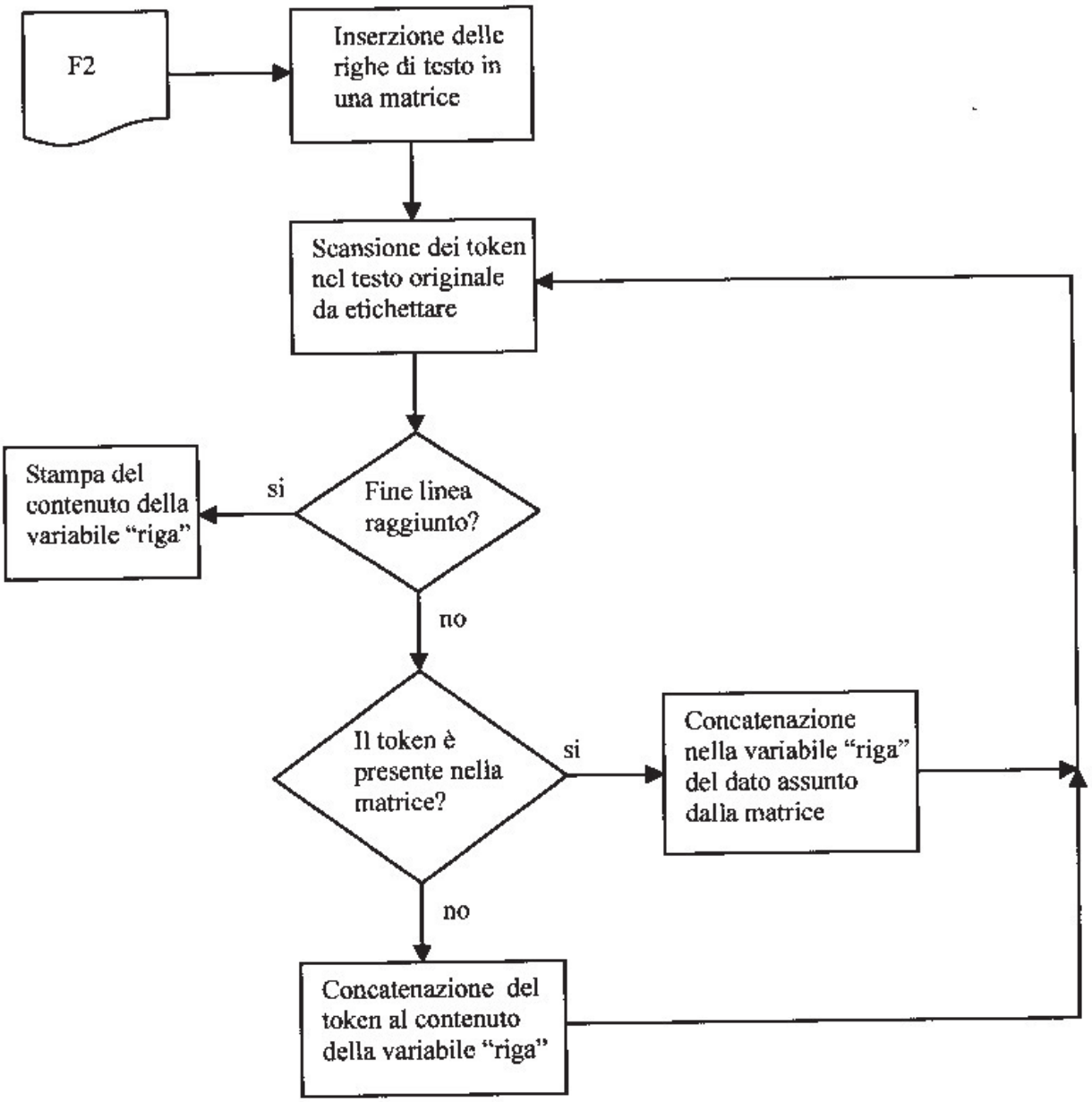
presenza all'interno del testo di informazioni metatestuali (markup: cfr. ¶ 8) che non richiedono alcun tipo di etichettatura morfosintattica, all'interno del ciclo `while` il sistema si occupa anche di eseguire un controllo su ciascun token, lasciandolo invariato nel caso in cui all'interno del formario non si incontri un type corrispondente.

Per meglio chiarire quanto appena descritto, oltre al listato del programma mostrato sopra (Tav. 66), mostriamo anche il corrispondente diagramma di flusso (Tav. 67):



Tav. 67: Diagramma di flusso di `spalmaF2`, script in GAWK.

7.5.1.2 L'ESTRATTORE. [MT] Esaminiamo ora il programma corrispondente di estrazione, che prende il nome di `estraF`.

Lo script in questione nasce con il ben preciso compito di estrarre le forme dal testo spalmato. Sebbene fosse originariamente utilizzato come semplice sistema di controllo del processo di spalmatura, la sua esistenza si è rivelata di primaria importanza solo in una fase successiva. Lo script in questione, infatti, riveste un ruolo chiave all'interno delle varie fasi di costruzione del corpus, in particolare per quanto concerne l'aspetto della verifica della

correttezza delle operazioni di disambiguazione. È possibile attuare questo meccanismo di controllo indiretto del testo (definibile tale in quanto la procedura non prevede la diretta visualizzazione su schermo del testo disambiguato), mediante il confronto dei dati di uscita prodotti dal programma di estrazione con quanto previsto dalle regole di disambiguazione.

Anche qui, come in altri casi, il programma si presenta compatto e leggero:

```
{
nf = 0
while(nf < NF)
        {
        nf++
        if($nf ~ /\#/ || $nf ~ /\@/ || $nf ~ /\%/ || $nf ~ /\$/ || $nf ~ /\£/)
                continue
        # omette gli elementi di markup
        #
        tabella[$nf]++
        }
}
END {
for(word in tabella)
        {
        z = split(word,fiel,"_")
        printf "%-25s %5d %s\n", fiel[1], tabella[word], fiel[2]
        }
}
```

Tav. 68: Listato di `estraF`, script in GAWK.

Seppur caratterizzato da un numero relativamente ridotto di linee di codice, il programma possiede una potenza computazionale non da poco. In primo luogo, è possibile scorgere dal listato la netta divisione tra due differenti blocchi operativi: il primo, relativamente semplice e introdotto dal comando di iterazione condizionata `while`, si limita a saltare all'interno del testo disambiguato tutti i codici di markup, individuando in tal modo unicamente le forme dotate di POS e costruendo con queste una matrice numericamente indicizzata da un valore progressivo assegnato automaticamente dal programma stesso (`tabella[$nf]++`). L'esecuzione del blocco in questione, pertanto, ha termine con la completa lettura di tutte le righe costituenti il testo oggetto di elaborazione.

Il secondo blocco, invece, che entra in azione una volta concluso il precedente, esamina sequenzialmente tutti i dati contenuti nella matrice di cui sopra, li divide estraendone le componenti rilevanti e li stampa attenendosi a una formattazione spaziale ben precisa definita dall'utente.

È proprio la possibilità di realizzare un'associazione automatica dei dati testuali con un indice numerico che permette all'operatore di ottenere agevolmente tutte le informazioni statistiche relative alle frequenze delle varie forme all'interno del testo disambiguato.

7.5.1.3 GLI ALTRI SCRIPT. Accanto ai fondamentali spalmatore (`spalmaF2`, cfr. § 7.5.1.1) ed estrattore (`estraF`, cfr. § 7.5.1.3), nella fase iniziale del lavoro si era resa necessaria anche la programmazione di altre procedure ausiliarie od alternative. Così altri moduli GAWK[11], o versioni alternative dei precedenti finalizzate a scopi particolari, sono stati preparati (dal medesimo *team* specificato nel § 7.5.1 a proposito dello spalmatore) per

[11] Sempre tutti liberamente disponibili sul sito `http://www.bmanuel.org/tools` sotto licenza GNU.