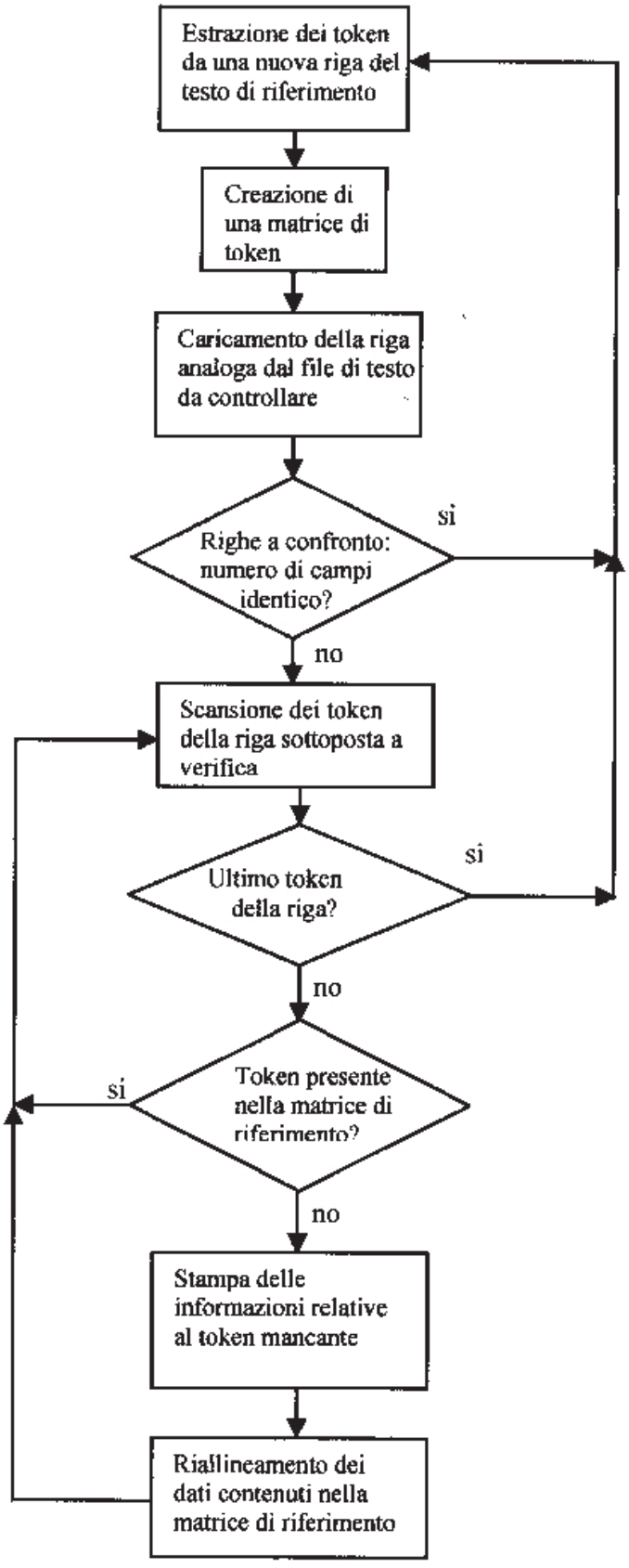
viduate, al fine di garantire al sistema il proseguimento regolare dell'attività di controllo fino al termine della riga.



Tav. 72: Diagramma di flusso di `checkline`, script in GAWK.

7.5.1.6 IL "TRANSORD". [MT] Come accennato a inizio paragrafo, oltre agli importantissimi sistemi di controllo, nel corso dell'opera si è rivelato necessario possedere un livello

di conoscenza più dettagliato della distribuzione delle informazioni di POS e lemma associate ad ogni token. L'idea era quella di sviluppare un sistema che permettesse all'utente di esaminare una sorta di formario esteso in cui ad ogni token non fossero unicamente associate le voci di lemma e numero di occorrenze, caratteristica tipica del formato dell'effeliscio (es. 24b), bensì fosse visibile anche la catena di transcategorizzazioni originale tratta dall'f+2 (es. 24a) disposta in ordine crescente sulla base del valore di POS selezionato dal disambiguatore (es. 24c). L'obiettivo del programma, denominato `transord` (transcategorizzazioni ordinate), consisteva pertanto nel realizzare un testo caratterizzato da una sapiente mescolanza di caratteristiche differenti, univocamente possedute dai file f e f+2.

Per maggiore chiarezza, si invita il lettore a confrontare il medesimo campione di materiali estratto rispettivamente dal formario di origine f+2 (es. 24a), dall'effeliscio (es. 24b) e dall'uscita del `transord` (es. 24c):

[24a]
```
lla     117   (lem=la,60,0,5,6,0,0);(lem=la,39,3,5,6,0,0)
[...]
lle     60    (lem=la,60,0,5,7,0,0);(lem=la,39,3,5,7,0,0);
[...]         (lem=gli,39,3,5,6,0,0)
lli     84    (lem=lo,60,0,4,7,0,0);(lem=lo,39,3,4,7,0,0);
[...]         (lem=gli,39,3,4,6,0,0)
llo     66    (lem=lo,60,0,4,6,0,0);(lem=lo,39,3,4,6,0,0)     effedue,
```

[24b]
```
lla     18    lem=la,39,3,5,6,0,0
lla     99    lem=la,60,0,5,6,0,0
[...]
lle     4     lem=gli,39,3,5,6,0,0
lle     12    lem=la,39,3,5,7,0,0
lle     44    lem=la,60,0,5,7,0,0
[...]
lli     15    lem=lo,39,3,4,7,0,0
lli     29    lem=lo,60,0,4,7,0,0
lli     41    lem=gli,39,3,4,6,0,0
[...]
llo     30    lem=lo,60,0,4,6,0,0
llo     36    lem=lo,39,3,4,6,0,0                              effeliscio,
```

[24c]
```
lla     18    (lem=la,39,3,5,6,0,0);(lem=la,60,0,5,6,0,0)
lle     4     (lem=gli,39,3,5,6,0,0);(lem=la,60,0,5,7,0,0);
              (lem=la,39,3,5,7,0,0)
lle     12    (lem=la,39,3,5,7,0,0);(lem=la,60,0,5,7,0,0);
              (lem=gli,39,3,5,6,0,0)
lli     15    (lem=lo,39,3,4,7,0,0);(lem=lo,60,0,4,7,0,0);
              (lem=gli,39,3,4,6,0,0)
lli     41    (lem=gli,39,3,4,6,0,0);(lem=lo,60,0,4,7,0,0);
              (lem=lo,39,3,4,7,0,0)
llo     36    (lem=lo,39,3,4,6,0,0);(lem=lo,60,0,4,6,0,0)
lla     99    (lem=la,60,0,5,6,0,0);(lem=la,39,3,5,6,0,0)
lle     44    (lem=la,60,0,5,7,0,0);(lem=la,39,3,5,7,0,0);
              (lem=gli,39,3,5,6,0,0)
lli     29    (lem=lo,60,0,4,7,0,0);(lem=lo,39,3,4,7,0,0);
              (lem=gli,39,3,4,6,0,0)
llo     30    (lem=lo,60,0,4,6,0,0);(lem=lo,39,3,4,6,0,0)
```

uscita del `transord`.

Nonostante la semplicità concettuale dell'idea di base, l'implementazione computazio-

nale del sistema ha presentato una difficoltà di sviluppo non indifferente, richiedendo per il suo corretto funzionamento la creazione di tre moduli distinti, azionati secondo una sequenza logica ben precisa da un comando batch di DOS. Le ragioni di tale complessità operativa sono da ricercarsi nel fatto che al fine di ottenere un testo in uscita con le caratteristiche dell'esempio di cui sopra, era necessario disporre dei dati presenti nell'effeliscio estratto dal testo disambiguato, mescolandoli opportunamente con quelli presenti nel formario generale effedue.

Proprio il primo di questi moduli, che come risulta facilmente intuibile dal listato riportato poco sotto presenta il livello di complessità maggiore, ha il compito di estrarre i dati da entrambi i file e operare tutte la serie di modifiche necessarie al fine di ricostruire opportunamente la catena di transcategorizzazioni, inserendo in testa la POS selezionata dal disambiguatore. In questo modo è possibile facilmente definire l'assetto orizzontale del testo, mentre l'assestamento verticale sarà responsabilità dei due moduli successivi.

```
BEGIN {
while ((getline < "f2") > 0)
          {
          indice = $1
          lista[indice] = $NF
          }
}
{
pos = $NF
if (pos ~ /,3[0123456789],/ || pos ~ /,4[01567],/ || pos ~ /,5[01],/
|| pos ~ /,5[67],/ || pos ~ /,6[01],/)
     {
     pys = split (pos, ghy, /,/)
     gok = ghy[1]ghy[2]
     linea = lista[$1]
     if (linea ~ /\);\(/)
          {
          cpn = 1
          sub (/\(/, "", linea)
          cp = split (linea, sp, /\);\(/)
          pt = ""
          while (cpn <= cp)
               {
               cys = split (sp[cpn], cxn, /,/)
               gtr = cxn[1]cxn[2]
               if (cpn > cp)
                    break
               if (gtr != gok)
                    {
                    if (cpn == 1)
                         {
                         pt = pt "(" sp[cpn] ");"
                         cpn++
                         }
                    else
                    if (cpn > 1 && cpn < cp)
```

```
                    cpn++
                    }
               else
               if (cpn == cp)
                    {
                    pt = pt "(" sp[cpn]
                    cpn++
                    }
               }
          else
               cpn++
          }
     printf "%-25s %5d %s\n", $1, $2, "(" $3 ");" pt
     }
else
     print $0
}
}
```

Tav. 73: Listato di `transord`, modulo 1, script in GAWK.

La struttura di testo generato dal primo modulo di programma corrisponde all'esempio seguente:

```
[25]  lla      18   (lem=la,39,3,5,6,0,0);(lem=la,60,0,5,6,0,0)
      lla      99   (lem=la,60,0,5,6,0,0);(lem=la,39,3,5,6,0,0)
      lle      4    (lem=gli,39,3,5,6,0,0);(lem=la,60,0,5,7,0,0);
                    (lem=la,39,3,5,7,0,0)
      lle      12   (lem=la,39,3,5,7,0,0);(lem=la,60,0,5,7,0,0);
                    (lem=gli,39,3,5,6,0,0)
      lle      44   (lem=la,60,0,5,7,0,0);(lem=la,39,3,5,7,0,0);
                    (lem=gli,39,3,5,6,0,0)
      llei     13   lem=lei,38,3,5,6,0,0
      lli      15   (lem=lo,39,3,4,7,0,0);(lem=lo,60,0,4,7,0,0);
                    (lem=gli,39,3,4,6,0,0)
      lli      29   (lem=lo,60,0,4,7,0,0);(lem=lo,39,3,4,7,0,0);
                    (lem=gli,39,3,4,6,0,0)
      lli      41   (lem=gli,39,3,4,6,0,0);(lem=lo,60,0,4,7,0,0);
                    (lem=lo,39,3,4,7,0,0)
      llo      30   (lem=lo,60,0,4,6,0,0);(lem=lo,39,3,4,6,0,0)
      llo      36   (lem=lo,39,3,4,6,0,0);(lem=lo,60,0,4,6,0,0)
```

uscita del modulo 1 di `transord`.

Da questo esempio si può facilmente notare come nonostante il testo di uscita del primo script si avvicini molto alla forma definitiva voluta, rimangono ancora da affrontare una serie di problematiche legate alle righe, che devono essere disposte in ordine crescente sulla base della POS iniziale. Per ottenere tale risultato non è purtroppo possibile ricorrere a soluzioni semplici come il comando `sort` del DOS od agli analoghi nel mondo Unix, in quanto all'interno della riga la posizione del codice numerico della POS non è mai costante, bensì è funzione della lunghezza del lemma. Pertanto bisogna aggirare l'ostacolo mediante l'uso di un secondo modulo che, sfruttando la duplicazione a inizio riga dei dati di POS, seguita dal comando di `sort`, permette di ottenere il perfetto ordinamento crescente delle informazioni associate al lemma.

Mostriamo ora il listato del secondo modulo, seguito da un frammento di testo di uscita:

```
{
sub (/;$/, "", $NF)
cp = split ($3, sp, /,/)
pos = sp[2]
fld1 = pos " " $1
printf "%-20s %5d %s\n", fld1, $2, $NF
}
```

Tav. 74: Listato di `transord`, modulo 2, script in GAWK.

```
[26]    39      lla   18      (lem=la,39,3,5,6,0,0);(lem=la,60,0,5,6,0,0)
        39      lle   4       (lem=gli,39,3,5,6,0,0);(lem=la,60,0,5,7,0,0);
                              (lem=la,39,3,5,7,0,0)
        39      lle   12      (lem=la,39,3,5,7,0,0);(lem=la,60,0,5,7,0,0);
                              (lem=gli,39,3,5,6,0,0)
        39      lli   15      (lem=lo,39,3,4,7,0,0);(lem=lo,60,0,4,7,0,0);
                              (lem=gli,39,3,4,6,0,0)
        39      lli   41      (lem=gli,39,3,4,6,0,0);(lem=lo,60,0,4,7,0,0);
                              (lem=lo,39,3,4,7,0,0)
        39      llo   36      (lem=lo,39,3,4,6,0,0);(lem=lo,60,0,4,6,0,0)
                              [...]
        60      lla   99      (lem=la,60,0,5,6,0,0);(lem=la,39,3,5,6,0,0)
        60      lle   44      (lem=la,60,0,5,7,0,0);(lem=la,39,3,5,7,0,0);
                              (lem=gli,39,3,5,6,0,0)
        60      lli   29      (lem=lo,60,0,4,7,0,0);(lem=lo,39,3,4,7,0,0);
                              (lem=gli,39,3,4,6,0,0)
        60      llo   30      (lem=lo,60,0,4,6,0,0);(lem=lo,39,3,4,6,0,0)
```

uscita del modulo 2 di `transord`.

Sulla base dell'uscita del secondo modulo, non resta altro che produrre il documento definitivo. Il che possibile realizzare facilmente mediante un comando, contenuto nel terzo modulo, che si limiti a stampare la riga intera opportunamente formattata, escluso il primo campo numerico:

```
printf "%-20s %5d %s\n", $2, $3, $NF
```

Tav. 75: Listato di `transord`, modulo 3, script in GAWK.

7.5.1.7 L'ESTRATTORE DI LEMMARIO DEFINITIVO. [MT] Secondo quanto già accennato nel § 7.5.1.6, relativo al sistema `transord`, la conoscenza approfondita della distribuzione statistica dei dati all'interno del testo disambiguato gioca un ruolo chiave in seno al progetto. Oltre ai dati generati dal modulo citato poc'anzi, in effetti, si è rivelato necessario produrre una nuova struttura (disponibile a partire dalla Ver. 1.8 del corpus), questa volta incentrata sul lemma, che risultasse più articolata di quanto proposto dal vecchio `extraL` (cfr. § 7.5.1.3 numero 4). A differenza del semplice lemmario ottenibile da `extraL`, infatti, il nuovo lemmario generato da questa procedura è una struttura complessa, che associa ad ogni lemma altri tre elementi: il numero di type associati al lemma, i token complessivi di tutti i type del lemma, e, per ultimo, la POS (o meglio: il *type* della POS[12], ossia il secondo *branching* di ogni gerarchia) espressa in notazione estesa ridotta.

[12] Questo consente di sdoppiare in due lemmi distinti quello che in effetti va distinto (ad es. nomi comuni da nomi propri propri, indefiniti da dimostrativi, ecc.) ma di collassare in un unico lemma tutte le forme verbali di un paradigma.