")
        ((x0 condition) = *DEFINED*)
        (x2 == (x0 condition))
        (x1 = x0)))

;;;vp 14

(<vp> --> (<vp> <pp>)
      (((x0 root) =c "暴露")
        ((x0 condition) = *DEFINED*)
        (x2 == (x0 condition))
        ((x2 prep root) <= "在")
        ((x2 prep surf) <= "中")
        (x1 = x0)))

;;;vp 15

(<vp> --> (<vp> <pp>)
      ((*EOR* (((x0 subcat) =c v0))(((x0 root) =c "是")))
        ((x0 purpose) = *DEFINED*)
        (x2 == (x0 purpose))
        ((x2 prep root) <= "有")
        (x1 = x0)))

;;;------------------------------------------------------------;
;;; RULE 8 for Simple declarative setence                      ;
;;;                                                            ;
;;; Declarative sentence(Transitive verb, active)
;;; Eample: You must use the correct type of fuel in order...

(<dec-act-s> --> (<np> <vp>)
              (((x0 mood) =c dec)
               ((x0 subcat) =c vt)
               ((x0 passive) = *UNDEFINED*)
               ((x0 agent) = *DEFINED*)
               (x1 == (x0 agent))
```

```
               (x2 = x0)))

;;;-----------------------------------------------------------;
;;; RULE 9 for Simple declarative setence                      ;

(<dec-act-s> --> (<np> <vp>)
              (((x0 mood) =c dec)
               ((x0 subcat) =c vt)
               ((x0 passive) = *UNDEFINED*)
               ((x0 experiencer) = *DEFINED*)
               (x1 == (x0 experiencer))
               (x2 = x0)))

;;;-----------------------------------------------------------;
;;; RULE 10 for Simple declarative setence                      ;

(<dec-act-s> --> (<np> <vp>)
              (((x0 mood) =c dec)
               ((x0 subcat) =c v0)
               ((x0 passive) = *UNDEFINED*)
               ((x0 benefactor) = *DEFINED*)
               (x1 == (x0 benefactor))
               (x2 = x0)))

;;;-----------------------------------------------------------;
;;; RULE 11 for Simple declarative setence                      ;

(<dec-act-s> --> (<np> <vp>)
              ((*EOR* (((x0 mood) =c dec))(((x0 mood) = *undefined*)))
              ;;;((x0 mood) =c dec) modified for 159
               (*EOR* (((x0 subcat) =c v0))
                      (((x0 root) =c "是")))
               ((x0 passive) = *UNDEFINED*)
               ((x0 theme) = *DEFINED*)
               (x1 == (x0 theme))
               (x2 = x0)))

;;;-----------------------------------------------------------;
;;; RULE 12 for Simple declarative setence                      ;

(<dec-act-s> --> (<np> <vp>)
              (((x0 mood) =c dec)
               ((x0 subcat) =c vt)
               ((x0 root) =c "有")
               ((x0 passive) = *UNDEFINED*)
               ((x0 possessor) = *DEFINED*)
               (x1 == (x0 possessor))
               (x2 = x0)))

;;;vp 16

(<vp> --> (<modal> <vp>)
```

```
       (((x0 modal) = *defined*)
         ((x0 modal-processed) = *undefined*)
         (x1 = x0)
        ((x0 modal-processed) = +)
        (x2 = x0)))

;;; vp 18

(<vp> --> (<pp> <vp>)
      (((x0 condition) = *DEFINED*)
        (x1 == (x0 condition))
        ((x1 prep root) <= "在")
        (x2 = x0)))

;;; vp 19

(<vp> --> (<pre-event> <comma> <vp>)
      (((x0 pre-event) = *DEFINED*)
        (x1 == (x0 pre-event))
        (x3 = x0)))

;;; vp 20

(<vp> --> (<dur-event> <vp>)
      (((x0 during-event) = *DEFINED*)
       ((x0 during-event cat) =c n)
        (x1 == (x0 during-event))
        (x2 = x0)))


;;;vp 21  moved here for example 123

(<vp> --> (<adv> <vp>)
        (((x0 event-contrast) = *defined*)
         (x1 == (x0 event-contrast))
         (x2 = x0)))

 ;;;vp 22

(<vp> --> (<pp> <vp>)
        (((x0 proportion) = *defined*)
         (x1 == (x0 proportion))
         ((x1 prep root) <= "以")
         (x2 = x0)))

(<dur-event> --> (<pp>)
      (((x0 cat) =c n)
       (x1 = x0)
       ((x1 prep surf) <= "中")
        ((x1 prep root) <= "在")))

;;;vp 23
```

```
(<vp> --> (<future-aux> <vp>)
       (((x0 tense-processed) = *undefined*)
        ((x0 tense) =c future)
        ((x0 tense-processed) = +)
        (x1 = x0)
        (x2 = x0)))

;;;vp 24

(<vp> --> (<progre-aux> <vp>)
       (((x0 prog-processed) = *undefined*)
        ((x0 prog) =c +)
        ((x0 prog-processed) = +)
        (x1 = x0)
         (x2 = x0)))

;;;vp 17

(<vp> --> (<negation> <vp>)
       (((x0 modal-processed) = *undefined*)
        ((x0 tense-processed) = *undefined*)
        ((x0 prog-processed) = *undefined*)
        ((x0 negation) = *defined*)
        ((x0 negation) = *remove*)
        (x1 = x0)
        (x2 = x0)))

;;;vp 25

(<vp> --> (<adv> <vp>)
        (((x0 event-frequency) = *defined*)
         (x1 == (x0 event-frequency))
         (x2 = x0)))

;;;vp 26

(<vp> --> (<pp> <vp>)
      (((x0 time-point) = *DEFINED*)
        (x1 == (x0 time-point))
        ((x1 prep root) <= "在")
        ((x1 surf root) <= "时")
        (x2 = x0)))

;;;vp 27

(<vp> --> (<pp> <vp>)
      (((x0 temp-point) = *DEFINED*)
        (x1 == (x0 temp-point))
        ((x1 prep root) <= "在")
;       ((x1 surf root) <= "时")
        (x2 = x0)))
```

```
;;;

;;; vp 28
(<vp> --> (<pp> <vp>)
      (((x0 location) = *DEFINED*)
        (x1 == (x0 location))
        ((x1 prep root) <= "在")
        (x2 = x0)))

;;;vp 29

(<vp> --> (<pp> <vp>)
      (((x0 source) = *DEFINED*)
        (x1 == (x0 source))
        ((x1 prep root) <= "从")
        (x2 = x0)))

;;; vp 30
(<vp> --> (<vp> <pp>)
      (((x0 destination) = *DEFINED*)
        (x2 == (x0 destination))
        ((x2 prep root) <= "在")
        (x1 = x0)))

;;;vp 31

(<vp> --> (<pp> <vp>)
        (((x0 instrument) = *defined*)
        (x1 == (x0 instrument))
        ((x1 prep root) = "用")
        (x2 = x0)))

;;;vp 32

(<vp> --> (<perfect-aux> <vp>)
     (((x0 perfect) =c +)
      ((x0 perfect) = *REMOVE*)
      (x2 = x0)))

;;;vp 33

(<vp> --> (<pp> <vp>)
     (((x0 passive) =c +)
      ((x0 agent) = *DEFINED*)
      (x1 == (x0 agent))
      ((x1 prep root) <= "被")
      (x2 = x0)))

;;;vp 34

(<vp> --> (<pp> <vp>)
     (((x0 subcat) =c vt)
```

```
      ((x0 passive) = *UNDEFINED*)
      (*EOR* (((x0 goal) = *DEFINED*)) (((x0 above) = *DEFINED*)))
      ((x0 patient) = *DEFINED*)
      (x1 == (x0 patient))
      ((x1 prep root) <= "把")
      (x2 = x0)))

;;;vp 35

(<vp> --> (<pp> <vp>)
     (((x0 subcat) =c vt)
      ((x0 passive) = *UNDEFINED*)
      (*EOR* (((x0 goal) = *DEFINED*)) (((x0 above) = *DEFINED*)))
      ((x0 theme) = *DEFINED*)
      (x1 == (x0 theme))
      ((x1 prep root) <= "把")
      (x2 = x0)))

;;; vp 36
(<vp> --> (<pp> <vp>)
     ((*EOR* (((x0 root) =c "最小化"))(((x0 root) =c "最优化")))
      ((x0 passive) = *UNDEFINED*)
      ((x0 theme) = *DEFINED*)
      (x1 == (x0 theme))
      ((x1 prep root) <= "把")
      (x2 = x0)))

;;;vp 37
;;; Following two rules are moved to here from rule 11.
;;; vp 11

(<vp> --> (<vp> <simp-s>)
      (((x0 event) = *DEFINED*)
        (x2 == (x0 event))
        (x1 = x0)))

;;;vp 38

(<vp> --> (<vp> <vp>)
      (((x0 event) = *DEFINED*)
        (x2 == (x0 event))
        (x1 = x0)))

;;;vp 39

(<vp> --> (<vp> <np>)  ;;; special rule for "proceed with", example 187.
      (((x0 root) =c "进行")
       ((x0 cofactor) = *defined*)
        (x2 == (x0 cofactor))
        (x1 = x0)))

;;;vp 40
```

```
(<vp> --> (<vp> <pp>)
        (((x0 predicate) = *defined*)
         (x2 == (x0 predicate))
         ((x2 prep root) <= "为")
         (x1 = x0)))

;;;vp 41

(<vp> --> (<pp> <vp>)
        (((x0 match) = *defined*)
         (x1 == (x0 match))
         ((x1 prep root) <= "为")
         (x2 = x0)))

;;;vp 42

(<vp> --> (<vp> <np>)
      (((x0 time-extent) = *DEFINED*)
        (x2 == (x0 time-extent))
        (x1 = x0)))

;;;vp 43

(<vp> --> (<vp> <pp>)
      (((x0 opposition) = *DEFINED*)
        (x2 == (x0 opposition))
        ((x2 prep root) <= "免受")
        ((x2 surf root) <= "的危害")
        (x1 = x0)))

;;;vp 44

(<vp> --> (<vp> <pp>)
     (((x0 subcat) =c vt)
      ((x0 as-object) = *DEFINED*)
      (x2 == (x0 as-object))
      ((x2 prep root) <= "作为")
      (x1 = x0)))

;;;vp 45

(<vp> --> (<vp> <pp>)
     (((x0 subcat) =c vt)
      ((x0 goal) = *DEFINED*)
      (x2 == (x0 goal))
      ((x2 prep root) <= "到")
      (x1 = x0)))

(<vp> --> (<vp> <pp>)
     (((x0 subcat) =c vt)
      ((x0 above) = *DEFINED*)
      (x2 == (x0 above))
```

```
      ((x2 prep root) <= "到高于")
      (x1 = x0)))

;;;vp 46

(<vp> --> (<vp> <pp>)
     (;((x0 subcat) =c vt)
      ((x0 location) = *DEFINED*)
      (x2 == (x0 location))
      ((x2 prep root) <= "在")
      (x1 = x0)))

;;;vp 47

(<vp> --> (<vp> <np>)
     (((x0 subcat) =c vt)
      ((x0 passive) = *UNDEFINED*)
      ((x0 patient) = *DEFINED*)
      (x2 == (x0 patient))
      (x1 = x0)))

;;;vp 48

(<vp> --> (<vp> <np>)
     (((x0 subcat) = vt)
      ((x0 passive) = *UNDEFINED*)
      ((x0 theme) = *DEFINED*)
      (x2 == (x0 theme))
      (x1 = x0)))

;;; vp 49

(<vp> --> (<vp> <np>)
     (((x0 subcat) = vt)
      ((x0 passive) = *UNDEFINED*)
      ((x0 object) = *DEFINED*)
      (x2 == (x0 object))
      (x1 = x0)))

;;;vp 50

;(<vp> --> (<v1>)
;         ((*EOR* (((x0 subcat) =c v0))
;                 (((x0 subcat) = *undefined*)))
;          (x1 = x0)))

;;;vp 51

(<vp> --> (<v1>)
         ((x1 = x0)))

;;;-------------------------------------------------------------;
```

```
;;; Rule 1 for VP (very special case)
;
;;;                                                                          ;
;;; Heading NP                                                               ;
;;; vp 51

(<vp> --> (<np>)
     (((x0 cat) =c n)
      (x1 = x0)))

;;;=====================================================
;;;
;;;

(<v1> --> (<pp-coordination> <v1>)
      (((x0 manner) = *defined*)
       ((x0 manner cat) =c *pp-coordination)
        (x1 == (x0 manner))
        (x2 = x0)))

(<v1> --> (<advp> <v1>)
      (((x0 manner) = *defined*)
        ((x0 intensifier) = *defined*)
        (x1 == (x0 manner))
        ((x1 intensifier) == (x0 intensifier))
        (x2 = x0)))

(<v1> --> (<advp> <v1>)
      (((x0 manner) = *defined*)
        (x1 == (x0 manner))
        (x2 = x0)))

(<v1> --> (<advp> <v1>)
      (((x0 intensifier) = *defined*)
        (x1 == (x0 intensifier))
        (x2 = x0)))

(<v1> --> (<v> <perfect-surf>)
      (((x0 perfect) =c +)
        (x1 = x0)))

(<v1> -->(<v>)
         (((x0 cat) =c v)
          (x1 = x0)))

(<v> --> (%)
         (((x0 cat) =c v)
          ((x1 value) = (x0 root))))


(<pre-event> --> (<pp>)
               (((x0 prep root) <= "在")
                ((x0 surf root) <= "之前")
```

```
              (x1 = x0)))

(<progre-aux> --> (%)
        (((x0 negation) = *defined*)
        ((x1 value) <= "不在")))

(<progre-aux> --> (%)
        (((x1 value) <= "正在")))

(<future-aux> --> (%)
       (((x0 negation) = *defined*)
        ((x0 root) = "有")
        ((x1 value) <= "将没")))

(<future-aux> --> (%)
       (((x0 negation) = *defined*)
        ((x1 value) <= "将不")))

(<future-aux> --> (%)
       (((x1 value) <= "将")))

(<negation> --> (%)
       (((x0 tense) =c past)
        ((x1 value) <= "没")))

(<negation> --> (%)
       (((x1 value) <= "不")))

(<perfect-aux> --> (%)
               (((x1 value) <= "已经")))

(<perfect-surf> --> (%)
                (((x1 value) <= "了")))

;;;-----------------------------------------------------------;
;;;                                                           ;
;;; RULE 1 for Simple passive setence                         ;
;;;                                                           ;
;;;                                                           ;
;;; Declarative sentence(Transitive verb, passive)
;;; Eample: The use of fuel heaters should be limited to cold
;;;         weather condition.
;;;
;;;

(<dec-pas-s> --> (<np> <vp>)
             (((x0 mood) =c dec)
              ((x0 passive) =c +)
              ((x0 subcat) =c vt)
              ((x0 theme) = *DEFINED*)
              (x1 == (x0 theme))
              (x2 = x0)))

;;;-----------------------------------------------------------;
;;;                                                           ;
;;; RULE 2 for Simple passive setence                         ;
```

```
 (<dec-pas-s> --> (<np> <vp>)
              (((x0 mood) =c dec)
               ((x0 passive) =c +)
               ((x0 subcat) =c vt)
               ((x0 patient) = *DEFINED*)
               (x1 == (x0 patient))
               (x2 = x0)))

 (<advp> --> (<adv> <adv>)
      (((x0 intensifier) = *defined*)
        (x1 == (x0 intensifier))
        (x2 = x0)))

 (<advp> --> (<adv>)
      (((x0 intensifier) = *undefined*)
        (x1 = x0)))

;;;============================================================;;;
;;;                          MODAL                             ;;;
;;;============================================================;;;
;;;------------------------------------------------------------;
;;; Modal 1
;;;
;;; Must(bishu)

(<modal> --> (%)
         (((x0 modal) =c necessity)
         ((x0 compulsion) =c +)
         ((x0 negation) =c +)
         ((x1 value) <= "不必")))

;;;------------------------------------------------------------;
;;; Modal 2
;;;
;;; Must not(bubi)

(<modal> --> (%)
         (((x0 modal) =c necessity)
         ((x0 compulsion) = *defined*)
         ((x1 value) <= "必须")))

;;;------------------------------------------------------------;
;;; Modal 3
;;;
;;; should(yingdang)

(<modal> --> (%)
         (((x0 modal) =c necessity)
         ((x0 obligation) =c +)
         ((x0 negation) =c +)
         ((x1  value) <= "不应当")))
```

```
;;;-------------------------------------------------------------;
;;; Modal 4
;;;
;;; should not(buyingdang)

(<modal> --> (%)
         (((x0 modal) =c necessity)
         ((x0 obligation) =c +)
         ((x1 value) <= "应当")))

;;;-------------------------------------------------------------;
;;; Modal 5
;;;
;;; Could(hui)

(<modal> --> (%)
         (((x0 modal) =c possibility)
         ((x0 tentative) =c +)
         ((x0 negation) =c +)
         ((x1 value) <= "不会")))

;;;-------------------------------------------------------------;
;;; Modal 6
;;;
;;; Could not(buhui)

(<modal> --> (%)
         (((x0 modal) =c possibility)
         ((x0 tentative) =c +)
         ((x1 value) <= "会")))

;;;-------------------------------------------------------------;
;;; Modal 7
;;;
;;; may(keneng)

(<modal> --> (%)
         (((x0 modal) =c possibility)
         ((x0 negation) =c +)
         ((x1 value) <= "不会")))

;;;-------------------------------------------------------------;
;;; Modal 8
;;;
;;; Can not(buhui)

(<modal> --> (%)
         (((x0 modal) =c possibility)
         ((x1 value) <= "可能")))

;;;-------------------------------------------------------------;
;;; Modal 9
```

```
;;;
;;; Can(keneng)

(<modal> --> (%)
         (((x0 modal) =c ability)
         ((x0 negation) =c +)
         ((x1 value) <= "不能")))

;;;-------------------------------------------------------------;
;;; Modal 10
;;;
;;; Can not(buneng)

(<modal> --> (%)
         (((x0 modal) =c ability)
         ((x1 value) <= "能")))

(<modal> --> (%)
         (((x0 negation) =c +)
         ((x0 tense) =c past)
         ((x1 value) <= "没有")))

(<modal> --> (%)
         (((x0 negation) =c +)
          ((x0 mood) =c imp)
         ((x1 value) <= "不要")))
```