7.5.1 IL FRONTE INFORMATICO E LA "PROCEDURA-CT". Con il disegno e la creazione degli spalmatori da un lato e degli estrattori dall'altro, il cuore della "procedura-CT", ossia di quella serie (*recte* batch) di operazioni informatiche che mettono capo alla creazione di un pre-corpus pronto ad essere codificato come corpus pienamente funzionante sotto CWB, è ormai abbozzato.

Ad ogni modifica dell'effedue, o del testo di base, lo spalmatore applicherà le informazioni del primo sul secondo, generando un nuovo testo taggato (ossia il pre-corpus), dal quale gli estrattori ricaveranno i nuovi formari e lemmari, e le script di *encoding* del CWB genereranno un nuovo corpus interrogabile. Qualsiasi tipo di intervento (correzione, modifica, ampliamento, ecc.), pertanto, nella strategia che abbiamo progettato, non viene mai compiuto direttamente sul corpus, ma sempre e solo sui due file che lo genèrano: il testo markuppato e tokenizzato, e l'effedue con i tag di ogni type. Questo in un ottica di ottimizzazione dei lavori, non dovendo così ogni operatore avere sempre a disposizione la versione codificata del corpus.

Nelle fasi di lavorazione seguenti, nuovi moduli verranno aggiunti alla procedure (il più importante è quello della disambiguazione, cfr. ¶ 9), ma questa caratteristica è sempre stata mantenuta inalterata.

7.5.1.0 IL FRONTE INFORMATICO: SOMMARIO. Nei paragrafi seguenti inizieremo descrivendo lo spalmatore (cfr. § 7.5.1.1), poi l'estrattore (cfr. § 7.5.1.2), accennando agli altri script secondari (e versioni diverse di spalmatore ed estrattore) usate nella prima fase del lavoro (cfr. § 7.5.1.3). Poi verranno presentati[9] i moduli di maggiore rilevanza pratica introdotti nel lungo periodo di sviluppo del sistema, `checksum` (cfr. § 7.5.1.4), `checkline` (cfr. § 7.5.1.5) e `transord` (cfr. § 7.5.1.6). Ed infine verrà presentata la nuova procedura di estrazione del lemmario (cfr. § 7.5.1.7), in realtà creata, perché resasi ormai necessaria, solo a partire dalla Ver. 1.8 del corpus.

Tali programmi, tutti scritti in GAWK, possono in generale essere distinti in due categorie principali: le procedure per la verifica e il controllo dell'integrità strutturale del corpus durante le varie fasi di elaborazione e le procedure per l'estrazione delle statistiche testuali più rilevanti.

7.5.1.1 LO SPALMATORE. [MT] Il *tool* che materialmente "spalma" le informazioni contenute nell'effedue sul testo, ormai tokenizzato e markuppato (con le modalità per cui cfr. il ¶ 8), è stato chiamato, appunto, `spalmaF2`, e produce come risultato un testo annotato (in cui ossia i tag apposti ai type nel formario sono stabilmente congiunti ai token del testo) modellato come quello presentato nella simulazione del § 7.2.5.2. Si tratta di uno script GAWK elaborato da Cesare Oitana (programmazione), Manuel Barbera (analisi e consulenza linguistica) e Marco Tomatis (revisioni applicative) appositamente per questo progetto, e posto in libera distribuzione sul sito `www.bmanuel.org`.

Il principio di funzionamento del programma è piuttosto semplice: dato un formario opportunamente organizzato (l'effedue), il sistema di etichettatura si limita ad attribuire tutte le categorie grammaticali associate a una data forma (type) a tutti i token corrispondenti presenti nel testo. Da ciò, tra l'altro, ne consegue che all'interno del corpus le varie forme potranno ricevere sia categorie univoche, sia transcategorizzazioni, in funzione dei dati presenti sul formario di riferimento. Anche l'architettura dello script non è, in effetti, molto complessa. Il programma si compone sostanzialmente di due parti.

[9] Ci si perdonerà in questa occasione l'abbandono dello stretto ordine cronologico cui in questo capitolo ci siamo in genere attenuti per presentare le procedure nella loro interezza.

La prima parte è caratterizzata dal comando `begin`, che indica all'interprete GAWK che il blocco di istruzioni immediatamente successivo, racchiuso tra parentesi graffe, deve essere eseguito soltanto una volta all'avvio del programma stesso. Tale blocco di comandi agisce sul formario, assolvendo principalmente a tre funzioni[10]: gestione dell'apertura e dell'acquisizione dei dati dal formario, eliminazione di eventuali caratteri di spazio presenti ad inizio riga e, infine, creazione di una matrice (array) contenente le informazioni morfosintattiche da "spalmare" sul corpus, indicizzato dalle rispettive forme: in pratica si assiste alla trasformazione del formario stesso in un array di tipo associativo.

```
BEGIN {
while((getline < "f2") > 0)
        {
        indic = $1
        $1 = ""
        $2 = ""
        sub (/^ +/,"",$0)
        tabel[indic] = $0
        }
close("f2")
}
{
riga = ""
nf = 0
while(nf < NF)
        {
        nf++
        if(($nf in tabel) == 0)
            {
            riga = riga " " $nf
            continue
            }
        riga = riga " " $nf"_"tabel[$nf]
        }
print riga
}
```

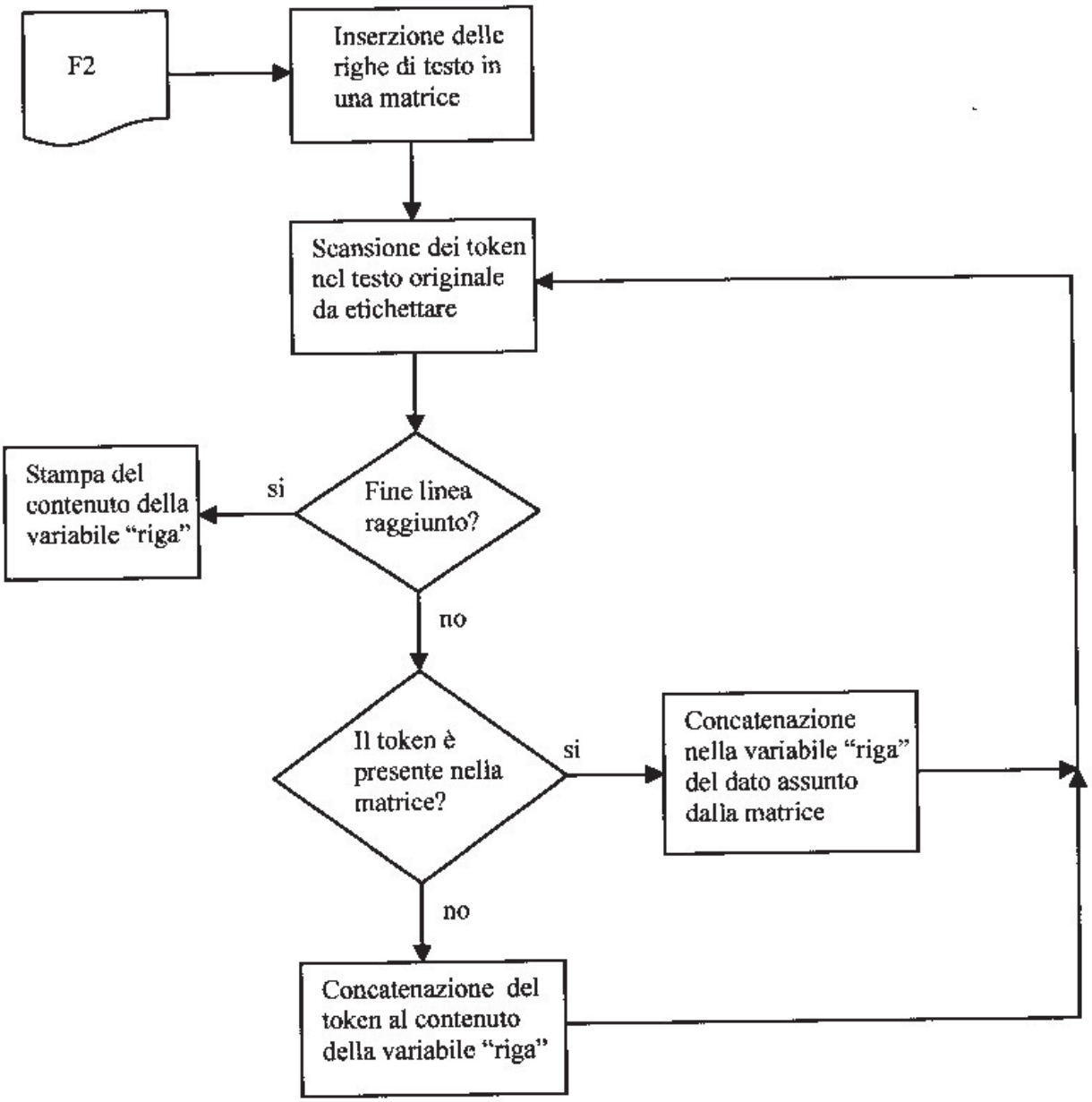
Tav. 66: Listato di spalmaF2, script in GAWK.

La seconda parte del programma si occupa dell'operazione di spalmatura vera e propria. Anche in questo caso il principio di funzionamento è piuttosto semplice: sfruttando la caratteristica dell'interprete di dividere automaticamente ogni linea del corpus in campi, tenendo traccia della loro quantità all'interno della variabile di sistema NF (*Number of Field*), il motore di spalmatura utilizza il ciclo impostato dal comando `while` al fine di rigenerare efficacemente ogni linea di testo, prelevando i dati morfosintattici direttamente dall'array precedentemente costituito e impiegando i vari token presenti sulla riga stessa come chiavi di ricerca. Naturalmente, per far ciò, il sistema non introduce le dovute modifiche direttamente sul testo originale, bensì immagazzina con logica sequenziale tutti i dati rilevanti in una determinata variabile, stampandoli solo successivamente, al termine del ciclo, una volta superato l'ultimo elemento della riga. Naturalmente va forse anche precisato che, data la

[10] Queste operazioni preparatorie sono di vitale importanza per il funzionamento del sistema, ma chiaramente debbono essere effettuate una volta sola; ecco quindi il motivo per l'adozione del comando `begin`.

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